Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
τομεας σηματων, ελεγχου και ρομποτικης

Αναγνώριση ανθρωπίνων δράσεων και συμπεριφορών με αξιοποίηση δεδομένων φορητών αισθητήρων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Αλεξάνδρας Βιώνη

Επιβλέπων: Πέτρος Μαραγκός
Καθηγητής Ε.Μ.Π.

Συν-επιβλέπουσα: Δρ. Αθανασία Ζλατίντση
Μεταδιδακτορική Ερευνήτρια Ε.Μ.Π.

ΕΡΓΑΣΤΗΡΙΟ ΟΡΑΣΗΣ ΥΠΟΛΟΓΙΣΤΩΝ, ΕΠΙΚΟΙΝΩΝΙΑΣ ΛΟΓΟΥ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

Αθήνα, Οκτώβριος 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Σημάτων, Ελέγχου και Ρομποτικής
Εργαστήριο Όρασης Υπολογιστών, Επικοινωνίας Λόγου και Επεξεργασίας Σημάτων

Αναγνώριση ανθρωπίνων δράσεων και συμπεριφορών με αξιοποίηση δεδομένων φορητών αισθητήρων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Αλεξάνδρας Βιώνη

Επιβλέπων: Πέτρος Μαραγκός
Καθηγητής Ε.Μ.Π.

Συν-επιβλέπουσα: Δρ. Αθανασία Ζλατίντση
Μεταδιδακτορική Ερευνήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 20η Οκτωβρίου, 2023.

.....	.....	.....
Πέτρος Μαραγκός	Αλέξανδρος Ποταμιάνος	Αθανάσιος Ροντογιάννης
Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2023

.....

Αλεξανδρα Βιωνη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Στη γιαγιά μου, Μόσχα, που πίστεψε σ' εμένα πριν απ' όλους.
Στη Βάλια Κορόζη, που στάθηκε δίπλα μας ως την ύστατη ώρα.
Στο νονό μου, Γιάννη Παππά, που με δίδαξε πολλά.

Περίληψη

Η ανάπτυξη συστημάτων ικανών να ανιχνεύουν και να αναγνωρίζουν αυτόματα τις δραστηριότητες που εκτελεί ένα άτομο, καθώς και το πλαίσιο στο οποίο εκτελείται κάθε δραστηριότητα, είναι η επιτομή της Αναγνώρισης Ανθρωπίνων Δράσεων (HAR). Τα συστήματα αυτά μπορούν να χρησιμοποιούν δεδομένα που συλλέγονται από ένα φάσμα αισθητήρων, στο οποίο συμπεριλαμβάνονται οπτικοί αισθητήρες, π.χ. βιντεοκάμερες, και μη οπτικοί αισθητήρες, όπως οι φορητοί αισθητήρες και οι αισθητήρες περιβάλλοντος. Η αναγνώριση ανθρωπίνων δράσεων βρίσκει αναρίθμητες εφαρμογές στην καθημερινή ζωή, κυρίως στον τομέα της υγείας, στη φροντίδα ηλικιωμένων, στην υποβοηθούμενη διαβίωση, στην αλληλεπίδραση ανθρώπου-υπολογιστή και στον αθλητισμό.

Σε αυτή τη διπλωματική εργασία, ασχολούμαστε με την αναγνώριση ανθρωπίνων δράσεων χρησιμοποιώντας δεδομένα φορητών αισθητήρων που συλλέγονται σε μη ελεγχόμενες συνθήκες (*in-the-wild*), από έξυπνα τηλέφωνα και έξυπνα ρολόγια. Η συλλογή δεδομένων *in-the-wild* πραγματοποιείται στην καθημερινή ζωή, ώστε να εκπαιδευθούν μετά συστήματα ικανά να αναγνωρίζουν καλύτερα τις δραστηριότητες που εκτελούνται σε πραγματικές συνθήκες, και απαιτεί να πληρούνται οι ακόλουθες προϋποθέσεις: οι συμμετέχοντες να χρησιμοποιούν συσκευές που θα χρησιμοποιούσαν ούτως ή άλλως, χωρίς περιορισμούς στην τοποθέτηση των συσκευών, και να εκτελούν δραστηριότητες που θα εκτελούσαν πραγματικά, στο πραγματικό τους περιβάλλον.

Κατά τη χρήση δεδομένων που έχουν συλλεχθεί σε μη ελεγχόμενες συνθήκες σε συστήματα αναγνώρισης ανθρωπίνων δράσεων προκύπτουν πολλές προκλήσεις, που αφορούν κυρίως στην μη-ισορροπημένη επισημείωση των δεδομένων με πολλές ετικέτες, στα σφάλματα και στις παραλείψεις κατά την επισημείωση, στο θόρυβο που ενυπάρχει στις μετρήσεις των αισθητήρων, στην απώλεια μετρήσεων από επιμέρους αισθητήρες, και στη μεγάλη διαπροσωπική και ενδοπροσωπική μεταβλητότητα που χαρακτηρίζει τον τρόπο εκτέλεσης των δραστηριοτήτων.

Χρησιμοποιούμε το σύνολο δεδομένων ExtraSensory που περιέχει επισημειωμένα δεδομένα από 60 χρήστες συνολικής διάρκειας άνω των 300000 λεπτών, τα οποία έχουν συλλεχθεί από αισθητήρες έξυπνου κινητού και έξυπνου ρολογιού. Κάθε δείγμα του συνόλου δεδομένων αντιστοιχεί σε ένα λεπτό για το οποίο παρέχονται μετρήσεις πολλαπλών αισθητήρων και πολλαπλές ετικέτες δραστηριότητας και πλαισίου. Το υποσύνολο του συνόλου δεδομένων που χρησιμοποιούμε περιλαμβάνει μετρήσεις αισθητήρων από το επιταχυνσιόμετρο, το γυροσκόπιο, το GPS, τον ήχο και την κατάσταση του έξυπνου τηλεφώνου και από το επιταχυνσιόμετρο του έξυπνου ρολογιού, και περιλαμβάνει συνολικά 51 ετικέτες δραστηριότητας και πλαισίου. Το σύνολο δεδομένων περιέχει τόσο τις ανεπεξέργαστες μετρήσεις από τους αισθητήρες όσο και εξαχθέντα στατιστικά χαρακτηριστικά.

Αφού μελετήσουμε το σύνολο δεδομένων, αναπαράγουμε δύο baselines, λογιστική παλινδρόμηση και ένα multi-layer perceptron, χρησιμοποιώντας τα εξαχθέντα χαρακτηριστικά. Έπειτα, χρησιμοποιούμε ένα αμφίδρομο LSTM (BiLSTM) για να μοντελοποιήσουμε μια ακολουθία δειγμάτων, το οποίο επαυξάνουμε με ένα μηχανισμό αυτο-προσοχής (Self-Attention) που ενισχύει την απόδοση του μοντέλου, ή ένα μηχανισμό διασταυρούμενης προσοχής (Cross-Attention) που χρησιμοποιείται για την ερμηνευσιμότητα που προσδίδει. Εκτελούμε επίσης πειράματα βασιζόμενοι στα ανεπεξέργαστα δεδομένα από τους αισθητήρες, χρησιμοποιώντας στρώματα συνελικτικών νευρωνικών δικτύων (CNN) και στρώματα Transformer Encoder για την εξαγωγή χαρακτηριστικών, για τη μοντελοποίηση ενός δείγματος, και τα συνδυάζουμε περαιτέρω με ένα BiLSTM για τη μοντελοποίηση μιας ακολουθίας δειγμάτων. Σε όλα τα πειράματα, χρησιμοποιούμε μία συνάρτηση κόστους βασισμένη στο binary cross-entropy loss, με στάθμιση δειγμάτων ανά ετικέτα και ανά batch, για να χειριστούμε την ανισορροπία των ετικετών και τις ετικέτες που λείπουν. Στο τέλος παρέχουμε χρήσιμες κατευθύνσεις για την περαιτέρω βελτίωση της απόδοσης των μοντέλων, με βάση την εμπειρία μας από το συγκεκριμένο πρόβλημα.

Λέξεις Κλειδιά — Αναγνώριση Ανθρωπίνων Δράσεων, In-the-wild, Φορητές Συσκευές, ExtraSensory Dataset, Μηχανική Μάθηση, Βαθιά Μάθηση, Interpretability

Abstract

Building systems capable of automatically detecting and identifying activities performed by a person, and also the context in which each activity is performed, is the essence of Human Activity Recognition (HAR). These systems can use data collected from a wide range of sensors, including visual sensors e.g., video cameras, and non-visual sensors, including wearable sensors and ambient sensors. Human Activity Recognition has widespread applications in everyday life, predominantly in healthcare, elderly care, assisted living, human-computer interaction, assisted learning, and sports.

In this thesis, we tackle the HAR problem using wearable sensor data collected *in-the-wild*, from smartphones and smartwatches. Contrary to the scripted and heavily constrained procedure of collecting HAR data in the lab, *in-the-wild* data collection takes place in everyday life, in order to build systems capable of better recognizing activities performed in real-life conditions, and requires that four terms are met: the participants use their devices which they would naturally use, with unconstrained device placement, and they perform activities that they would naturally perform, in their natural environment.

This way, the collected data might better reflect real-life activities and contexts, but also a lot of challenges arise regarding using them for HAR systems. The task of labeling is harder when self-reporting during everyday life, and label annotation is worse both quantity-wise and quality-wise, since misusing or forgetting labels can lead to large portions of not annotated or wrongly annotated data. Open-ended label annotation leads to multi-label datasets that are extremely unbalanced. Collecting data using different types of devices combined with unconstrained device placement lead in data prone to noise, and missing sensors modalities are very common. Also, there is large inter-personal and intra-personal variability in the collected data since an activity might be performed differently among users, but also by a specific user at different times.

We use the ExtraSensory dataset which contains labeled data from 60 users totaling over 300k minutes, collected from smartphone and smartwatch sensors. Each data instance corresponds to one minute for which multi-sensor measurements and multiple relevant labels are provided. The dataset's subset that we use includes sensor measurements from the smartphone's accelerometer, gyroscope, GPS, audio and phone state and from the smartwatch's accelerometer, and includes 51 activity and context labels in total. The dataset contains both raw sensor measurements and pre-extracted statistical features. Regarding the GPS and audio sensors, we use only processed data, for privacy reasons.

After exploring the dataset to understand how unbalanced it is and to investigate how the features change when performing different activities or when different users perform the same activity, we reproduce some baseline prediction models from previous work, which include logistic regression and a simple multilayer perceptron, using the pre-extracted features. We build upon this work, and use a bidirectional LSTM (BiLSTM) to model a sequence of examples, again using the pre-extracted features. We also augment the BiLSTM model with a Self-Attention module which increases model performance, or a Cross-Attention module which is used for interpretability. We also run experiments using the raw sensor data, using convolutional neural network (CNN) layers and Transformer Encoder layers for feature extraction, to model a single example, and we further combine them with a BiLSTM to model a sequence of examples. In all our experiments we use a custom loss based on binary cross-entropy with instance weighting per-label and per-batch, to account for label imbalance and missing labels. At the end, we provide valuable insights to further improve HAR performance in future work, based on our experience on the task.

Keywords — Human Activity Recognition, In-the-wild, Wearable Devices, ExtraSensory Dataset, Machine Learning, Deep Learning, Interpretability

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, ολοκληρώνεται ο κύκλος των προπτυχιακών σπουδών μου στη σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Στο σημείο αυτό, θα ήθελα να ευχαριστήσω θερμά όλους τους ανθρώπους που στάθηκαν δίπλα μου στην πορεία αυτή και χωρίς τους οποίους δεν θα μπορούσα να φέρω εις πέρας αυτό το μεγάλο ταξίδι.

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Όρασης Υπολογιστών, Επικοινωνίας Λόγου και Επεξεργασίας Σημάτων. Πρώτα απ' όλα ευχαριστώ τον επιβλέποντα Καθηγητή μου, κ. Πέτρο Μαραγκό, για τη βοήθεια και τη στήριξή του. Στα φοιτητικά μου χρόνια, τα μαθήματά του αποτελούσαν σημείο αναφοράς, πηγή έμπνευσης, και έναυσμα για μελέτη και αναζήτηση της γνώσης. Τον ευχαριστώ για τη διδασκαλία, τις συμβουλές, την καθοδήγηση και το μεράκι του για την επιστήμη. Ευχαριστώ βαθύτατα, επίσης, την μεταδιδακτορική ερευνήτρια Δρ. Νάνσυ Ζλατίντση, η οποία καθ'όλη τη διάρκεια της παρούσας εργασίας στάθηκε δίπλα μου και παρείχε την υποστήριξη που χρειαζόταν για να καταστεί εφικτή η περάτωσή της. Ευχαριστώ ακόμα τον Μάνο Θεοδώση για την αμέριστη βοήθεια και τις συμβουλές του.

Αισθάνομαι επίσης την ανάγκη να ευχαριστήσω από τα βάθη της καρδιάς μου τους φίλους και συνοδοιπόρους στη φοιτητική μου ζωή, για την αγάπη και τη συμπαράστασή τους στα εύκολα και στα δύσκολα, στις χαρές και στις λύπες. Ευχαριστώ τον Μανώλη, την Αθανασία, τον Αλέξη, τον Κώστα, την Κλαίρη, τον Γιάννη Πλ., τη Νάγια, τον Γιάννη Β. και τον Ορέστη. Οι συνεργασίες, οι συζητήσεις, οι βόλτες και τα ταξίδια μας ήταν όλα αυτά τα χρόνια η καλύτερη συντροφιά, και ο καθένας έχει μια ιδιαίτερη θέση στην καρδιά μου. Ευχαριστώ επίσης τη Μυρσίνη, τη Χριστίνα και τον Κωνσταντίνο, με τους οποίους, αν και ήμασταν συμφοιτητές, ήρθαμε κοντά αργότερα, και η συμπαράστασή τους και η ψυχολογική αλληλοϋποστήριξη αποτέλεσε καθοριστικό παράγοντα για την ολοκλήρωση της παρούσας εργασίας.

Δεν ξεχνώ, όμως, τους φίλους από τα μαθητικά και τα προηγούμενα χρόνια. Κωνσταντίνα, Βασίλη, Ορέστη, Αλέξανδρε, Χάρη, Χριστίνα, Μυρτώ, Ειρήνη, ευχαριστώ πολύ για την κατανόηση και την υπομονή, για τη στήριξη και την εμψύχωση, για το γέλιο και για όλες τις χαρούμενες αλλά και συγκινητικές στιγμές που έχουμε ζήσει μαζί. Ευχαριστώ τη Στέλλα και το Θοδωρή, με τους οποίους ήρθαμε κοντά λόγω συγκυριών αλλά μείναμε κοντά γιατί γίναμε πραγματικοί φίλοι.

Στο σημείο αυτό οφείλω επίσης να ευχαριστήσω θερμά μια πολύ ξεχωριστή ομάδα, την ομάδα της Innoetics, στην οποία έχω την τιμή και τη χαρά να συμμετέχω στο πρώτο βήμα της επαγγελματικής μου ζωής. Η ομάδα αυτή μου έχει προσφέρει μαθήματα ζωής, συνεργασίες, γνώσεις, έμπνευση και κίνητρο, και είμαι εξαιρετικά ευγνώμων για την εμπιστοσύνη, τη στήριξη και την κατανόηση που έχω λάβει και εξαιρετικά περήφανη για όσα έχουμε πετύχει μαζί.

Θα ήθελα να αφιερώσω την εργασία αυτή στην οικογένειά μου, τους γονείς μου Παντελή και Σταυρούλα, τον αδερφό μου Στέφανο, τον παππού Στέφανο, τη γιαγιά Μόσχα και τη γιαγιά Βασιλική, οι οποίοι σε όλη μου τη ζωή με στηρίζουν με κάθε δυνατό τρόπο για να έχω την ευκαιρία να κάνω τα όνειρά μου πραγματικότητα.

Θα ήθελα επίσης να αφιερώσω την εργασία αυτή στον άνθρωπο που πιστεύει σε εμένα πάντα, ακόμα και τις φορές που εγώ δεν πίστευα στον εαυτό μου, και έχει σταθεί δίπλα μου σε όλες τις δυσκολίες, όπως στέκεται πάντα και σε όποιον άλλον τον χρειάζεται, με αμέριστη υπομονή και καλοσύνη. Είμαι ευτυχισμένη που έχω την ευκαιρία να σε βλέπω να χαμογελάς ξανά.

Αλεξάνδρα Βιώνη
Οκτώβριος 2023

Περιεχόμενα

Λίστα Σχημάτων

Κατάλογος Πινάκων

Chapter 0

Εκτεταμένη Περίληψη

0.1 Εισαγωγή

0.1.1 Αναγνώριση Ανθρωπίνων Δράσεων

Ο όρος *Αναγνώριση Ανθρωπίνων Δράσεων* (Human Activity Recognition – HAR) αναφέρεται στη διαδικασία ανάλυσης των χειρονομιών ή της κίνησης του ανθρώπινου σώματος, χρησιμοποιώντας δεδομένα που συλλέγονται από αισθητήρες, με σκοπό την αυτόματη αναγνώριση των δραστηριοτήτων που εκτελεί το άτομο [AT14; BBS14]. Πρόκειται για ένα ερευνητικό πεδίο που χαρακτηρίζεται από αυξημένη πολυπλοκότητα και αποτελεί σημείο διάδρασης και συγκερασμού πολλών επιστημονικών πεδίων, συμπεριλαμβανομένης της αλληλεπίδρασης ανθρώπου-υπολογιστή, της επεξεργασίας σήματος, της όρασης υπολογιστών, της στατιστικής ανάλυσης και της μηχανικής μάθησης. Ένα σύστημα αναγνώρισης ανθρωπίνων δράσεων βασισμένο στη μηχανική μάθηση μπορεί να χρησιμοποιεί τεχνικές επιβλεπόμενης, μη επιβλεπόμενης ή αυτο-επιβλεπόμενης μάθησης ή συνδυασμό τους.

Η αναγνώριση ανθρωπίνων δράσεων έχει ευρύ πεδίο εφαρμογών, στο οποίο περιλαμβάνεται η υγεία και η υγειονομική περίθαλψη (παρακολούθηση ασθενών για την υποστήριξη συστημάτων λήψης κλινικών αποφάσεων), τα συστήματα υποβοηθούμενης διαβίωσης για έξυπνα σπίτια, η καταγραφή και η αξιολόγηση της άσκησης και της ποιότητας ζωής, και οι εφαρμογές τηλε-επικοινωνίας [AT14; Che+12; RMM16]. Ένας σημαντικός στόχος της αναγνώρισης ανθρωπίνων δράσεων είναι να παρέχει πληροφορίες σχετικά με την ανθρώπινη συμπεριφορά που θα επιτρέψουν στα υπολογιστικά συστήματα να υποστηρίξουν ενεργά τις δραστηριότητες της καθημερινής ζωής του ανθρώπου.

Σε αδρές γραμμές, οι τεχνικές αναγνώρισης ανθρωπίνων δράσεων μπορούν να χωριστούν σε δύο βασικές κατηγορίες, ανάλογα με τον τύπο των αισθητήρων που χρησιμοποιούνται: στην *αναγνώριση δράσεων βάσει οπτικών αισθητήρων* και στην *αναγνώριση δράσεων βάσει μη οπτικών αισθητήρων* [Che+12; CJK15; RMM16]. Αυτές οι δύο κατηγορίες δεν είναι αλληλοαποκλειόμενες, καθώς μπορεί επίσης να χρησιμοποιηθεί ένας συνδυασμός και των δύο τύπων αισθητήρων.

- Η αναγνώριση δράσεων βάσει οπτικών αισθητήρων στηρίζεται σε μεγάλο βαθμό σε αισθητήρες όπως οι βιντεοκάμερες, οι οποίες συλλέγουν οπτικά δεδομένα στα οποία καταγράφονται οι ανθρώπινες δράσεις και συμπεριφορές ή οι σχετιζόμενες αλλαγές στις συνθήκες του περιβάλλοντος. Στην περίπτωση αυτή, μέθοδοι της όρασης υπολογιστών είναι απαραίτητες για την εξαγωγή χαρακτηριστικών από τις οπτικές παρατηρήσεις.

- Η αναγνώριση δράσεων βάσει μη οπτικών αισθητήρων στηρίζεται στη χρήση μεμονωμένων αισθητήρων ή δικτύων αισθητήρων προσαρτημένων στο άτομο του οποίου η συμπεριφορά καταγράφεται (φορητοί αισθητήρες ή έξυπνα κινητά) ή/και δίκτυα αισθητήρων προσαρτημένων σε αντικείμενα στο περιβάλλον εντός του οποίου επιθυμούμε να καταγράψουμε τις ανθρώπινες δρασεις (dense sensing). Σε γενικές γραμμές, τα δεδομένα που συλλέγονται μέσω της καταγραφής από μη οπτικούς αισθητήρες είναι, στην πλειοψηφία των περιπτώσεων, χρονοσειρές μεταβλητών που αναπαριστούν αλλαγές κατάστασης, και υποβάλλονται κατά το πλείστον σε επεξεργασία χρησιμοποιώντας πιθανοτική ή στατιστική ανάλυση, επεξεργασία σήματος και τεχνολογίες γνώσης.

Επιπροσθέτως, οι τεχνικές αναγνώρισης ανθρωπίνων δράσεων μπορούν να χωριστούν σε δύο βασικές κατηγορίες, με βάση τη θεμελιώδη αρχή πάνω στην οποία στηρίζονται τα μοντέλα πρόβλεψης δράσεων: στην *αναγνώριση δράσεων βασισμένη στα δεδομένα* (data-driven) και στην *αναγνώριση δράσεων βασισμένη στη γνώση* (knowledge-driven) [Che+12; Que+15]. Υπάρχουν επίσης υβριδικές προσεγγίσεις αναγνώρισης δράσεων που συνδυάζουν τεχνικές που βασίζονται σε δεδομένα και σε γνώση [OCW14].

- Η αναγνώριση δράσεων βασισμένη στα δεδομένα στηρίζεται στη μοντελοποίηση των δράσεων χρησιμοποιώντας ευρέως διαθέσιμα σύνολα δεδομένων μεγάλης κλίμακας για να εκπαιδευτούν πιθανοτικά ή στατιστικά μοντέλα δράσεων, ακολουθώντας την επονομαζόμενη bottom-up προσέγγιση. Αυτή η προσέγγιση υπερτερεί όσον αφορά στον χειρισμό της αβεβαιότητας και των δεδομένων σε μορφή χρονοσειρών, καθώς τα μοντέλα εκπαιδεύονται χρησιμοποιώντας αυτά τα δεδομένα, αλλά οι απαιτήσεις σε δεδομένα είναι συνήθως πολύ υψηλές, και ταυτόχρονα απαιτείται ειδικός χειρισμός για να εξασφαλιστεί η επεκτασιμότητα και να μπορούν τα μοντέλα να προσαρμοστούν σε καινούριους χρήστες.

- Η αναγνώριση δράσεων βασισμένη στη γνώση στηρίζεται στην πρότερη γνώση στον τομέα της εκάστοτε εφαρμογής, προκειμένου να δημιουργηθούν άμεσα μοντέλα δράσεων αξιοποιώντας μεθόδους μοντελοποίησης και αναπαράστασης γνώσης. Στην περίπτωση αυτή, τα μοντέλα δράσεων χρησιμοποιούνται

για την αναγνώριση δράσεων μέσω τυπικής λογικής συλλογιστικής, ακολουθώντας την επονομαζόμενη top-down προσέγγιση. Αυτή η προσέγγιση βασισμένη στη γνώση φέρει σαφή σημασιολογική ερμηνεία, αλλά η χρηστικότητά της είναι περιορισμένη στο χειρισμό της αβεβαιότητας και των δεδομένων σε μορφή χρονοσειρών. Επιπλέον, η ελλιπής γνώση στον τομέα της εφαρμογής θα έχει πιθανώς ως αποτέλεσμα να προκύψουν ελλιπή μοντέλα [Che+12; Que+15].

Η παρούσα διπλωματική εργασία εστιάζει στην αναγνώριση ανθρωπίνων δράσεων βασισμένη στα δεδομένα, χρησιμοποιώντας δεδομένα που έχουν συλλεχθεί από μη οπτικούς αισθητήρες φορητών συσκευών, όπως τα έξυπνα κινητά και τα έξυπνα ρολόγια. Η επιλογή αυτή έγινε κυρίως για πρακτικούς λόγους, καθώς πλέον τα κινητά τηλέφωνα φέρουν μεγάλο αριθμό αισθητήρων, όπως οι αδρανειακοί φορητοί αισθητήρες (επιταχυνσιόμετρα και γυροσκόπια), οπότε η συλλογή των δεδομένων μπορεί να γίνει με μη παρεμβατικό τρόπο. Επίσης, δεν επιλέχθηκαν οπτικοί αισθητήρες για λόγους προστασίας της ιδιωτικότητας.

0.1.2 Στοχοθεσία

Η ενδελεχής κατανόηση του αντικειμένου της παρούσας εργασίας προϋποθετει τη μελέτη της σύγχρονης βιβλιογραφίας σχετικά με την αναγνώριση ανθρωπίνων δράσεων που βασίζεται σε δεδομένα φορητών αισθητήρων και την κατανόηση των επιμέρους τμημάτων της αλυσίδας αναγνώρισης δράσεων, στην οποία συμπεριλαμβάνονται η προεπεξεργασία και η κατάτμηση των δεδομένων, η εξαγωγή χαρακτηριστικών και η ταξινόμηση. Είναι επίσης απαραίτητη η μελέτη των δημοφιλέστερων τεχνικών μηχανικής μάθησης και βαθιάς μάθησης, και των εφαρμογών τους στην αναγνώριση ανθρωπίνων δράσεων, με σκοπό να χρησιμοποιηθούν στα μοντέλα και στα πειράματά μας στη συνέχεια.

Οι πιο σύγχρονες και συνάμα απαιτητικές εφαρμογές που αφορούν στην αναγνώριση ανθρωπίνων δράσεων άπτονται της συλλογής δεδομένων μέσω φορητών αισθητήρων σε μη ελεγχόμενες συνθήκες (*in-the-wild*), που σημαίνει πως η συλλογή δεδομένων από τους χρήστες γίνεται σε συνθήκες καθημερινής διαβίωσης. Η συλλογή δεδομένων με αυτόν τον τρόπο εξασφαλίζει πλούσιο περιεχόμενο στο σύνολο δεδομένων που θα προκύψει, όσον αφορα στις αναπαραστάσεις και στις επισημειωμένες ετικέτες, σε σύγκριση με τη συλλογή δεδομένων στο εργαστήριο. Όμως, ταυτόχρονα, έχει ως αποτέλεσμα να γίνεται η αναγνώριση ανθρωπίνων δράσεων πολύ πιο απαιτητική διαδικασία, καθώς το σύνολο δεδομένων που προκύπτει είναι επίσης πολύ πιο δύσκολο στο χειρισμό του, πολύ λιγότερο αξιόπιστο, και περιέχει αρκετό θόρυβο. Θα ακολουθήσουμε αυτό το μονοπάτι, καθώς στόχος μας είναι να πειραματιστούμε με δεδομένα που συλλέγονται *in-the-wild*, και θα χρησιμοποιήσουμε το σύνολο δεδομένων ExtraSensory το οποίο περιέχει επισημειωμένα δεδομένα από 60 χρήστες συνολικής διάρκειας άνω των 300000 λεπτών, τα οποία έχουν συλλεχθεί από αισθητήρες έξυπνων κινητών και έξυπνων ρολογιών, και τα οποία είναι επισημειωμένα με 51 ετικέτες δράσεων (activities) και πλαισίων δράσεων (contexts).

Προκειμένου να διερευνήσουμε διεξοδικά την αναγνώριση ανθρωπίνων δράσεων *in-the-wild* χρησιμοποιώντας το σύνολο δεδομένων ExtraSensory, αρχικά είναι απαραίτητο να μελετήσουμε τις δημοσιεύσεις των ερευνητών που δημιούργησαν αυτό το σύνολο δεδομένων, με σκοπό να κατανοήσουμε τη διαδικασία συλλογής δεδομένων και τα επιμέρους τμήματα του συνόλου δεδομένων, τα οποία περιλαμβάνουν μεταξύ άλλων: τις ανεπεξέργαστες μετρήσεις των αισθητήρων, κάποια ήδη εξαχθέντα στατιστικά χαρακτηριστικά των μετρήσεων των αισθητήρων, και ετικέτες δράσεων και πλαισίου. Είναι επίσης απαραίτητο να μελετήσουμε και να λάβουμε υπόψη μας τα μοντέλα βάσης (baselines) που υλοποιήθηκαν από τους ερευνητές αυτούς, αλλά και τις δημοσιεύσεις άλλων ερευνητών που χρησιμοποιούν το ίδιο σύνολο δεδομένων.

Προτού υλοποιήσουμε και εκπαιδεύσουμε τα μοντέλα μας, πρέπει να εξερευνήσουμε το σύνολο δεδομένων για να κατανοήσουμε τις ιδιαιτερότητές του: την ανισορροπία στον αριθμό των επισημειωμένων δειγμάτων που αντιστοιχούν σε κάθε ετικέτα, πόσα δείγματα υπάρχουν ανά χρήστη και ανά ετικέτα, πώς αλλάζουν τα εξαχθέντα στατιστικά χαρακτηριστικά όταν εκτελούνται από τους χρήστες διαφορετικές δράσεις ή όταν διαφορετικοί χρήστες εκτελούν την ίδια δράση. Στη συνέχεια, αφού επαναλάβουμε τα πειράματα με τα baseline μοντέλα, τα οποία δέχονται ως είσοδο τα εξαχθέντα χαρακτηριστικά για ένα δείγμα, θα προσπαθήσουμε να βελτιώσουμε τα αποτελέσματα της αναγνώρισης δράσεων επαυξάνοντας το baseline μέσω ενός αμφίδρομου-LSTM (BiLSTM) για τη μοντελοποίηση όχι μόνο ενός δείγματος, αλλά μιας ακολουθίας δειγμάτων. Θα προσπαθήσουμε να βελτιώσουμε περαιτέρω αυτό το μοντέλο με την προσθήκη μηχανισμού αυτοπροσοχής (Self-Attention). Επίσης, θα διερευνήσουμε αν η προσθήκη ενός μηχανισμού διασταυρούμενης προσοχής (Cross-Attention) ανάμεσα στις εξόδους του BiLSTM και στα χαρακτηριστικά της εισόδου μπορεί να προσφέρει χρήσιμες πληροφορίες όσον αφορά στην ερμηνευσιμότητα του μοντέλου. Επιπλέον, θα κατασκευάσουμε μοντέλα που θα λαμβάνουν σαν

είσοδο τα ανεπεξέργαστα δεδομένα από τους αισθητήρες και θα πειραματιστούμε με πολλές αρχιτεκτονικές, που θα εμπεριέχουν στρώματα συνελικτικών νευρωνικών δικτύων (CNN), στρώματα Transformer Encoder και στρώματα BiLSTM. Στο τέλος της εργασίας, ευελπιστούμε να προτείνουμε μελλοντικές προεκτάσεις και να μεταβιβάσουμε τις εμπειρίες και τις γνώσεις που αποκτήθηκαν κατά τη διενέργεια αυτής της εργασίας προκειμένου να βελτιώσουμε περαιτέρω τα μοντέλα αναγνώρισης ανθρωπίνων δράσεων σε μελλοντικές εργασίες.

0.2 Το Σύνολο Δεδομένων ExtraSensory

Η Ενότητα αυτή έχει βασιστεί στην πρωτότυπη δουλειά των Yonatan Vaizman, Katherine Ellis, Gert Lanckriet και Nadir Weibel [VEL17; VWL18; Vai+18], στην οποία περιγράφεται η διαδικασία συλλογής και επεξεργασίας του συνόλου δεδομένων ExtraSensory, και η πρώτη χρήση του για την ανάπτυξη baseline μοντέλων για την αναγνώριση ανθρωπίνων δράσεων και συμπεριφορών.

0.2.1 Συλλογή Δεδομένων σε Μη Ελεγχόμενες Συνθήκες

Η ενασχόλησή μας με δεδομένα που έχουν συλλεχθεί *in-the-wild* έχει την αφετηρία της στη διαπίστωση πως τα συστήματα που έχουν εκπαιδευτεί για να αναγνωρίζουν δράσεις που εκτελούνται σε αυστηρά ελεγχόμενες συνθήκες και πλαίσιο, ενδέχεται να μην μπορούν να γενικεύσουν καλά σε δράσεις που λαμβάνουν χώρα σε πραγματικές συνθήκες, λόγω της αυξημένης μεταβλητότητάς τους σε ένα μη ελεγχόμενο περιβάλλον. Προς αυτή την κατεύθυνση, οι ερευνητές που δημιούργησαν το σύνολο δεδομένων ExtraSensory εισήγαγαν τέσσερις συνθήκες *in-the-wild* [VEL17] που πρέπει να πληρούνται για να διασφαλίζεται ότι η συλλογή δεδομένων και κατ' επέκταση η έρευνα γίνεται σε φυσικές και ρεαλιστικές συνθήκες:

1. *Προσωπικές συσκευές των χρηστών.* Η χρήση μιας ή περισσότερων ξένων συσκευών μπορεί να επηρεάσει τη φυσικότητα της συμπεριφοράς ενός χρήστη, επομένως το ιδανικό είναι οι χρήστες να χρησιμοποιούν τις προσωπικές τους συσκευές (έξυπνα τηλέφωνα, έξυπνα ρολόγια κ.λπ.), όπως θα έκαναν κανονικά.
2. *Τοποθέτηση συσκευών χωρίς περιορισμούς.* Παρόλο που η τοποθέτηση και ο προσανατολισμός της φορητής συσκευής μπορεί να έχει καθοριστική επίδραση στην ακρίβεια της ταξινόμησης δραστηριοτήτων, η επιβολή συγκεκριμένης τοποθέτησης και προσανατολισμού της συσκευής δεν είναι πρακτική σε εφαρμογές του πραγματικού κόσμου, επομένως θα πρέπει να αποφεύγεται και η προκύπτουσα μεταβλητότητα των συλλεχθέντων σημάτων θα πρέπει να αντιμετωπίζεται ως μια πρόκληση που πρέπει να διαχειριστούμε κατάλληλα.
3. *Ρεαλιστικό περιβάλλον.* Οι δραστηριότητες που εκτελεί ο κάθε χρήστης θα πρέπει να εκτελούνται σε τόπο και χρόνο κατάλληλο για κάθε χρήστη, και θα πρέπει να είναι ο τόπος και ο χρόνος στον οποίο ο κάθε χρήστης θα εκτελούσε φυσικά τις δραστηριότητες αυτές.
4. *Ρεαλιστικές συμπεριφορές.* Οι δραστηριότητες που εκτελεί ο κάθε χρήστης θα πρέπει να είναι δραστηριότητες που ο συγκεκριμένος χρήστης θα εκτελούσε φυσιολογικά ως μέρος της καθημερινότητας του. Επίσης, οι χρήστες δεν θα πρέπει να καθοδηγούνται να εκτελούν προδιαγεγραμμένες αλληλουχίες δραστηριοτήτων.

Όταν εφαρμόζονται οι παραπάνω προϋποθέσεις για τη συλλογή δεδομένων ανθρωπίνων δράσεων σε μη ελεγχόμενες συνθήκες, η επισημείωση των δεδομένων με ετικέτες δράσεων και σχετικών πλαισίων μετατρέπεται σε αρκετά δύσκολο έργο, επειδή οι επιμέρους δράσεις δεν είναι προσχεδιασμένες και η διάρκεια και η αλληλουχία τους δεν είναι προκαθορισμένες. Η προτεραιοποίηση των συνθηκών *in-the-wild* μπορεί να οδηγήσει σε λιγότερα επισημειωμένα δείγματα και συνεπώς σε λιγότερες επισημειώσεις ανά δραστηριότητα, καθώς η ανάγκη για διαρκή επισημείωση όλων των δραστηριοτήτων για κάθε χρήστη μπορεί να είναι κουραστική και δύσκολη, και επίσης είναι μια επισφαλής διαδικασία καθώς οι χρήστες που επισημειώνουν τις δραστηριότητες τους κατά τη διάρκεια όλης της καθημερινότητας μπορεί να είναι επιρρεπείς σε λάθη.

Η μεγάλης κλίμακας συλλογή δεδομένων για τη δημιουργία του συνόλου δεδομένων ExtraSensory πραγματοποιήθηκε με τη βοήθεια της εφαρμογής ExtraSensory App για έξυπνα κινητά (Android και iOS), καθώς και μιας εφαρμογής προσαρμοσμένης για το έξυπνο ρολόι Pebble. Η εφαρμογή αυτή ήταν υπεύθυνη για την αυτόματη συλλογή δεδομένων από τους αισθητήρες στο έξυπνο κινητό και στο έξυπνο ρολόι, καθώς για κάθε λεπτό που ήταν ενεργή, κατέγραφε τις μετρήσεις των αισθητήρων για ένα παράθυρο 20 δευτερολέπτων.

Επιπροσθέτως, η εφαρμογή παρείχε μια εύχρηστη διεπαφή για να μπορούν οι χρήστες να επισημειώνουν τις δραστηριότητές τους.

Για τη δημιουργία του συνόλου δεδομένων ExtraSensory συλλέχθηκαν δεδομένα από 60 χρήστες, συνολικής διάρκειας άνω των 300 χιλιάδων λεπτών, προκειμένου να μπορούν να χρησιμοποιηθούν για τη μοντελοποίηση ανθρωπίνων δράσεων και πλαισίων ανθρώπινης συμπεριφοράς *in-the-wild*. Κάθε δείγμα του συνόλου δεδομένων αντιστοιχεί σε ένα λεπτό, για το οποίο παρέχονται οι μετρήσεις πολλαπλών αισθητήρων και οι συναφείς ετικέτες. Η επισημείωση των ετικετών δράσεων και πλαισίου συμπεριφοράς έχει γίνει με self-reporting, και τα δεδομένα έχουν επισημειωθεί με *πολλαπλές* συναφείς ετικέτες, έτσι ώστε να αναπαριστώνται ρεαλιστικά οι συνθήκες multi-tasking στην ανθρώπινη καθημερινότητα.

0.2.2 Δεδομένα Αισθητήρων

Το σύνολο δεδομένων ExtraSensory περιέχει τις ακόλουθες μετρήσεις από αισθητήρες:

- **Μετρήσεις υψηλής συχνότητας.** Οι ακόλουθοι αισθητήρες των έξυπνων κινητών τηλεφώνων δειγματοληπτήθηκαν στα 40Hz κατά τη διάρκεια του παραθύρου καταγραφής 20sec κάθε λεπτού, παράγοντας μια χρονοσειρά περίπου 800 χρονικών σημείων.
  - *Επιταχυνσιόμετρο.* Χρονοσειρές διανυσμάτων επιτάχυνσης τριών αξόνων κατά μήκος των τυπικών αξόνων των συσκευών έξυπνων τηλεφώνων.
  - *Γυροσκόπιο.* Χρονοσειρές διανυσμάτων τριών αξόνων του ρυθμού περιστροφής γύρω από κάθε τυπικό άξονα των συσκευών έξυπνων τηλεφώνων.
  - *Μαγνητόμετρο.* Χρονοσειρά διανυσμάτων τριών αξόνων του μαγνητικού πεδίου.
- **Μετρήσεις έξυπνου ρολογιού.** Το έξυπνο ρολόι Pebble κατέγραψε σήματα από τους δύο διαθέσιμους αισθητήρες του.
  - *Επιταχυνσιόμετρο.* Χρονοσειρές διανυσμάτων επιτάχυνσης τριών αξόνων κατά μήκος των τυπικών αξόνων της συσκευής έξυπνου ρολογιού, με συχνότητα δειγματοληψίας 25Hz.
  - *Πυξίδα.* Χρονοσειρά της γωνίας πορείας, η οποία δεν έχει σταθερό ρυθμό δειγματοληψίας, αλλά παρείχε ενημέρωση κάθε φορά που ανιχνευόταν αλλαγή μεγαλύτερη από μία μοίρα.
- **Μετρήσεις θέσης.** Η εφαρμογή ExtraSensory App για έξυπνα κινητά τηλέφωνα συλλέγει δεδομένα τοποθεσίας χρησιμοποιώντας το GPS του κινητού, που δεν έχει σταθερό ρυθμό δειγματοληψίας, αλλά παρέχει ενημέρωση κάθε φορά που ανιχνεύει κίνηση. Οι μετρήσεις θέσης που έχουν συλλεχθεί είναι χρονοσειρές μεταβλητού μήκους, που κυμαίνεται από ένα μόνο χρονικό σημείο έως και περισσότερα από είκοσι χρονικά σημεία. Κάθε ενημέρωση θέσης περιέχει τις εκτιμώμενες μετρήσεις θέσης (γεωγραφικό πλάτος, γεωγραφικό μήκος, υψόμετρο, ταχύτητα, κατακόρυφη ακρίβεια, οριζόντια ακρίβεια) που αντιστοιχούν σε μια συγκεκριμένη χρονική στιγμή αναφοράς.
- **Μετρήσεις χαμηλής συχνότητας.** Οι μετρήσεις αυτές συλλέγονται μία φορά σε κάθε παράθυρο καταγραφής.
  - *Κατάσταση έξυπνου τηλεφώνου.* Συμπεριλαμβάνει την κατάσταση της εφαρμογής (foreground, background), την κατάσταση της συνδεσιμότητας Wi-Fi, την κατάσταση της μπαταρίας (φόρτιση, εκφόρτιση), το επίπεδο μπαταρίας, και την κατάσταση τηλεφωνικής κλήσης.
  - *Αισθητήρες συνθηκών περιβάλλοντος.* Μετρήσεις από αισθητήρες εγγύτητας, περιβάλλοντος φωτισμού, θερμοκρασίας, υγρασίας, πίεσης αέρα κ.ά. (αν υπάρχουν οι αισθητήρες αυτοί στο έξυπνο τηλέφωνο)
- **Καταγραφή ήχου.** Κατά τη διάρκεια κάθε διαστήματος εγγραφής διάρκειας περίπου 20sec, καταγράφηκε ήχος από το μικρόφωνο του έξυπνου τηλεφώνου, με συχνότητα δειγματοληψίας 22050Hz, όταν το τηλέφωνο δεν χρησιμοποιούνταν για κλήση. Κάθε μη επεξεργασμένο ηχητικό σήμα κανονικοποιήθηκε ώστε να έχει μέγιστο πλάτος 1 και στη συνέχεια υπολογίστηκαν στο τηλέφωνο οι συντελεστές Mel Frequency Cepstral Coefficients (MFCC), χρησιμοποιώντας παράθυρα μισής επικάλυψης 2048 δειγμάτων, 40 ζώνες συχνοτήτων με κλίμακα Mel και 13 συντελεστές Cepstral, συμπεριλαμβανομένου του μηδενικού συντελεστή. Μόνο ο συντελεστής κανονικοποίησης και τα MFCC στάλθηκαν στον διακομιστή, προκειμένου να διασφαλιστεί η ιδιωτικότητα των χρηστών.

0.2.3 Εξαχθέντα Χαρακτηριστικά

Το σύνολο δεδομένων ExtraSensory περιέχει επίσης τα ακόλουθα χαρακτηριστικά, τα οποία έχουν εξαχθεί από τα σήματα από τους προαναφερθέντες αισθητήρες, ως ακολούθως:

- **Επιταχυνσιόμετρο και γυροσκόπιο έξυπνου κινητού** (*26 χαρακτηριστικά*)**.** Δεδομένου ότι οι χρήστες είναι ελεύθεροι να χρησιμοποιούν το έξυπνο τηλέφωνό τους με οποιονδήποτε τρόπο τους βολεύει και δεν είναι υποχρεωμένοι να το προσαρτούν σε συγκεκριμένη θέση, δεν μπορεί να θεωρηθεί ότι είναι προσανατολισμένο με συγκεκριμένο τρόπο. Για τον λόγο αυτό, δεν αποδόθηκε ιδιαίτερη σημασία σε κανέναν από τους άξονες του έξυπνου τηλεφώνου και τα περισσότερα χαρακτηριστικά εξάγονται από το πλάτος του σήματος. Το διανυσματικό σήμα πλάτους υπολογίστηκε ως η Ευκλείδεια νόρμα της μέτρησης της επιτάχυνσης στους τρεις άξονες, σε κάθε χρονική στιγμή. Έπειτα, εξήχθησαν τα ακόλουθα χαρακτηριστικά:
  - Εννέα βασικά στατιστικά χαρακτηριστικά του σήματος πλάτους: μέσος όρος, τυπική απόκλιση, τρίτη ροπή, τέταρτη ροπή, 25° εκατοστημόριο, 50° εκατοστημόριο, 75° εκατοστημόριο, εντροπία τιμής (εντροπία που υπολογίζεται από ένα ιστόγραμμα κβαντισμού των τιμών πλάτους σε 20 bins) και εντροπία χρόνου (εντροπία που υπολογίζεται από την κανονικοποίηση του σήματος πλάτους και τον χειρισμό του ως κατανομή πιθανότητας, η οποία έχει σχεδιαστεί για να ανιχνεύει αιχμές στο χρόνο, δηλαδή ξαφνικές ακραίες αυξομειώσεις στο πλάτος).
  - Έξι φασματικά χαρακτηριστικά του σήματος πλάτους: λογαριθμικές ενέργειες σε 5 υποζώνες (0-0.5Hz, 0.5-1Hz, 1-3Hz, 3-5Hz, >5Hz) και φασματική εντροπία.
  - Δύο χαρακτηριστικά αυτοσυσχέτισης από το σήμα πλάτους. Ο μέσος όρος του σήματος πλάτους (συνιστώσα DC) αφαιρέθηκε και η συνάρτηση αυτοσυσχέτισης υπολογίστηκε και κανονικοποιήθηκε έτσι ώστε η τιμή αυτοσυσχέτισης στο lag 0 να είναι 1. Εντοπίστηκε η μέγιστη τιμή μετά τον κύριο λοβό. Η αντίστοιχη περίοδος σε δευτερόλεπτα υπολογίστηκε ως η κυρίαρχη περιοδικότητα και εξήχθη επίσης η κανονικοποιημένη τιμή αυτοσυσχέτισής της.
  - Εννέα στατιστικά στοιχεία της χρονοσειράς τριών αξόνων: ο μέσος όρος και η τυπική απόκλιση κάθε άξονα και οι 3 συντελεστές συσχέτισης μεταξύ των αξόνων.
- **Επιταχυνσιόμετρο έξυπνου ρολογιού** (*46 χαρακτηριστικά*)**.** Δεδομένου ότι το έξυπνο ρολόι τοποθετείται με συγκεκριμένο τρόπο γύρω από τον καρπό του χρήστη, μπορεί να αποδοθεί νόημα στους άξονές του. Έτσι, εκτός από τα ίδια 26 χαρακτηριστικά που προαναφέρθηκαν, 20 επιπλέον χαρακτηριστικά εξήχθησαν από τα μη επεξεργασμένα σήματα που συλλέχθηκαν από το επιταχυνσιόμετρο του έξυπνου ρολογιού:
  - Δεκαπέντε χαρακτηριστικά ειδικά για κάθε άξονα: λογαριθμικές ενέργειες στις ίδιες υποζώνες όπως παραπάνω (0-0,5Hz, 0,5-1Hz, 1-3Hz, 3-5Hz, >5Hz), αλλά υπολογισμένες για το σήμα κάθε άξονα ξεχωριστά.
  - Πέντε χαρακτηριστικά σχετικής κατεύθυνσης: μετά τον υπολογισμό της ομοιότητας συνημίτονου μεταξύ των κατευθύνσεων της επιτάχυνσης οποιωνδήποτε δύο χρονικών σημείων της χρονοσειράς, υπολογίστηκαν οι μέσες τιμές αυτών των τιμών σε 5 διαφορετικές περιοχές time-lag μεταξύ των συγκρινόμενων χρονικών στιγμών (0-0.5sec, 0.5-1sec, 1-5sec, 5-10sec, >10sec).
- **Καταγραφή θέσης** (*17 χαρακτηριστικά*)**.** Τα χαρακτηριστικά θέσης που εξήχθησαν βασίζονται μόνο σε σχετικές θέσεις, προκειμένου το σύνολο δεδομένων να μπορεί να γενικευτεί καλύτερα σε οποιαδήποτε θέση και να μην περιορίζεται στην περιοχή της πανεπιστημιούπολης USCD όπου έγινε η συλλογή των δεδομένων. Εξήχθησαν τα ακόλουθα χαρακτηριστικά:
  - Έξι χαρακτηριστικά υπολογίστηκαν απευθείας στο τηλέφωνο: τυπική απόκλιση του γεωγραφικού πλάτους, τυπική απόκλιση του γεωγραφικού μήκους, μεταβολή του γεωγραφικού πλάτους, μεταβολή του γεωγραφικού μήκους, μέση απόλυτη τιμή της παραγώγου του γεωγραφικού πλάτους και μέση απόλυτη τιμή της παραγώγου του γεωγραφικού μήκους.
  - Έντεκα ακόμη χαρακτηριστικά υπολογίστηκαν εξ αποστάσεως με βάση τις μεταδιδόμενες μετρήσεις θέσης: αριθμός ενημερώσεων, λογάριθμος του γεωγραφικού πλάτους, λογάριθμος του γεωγραφικού μήκους, ελάχιστο υψόμετρο, μέγιστο υψόμετρο, ελάχιστη ταχύτητα, μέγιστη ταχύτητα, καλύτερη (χαμηλότερη) κατακόρυφη ακρίβεια, καλύτερη (χαμηλότερη) οριζόντια ακρίβεια και διάμετρος (μέγιστη απόσταση μεταξύ δύο θέσεων στο παράθυρο καταγραφής, σε μέτρα).

- **Καταγραφή ήχου** (*26 χαρακτηριστικά*)**.** Από τη χρονοσειρά των 13-διάστατων διανυσμάτων MFCC υπολογίστηκαν η μέση τιμή και η τυπική απόκλιση καθενός από τους 13 συντελεστές.
- **Κατάσταση έξυπνου κινητού** (*34 χαρακτηριστικά*)**.** Χρησιμοποιήθηκαν μόνο οι διακριτές πληροφορίες κατάστασης των έξυπνων τηλεφώνων, οι οποίες αναπαρίστανται με ένα 26-διάστατο διάνυσμα one-hot. Για κάθε μία από τις ακόλουθες ιδιότητες, χρησιμοποιήθηκε ένας δυαδικός δείκτης για κάθε μία από τις πιθανές τιμές, καθώς και ένας επιπρόσθετος δείκτης που υποδεικνύει ελλιπή δεδομένα.
  - Κατάσταση εφαρμογής (3 επιλογές: ενεργό, ανενεργό, φόντο)
  - Μπαταρία συνδεδεμένη (3 επιλογές: AC, USB, ασύρματο)
  - Κατάσταση μπαταρίας (6 επιλογές: άγνωστη, αποσυνδεδεμένη, μη φόρτιση, αποφόρτιση, φόρτιση, πλήρης)
  - Σε τηλεφωνική κλήση (2 επιλογές: True, False)
  - Λειτουργία κουδουνίσματος (3 επιλογές: κανονικό, αθόρυβο χωρίς δόνηση, αθόρυβο με δόνηση)
  - Κατάσταση Wi-Fi (3 επιλογές: μη προσβάσιμο, προσβάσιμο μέσω WiFi, προσβάσιμο μέσω WWAN)

  Όσον αφορά στις πληροφορίες για την ώρα της ημέρας για κάθε δείγμα, η χρονοσφραγίδα του χρησιμοποιήθηκε για να εξαχθεί η συνιστώσα της ώρας (μία από τις 24 διακριτές τιμές). Στη συνέχεια, δημιουργήθηκαν 8 χρονικά εύρη με μισή επικάλυψη: μεσάνυχτα-6πμ, 3πμ-9πμ, 6πμ-μεσημέρι, 9πμ-3μμ, μεσημέρι-6μμ, 3μμ-9μμ, 6μμ-μεσάνυχτα και 9μμ-3πμ. Η ώρα κάθε δείγματος αναπαριστάται με μια δυαδική τιμή 8-bit, όπου ακριβώς 2 bins θα είναι ενεργά.

0.2.4 Ετικέτες

Στον Πίνακα 1 παρουσιάζουμε τις 51 ετικέτες δράσεων και πλαισίου που περιλαμβάνονται στο σύνολο δεδομένων ExtraSensory. Ο κάθε χρήστης μπορούσε να επισημειώνει κάθε στιγμή όλες τις συναφείς με τις δράσεις του ετικέτες, και έτσι κάθε δείγμα του προκύπτοντος συνόλου δεδομένου είναι επισημειωμένο με μία ή περισσότερες ετικέτες. Η ομαδοποίηση στον Πίνακα δεν εμπεριέχει πληροφορία για αλληλοαποκλειόμενες ετικέτες, καθώς οι χρήστες κατά τη διάρκεια της συλλογής δεδομένων είχαν ελευθερία κατά την επισημείωση των δράσεων. Ο Πίνακας δημιουργήθηκε από εμάς και αποσκοπεί απλά στην καλύτερη παρουσίαση των ετικετών και στην εννοιολογική ομαδοποίησή τους για την καλύτερη κατανόηση του προβλήματος αναγνώρισης ανθρωπίνων δράσεων που έχουμε να διαχειριστούμε.

Οι ετικέτες του συνόλου δεδομένων ExtraSensory ομαδοποιημένες εννοιολογικά

Θεματική	**Ετικέτες**
Στάση/Κίνηση	“Lying down”, “Sitting”, “Standing”, “Walking”, “Running”, “Bicycling”
Ειδική Κίνηση	“Strolling”, “Stairs - Going up”, “Stairs - Going down”, “Elevator”
Θέση Κινητού	“Phone in pocket”, “Phone in hand”, “Phone in bag”, “Phone on table”
Εργασία	“In class”, “Lab work”, “Computer work”, “In a meeting”
Τοποθεσίες	“At home”, “At school”, “At main workplace”, “At a restaurant”, “At a bar”, “At a party”, “At the gym”, “At the beach”
Μετακίνηση	“In a car”, “On a bus", “Drive - Driver", “Drive - Passenger"
Δουλειές Σπιτιού	“Shopping”, “Cooking”, “Cleaning”, “Doing laundry”, “Washing dishes”
Αυτοφροντίδα	“Bathing - Shower”, “Toilet”, “Grooming”, “Dressing”, “Sleeping”
Ελεύθερος Χρόνος	“Exercise”, “Eating”, “Drinking alcohol”, “Watching TV”, “Surfing the internet”, “Talking”, “Singing”
Συντροφιά	“With co-workers”, “With friends”
Περιβάλλον	“Indoors”, “Outside”

Table 1: Διαισθητική ομαδοποίηση των ετικετών δράσεων και πλαισίου του συνόλου δεδομένων ExtraSensory

Κατά τη διάρκεια της συλλογής δεδομένων, ο χρήστης ανά πάσα στιγμή μπορούσε να επισημειώνει στη εφαρμογή τις δραστηριότητες που εκτελεί και τα συναφή πλαίσια. Συνεπώς το σύνολο δεδομένων που δημιουργήθηκε, για

κάθε δείγμα περιείχε και ένα διάνυσμα 51 δυαδικών τιμών. Κατά την επεξεργασία του συνόλου δεδομένου από τους δημιουργούς του, κάποιες ετικέτες μετατράπηκαν σε "missing labels" και επισημειώθηκαν με `NaN`. Επί της ουσίας, οι ετικέτες που είναι επισημειωμένες με `NaN` στο τελικό σύνολο δεδομένων, είναι ένα υποσύνολο των *πραγματικών* missing labels (των ετικετών που είτε ο χρήστης παρέλειψε να επισημειώσει, είτε επισημείωσε λανθασμένα οπότε δεν μπορούμε να γνωρίζουμε αν είναι αξιόπιστες), και έχουν ανιχνευθεί με ευριστικές τεχνικές, όπως π.χ. αν ένα δείγμα έχει επισημειωθεί με δύο ρεαλιστικά αμοιβαίως αποκλειόμενες ετικέτες ταυτόχρονα, ας πούμε "Sleeping" και "Running", τότε και οι δύο μπορούν να τεθούν ως missing labels καθώς δεν μπορούμε να γνωρίζουμε ποια είναι η σωστή οπότε καλύτερα να μη λάβουμε υπόψη καμία απο τις δύο. Αυτή η διαδικασία έγινε από τους ερευνητές που δημιούργησαν το σύνολο δεδομένων, έτσι ώστε να μειωθούν όσο γίνεται τα σφάλματα στις επισημειώσεις, χωρίς βέβαια να είναι εφικτό να ανιχνευθούν με αυτόν τον τρόπο όλα τα λάθη και οι παραλείψεις. Βοηθάει όμως στο να μη λαμβάνουν τα μοντέλα υπόψη κάποιες επισημειώσεις που μπορούμε με σιγουριά να καταλάβουμε πως είναι επισφαλείς, αγνοώντας τις missing ετικέτες στον υπολογισμό του loss.

0.3 Πειραματική Διάταξη

0.3.1 Ρυθμίσεις Πειραμάτων

Σε όλα τα πειράματα χρησιμοποιείται το "Core" υποσύνολο του συνόλου δεδομένων ExtraSensory, το οποίο περιλαμβάνει 169001 δείγματα που έχουν μετρήσεις και από τους έξι βασικούς αισθητήρες: επιταχυνσιόμετρο, γυροσκόπιο, μετρήσεις θέσης, καταγραφή ήχου και κατάσταση τηλεφώνου από το έξυπνο κινητό τηλέφωνο, και επιταχυνσιόμετρο από το έξυπνο ρολόι.

Τα μοντέλα μας με βάση το είδος των δεδομένων που χρησιμοποιούνται για την εκπαίδευσή τους, μπορούν να χωριστούν σε δύο κατηγορίες:

- **Μοντέλα που χρησιμοποιούν τα εξαχθέντα χαρακτηριστικά.** Στα μοντέλα αυτά χρησιμοποιούμε ως είσοδο για κάθε δείγμα τα συνολικά 175 προαναφερθέντα χαρακτηριστικά από τους έξι βασικούς αισθητήρες.
- **Μοντέλα που χρησιμοποιούν τις μετρήσεις των αισθητήρων**. Στα μοντέλα αυτά χρησιμοποιούμε τις μετρήσεις από το επιταχυνσιόμετρο και το γυροσκόπιο του έξυπνου τηλεφώνου και από το επιταχυνσιόμετρο του έξυπνου ρολογιού, και τη χρονοσειρά των MFCC όσον αφορά στον ήχο. Για τις μετρήσεις θέσης και την κατάσταση του τηλέφωνου χρησιμοποιούμε τα προαναφερθέντα εξαχθέντα χαρακτηριστικά. Όσον αφορά το μήκος των χρησιμοποιούμενων χρονοσειρών, επειδή διαφέρει ανά αισθητήρα από δείγμα σε δείγμα λόγω διαφοροποιήσεων στη διάρκεια του παραθύρου καταγραφής, για να μπορούμε να χρησιμοποιήσουμε Συνελικτικά Νευρωνικά Δίκτυα, ορίζουμε ένα σταθερό μήκος χρονοσειράς ανά αισθητήρα (το αναμενόμενο με βάση τη συχνότητα δειγματοληψίας του κάθε αισθητήρα για τους αδρανειακούς, και αυθαίρετα για τον ήχο), το οποίο είναι 800 για το επιταχυνσιόμετρο και το γυροσκόπιο του έξυπνου τηλεφώνου, 500 για το επιταχυνσιόμετρο του έξυπνου ρολογιού, και 700 για τη χρονοσειρά των MFCC, και στην περίπτωση που η χρονοσειρά του κάθε δείγματος δεν έχει το επιθυμητό μήκος, την επαναλαμβάνουμε ολόκληρη έως ότου φτάσουμε στο επιθυμητό μήκος. Αυτή η διαδικασία εισάγει κάποια artifacts στα δεδομένα μας, αλλά είναι προτιμητέα έναντι του zero padding ή του padding χρησιμοποιώντας την τελευταία διαθέσιμη τιμή της χρονοσειράς.

Τα μοντέλα μας με βάση το πλήθος των δειγμάτων που λαμβάνουν στην είσοδό τους, μπορούν να χωριστούν σε δύο κατηγορίες:

- **Μοντέλα που χρησιμοποιούν ένα δείγμα.** Τα μοντέλα αυτά λαμβάνουν στην είσοδο ένα δείγμα και προβλέπουν τις ετικέτες του.
- **Μοντέλα που χρησιμοποιούν μια ακολουθία διαδοχικών δειγμάτων.** Τα μοντέλα αυτά λαμβάνουν στην είσοδο μια ακολουθία δειγμάτων διαδοχικών λεπτών, και έχουν ως στόχο να προβλέψουν τις ετικέτες του τελευταίου δείγματος. Παρέχουμε συνεπώς πληροφορίες για το άμεσο παρελθόν. Στα πειράματά μας δοκιμάσαμε ακολουθίες δειγμάτων με μήκη {5, 10, 15, 30} δείγματα, δηλαδή από 5 έως 30 λεπτά, αφού κάθε δείγμα αντιστοιχεί σε ένα λεπτό. Σε περίπτωση που το δείγμα για κάποιο από τα λεπτά της ακολουθίας διαδοχικών παρελθοντικών δειγμάτων δεν είναι διαθέσιμο στο σύνολο δεδομένων, χρησιμοποιούμε ξανά το δείγμα του ακριβώς *επόμενου* λεπτού.

Για κάθε μοντέλο, σε κάθε πείραμα, είτε χρησιμοποιούμε τα ήδη εξαχθέντα χαρακτηριστικά είτε τις ανεπεξέργαστες μετρήσεις των αισθητήρων, πριν χρησιμοποιήσουμε τα δεδομένα, τα κανονικοποιούμε (standardization). Για τη διαδικασία αυτή χρησιμοποιούμε τη μέση τιμή και την τυπική απόκλιση για κάθε χαρακτηριστικό (όταν χρησιμοποιούμε χαρακτηριστικά ως είσοδο) ή για κάθε αισθητήρα (όταν χρησιμοποιούμε τα σήματα από τους αισθητήρες ως είσοδο), όπως αυτές έχουν προσδιοριστεί από το εκάστοτε σύνολο εκπαίδευσης για την κάθε επανάληψη του five-fold cross validation. Όταν χρησιμοποιούμε τα ήδη εξαχθέντα χαρακτηριστικά ως είσοδο, αν υπάρχουν missing values σε κάποια χαρακτηριστικά, τα θέτουμε στην τιμή μηδέν αφού έχουμε κανονικοποιήσει τα χαρακτηριστικά, και έτσι φέρουν πλέον τη μέση τιμή του χαρακτηριστικού βάσει του συνόλου εκπαίδευσης.

Χρησιμοποιούμε `batch_size = 32` κατά την εκπαίδευση όλων των μοντέλων. Όσον αφορά τη συνάρτηση κόστους, χρησιμοποιούμε ένα custom loss βασισμένο στο `torch.nn.BCEWithLogitsLoss`, ελαφρώς τροποποιημένο ώστε να επιτρέπει το δυναμικό masking ανά batch και ανά στοιχείο στα loss matrices (έτσι ώστε να μηδενίζουμε τα στοιχεία του πίνακα που αντιστοιχούν σε missing ground-truth labels σε κάθε batch και να μην χρησιμοποιούνται στον υπολογισμό του loss). Επίσης, το `pos_weight` χρησιμοποιείται για τη στάθμιση δειγμάτων ώστε να ληφθεί υπόψη η ανισορροπία στον αριθμό των θετικών δειγμάτων ανά ετικέτα, πολλαπλασιάζοντας τον όρο που αντιστοιχεί στα θετικά δείγματα για την ετικέτα στη συνάρτηση κόστους, με το λόγο των αρνητικών προς τα θετικά δείγματα για αυτή την ετικέτα στο σύνολο εκπαίδευσης. Επιπλέον, χρησιμοποιούμε τον Adam [KB15] optimizer για την εκπαίδευση όλων των μοντέλων, και προσαρμόζουμε το `learning rate` ανάλογα με την αρχιτεκτονική και τον αριθμό παραμέτρων του κάθε μοντέλου.

Κατά το inference, οι συνεχείς προβλέψεις των μοντέλου περνούν από μια συνάρτηση ενεργοποίησης Sigmoid και στη συνέχεια μετατρέπονται σε δυαδικές εξόδους εφαρμόζοντας κατώφλι 0.5. Οι μετρικές αξιολόγησης υπολογίζονται για κάθε ετικέτα χρησιμοποιώντας όλα τα δείγματα τα οποία δεν έχουν ground-truth επισημείωση "missing label" για τη συγκεκριμένη ετικέτα.

Η λογιστική παλινδρόμηση και οι μετρικές αξιολόγησης για όλα τα μοντέλα έχουν υλοποιηθεί χρησιμοποιώντας τη βιβλιοθήκη `scikit-learn` [Ped+11]. Τα νευρωνικά δίκτυα έχουν υλοποιηθεί με τη χρήση `PyTorch` [Pas+19].

0.3.2 Μετρικές Αξιολόγησης

Προκειμένου να αξιολογήσουμε την απόδοση της αναγνώρισης ανθρωπίνων δράσεων με πολλές και πολλαπλές ετικέτες, *in-the-wild*, εφαρμόζουμε ένα σύστημα 5-fold cross validation, με 12 χρήστες σε κάθε fold. Κρατώντας σε κάθε επανάληψη το ένα από τα πέντε folds ως test set, καταλήγουμε να έχουμε 48 χρήστες στο σύνολο εκπαίδευσης και 12 χρήστες στο σύνολο δοκιμής σε κάθε επανάληψη. Κατά τη διαδικασία cross validation για κάθε επανάληψη: κρατάμε το επιλεγμένο fold για να χρησιμοποιηθεί ως σύνολο δοκιμής, εκπαιδεύουμε έναν ταξινομητή στα υπόλοιπα τέσσερα folds, και έπειτα κάνουμε infernece στο σύνολο δοκιμής.

Επαναλαμβάνουμε αυτή τη διαδικασία πέντε φορές συνολικά, έτσι ώστε κάθε fold να είναι το test set σε μία επανάληψη. Με αυτόν τον τρόπο συλλέγουμε τις προβλέψεις ετικετών του μοντέλου μας, για ολόκληρο το σύνολο δεδομένων ExtraSensory. Για κάθε fold και για κάθε ετικέτα, μετράμε το πλήθος των Αληθώς Θετικών (True Positives – TP), των Αληθώς Αρνητικών (True Negatives – TN), των Ψευδώς Θετικών (False Positives – FP) και των Ψευδώς Αρνητικών (False Negatives – FN) των αποτελεσμάτων της πρόβλεψης στο test set. Οι αριθμοί των TP, TN, FP και FN αθροίζονται για τα 5 folds και υπολογίζονται οι ακόλουθες μετρικές:

- *Accuracy* είναι το ποσοστό των σωστά ταξινομημένων δειγμάτων επί του συνόλου των δειγμάτων.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{0.3.1}$$

- *Ακρίβεια (Precision)* είναι το ποσοστό των σωστά ταξινομημένων δειγμάτων από τα δείγματα που ταξινομήθηκαν ως θετικά:

$$Precision = \frac{TP}{TP + FP} \tag{0.3.2}$$

- *True Positive Rate (TPR)*, που ονομάζεται επίσης *Ευαισθησία (Sensitivity)* ή *Ανάκληση (Recall)*, είναι το ποσοστό των θετικών δειγμάτων που ταξινομήθηκαν σωστά ως θετικά:

$$Sensitivity = Recall = TPR = \frac{TP}{TP + FN} \tag{0.3.3}$$

- *True Negative Rate (TNR)*, που ονομάζεται επίσης *Ειδικότητα (Specificity)*, είναι το ποσοστό των αρνητικών δειγμάτων που ταξινομήθηκαν σωστά ως αρνητικά:

$$Specificity = TNR = \frac{TN}{TN + FP} \tag{0.3.4}$$

- *F1-score* (F1) είναι ο αρμονικός μέσος όρος της Ακρίβειας και της Ανάκλησης:

$$F1 = \frac{2 * TPR * Prec}{TPR + Prec} \tag{0.3.5}$$

- *Balanced Accuracy (BA)* είναι ο μέσος όρος της Ευαισθησίας και της Ειδικότητας:

$$BA = \frac{TPR + TNR}{2} \tag{0.3.6}$$

0.4 Πειράματα και Αποτελέσματα

Στον Πίνακα 2, έχουμε συγκεντρώσει τις υψηλότερες τιμές των μετρικών απόδοσης που λάβαμε για καθένα από τα μοντέλα που υλοποιήσαμε και δοκιμάσαμε σε αυτή τη διπλωματική εργασία, για να συνοψίσουμε τα αποτελέσματά μας και να εξάγουμε χρήσιμες παρατηρήσεις.

Συγκριτική επισκόπηση των μέσων μετρικών αξιολόγησης για όλες τις ετικέτες χρησιμοποιώντας για κάθε μοντέλο τις βέλτιστες υπερπαραμέτρους βάσει των πειραμάτων μας

Είσοδος	**Μοντελοποίηση χρονοσειρών**	**Μοντέλο**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
		Τυχαίος ταξινομητής	0.500	0.110	0.500	0.500	0.137	0.500
		Ταξινομητής πλειοψηφικής κλάσης	0.915	`NaN`	0.040	0.958	0.037	0.499
Εξαχθέντα χαρακτηριστικά	Ένα δείγμα (λεπτό)	LR [VWL18]	0.832	-	0.597	0.838	-	0.718
		LR	0.839	0.246	0.612	0.844	0.314	0.728
		MLP [VWL18]	0.773	-	0.773	0.773	-	0.773
		MLP	0.786	0.228	0.757	0.786	0.298	0.772
Εξαχθέντα χαρακτηριστικά	Ακολουθία δειγμάτων (λεπτών)	BiLSTM (last output)	0.813	0.243	0.753	0.814	0.316	0.784
		BiLSTM (all outputs)	0.810	0.241	0.761	0.811	0.314	0.786
		Self-Attention & BiLSTM	0.818	0.248	0.756	0.819	0.323	0.788
		BiLSTM & Cross-Attention	0.800	0.238	0.767	0.801	0.309	0.784
Δεδομένα αισθητήρων	Ένα δείγμα (λεπτό)	CNN	0.782	0.228	0.762	0.781	0.296	0.772
		CNN & Transfomer	0.792	0.229	0.759	0.790	0.300	0.774
Δεδομένα αισθητήρων	Ακολουθία δειγμάτων (λεπτών)	CNN + BiLSTM	0.819	0.241	0.728	0.821	0.313	0.775

Table 2: Επισκόπηση των μέσων μετρικών απόδοσης για όλες τις ετικέτες, για κάθε μοντέλο

Συμπεριλαμβάνουμε δύο τετριμμένους ταξινομητές για να αποδείξουμε ότι τα μοντέλα που κατασκευάσαμε είναι σημαντικά καλύτερα από έναν τυχαίο ταξινομητή ή έναν ταξινομητή πλειοψηφικής κλάσης. Περιλαμβάνουμε

επίσης τα δύο baseline μοντέλα που προτάθηκαν από τους ερευνητές που δημιούργησαν το σύνολο δεδομένων ExtraSensory [VEL17; VWL18], τα οποία είναι άμεσα συγκρίσιμα με τα μοντέλα που αναπτύξαμε, καθώς στα πειράματά μας χρησιμοποιούμε το ίδιο υποσύνολο των δεδομένων, το ίδιο σχήμα διαχωρισμού και cross-validation του συνόλου δεδομένων και το ίδιο σύνολο 51 ετικετών.

Στον Πίνακα 4.4 παραθέτουμε όλες τις δημοσιεύσεις που προτείνουν μοντέλα αναγνώρισης ανθρωπίνων δράσεων χρησιμοποιώντας το ίδιο σύνολο δεδομένων. Ωστόσο, εξ όσων γνωρίζουμε, καμία από αυτές τις δημοσιεύσεις δεν περιλαμβάνει πειράματα χρησιμοποιώντας αρχιτεκτονικές Μηχανικής Μάθησης και Βαθιάς Μάθησης που να είναι άμεσα συγκρίσιμα με τη δικά μας (για να συγκριθούν δύο αρχιτεκτονικές αυτές καθ' αυτές προϋπόθεση είναι να χρησιμοποιείται το ίδιο υποσύνολο των δεδομένων και το ίδιο σχήμα διαχωρισμού και cross-validation ή το ίδιο test set γενικότερα, και το ίδιο σύνολο ετικετών που θέλουμε να προβλέψουμε) και επίσης να παράγει αποτελέσματα παρόμοια ή καλύτερα από τα baseline αποτελέσματα. Σε πολλές από αυτές τις εργασίες, χρησιμοποιείται ένα διαφορετικό, μικρότερο test set με λίγους χρήστες, ενώ στα δικά μας πειράματα κάθε χρήστης περιλαμβάνεται στο test set ακριβώς σε μία από τις πέντε επαναλήψεις του five-fold cross validation, και έτσι για να εξάγουμε τα τελικά αποτελέσματα και τις μετρικές χρησιμοποιούμε τις προβλέψεις του μοντέλου για όλους τους χρήστες, από τις επιμέρους εκπαιδεύσεις του μοντέλου. Επίσης, σε πολλές από τις εργασίες, η διερευνάται η αναγνώριση ανθρωπίνων δράσεων όπου χρησιμοποιείται ένα πολύ μικρότερο υποσύνολο ετικετών-στόχων.

Στον Πίνακα 2, έχουμε συμπεριλάβει τις μετρικές απόδοσης που προέκυψαν από τα ακόλουθα μοντέλα, όταν εκπαιδεύτηκαν χρησιμοποιώντας τις υπερπαραμέτρους που προσδιορίστηκαν ως βέλτιστες μέσω των πειραμάτων και των δοκιμών μας. Τα αποτελέσματα παρουσιάζονται ομαδοποιημένα ανάλογα με τον τύπο της εισόδου και ανάλογα με το αν μοντελοποιούμε ένα μόνο δείγμα ή μια ακολουθία δειγμάτων.

0.4.1 Baselines

Αρχικά, χρησιμοποιήσαμε τα εξαχθέντα χαρακτηριστικά που παρέχονται στο σύνολο δεδομένων ExtraSensory για κάθε δείγμα και μοντελοποιήσαμε ένα μόνο δείγμα προκειμένου να προβλέψουμε τις ετικέτες δράσεων και πλαισίου του, και αναπαραγάγαμε τα baseline μοντέλα: λογιστική παλινδρόμηση για κάθε ετικέτα ξεχωριστά, και ένα νευρωνικό δίκτυο τύπου multilayer perceptron το οποίο συμβολίζουμε ως MLP(16, 16) που περιλαμβάνει δύο κρυφά επίπεδα των 16 κόμβων το καθένα, και ένα επίπεδο εξόδου με 51 κόμβους που αντιστοιχούν στις 51 ετικέτες (Σχήμα 0.4.1).

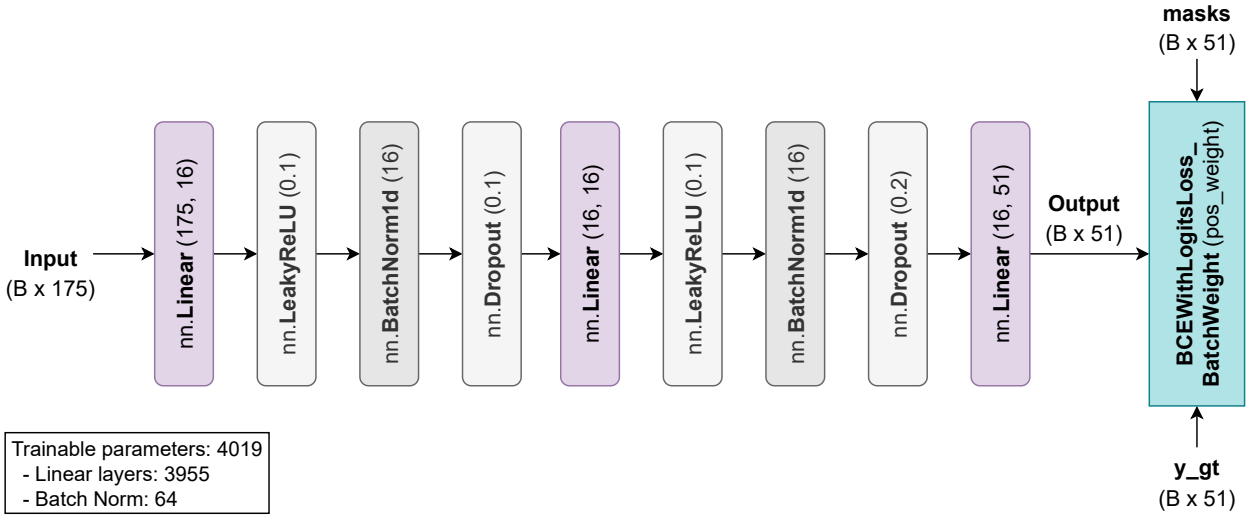
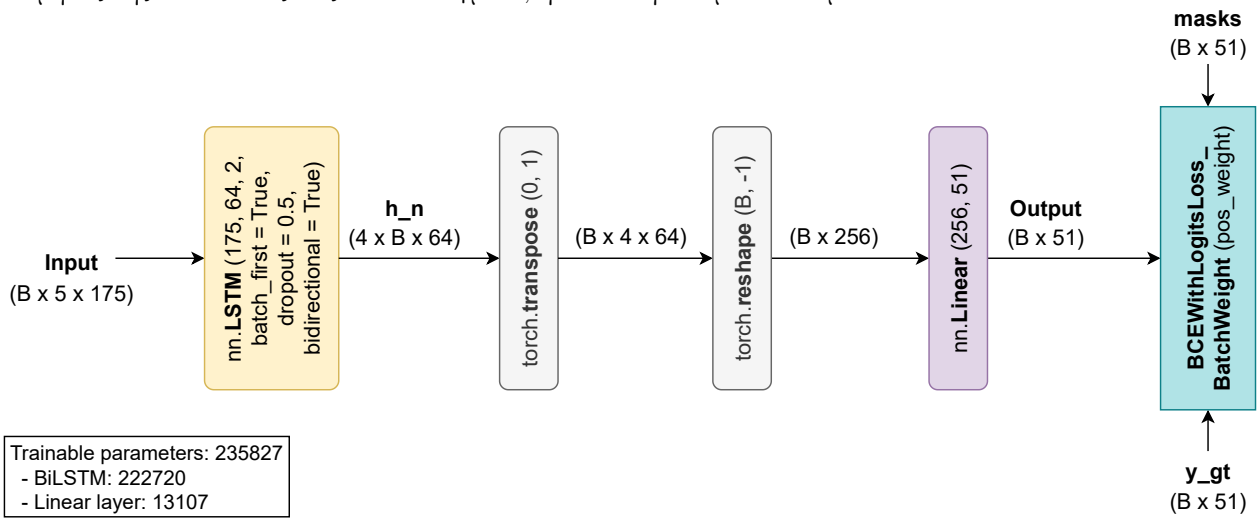


Figure 0.4.1: Το μοντέλο MLP(16, 16) που χρησιμοποιήθηκε ως baseline

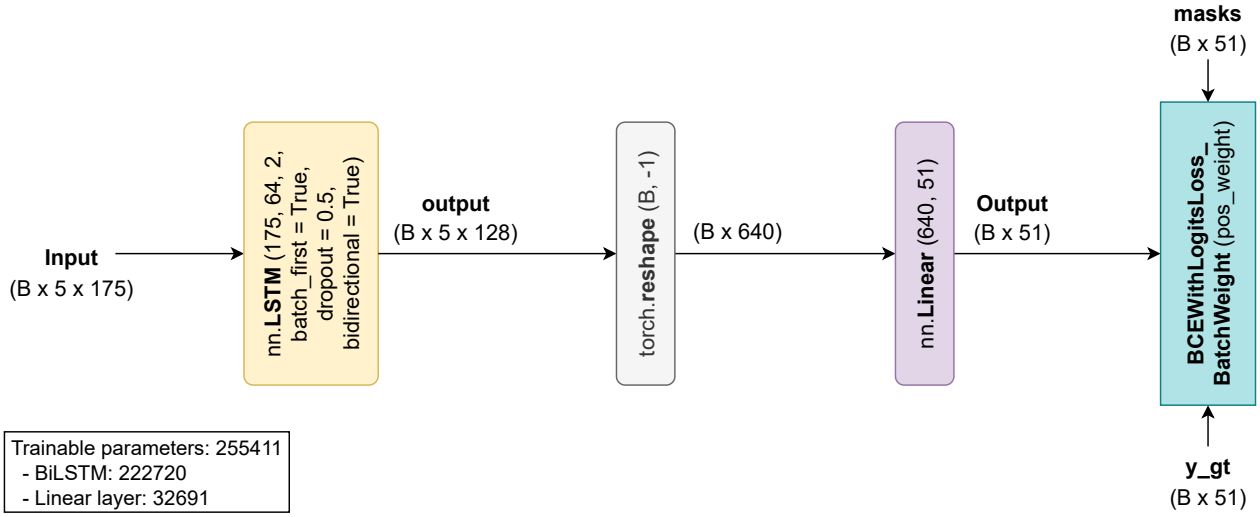
0.4.2 Αμφίδρομο LSTM

Στη συνέχεια, συνεχίσαμε να χρησιμοποιούμε τα εξαχθέντα χαρακτηριστικά, αλλά τώρα μοντελοποιούμε μια ακολουθία δειγμάτων, προκειμένου να προβλέψουμε τις ετικέτες του τελευταίου δείγματος παρέχοντας το πλαίσιο του άμεσου παρελθόντος. Χρησιμοποιήσαμε ένα BiLSTM δύο επιπέδων για να μοντελοποιήσουμε την ακολουθία δειγμάτων εισόδου και δοκιμάσαμε δύο μοντέλα, ένα χρησιμοποιώντας μόνο τις τελικές κρυφές καταστάσεις των δύο επιπέδων του BiLSTM (Σχήμα 0.4.2a) και ένα χρησιμοποιώντας τις εξόδους του τελευταίου επιπέδου του

BiLSTM για όλα τα βήματα της χρονικής ακολουθίας (Σχήμα 0.4.2b). Λάβαμε τα καλύτερα αποτελέσματα και για τα δύο μοντέλα, όταν δώσαμε μια ακολουθία εισόδου μήκους `window_len = 5` και όταν χρησιμοποιήσαμε τα `hidden_size = 64` σε κάθε επίπεδο του BiLSTM. Παρατηρούμε ότι οι τιμές της μετρικής BA των μοντέλων, που είναι 0.784 και 0.786 αντίστοιχα, είναι βελτιωμένες σε σχέση με τις τιμές των baseline μοντέλων, πράγμα που σημαίνει ότι η τροφοδότηση του μοντέλου με μιας ακολουθία 5 δειγμάτων (λεπτών), συμπεριλαμβανομένου του τρέχοντος δείγματος για το οποίο θέλουμε να προβλέψουμε τις ετικέτες, είναι ωφέλιμη για το μοντέλο μας. Πειραματιστήκαμε επίσης με ακολουθίες δειγμάτων μεγαλύτερου μήκους και διαπιστώσαμε ότι καθώς αυξάνουμε το μήκος της ακολουθίας έως και 30 δείγματα, η απόδοση του μοντέλου μειώνεται.



(a) Μοντέλο BiLSTM από το οποίο χρησιμοποιούνται μόνο τα τελικά hidden states



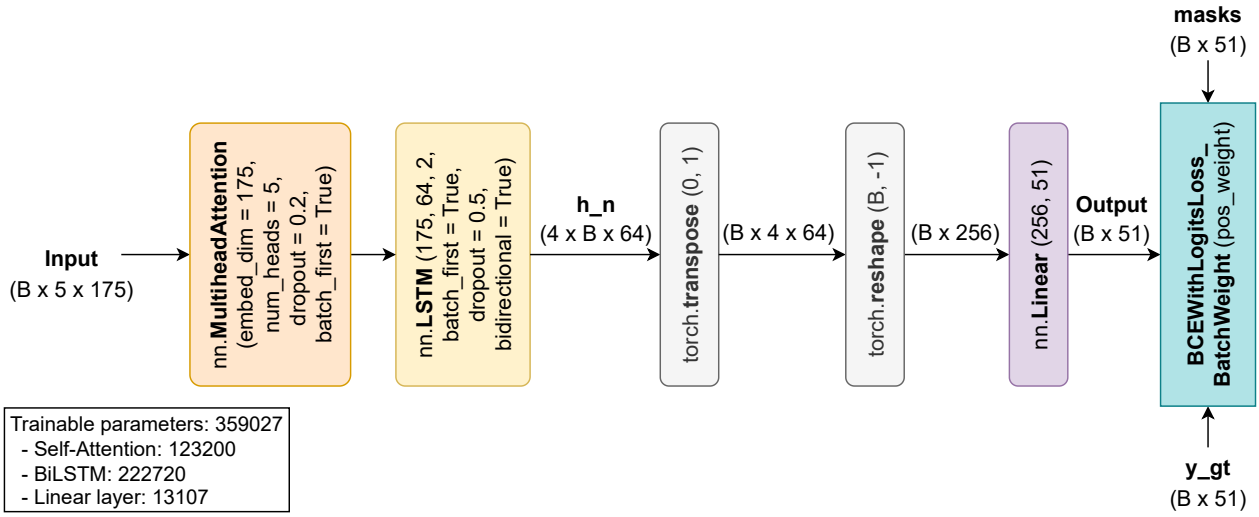
(b) Μοντέλο BiLSTM από το οποίο χρησιμοποιείται η έξοδος για όλα τα βήματα της χρονικής ακολουθίας

Figure 0.4.2: Τα μοντέλα που υλοποιήσαμε βασισμένα σε BiLSTM

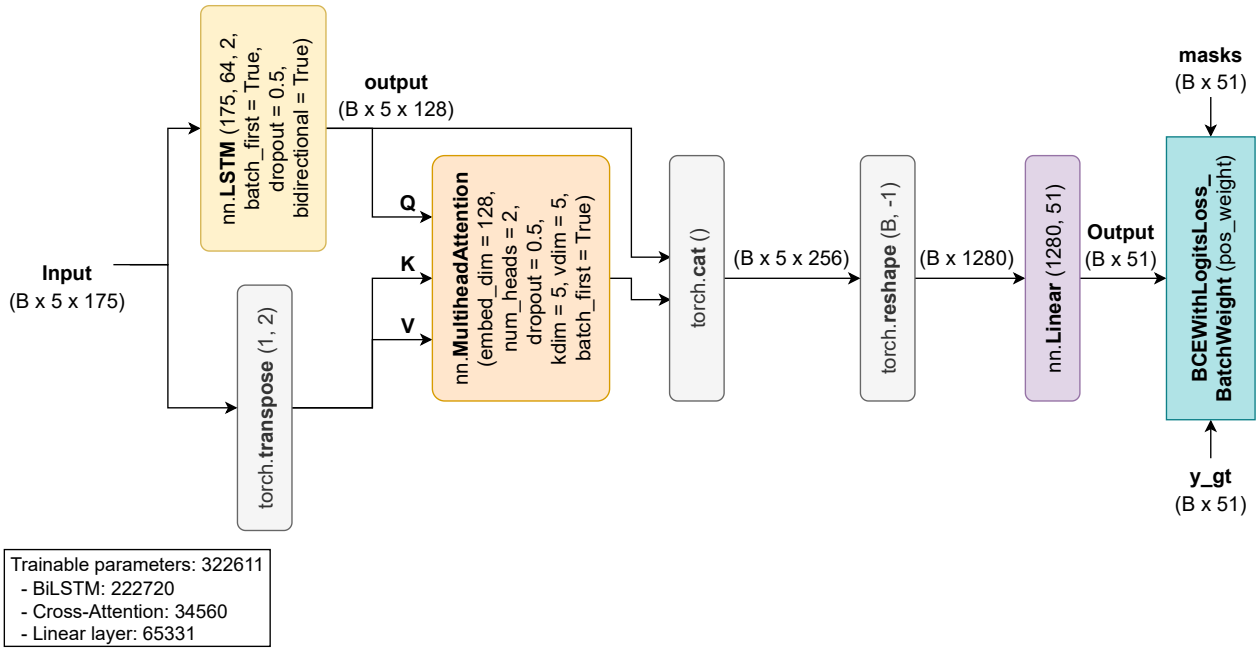
0.4.3 Αμφίδρομο LSTM με Μηχανισμό Προσοχής

Τα επόμενα πειράματά μας ήταν προσπάθειες περαιτέρω βελτίωσης του μοντέλου BiLSTM. Αρχικά, επαυξάνοντας το μοντέλο βασισμένο στο BiLSTM δύο επιπέδων με Μηχανισμό Αυτοπροσοχής που προηγείται του BiLSTM δύο επιπέδων (Σχήμα 0.4.3a), επιτυγχάνουμε περαιτέρω βελτίωση του BA, που φθάνει έως το 0.788, χρησιμοποιώντας μόνο τις τελικές κρυφές καταστάσεις του BiLSTM δύο επιπέδων και έναν Μηχανισμό Αυτοπροσοχής πολλαπλών κεφαλών με `num_heads = 5`. Ένα άλλο μοντέλο το οποίο υλοποιήσαμε περιλαμβάνει έναν Μηχανισμό Διασταυρούμενης Προσοχής πολλαπλών κεφαλών μετά το BiLSTM δύο επιπέδων (Σχήμα 0.4.3b). Η έξοδος του BiLSTM για όλα τα βήματα της χρονικής ακολουθίας χρησιμοποιείται για την παραγωγή

του query Q, ενώ τα χαρακτηριστικά εισόδου χρησιμοποιούνται για την παραγωγή του key K και του value V του Μηχανισμού Διασταυρούμενης Προσοχής. Αυτό το μοντέλο έχει παρόμοια απόδοση με το απλό μοντέλο BiLSTM με BA 0.784, αλλά το χρησιμοποιούμε για να οπτικοποιήσουμε και να μελετήσουμε τα βάρη του Μηχανισμού Διασταυρούμενης Προσοχής κατά το inference, για να διερευνήσουμε την ερμηνευσιμότητα του μοντέλου.



(a) Μοντέλο BiLSTM επαυξημένο με Μηχανισμό Αυτοπροσοχής

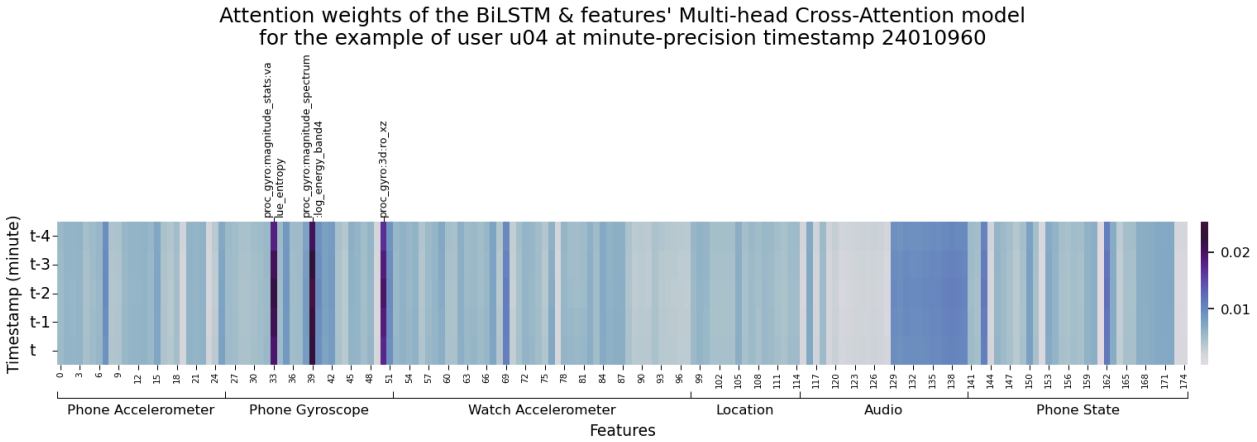


(b) Μοντέλο BiLSTM επαυξημένο με Μηχανισμό Διασταυρούμενης Προσοχής

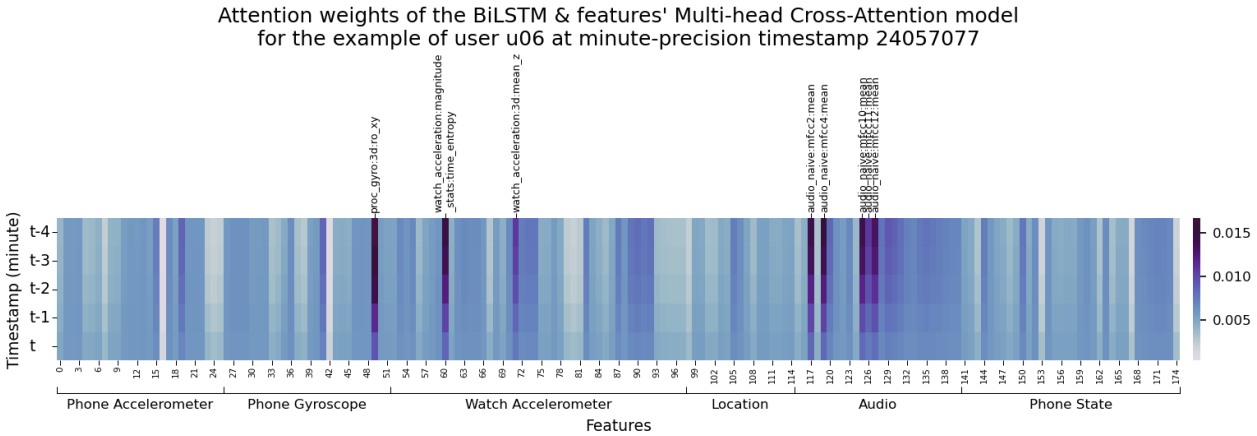
Figure 0.4.3: Τα μοντέλα που υλοποιήσαμε βασισμένα σε BiLSTM και επαυξημένα με Μηχανισμό Προσοχής

0.4.4 Ερμηνευσιμότητα

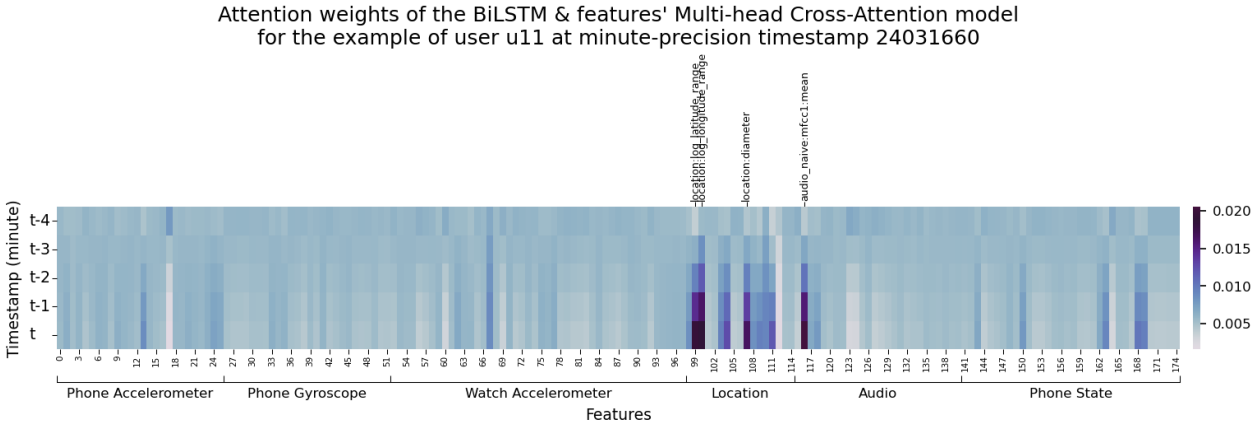
Στις Εικόνες 0.4.4, παραθέτουμε τα βάρη του Μηχανισμού Προσοχής κατά το inference για ενδεικτικά δείγματα, σε συνδυασμό με τις ground-truth ετικέτες και τις ετικέτες που προέβλεψε το μοντέλο μας. Στα plots αυτά, ο άξονας x περιέχει τα 175 χαρακτηριστικά εισόδου των βασικών αισθητήρων, ομαδοποιημένα ανά αισθητήρα. Ο άξονας y αντιστοιχεί στα 5 χρονικά βήματα της εξόδου του BiLSTM (το τρέχον λεπτό και τα προηγούμενα τέσσερα λεπτά). Το πιο σκούρο χρώμα αντιστοιχεί σε μεγαλύτερα βάρη. Τα βάρη του Μηχανισμού Διασταυρούμενης Προσοχής για περισσότερα δείγματα παρουσιάζονται εκτενώς στην Υποενότητα 5.6.3.



(a) Τα βάρη του Μηχανισμού Διασταυρούμενης Προσοχής για τον χρήστη u04 και το λεπτό t24010960
Πραγματικές ετικέτες: Sitting, Indoors, At home, Computer work, Phone on table
Ετικέτες που προβλέφθηκαν: Sitting, Indoors, At home, Surfing the internet, Computer work, Eating, Phone on table



(b) Τα βάρη του Μηχανισμού Διασταυρούμενης Προσοχής για τον χρήστη u06 και το λεπτό t24057077
Πραγματικές ετικέτες: Lying down, Sleeping, Indoors, At home, Phone on table
Ετικέτες που προβλέφθηκαν: Lying down, Sleeping, Indoors, At home, Phone on table



(c) Τα βάρη του Μηχανισμού Διασταυρούμενης Προσοχής για τον χρήστη u11 και το λεπτό t24031660
Πραγματικές ετικέτες: Sitting, In a car
Ετικέτες που προβλέφθηκαν: Sitting, Outside, In a car, On a bus, Drive - Driver, Drive - Passenger, Phone in pocket, Shopping, At a party, At the beach, Phone in hand, Phone in bag, With friends

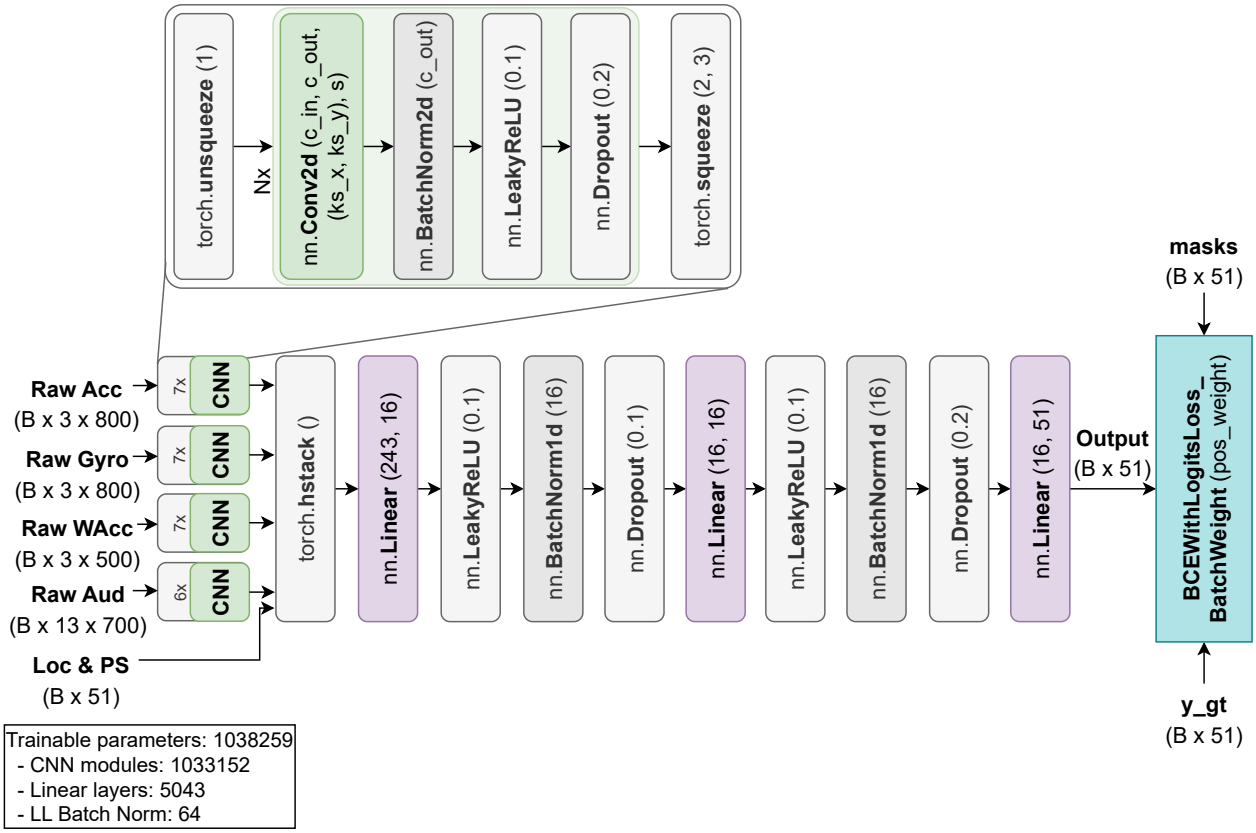
Figure 0.4.4: Τα βάρη του Μηχανισμού Διασταυρούμενης Προσοχής

0.4.5 Εξαγωγή Χαρακτηριστικών με Συνελικτικά Νευρωνικά Δίκτυα

Το επόμενό μας βήμα ήταν να αρχίσουμε να χρησιμοποιούμε τις ανεπεξέργαστες μετρήσεις των αισθητήρων για κάθε δείγμα, αντί για τα εξαχθέντα χαρακτηριστικά. Προτείνουμε δύο αρχιτεκτονικές που βασίζονται στη Βαθιά Μάθηση για την εξαγωγή χαρακτηριστικών από ανεπεξέργαστα δεδομένα αισθητήρων όσον αφορά στις χρονοσειρές του επιταχυνσιόμετρου, γυροσκοπίου και των MFCC από το έξυπνο κινητό και του επιταχυνσιόμετρου του έξυπνου ρολογιού.

Η πρώτη αρχιτεκτονική (Σχήμα 0.4.5a) είναι βασισμένη σε πολλά επίπεδα Συνελικτικών Νευρωνικών Δικτύων για την επεξεργασία της χρονοσειράς από κάθε αισθητήρα. Καθένα από τα επίπεδα περιλαμβάνει: 2D/1D Conv layer, leaky ReLU, Batch Normalization και Dropout. Η δεύτερη αρχιτεκτονική (Σχήμα 0.4.5b) είναι βασισμένη σε Συνελικτικά Νευρωνικά Δίκτυα σε συνδυασμό με επίπεδα Transformer Encoder για την επεξεργασία των χρονοσειρών κάθε αισθητήρα, και περιλαμβάνει τέσσερα επίπεδα Συνελικτικών Νευρωνικών Δικτύων ακολουθούμενα από δύο επίπεδα Transformer Encoder. Και στις δύο περιπτώσεις, ένα MLP(16, 16) λαμβάνει ως είσοδο τις αναπαραστάσεις βαθιάς μάθησης που προκύπτουν για να προβλέψει τις ετικέτες δράσεων και πλαισίου.

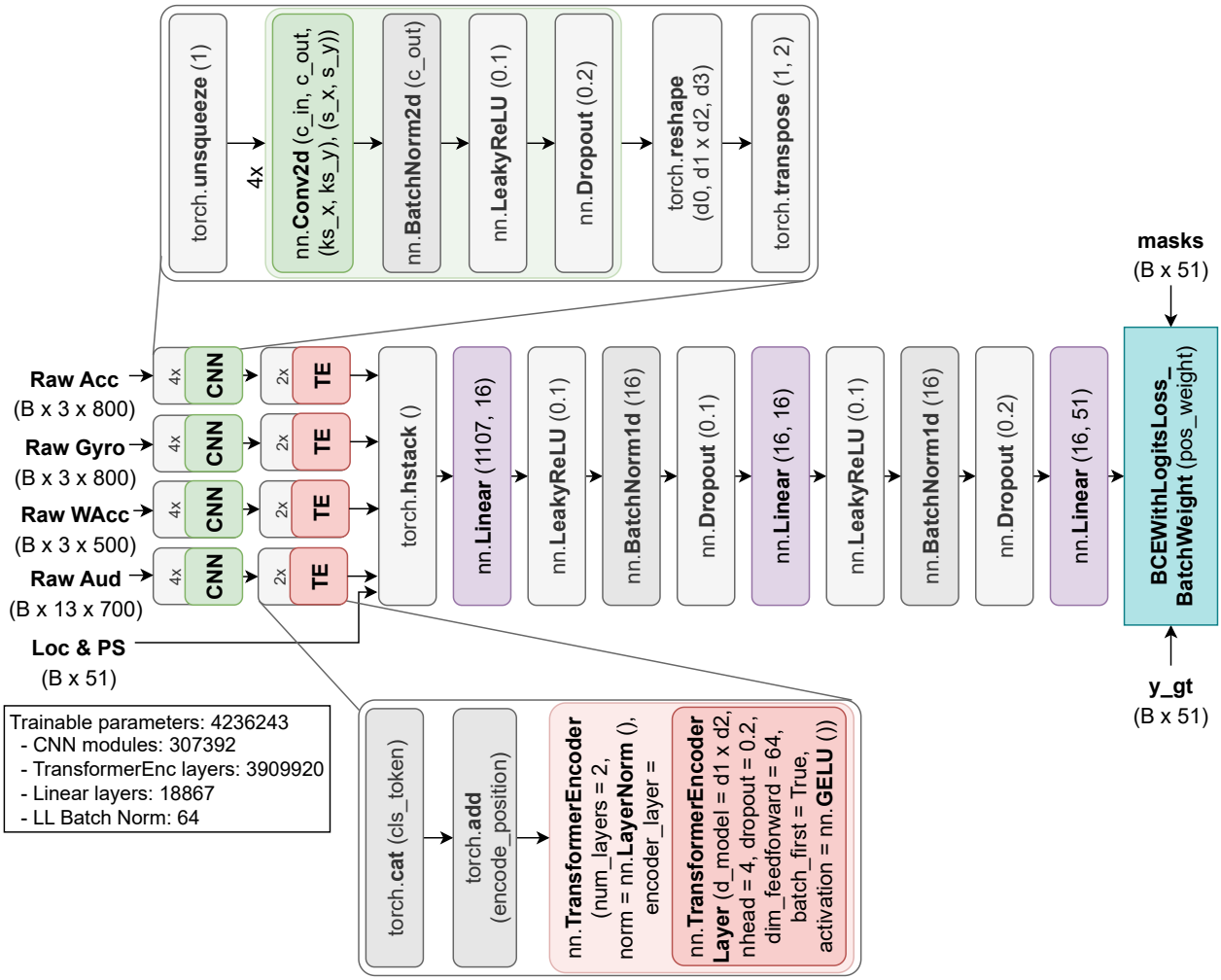
Παρόλο που αυτές οι αρχιτεκτονικές εξαγωγής χαρακτηριστικών παρήγαγαν τα καλύτερα αποτελέσματα μεταξύ όλων των αρχιτεκτονικών που δοκιμάσαμε για την εξαγωγή χαρακτηριστικών με βάση τη βαθιά μάθηση σε αυτό το σύνολο δεδομένων, εξακολουθούν να μην ξεπερνούν τα αποτελέσματα του baseline MLP(16, 16) που χρησιμοποιεί ως είσοδο τα προ-εξαχθέντα χαρακτηριστικά.



(a) Μοντέλο βασισμένο σε Συνελικτικά Νευρωνικά Δίκτυα

0.4.6 Συνελικτικά Νευρωνικά Δίκτυα και Αμφίδρομο LSTM

Το τελευταίο πείραμα στα πλαίσια της παρούσας διπλωματικής εργασίας βασίστηκε και πάλι στη χρήση των ανεπεξέργαστων μετρήσεων από τους αισθητήρες, αλλά τώρα για μια ακολουθία δειγμάτων. Χρησιμοποιήθηκε μια αρχιτεκτονική Χρονικά Κατανεμημένων Συνελικτικών Νευρωνικών Δικτύων (Time-Distributed CNN) για την εξαγωγή χαρακτηριστικών για όλα τα δείγματα της ακολουθίας εισόδου και έπειτα μοντελοποιήσαμε την



(b) Μοντέλο βασισμένο σε Συνελικτικά Νευρωνικά Δίκτυα και στρώματα Transformer Encoder

Figure 0.4.5: Μοντέλα για εξαγωγή χαρακτηριστικών μέσω Συνελικτικών Νευρωνικών Δικτύων

ακολουθία των εξαχθέντων αναπαραστάσεων με ένα BiLSTM δύο επιπέδων (Σχήμα 0.4.6), όπως και στα προηγούμενα πειράματά μας τα οποία περιείχαν μοντελοποίηση ακολουθίας διαδοχικών δειγμάτων. Ωστόσο, σε αυτή την περίπτωση, δεν έχουμε καμία ουσιαστική βελτίωση στην προκύπτουσα μετρική BA που έλαβε μέγιστη τιμή 0.775 στα πειράματά μας, σε σύγκριση με την τιμή 0.772 της μετρικής BA του μοντέλου βασισμένου στα Συνελικτικά Νευρωνικά Δίκτυα που μοντελοποιεί ένα μόνο δείγμα. Και πάλι, πειραματιστήκαμε εκτενώς με τις υπερπαραμέτρους του μοντέλου μας και εκπαιδεύσαμε δεκάδες μοντέλα, αλλά δεν καταφέραμε να παραγάγουμε καλύτερα αποτελέσματα από τα αποτελέσματα που παρουσιάζονται εδώ.

0.4.7 Συζήτηση

Παρατηρούμε ότι, όταν χρησιμοποιήσαμε τα ανεπεξέργαστα δεδομένα από τους αισθητήρες και τις αρχιτεκτονικές εξαγωγής χαρακτηριστικών μέσω Βαθιάς Μάθησης που ενσωματώθηκαν στα μοντέλα μας, δεν καταφέραμε να δημιουργήσουμε μοντέλα που να υπερτερούν σημαντικά έναντι του baseline MLP(16, 16). Αυτή η δυσκολία μπορεί ενδεχομένως να αποδοθεί σε τουλάχιστον δύο παράγοντες που μπορούμε να σκεφτούμε. Ο πρώτος παράγοντας είναι ότι είναι πολύ πιο κουραστική, και ως εκ τούτου, πολύ πιο δύσκολη, η λεπτομερής ρύθμιση των υπερπαραμέτρων των βαθιών νευρωνικών μοντέλων που περιέχουν πολυάριθμα επίπεδα, το καθένα με πολλές ρυθμίσιμες υπερπαραμέτρους. Η βελτιστοποίηση αυτών των υπερπαραμέτρων από το μηδέν είναι πολύ χρονοβόρα και απαιτεί πολλούς πόρους, και δεν είναι προφανές πώς μπορεί να γίνει με έξυπνο και αποτελεσματικό τρόπο. Έτσι, ένας λόγος που δεν έχουμε πολύ βελτιωμένα αποτελέσματα μπορεί να είναι ότι, παρόλο που κάναμε μια εκτεταμένη αναζήτηση για να ρυθμίσουμε τις τιμές των υπερπαραμέτρων του μοντέλου, μπορεί να μην καταφέραμε

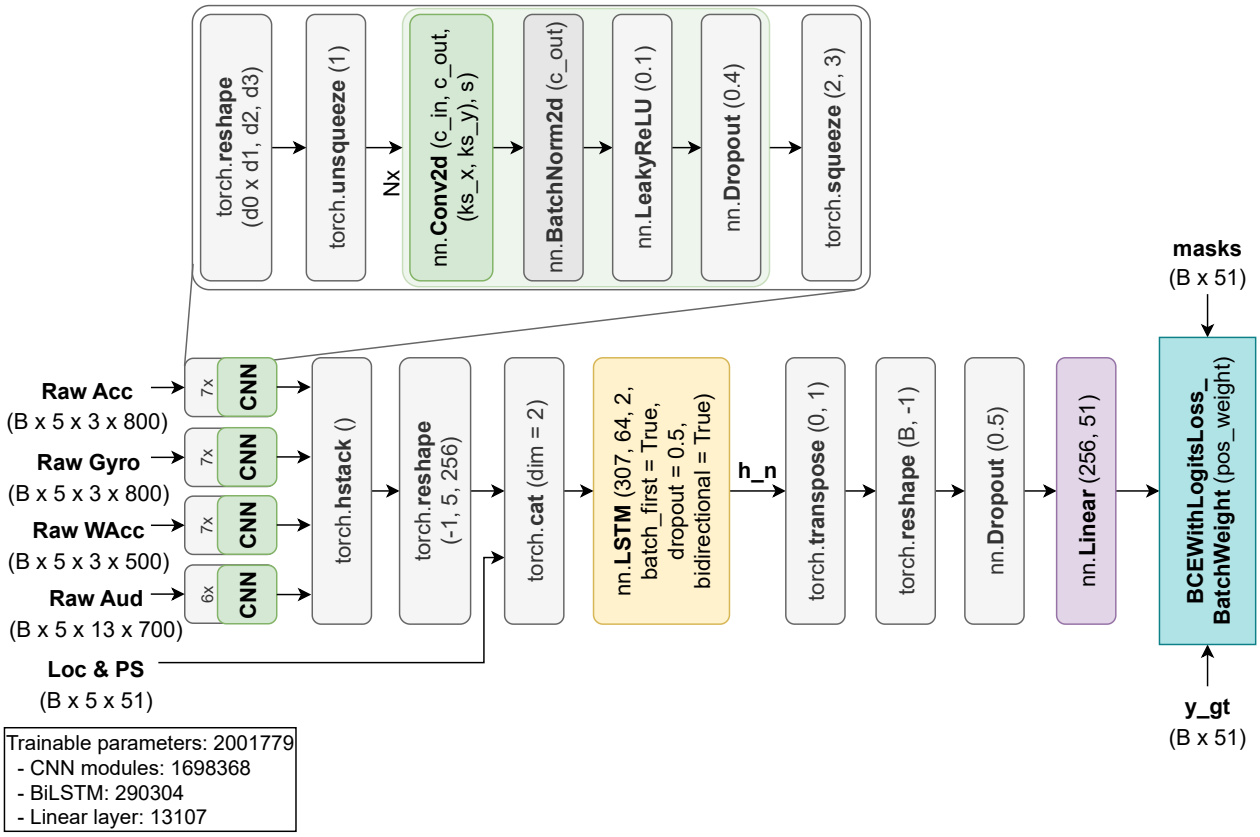


Figure 0.4.6: Το μοντέλο που συνδυάζει εξαγωγή χαρακτηριστικών μέσω Συνελικτικών Νευρωνικών Δικτύων και μοντελοποίηση μιας ακολουθίας δειγμάτων με BiLSTM

να βρούμε τις καλύτερες τιμές των υπερπαραμέτρων για τον συγκεκριμένο συνδυασμό μοντέλου και συνόλου δεδομένων. Στα μοντέλα που χρησιμοποιούσαν ως είσοδο τα ήδη εξαχθέντα χαρακτηριστικά, όπου ο αριθμός των υπερπαραμέτρων που έπρεπε να βελτιστοποιηθούν ήταν τάξεις μικρότερος, ήταν πιο απλή η ρύθμισή τους και καταφέραμε να παράγουμε καλύτερα αποτελέσματα πολύ πιο εύκολα.

Ο δεύτερος παράγοντας είναι ότι υπάρχει ένα δίλημμα σχετικά με το αν τα hand-crafted χαρακτηριστικά ή οι βαθιές νευρωνικές αναπαραστάσεις είναι καταλληλότερες για την αναγνώριση ανθρωπίνων δράσεων, ειδικά όταν πρόκειται για δεδομένα που συλλέγονται εκτός εργαστηρίου, καθώς και σε περιπτώσεις όπου στο inference έχουμε χρήστες που δεν έχει "δει" το μοντέλο κατά την εκπαίδευση, για τους οποίους η απόσταση της κατανομής των δεδομένων τους από την κατανομή των δεδομένων εκπαίδευσης μπορεί να είναι μεγαλύτερη. Προηγούμενη έρευνα στο θέμα αυτό έχει διεξαχθεί από τους Bento *et al.* [Ben+22] και διαπιστώθηκε ότι κατά το inference σε out-of-domain σύνολα δεδομένων, τα μοντέλα που χρησιμοποιούν hand-crafted χαρακτηριστικά ως είσοδο τείνουν να αποδίδουν καλύτερα. Τα ευρήματά μας είναι σύμφωνα με αυτά τα αποτελέσματα. Επιπλέον, δεδομένου ότι χρησιμοποιούμε ένα σύνολο δεδομένων που περιλαμβάνει πολύ περισσότερες ετικέτες και είναι πολύ μη ισορροπημένο, για πολλές από τις ετικέτες έχουμε πολύ λίγα δείγματα και διαισθητικά σκεφτόμαστε ότι οι αναπαραστάσεις που εξάγονται από τα βαθιά νευρωνικά δίκτυα για αυτές τις ετικέτες δεν θα είναι robust. Δεδομένου ότι στα πειράματά μας όλες οι ετικέτες έχουν την ίδια βαρύτητα κατά τον υπολογισμό των μέσων μετρικών για όλες τις ετικέτες, ανεξάρτητα από τον αριθμό των δειγμάτων που έχουν επισημειωθεί από τους χρήστες με κάθε ετικέτα, είναι σημαντικό για το μοντέλο να προβλέπει τόσο τις κοινές όσο και τις σπάνιες ετικέτες με όσο το δυνατόν μεγαλύτερη ακρίβεια, ώστε να πετύχουμε καλύτερες μέσες μετρικές αναγνώρισης.

Συνολικά, παρατηρώντας τον Πίνακα 2, μπορούμε να δούμε ότι υπάρχει ένα trade-off μεταξύ Sensitivity και Specificity. Παρατηρούμε ότι παρόλο που τα μοντέλα που έχουν καλύτερα αποτελέσματα στο συγκεκριμένο πρόβλημα αναγνώρισης ανθρωπίνων δράσεων μπορεί να παράγουν υψηλότερες τιμές και στις δύο μετρικές, σε σύγκριση με τα baselines ή τα πιο αδύναμα μοντέλα. Όμως όταν εκπαιδεύουμε ένα συγκεκριμένο μοντέλο με διαφορετικές τιμές υπερπαραμέτρων ή όταν εκπαιδεύουμε παρόμοια μοντέλα, όταν το Sensitivity που προκύπτει

είναι υψηλότερο, το Specificity που προκύπτει είναι συνήθως χαμηλότερο. Επιπλέον, παρατηρούμε ότι για όλα τα μοντέλα, οι τιμές του Sensitivity δεν βελτιώνονται σημαντικά και κυμαίνονται σε τιμές έως και 0.773. Ωστόσο, οι τιμές του Specificity, ειδικά στις περιπτώσεις όπου χρησιμοποιούμε μοντέλα βασισμένα σε BiLSTM και ακολουθίες διαδοχικών δειγμάτων ως είσοδο, είναι αυξημένες και φτάνουν έως και 0.810-0.820, πράγμα που σημαίνει ότι κατά μέσο όρο οι Ψευδώς Θετικές προβλέψεις αυτών των μοντέλων μειώνονται σημαντικά, σε σύγκριση με το baseline MLP(16, 16). Σε γενικές γραμμές, δεδομένου ότι δοκιμάσαμε ένα ευρύ φάσμα μοντέλων και δεν μπορέσαμε να πετύχουμε BA μεγαλύτερη από μια τιμή περίπου 0.788, συμπεραίνουμε ότι είτε το πρόβλημα είναι εγγενώς δυσεπίλυτο εξαιτίας του θορύβου και των ατελειών του συνόλου δεδομένων και τα περιθώρια βελτίωσης είναι εξ ορισμού σχετικά μικρά, είτε απαιτείται μια ριζική αλλαγή στην προσέγγισή μας στο συγκεκριμένο πρόβλημα αναγνώρισης ανθρωπίνων δράσεων για να προκύψουν μεγαλύτερες βελτιώσεις στις μετρικές αναγνώρισης, είτε και τα δύο. Παρουσιάζουμε ιδέες για μελλοντική έρευνα στην ενότητα 0.5.2.

0.5 Επίλογος

0.5.1 Συνεισφορές

Στην παρούσα διπλωματική εργασία, πραγματοποιήσαμε μια εκτενή μελέτη πάνω στις αρχιτεκτονικές νευρωνικών δικτύων για την αναγνώριση ανθρωπίνων δράσεων χρησιμοποιώντας δεδομένα που συλλέχθηκαν σε μη ελεγχόμενες συνθήκες (*in-the-wild*) μέσω φορητών συσκευών. Χρησιμοποιήσαμε το σύνολο δεδομένων ExtraSensory, το οποίο περιλαμβάνει δεδομένα που συλλέχθηκαν από αισθητήρες έξυπνων τηλεφώνων και έξυπνων ρολογιών από 60 χρήστες με συνολικά πάνω από 300000 δείγματα (λεπτά), που συνοδεύονται από 51 ετικέτες δράσεων και πλαισίου, ενώ κάθε παράδειγμα είναι επισημειωμένο με πολλαπλές σχετικές ετικέτες. Έχουμε υλοποιήσει και εκπαιδεύσει πολυάριθμα μοντέλα αναγνώρισης ανθρωπίνων δράσεων γενικού σκοπού, και επίσης χωρίς τα μοντέλα να λαμβάνουν υπόψη την ταυτότητα του χρήστη, ενώ στο σύνολο δοκιμής για τα μοντέλα μας περιλαμβάνουμε μόνο χρήστες που δεν "βλέπει" το μοντέλο κατά την εκπαίδευση, γεγονός που καθιστά το έργο μας ακόμη πιο δύσκολο. Οι συνεισφορές της εργασίας μας συνοψίζονται ως ακολούθως:

- Έχουμε μελετήσει εκτενώς τη σχετική βιβλιογραφία για την αναγνώριση ανθρωπίνων δράσεων βάσει μη οπτικών αισθητήρων φορητών συσκευών και έχουμε παρουσιάσει μια διεξοδική βιβλιογραφική ανασκόπηση σχετικά με την τυποποιημένη διαδικασία αναγνώρισης ανθρωπίνων δράσεων, που περιλαμβάνει την προεπεξεργασία των δεδομένων που έχουν συλλεχθεί μέσω των αισθητήρων, την τμηματοποίησή τους, την εξαγωγή χαρακτηριστικών και την ταξινόμηση. Έχουμε καταγράψει τα διαθέσιμα, ανοικτά σύνολα δεδομένων και έχουμε συμπεριλάβει τις σημαντικότερες προκλήσεις και τα ανοικτά προβλήματα της αναγνώρισης ανθρωπίνων δράσεων. Έχουμε, επίσης, μελετήσει την αρχιτεκτονική και την εφαρμογή δημοφιλών μοντέλων βαθιάς μάθησης, συμπεριλαμβανομένων των τεχνητών νευρωνικών δικτύων (DNN), των συνελικτικών νευρωνικών δικτύων (CNN), των αναδρομικών νευρωνικών δικτύων (RNN) όπως τα LSTM και τα GRU, του μηχανισμού προσοχής και των Transformers, σε εργασίες αναγνώρισης ανθρωπίνων δράσεων.

- Έχουμε εντρυφήσει στην αναγνώριση ανθρωπίνων δράσεων *in-the-wild*, η οποία μεταφέρει τη διαδικασία συλλογής δεδομένων από τα προδιαγεγραμμένα σενάρια δράσεων και την ελεγχόμενη καταγραφή σε εργαστηριακό περιβάλλον, σε μη ελεγχόμενα περιβάλλοντα της πραγματικής ζωής. Έχουμε μελετήσει δεδομένα που έχουν συλλεχθεί κατά αυτόν τον τρόπο, όπως το σύνολο δεδομένων ExtraSensory, το οποίο είναι πολύ πλούσιο σε περιεχόμενο και ετικέτες και αντικατοπτρίζει τις δραστηριότητες της καθημερινής ζωής, αλλά η χρήση του για την ανάπτυξη μοντέλων αναγνώρισης ανθρωπίνων δράσεων έχει αυξημένη πολυπλοκότητα λόγω των εγγενών χαρακτηριστικών του, όπως η ακραία ανισορροπία στο πλήθος των ετικετών, η μεγάλες ελλείψεις δεδομένων από τους αισθητήρες, οι ελλείψεις, οι παραλείψεις και τα λάθη στη διαδικασία της επισημείωσης, ο θόρυβος στα δεδομένα από τους αισθητήρες, καθώς και η διαπροσωπική και ενδοπροσωπική μεταβλητότητα στην εκτέλεση των διάφορων δραστηριοτήτων της καθημερινότητας.

- Έχουμε ολοκληρώσει μια συστηματική και εκτεταμένη διερεύνηση μοντέλων Μηχανικής Μάθησης και Βαθιάς Μάθησης για γενικού σκοπού ανίχνευση ανθρωπίνων δράσεων, χαρακτηριζόμενη από πολλές και πολλαπλές ετικέτες, στηριζόμενη στη συλλογή δεδομένων *in-the-wild*. Στα πειράματά μας έχουμε χρησιμοποιήσει το σύνολο δεδομένων ExtraSensory και τις προαναφερθείσες 51 ετικέτες δράσεων και πλαισίου. Μια λεπτομερής επισκόπηση όλων των πειραμάτων και των αποτελεσμάτων μας είναι διαθέσιμη στην Ενότητα 0.4. Συνοπτικά, έχουμε αναπαραγάγει τα καλύτερα διαθέσιμα baseline μοντέλα, τα οποία χρησιμοποιούν τα ήδη εξαχθέντα χαρακτηριστικά του συνόλου δεδομένων, και έχουμε προτείνει μοντέλα

τα οποία σύμφωνα με τα πειράματά μας εμφανίζουν βελτιωμένη BA συγκριτικά με τα baseline. Αυτά τα μοντέλα βασίζονται σε ένα BiLSTM δύο επιπέδων, το οποίο δέχεται ως είσοδο μια ακολουθία διαδοχικών δειγμάτων, χρησιμοποιώντας και πάλι τα ήδη εξαχθέντα χαρακτηριστικά. Η προσθήκη ενός μηχανισμού αυτοπροσοχής πριν από το BiLSTM δύο επιπέδων βελτίωσε περαιτέρω τη μετρική BA. Πειραματιστήκαμε επίσης με τη χρήση των ανεπεξέργαστων μετρήσεων των αισθητήρων για την εξαγωγή χαρακτηριστικών μέσω πολλαπλών επιπέδων CNN ή CNN-Transformer που ενσωματώσαμε στο μοντέλο μας, για ένα μόνο δειγμα, ή σε συνδυασμό με ένα BiLSTM για τη μοντελοποίηση μιας ακολουθίας δειγμάτων. Ωστόσο, αυτά τα πειράματα δεν παρήγαγαν πολύ καλύτερα αποτελέσματα από τα baseline.

- Επιπλέον, προσπαθήσαμε να μελετήσουμε τον τρόπο λειτουργίας των μοντέλων μας, υλοποιώντας ένα μοντέλο συγκεκριμένα για αυτόν τον σκοπό. Επαυξήσαμε το μοντέλο BiLSTM δύο επιπέδων με ένα μηχανισμό διασταυρούμενης προσοχής και χρησιμοποιούμε τις εξόδους του BiLSTM για όλα τα βήματα της χρονικής ακολουθίας εισόδου για να παράγουμε τα queries του μηχανισμού προσοχής και τα χαρακτηριστικά της ακολουθίας εισόδου για να παράγουμε τα keys και τα values του μηχανισμού προσοχής. Κατά το inference του μοντέλου, αποθηκεύουμε τα βάρη του μηχανισμού διασταυρούμενης προσοχής για κάθε δείγμα και τις αντίστοιχες προβλέψεις του μοντέλου, ενώ έχουμε επίσης διαθέσιμες τις πραγματικές ετικέτες κάθε δείγματος που είχαν δοθεί κατά την επισημείωση από τον χρήστη. Στην υποενότητα 5.6.3, έχουμε απεικονίσει τα βάρη του μηχανισμού διασταυρούμενης προσοχής και έχουμε διερευνήσει διεξοδικά κατά πόσο μπορούν να παρέχουν χρήσιμες πληροφορίες σχετικά με το ποια χαρακτηριστικά έχουν μεγαλύτερη επίδραση στις προβλέψεις του μοντέλου για διάφορες ετικέτες, καθώς και κατά πόσο αυτές οι προβλέψεις έχουν νόημα ή αν το μοντέλο πολλές φορές παραπλανάται από ψευδείς συσχετίσεις μεταξύ των χαρακτηριστικών εισόδου και των ετικετών.

- Κατά τη διάρκεια της μελέτης και της διεξαγωγής των πειραμάτων για την παρούσα διπλωματική εργασία, δημιουργήσαμε το αποθετήριο `alexvioni/ExtraSensory-functionality` στο Github, το οποίο περιλαμβάνει και υποστηρίζει όλες τις επιμέρους λειτουργίες από τη φόρτωση και τη διαχείριση των δεδομένων, την εξερεύνηση και την οπτικοποίηση των χαρακτηριστικών μέχρι το σχεδιασμό και την υλοποίηση των μοντέλων μηχανικής μάθησης και βαθιάς μάθησης, και όλα τα απαιτούμενα τμήματα κώδικα για την εκπαίδευση, το inference και την αξιολόγησή τους, συμπεριλαμβανομένων των μετρικών και των διαγραμμάτων. Αυτό το αποθετήριο είναι ιδιωτικό ως τη στιγμή που γράφεται αυτή τη εργασία, αλλά ενδέχεται μελλοντικά να γίνει ανοιχτό στο κοινό, και αποτελεί ένα ready-to-use framework που μπορεί να χρησιμοποιηθεί σε ευρύ φάσμα εργασιών, επιτρέποντας τη γρήγορη ενσωμάτωση περισσότερων χαρακτηριστικών και μοντέλων και τη γρήγορη επέκταση σε περισσότερα σύνολα δεδομένων, τομείς ή εργασίες.

0.5.2 Μελλοντικές Προεκτάσεις

Συνοψίζουμε ακολούθως ορισμένες κατευθύνσεις για μελλοντική έρευνα με βάση τις γνώσεις που αποκτήθηκαν κατά τη μελέτη και τη διεξαγωγή πειραμάτων για την παρούσα εργασία:

- Όπως αναφέραμε προηγουμένως, στην εργασία μας υλοποιήσαμε και εκπαιδεύσαμε γενικού σκοπού μοντέλα αναγνώρισης ανθρωπίνων δράσεων, τα οποία μπορούν να χρησιμοποιηθούν σε οποιοδήποτε πρόβλημα αναγνώρισης ανθρωπίνων δράσεων πολλαπλών ετικετών, με αυθαίρετο αριθμό ετικετών, αφού δεν θέτουμε κανενός είδους περιορισμούς στην πρόβλεψή τους. Ωστόσο, σε πολλά προβλήματα αναγνώρισης ανθρωπίνων δράσεων μπορούμε να επωφεληθούμε από συγκεκριμένες συσχετίσεις μεταξύ ετικετών, όπως ταξινομήσεις των δράσεων ή των πλαισίων, ιεραρχικές σχέσεις μεταξύ ετικετών ή ετικέτες που είναι αμοιβαίως αποκλειόμενες, για να ρυθμίσουμε πιο αποτελεσματικά τη συνάρτηση κόστους που χρησιμοποιείται για την εκπαίδευση του μοντέλου, για να βελτιώσουμε την απόδοσή του. Για παράδειγμα, αν γνωρίζουμε ότι κάποιες ετικέτες είναι αμοιβαίως αποκλειόμενες, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση ενεργοποίησης Softmax, καθώς χρειάζεται να προβλέψουμε μόνο μία ετικέτα από τις αμοιβαίως αποκλειόμενες.

- Επιπροσθέτως, τα μοντέλα μας δεν λαμβάνουν υπόψη την ταυτότητα του χρήστη στον οποίο ανήκει το δείγμα εισόδου κατά την εκπαίδευση και κατά την πρόβλεψη, ενώ επίσης πάντα δοκιμάζουμε τα μοντέλα μας σε χρήστες που τα μοντέλα δεν έχουν δει κατά την εκπαίδευση, ώστε να διασφαλίσουμε ότι έχουν επαρκή δυνατότητα γενίκευσης. Ωστόσο, δεδομένου ότι κάθε χρήστης έχει τη δική του ρουτίνα και τις δικές του συνήθειες και ενδέχεται να εκτελεί δραστηριότητες με τον δικό του τρόπο, η εξατομίκευση του μοντέλου

για κάθε χρήστη θα μπορούσε να είναι επωφελής για το μοντέλο μας και ενδεχομένως να βελτιώσει την απόδοσή του. Λαμβάνοντας υπόψη ότι είναι δύσκολο να συλλέξουμε μεγάλο αριθμό επισημειωμένων δεδομένων ανθρωπίνων δράσεων από κάθε νέο χρήστη στον οποίο θέλουμε να προσαρμόσουμε το μοντέλο, ώστε να εκπαιδεύσουμε το μοντέλο από την αρχή χρησιμοποιώντας μόνο τα δικά του δεδομένα, μπορούμε να χρησιμοποιήσουμε μια μικρή ποσότητα επισημειωμένων δεδομένων για να προσαρμόσουμε πάνω στον συγκεκριμένο χρήστη ένα ήδη εκπαιδευμένο μοντέλο γενικού σκοπού. Αυτή η προσέγγιση συνδυάζει τη βελτιωμένη απόδοση και την ανάγκη για μικρή μόνο ποσότητα επισημειωμένων δεδομένων από κάθε νέο χρήστη, ενώ το μοντέλο που θα προκύψει θα είναι βελτιστοποιημένο για τον συγκεκριμένο χρήστη και διαθέσιμο στη φορητή συσκευή του.

- Μια άλλη ενδιαφέρουσα κατεύθυνση θα ήταν να διερευνηθεί αν μπορούμε να χρησιμοποιήσουμε μη επιβλεπόμενες ή ημι-επιβλεπόμενες μεθόδους, όπως η ομαδοποίηση (clustering), για να προσδιορίσουμε ποιες ετικέτες του συνόλου δεδομένων είναι σωστές και ποια δείγματα έχουν επισημειωθεί λανθασμένα εκ περιτροπής. Αυτή η διαδικασία θα μείωνε σημαντικά την ανάγκη για χειροκίνητο καθαρισμό του συνόλου δεδομένων και η εκπαίδευση των μοντέλων θα επωφελούνταν από ένα "καθαρότερο" ή καλύτερα επιμελημένο σύνολο δεδομένων. Επίσης, αντί να απορρίπτονται τα δείγματα ή οι ετικέτες που κρίθηκαν αναξιόπιστες, θα μπορούσαμε ίσως να προσαρμόσουμε τη συνάρτηση κόστους χρησιμοποιώντας βάρη ανάλογα με την εμπιστοσύνη που έχουμε αποδώσει σε κάθε ετικέτα για το συγκεκριμένο δειγμα.

- Η επόμενη ιδέα μας βασίζεται στο ότι, παρ'όλο που μετριάζουμε την επίδραση της σπανιότητας κάποιων ετικετών μέσω της χρήσης βαρών αντίστροφων με τη συχνότητα της κάθε ετικέτας στη συνάρτηση κόστους και καταφέρνουμε να καθοδηγήσουμε το μοντέλο να προβλέπει όλες τις ετικέτες και όχι μόνο τις πιο κοινές, η έλλειψη δειγμάτων που έχουν επισημειωθεί με τις σπάνιες ετικέτες έχει ως αποτέλεσμα συστήματα που δυσκολεύονται να μοντελοποιήσουν αυτές τις ετικέτες. Προτείνουμε την χρήση τεχνικών ή μηχανισμών που μπορούν να βοηθήσουν προς αυτή την κατεύθυνση, χρησιμοποιώντας περισσότερα επισημειωμένα σύνολα δεδομένων για την αύξηση των δειγμάτων που έχουν επισημειωθεί με σπάνιες ετικέτες ή αξιοποιώντας μη επισημειωμένα σύνολα δεδομένων με χρήση αυτοεπιβλεπόμενης μάθησης για την εξαγωγή αναπαραστάσεων από αυτά. Επιπλέον, θα μπορούσαμε να επεκτείνουμε το προαναφερθέν δίλημμα σχετικά με το αν τα hand-crafted στατιστικά χαρακτηριστικά ή οι βαθιές νευρωνικές αναπαραστάσεις είναι καταλληλότερες για την αναγνώριση ανθρωπίνων δράσεων σε χρήστες τους οποίους δεν έχει "δει" το μοντέλο κατά την εκπαίδευση, συμπεριλαμβάνοντας στη συζήτηση αναπαραστάσεις που εξάγονται από μοντέλα αυτοεπιβλεπόμενης μάθησης [Yua+23].

- Μια ακόμα πρόταση για μελλοντική διερεύνηση θα ήταν να εκμεταλλευτούμε το σύνολο δεδομένων ExtraSensory, που χαρακτηρίζεται από μεγάλη ποικιλία σε αισθητήρες, δεδομένα και επισημειώσεις, και να μεταφέρουμε τη γνώση σε μοντέλα που θα χρησιμοποιούν μόνο έναν αισθητήρα κατά τη διάρκεια του inference. Θα ήταν πολύ ενδιαφέρον να ελέγξουμε αν θα μπορούσαμε να δημιουργήσουμε και να εκπαιδεύσουμε μοντέλα που θα επιτύχουν παρόμοια απόδοση αναγνώρισης ανθρωπίνων δράσεων χρησιμοποιώντας λιγότερους αισθητήρες ή ακόμη και μόνο έναν αισθητήρα. Προς την ίδια κατεύθυνση, θα ήταν επίσης πολύ ενδιαφέρον να εξετάσουμε αν μπορούμε να εφαρμόσουμε ένα μοντέλο που εκπαιδεύτηκε στο σύνολο δεδομένων ExtraSensory, σε ένα πολύ διαφορετικό σύνολο δεδομένων από διαφορετικό domain, π.χ. στο σύνολο δεδομένων E-Prevention [Zla+22], και να διερευνήσουμε τις δυνατότητες αναγνώρισης ανθρωπίνων δράσεων σε σύνολα δεδομένων τα οποία δεν είναι κατά τη δημιουργία τους επισημειωμένα με ετικέτες δράσεων.

- Τέλος, προκειμένου το μοντέλο να χρησιμοποιηθεί σε πραγματικές εφαρμογές, απαιτείται το inference να μπορεί να γίνει σε πραγματικό χρόνο. Παρόλα αυτά, το μοντέλο μας χρησιμοποιεί καταγραφές διάρκειας περίπου 20 δευτερολέπτων προκειμένου να προβλέψει ετικέτες για κάθε λεπτό της καθημερινής ζωής. Μένει να διερευνηθεί αν το μοντέλο μπορεί να προβλέψει ετικέτες με παρόμοια ακρίβεια χρησιμοποιώντας καταγραφές πολύ μικρότερης διάρκειας από τους αισθητήρες ή να χρησιμοποιηθούν άλλες προσεγγίσεις για ταχύτερη αναγνώριση ανθρωπίνων δράσεων. Επιπλέον, η ακρίβεια του μοντέλου πρέπει να βελτιωθεί προκειμένου να χρησιμοποιηθεί απρόσκοπτα στην καθημερινή ζωή. Δεδομένου ότι είδαμε ότι είναι πολύ δύσκολο να βελτιωθεί η μετρική BA και ότι υπάρχει trade-off μεταξύ Sensitivity και Specificity, σε ορισμένες περιπτώσεις θα μπορούσαμε να επιλέξουμε να βελτιστοποιήσουμε το Sensitivity, αν μας ενδιαφέρει περισσότερο η εξάλειψη των Ψευδώς Αρνητικών, π.χ. σε μια εφαρμογή στον τομέα της υγείας, ή το Specificity, αν προτιμάμε να εξαλείψουμε τα Ψευδώς Θετικά.

Chapter 1

Introduction

1.1 Human Activity Recognition

The term *Human Activity Recognition (HAR)* – automatic recognition of physical activities – refers to the procedure of analyzing human body gesture or motion, using data retrieved from sensors, to determine the activity performed by the person [AT14; BBS14]. It is a research field characterized by increased task complexity, and it involves multiple scientific fields, most notably human-computer interaction, signal processing, computer vision, statistical analysis, and machine learning [AT14]. An HAR system might utilize supervised learning techniques, unsupervised learning techniques, or a combination of both.

HAR is very promising and has become enormously valuable in multiple application contexts, including healthcare (patient monitoring for clinical decision support systems), active assisted living (AAL) systems for smart homes, surveillance-based security, and tele-immersion (TI) applications [AT14; Che+12; RMM16]. A major objective of HAR is to provide information about human behavior that will let computing systems actively support human daily life activities, in key areas of need [BBS14]. If a system that correctly identifies human activities automatically is implemented, a variety of applications and services can take advantage of it, in order for example to monitor health status, to detect diseases or health emergencies [Llo+15; Agh+07], or to give advice on daily routine and lifestyle [Ali+15; NHC15].

HAR as a procedure can be divided in broad terms in four fundamental stages. These stages include, in sequential order:

1. to select and arrange fitting sensors in humans, objects and/or environments in order to effectively observe and track human behavior, along with state transitions in the surrounding environment
2. to store and process the data collected from sensors using suitable data analysis methods and/or knowledge representation techniques
3. to choose and/or develop appropriate machine learning algorithms, which can be applied to the features extracted from the processed data
4. to apply the selected algorithms in order to classify and recognize the performed activities that were captured by the sensors

Over the years, methodologies and tools for each of the above stages have been developed and implemented in abundance. In many cases, the selections of techniques for different stages are interdependent upon one another [Che+12; Que+15].

1.1.1 Notion of "activity"

By now, there is no ontological definition of the notion of "activity" that has gained universal approval. As follows, the most common approach to verge on this notion is to decompose it into granularity levels. According to the activity theory [Leo78; May+16], *activities* (e.g. "preparing a sandwich") hierarchically consist of *actions* (e.g. "enter the kitchen", "slice bread"), that again are composed out of *operations* (e.g. "grab the door handle", "grab the knife"), atomic steps which sequentially implement the action. In this way, actions can be seen as aggregations of operations, and activities can be seen as aggregations of actions [RMM16].

Another approach to the notion of "activity" can be made from a specialization viewpoint, by elaborating on a general activity, using more specific activities which are considered to be subcategories of the aforementioned activity. For example, "preparing meals" can be specialized into "preparing hot meal" and "preparing cold meal", while "preparing cold meal" can be specialized into "preparing a sandwich", "preparing a salad" and "preparing a cold dessert" [NHC15].

Taxonomies

Different activity taxonomies have been proposed over the years, largely depending on the domain of application and the respective living scenarios. Regarding the AAL domain, Katz [Kat+70; Kat83] has proposed a so-called taxonomy for the activities of daily living (ADLs) and developed the Index of Independence of ADLs. According to Katz [KA76] and Lawton [LB69], the ADLs can be subdivided in two categories:

- Basic ADLs (BADLs), necessary personal self-care activities, such as dressing, eating, drinking, using toilet, bathing, grooming, and functional mobility
- Instrumental ADLs (IADLs), activities not indispensable in the strict sense, but fundamental for a person's independent life in a community, such as preparing food, housekeeping, managing money, shopping, taking medication, using telephone, and transportation

The capability to perform ADLs without assistance can be used to assess a person's level of independent living [NHC15]. Moreover, the term of ambulatory activities describes another type of activities, the spectrum of postures and movements of the person. These activities can be divided in three basic categories:

- Stationary activities, which describe a posture, such as standing, sitting or lying
- Transitional activities, which describe the transition between two states, such as sit-to-stand, stand-to-walk or stand-to-lie
- Dynamic activities, which describe movements and dynamic actions, such as walking, jogging, or running

By tracking ambulatory activities, it is possible to determine a person's physical activity level, in order to promote a healthy lifestyle, to detect related health conditions, or to estimate the person's psychological well-being. Moreover, ambulatory activities are useful for detecting falls and other accidents [NHC15].

Each taxonomy establishes a solid foundation regarding the types and definitions of activities included, and provides a certain level of understanding which affects choices concerning proper sensor selection and system design. As the research in the field of HAR progresses, new taxonomies of activities, contingent on the needs of each application, are required and are being developed. Depending on each application's goals, a subset of all activities that can be performed by humans needs to be detected and recognized, requiring the appropriate taxonomy, and matching sensors and processing techniques [RMM16; NHC15].

Time-related connections

Each activity is performed within a finite time interval which is defined by the durations of the composing actions. According to their composition and their concurrency, activities can be characterized as:

- Composite, regarding their complexity, when each one of them consists of an ordered succession of simpler activities or actions (the order of the simpler steps might differentiate from person to person according to their habits or preferences, and this might be the reason behind different variations of a composite activity)
- Sequential, regarding their concurrency, when two or more activities are performed successively, with the corresponding time intervals having or not having a gap between them, but without them overlapping
- Concurrent, regarding their concurrency, when two or more activities are performed at the same time, so the corresponding time intervals overlap fully or partially
- Interleaved, regarding their concurrency, when an ordinarily longer activity is temporally interrupted to perform a shorter and possibly sporadic activity, in a sense that the short activity is contained in the break of the longer activity [NHC15]

1.1.2 Categorization

Human Activity Recognition techniques can be divided into two basic categories, according to the types of sensors that are being used: *visual sensor-based activity recognition* and *non-visual sensor-based activity recognition* [Che+12; RMM16]. These two categories are not necessary mutually exclusive, since it is possible to use a combination of both types of sensors for an activity recognition task [CJK15; RMM16]. A comparison of these approaches, in general terms, is presented in Table 1.1.

Visual sensor-based activity recognition relies heavily on the use of visual sensors such as video cameras which gather visual data that capture human behavior or relevant environmental state changes. In this case, computer vision techniques are essential in order to extract valuable features from visual observations [Che+12].

Non-visual sensor-based activity recognition relies on the use of sensor networks, by employing sensors attached to the person whose behavior is being observed (wearable sensors or smartphones) and/or sensors attached to objects in the activity environment (dense sensing). Wearable sensors capture human behavioral information in order to identify physical movements, and by extension corresponding activities. On the contrary, dense sensing captures human-object interactions, exploiting multiple multi-modal sensors attached to different objects. In general, the collected sensor data from sensor-based monitoring are, in the majority of cases, time series of variables depicting state changes. These sensor data are processed using probabilistic or statistical analysis, data fusion, and knowledge technologies [Che+12].

Comparison of HAR approaches		
	Advantages	**Disadvantages**
Visual sensor-based approaches	A single camera can track a wide angle of an environment One camera can replace many sensory devices Easy to operate Provide reliable data Suitable for security and surveillance systems and tele-immersion systems	Privacy issues Acceptance issues Cameras track only specific details of the environment High-sensitive video cameras are comparatively expensive Sensitivity to light/brightness factors Cameras require higher power consumption to operate More computer processing power and higher processing times required in general
Non-visual sensor-based approaches	A network of sensors can track more aspects of human behavior Privacy can be ensured Sensors cost less in general Comparatively lower power consumption for the sensors to operate Comparatively less computer processing power and less processing time are required Suitable for healthcare and AAL systems	Acceptance issues Intrusiveness of wearing single or multiple sensors Might need a large set of sensors, specifically to track each behavior Might provide unreliable data Accuracy issues Prone to errors due to single sensor malfunctions
Multimodal approaches	Suitable for detecting complex activities Light-weight sensors Comparatively low power consumption to operate Suitable for healthcare and AAL systems	Acceptance issues Require multiple sensors to capture full body movements Intrusiveness of wearing single or multiple sensors Data fusion algorithms may lead to false predictions

Table 1.1: Comparison of HAR approaches, based on the types of sensors that are being used [RMM16]

Human Activity Recognition techniques can be divided into two basic categories, based on the principles the activity models are built on: *data-driven activity recognition* and *knowledge-driven activity recognition* [Che+12; Que+15]. There are also hybrid HAR approaches which combine data-driven and knowledge-driven techniques [OCW14].

Data-driven activity recognition relies on modeling activities using available large-scale datasets, by creating probabilistic or statistical activity models which are trained and tested using the aforementioned datasets (a so called *bottom-up* approach). This approach is superior regarding the ability to handle uncertainty and temporal data, since it is built using them, but the requirements in data are usually very demanding, and it also requires special handling to ensure scalability and adaptation to distinct individuals [Che+12; Que+15].

Knowledge-driven activity recognition is based on prior knowledge in the application domain, in order to build activity models directly, exploiting methods from knowledge engineering and management technologies

(knowledge modeling and representation). In this case, activity models are used for activity recognition via formal logical reasoning (a so called *top-down* approach). This approach is clear and delicate from a semantical and logical perspective, but its usability is limited when in need to handle uncertainty and temporal data. Moreover, insufficient knowledge in the application domain will probably lead to activity models that will be viewed as incomplete [Che+12; Que+15].

This diploma thesis will focus on non-visual sensor-based, data-driven activity recognition, mainly for practical and privacy reasons. Attempts to perform activity recognition tasks in an unconstrained daily life context have brought inertial wearable sensors, such as accelerometers and gyroscopes, in the forefront of this research endeavor, since they allow activity tracking beyond the laboratory's instrumented rooms, practically anywhere and at any time portable sensor recording is possible [BBS14].

1.1.3 History and Applications

Since the late 1990s, there have been ongoing feasibility studies regarding activity recognition using portable sensors [FSF99]. As the years passed, successful activity recognition studies and technology advancements have propelled HAR towards more ambitious, real-world applicable projects [BBS14]. There are numerous domains where HAR using wearable sensors could have a beneficial impact, most notably in the following.

Healthcare and Active Assisted Living

Monitoring daily life activities can effectively enhance traditional medical practices, in order to support the procedure of medical diagnosis, to facilitate recovery and/or rehabilitation, and to assist patients with chronic diseases or impairments, in multiple ways [BBS14]. In many cases, it is useful to monitor daily activities and/or vital signals of patients or older adults in order to check if a medical condition is under control, to examine the development of an illness, and to take action when a rapid change in one of the parameters is observed [NHC15].

In order to battle against any kind of cognitive decline that is observed among elderly people, it is crucial to capture activity patterns in daily routines and use the irregularities in them to diagnose cognitive diseases, such as dementia and Alzheimer's disease, or other common, age-related diseases, such as depression and diabetes, which naturally influence the patients' daily activity routines. In addition, in these cases, a very useful initiative would be to use the activity patterns to remind patients of activities they have forgotten or neglected to do, or even remind them of the way a certain activity is performed, if needed [NHC15].

Anomaly detection aims to discover anomalies, irregularities in activities or in activity patterns which deviate from normal behavior [NHC15]. One of the most valuable anomaly detection applications is fall detection for older adults, since it is a common health condition among the elderly and its impact on quality of life and rehabilitation is not negligible [WX08].

Unobtrusive body-worn sensors can also be used by doctors and nurses in hospitals and health centers in order to interact with the Hospital Information System (HIS) exclusively using gestures. In this way, the nurses' and doctors' hands are kept sterile, and they can assess the patient's condition and access the patient's data at the same time [Luk+07].

Sports and Leisure

Body-worn sensors can be used to improve the daily quality of life, by recognizing athletic activities and related movements [Avc+10]. Many different methods for sportive activity recognition have been developed, including recognizing activities such as walking, running, cycling, rowing, using both supervised and unsupervised data [Erm+08], recognizing walking and running at different intensities [Als+13], and recognizing athletic activities in order to detect running asymmetries to prevent and manage injuries [Mor+15]. Furthermore, wearable sensors can be used to segment sports motions and motion sequences, to contribute to sports motion analysis and performance evaluation, and physical education [Hei+06; KK18].

Industrial Sector

Unobtrusive body-worn sensors can be used to monitor work activities of production and maintenance workers. In this way, the collected data regarding the way each activity in the production line is performed,

can be used to create relevant how-to manuals, to validate the workers' performance accuracy, and to provide help and evaluate trainees' progress [Luk+07].

1.2 Thesis Aims and Objectives

The scope of this work includes studying state-of-the-art literature on wearable sensor-based Human Activity Recognition and understanding the constituent parts of the Activity Recognition Chain, including preprocessing, segmentation, feature extraction and classification. Subsequently, a review on popular Machine Learning and Deep Learning architectures and frameworks and their applications on HAR is necessary to integrate such models in our experiments later on.

The most interesting and innovative part of HAR involves sensor data collection *in-the-wild*, which means collected by users in free-living conditions. Although this data collection setup ensures the richness of representations and labels compared to data collection in the lab, HAR tasks become more challenging since the resulting dataset is much noisier, much less reliable, and much more difficult to handle. Following this scenario, we aim to experiment with data collected *in-the-wild* and we will use the ExtraSensory dataset which contains labeled data from 60 users totaling over 300k minutes, collected from smartphone and smartwatch sensors, accompanied with 51 activity and context labels.

In order to conduct a thorough investigation on *in-the-wild* HAR using the ExtraSensory dataset, we will first study the papers of the original researchers who created this dataset. We aim to understand the data collection procedure and the provided dataset parts, which include among others: raw sensor measurements, pre-extracted statistical features, and activity and context labels. We will study the baseline models implemented by the original researchers and we will replicate their experiments. We will also study the publications of other researchers using this dataset and summarize their efforts.

Our novel experiments will be preceded by exploring the dataset to understand its unique properties: how unbalanced it is, how many examples exist per user and per label, how the features change when performing different activities or when different users perform the same activity. Then, we will try to improve the baseline recognition scores by enhancing the baseline models which use the pre-extracted features, by adding a bidirectional LSTM (BiLSTM) to model sequences of examples. We will try to enhance this model further by adding a Self-Attention module. Also, we will try to validate whether adding a Cross-Attention module between the BiLSTM outputs and the input features will can provide useful insights on model interpretability. Furthermore, we will also build models using the raw sensor data as input, and we will experiment with many architectures, including BiLSTM layers, convolutional neural network (CNN) layers and Transformer Encoder layers. Finally, we hope to be able to share valuable insights in order to further improve HAR performance in future work, based on our experience on the task.

1.3 Thesis Structure

In Chapter 2 a literature review on wearable sensor-based Human Activity Recognition will be presented, in order to understand the evolution and the fundamental state-of-the-art techniques used in the field.

In Chapter 3 an overview of the most common Deep Learning techniques for wearable sensor-based HAR will be provided.

In Chapter 4 the concept of Activity Recognition *in-the-wild* will be introduced and the ExtraSensory dataset will be presented and explained, referring to the work of the researchers of the original publications, followed by other researchers who have used it.

In Chapter 5 we present our work on Human Activity and Context Recognition based on wearable sensors and *in-the-wild* data collection, using the ExtraSensory dataset, including brief exploration and visualization tasks, followed by experiments using Machine Learning and Deep Learning models.

In Chapter 6 we summarize the most important conclusions of our experiments and we suggest directions for future work.

Chapter 2

Literature Review

In this Chapter, an attempt is made to give a thorough insight of the current literature, focusing on the evolution of the state-of-the-art methods used in the field of Human Activity Recognition using wearable sensors.

2.1 Formal Definition of the Problem

According to Lara and Labrador [LL13], the problem of non-visual sensor-based Human Activity Recognition can be defined as follows.

Definition 2.1.1: Human Activity Recognition Problem (HARP)

> Given a set $S = \{S_1, ..., S_k\}$ of k time series, each one from a particular measured attribute, and all defined within time interval $I = [t_a, t_w]$, the goal is to find a temporal partition $\langle I_1, ..., I_r \rangle$of I, based on the data in S, and a set of labels representing the activity performed during each interval I_j. This implies that time intervals I_j are consecutive, non-empty, non-overlapping, and such that $\bigcup_{j=1}^{r} I_j = I$.

Definition 2.1 does not apply in cases where the activities performed in the initial time interval I are concurrent, and it does not work for composite activities when it is desired that they are recognized in multiple different granularity levels.

It should be noted that the HARP cannot be solved deterministically, since the number of combinations of attribute values and activities could be infinite. Furthermore, as long as the activities' durations are unknown, finding the transition points remains a challenging task. For most machine learning techniques to be applied (excluding some unsupervised ones), the problem is relaxed and reformulated as follows (Definition 2.1), dividing the time series into fixed length time windows, and providing activity labels. The objective now is to find a function that maps time windows to activity labels in the best possible way [LL13].

Definition 2.1.2: Relaxed Human Activity Recognition Problem (RHARP)

> Given (1) a set $W = \{W_1, ..., W_m\}$ of m equally sized time windows, totally or partially labeled, and such that each W_i contains a set of time series $S_i = \{S_{i,1}, ..., S_{i,k}\}$ from each of the k measured attributes, and (2) a set $A = \{a_1, ..., a_n\}$ of activity labels, the goal is to find a mapping function $f : S_i \rightarrow A$ that can be evaluated for all possible values of S_i, such that $f(S_i)$ is as similar as possible to the actual activity performed during W_i.

Due to the relaxation in Definition 2.1, some error is introduced to the model during transition windows. This happens because although in the definition it is assumed that only one activity is performed during each time window, in reality a person might perform two sequential activities in it, since the initial time interval is arbitrarily divided into time windows. However, this relaxation error can be considered as insignificant for most applications, as long as the number of transitions between activities is much smaller than the total number of time windows [LL13].

To the extent of our knowledge, no definition which includes the case of concurrent activities was found in the literature. Thus, in this work the following definition is proposed for this case (Definition 2.1).

Definition 2.1.3: Relaxed Concurrent Human Activity Recognition Problem (RCHARP)

> Given (1) a set $W = \{W_1, ..., W_m\}$ of m equally sized time windows, totally or partially labeled, and such that each W_i contains a set of time series $S_i = \{S_{i,1}, ..., S_{i,k}\}$ from each of the k measured attributes, and (2) a set $A_c \subseteq \mathcal{P}(A)$ where $A = \{a_1, ..., a_n\}$ is the set of possible activity labels and A_c is the subset of the powerset of A which contains all possible activities and all possible combinations of concurrent activities, the goal is to find a mapping function $f : S_i \rightarrow A_c$ that can be evaluated for all possible values of S_i, such that $f(S_i)$ is as similar as possible to the actual activity or set of concurrent activities performed during W_i.

2.2 Activity Recognition Chain

An Activity Recognition Chain (ARC) is a sequence of signal processing, pattern recognition, and machine learning techniques that comprise a specific activity recognition system (Figure 2.2.1). An ARC bears some similarity to general-purpose pattern recognition systems, but also has a number of specific requirements and constraints related to the activity recognition process. If supervised classification algorithms are used, the chain can be executed in two different modes of operation, namely training (modeling) and testing (classification). Unsupervised classification does not include a dedicated training step, since it directly infers activities from the sensor data [BBS14].

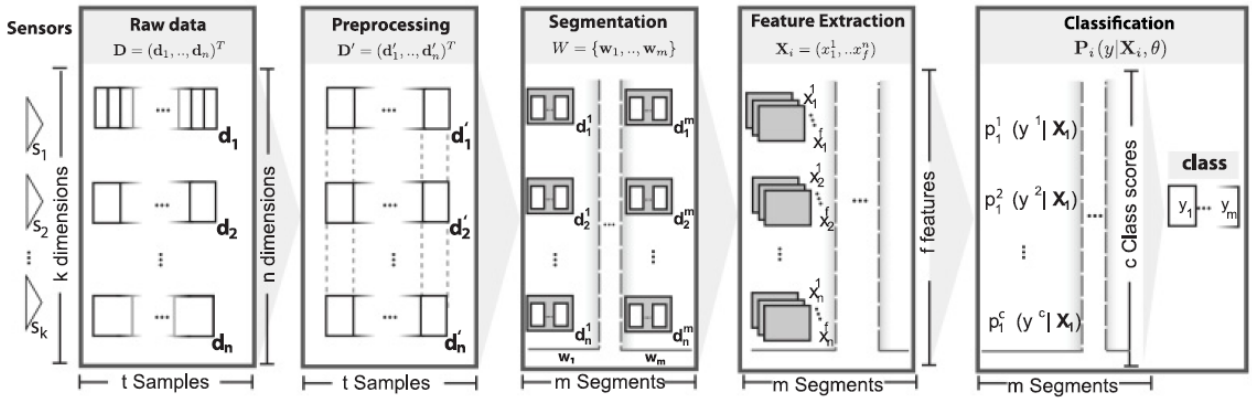


Figure 2.2.1: Typical Activity Recognition Chain (ARC) to recognize activities from wearable sensors [BBS14]

The ARC input consists of sensor data streams acquired from sensors. Raw signals ($\mathbf{D}$) are first preprocessed ($\mathbf{D}'$) to filter out signal variability and artifacts, and then segmented into sections of interest ($\mathbf{W}_i$) that are likely to contain an activity. Afterwards, feature vectors ($\mathbf{X}_i$) that capture the activity characteristics are extracted from the signals within each segment. In training mode, the extracted features and the corresponding ground truth labels are used as input to train a classifier model (determine parameters θ). In classification mode, the features and the previously trained model are used to score c activity classes $\mathbf{Y}_i = \{y^1, ..., y^c\}$ with a confidence vector $\mathbf{p}_i$, and map these scores into a single class or multi class label. If multiple sensors or classifiers are used, the output of several classifiers might be fused into a single decision. Finally, a performance evaluation stage is used during design time to assess the performance of the ARC [BBS14].

2.3 Sensors

The most commonly used sensors for monitoring human activity in recent research, in both visual and non-visual approaches, and their main characteristics are presented in Table 2.1. Although cameras and microphones, the primary sensors used in visual sensor-based activity recognition, undoubtedly can provide rich context information about human activities, in general the acquired data from them require more computational power in order to be processed. In addition, cameras and microphones are often perceived as privacy threats, especially in a daily life setting [NHC15]. As long as the acceptance of the technology and the convenience of the users are considered to be a setting stone of the system design and implementation, the focus should be put on non-visual approaches, mainly based on portable (wearable) sensors.

During the past two decades, there have been exceptional advancements in the development of microelectronics and computer systems, which enabled sensors and mobile devices with unprecedented characteristics. They are characterized by their small size, high computational power and relatively low cost, hence they have become obtainable for the general public, triggering the genesis of *Ubiquitous Sensing*, an active research area with the main purpose of extracting knowledge from the data acquired by pervasive sensors [LL13]. HAR using non-visual sensors can be approached in two different ways, namely using *external* sensors (i.e. in smart homes) and *wearable* sensors, and in this thesis the focus is put on the latter approach.

Sensors for activity recognition and their main characteristics				
Sensor	**Measurement**	**Data**	**Advantages**	**Disadvantages**
Video cameras	Human actions Environmental state	Image, video	Precise information	Privacy issues Computational expense Acceptability issues
Microphones	Voice detection Other sounds	Audio	Certain and rich information about sound	Implementation difficulty Acceptability issues
Simple binary sensors	User-object interaction detection Movements and location identification	Categorical	Low-cost Low-maintenance Easy to install and replace Less privacy sensitivity Minimal computational requirements	Provide simple and limited information for composite and multi-user activity recognition
RFID	Object and user identification	Categorical	Small size Low cost	Reader collision Tag collision Limited range
Wearable inertial sensors	Acceleration Orientation	Time series	Compact size Low cost Non-intrusiveness High accuracy User identification Location tracking	Cumbersome and uncomfortable feeling Insufficient context information
Wearable vital signs sensors	Vital signs	Analog signal	Sensitive to slight changes More accurate in emergency situation detecting	Reliability constraints Security issues Uncomfortable feeling for long-time skin attaching

Table 2.1: Commonly used sensors and their main characteristics related to activity detection [NHC15]

2.3.1 Wearable Sensors

The term *wearable sensors* refers to sensors that are directly or indirectly attached to the human body. Usually, due to their small size, they can be embedded into clothes, shoes, accessories, and more recently into wristwatches (smartwatches) and mobile devices (smartphones), in order to be easier to wear and to interfere minimally with the users' habits and daily activities. These sensors can be divided in two main categories, *inertial sensors* and *vital signs sensors* (or *biosensors*). Wearable inertial sensors provide descriptive features of the user's body posture and movements. Wearable vital signs sensors capture vital signs of the user such as heart rate, blood pressure and skin temperature, that are most useful for health monitoring applications [NHC15].

Inertial Sensors

Inertial measurement units (IMU) usually consist of *accelerometers*, *gyroscopes* and potentially *magnetometers*. Accelerometers are the most popular inertial sensors used for activity monitoring. They measure the value of acceleration along a sensitive axis, and they are mainly targeted to monitor ambulatory activities and activities involving movement in general, such as walking, standing, sitting, walking upstairs or downstairs, exercising, or moving particular parts of the body, depending on the part or parts of the body the accelerometers are attached to. They can also provide useful information in order to assess a person's posture.

Data collected from accelerometers has at least the following four attributes: acceleration value in x-axis, y-axis and z-axis, and time. Due to their small size and their relatively low cost, accelerometers can be embedded into belts, bracelets, wristbands and watches, and nowadays they are integrated in most mobile phones. Recent research focuses in two different activity recognition challenges regarding accelerometers:

researchers either try to place accelerometers in different body parts in order to identify the optimal positioning combination for more accurate activity recognition, or try to get optimal activity recognition predictions while minimizing the number of accelerometers used and their obtrusiveness.

Gyroscopes are also commonly used sensors for activity recognition, especially combined with accelerometers. Gyroscopes use a small vibrating mass inserted into the sensor for measuring angular velocity and maintain orientation. The change of the angle compared to the initial known value can be detected over a period of time. The sensor's limitations include output drift over time, output offsets when the device is in static state, and sensitivity to a particular range of angular velocities.

The advantages of inertial sensors include their compact size, low power requirements, low cost, non-intrusiveness and the capacity to provide data directly related to the user's motion. Moreover, nowadays, these sensors are usually integrated in smartphones and smartwatches, and thus can be used in a minimally intrusive way. On the other hand, their limitations are not negligible. Their placement on the human body might cause an uncomfortable feeling, especially when they are placed in diverse positions. Moreover, as most inertial sensors collect data continuously, it is important to take precautions to protect the device's performance and battery life. Using inertial sensors cannot directly provide adequate context information, in particular when the activities include interactions with environment objects [NHC15; Nwe+18].

Vital Signs Sensors

There are multiple biosensors commonly used to monitor the respective human vital signals. These include Electroengephalography (EEG) sensors for electrical brain activity, Electrooculography (EOG) sensors for ocular activity, Electromyography (EMG) sensors for muscle activity, Electrocardiography (ECG) sensors for cardiac activity, pressure sensors for blood pressure, CO2 gas sensors for respiration, thermal sensors for body temperature and Galvanic Skin Response (GSR) for skin sweating.

These vital signs parameters are useful in order to observe a person's health status while performing daily activities. From EEG sensors, delta, alpha and beta waves are mainly used to detect sleep state, panic disorder, and sudden unexplained nocturnal death syndrome. EOG sensors detect eye movements: rapid eye movements would indicate that the person is awake, and slowly rolling eye movements would indicate that the user is in the state of transition from being awake to sleeping.

EMG sensors can capture muscle activity in different parts of the body and associate it with matching daily life activities. ECG sensors allow heart rate monitoring, which gives indications of arrythmias. Blood pressure monitoring using pressure sensors, or indirectly using ECG sensors, is indicative of immediate changes, such as hypotension or nose bleeding. Respiration monitoring measures the airflow through nose and mouth. Monitoring body temperature is valuable for detecting fever, and skin sweating can be used to detect sport and housework activities.

The advantages of vital signs sensors include their low cost, low error levels, non-intrusiveness and high accuracy. Their main disadvantages are related to their reliability constraints and the uncomfortable feeling that they cause when attached on the skin for long time periods, let alone on a daily basis [NHC15].

2.3.2 Sensor Data Acquisition

In the first stage of a typical ARC, raw data is acquired from the wearable sensors. Sensors placed in the surrounding environment as part of advanced HAR systems are out of scope for the current study. Since some sensors can provide multiple values (e.g., an accelerometer), or multiple sensors are jointly sampled, vector notation is used to describe the collected data:

$$\mathbf{s}_i = (\mathbf{d}^1, \mathbf{d}^2, \mathbf{d}^3, ..., \mathbf{d}^t), \text{ for } i = 1, ..., k, \tag{2.3.1}$$

where k represents the number of sensors and $\mathbf{d}^t$ the sensor i values at time t. Each sensor is usually sampled at regular time intervals, thus the resulting data stream consists of multivariate time series. However, the sampling rates of different sensors may differ, since they are dependent on the sensor type. For example, a typical sampling frequency for GPS devices is $5Hz$, while acceleration is usually sampled at $25Hz$ or more. Sensors' sampling frequency can also be altered for practical reasons, including power saving and application

requirements. In order to synchronize multimodal sensors, to remove signal noise and artifacts, and to prepare the data for the inference algorithms, a data processing stage is introduced next [BBS14].

2.4 Sensor Data Processing

Raw data collected by sensors has to be processed in order to extract valuable features that will be used in the learning process later on. Sensor data processing plays a critical role in the output accuracy and the HAR results, thus this is a pipeline step that must be well taken care of for optimal recognition results [NHC15]. The most common processing methods are presented in Figure 2.4.1.

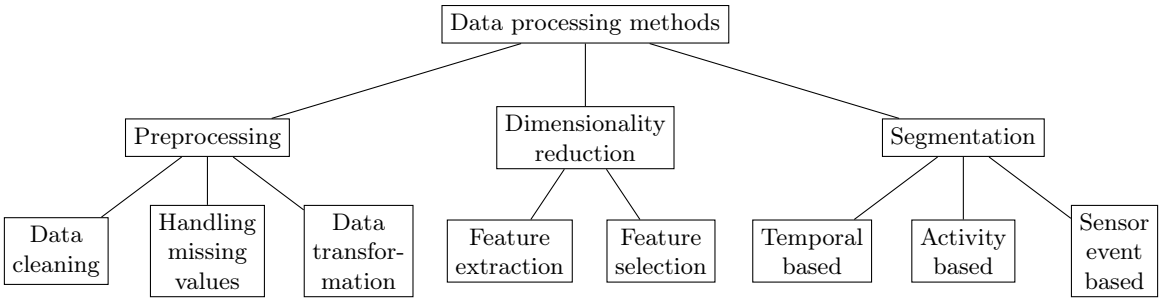


Figure 2.4.1: Most common sensor data processing methods [NHC15]

2.4.1 Data Preprocessing

Sensor data is noisy by nature. *Data preprocessing* is necessary in most cases, and it usually consists of *data cleaning* to remove artifacts and discard unwanted samples, *data interpolation* to handle missing values, and *data transformation* to convert data to the desired format [NHC15]. Preprocessing of acceleration and gyroscope signals might also include resampling synchronization, or signal-level fusion.

The preprocessing stage transforms the raw multivariate and nonsynchronous time series data into a preprocessed time series $\mathbf{D}'$:

$$\mathbf{D}' = \begin{pmatrix} {d'_1}^1 & \cdots & {d'_1}^t \\ \vdots & \ddots & \vdots \\ {d'_n}^1 & \cdots & {d'_n}^t \end{pmatrix} = (\mathbf{d}'_1, ..., \mathbf{d}'_n)^T, \tag{2.4.1}$$

where $\mathbf{d}'_i$ corresponds to one dimension of the preprocessed time series, n to the number of total dimensions, and t to the number of samples [BBS14].

Data Cleaning

Raw sensor data is probably noisy and contains erroneous and possibly redundant information, because of hardware imperfections and failures, and possible intermittent communication loss in wireless networks. In order for this data distortion not to impact the HAR process, data cleaning must be applied. The goal is to filter out artifacts and only keep the signals' characteristics that can provide valuable information, by using appropriate filters according to the type of sensor the data came from [NHC15].

The triaxial acceleration data from accelerometers contains a constant component due to gravity, which should be removed in order to obtain the real acceleration (corresponding only to the person's movements) captured by the sensor. A low-pass filter can be applied to isolate the constant gravity acceleration in the time series dataset obtained from the accelerometer, as demonstrated by Khan et al. in [KSL13].

Moreover, signal denoising is useful to reduce signal noise and artifacts. In [Wan+11], four common filters: median filter, Butterworth low-pass filter, discrete wavelet package shrinkage, and Kalman filter, are used for 3D acceleration signals denoising, and their performance is evaluated using signal-to-noise ratio (SNR)

and correlation coefficient (R) between the filtered signal and a reference signal. The results showed that the Kalman filter had the largest SNR and R values. Wavelet package shrinkage and median filter were similar in terms of the metrics values, while the Butterworth low-pass filter had the worst performance due to the waveform delay it causes.

When performing data cleaning, there are two more issues that might need to be taken into consideration, namely *class imbalance*, when some classes exhibit many more samples than others, and *class overlapping*, when a subset of the samples might correspond to more than one class, causing ambiguity. There are numerous different approaches to cope with these issues, probably also depending on the type of machine learning that will be later applied (supervised or unsupervised, single-class or multi-class) [NHC15]. Two of the most common techniques applied to overcome the issue of class imbalance are undersampling the majority class [Goo+10], or oversampling the minority class or some of the minority classes [BWG13].

Handling Missing Values

Data collected from sensor networks might have missing values. The most common practice to fill in the missing readings is to use interpolation techniques, most notably linear interpolation [MM13], cubic interpolation, or nearest neighbor interpolation [NHC15].

Data Transformation

Data must be transformed in an appropriate format in order to apply further data processing and analysis, depending on the data mining approach and methods that will be used later on. For instance, nominal or categorical attributes might be more convenient than numerical ones for some data mining algorithms, and thus it is useful to convert them, if possible, to help optimize recognition performance later [NHC15].

Data normalization is also commonly used as a data transformation technique, to convert all sensor data to a common scale, hence allowing quantitative or qualitative direct comparison. Because of the immense range of different scales, data normalization should be performed while taking the following four criteria into account [Mit07]:

1. nature of scale, referring to the fundamental mathematical properties of each scale, regarding scale boundaries, ordering, and allowed operations
2. homogeneity, referring to the sensors measurements' types similarity
3. statistical distribution, referring to the empirical statistical distribution that the sensor values follow when the sensors are attached to a given population
4. semantics, referring to the meaning of a scale e.g. probabilistic, possibilistic, utility, or degree-of-similarity, and the incommensurability issues that might arise with different scale semantics

2.4.2 Data Segmentation

Segmentation methods are introduced in order to resolve one of the main challenges in sensor data preprocessing, to produce a proper division of the continuous input data flow into small segments of information. Each data segment $\mathbf{w}_i = (t_1, t_2)$ is defined by its start time t_1 and its end time t_2 within the time series. The segmentation stage yields a set of segments W containing potential activities:

$$W = \{\mathbf{w}_1, ..., \mathbf{w}_m\} \tag{2.4.2}$$

Segmenting a continuous data stream is a difficult task and it is crucial to select the appropriate segmentation technique and parametrize it accordingly, because the derived segments will be the fundamental units on which feature extraction and inference algorithms will be applied. Current literature emphasizes on three categories of segmentation approaches: *temporal-based segmentation*, *activity-based segmentation* and *sensor event-based segmentation*. [BBS14; NHC15]

Temporal-based Segmentation

There are two major temporal-based segmentation approaches: *time interval segmentation* and *sliding window segmentation*. In the first approach, data obtained from sensors is divided into chunks of equal time duration, and this approach is frequently used to subdivide the temporal data collected by accelerometers and gyroscopes [NHC15; KP08].

The time interval windowing approach is highly convenient for data streams that are continuously collected by sensors over long time periods, and it is usually preferred because of its lower computational complexity. Nonetheless, it is crucial to select an appropriate value for the time interval. An excessively small interval might divide activities into two or more adjacent windows, and this can deteriorate the chances of accurate classification, while an excessively wide interval might cluster two or more activities into the same segment, leading again in classification issues [NHC15].

The sliding window approach is also very commonly used for activity monitoring. In this approach, the data obtained from sensors is divided into windows with static or dynamic length. In the case of fixed window length, it can be based on equal time intervals or equal number of sensor events. Three segmentation algorithms of this kind are of particular interest: Fixed-size Non-overlapping Sliding Window (FNSW), Fixed-size Overlapping Window (FOSW), and Sliding Window and Bottom-up (SWAB) [NHC15; Keo+01].

FNSW is the simplest approach, without any window overlap, but since it employs fixed length windows, it carries the drawbacks already mentioned above. FOSW is an improvement to FNSW that implements data overlap between adjacent windows. The overlap percentage may vary from 0% (practically FNSW) to 100% (non-sliding window). SWAB segmentation is a combination between the aforementioned techniques and a more accurate offline processing of the data stream. It processes a buffer of samples that span several segments, based on the structure of the buffer data [NHC15; Keo+01].

In [Ach+12], Achumba et al. conduct experiments to investigate the impact of window length when using the FNSW algorithm, and the impact of overlap percentage and window length when using the FOSW algorithm for segmenting the sensor data stream. After extracting features based on previous literature and applying the Support Vector Machine (SVM) classifier, the results demonstrated that, for the particular feature and classifier choices, the optimal segmentation approach out of the ones tested was the FOSW algorithm with a window length of 12s and 90% window overlap.

Dynamic sliding window methods allow varying window sizes, depending on sensor state changes, location changes, and/or typical activity characteristics. In [KC14], Krishnan and Cook propose combinations of static and dynamic window lengths. They enhance the sliding window approach by making modifications in order to calculate the window length dynamically based on the sensor data, the environment and the monitored activity. Their approach evaluates the following factors to determine the window size: time-based weighting, mutual information-based weighting, activity probabilities in the previous window and previously detected activity.

The sliding window approach is very useful when monitoring activities that contain a significant periodic component, such as walking and running, and when monitoring static activities, such as sitting and standing. The computational complexity depends on the window length and whether it is fixed or not, and on the factors that are being considered in the case of dynamic window length. Hence, there is a trade-off between speed and accuracy in the activity recognition system [NHC15].

In current literature, a wide range of window sizes can be observed, depending on the type and the characteristics of monitored activities. Typical window sizes for accelerometer data vary from 0.25s to 10s [KWM11; SR12]. In [Bañ+14b], Banos et al. evaluate the impact of different window sizes, from 0.25s to 7s, used for activity recognition with a non-overlapping sliding window approach. From their results, they have reached two main conclusions: window size significantly matters, and short windows normally lead to better recognition accuracy. The use of larger windows appears to be a necessity when a simple set of features is selected, while for richer feature sets it might be redundant. Furthermore, generally, activities with higher motion variability (e.g. household activities) typically require longer window sizes, whilst activities with lower motion variability and more repetitive movements (e.g. walking, running, cycling) work well with shorter window sizes [NHC15].

Activity-based Segmentation

Activity-based segmentation is based on identifying the start and end points of each activity and dividing the sensor data stream accordingly. Thus, the main challenge in this type of segmentation is to identify the time boundaries of each activity properly, and this can be done by using various methods. For instance, in [YTO13], Yoshizawa et al. attempt to identify the boundaries, distinguishing between static and movement-related activities. Regarding static activities, a threshold is set to detect the changing points, while for movement-related activities an analysis of variations in the frequency domain is used to locate the boundaries. Another way to obtain information about the time boundaries of the monitored activities, more approximately this time, is to ask the users to give explicit feedback on the activities they perform, through a smartphone interface for example [Der+12].

Another related approach, energy-based segmentation uses signal energy levels to identify the boundaries of different activities. Since in many HAR problems different activities are performed with different intensity levels (e.g., walking and running), and these intensity levels correspond directly to different intensity levels of the recorded accelerometer signals. The energy E of a signal is defined as:

$$E = \int_{-\infty}^{\infty} |s(t)|^2 dt \tag{2.4.3}$$

The differences in signal intensity translate to different energy levels, hence by thresholding on E, data segments that are likely to belong to the same activity can be identified. A special case of energy-based segmentation is to require the user to assume a predefined rest position between each two activities. This way, whenever the predefined rest position is detected by the HAR system, a segment boundary will be placed. [BBS14; Gue+09]

Sensor Event-based Segmentation

Sensor event-based segmentation methods are focused towards splitting the sensor data stream into chunks based on sensor events rather than (static or dynamic) time intervals. They are mostly used in cases where the events an that form the activity or the sensor measurements may not be distributed uniformly in time, thus a time windowing approach is not appropriate. In most cases, this approach divides the time series of input data into into segments with equal number of sensor events or into segments containing all the sensor events of a single activity [NHC15]. Another approach is to segment sensor data using information derived from additional modalities [BBS14]. For example, long-term acceleration data recorded on a mobile phone can be segmented using GPS traces [AS03].

2.4.3 Dimensionality Reduction

Dimensionality reduction includes the extraction of features that are representative of the significant data characteristics and discriminative of the different classes (activities) and the selection of specific features or transformed features to reduce the feature vector's dimensionality while maintaining most of the relevant information and the portrayed discriminative capability. The ability to extract useful, substantial features from raw sensor data is extremely practical in order to manage the potential huge volume of sensor data, and becomes a key factor for the performance and accuracy of HAR systems [NHC15].

Feature Extraction

Features represent the major characteristics of the sensor data obtained from its quantitative values. It is crucial to extract a relative low number of high representative and discriminative value, to improve the accuracy of machine learning algorithms and reduce the computational cost of activity inference. In this phase, the processed dataset is transformed into a collection of feature vectors, each one corresponding to a single data segment, which must contain proper information to be used as input in the activities discrimination and learning algorithms. Features are extracted as feature vectors $\mathbf{X}_i$ on the set of segments W, with $\mathcal{F}$ being the feature extraction function:

$$\mathbf{X}_i = \mathcal{F}(\mathbf{D}', \mathbf{w}_i) \tag{2.4.4}$$

The total number of features extracted from the data form the so-called feature space. In general, the more clearly each activity can be separated in the feature space, the higher the achieved recognition performance will be. Ideally, feature values corresponding to the same activity should be clustered in the feature space, while feature values corresponding to different activities should be far apart. Furthermore, reliable features should be robust across different people and to the activity's interclass variability [NHC15; BBS14].

The most common approaches for feature extraction operate in three domains: time domain, frequency domain, and discrete domain. Table 2.2 contains the main features which can be extracted in these domains.

Most common extracted features for HAR	
Domain	**Extracted Features**
Time domain	Mean, Median, Average, Variance, Standard Deviation, Minimum, Maximum, Range, Root Mean Square (RMS), Correlation, Cross-Correlation, Zero-Correlation, Integration, Differences, Velocity, Signal magnitude area (SMA), Signal vector magnitude (SVM), Difference, Zero-crossing
Frequency domain	Fourier Transform (DC component, Key Coefficients, Coefficients sum, Dominant frequency, Spectrum Energy, Spectrum Entropy, Spectrum centroid), Wavelet Transform
Discrete domain	Euclidean-based Distances, Dynamic Time Warping, Levenshtein Edit Distance

Table 2.2: Taxonomy of extracted features in three domains [NHC15]

Time Domain Metrics Simple mathematical and statistical metrics can be used to extract basic signal information from raw sensor data. They can also be used as indicators in other domains, in order to select key signal characteristics or features [Fig+10].

Among time domain features, mean is one of the most commonly extracted features because of its low computational cost and representative ability. There have been various uses of the mean metric in activity recognition: it has been used to discriminate between static and dynamic activities [Vel+96], and also as input to classifiers, such as Naive Bayes, Neural Networks, Decision Trees, and Kohonen Self-Organizing Maps. [NHC15; Fig+10]

The standard deviation, which represents the stability of a signal around its mean, is also frequently used as a basic metric for classifiers or threshold based algorithms. Another valuable feature, the median, can be efficiently used to replace missing values from a sequence of discrete measurements.

Frequency Domain Features focus on the periodic structure of sensor data. The wavelet transform can capture sudden changes in signals, thus it is usually used to detect activity boundaries. The Fourier transform provides information about the main frequency components of a signal, and can be computed for a time-based discrete signal over a specific window length by using algorithms, such as the Fast Fourier Transform (FFT).

Discrete Domain Features map sensor signals to strings of discrete symbols. Euclidean-based distances, Dynamic Time Warping, and Levenshtein Edit distance are some commonly used approaches used to extract discrete features for classifying human activities [NHC15].

Feature Selection

If the feature space dimensionality is high, more training data is needed for model parameter estimation and the classification becomes more computationally intensive. Especially for real-time processing on embedded systems, where it is crucial to minimize memory, computational power, and bandwidth requirements, the minimum number of features that still allow the ARC to achieve the target recognition performance should be used.

Feature selection is the procedure responsible for selecting the most discriminative features and possibly transforming the features in order to reduce the dimensionality of the feature vector. Hence, the main task

here is to find and extract a more relevant subset of features from the feature vector created in the previous step, to confine data redundancy, noise, and computational expense, before applying learning models to the extracted features. Since manual selection of these features is a challenging task, various methods for automatic feature ranking and selection have been developed [BBS14; NHC15].

The various different features selection methods that are being used, depending on the characteristics of the extracted features, include Minimum Redundancy and Maximum Relevance (mRMR) feature selection [PLD05], Correlation-based Feature Selection (CFS) [Hal99], SVM-based feature selection [NSS05], and Sequential Forward Floating Search [GD14]. Moreover, Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Linear Discriminant Analysis (LDA) are used to map the high dimensional feature vector into a lower dimensional one [NHC15].

2.5 Model Training and Classification

There is an abundance of inference methods that can be used in activity recognition systems, due to the extensive research in machine learning and computational statistics over the last decades. HAR researchers have successfully utilized temporal probabilistic models, such as Hidden Markov Models (HMMs) [Fin14], Conditional Random Fields (CRFs) [LFK05; Kas+08], and Dynamic Bayesian Networks [Pat+05]. Moreover, discriminative approaches, such as k-Nearest Neighbors (k-NN) [UB16], Support Vector Machines (SVMs) [HJ09], Multilayer Perceptrons (MLPs) [AS17; PY19], Decision Trees [BI04; SS18], and Joint Boosting [Les+05], have also been used in multiple activities and sensor settings.

There is no single best solution for all activity recognition tasks when choosing the suitable machine learning algorithms. The choice for a particular inference method depends on the classification task and is subject to the trade-off between computational complexity and recognition performance. In particular, when performing classification on embedded systems with limited resources, the goal is to minimize computational complexity and memory requirements while still achieving relatively high recognition performance. Consequently, machine learning algorithms are selected contingent upon the subset of activities that must be recognized, the complexity of the feature space, and other factors, such as the response latency and the mode of operation (online or offline) [BBS14].

2.5.1 Training

Supervised learning models need to be trained before use for inference purposes. Training is performed using training data $\mathcal{T} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{N}$, with N pairs of feature vectors $\mathbf{X}_i$ and corresponding ground truth labels y_i. Model parameters θ can be learned, in order to minimize the classification error on $\mathcal{T}$. For instance, Hidden Markov Models are defined by parameters $\theta = (\pi, A, B)$, with matrix A corresponding to transitions between states, B to the output probabilities of each state, and π to the initial state probabilities. Given the training data $\mathcal{T}$ and an initialization of the parameters θ, a separate model is trained for each class using the expectation-maximization (EM) algorithm [Rab89; Fin14]. Discriminative methods minimize the error by employing gradient descent. In contrast, non-parametric classifiers, such as k-NN, take as parameters the labeled training data $\theta = (\mathcal{T})$ and match the label of the k-nearest neighbors to the test sample.

2.5.2 Classification

The classification stage can be divided in two distinct steps. In the first step, each feature vector $\mathbf{X}_i$ that corresponds to an instance of the test set is mapped to a set of class labels $\mathcal{Y} = \{y^1, ..., y^c\}$ with corresponding scores $\mathcal{P}_i = \{p_i^1, ..., p_i^c\}$, using a trained model with parameters θ:

$$p_i(y|\mathbf{X}_i, \theta) = \mathcal{I}(\mathbf{X}_i, \theta), \text{ for } y \in \mathcal{Y}, \tag{2.5.1}$$

with the inference method $\mathcal{I}$. In many cases, including Bayesian approaches (e.g. Naive Bayes) and other, non-Bayesian approaches with the appropriate calibration, the scores $\mathcal{P}_i$ correspond to probabilities.

In the second step, the calculated scores can be used in various ways to determine the dominant class. The most common practice is to find the maximum score and to consider the corresponding class as the

classification output:

$$y_i = \mathbf{argmax}_{y \in \mathcal{Y}, p \in \mathcal{P}_i} \, p(y|\mathbf{X}_i, \theta) \tag{2.5.2}$$

Moreover, the scores can be fed as input to other functions to determine the classification output, and also can be used in criteria to evaluate if the system's output can be trusted [BBS14].

2.5.3 Decision Fusion

Decision fusion is the practice of combining several intermediate classification results into a single decision. It is specially valued in HAR applications where it is usual to use multiple sensors and/or multiple classifiers. Fusion can take place in earlier stages (feature level) or in a later stage (classification level) of an ARC [BBS14]. The most common fusion rules that are used in HAR research are summation, majority voting [Sti+08], Borda count [War+06], and Bayesian fusion [Zap+07]. In addition, boosting as a variant of decision fusion has been successfully applied in HAR problems. Boosting works by sequentially applying a classification algorithm to reweighted versions of the training data and then taking a weighted majority vote of the sequence of classifiers thus produced [FHT00].

Aside from the potential increase in recognition performance, sensor and decision fusion provide multiple additional benefits for HAR systems, including:

1. increased robustness to faults and variability in the systems used
2. reduced classification problem complexity by using partial classifiers matching sensor modalities
3. derivation of confidence measures by evaluating the agreement between classifiers
4. classification with missing features (potentially)

2.5.4 Performance Evaluation

It is crucial to evaluate the recognition performance of an ARC and to do so optimally for each specific activity recognition task, since different tasks might require different evaluation metrics. In general, activity recognition systems can miss, confuse, or falsely detect activities. Classification can be correct and lead to *True Positives* (TPs) and *True Negatives* (TNs), or wrong and lead to *False Positives* (FPs) and *False Negatives* (FNs). The optimization objective might be to maximize a single performance metric or multiple metrics at the same time; this highly depends on the application.

Activity recognition has adopted the most common performance metrics that are widely used in pattern recognition in general, such as *confusion matrices*, related measures including *accuracy*, *precision*, *recall*, and *F-scores*, and decision-independent *Precision-Recall* (PR) and *Receiver Operating Characteristic* (ROC) curves. These metrics are briefly explained below [BBS14].

Confusion Matrix

A confusion matrix displays quantitatively how many instances of each activity class were classified correctly and how many got misclassified (i.e., confused) by the system. Typically, the rows of a confusion matrix correspond to the number of instances in each actual activity class, while the columns correspond to the number of instances for each predicted activity class. To fill each row of the matrix, all ground truth instances of the corresponding (actual) class are compared with the predicted class labels. Using the confusion matrix, *precision* ($\frac{TP}{TP+FP}$), *recall* ($\frac{TP}{TP+FN}$), *accuracy* ($\frac{TP+TN}{all}$), and *F1 score*, the harmonic mean of precision and recall, ($\frac{2*precision*recall}{precision+recall}$), can be calculated for each activity class.

The overall accuracy is representative of the true performance of a classifier only when the dataset is relatively balanced. If a dataset is unbalanced, namely if the number of actual instances of the activity classes varies significantly, accuracy can be strongly biased by dominant classes. To address this problem, normalized confusion matrices and balanced performance metrics should be used to allow for objective comparison and evaluation [BBS14].

ROC and PR Curves

Since it is not always possible to set the optimal decision threshold on the classifier's score in advance, it is common practice to stretch the threshold of the score for each individual class and analyze the behavior in so-called Receiver Operating Characteristic (ROC) or Precision-Recall (PR) curves [Faw06]. In an ROC curve, the *true positive rate* (recall) is plotted against the *false positive rate* ($\frac{FP}{FP+TN}$), while moving the decision threshold. An ROC curve starts at the lower left corner, where $(FPR = 0, TPR = 0)$, which corresponds to a decision threshold of 1, and ends at the upper right corner, where $(FPR = 1, TPR = 1)$, which corresponds to a decision threshold of 0.

Best-case results approach the upper left corner, while random results follow the diagonal (if class distributions are balanced). Since ROC curves depend on TN counts, having imbalanced class distributions can lead to "overoptimistic" ROC curves. On the other hand, PR curves do not depend on the TN count, and thus are more suitable for activity detection tasks, where several instances of a particular activity of interest are hidden into a large corpus of various activities. A PR curve starts at the upper left corner, approximating the point where $(Precision = 1, Recall = 0)$, which corresponds to a decision threshold of 1, and ends at the upper right corner, approximating the point where $(Precision = 0, Recall = 1)$, which corresponds to a decision threshold of 0. Best-case results approach the upper right corner.

Several metrics can be extracted from ROC and PR curves in order to render the depictions comparable. *Equal Error Rate* (EER) represents the point in the PR curve where precision equals recall, and the higher this value, the better. *Average precision* is calculated from the PR curve by measuring precision at uniform steps of recall and then by taking the average. Finally, the most popular metric is the *Area Under the Curve* which is calculated from ROC curves as a measure of the overall performance of a classifier. The AUC corresponds to the probability that a classifier will rank a randomly chosen positive instance with a higher score than a randomly chosen negative one [BBS14].

Evaluation Schemes

In general, activity recognition datasets may include recordings of multiple persons, on multiple days. and of multiple runs containing repetitions of multiple activities from a set. Evaluation is generally conducted using a leave-one-out cross validation scheme, in order to assess how the activity recognition performs in unseen data. Thus, the experimental dataset is split into multiple folds. All but one folds are used in training the recognition system, and the remaining fold is used for testing. This process is repeated several times, alternating the left-out fold that is used for testing, until all folds have been used for testing once.

Dataset folds can be built by taking different aspects of the dataset into account, when trying to evaluate different generalization aspects of the activity recognition system. When building a user-independent system, leave-one-person-out is used to evaluate generalization to unseen users. In a user-specific system, leave-one-run-out is used to assess performance in unseen runs. Leave-one-day-out is commonly used to evaluate the robustness of the system over time [BBS14].

2.6 Challenges

HAR research is considered to be an exciting and demanding research field as it inherits various pattern recognition challenges, and also is characterized by many inherent activity and sensor-related challenges.

2.6.1 Pattern Recognition Challenges

Intraclass Variability

As in pattern recognition in general, in HAR it is important to develop recognition systems that are robust to interclass variability. Such variability is inevitable when capturing human activities and habits that might be performed in different ways by different individuals, or even in different ways by the same individual due to different conditions (varying factors include hour of the day, emotional and environmental state, stress and fatigue). In cases where generalization performance across many users is of great interest, robustness to intraclass variability must be ensured. HAR systems are subject to a sensitive trade-off between using a

highly specific and discriminative feature set, and using a more generic feature set that has less discriminative power but is potentially more robust across different people [BBS14].

Interclass Similarity

Interclass similarity poses an inverse challenge, where different activities might show similar characteristics in the sensor data, rendering the discrimination between them much more difficult [BBS14]. Interclass similarity might be present in different sports (e.g., between baseball and softball, or between basketball and netball), and in daily activities (e.g., between eating food and eating desert) that consist of or include very similar movements. Such close similarity could possibly be resolved by using additional cues captured by different sensor modalities [Sti+08] or by analyzing concurrent or adjacent (predicted) activities [HFS08].

The NULL Class Problem

Usually, only part of a continuous data stream is relevant for HAR systems, especially when the subset of activities to be detected is limited to a specific domain (e.g., ADLs, movements, sports). Thus, activities of interest can get confused with similar-patterned activities that are irrelevant to the desired application (the so called NULL class). In most cases, it is impossible to explicitly model the NULL class, since it represents a theoretically infinite space of arbitrary activities (all the activities not included in the target domain in an uncontrolled environment). In some cases, it might be possible to implicitly identify the NULL class if some corresponding signal characteristics differ considerably from those of the desired activities. In these cases, the NULL class can be be identified by using a threshold on either the raw feature values or the confidence scores calculated by the classifier [BBS14].

2.6.2 HAR-specific Challenges

Definition and Diversity of Activities

It is crucial for the design and development of HAR systems to establish a clear and precise definition of the activities under investigation and their characteristics. Since human activity is complex and diverse, an activity can be performed in many different ways and for various purposes. In this context, it is imperative to use established activity taxonomies (briefly presented in Subsection 1.1.1), definitions, rules, detailed descriptions, and any other means available to explicitly define and border the scope of each activity under investigation in a specific HAR application [BBS14].

Class Imbalance

Another common difficulty is that of modeling different activity classes in the face of considerable class imbalance. In uncontrolled environments and unscripted activity tracking, as in long-term behavioral monitoring, some activities occur very ofter, such as sleeping, sitting, or walking, while many other are performed less frequently, sparsely during the day, such as eating or brushing teeth. As already mentioned in Subsection 2.4.1, class imbalance can often be addressed in various ways, for example by recording additional training data, or, as an alternative, by generating artificial training data to extend the minority classes. Recording additional training data might be challenging, or even impossible if experimental procedures are not to be constrained or scripted in any way, in order to capture real-life behavior. On the other hand, oversampling (i.e., duplicating) the classes which originally contained much less instances (refering to labeled training data) is possible and can mitigate class imbalance [BWG13; BBS14]. It is important to note that the extent to which a recognition system's performance is impacted by class imbalance is largely contingent on the selected models and algorithms that are used (different models are affected in different extents, depending on they way they model data) and on the degree of imbalance itself.

Ground Truth Annotation

Another challenge for supervised HAR recognition tasks is the collection of annotated or "ground truth labeled" training data. Ground truth annotation is a wearisome task, as it has to be completed manually, in real time, or afterwards in cases where there is also available video footage of the activities performed, since motion data recorded from an accelerometer or gyroscope is much more difficult to interpret. Posterior

annotation using video footage is mainly possible in stationary and laboratory experiment settings, but in daily life settings, in cases where cameras are not used, ground truth annotation is a far more difficult problem. If only a few labeled training samples are available, semisupervised [Sti+11], unsupervised [HFS08], or knowledge transfer [ZHY09] learning techniques can be used.

Data Collection

More experimental challenges are associated with data collection and the evaluation of HAR systems in real-world environments. In contrast to other research fields such as speech recognition or computer vision, the research community in activity recognition has not agreed upon some standard general-purpose datasets of human physical activity, that could be widely used for system testing and direct comparison. This happens mainly because data collection varies largely and depends on multiple factors: data quality, number of modalities and sensors, recording duration, and number of participants, amongst others. Using standard datasets is crucial for reproducible research and is becoming increasingly important in HAR as a research discipline [BBS14].

2.6.3 Application Challenges

Variability in Sensor Characteristics

One major practical challenge for implementing HAR in real-world applications is caused by the variability in sensor characteristics. This variability may be caused by various factors, including hardware errors and failures, variety in recording devices, changes in the operating temperature or loose straps. Some sensors are particularly sensitive to the environment and might need frequent recalibration. Moreover, portable devices containing sensors, such as smartphones, might be carried and used in different ways [BS08], resulting to obvious differences in the recorded signals [BBS14].

Trade-offs in HAR System Design

Another practical challenge when implementing HAR systems involves the trade-off between accuracy, system latency, and processing power [Yan+12]. For many real-world applications, such as systems supporting elderly care, real-time signal processing and classification might be essential, while for others, such as behavioral monitoring, offline data analysis and classification may be sufficient. Microscopic embedded sensors for data recording typically have relatively limited processing power. In order to decrease runtime and render real-time processing and classification possible, increasing the processing power is required. This requirement can be fulfilled by introducing a central component (e.g., a smartphone or even a remote server) in the experimental setup to aggregate and process the information drawn from different sensors [Lu+10]. In cases where a remote server is used to receive, process and retransmit the results back to the original device, the devices must be connected in a network.

2.7 Datasets

Activity recognition datasets are essential in order to train and evaluate activity recognition models and systems. There are two basic types of data acquisition schemes: *self data collection* and *public datasets* [Wan+19].

- Self data collection is performed in some work, in order to fully control the data collection procedure, regarding the setting, the sensor modalities and the quality, and to create datasets in different contexts, for different sets of activities or with combinations of sensor modalities not available in existing datasets. Self data collection requires thoughtfulness, patience and willingful volunteers, and it can be a tedious procedure to collect, prepare and process all the data.
- As an alternative to self data collection, over the last years many HAR datasets have been published, and these are adopted by many researchers for their work.

Although there is no single standard dataset that is used as a benchmark to test all activity recognition systems, there are some widely used publicly available datasets. An attempt to summarize them has been

made in Table 2.3. This table is focused on datasets with data from wearable inertial sensors and (potentially) other additional sensors.

Public HAR datasets based on wearable sensors							
Dataset	**Type**	**Subjects**	**Activities**	**Samples**	**Sensors**	**Setting**	**Ref**
OPPORTUNITY	ADL	4	18	701,366	A,G,M,O,AM	controlled	[Rog+10; Cha+13]
Skoda Checkpoint	Factory	1	10	22,000	A	controlled	[Zap+07]
UCI Smartphone	ADL	30	12	10,929	A,G	controlled	[RO+15; RO+16]
PAMAP2	Physical	9	18	3,850,505	A,G,M,HR	uncontrolled	[Rei12; RS12]
USC-HAD	ADL	14	12	2,520,000	A,G	uncontrolled	[ZS12]
WISDM	Mixed	51	18	15,630,426	A,G	controlled	[Wei19; WYH19]
DSADS	Physical	8	19	9,120	A,G,M	controlled	[ABT10; BA10]
Darmstadt DR	ADL	1	35	24,000	A	uncontrolled	[HFS08]
CHAD	Mixed	10	13	n/a	A,G	controlled	[Sho+16; Sho+15]
MHEALTH	Physical	10	12	16,740	A,G,M,ECG	controlled	[Bañ+14a; BGS14]
Daphnet FoG	Gait	10	2	1,917,887	A	uncontrolled	[Bac+10; RPH10]
REALDISP	Physical	17	33	n/a	A,G,M,Q	controlled	[Bañ+12; BTA12]
HHAR	Physical	9	6	43,930,257	A,G	controlled	[Sti+15a; Sti+15b]
UniMiB SHAR	Mixed	30	17	11,771	A	controlled	[MMN17]
MobiAct v2.0	Mixed	66	16	n/a	A,G	controlled	[Cha+17]
KU-HAR	Physical	90	18	20,750	A,G	controlled	[SN21]

A: accelerometer, G: gyroscope, M: magnetometer, Q: quaternion sensor, O: object sensor, AM: ambient sensor, HR: heart rate monitor, ECG: electrocardiogram

Table 2.3: Public HAR datasets based on wearable sensors

Chapter 3

Deep Learning for Wearable Sensor-based HAR

In this Chapter, recent advances in wearable sensor-based HAR using deep learning techniques are presented, by describing the basic architectural elements and some HAR applications of each deep learning approach.

3.1 Introduction

Although conventional pattern recognition approaches have made enormous progress on HAR by adopting machine learning algorithms, such as Decision Trees, Support Vector Machines (SVMs), naive Bayes, and Hidden Markov Models (HMMs), there are several limitations that restrict the performance of conventional pattern recognition methods regarding classification accuracy and model generalization. In controlled environments with a small set of activities and labeled sensor data, conventional pattern recognition methods are capable of achieving satisfying results. However, since these methods heavily rely on heuristic hand-crafted feature extraction which provides only relatively shallow features, there is much potential for deep learning techniques, especially regarding daily HAR tasks, uncontrolled environments, few labeled data and online or transfer learning tasks.

Deep learning can largely diminish the effort on designing features and can learn much more high-level and meaningful features by training an end-to-end neural network. The features can be learned automatically from the raw or preprocessed signals through the network, instead of being manually designed. In addition, deep neural networks can extract high-level representations in deep layers, rendering them more appropriate for complex activities recognition tasks. Moreover, deep networks' structure could be suitable for unsupervised, transfer and incremental learning [Wan+19]. A comparison of the two approaches for feature representation and feature learning is summarized in Table 3.1.

Comparison of deep learning feature representation and conventional feature extraction

Characteristics	**Deep learning-based feature representation**	**Conventional feature extraction**
Feature extraction and representation	Ability to learn features from raw sensor data and discover the most efficient patterns to improve recognition accuracy	Manually engineered feature vectors that are application dependent and unable to model complex activities
Generalization and diversity	Potential to automatically capture spatial and temporal dependencies and scale invariant features from unlabeled raw sensor data	Requirement of labeled sensor data and use of arbitrary feature selection and dimensionality reduction approaches that are not necessarily generalizable
Data preparation	Data preprocessing and normalization not always compulsory to obtain improved results	Necessary data preprocessing, segmentation and dimensionality reduction
Temporal and spatial changes in activities	Hierarchical and translation-invariant representations useful to cope with such changes	Handcrafted features mostly inefficient in handling such changes
Model training and execution time	Requirement of large training dataset to avoid overfitting, and high computational power, mostly GPUs, to speed up training	Requirement of less training data, less computational power, and less memory usage

Table 3.1: Comparison of deep learning feature representation and conventional feature extraction [Nwe+18]

3.2 Deep Learning Approaches

Deep Learning [HOT06; Ben09] as a machine learning method has come a long way since its resurgence in 2006. Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input, embodying representation learning [Den14]. Over the years, deep learning has been extensively used in image recognition, speech recognition, natural language processing, amongst others. Research on the use of deep learning for feature representation and classification is growing

rapidly. In general, deep learning techniques can be subdivided into *discriminative models*, *generative models* and *hybrid models*.

Discriminative models are a class of supervised machine learning models used for classification or regression. These distinguish decision boundaries by inferring knowledge from observed data. A discriminative model is a model of the conditional probability of the target Y, given an observation x, symbolically $p(Y|X = x)$. Moreover, classifiers computed without using a probability model are also referred to loosely as discriminative. Typical conventional discriminative classifiers include linear regression, logistic regression, support vector machines (SVMs), conditional random fields (CRFs), decision trees, and neural networks. Deep discriminative classifiers include convolutional neural networks (CNNs), recurrent neural networks (RNNs) and deep fully-connected neural networks (DNNs).

Generative models are another class of machine learning models. Given an observable variable X and a target variable Y, a generative model is a statistical model of the joint probability distribution on $X \times Y$, $p(X, Y)$. From that, the conditional probability $p(Y|X = x)$ can be computed and classification can be based on that. Typical generative model approaches include naive Bayes classifiers, Gaussian mixture models (GMMs), and hidden Markov models (HMMs). Deep generative models include variational autoencoders (VAEs), generative adversarial networks (GANs), deep belief models, restricted Boltzmann machines, and others [Nwe+18; NJ02].

Hybrid models are models that combine generative and discriminative models for pattern recognition tasks. In these cases, generative models are used to derive feature maps and output a set of fixed length features that are used by discriminative models to perform classification. These hybrid schemes sought to integrate the intra-class information from generative models and the complementary inter-class information from discriminative methods [LLL11]. Some examples of hybrid models include convolutional restricted Boltzmann machine, convolutional sparse coding, and LSTM-density mixture model [Nwe+18].

3.3 Deep Learning Models

3.3.1 Deep Neural Networks

Architecture

The *perceptron* is the simplest form of a neural network. It can be used to classify patterns that are *linearly separable*. It consists of a single neuron with adjustable synaptic weights and bias. As shown in Figure 3.3.1, the summing node computes a linear combination of the inputs, while also taking account of an externally applied bias. The resulting sum is applied to a hard delimiter, which produces an output equal to $+1$ if the hard limiter input is positive, and -1 if it is negative, as follows [Hay09].

$$y = sgn[\mathbf{w}^T\mathbf{x} + b] \tag{3.3.1}$$

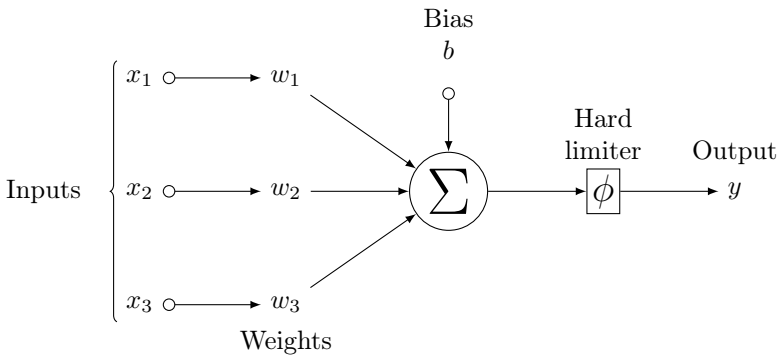


Figure 3.3.1: Signal-flow graph of a perceptron [Hay09]

Deep Feedforward Networks, also often called *Feedforward Neural Networks*, or *Multilayer Perceptrons (MLPs)*, are the typical, quintessential deep learning models. These network structures are built to overcome the practical limitations of the perceptron. According to the *universal approximation theorem*, a feedforward network with a single hidden layer containing a finite number of neurons can approximate arbitrary well real-valued continuous functions on compact subsets of $\mathbb{R}^n$ [Csá01; GBC16; Hay09].

A feedforward network defines a mapping $\mathbf{y} = f(\mathbf{x}; \theta)$ between the inputs $\mathbf{x}$ and the outputs $\mathbf{y}$, and learns the value of the parameters θ that result in the best function approximation. They are called *feedforward* because there are no *feedback* connections in which outputs of the model are fed back into itself. MLPs are usually characterized by the utilization of a non-linear, differentiable (or non-differentiable at only a small number of points) activation function following each neuron, by the existence of hidden layers and by a high degree of connectivity [GBC16; Hay09]. An example of an MLP with four hidden layers is presented in Figure 3.3.2.

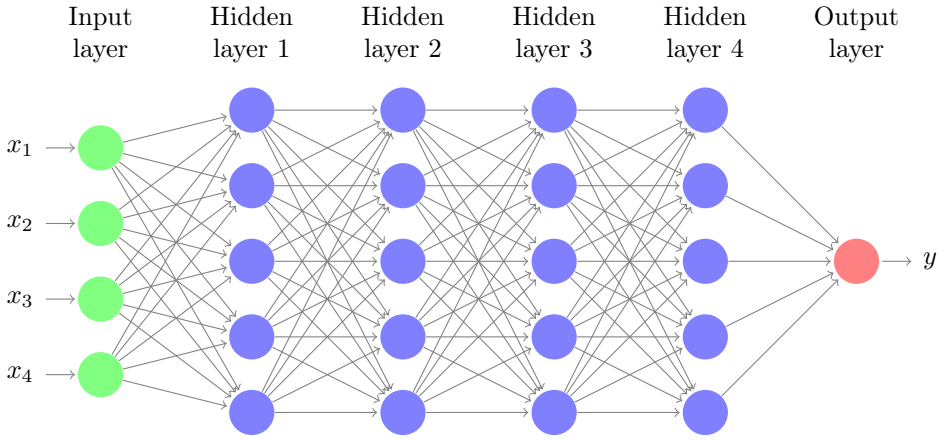


Figure 3.3.2: Architectural graph of a multilayer perceptron with four hidden layers

Application

MLPs usually serve as dense layers in other deep models, as seen for example in convolutional neural networks. However, there are cases where they are used alone in HAR, for classification tasks, or, more rarely, for both feature learning and classification. In [Vep+15], the A-Wristocracy framework that is implemented takes advantage of hand-engineered features extracted from the sensors, which are fed into a feedforward artificial neural network with two hidden layers. Moreover, in [WDT16], PCA for dimensionality reduction was applied on the hand-crafted features before using them as input in an MLP with one hidden layer. A comparative study between different deep learning approaches was conducted in [HHP16], where the researchers compared the performance of a five-hidden-layer MLP with that of CNN and RNN architectures for HAR tasks, performing automatic feature learning and classification [Wan+19].

3.3.2 Convolutional Neural Networks

Architecture

Convolutional Neural Networks (CNNs or ConvNets) are a specialized type of neural networks for processing data characterized by a grid-like topology, such as image data. As indicated by their name, convolutional neural networks employ convolutions instead of general matrix multiplication in at least one of their layers [GBC16]. The architecture of a typical CNN is structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. The role of the convolutional layer is to detect local conjunctions of features from the previous layer and the role of the pooling layer is to merge semantically similar features into one. Two or three stages of convolution, non-linearity and pooling are

stacked, followed by more convolutional and fully-connected layers, which perform classification or regression tasks [LBH15; Wan+19].

The convolution of a two-dimensional image I, used as input, with a two-dimensional kernel K, is illustrated in Figure 3.3.3 and given by:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3.3.2}$$

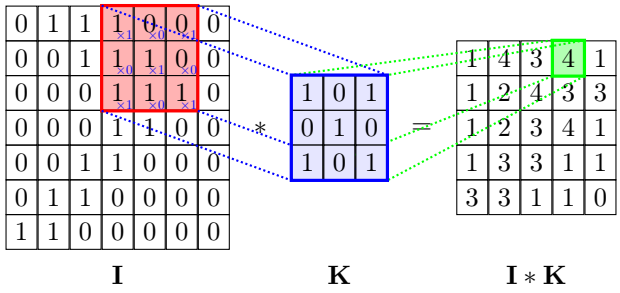


Figure 3.3.3: A two-dimensional convolution operation [Vel18]

Application

Due to the effectiveness of CNNs, they are largely used in HAR applications. Most HAR wearable sensors provide one-dimensional (1D) time series readings as input for a CNN. Therefore, input adaptation might be required, in order to transform these 1D vectors into a 2D virtual image. This however, is not necessary since CNNs can also be applied to 1D signals directly. Thus, input adaptation remains open as a design choice in each HAR task implementation [Wan+19].

When treating each 1D signal as a channel and using 1D kernels for the convolution layers and 1D pooling, the system identifies local patterns over time for each signal. In [Zen+14], each 1D accelerometer signal, corresponding to the measurements along one of the accelerometer's axes, is treated independently and the convolution and pooling layers are not shared between the different signals. After convolution and pooling, the outputs of each channel are flattened to unified hidden DNN layers. In [Yan+15], a similar network is implemented, with the difference that the kernels' weights of the convolutional layers are unified and shared across all signals. Convolutional practices and network designs similar to the aforementioned are included in [HHP16].

When forming a 2D virtual image by stacking or by transforming the 1D signals and applying 2D convolutions and 2D pooling, the system can capture both local dependencies over time and spatial dependencies over sensors. In [HYC15], signals are stacked to form a 2D virtual image, after sensors placed in the same body position have been grouped together, and within each group of signals, signals from each different type of sensor (accelerometer, magnetometer, gyroscope) are separated by padding zeros between them, to not be convolved together by the convolution kernel in the first convolutional layer. In [JY15], a novel *activity image* algorithm is proposed, based on signals of gyroscope, total acceleration and linear acceleration.

Moreover, in [Rav+16], each inertial signal is converted to a spectrogram, a representation of the signal as a function of frequency and time that is formed by calculating the squared magnitude of the short-time Fourier transform (STFT) after dividing the signal into shorter segments of equal length. The network applies 1D temporal convolutions on the spectrograms, so this is a case of an implementation that transforms the 1D signal into a 2D virtual image but then applies 1D convolutions over time and, in addition, signals corresponding to different axes or coming from different sensors are not combined together in the convolution stage, since each spectrogram depicts only one 1D signal. Thus, this work fundamentally differs from the previous mentioned ones regarding transforming the 1D signals to a 2D virtual image.

Another work, [CX15], explores the impact of the convolutional kernel's shape over a 2D virtual image created by stacking the signals corresponding to the three axes of an accelerometer. The researchers test the three possible kernel widths, 1 (kernel Nx1, in essence temporal convolution), 2 (kernel Nx2, combining signals from two axes), and 3 (kernel Nx3, combining signals from all three axes), on system accuracy, and found that kernel width of 2 performed the best in their HAR task setting.

Most convolutional layers in most of the aforementioned CNN architectures are followed by a pooling layer. Pooling helps to render the representation approximately invariant to small translations of the input [GBC16], and progressively reduces the spatial size of the representation, the number of parameters, the memory footprint and the computational requirements of the network. In most approaches, max [Zen+14; Yan+15; HHP16] or average [Yan+15; JY15] pooling is employed.

Regarding weight sharing choices, weight sharing across sensors in 1D CNNs was discussed in a previous paragraph. Weight sharing across the input space of each implementation has also been questioned. Although weights of local filters are tied and shared by all positions within the whole input space in traditional CNNs, serving the purpose of identifying an object or a feature regardless of its position in an image, in HAR different patterns might appear in different frames for a reason, therefore it may be better to relax the weight sharing constraint and adopt a partial weight sharing technique, as described in [Zen+14] and [HC16]. In general, although total weight sharing helps to speed up the training process, in these papers it is shown that partial weight-sharing could improve the performance of CNNs for HAR tasks [Wan+19].

3.3.3 Recurrent Neural Networks

Architecture

When feedforward neural networks are extended to include feedback connections, they are called *Recurrent Neural Networks* (RNNs). Recurrent neural networks were developed to model sequential data, such as time series data. RNNs process an input sequence one element at a time, maintaining in their hidden units a "state vector" that implicitly contains information about the history of the past elements of the sequence. RNNs, once unfolded in time, can be seen as very deep feedforward networks in which all the layers share the same weights, as it can be seen in Figure 3.3.4. RNNs are very powerful, but training them has proved to be challenging because of the backpropagated gradients that explode or vanish over many time steps. Moreover, it has been shown that it is difficult for RNNs to learn to store information for very long [BSF94; GBC16; LBH15; Nwe+18].

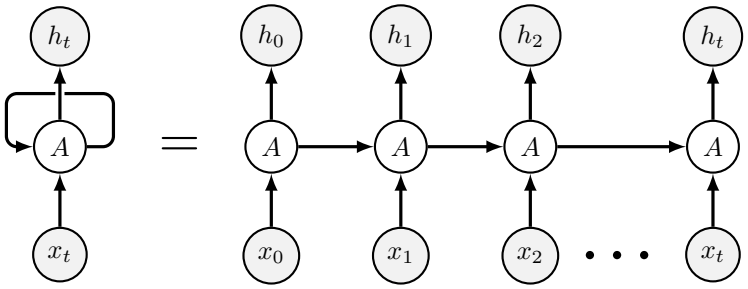


Figure 3.3.4: A recurrent neural network unfolding in time

To solve these problems, a helpful idea is to augment the network with an explicit memory. The *Long Short-Term Memory* (LSTM) model [HS97; GSC00] includes self-loops to produce paths where the gradient can flow for long durations, with the weight on this self-loop conditioned on the context. This is implemented using a memory cell that acts like an accumulator or a gated leaky neuron: it has a connection to itself at the next time step that has a weight of one, so it copies its own real-valued state and accumulates the input signal, but this self-connection is multiplicatively gated by another unit that learns to decide when to clear the content of the memory [GBC16; LBH15]. A block diagram of an LSTM unit is depicted in Figure 3.3.5.

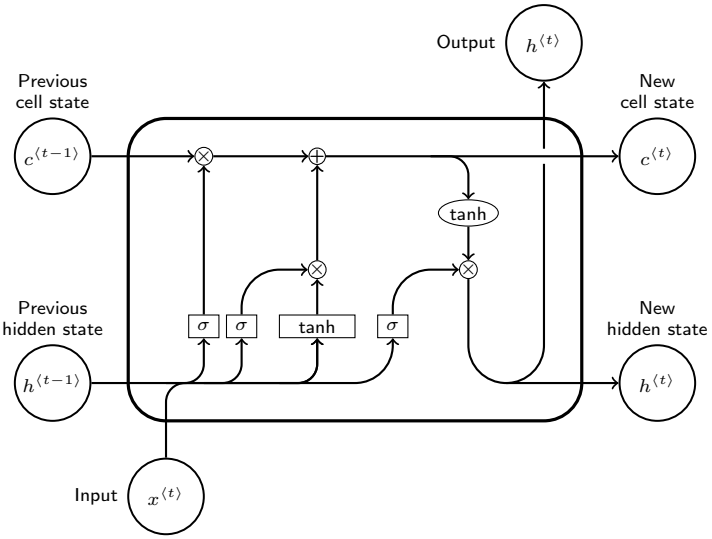


Figure 3.3.5: LSTM block diagram

Another type of gated RNN is the one whose units are called *Gated Recurrent Units* (GRUs) [Cho+14]. The main difference with the LSTM is that there is no cell state and a single gating unit simultaneously controls the forgetting factor and the decision to update the state unit. A block diagram of a GRU is depicted in Figure 3.3.6. The GRU has fewer parameters than the LSTM unit and thus networks using GRUs can be trained faster. GRU's performance in certain tasks was found to be similar to that of LSTM, while it has been proved that the LSTM is "strictly stronger" than the GRU [WGY18]. Many more variants around this theme can be designed. However, several investigations over architectural variations of the LSTM and GRU found no variant that would clearly beat both of these across a wide range of tasks [Gre+17; JZS15] and these two remain two of the most widely used RNN variations [GBC16].

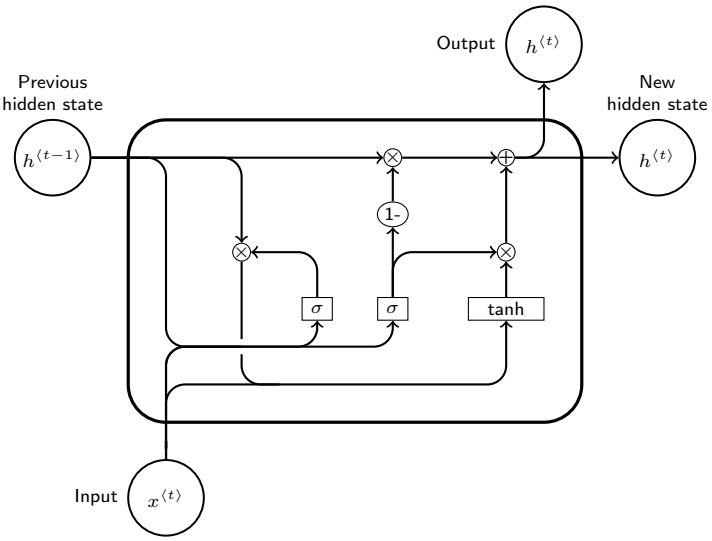


Figure 3.3.6: GRU block diagram

Application

HAR is a classic time series classification problem. Capturing the temporal dynamic in movement patterns is essential to model complex activity details and improve the performance of recognition algorithms. RNNs can be much more efficient than CNNs in modeling global temporal dependencies in sensor data. The potential use of RNNs in HAR tasks using data from wearable sensors is explored in numerous publications [Nwe+18].

In [IIN16], a deep recurrent neural network (DRNN) with LSTM units is used for HAR with high throughput from raw accelerometer data. Various architectures are investigated in order to determine the optimal parameter values regarding the number of hidden layers, the number of units, time truncation, the gradient clipping parameter, and the dropout rate. After determining a set of optimal parameter values for a specific dataset, additional experiments using these parameters and a different HAR dataset were conducted to assess their transferability and their generalization ability, with relatively satisfying results.

In [HHP16], recurrent approaches for wearable-based HAR are compared with MLPs and convolutional approaches. The implemented recurrent neural networks are based on LSTM cells in two variants: deep forward LSTMs, which contain multiple layers of recurrent units and are connected "forward" in time, and bidirectional LSTMs which contain two parallel recurrent layers that expanding in both the "future" and the "past" direction of the current time step, followed by a layer that concatenates their internal states for the current time step. These recurrent approaches outperform the MLP and CNN ones in most experiments, and they show similar behavior across multiple datasets and various model configurations.

Besides, in [Tam+17], LSTM-based RNNs are applied on multi-modal (acoustic and acceleration) singals for an HAR task. LSTM-RNNs can capture longer temporal context and their performance on the task is found to be superior to that of feedforward neural networks. Moreover, a system based on LSTM-RNN with a recurrent projection layer (LSTMP-RNN) is implemented for a subject adaptation task, to build a customized classifier for each user. In the experiments, it was observed that in subject adaptation, when providing 5 or more adaptation samples per class, LSTMP-RNN outperformed LSTM-RNN and other variants, including GRU-based RNNs.

In [GP17], fusion of multiple LSTM learners is implemented and the HAR framework that is built is based on ensembles of LSTMs, using an epoch-wise bagging scheme, in order to achieve reliability in real-world scenarios. The general idea of ensemble classifiers is to create collections of individual learners that are trained on different views of the dataset, and thus focus on different aspects of the problem domain, resulting in increased robustness and typically better average classification accuracy. The collection of LSTM learners is created by introducing variation in two ways: random selection of subsets of the dataset used as training data and two different loss functions for model training.

Moreover, in many works, additions or modification are made to the LSTM-RNN archicture in order to improve its performance in HAR tasks, including hierarchical deep LSTM (H-LSTM) [WL19], deep residual bidirectional LSTM [Zha+18], and attention-based LSTM [Sun+19] networks. [EK16] proposes a binarized bidirectional Long Short-Term Memory network (B-BLSTM-RNN) for HAR, suitable for resource-constrained environments, such as mobile and wearable devices. In this model, all weight parameters, input, and intermediate hidden layer output signals are binary-valued, and by replacing either floating or fixed-point arithmetic with significantly more efficient bitwise operations, it reduces its memory size and accesses and improves power efficiency, while maintaining performance comparable to that of the full-precision network.

3.3.4 Hybrid CNN-RNN Models

Various research efforts have been made to explore the potential of hybrid models that combine CNN and RNN layers for HAR tasks. The fundamental idea portrayed in this combination is that, in most cases, the CNN layers are used for feature extraction and they are followed by RNN layers, that exploit the temporal dependencies in the signal data [Nwe+18; Wan+19].

In [OR16], a generic deep framework for activity recognition based on convolutional and LSTM recurrent layers (DeepConvLSTM) is proposed, that is capable of automatically learning feature representations and modeling the temporal dependencies between their activation. The framework is suitable for multimodal wearable sensors' data, it can perform sensor fusion naturally, it does not require expert knowledge in designing features, and it explicitly models the temporal dynamics of feature activations. Compared to a baseline CNN model followed by feedforward dense layers, DeepConvLSTM performs better and allows to distinguish similar movements (e.g., open door, cloose door). A similar system combining convolutional and LSTM-based recurrent layers, called C-LSTM, is described in [YQY18], while a comparable deep learning framework, combining convolutional and GRU-based recurrent layers, called DeepSense, is proposed in [Yao+16].

In [MR16], the researchers employ a CNN-RNN architecture comprised of three convolutional layers, with max pooling layers in between, followed by an LSTM-based layer. They investigate whether the kernels in the convolutional layers are transferable in wearable activity recognition, considering transfer potential between users, application domains, sensor modalities and sensor locations.

3.3.5 The Attention Mechanism

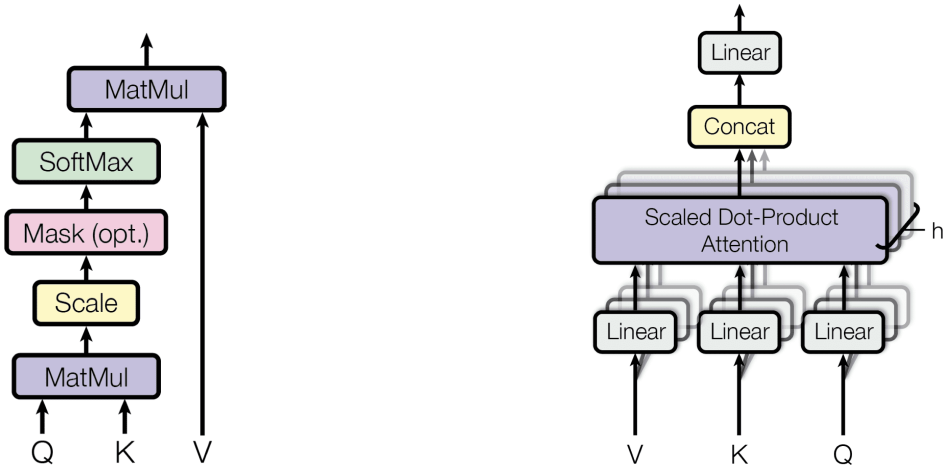
Architecture

An *Attention* function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

One of the most popular attention functions is the *Scaled Dot-Product Attention* [Vas+17]. The input consists of queries and keys of dimension d_k, and values of dimension d_v. The dot products of the query with all keys are computed, then each is divided by $\sqrt{d_k}$, and a softmax function is applied to obtain the weights on the values. In practice, the attention function is computed on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. The output matrix is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.3.3}$$

Two other commonly used attention functions are Bahdanau (additive) attention [BCB15], and Dot-Product (multiplicative) attention. The Scaled Dot-Product attention is similar to the Dot-Product attention, with the addition of the scaling factor $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While Bahdanau and Dot-Product are similar in theoretical complexity, Dot-Product attention can be implemented using matrix multiplication which is optimizable, and is much faster and more space-efficient in practice. The scaling factor in Dot-Product attention is added since, while for small values of d_k the two mechanisms perform similarly, additive attention outperforms Dot-Product attention without scaling for larger values of d_k.



(a) Scaled Dot-Product Attention

(b) Multi-head Attention

Figure 3.3.7: Attention mechanism details introduced in [Vas+17]

The concept of *Multi-head Attention* was introduced also in [Vas+17], where instead of applying a single attention function with d_{model}-dimensional keys, values and queries, it was found beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values the attention function is applied

in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values. Multi-head Attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$\begin{aligned} MultiHead(Q,K,V) &= Concat(head_1,...,head_h)W^O \\ \text{where } head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{3.3.4}$$

The attention projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

Another attention taxonomy refers to the attention input source. When keys, values and queries are generated from the same sequence, it is called *Self-Attention*. When the queries are generated from a different sequence than that of the key-value pairs, it is called *Cross-Attention*.

3.3.6 Transformers

Architecture

The *Transformer* is a sequence-to-sequence Deep Learning architecture based on the Multi-head Attention, introduced in [Vas+17]. It was proposed as an efficient architecture to handle long-distance dependencies in sequential data, by replacing the convolutional and recurrent layers of traditional encoder-decoder architectures, with stacked Self-Attention and point-wise, fully connected layers, as depicted in Figure 3.3.8, for both the encoder (left) and decoder (right). The encoder consists of encoding layers that process the input tokens iteratively one layer after another, while the decoder consists of decoding layers that iteratively process the encoder output as well as the decoder output tokens so far. The main components of this architecture are the following:

- The *Encoder* is composed of 6 stacked identical *Transformer Encoder layers*. Each layer has two sub-layers: a Multi-head Self-Attention mechanism, and a simple, position-wise fully connected Feed-Forward Network. A residual connection is employed around each of the two sub-layers, followed by layer normalization. Thus, the output of each sub-layer is $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the function implemented by the sub-layer itself. All sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{model} = 512$, to facilitate these residual connections.

- The *Decoder* is also composed of 6 stacked identical *Transformer Decoder layers*. In addition to the two sub-layers of each encoder layer, the decoder layers include a third sub-layer, which performs Multi-head Attention over the output of the encoder stack. Residual connections around each of the sub-layers, followed by layer normalization are used as in the encoder layers. Moreover, the Self-Attention sub-layer in the decoder layer is masked to prevent positions from attending to subsequent positions. Since the output embeddings are also offset by one position, it is ensured that the predictions for each position can depend only on the known outputs at past positions.

- *Multi-head Attention* is leveraged in three different ways:
  - Each encoder layer contains a Self-Attention sub-layer, whose input is the output of the previous encoder layer. Each position in the encoder can attend to all positions in the previous layer of the encoder.
  - Each decoder layer contains a Self-Attention sub-layer, whose input is the output of the previous decoder layer. Each position in the decoder can attend to all positions in the decoder up to and including that position. Leftward information flow in the decoder must be prevented to preserve the autoregressive property.
  - In the encoder-decoder attention layers, the queries come from the previous decoder layer, and the keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence, which mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models.

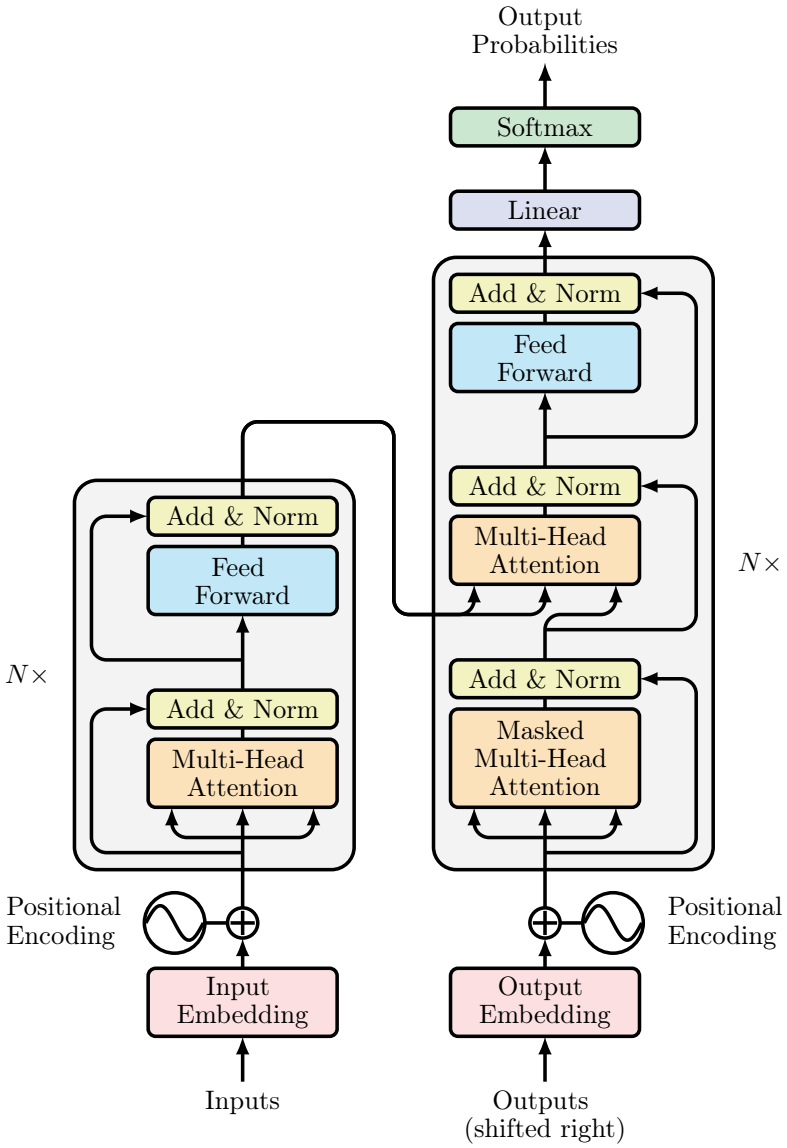


Figure 3.3.8: The Transformer model architecture introduced in [Vas+17]

- Each of the encoder layers and the decoder layers contains a fully connected *Position-wise Feed-Forward Network*, which means that it is applied to each position separately and identically. It consists of two Linear transformations with a ReLU activation in between. Its ouput is given by the equation $FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$. While the Linear transformations are the same across different positions, they are different from layer to layer. The dimensionality of input and output is $d_{model} = 512$, and the hidden layer's size is $d_{ff} = 2048$.

- Learned *Embeddings* are used as usually to convert the input tokens and output tokens to vectors of dimension d_{model}. A learned Linear transformation and *Softmax* function is used to convert the decoder output to predicted next-token probabilities. The same weight matrix is shared between the two embedding layers and the pre-softmax Linear transformation, as in [PW17]. In the embedding layers, those weights are multiplied by $\sqrt{d_{model}}$.

- The addition of some kind of *Positional Encoding* is necessary in order for the model to make use of the order of the sequence since the input is not processed in a sequential order by the model. The positional encodings are added to the input embeddings at the bottom of the encoder and decoder stacks, and

they have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, fixed (e.g., sine and cosine functions) or learned. The sinusoidal positional encodings are selected by [Vas+17] since they enable the model to extrapolate to sequence lengths longer than the ones encountered during training.

Application

In [SK21], a general framework for inertial-based activity recognition with Transformers is proposed. IMU data collected from mobile devices are given as input to a Transformer-based model, in order to perform HAR tasks. The proposed model, which includes four convolutional layers to embed the input to a higher dimension, followed by six Transformer Encoder layers and two Linear layers to produce the outputs for each class, is shown to outperform a CNN baseline on smartphone location recognition and human activity recognition. Also, in [DLKP22], a Transformer-based model which includes three Transformer Encoder layers followed by a Linear classification head is used in a HAR task based on IMU data from smartphones, and outperforms a Random Forest baseline. A lightweight Transformer for HAR using federated learning is proposed in [Raz+21].

3.3.7 Restricted Boltzmann Machines and Deep Belief Networks

Architecture

Restricted Boltzmann Machines (RBMs) [Smo86] are generative stochastic artificial neural networks that can generatively model input data. An RBM can be represented as an undirected bipartite graph, that consists of a set of stochastic visible units and a set of stochastic hidden units [BL16]. RBMs are a variant of Boltzmann machines, in which there are no connections between the units of each set, thus allowing for more efficient training algorithms, such as the gradient-based contrastive divergence algorithm. Contrastive divergence is an approximation of the log-likelihood gradient that has been found to be a successful update rule for training RBMs [CPH05; Ben09].

RBMs can be used in deep learning networks. In particular, *Deep Belief Networks* (DBNs) [HOT06] can be formed by "stacking" RBMs, where each sub-network's hidden layer serves as the visible layer for the next. DBNs can be trained greedily, one layer at a time, using the contrastive divergence algorithm, and learn to probabilistically reconstruct their inputs, while the layers act as feature detectors. DBNs can be further trained with supervision in order to be used for classification tasks.

Application

In [Ham+15], an HAR pipeline, including hand-crafted feature extraction and two RBMs to learn a generative model of the input features, is designed as a Parkinson's Disease assessment system. In [PHO11], RBMs are used for feature learning in HAR tasks, and their effectiveness is compared to hand-crafted statistical feature extraction and PCA. In [Als+15], RBMs are also used for feature learning as part of a HAR framework that combines deep learning and HMMs. Spectrograms of the segmented sensor data are fed as input to the first RBM layer.

In [Rad+16], a multimodal RBM (MM-RBM) architecture is proposed to integrate multimodal sensor streams and perform activity recognition tasks in resource-constrained computation units like wearable devices. The proposed MM-RBM system outperforms an RBM architecture with concatenated sensor streams input and three shallow classifiers (C4.5, SVM and Random Forest). Moreover, in [ZWL15], real-time HAR is implemented on smartphones, employing a DBN with three hidden layers, whose parameters are initialized by a generative pretraining and then finetuned using backpropagation. The model was trained offline and then loaded in a smartphone for inference, using sensor data collected by the phone.

Regarding hybrid CNN-RBM models, Lasagna, proposed in [Liu+16], is a system aimed at deep hierarchical understanding over mobile sensing data. In this system, all activity sensor data are embedded to multi-resolution descriptors through a hierarchical descriptor extraction module, which is composed of multiple stacked Convolutional Restricted Boltzmann Machines (CRBMs).

3.3.8 Autoencoders

Architecture

Autoencoders (AEs) are artificial neural networks trained to attempt to copy their input to their output. An AE has a hidden layer that describes a code used to represent the input, and can be viewed as an encoder-decoder system. The encoder function maps the input into a latent space, and the decoder function maps the latent space representation of the input signal to a reconstruction of the input. Through stacked hidden layers of encoding-decoding operations, autoencoders learn efficient latent representations of the input data in an unsupervised fashion. Several AE variants are proven to be efficient in learning representations for subsequent classification tasks, including *sparse*, *denoising* and *contractive* AEs [GBC16; Var+18].

Application

In [AAA17], the use of a stacked autoencoder model, built by stacking two AEs, is proposed for HAR, to enhance recognition accuracy and decrease recognition time. Similar work employing stacked autoencoders for HAR is included in [ZGW15], while in [KT17] a channel-wise ensemble of AEs is used to detect unseen falls from wearable devices' sensor data. Furthermore, in [Var+18], researchers introduce a novel HAR approach by expressing the problem as a set prediction problem, and propose Deep Auto-Set, a unified encoding-decoding architecture that consists of convolutional and deconvolutional layers. The system is trained in two steps: at start, in an unsupervised manner using the entire network, followed by a supervised finetuning step employing only the encoder part of the network.

In [Li+14], a sparse autoencoder (SAE), a denoising autoencoder (DAE) and PCA are compared as techniques for unsupervised feature leaning in HAR tasks, using SVM as a classifier. A similar comparison was made in [LT14], where a shallow sparse autoencoder, a deep sparse autoencoder and PCA are compared, using Random Forests as a classifier. In the same direction, a comparative study between data preprocessing methods and their impact on autoencoders' feature learning performance was conducted in [Har14].

3.3.9 Sparse Coding

Architecture

Sparse Coding [OF96] is a representation learning method that aims to find a sparse representation of the input data in the form of a linear combination of the elements of a basis as well as the basis itself. The learned basis may be overcomplete, and sparse coding is used in order to derive convenient data representations [Nwe+18]. Sparse coding provides an efficient way to find succinct representations of unlabeled data, since it can learn basis functions which capture high-level features in the data [Guo+14], rendering classification tasks easier and more accurate.

Application

Sparse coding is employed in HAR tasks to extract a compact, sparse and meaningful feature representation of sensor data that generalizes well across activity domains and does not rely on labeled sensor data. In [Bha+14], a sparse-coding framework for HAR is proposed, which uses a codebook of basis vectors that capture characteristic and latent patterns in the sensor data. The first phase of the framework consists of codebook learning from unlabeled data that produces a codebook of basis vectors, while the second phase consists of extracting feature vectors from a small amount of labeled sensor data using the basis codebook and training a standard classifier based on these features. A similar pipeline using sparse coding and sparse sensor data representations for wearable-based HAR is described in [ZS13].

Chapter 4

The ExtraSensory Dataset

In this Chapter, the concept of activity recognition in-the-wild is introduced and the ExtraSensory dataset is presented thoroughly. This Chapter is based on the original work by Yonatan Vaizman, Katherine Ellis, Gert Lanckriet, and Nadir Weibel, who collected, processed, and initially used this dataset for Human Activity and Context Recognition. Their work can be found in the following papers: [VEL17; VWL18; Vai+18]. Additional references will be cited accordingly.

4.1 Introduction

Health monitoring and lifestyle interventions nowadays require automatic context recognition, which can be helpful in order to offer immediate support when needed, by directly (e.g., fall detection) or indirectly (by correlating health lapses or crises with specific contexts) detecting critical conditions. It has been proven essential and beneficial to integrate automatic context recognition systems into large scale studies regarding lifestyle and health monitoring, while at the same time emphasis should be put on preserving real-life settings and sensor unobtrusiveness.

4.2 In-the-wild Conditions

Most activity and, more generally, context recognition datasets using wearable sensors prior to the Extrasensory Dataset contain data that has been collected under heavily controlled conditions, with subjects performing scripted sequences of tasks, accompanied with specific instructions, in constrained or unconstrained, laboratory or other environments. However, systems trained to recognize activities performed in such heavily controlled conditions might not generalize well in activities performed in real-life conditions due to the increased activities' variability. In this direction, four *in-the-wild* conditions are introduced [VEL17] to ensure that research is done in natural and realistic settings:

1. *Naturally used devices.* Using one or multiple foreign devices might affect the naturalness of a subject's behavior, hence it would be ideal for the subjects to use their own devices (smartphones, smartwatches etc.), as they naturally do.
2. *Unconstrained device placement.* Although device placement and orientation can have a great impact on activity classification accuracy, forcing specific device placement and orientation is not practical in real-world applications, thus it should be avoided, and the resulting variability should be addressed as a challenge to overcome.
3. *Natural environment.* The recorded activities should be performed in a place and time convenient for each subject, and they should be the place and time each subject would naturally perform these activities.
4. *Natural behavioral content.* The recorded activities should be activities that each individual subject would naturally perform. Moreover, the subjects should not be instructed to perform scripted sequences of tasks.

Applying in-the-wild conditions on data collection for activity recognition research renders acquiring labels of the subjects' behavioral context a much more difficult task, because the activities are not scripted and their duration and sequence are not predefined. Prioritizing in-the-wild conditions may result in fewer labeled activity instances and a smaller range of activity labels, unless special attention is paid to the label collection effort. In the ExtraSensory Dataset data collection process, the ExtraSensory App (Subsection 4.3.1) has been carefully designed to facilitate label reporting.

4.3 ExtraSensory

In the work conducted by the ExtraSensory team (ExtraSensory App and ExtraSensory Dataset), labeled data from over 300k minutes from 60 subjects was collected using smartphone and smartwatch sensors, in order to recognize activity and context information of human behavior in-the-wild. Each data instance corresponds to one minute for which multi-sensor measurements and relevant labels are provided. Labeling

activities and context information has been provided through self-reporting and the data has been labeled with *combinations* of relevant labels, to better depict possible multi-tasking conditions in natural behavior.

4.3.1 Data Collection

Large-scale data collection was possible using the ExtraSensory mobile app [Vai+18] developed by the ExtraSensory team for both iPhone and Android smartphones, and an adapted app for the Pebble smartwatch. The app's high usability allows for automatic sensor data collection and manual ground truth label annotation. For every minute that it is active, the app records a 20sec window of sensor measurements from the phone and watch (there is no guarantee that the time samples of different sensors are strictly aligned). The users can self-report the activity and context labels through the app's user-friendly interface that comes with practical and convenient mechanisms for self-reporting in order to maximize the number and the variety of annotated labels while minimizing the interference of self-reporting with the performed activity at the time.

Sixty subjects have participated in data collection, recruited from the UC San Diego campus. The subjects were of diverse ethnic backgrounds, including Chinese, Mexican, Indian, Caucasian, African-American, and more. Thirty four of the subjects were iPhone users, while the remaining twenty six were Android users with various devices. The majority of the subjects were right-handed, and subsequently wore the smartwatch on the left wrist. Thirty four subjects were female and twenty six subjects were male, and almost all the subjects were students or research assistants. Additional characteristics are described in Table 4.1.

Additional subject characteristics		
	Range	**Mean (SD)**
Age (years)	18-42	24.7 (5.6)
Height (cm)	145-188	171 (9)
Weight (kg)	50-93	66 (11)
BMI (kg/m^2)	18-32	23 (3)
Labeled examples	685-9706	5139 (2332)
Additional unlabeled examples	2-6218	1150 (1246)
Average applied labels per example	1.1-9.7	3.8 (1.4)
Participation duration (days)	2.9-28.1	7.6 (3.2)

Table 4.1: Statistics over the 60 research subjects [VEL17]

The ExtraSensory app was installed on the subjects' personal smartphone and the smartwatches were provided to the subjects by the researchers. Fifty six out of the sixty subjects agreed to wear the smartwatch. Rather than being instructed to perform any specific activities, the subjects were free to continue their daily life as usual and engaged in their natural behavior for about a week. They had to keep the app running in the background on their smartphone for as long as possible, but they were free to stop it when inconvenient, and as well to remove the watch and turn off the watch-app. Each subject was given a basic compensation of $40 and an additional compensation of up to $35 depending on the amount of the data they annotated. The resulting ExtraSensory Dataset contains a total of 308,320 labeled instances (minutes) from sixty users altogether. Details regarding the sensors characteristics and measurements are presented in Table 4.2.

4.3.2 Sensor Measurements

The publicly available ExtraSensory dataset contains all the following raw sensor measurements.

High Frequency Measurements

The following sensors were sampled at 40Hz during the 20sec recording window of each minute, producing a time series of approximately 800 time points. Since the sampling rate of the devices could not be guaranteed to be accurate, the time stamps of the samples of the time series were also recorded, and the differences between consecutive time points were verified to be approximately 25msec.

- Accelerometer. Time series of 3-axis vectors of acceleration along the standard axes of the smartphone devices.
- Gyroscope. Time series of 3-axis vectors of rotation rate around each of the phone's standard axes.
- Magnetometer. Time series of 3-axis vectors of the magnetic field.

Moreover, smartphones provide processed versions of the aforementioned signals. Regarding the accelerometer, the raw acceleration is split into the gravity acceleration and the user-generated acceleration. Regarding the gyroscope, a processed version is provided, in which the estimated drift effects have been removed. Regarding the magnetometer, an unbiased version is provided, in which the estimated bias of the device's magnetic field has been subtracted from the original signal. The raw acceleration signal and the calibrated gyroscope signal are included in the ExtraSensory Dataset, and attention was paid to unify the units of measurement for each type of signal across all operating systems.

Sensor characteristics and measurements

Sensor	**Raw measurements**	**Examples**	**Users**
Accelerometer	3-axis (40Hz)	308,306	60
Gyroscope	3-axis (40Hz)	281,883	57
Magnetometer	3-axis (40Hz)	282,527	58
Watch Accelerometer	3-axis (25Hz)	210,716	56
Watch Compass	heading angle (var)	126,781	53
Location	long-lat-alt (var)	273,737	58
Location precomputed	location variability (1pe)	263,899	58
Audio	13MFCC (46ms frames)	302,177	60
Audio power	1pe	303,877	60
Phone state	1pe	308,320	60
Low frequency sensors	1pe	308,312	60
Core		176,941	51

Core: instances that have measurements from all six core sensors
(Acc, Gyro, WAcc, Loc, Aud and PS), 1pe: sampled once per instance,
var: variable sampling rate - sampled whenever the value changes

Table 4.2: Sensor characteristics and collected measurements [VEL17]

Watch Measurements

The Pebble smartwatch has provided signals from the two available sensors, accelerometer and compass. Acceleration was sampled at 25Hz, while the compass does not have a constant sampling rate, but provided an update every time a change of more than one degree was detected.

Location Measurements

The ExtraSensory smartphone app samples location data utilizing the phone's location service, which does not have a constant sampling rate but provides an update every time it detects movement. Thus, the collected location measurements are time series of variable length, varying from a single time point up to more than twenty time points. Each location update contains the estimated location measurements (latitude, longitude, altitude, speed, vertical accuracy, horizontal accuracy) corresponding to a specific time reference.

In order to secure the subects' privacy, the app allows users to disguise a location (preferably their home) since the collected coordinates for instances annotated with the "at home" label could reveal their actual home location. Thus, for users who have opted to disguise their home, whenever they were in a 500 meter radius around it, no latitude and longitude coordinates were collected by the app, while all the other location measurements were collected as usual.

Low Frequency Measurements

These measurements are related to the smartphone's phone state (PS): app state (foreground/background), WiFi connectivity status, battery status (charging/discharging), battery level, phone call status, or are collected from other, built-in sensors, if available, such as: proximity sensor, ambient light, temperature, humidity, air pressure. These measurements were collected once in each recording window.

Audio Data

During each recording session of approximately 20sec, audio was recorded from the smartphone's microphone at a sampling rate of 22,050Hz, when the phone was not used for a call. Each raw audio signal was normalized to have a maximal magnitude of 1, and then Mel Frequency Cepstral Coefficients (MFCCs) were calculated on the phone, using half-overlapping windows of 2048 samples, 40 Mel-scaled frequency bands and 13 cepstral coefficients including the 0th coefficient. Only the normalizing factor and the MFCCs for each raw audio file were sent to the server, in order to ensure the subjects' privacy.

4.3.3 Feature Extraction

The publicly available ExtraSensory dataset also contains the following features, which are extracted from the raw signals collected by six particular sensors: accelerometer, gyroscope, watch accelerometer, location, audio, and phone state.

Accelerometer and Gyroscope

Since the users are free to use their smartphone in any way that is convenient to them and they are not obliged to keep it in a specific position, it cannot be assumed that it is oriented in any particular way. Thus, no particular meaning was attributed to any of the smartphone's axes and most of the features are extracted from the overall magnitude of the signal. The vector magnitude signal was calculated as the Euclidean norm of the 3-axis acceleration measurement at each time step. Afterwards, the following features were extracted:

- Nine basic statistics of the magnitude signal: mean, standard deviation, third moment, fourth moment, 25th percentile, 50th percentile, 75th percentile, value-entropy (entropy calculated from a histogram of quantization of the magnitude values to 20 bins), and time-entropy (entropy calculated from normalizing the magnitude signal and treating it as a probability distribution, which is designed to detect peakiness in time, in other words sudden bursts of magnitude).
- Six spectral features of the magnitude signal: log energies in 5 sub-bands (0-0.5Hz, 0.5-1Hz, 1-3Hz, 3-5Hz, >5Hz) and spectral entropy.
- Two autocorrelation features from the magnitude signal. The average of the magnitude signal (DC component) was subtracted and the autocorrelation function was computed and normalized such that the autocorrelation value at lag 0 will be 1. The highest value after the main lobe was located. The corresponding period in seconds was calculated as the dominant periodicity and its normalized autocorrelation value was also extracted.
- Nine statistics of the 3-axis time series: the mean and standard deviation of each axis and the 3 inter-axis correlation coefficients.

Watch Accelerometer

Since the smartwatch is positioned in a specific way, tightly worn around the users' wrist, meaning can be attributed to its axes. Thus, in addition to the same 26 features as above, 20 more features were extracted from the raw signals that were collected by the smartwatch accelerometer:

- Fifteen axis-specific features: log energies in the same sub-bands as above (0-0.5Hz, 0.5-1Hz, 1-3Hz, 3-5Hz, >5Hz) but calculated for each axis' signal seperately.
- Five relative-direction features: after calculating the cosine similarity between the acceleration directions of any two time points in the time series, these values were averaged in 5 different ranges of time-lag between the compared time points (0-0.5sec, 0.5-1sec, 1-5sec, 5-10sec, >10sec)

Location

The location features that were extracted are based only on relative locations, in order for the dataset to better generalize to any location and not be limited in the USCD campus area. The following features were extracted:

- Six features were calculated directly on the phone: standard deviation of latitude, standard deviation of longitude, change in latitude, change in longitude, average absolute value of derivative of latitude and average absolute value of derivative of longitude.
- Eleven more features were calculated remotely based on the transmitted location measurements: number of updates, log of latitude-range, log of longitude-range, minimum altitude, maximum altitude, minimum speed, maximum speed, best (lowest) vertical accuracy, best (lowest) horizontal accuracy and diameter (maximum distance between two locations in the recording session, in meters).

Audio

From the time-series of 13-dimensional MFCC vectors (typically about 400 time frames) the average and standard deviation of each of the 13 coefficients were calculated.

Phone State

Only the discrete smartphone states were used, and they were represented with a 26-dimensional one-hot representation. For each of the following properties, a binary indicator was used for each of the possible values, plus one indicator denoting missing data.

- App state (3 options: active, inactive, background)
- Battery plugged (3 options: AC, USB, wireless)
- Battery state (6 options: unknown, unplugged, not charging, discharging, charging, full)
- In a phone call (2 options: false, true)
- Ringer mode (3 options: normal, silent without vibration, silent with vibration)
- WiFi status (3 options: not reachable, reachable via WiFi, reachable via WWAN)

Regarding time-of-day information for each instance, its timestamp was used to extract the hour component (one out of 24 discrete values). Then, 8 half-overlapping time ranges were created: midnight-6am, 3am-9am, 6am-midday, 9am-3pm, midday-6pm, 3pm-9pm, 6pm-midnight and 9pm-3am. Each example's hour was represented with an 8-bit binary value, where exactly 2 bins will be active.

4.3.4 Label Correction

Since the annotation process is performed by the subjects while they are engaging in their natural behaviors, the reliability of the collected labels cannot be ensured. Forgetting or neglecting to report accurate labels during the day due to heavy workload or distractions might be a reason for that. In order to clean the collected data, some labels were altered to better reflect the activities that were performed or the context they were performed in, using two methods: based on location data and based on other labels.

Location-based Adjustment

It has been possible to verify whether many location context labels are accurate by checking the absolute location coordinates of the examples that had location measurements and by visualizing them on a map. This procedure facilitated the correction of falsely reported labels. For example, the "at the beach" label was removed from examples whose absolute location coordinates did not belong to a beach when they were visualized on a map and the same label was added to the examples whose absolute location coordinates belonged to a beach. A similar procedure was also followed for the "at home" and "at main workplace" labels, after determining the locations of the home and the main workplace of each subject, using the location coordinates of examples already annotated with these labels for each subject.

Label-based Adjustment

Labels have been added to examples or removed from examples by applying logical reasoning using the existing labels of each example. The existing labels of an example might reveal that a mistake was made while annotating, if the labels are not compatible with each other, or that a compatible label was omitted by mistake. For instance, the labels "walking" and "in a car" cannot coexist as labels in an example, since it is not possible to be in a car and walk at the same time, and thus one of the two labels should be removed for the labels to be consistent and as accurate as possible.

4.3.5 Classification Experiments using Logistic Regression

In [VEL17], a relatively simple context recognition system is implemented, using binary logistic regression (LR) classifiers, with a separate model corresponding to each context label. Each model receives the features described in Subsection 4.3.3 as input, and the real-valued output of the model is interpreted as an indication of the relevance of the specific conetext label. The goal is to detect the combination of relevant context labels for each input sample, that is, for each minute. Each input sample is treated independently and the sequence of minutes is not modeled as a time-series. These experiments using logistic regression are aimed at establishing a baseline for context recognition in-the-wild using the ExtraSensory dataset.

Sensor Segregation and Sensor Fusion

At first, in order to specify how insightful each sensor can be, single-sensor classifiers are used. The output of the logistic regression classifier is a continuous value that is interpreted as the probability of relevance (which will be useful in the next stage, sensor fusion). A binary decision about the relevance of each context label is made by applying a threshold of 0.5 to the continuous value. The procedure for single-sensor classifiers can be summarized in the following steps, for a given sensor s and a given context label l:

1. for each example of the dataset, compute the feature vector x_s (d_s-dimensional), according to Subsection 4.3.3
2. standardize each feature by subtracting mean and dividing by standard deviation (mean and standard deviation are calculated using the training set only)
3. learn a d_s-dimensional logistic regression classifier using the training set as input
4. apply the trained logistic regression classifier to a test example to obtain a probability value $P(y_l = 1|x_s)$ and a binary classification y_l

Balanced class weights were used to neutralize the imbalance between the positive and the negative class for each label. The weights were inversely proportional to the class frequency in the training set.

By taking advantage of prior domain knowledge, it is possible to correlate some sensors with corresponding labels, i.e., the watch accelerometer is probably useful in detecting activities that include hand motions. However, in order to reap the benefits from all sensors, sensor fusion is implemented in three alternative ways.

In *Early Fusion (EF)* classifiers, information from multiple sensors is combined prior to the classification stage. The procedure to build such a classifier for a given label l consists of the following steps:

1. starting from the sensor-specific feature vectors $\{x_s\}_{s=1}^{N}$, standardize each feature, and then concatenate the sensor-specific feature vectors into a single vector x of dimension $d = \sum_{s=1}^{N} d_s$
2. learn a d-dimensional logistic regression classifier using the training set as input
3. apply the trained logistic regression classifier to a test example to obtain a probability value $P(y_l = 1|x)$ and a binary classification y_l

In *Late Fusion classifiers (LF)* the predictions of the N single-sensor classifiers are combined. More specifically, the probability outputs $P(y_l = 1|x_s)$ are combined, so to benefit from the "confidence" of each classifier.

In *Late Fusion using Average Probability (LFA)*, the probability values from all the single-sensor classifiers are

averaged to obtain the final "probability" value. In other words, $P(y_l = 1|x_1, x_2, ..., x_N) = \frac{1}{N}\sum_{s=1}^{N} P(y_l = 1|x_s)$. By using this fusion technique, no further training is required after the single-sensor classifiers are learned. This technique practically assigns equal weights to all sensors, relying on the fact that more relevant sensors will hopefully output probabilities with higher confidence.

In *Late Fusion using Learned Weights (LFL)*, the sensor weights are learned from the data, by introducing a second layer of N-dimensional logistic regression, which takes the N probability outputs of the single-sensor classifiers as input, and the output is the final decision y_l. This way, sensors that are more informative for certain activites, thus certain labels, would end up with higher weights, and will contribute more to the corresponding decisions.

Classification Configuration

As mentioned above, an independent logistic regression model was trained for each context label. At first, the training set was internally partitioned in a training and a validation subset to run tests in order to determine the cost parameter C for logistic regression. For each value out of $\{0.001, 0.01, 0.1, 1, 10, 100\}$ a logistic regression model was trained on the training subset and tested on the validation subset, and the value of C was selected based on the highest F1-score on the validation subset.

Performance Evaluation

The performance of the classifiers was evaluated using five-fold cross validation, with each fold containing 48 users in the training set and 12 users in the test set. Moreover, leave-one-user-out (LOO) experiments were conducted.

Regarding performance metrics, in cases of imbalanced data, classification accuracy can be misleading, since for example for rare labels, a trivial classifier can achieve very high accuracy but is useless. A common approach is to observe recall against precision, or to calculate their harmonic mean (F1-score), but when averaging these metrics over labels, some labels will unfairly dominate the score. Moreover, precision and F1-score can be sensitive to dataset noise. However, the balanced accuracy, $BA = 0.5 * (TPR + TNR)$ can be used in these cases as a reliable objective that balances competing metrics.

A

Label	examples	Sensor	BA	Sensor	BA
Stairs - going up	399	Gyro	0.73	Acc	0.70
Stairs - going down	390	Gyro	0.73	Acc	0.71
Elevator	124	Gyro	0.76	Aud	0.71
Cleaning	1839	WAcc	0.71	Gyro	0.64
Laundry	473	WAcc	0.66	Acc	0.65
Washing dishes	851	WAcc	0.70	Aud	0.60
Singing	384	Aud	0.68	Loc	0.61
At a party	404	Aud	0.81	Acc	0.74
At the beach	122	Loc	0.72	PS	0.70
At a bar	520	PS	0.93	Gyro	0.66

C

	classifier	accuracy	sensitivity	specificity	BA	precision	F1
	p99	0.50	0.50	0.50	0.50	0.11	0.13
	Acc	0.73	0.64	0.73	0.68	0.17	0.22
	Gyro	0.70	0.64	0.69	0.66	0.16	0.20
	WAcc	0.73	0.67	0.72	0.70	0.18	0.22
	Loc	0.71	0.63	0.70	0.67	0.17	0.22
	Aud	0.75	0.65	0.75	0.70	0.18	0.22
	PS	0.76	0.74	0.76	0.75	0.20	0.24
	EF	0.87	0.67	0.87	0.77	0.24	0.30
	LFA	0.84	0.76	0.83	0.80	0.23	0.29
	LFL	0.85	0.76	0.85	0.80	0.24	0.30
	EF-LOO	0.86	0.69	0.86	0.78	0.24	0.30

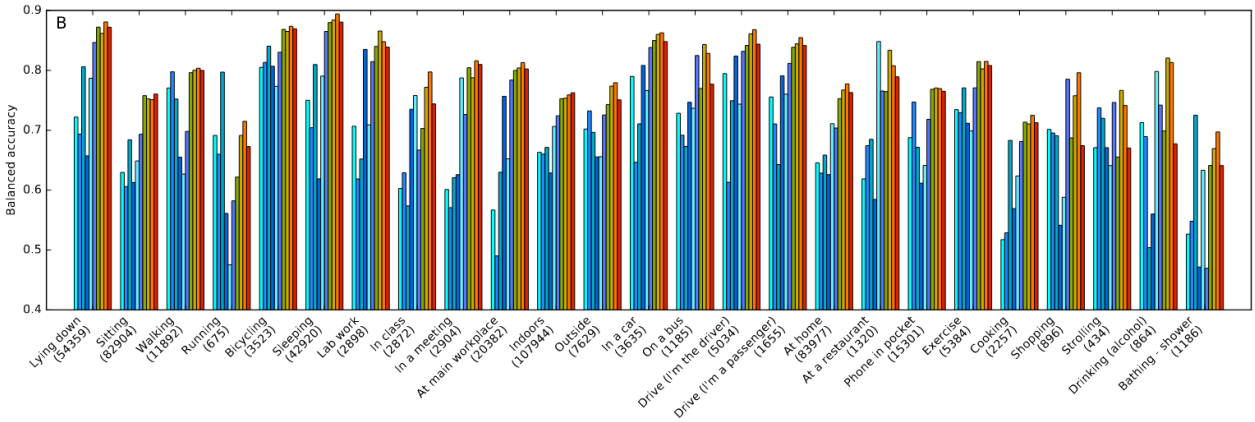


Figure 4.3.1: Performance of the LR single-sensor classifiers (Acc, Gyro, WAcc, Loc, Aud and PS) and sensor fusion classifiers (EF, LFA, LFL,EF-LOO) [VEL17]

Insightful results are presented in Figure 4.3.1. In subfigure (A), it is shown that for specific labels with few examples in the dataset, when utilizing the first or the second more intuitively relevant sensors, reasonable recognition of the labels can be achieved.

Regarding sensor fusion improvements, in subfigure (B) depicts BA scores for selected context labels (number of examples in parenthesis) for single sensor and sensor fusion strategies, according to the legend in subfigure (C). It is shown that in most cases sensor fusion performance matches the best single sensor system, and in many cases sensor fusion improves performance. This indicates that the system learns from data how to take advantage from different sensors, and that there is complementary information in different sensors. In subfigure (C), average performance metrics over the 25 labels from (B) can be seen. Sensor fusion methods perform better in average, with LFL slightly ahead.

In general, it is shown that having sensors of different modalities is useful for distinguishing between a large number of different activities and contexts. Different sensors can be helpful in recognizing different types of activities, and multiple sensor placements (smartphone, smartwatch) are helpful in different contexts. For instance, when in-the-wild conditions apply, in cases where the smartphone is placed on a surface, the smartwatch is highly needed to provide additional sensor data, to aid context recognition.

User Personalization

Since people act, behave, and use their phone in different ways, fine-tuning a model on user-specific data might be beneficial for context recognition. The researchers left out a single test user, and compared three models: (1) universal, trained on data from other users, (2) individual, trained on data from the same test user, and (3) adapted, which merges both, using LFA.

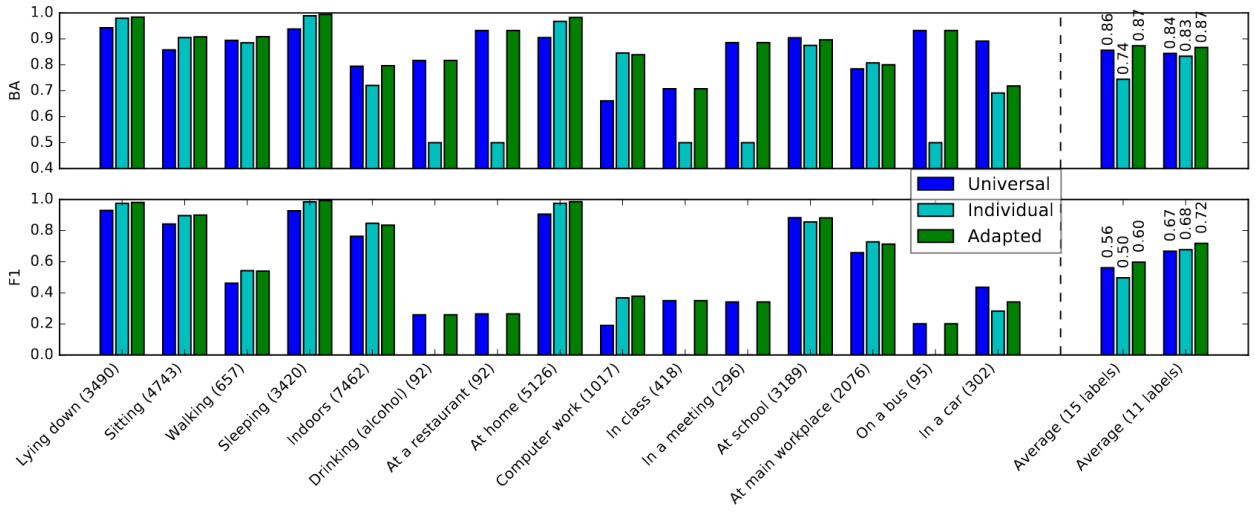


Figure 4.3.2: Performance of LR user adaptation for a single test user, comparing models: universal, individual and adapted [VEL17]

Figure 4.3.2 depicts the experimental results of this comparison. BA and F1-score metrics are calculated for each of 15 tested labels. The bars on the right side of the plot show the average scores over the 15 tested labels, and the average scores over 11 labels that had over 300 examples. The universal model demonstrates good performance. The individual model performed better than the universal one, on labels that had many individual examples, and poorly for the labels that had very limited examples. The user-adapted model shows improvement over the two aforementioned models in most labels as well as in overall recognition performance.

4.3.6 Classification Experiments using Multilayer Perceptron

In [VWL18], multi-task context recognition employing a Multilayer Perceptron (MLP) is introduced to improve recognition in-the-wild compared to the baseline described in the previous Subsection (4.3.5). The output of the MLP model is *multi-label*, meaning that multiple context labels can be predicted for each

input sample. In addition, by training with multi-label instance-weighting, the model can handle data with incomplete and unbalanced labeling. Moreover, by employing sensor-dropout, the model becomes robust to missing sensors. It should also be noted that this model enables transfer learning, and it can be used when collecting new data and extending the system to new behavioral aspects.

Multilayer Perceptron Configuration

The proposed recognition model is based on the MLP architecture, and is carefully adapted to fit the uncontrolled in-the-wild data. As previously mentioned, the model processes an input feature vector $x \in \mathbb{R}^d$ with a sequence of J affine transformations, each followed by an element-wise non-linear activation function. This allows all the sensor-features to be mixed together in a non-linear transformation to form a hidden representation, which is then shared to predict the labels in the last layer. In a multi-class MLP setting, multiple labels L can be classified simultaneously as positive.

For the hidden layers, a leaky rectified linear unit (ReLU) is used as activation: $g(v) = max[\frac{v}{10}, v]$. For the output layer, the logistic function (sigmoid) is used: $g(v) = \frac{1}{1+e^{-v}}$, to produce valid probability outputs. The actual binary predictions are extracted using a threshold of 0.5 on the continuous outputs.

We can define the MLP as a function $f : \mathbb{R}^{N \times d} \longrightarrow [0,1]^{N \times L}$, considering it processes a batch of N instances. This function is parametrized by the free parameters of the model; that are the weight matrices and bias vectors of the affine transformations:

$$\Theta = \{W_j, b_j\}_{j=1}^{J} \tag{4.3.1}$$

The training set is denoted with the feature matrix $X \in \mathbb{R}^{N \times d}$, the ground truth labels matrix is denoted as $Y \in \{0,1\}^{N \times L}$, and the missing-label matrix is denoted as $M \in \{0,1\}^{N \times L}$. To train the model, the following optimization problem is defined:

$$\min_{\Theta} \left(\frac{1}{NL} \sum_{i=1}^{N} \sum_{i=1}^{L} \Psi_{i,l} c(f(X)_{i,l}, Y_{i,l}) \right) + \lambda \varphi(\Theta) \tag{4.3.2}$$

For every instance i and label l, the entry's prediction cost is the traditional cross entropy loss:

$$c(\tilde{y}, y) = -(y log(\tilde{y}) + (1-y) log(1-\tilde{y})) \tag{4.3.3}$$

As a regularization term, $\varphi(\Theta)$ was selected to be the total Frobenius norm of the weight matrices of the model. This optimization problem is an instance-weighted version of maximum a posteriori probability (MAP) estimation, where $\varphi(\Theta)$ accounts for the prior.

The instance-weighting matrix Ψ is used to handle incomplete and unbalanced labeling. For entries (i,l) that are regarded as "missing label" ($M_{i,l} = 1$), $\Psi_{i,l}$ is set to zero, to make sure this example-label pair contributes nothing to the total cost. The other entries are normalized for each label l by their class (positive or negative), with weights that are inversely proportional to the frequency of that class for this label in the training set. As a result, for every label, the total contribution of the positive examples is equal to the total contribution of the negative examples.

Instance-weighting is very important for various reasons. First, without the weighting matrix, the learned model tends to be trivial - always declaring "no" for most labels. Second, the weighting matrix also incorporates the missing label information, which enables training a multi-task model when the data has incomplete labeling. In this way, instances with incomplete labeling can still contribute to the training, there is no need to throw them away.

In all the experiments, early fusion of the sensors was used (the 175 features from six sensors are used as input). Training was done using gradient descent with back-propagation, for forty epochs, with mini-batch size of 300 examples. The learning rate was linearly decreasing at every epoch, from 0.1 to 0.01. Momentum was used with weight 0.5.

Data Preparation

The same six core sensors are kept, the same d = 175 extracted sensor features are used, and each feature is standardized according to the mean and standard deviation estimated from the training set, as in the previous experiments (Subsection 4.3.5). Also, the same evaluation protocol is retained: five-fold cross validation using the same partition of the sixty participants to five folds.

In the previous experiments, every instance-label entry was treated as either positive or negative. In this work, the concept of "missing labels" is introduced in order to better represent cases where a relevant label was ignored or omitted unintentionally by a user. The original analysis assumed that whenever a label was not marked, it was not relevant to the instance. However, it might be more fitting to apply several common sense rules to infer when it is better to treat an entry as missing rather than negative. For example:

- In cases where no labels were used, all labels were marked as "missing" except for those which could be adjusted based on location, e.g., "at home".
- Subsets of labels that represent mutually-exclusive alternatives that typically cover all the possible options for a certain aspect were identified (e.g., for body posture/movement, thse are "Lying down", "Sitting", "Standing", "Walking", "Running", and "Bicycling"). For every example, all these label subsets were examined. If none of the labels in the set was selected, they were all marked as "missing" for this example.
- For the phone position label subset, in the cases where a participant reported two of the labels (e.g., "Phone in hand" and "Phone in pocket"), both labels were marked as "missing" since there was no safe way to determine which one was correct.
- For every participant, after determining the subset of labels that were used, all the other labels were marked as missing for all the participant's examples.

In this work, performance metrics are calculated by counting correct classification and errors only over non-missing entries.

Evaluation

The full multi-task MLP, which is trained with all the labels, is compared to the baseline system, referred as Early Fusion (EF) logistic regression in Subsection 4.3.5. While in the baseline system, the main results focused on 25 labels with successful recognition, here the multi-task MLP predicts L = 51 context labels simultaneously; the initial 25 labels are jointly modeled with 26 additional labels that got poorer results with the baseline system.

Different MLP architectures with none, one or two hidden layers of different widths are evaluated. To determine the hyperparameter values, the training set is partitioned to 70% internal-training-set and 30% internal-validation-set. The internal-training-set is used to train the MLP, performing a grid search over possible hyperparameter values. The hyperparameter values are selected based on the highest internal-validation-set balanced accuracy, and are subsequently used to re-train the model over the entire training set.

The 70%/30% partition of the training set was done randomly, because there was no way to guarantee the same positive/negative ratio for all the 51 labels. In the experiments, the depth of the MLP is fixed, but the grid-search is done to select both λ among $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$, and the dimension (width) of the hidden layers, among $\{2, 4, 8, 16, 32\}$.

Recognition performance of the MLP with regard to layer size and depth is presented in Table 4.3, along with the inital baseline results of logistic regression (LR) per label. The "size" column corresponds to the number of free parameters for each model, including weight matrices and bias vectors. For the MLP, the dimensions of the hidden layers are put in parenthesis. For MLP (d) and MLP (d,d), where the hidden dimension or dimensions are selected via internal validation, the architecture can be different for each test fold, so the model size is not defined. Scores are averaged over all the 51 labels. The last two columns present the BA measured on the training examples (train-BA) and the gap between training and test performance (BA gap), to investigate the level of over-fitting.

Recognition scores reported for LR and for the multi-task MLP							
Model	**Size**	**Accuracy**	**Sensitivity**	**Specificity**	**BA**	**Train-BA**	**BA gap**
LR	8976	0.832	0.597	0.838	0.718	0.875	0.158
MLP (linear)	8976	0.760	0.746	0.757	0.752	0.813	0.061
MLP (2)	505	0.666	0.773	0.661	0.717	0.735	0.017
MLP (4)	959	0.730	0.773	0.727	0.750	0.773	0.023
MLP (8)	1867	0.776	0.768	0.775	0.772	0.806	0.035
MLP (16)	3683	0.781	0.755	0.781	0.768	0.820	0.052
MLP (32)	7315	0.799	0.736	0.800	0.768	0.847	0.079
MLP (64)	14579	0.806	0.687	0.808	0.747	0.865	0.118
MLP (d)	?	0.799	0.736	0.800	0.768	0.847	0.079
MLP (2,2)	511	0.662	0.759	0.656	0.707	0.736	0.029
MLP (4,4)	979	0.707	0.769	0.707	0.738	0.763	0.025
MLP (8,8)	1939	0.761	0.772	0.759	0.766	0.803	0.037
MLP (16,16)	3955	0.773	0.773	0.773	0.773	0.817	0.044
MLP (32,32)	8371	0.805	0.729	0.807	0.768	0.845	0.078
MLP (64,64)	18739	0.817	0.661	0.823	0.742	0.877	0.135
MLP (d,d)	?	0.805	0.729	0.807	0.768	0.845	0.078

Table 4.3: Recognition scores reported for logistic regression (LR) per label, and for the multi-task MLP (linear or with the hidden layers' dimensions in parenthesis) [VWL18]

As it can be seen in Table 4.3, the switch from LR to the linear MLP substantially improves sensitivity but we observe a drop in specificity. However the elevated BA reflects the overall improvement, which can be also attributed to less overfitting, as seen in the "BA gap" column. In the MLP model, a single value has been used for the hyper-parameter λ for all labels, unlike LR, where the hyper-parameter C_{logist} was tuned separately for every label.

MLPs with hidden layers have the integrated advantages of non-linearity and dimensionality reduction. A bottleneck hidden layer can greatly reduce model size, e.g. from the 8976 parameters of the linear model, to 3683 parameters for an MLP with a single hidden layer of sixteen nodes ($175 \times 16 + 16$ for the hidden layer and $16 \times 51 + 51$ for the ouput layer). A smaller model can be seen as an optimization constraint, which can be beneficial to avoid overfitting. As it can be seen in the table, it seems that the smaller the model size, the smaller the BA train-test gap column, this the model is less prone to overfitting.

It can be said that in general, MLP manages to capture relatively accurate predictive mappings from sensor features to many diverse labels, in concise representations. Selecting a moderately-sized architecture can balance the trade-off between sensitivity and specificity, and optimize the MLP performance. In summary, modeling all the labels with a shared structure, contrary to the LR per label, has been shown to improve the recognition performance.

Additional Experiments

Further evaluation was conducted to examine how each technique employed by the system contributes to the results. The crucial role of instance-weighting was proved by training LR and MLP models without it, and it was seen that the resulting models optimize the raw accuracy. In this way, the rare cases (positives) are neglected and this results in almost-trivial classifiers in this multi-task setup.

Moreover, separate MLPs per label were trained to assess whether non-linearity and hidden layers were the main contributing factors to the improvement of the multi-task MLP over the LR baseline. However, these MLPs per label performed roughly at the same level as the baseline and, also, overfitting was noticed. At the same time, when comparing the MLPs per label with a multi-task MLP with similar size, it was seen that the latter performs better, which is a strong indicator for the beneficial influence of sharing parameters.

Another factor that should be taken into consideration is the dimensionality reduction using a hidden layer with smaller dimension than the input features. By evaluating a pipeline of simply applying PCA to the

input features before training a linear MLP with the dimensionally-reduced features, it was shown that dimensionality reduction is not enough on its own to provide the full performance gain.

Regarding transfer learning scenarions where the model is extended or repurposed to include new labels, it was shown that it is beneficial to share the model among labels. The shared model helps boost recognition of the new labels, and when the parameters of the hidden layers are frozen and only the parameters of the output layer are finetuned using the new-labels-only dataset, the recognition performance of the model regarding the new labels can be nearly as good as the performance of the model that included the "new labels" in the initial training set.

Regarding the robustness issues that naturally arise in real-world scenarios with missing sensors, it was shown that, since some sensors are very important for accurate recognition, it is useful to adopt a sensor dropout strategy to retain recognition performance and be more resilient in cases of missing sensors.

Regarding the explainability and interpretability of the model, it is not always intuitive or straightforward to understand, and it is not easy to identify which specific features help the most, due to the architecture and the non-linearity. An analysis was performed by selecting subsets of the labels and the sensors, to identify which sensors mostly help in the recognition of different activity or context labels or settings. In total it was seen that several sensors play an important role in the recognition of logically "relevant" labels.

Interpreting the multi-task MLP was also attempted on a node basis when studying a small architecture with two hidden layers of two nodes (MLP (2,2)). The activations of hidden nodes were systematically observed, to examine what kind of examples cause a node to have a high activation value in an attempt to characterize the "meaning" of each node. This was mainly focused on features for which there is a strong contrast between the low-activation and high-activation examples, since these give indications about what each hidden node is sensitive to. From these observations it was possible to approach the MLP in a more human-interpretable way.

Finally, the multi-task MLP was validated on another HAR dataset to verify its performance. The "Activities of Daily Living (ADL)" dataset [PR12] was used, and it was validated that the multi-task MLP with one or two hidden layers can outperform a linear MLP (and also the SVM), while the same trends regarding model size were also observed with this dataset.

4.3.7 Other Works and Publications on the ExtraSensory Dataset

Since the ExtraSensory dataset has been the first publicly available in-the-wild dataset for HAR using smartphone and smartwatch sensors, it has been widely used in the HAR community, to develop and evaluate human activity/context recognition models and for other tasks. A summary of publications using the ExtraSensory dataset to train and evaluate HAR models can be found in Table 4.4. As it can be seen, plenty of models have been assessed for human activity recognition using this dataset, including traditional Machine Learning techniques, e.g. Decision Trees, Random Forests, k-Nearest Neighbors, and SVMs, as well as Deep Learning models based on CNNs and RNNs. In addition, unsupervised methods, such as clustering, have been employed to tackle HAR tasks.

As we can see in Table 4.4, the number of activities to be recognized differs in each work. In many works the goals is to recognize Activities of Daily Living (ADL) in general, while in others a subset of activities, mostly stationary/dynamic are the recognition target. In some works, multiple labels are merged, and the goal is to recognize composite activities. In addition, in most papers the already extracted features from [VEL17] are used, while in [Sou+23] features are extracted after converting the raw accelerometer, gyroscope and magnetometer signals to the frequency domain. Fewer papers use the raw accelerometer, gyroscope and magnetometer signals directly as input to the model, exclusively or complementary to the statistical features.

The ExtraSensory dataset has also found use in data generation tools which generate synthetic sensor data for HAR, such as the HAR-CTGAN [DeO+22]. *In-the-wild* data collection offers benefits when trying to model and reproduce real-world sensor signals, since the collected signals are more representative of real-world scenarios, and the synthesized results of data generation are more realistic. In addition, ExtraSensory has been widely used in studies regarding techniques for imputating missing data, such as [Huo+22] and [SOL18], where imputation is performed on-the-fly and is based on sensory and temporal relatedness using a

denoising or adversarial autoencoder. In [Ale+22], instead of imputating missing data, a data augmentation method is proposed, which generates all possible combinations of utilized sensors for recorded observations.

Since data collected in a realistic setting often contains significant label noise, it is crucial to develop techniques to mitigate the impact of noisy labels on model training. ExtraSensory has been used in a study where VALERIAN [HWZ23], an invariant feature learning method, is proposed, leveraging self-supervised learning and multi-task learning to create robust and personalized representations of HAR sensor data.

The ExtraSensory dataset in HAR models

Tools	**Sensors**	**Raw Signals**	**Extracted Features**	**# Labels**	**Application**	**Person-alized**	**Paper**
Logistic Regression	Acc, Gyro, WAcc, Loc, Aud, PS	✗	✓	51	ADL	✗, ✓	[VEL17]
DT, RF, BG k-NN, NB	Acc	✗	✓	6, 15**	Dynamic	✗	[Asi+20]
DT, RF, k-NN, BN	Acc, WAcc	✗	✓	5, 29**	Dynamic	✗	[HA20]
RF, k-NN	Acc, Gyro, WAcc, Loc, Aud, Comp	✗	✓	5	Dynamic	✗	[Tee+22]
XGBoost, AdaBoost, Boosted C5.0	Acc, Gyro, Mag, WAcc, Comp	✗	✓	5	Dynamic	✗	[TB21]
NN, SVM	Acc, Gyro, Mag, WAcc, Comp	✗	✓	5	Dynamic	✗	[TB21]
SVM	Acc	✗	✓	6, 15**	Dynamic	✗	[Asi+20]
SVM	Acc, Gyro, WAcc, Loc, Aud, Comp	✗	✓	5	Dynamic	✗	[Tee+22]
MLP	Acc, Gyro, WAcc, Loc, Aud, PS	✗	✓	51	ADL	✗	[VWL18]
MLP (+new loss)	Acc, Gyro, WAcc, Loc, Aud, PS	✗	✓	51	ADL	✗	[Li+19]
MLP	Acc, Gyro	✗	✓	5	Dynamic	✓	[Cru+20b]
MLP	Acc, Gyro, Mag, Comp, WAcc, Aud, PS	✗	✓	16**	ADL	✗	[Ale+22]
Hier. DNN	All	✗	✓	6	Dynamic	✗	[Faz+21]
Hier. DNN	Acc, Gyro, Mag, Comp, Grav, Loc, PS	✗	✓	6	Dynamic	✗	[Sou+23]
(CNN)-LSTM, w/ or w/o Att	All	✗	✓	51	ADL	✗, ✓	[CR19]
LSTM w/ Hier. Att	All	✗	✓	51	ADL	✗	[Che+20]
CNN-GRU w/ Att	All	✓	✗	51	ADL	✓	[She+22]
CNN	Acc, Gyro, Aud, PS	✓	✗	51	ADL	✗	[Sae+19]
Pre-trained CNN & RF	Acc, Gyro, Aud	✓	✓	51	ADL	✗	[Cru+20a]
MLP & CNN-QLSTM	Acc, Gyro, WAcc, Loc, Aud, PS, LF	✓	✓	51	ADL	✗	[GA22]

GCNN	All	✗	✓	3, 5, 10, 25	ADL	✗	[Moh+22]
CNN & WTSVM	All	✓	✗	96	ADL	✗	[Chu+23]
HHGNN	Acc, Gyro, Grav, Loc, Aud, PS	✗	✓	17	ADL	✓	[Ge+23]
AAE & NN	Acc, Gyro, WAcc, Loc, Aud, PS	✗	✓	51	ADL	✗	[SOL18]
LSTM-DAE with k-NN & XGBoost	Acc, Gyro, Mag, Loc	✓	✓	16**	Dynamic	✗	[Huo+22]
Clustering	All	✗	✓	2049*	ADL	✓	[CD21]
CNN & Knowledge Model	Acc, Gyro, WAcc, Loc, PS, LF	✓	✓	7	Dynamic	✗	[ACB23]

DT: Decision Tree, RF: Random Forest, BG: Bagging, k-NN: k-Nearest Neighbors, NB: Naïve Bayes
BN: Bayes Net, Att: Attention, Hier. Att: Hierarchical Attention, Hier. DNN: Hierarchical DNN
QLSTM: Quaternion LSTM, GCNN: Graph CNN, WTSVM: Weighted Twin SVM
AAE/DAE: Adversarial/Denoising Autoencoder, HHGNN: Heterogeneous Hyper-Graph NN
* multiple labels per instance merged in one
** specific combinations of labels merged in one

Table 4.4: Papers using the ExtraSensory dataset for HAR tasks

ExtraSensory has also been used in Transfer Learning scenarios. For example, in [Fat21], an LSTM-CNN model pretrained on ExtraSensory (which is mostly populated by younger adults) is used for activity recognition of older adults' smartwatch data. Although further improvement is required to achieve satisfactory recognition performance in this case, it is a promising real-life case with many applications on the well-being and health of the elderly. Moreover, in [She+23b], cross-individual HAR is tackled via a Tranfer Learning approach which identifies the approximately optimal source individuals by ranking interpretable meta-features using a linear scoring function. Additionally, in [Cru+20b], a clustering-based semi-population approach is adopted to select the optimal subset of users to be used to pre-train the model, before adapting it to the target user, for better personalization.

Furthermore, strategies for efficient HAR have been developed and evaluated using ExtraSensory. [AFB23] leverages Dissimilarity-Based Query Strategy (DBQS), an Active Learning-based approach which introduces selective sampling to find the informative and diverse samples in the sensor data to train the model. In this way, the recognition performance improves, while the model requires less training data. A similar concept based on Active Learning and Conditional Mutual Information is employed in [AT19] to optimize training batches and minimize the need for data annotation in HAR. Also, in [NPG18], Active Learning has been leveraged for a novel oversampling method in order to overcome class imbalance. Another algorithm for dynamic feature selection has been introduced in [Ard+20], where feature selection is formulated as an l_0 minimization problem across time, and the combinatorial optimization problem is cast into a stochastic optimization formulation. A differentiable relaxation is used to make the problem amenable to gradient-based optimization.

Another challenging HAR task, especially for health applications, is the "early" classification problem, in which a label set must be assigned to the time series before the series is entirely observed. ExtraSensory has been used to evaluate a potential solution to this problem [Har+20b] that includes a Recurrent Halting Chain (RHC), a combination of RNNs and a Reinforcement Learning-based halting network. At each timestep of the time series, RHC uses a transition model to represent both complex temporal dynamics and conditional dependencies between labels as they are progressively predicted, while a halting policy network reads the hidden state at each timestep and decides whether or not each label prediction should be returned as final.

4.4 Other In-the-wild Datasets

The ExtraSensory dataset has paved the way for the creation and release of more wearable devices-based HAR datasets collected *in-the-wild*, which satisfy the four conditions described in 4.2. A summary of the publicly available such datasets so far, to the best of our knowledge, is presented in Table 4.5 (the ones denoted as not open in the Table, appear to be available with restricted access or temporary unavailable or correspong to broken website/download links). These datasets are ADL-oriented and capture realistic activity sequences from everyday life, captured by smartphone and smartwatch sensors.

Public HAR datasets based on wearable sensors, collected *in-the-wild*

Dataset	Type	Subjects	Labels	Samples	Devices	IMU	Open	Ref
ULSTER HAR	ADL	10	10*	n/a	SP, SW	✓	✗	[Cru+19]
unnamed	ADL	n/a	16	n/a	SP	✓	✗	[Bra+20]
ETRI	ADL	22	>100	n/a	SP, SW	✓	✓	[Chu+22]
SmartJLU	ADL	50	23	30,000	SP	✓	✗	[She+22]
SmartUnitn2	ADL	158	55	139,239	SP	✗	✗	[Giu+22]
MarSense	ADL	7	6	5,047	SP	✓	✓	[Giu+22]
SAMoSA	ADL	20	27	n/a	SW	✓	✓	[Mol+22]
unnamed	Motionless Activities	25	3	6,000	SP	✓	✓	[Pir+22]
LifeSnaps	ADL	71	n/a	>1M	SW	✗	✓	[Yfa+22]
IDLab	ADL	18	5	394,318	SW	✓	✓	[Sto+23]

SP: Smartphone, SW: Smartwatch
* subjects were also able to add their custom activity labels

Table 4.5: Public HAR datasets based on wearable sensors, collected *in-the-wild*

It should also be added that from the datasets that we have inspected, some satisfy all in-the-wild conditions but one. DOMINO [Arr+23] does not satisfy condition 4: natural behavioral content, since during the data collection process, the participants were instructed to perform specific sequences of tasks. In the dataset collected in [BAT22] and in the HARTH dataset [Log+21], condition 2: unconstrained device placement is not met, since, although participants perform all activities in free-living conditions, the sensors used were placed in a specific -and realistically unnatural- way (chest-mounted smartphone, accelerometer in thigh and back, and chest-mounted camera). Also, regarding the WASHSensory dataset [Man+19], it is not explicitly stated to be proprietary or publicly available, but no digital footprint of it was found online.

Chapter 5

Experiments and Results

In this Chapter, we present our work on human activity and context recognition based on wearable sensors and *in-the-wild* data collection. Our work is based on the ExtraSensory dataset, which is very rich in content and labels. We perform a short dataset analysis, followed by machine learning and deep learning experiments aimed to improve recognition performance. We provide rich visuals including model architecture diagrams and activity plots, and we take a shot on model interpretability.

5.1 Introduction

Our work focuses on investigating HAR approaches on a free-living setup, and evaluating which features and architectures work best, while also considering model efficiency/size and explainability. To this end, our pursuit has included the following steps along the way:

- At first, it is necessary to explore and visualize the ExtraSensory dataset in order to better understand it and devise ways to handle the challenges it poses. We investigate how many examples exist per user and per label, how the features change when performing different activities or when different users perform the same activity, and how many missing sensor data we have to handle.
- Afterwards, we reproduce some baseline prediction models from previous work, which are already mentioned in the previous Section: logistic regression and an MLP, using the pre-extracted features.
- We build upon this work, and use a bidirectional LSTM (BiLSTM) to model a sequence of examples, again using the pre-extracted features. We also augment the BiLSTM model with a Self-Attention module, or a Cross-Attention module which is used for interpretability.
- We also run experiments using the raw sensor data, using convolutional neural network (CNN) layers and Transformer Encoder layers for feature extraction, to model a single example, and we further combine them with a BiLSTM to model a sequence of examples.

The ExtraSensory dataset poses some challenges that are addressed in this Chapter. It contains over 100 labels, out of which we use 51 as in the original work, and it is extremely unbalanced since the frequency of each label is commensurate with the frequency of the corresponding activity or context in real life. In addition, each example is labeled with more than one label, since many labels might be relevant at each minute. Another major challenge is that the dataset contains a significant portion of unlabeled data, and also a lot of wrongly annotated data, due to misusing or forgetting labels while self-reporting in everyday life, which is much harder than labeling scripted sequences of activities during data collection in a lab. Also, collecting data using different types of devices combined with unconstrained device placement lead in data prone to noise, and missing sensors modalities are very common. Also, there is large inter-personal and intra-personal variability in the collected data since an activity might be performed differently among users, but also by a specific user at different times.

In order for all our results to be comparable with each other and with the baselines that are introduced in previous works, we use the same five-fold cross validation scheme, with 48 users in the training set and 12 users in the test set in each iteration. This means that the test set always contains data only from unseen users, a.k.a. users that are not "seen" by the model during training. This makes the task of activity recognition even more difficult, because of the variability among users, and, in addition, we cannot use established personalization methods, such as a user embedding, in a standard way. However, we could use a fine-tuning approach based on the user of the training set that is closer to the current test set user, based on some metrics. Nevertheless, in this thesis we want to see how much we can improve the performance of universal models, and we only test their performance on unseen users in a cold-start mode, without fine-tuning or adapting them to each unseen user.

5.2 Dataset Exploration

At first, we try to explore the ExtraSensory dataset and unravel its noisy and unbalanced nature, resulting from data collection *in-the-wild*. As already mentioned in Chapter 4, the dataset contains more than 100 activity and context labels annotated by the subjects, and in the curated data published by Vaizman *et al.* in

their work, 51 labels are included. In order to make our lives easier, and create meaningful and interpretable plots in this Section, we group these labels in an intuitive manner, as presented in Table 5.1.

ExtraSensory labels grouped conceptually	
Concept	**Labels**
Posture/Movement	"Lying down", "Sitting", "Standing", "Walking", "Running", "Bicycling"
Special Movement	"Strolling", "Stairs - Going up", "Stairs - Going down", "Elevator"
Phone Location	"Phone in pocket", "Phone in hand", "Phone in bag", "Phone on table"
Work-related	"In class", "Lab work", "Computer work", "In a meeting"
Location-based	"At home", "At school", "At main workplace", "At a restaurant", "At a bar", "At a party", "At the gym", "At the beach"
Transportation	"In a car", "On a bus", "Drive - Driver", "Drive - Passenger"
Chores	"Shopping", "Cooking", "Cleaning", "Doing laundry", "Washing dishes"
Self-care	"Bathing - Shower", "Toilet", "Grooming", "Dressing", "Sleeping"
Leisure Time	"Exercise", "Eating", "Drinking alcohol", "Watching TV", "Surfing the internet", "Talking", "Singing"
Companion	"With co-workers", "With friends"
Environment	"Indoors", "Outside"

Table 5.1: Intuitive grouping of activity and context labels of the ExtraSensory dataset

5.2.1 Users and Labels

Our exploration starts by visualizing all 60 ExtraSensory dataset participants, denoted from here on as users (u00-u59), and all 51 labels included in the "Primary Data" directory of the dataset. In Figure 5.2.1 we get a first, qualitative analysis, visualizing which users have collected data for which labels. As we can see, no user has collected data for all labels. The users with the most collected labels have used up to 41 out of 51 labels, while the users with the least used labels have used only 13 out of 51 labels. This is a direct consequence of an *in-the-wild* setting, where the users are not instructed or expected to perform specific activities or sequences of activities, and is also logical from the perspective that in a real-life setting, the user might forget or disregard updating the labels in the app. However, as we can see, this often also leads to errors or non-accurate label annotation, since in many cases, some labels were omitted (e.g. some users have no "Indoors" or no "Outside" labels but that is impossible).

Regarding the Figure's column, we can easily see that only 3 labels, namely "Standing", "Walking" and "Sitting" were used by all users. In total, 20 labels were used by 40 or more users, 39 labels were used by 20 or more users, while only 6 labels, namely "At a party", "Lab work", "Singing", "At the beach", "At the gym" and "At a bar", were used by less than 10 users. As expected, the rarer labels appear less in the dataset, however it is unclear whether a lot of missing labels for many users can be attributed to the users not performing these activities or to omissions in label annotation (for example, only 30 users have used the "Cleaning" label and only 18-19 users the "Stairs - Going Up/Down" labels respectively).

We proceed to a more thorough, quantitative analysis of all examples collected by all users, presented in Figure 5.2.2, where the total number of examples annotated with each label for each user is depicted. As expected, the users' time is very unequally distributed across labels, since in everyday life, some activities and contexts occupy most of the person's time, such as "Sleeping", "Sitting", "At home", while others, especially chores or leisure time activities last much less. From this Figure it can also be seen that there are substantial differences in the use of different labels by different individuals. We observe that, even though the most "popular" labels overall are the most populated for nearly all users, each user distributes her/his time in a personalized way.

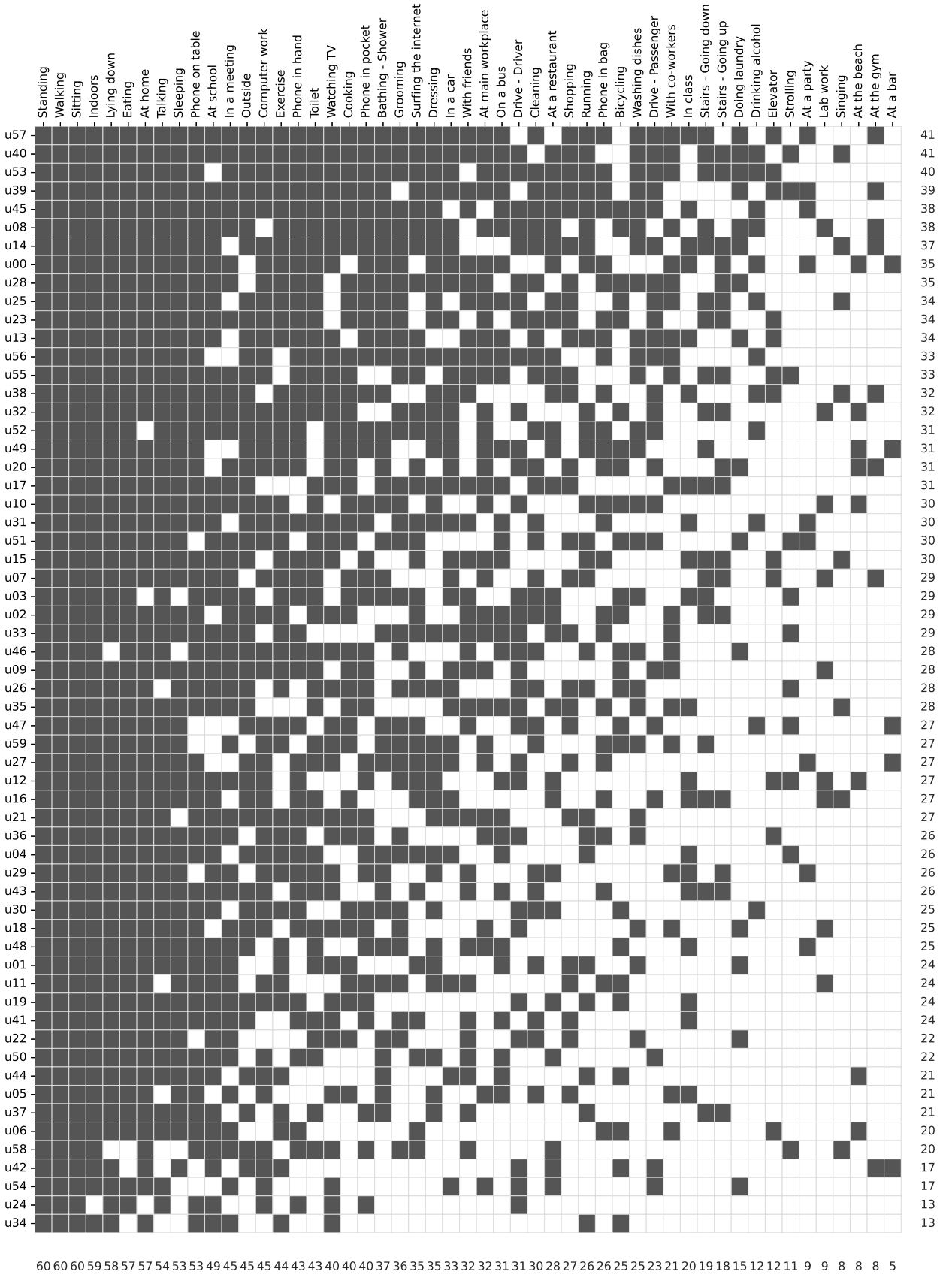


Figure 5.2.1: Qualitative visualization of context labels per subject

	Lying down	Sitting	Standing	Walking	Running	Bicycling	Strolling	Stairs - Going up	Stairs - Going down	Elevator	In a car	On a bus	Drive - Driver	Drive - Passenger	Indoors	Outside	
u00	3490	4743	512	657	2	0	0	1	0	0	302	95	0	0	7462	0	17264
u01	3264	2500	1054	302	4	0	0	0	0	0	0	43	0	0	5994	0	13161
u02	1874	2215	801	468	0	126	0	4	4	0	0	42	189	0	2731	52	8506
u03	2742	2521	259	273	0	92	4	0	2	0	67	0	158	0	1859	15	7992
u04	1799	3441	980	925	47	0	124	0	0	0	36	32	0	0	5655	848	13887
u05	657	1106	193	163	0	0	0	0	0	0	0	49	0	0	1284	0	3452
u06	2319	2972	576	384	0	136	0	0	0	23	0	0	0	0	3954	0	10364
u07	585	685	296	95	4	0	0	2	2	35	154	0	0	0	810	0	2668
u08	2849	2472	1207	800	88	56	0	0	1	0	250	16	202	0	4009	73	12023
u09	1810	1810	570	454	0	150	0	0	0	0	210	0	276	20	2157	506	7963
u10	2096	1004	763	298	10	413	0	0	0	0	0	0	55	60	3153	464	8316
u11	4022	2968	659	218	0	109	0	0	0	0	157	0	0	0	4333	0	12466
u12	3114	4539	143	289	0	0	1	0	0	1	0	1	216	0	7261	345	15910
u13	2554	2961	1010	714	13	0	0	0	0	1	0	65	0	217	4893	584	13012
u14	725	2577	454	205	36	0	0	6	12	0	59	0	169	1	3080	17	7341
u15	469	2523	122	402	14	0	0	231	222	45	5	1	0	0	2808	422	7264
u16	1554	3153	486	765	0	0	0	30	30	0	165	0	0	40	4021	23	10267
u17	1989	3089	866	1532	0	0	0	78	87	0	42	54	0	0	4549	546	12832
u18	1284	1312	620	132	0	0	0	0	0	0	0	0	265	0	2827	173	6613
u19	2378	1359	402	294	95	122	0	0	0	0	0	0	263	0	2929	190	8032
u20	2946	1694	637	164	0	62	0	3	0	0	432	0	430	2	3799	128	10297
u21	2548	3044	621	1181	8	0	0	0	0	0	20	141	0	0	2061	1184	10808
u22	1261	1010	456	107	0	0	0	0	0	0	0	0	48	0	2616	0	5498
u23	2411	3171	428	588	0	1	0	53	53	35	907	0	1012	47	3663	1033	13402
u24	516	1096	323	223	0	0	0	0	0	0	0	0	147	0	0	0	2305
u25	102	1728	978	363	0	59	0	86	88	0	0	30	410	10	631	99	4584
u26	2500	2627	918	15	88	692	17	0	0	0	81	0	73	0	5448	8	12467
u27	1939	207	2085	175	0	0	0	0	0	0	388	0	425	0	2652	120	7991
u28	3036	2765	1398	275	0	362	0	1	0	0	336	0	277	59	4506	0	13015
u29	2412	2735	617	662	0	0	0	27	0	0	0	0	0	0	4867	0	11320
u30	2541	2648	820	352	0	27	0	0	0	0	0	0	280	0	3287	73	10028
u31	1137	816	685	16	0	0	0	0	0	0	57	37	0	0	1880	91	4719
u32	2724	2246	158	187	33	120	0	1	1	0	117	0	108	34	2759	14	8502
u33	352	2420	501	507	0	0	148	0	0	0	388	86	424	0	2451	299	7576
u34	214	1766	177	397	283	2	0	0	0	0	0	0	0	0	1469	0	4308
u35	25	956	101	156	0	0	0	0	0	0	35	17	35	0	237	120	1682
u36	2573	4226	1079	261	2	0	0	0	0	1	0	35	239	0	2339	0	10755
u37	1145	2253	284	190	7	0	0	25	8	0	0	0	0	0	2177	77	6166
u38	1336	1916	384	132	0	0	0	0	0	1	342	0	0	409	2487	127	7134
u39	1242	2138	778	268	113	0	69	0	0	15	79	46	0	38	2988	128	7902
u40	748	1543	231	158	1	0	6	1	1	0	30	42	35	20	1455	126	4397
u41	1110	557	92	28	0	0	0	0	0	0	0	63	0	0	1470	22	3342
u42	2546	3706	1580	193	0	290	0	0	0	0	0	0	505	23	6464	205	15512
u43	1115	1184	379	207	0	0	0	192	192	0	0	46	0	0	1948	278	5541
u44	233	1561	399	236	9	369	0	0	0	0	76	2	0	0	714	74	3673
u45	2747	3880	1667	494	68	540	0	0	0	0	0	26	218	75	7903	1062	18680
u46	0	1301	370	375	35	106	0	0	0	0	0	9	136	0	648	44	3024
u47	2297	2143	395	78	0	579	219	0	0	0	0	0	557	25	2580	219	9092
u48	1522	2731	1201	234	0	232	0	0	0	0	0	60	0	0	2225	32	8237
u49	1856	1492	1170	265	37	60	0	0	1	0	433	24	422	0	2536	169	8465
u50	3432	4488	312	658	0	0	0	0	0	0	0	170	0	183	4477	0	13720
u51	577	946	454	187	12	47	2	0	0	0	0	30	0	68	624	2	2949
u52	1394	3467	467	412	1	0	0	0	0	0	298	0	0	298	1591	248	8176
u53	1628	2936	973	437	30	0	0	27	37	9	388	307	240	602	2680	357	10651
u54	339	1265	114	117	0	0	0	0	0	0	26	0	118	95	442	0	2516
u55	1206	3485	223	1141	0	0	204	30	27	8	16	43	0	0	4876	630	11889
u56	2733	3213	942	613	0	0	0	0	0	0	66	121	43	106	4648	425	12910
u57	3696	3668	1053	506	50	0	0	0	3	26	76	61	0	94	8093	442	17768
u58	0	621	2	61	0	0	12	0	0	0	0	0	0	0	477	20	1193
u59	497	757	357	147	0	268	0	0	3	0	45	0	0	0	755	0	2829
	104210	136356	37782	22136	1090	5020	806	798	774	200	6083	1794	7975	2526	184692	12114	

Figure 5.2.2: (a) Quantitative visualization of number of context labels per subject - Part 1 Including label subsets “Posture/Movement”, “Special Movement”, “Transportation”, “Environment”

	Phone in pocket	Phone in hand	Phone in bag	Phone on table	In class	Lab work	Computer work	In a meeting	At home	At school	At main workplace	At a restaurant	At a bar	At a party	At the gym	At the beach	With co-workers	With friends	
u00	931	1081	887	5717	418	0	1017	296	5151	3189	2085	92	299	297	0	28	1353	1500	24341
u01	0	0	0	34	0	0	0	4	6404	7	0	0	0	0	0	0	0	0	6449
u02	0	0	62	2935	0	0	1336	507	2729	0	1097	72	0	0	0	0	717	200	9655
u03	90	202	0	943	447	0	0	5	0	732	0	20	0	0	0	0	0	31	2470
u04	871	587	0	4082	361	0	1657	0	4842	841	0	0	0	0	0	0	0	0	13241
u05	0	0	0	1594	214	0	89	306	995	0	197	0	0	0	0	0	331	0	3726
u06	0	1480	467	3084	0	0	0	13	3040	2016	0	0	0	0	0	236	13	0	10349
u07	403	15	0	730	0	123	380	23	715	43	335	0	0	0	42	0	0	0	2809
u08	1958	164	0	3987	0	1351	0	81	3645	1984	1698	145	0	0	98	0	437	0	15548
u09	1865	113	0	2286	0	21	635	165	2052	27	1132	0	0	0	0	0	85	55	8436
u10	449	0	5	1588	0	362	52	47	1966	48	730	0	0	0	0	94	0	0	5341
u11	620	0	988	4031	0	982	1201	131	2219	1341	0	0	0	0	0	0	0	2572	14085
u12	1311	399	0	5748	661	525	2639	5	5816	1351	0	89	0	0	0	15	0	0	18559
u13	360	220	1225	335	286	0	1035	0	3066	770	369	0	0	0	0	0	143	2455	10264
u14	916	357	0	1709	82	0	493	0	4144	40	0	103	0	0	36	0	0	0	7880
u15	300	181	126	2446	98	0	0	482	1671	2381	2240	0	0	0	0	0	0	1326	11251
u16	0	214	11	984	696	82	417	0	3027	207	0	71	0	0	0	0	0	0	5709
u17	0	0	0	107	647	0	0	63	2494	1069	517	137	0	0	0	0	840	2418	8292
u18	33	105	0	2668	0	313	289	139	2158	0	908	0	0	0	0	0	286	0	6899
u19	885	218	0	307	121	0	530	44	4656	1232	0	21	0	0	0	0	0	0	8014
u20	0	270	623	4410	0	0	636	330	4706	0	1769	0	0	0	189	95	0	0	13028
u21	1304	240	0	5235	0	0	1080	27	2358	65	483	0	0	0	0	0	0	1654	12446
u22	0	0	0	0	0	0	0	21	2616	25	0	0	0	0	0	0	0	226	2888
u23	150	361	509	3042	0	0	1195	32	2752	1567	1921	323	0	0	0	0	0	0	11852
u24	878	37	0	984	0	0	234	0	0	929	0	0	0	0	0	0	0	0	3062
u25	1363	362	0	1137	0	0	517	0	1084	982	697	202	0	0	0	0	22	62	6428
u26	669	0	0	3771	0	0	0	243	5757	12	0	0	0	0	0	0	0	0	10452
u27	25	172	343	443	0	0	1041	0	1903	0	1519	0	26	26	0	0	0	0	5498
u28	187	50	100	200	0	0	978	57	3534	1451	729	0	0	0	0	0	152	412	7850
u29	52	689	0	0	269	0	1013	271	4577	1064	0	95	0	132	0	0	52	227	8441
u30	918	123	0	2738	0	0	1278	0	2438	1396	0	33	0	0	0	0	0	0	8924
u31	25	671	40	631	254	0	633	0	449	421	0	0	0	558	0	0	0	77	3759
u32	0	242	0	505	0	89	1346	81	2809	890	2322	0	0	0	0	8	0	0	8292
u33	0	72	1387	2049	0	0	0	44	1914	770	1	15	0	0	0	0	1	15	6268
u34	0	0	0	1	0	0	0	1	1724	1	0	0	0	0	0	0	0	0	1727
u35	9	0	314	802	35	0	0	404	772	940	1076	29	0	0	0	0	100	777	5258
u36	267	179	470	5581	0	0	1682	198	28	285	2228	0	0	0	0	0	0	0	10918
u37	300	0	0	553	0	0	0	0	1784	753	0	0	0	0	0	0	0	1648	5038
u38	706	64	33	2179	54	0	0	27	3040	105	0	59	0	0	162	0	0	1730	8159
u39	86	231	82	2318	0	0	1704	13	3348	135	1236	3	0	104	113	0	0	156	9529
u40	394	106	0	2076	0	0	85	21	1438	32	591	3	0	0	0	0	82	109	4937
u41	366	40	0	1309	61	0	0	31	1434	210	0	0	0	0	0	0	0	22	3473
u42	0	0	0	0	0	0	1530	0	6422	927	0	184	75	0	372	0	0	0	9510
u43	0	864	117	1137	174	0	335	7	610	52	0	0	0	0	0	0	0	662	3958
u44	0	0	0	45	0	0	542	0	307	579	0	0	0	0	0	102	0	57	1632
u45	2029	557	414	3159	409	0	106	24	5106	1532	0	11	0	291	0	0	0	1506	15144
u46	68	35	0	200	0	0	512	206	1223	892	0	0	0	0	0	0	364	127	3627
u47	0	7	0	0	0	0	1377	0	1782	0	0	0	14	0	0	0	0	959	4139
u48	699	0	0	1892	482	0	0	22	508	317	564	0	0	45	0	0	0	919	5448
u49	147	366	60	2023	0	0	791	0	2807	0	1693	39	137	0	0	7	0	0	8070
u50	0	550	0	3888	0	0	2313	22	3832	4137	0	34	0	0	0	0	0	705	15481
u51	0	0	0	0	0	0	424	78	475	20	0	0	0	2	0	0	0	0	999
u52	215	452	538	4300	0	0	1492	181	0	932	1020	13	0	0	0	0	0	0	9143
u53	41	640	643	2676	0	0	601	93	3009	0	1298	104	0	0	0	0	678	0	9783
u54	0	0	0	0	0	0	213	31	442	0	719	109	0	0	0	0	0	0	1514
u55	0	968	0	700	0	0	1136	135	2935	1805	500	62	0	0	0	0	219	250	8710
u56	685	71	447	4585	0	0	209	0	4504	0	1336	16	0	0	0	0	180	713	12746
u57	784	781	269	4613	341	0	799	43	6798	1190	173	0	0	15	139	0	167	1040	17152
u58	42	37	0	540	0	0	342	24	112	637	0	14	0	0	0	0	0	127	1875
u59	0	0	41	0	0	0	167	175	765	0	761	0	0	0	0	0	2	0	1911
	23401	14573	10201	115037	6110	3848	38081	5153	152892	42331	33944	2098	551	1470	1151	585	6224	24737	

(b) Quantitative visualization of number of context labels per subject - Part 2
Including label subsets “Phone Location”, “Work-related”, “Location-based”, “Companion”

	Shopping	Cooking	Cleaning	Doing laundry	Washing dishes	Bathing - Shower	Toilet	Grooming	Dressing	Sleeping	Exercise	Eating	Drinking alcohol	Watching TV	Surfing the internet	Talking	Singing	
u00 -	0	0	0	0	0	32	23	130	5	3420	2	446	92	928	0	2186	0	7264
u01 -	101	349	8	5	29	0	197	0	9	615	4	259	0	544	849	180	0	3149
u02 -	0	116	61	0	0	0	17	0	0	1913	244	441	0	101	47	742	0	3682
u03 -	0	94	32	0	16	24	3	5	0	0	92	59	0	0	103	97	0	525
u04 -	0	0	0	0	0	160	181	253	220	1035	47	461	0	0	1340	971	0	4668
u05 -	19	55	38	0	0	15	0	0	17	477	0	54	0	2	0	0	0	677
u06 -	0	0	0	0	0	0	0	0	0	2440	136	415	0	0	1674	2422	0	7087
u07 -	8	15	17	0	0	8	22	0	0	517	46	68	0	0	0	29	0	730
u08 -	0	22	15	56	16	97	93	76	42	2690	154	459	58	448	153	47	0	4426
u09 -	0	15	0	0	0	0	24	0	0	1317	150	169	0	0	464	118	0	2257
u10 -	0	60	0	0	11	13	4	37	16	1860	423	101	0	0	0	37	0	2562
u11 -	7	3	0	0	0	8	0	17	17	3714	109	322	0	0	0	0	0	4197
u12 -	0	0	0	0	0	0	0	22	28	3433	0	150	0	0	559	491	0	4683
u13 -	44	0	330	96	5	87	56	187	0	2422	13	457	0	49	69	1414	0	5229
u14 -	22	68	210	5	82	60	196	38	14	510	36	138	0	639	78	502	2	2600
u15 -	0	0	0	0	0	0	5	0	0	289	14	587	0	99	2120	1434	292	4840
u16 -	0	30	0	0	0	0	1	0	6	876	0	184	0	0	168	766	148	2179
u17 -	48	331	828	0	0	88	96	116	118	1312	0	376	0	62	231	2438	0	6044
u18 -	0	182	0	8	67	0	31	212	0	1107	0	383	0	570	0	874	0	3434
u19 -	0	30	0	0	0	0	0	0	0	1482	308	77	0	1108	0	266	0	3271
u20 -	36	23	153	82	0	11	0	0	0	2525	251	48	0	30	92	348	0	3599
u21 -	124	444	0	0	2	0	67	0	67	0	8	218	0	275	0	1826	0	3031
u22 -	102	59	68	28	32	22	5	139	0	1260	0	178	0	462	0	179	0	2534
u23 -	264	65	30	0	0	16	129	60	63	2409	31	245	0	0	0	297	0	3609
u24 -	0	0	0	0	0	0	0	0	0	0	0	127	0	15	0	21	0	163
u25 -	44	58	0	0	0	15	27	14	36	1	59	324	105	0	0	958	31	1672
u26 -	69	131	320	0	13	0	79	30	77	2485	787	344	0	270	1860	0	0	6465
u27 -	33	0	0	0	0	136	85	136	136	1876	0	622	0	35	340	837	0	4236
u28 -	106	191	137	10	25	78	121	217	13	2292	362	213	0	0	155	307	0	4227
u29 -	0	0	52	0	0	60	48	0	48	2286	235	256	0	115	0	403	0	3503
u30 -	0	3	117	0	0	20	0	2	4	2305	27	141	74	0	0	48	0	2741
u31 -	0	20	43	0	0	0	23	16	16	892	29	41	188	83	467	255	0	2073
u32 -	0	25	0	0	0	0	26	16	10	1742	350	25	0	1	136	221	0	2552
u33 -	263	0	0	0	0	18	0	30	30	352	242	156	0	0	1434	666	0	3191
u34 -	0	0	0	0	0	0	0	0	0	0	285	0	0	44	0	0	0	329
u35 -	15	79	0	0	79	0	14	0	0	25	0	125	0	0	0	491	35	863
u36 -	0	69	0	0	77	0	0	10	0	2310	2	471	0	1433	0	86	0	4458
u37 -	0	0	0	0	0	107	41	0	47	1145	7	136	0	0	0	537	0	2020
u38 -	111	5	0	0	0	85	12	0	67	1021	162	762	66	912	0	638	136	3977
u39 -	68	356	184	90	272	91	208	0	77	1031	199	280	0	131	1736	875	0	5598
u40 -	5	47	0	3	46	11	35	36	57	633	1	310	28	257	242	653	6	2370
u41 -	11	0	94	0	0	0	83	23	0	1082	0	102	0	189	43	152	0	1779
u42 -	0	0	0	0	0	0	0	0	0	2399	662	0	0	0	0	0	0	3061
u43 -	0	0	15	0	0	32	51	0	0	1115	0	108	0	45	845	673	0	2884
u44 -	0	0	0	0	0	56	0	0	0	63	378	115	0	0	0	31	0	643
u45 -	15	173	155	0	10	55	91	226	254	2552	676	503	351	710	512	2711	0	8994
u46 -	0	65	22	24	80	0	4	4	0	0	141	108	0	59	0	96	0	603
u47 -	226	77	11	0	0	20	0	11	0	2269	579	746	301	607	1293	687	0	6827
u48 -	0	0	0	0	0	142	13	78	78	1093	232	250	0	0	0	631	0	2517
u49 -	0	32	0	0	4	19	0	0	11	1607	97	101	0	27	0	72	0	1970
u50 -	0	0	0	0	0	160	89	0	10	3470	0	976	0	0	17	870	0	5592
u51 -	28	33	32	5	28	31	33	10	0	175	59	151	0	131	48	3	0	767
u52 -	0	15	173	0	4	36	0	70	45	1432	1	373	2	199	303	540	0	3193
u53 -	37	57	279	81	51	6	26	134	57	1458	33	331	40	97	76	553	0	3316
u54 -	0	0	0	35	0	0	0	0	0	0	0	172	0	420	0	27	0	654
u55 -	17	256	41	0	69	0	1	76	0	1097	0	586	0	176	297	1286	0	3902
u56 -	0	191	96	0	96	59	93	249	191	2074	0	594	151	639	443	2107	0	6983
u57 -	18	175	234	28	113	178	290	343	273	2702	140	626	0	1327	1128	1103	0	8678
u58 -	0	0	0	0	0	0	2	1	0	0	0	0	0	52	83	0	1	139
u59 -	0	10	11	0	1	31	10	40	74	448	268	124	0	20	11	91	0	1139
	1841	4029	3806	556	1228	2087	2655	3064	2233	83055	8081	16594	1456	13311	19416	36293	651	

(c) Quantitative visualization of number of context labels per subject - Part 3
Including label subsets “Chores”, “Self-care”, “Leisure Time”

We must remind that every example, which corresponds to 1 minute of daily-living (20 seconds of each minute are recorded by the sensors, to constitute one example), can be annotated with more than one label by the users, which means that the labels are not mutually exclusive and that the dataset is based on the co-occurence of multiple labels for each example. Taking out of the equation the fact that each user participated in the data collection process for a span of days varying from 2.9 to 28.1, the differences in the type and amount of labels annotated by each user reflect the individual, personalized habits and routines of each user, and also their label annotation habits (how often they used the app, which labels they ignored etc.).

5.2.2 Kernel Density Estimation for Feature Visualization

In order to delve deeper in the heterogeneity of the data, we adopt a feature visualization technique proposed in [She+23a], using a Kernel Density Estimator (KDE) to calculate and visualize the conditional distribution of each feature extracted from the dataset. The calculated distributions correspond to sensor features conditioned on activity or user, as it will be explained more thoroughly in the next few paragraphs. All sensor features have been analyzed, except for the discrete features, which have binary values in the one-hot dataset features' format.

Definition 5.2.1: Kernel Density Estimator (KDE)

Assuming that $\{X_1, ...X_n\}$ is an independent and identically distributed (IID) sample from an unknown density function p, the KDE function $\hat{p}_n$ can be formally defined as:

$$\hat{p}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) \tag{5.2.1}$$

where K is the kernel, a non-negative function, and $h > 0$ is the bandwidth, a smoothing parameter.

In our work, the KDE functions were calculated and plotted in Python using `seaborn.kdeplot`, which calls `scipy.stats.gaussian_kde` to estimate $\hat{p}_n$ using Gaussian K, while the bandwidth h is calculated using Scott's Rule [Sco92].

In Figure 5.2.3, we present a subfigure per sensor feature. Each density function corresponds to a single activity from the "Posture/Movement" subset, which includes the labels "Lying down", "Sitting", "Standing", "Walking", "Running", "Bicycling", as stated in Table 5.1, and all the data are from a single user, u45, who is the user with the most examples in total, who has used all six labels of the specific label subset. We can see that the distribution of feature values across the different activities has discriminatory power for a lot of the sensor features that have been extracted, while other features do not seem to be indicative of the label, particularly for this specific subset of labels.

In Figure 5.2.4, we present a similar setup with the previous Figure, but in this one, the data from all the users of the dataset for the particular activites has been used to calculate the density functions. Comparing the two Figures, we observe a lot of similarities in the distributions of the same sensor features, but we can also notice that in the second Figure, since the data of all the users has been merged, some trends and patterns have been averaged out and the distributions are less informative about the differences in sensor features across different activites.

In Figure 5.2.5, we also have a subfigure per sensor feature, but now the setup is different. We have a single activity label, "Walking", and in each subfigure, each density function corresponds to a single user from a subset of users including u01, u04, u21, u29, u45, and u55, that were carefully selected to have enough examples of the specific activity in order to be representative. We hypothesize that if we selected users with less data for the specific activity, the differences among the users' distributions would be even bigger.

In Figure 5.2.5, it can be observed that for most sensor features, the distributions across users are considerably similar, but also tend to have differences which probably can be attributed to the particularity in the way each user performs activities, or differences in the placement of the sensors, especially the smartphone which might be held in different ways or placed in pockets with different orientation. These differences might complicate HAR tasks in a real-life setting, but they are inherent in such a setting, thus they must be taken into consideration to build HAR models with satisfactory performance for everyday life.

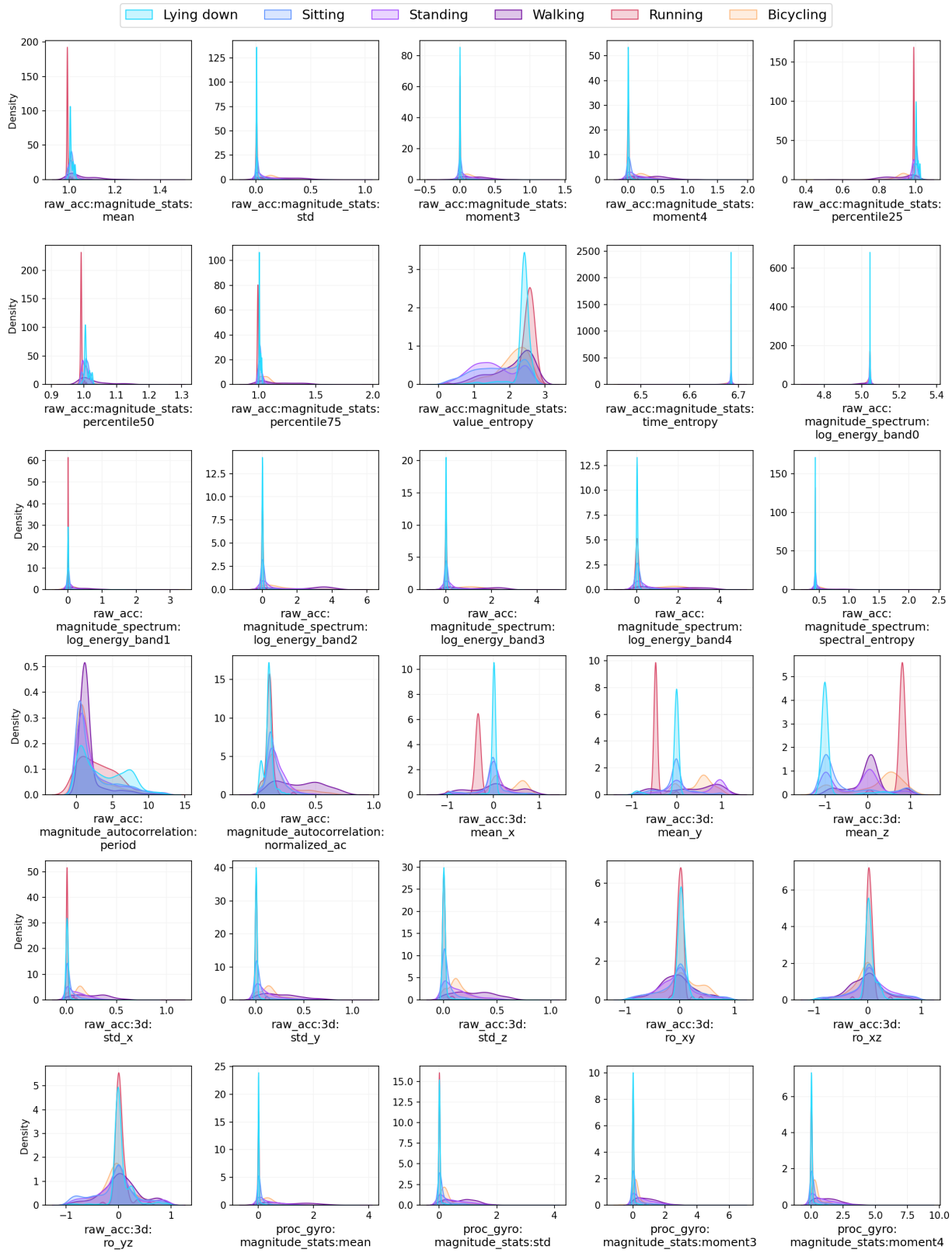
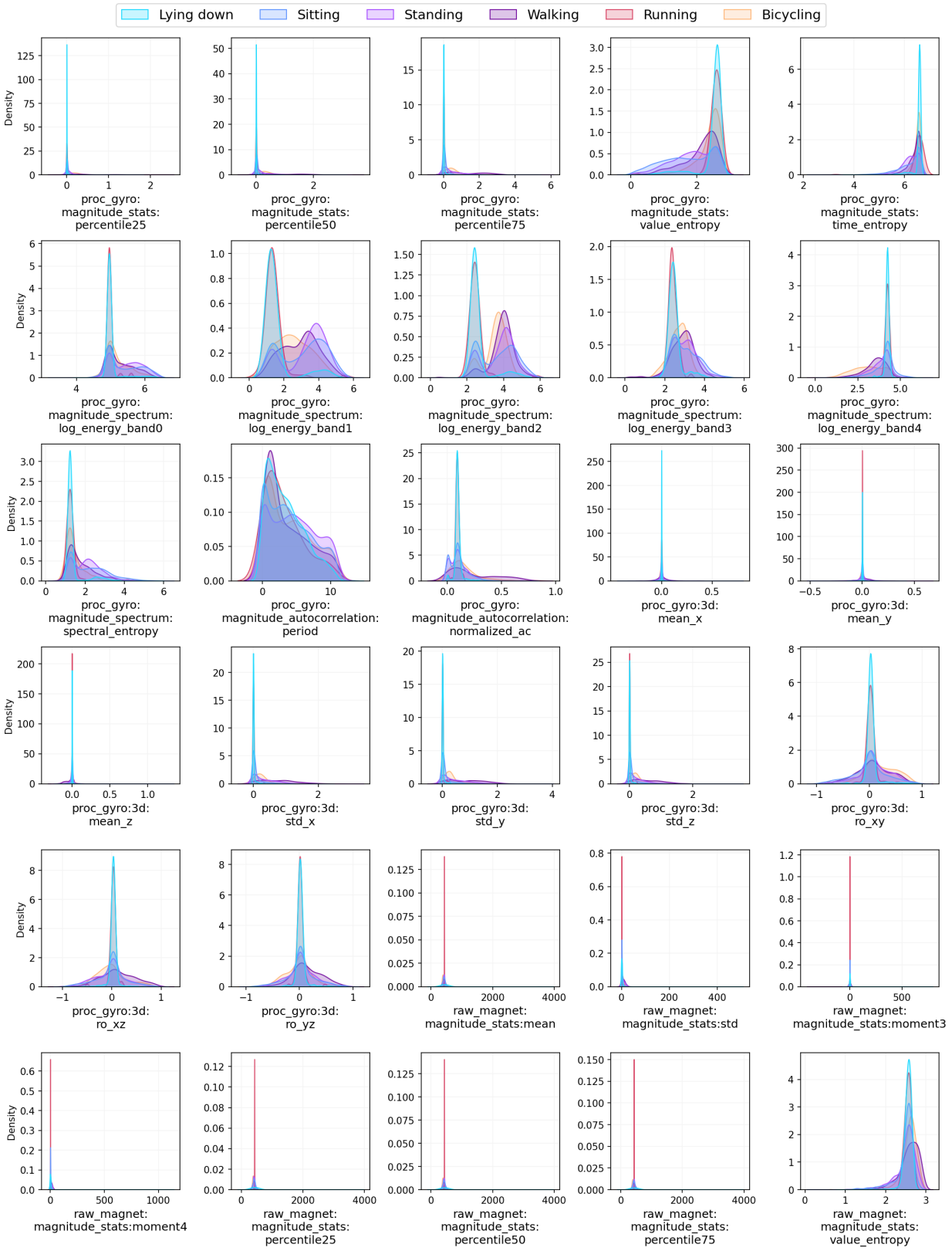
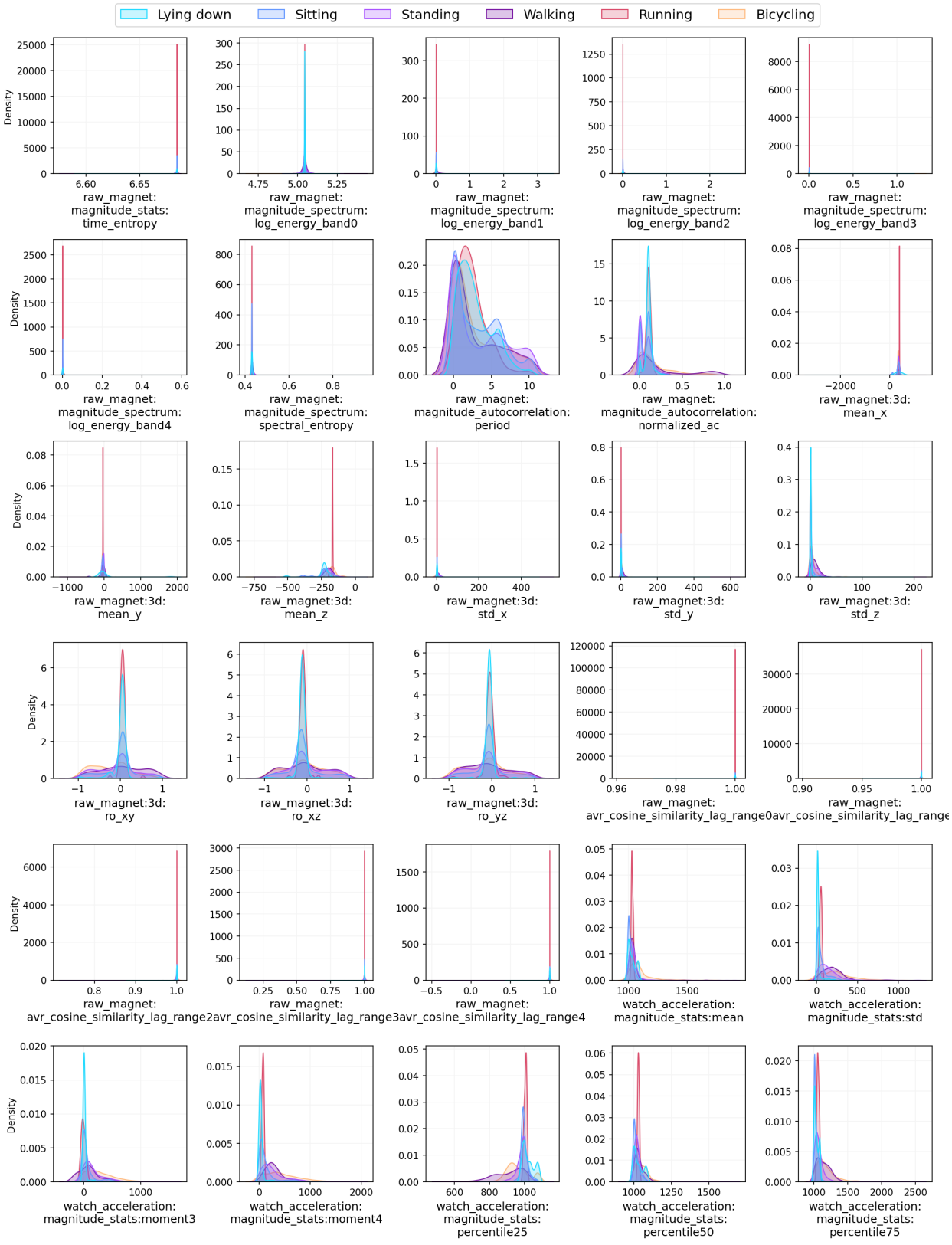


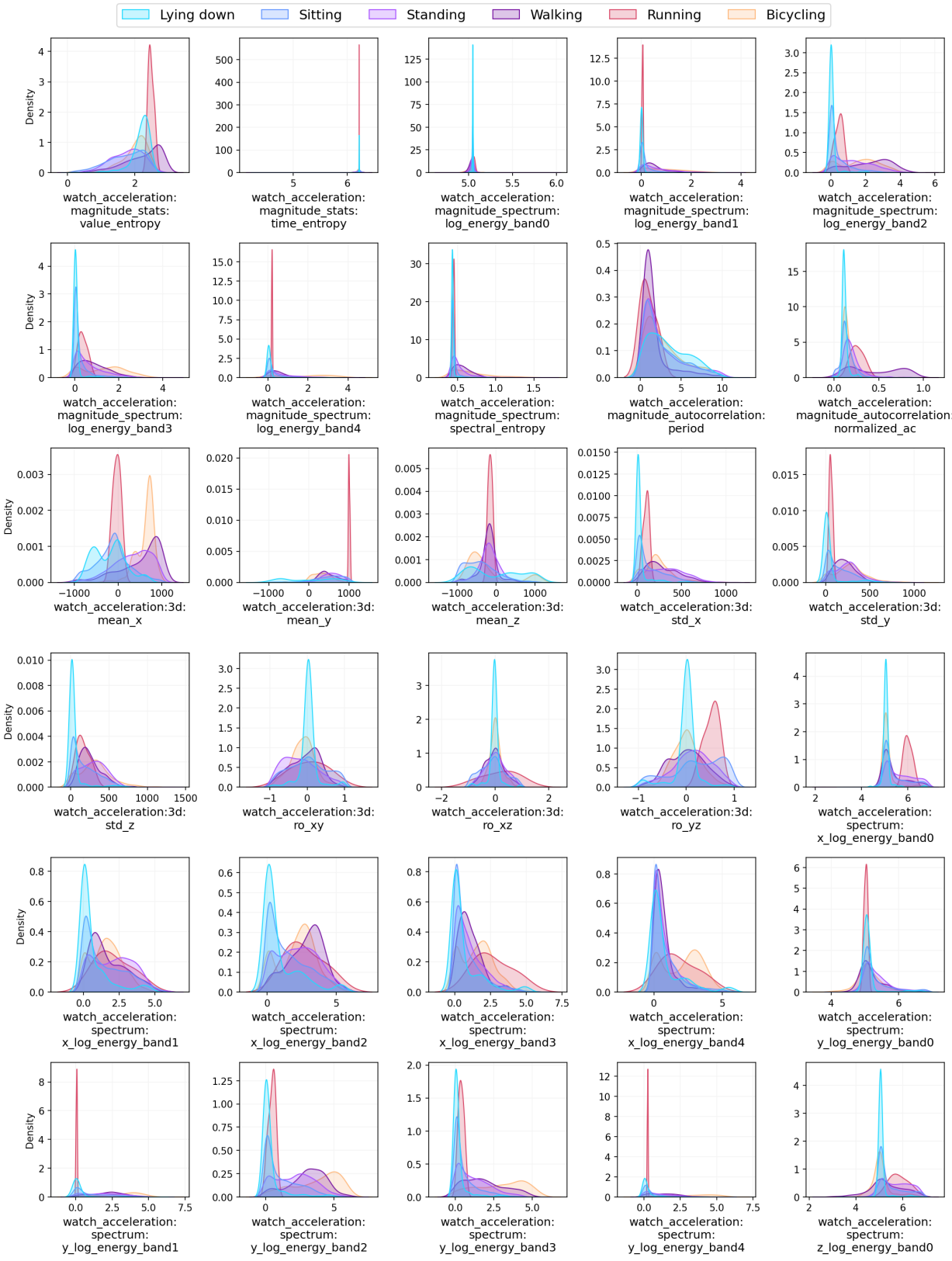
Figure 5.2.3: (a) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for user u45 - Part 1



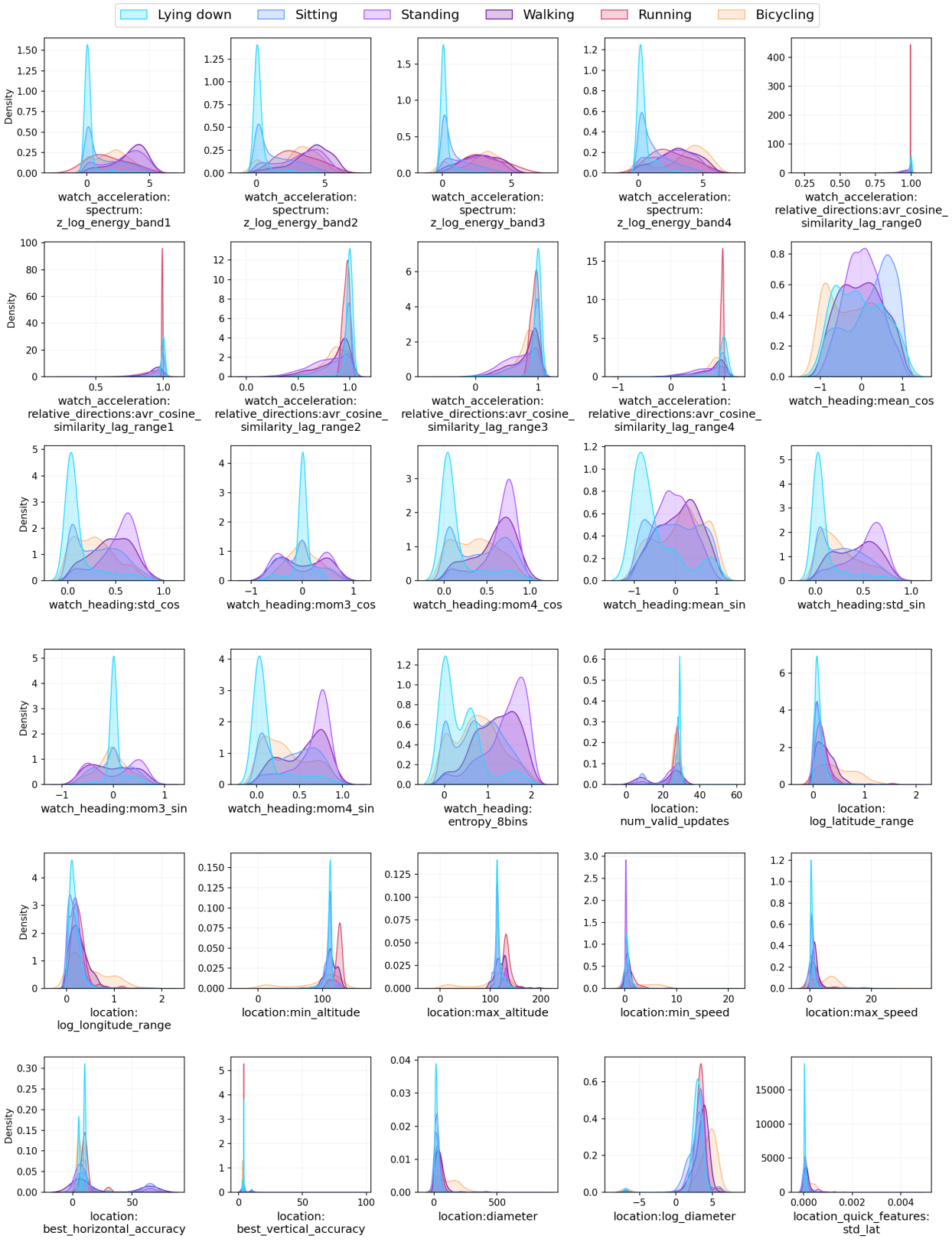
(b) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for user u45 - Part 2



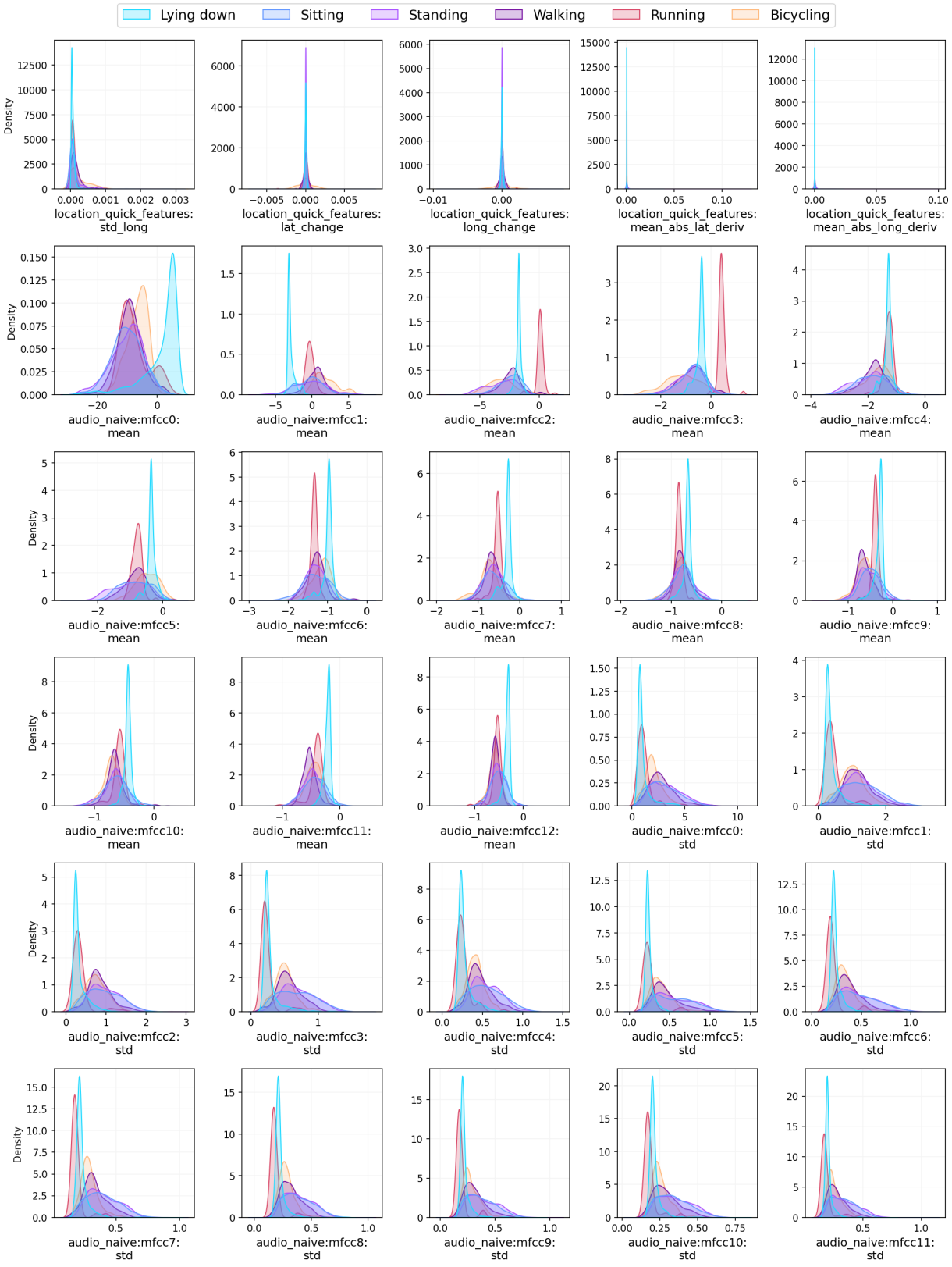
(c) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for user u45 - Part 3



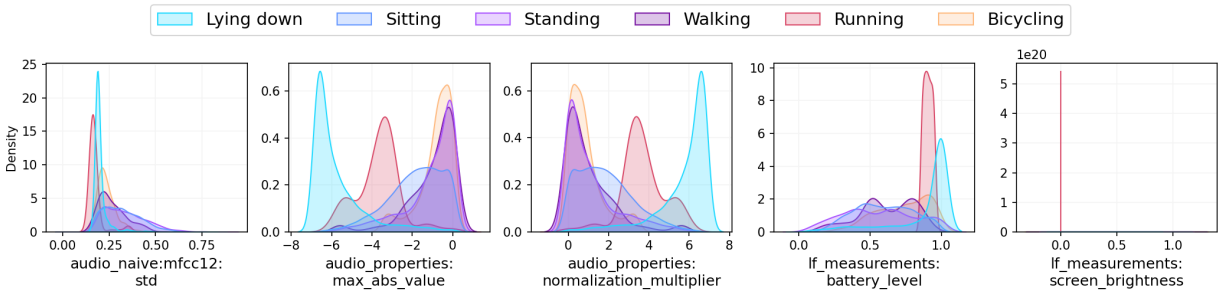
(d) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for user u45 - Part 4



(e) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for user u45 - Part 5



(f) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for user u45 - Part 6



(g) KDE plot visualizing the distribution of each feature
across the "Posture/Movement" activities subset, for user u45 - Part 7

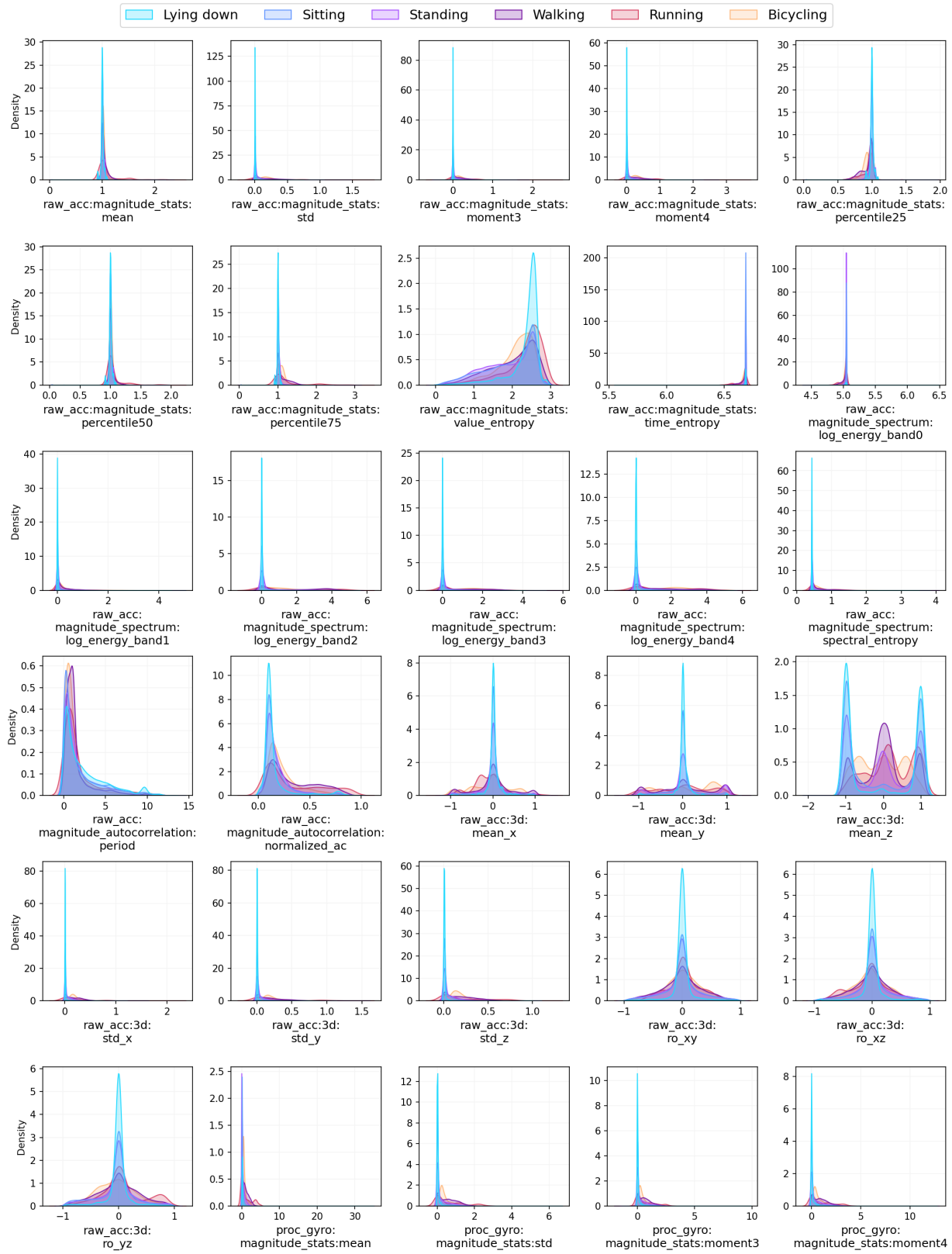
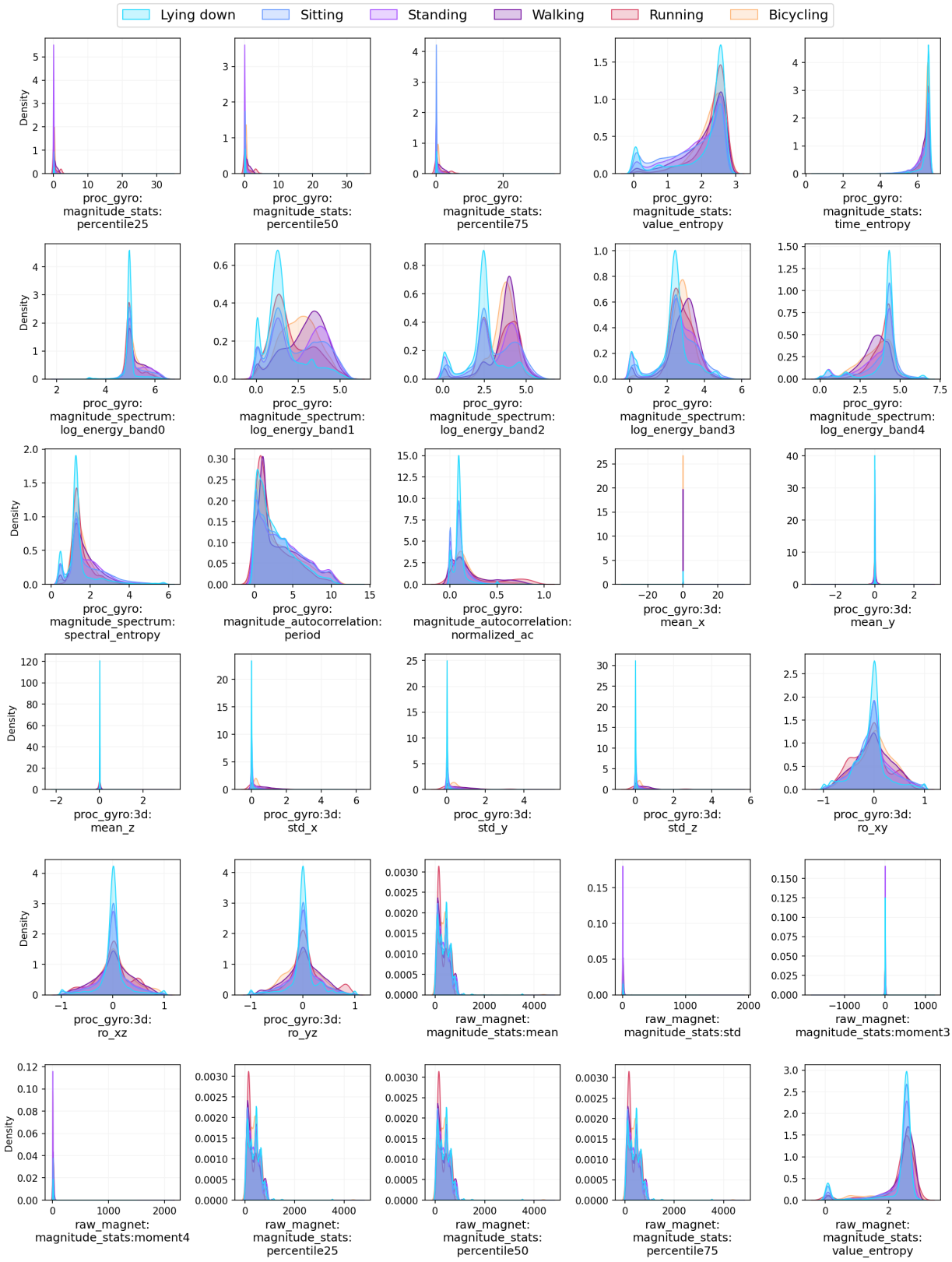
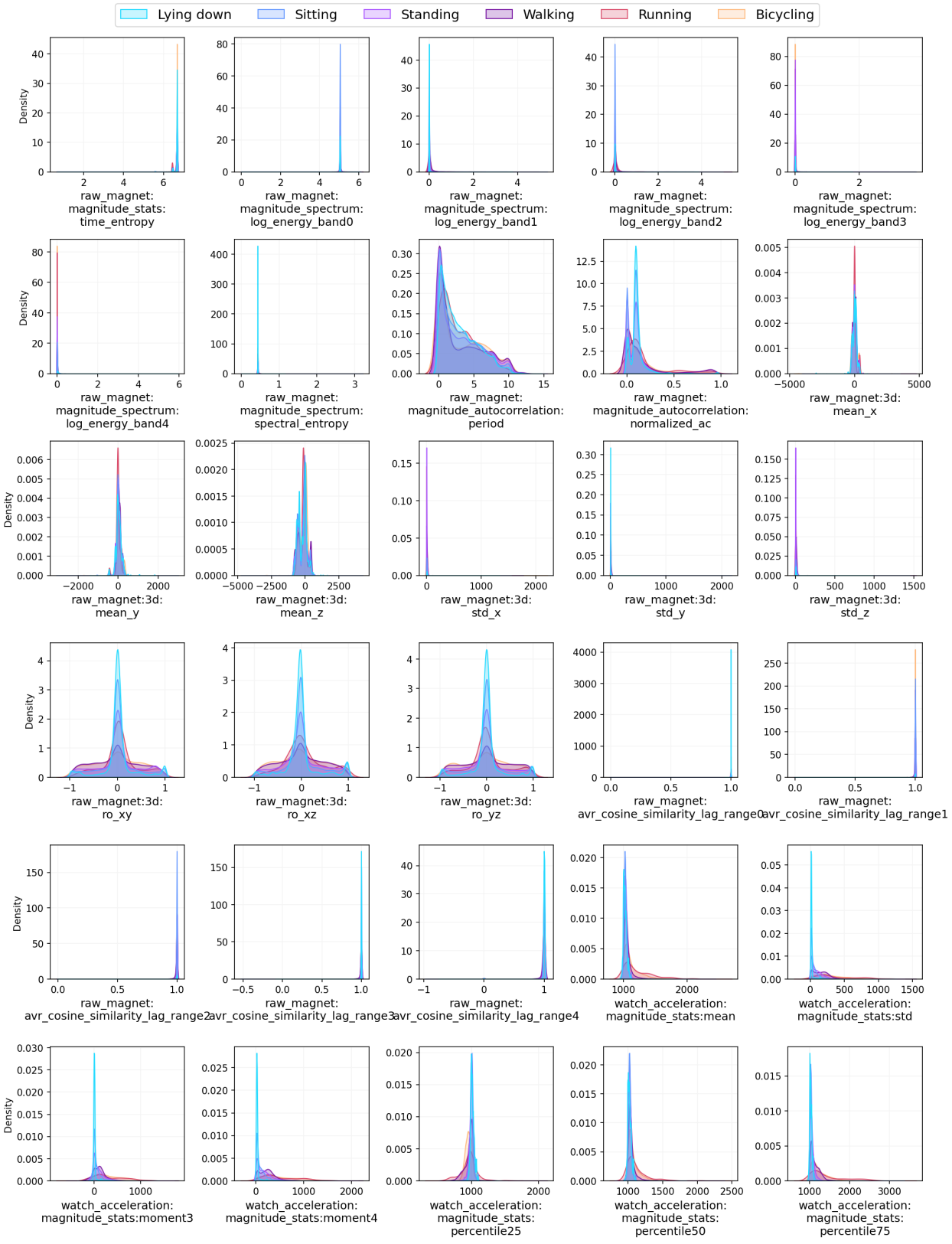


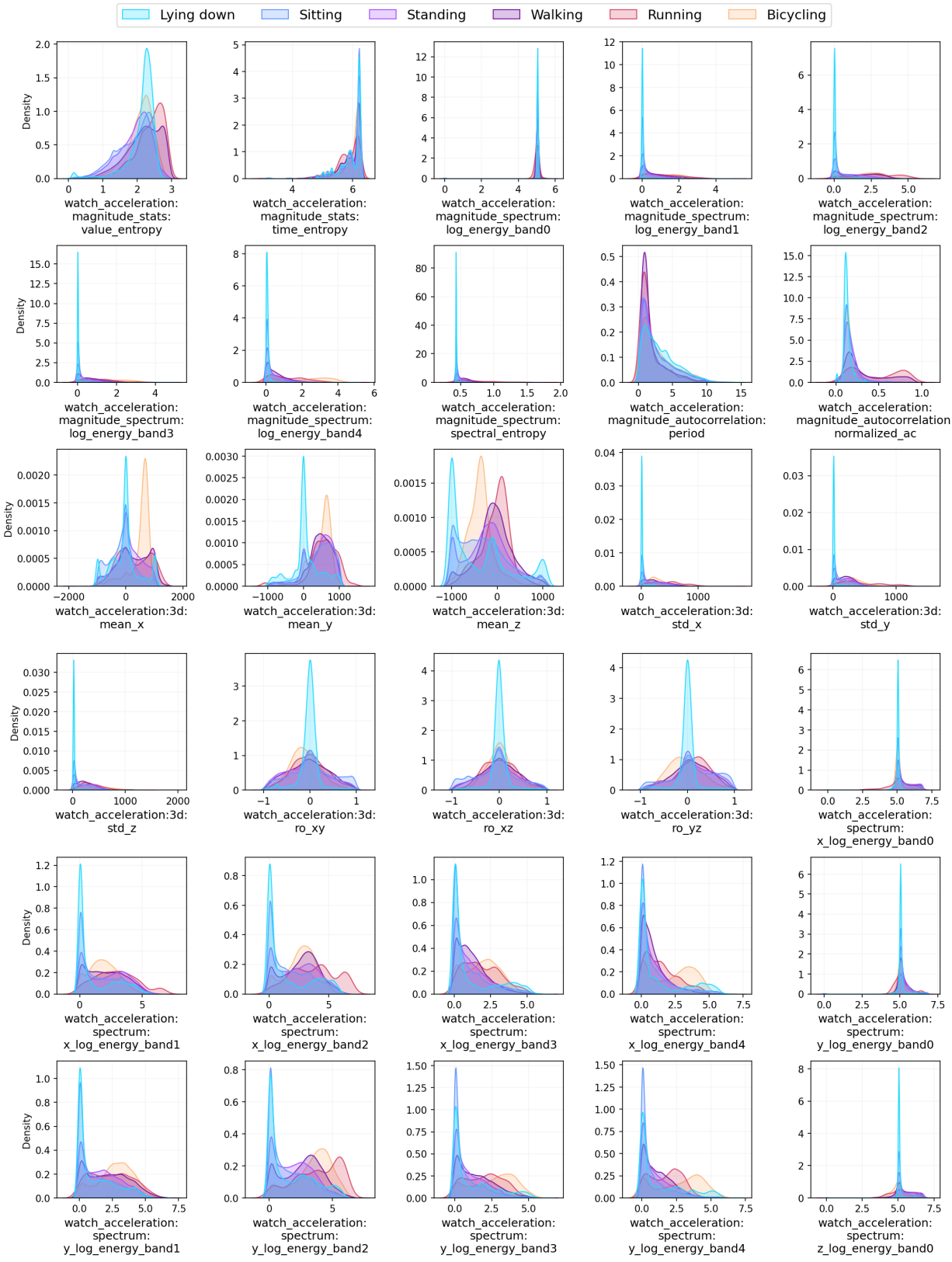
Figure 5.2.4: (a) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for all users - Part 1



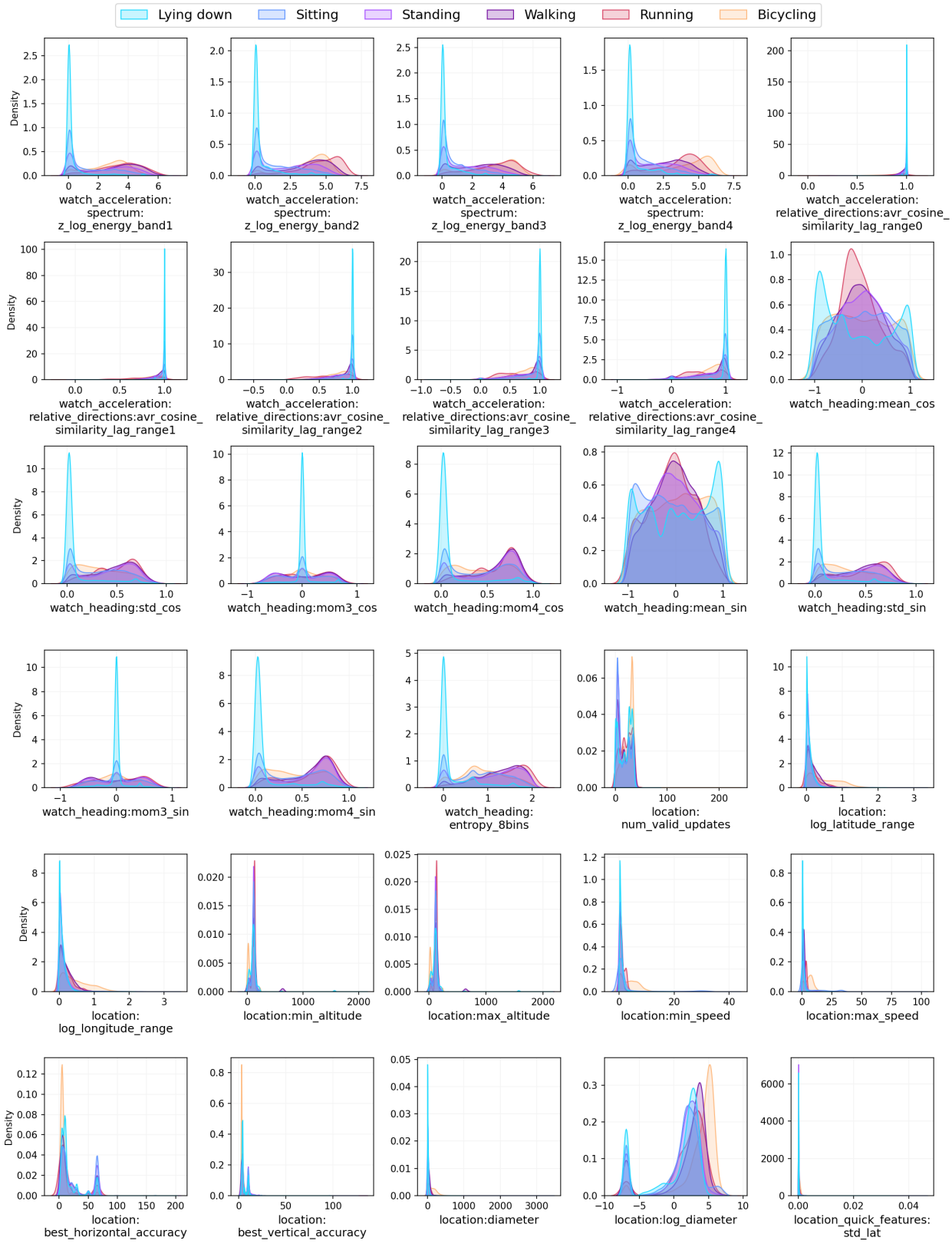
(b) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for all users - Part 2



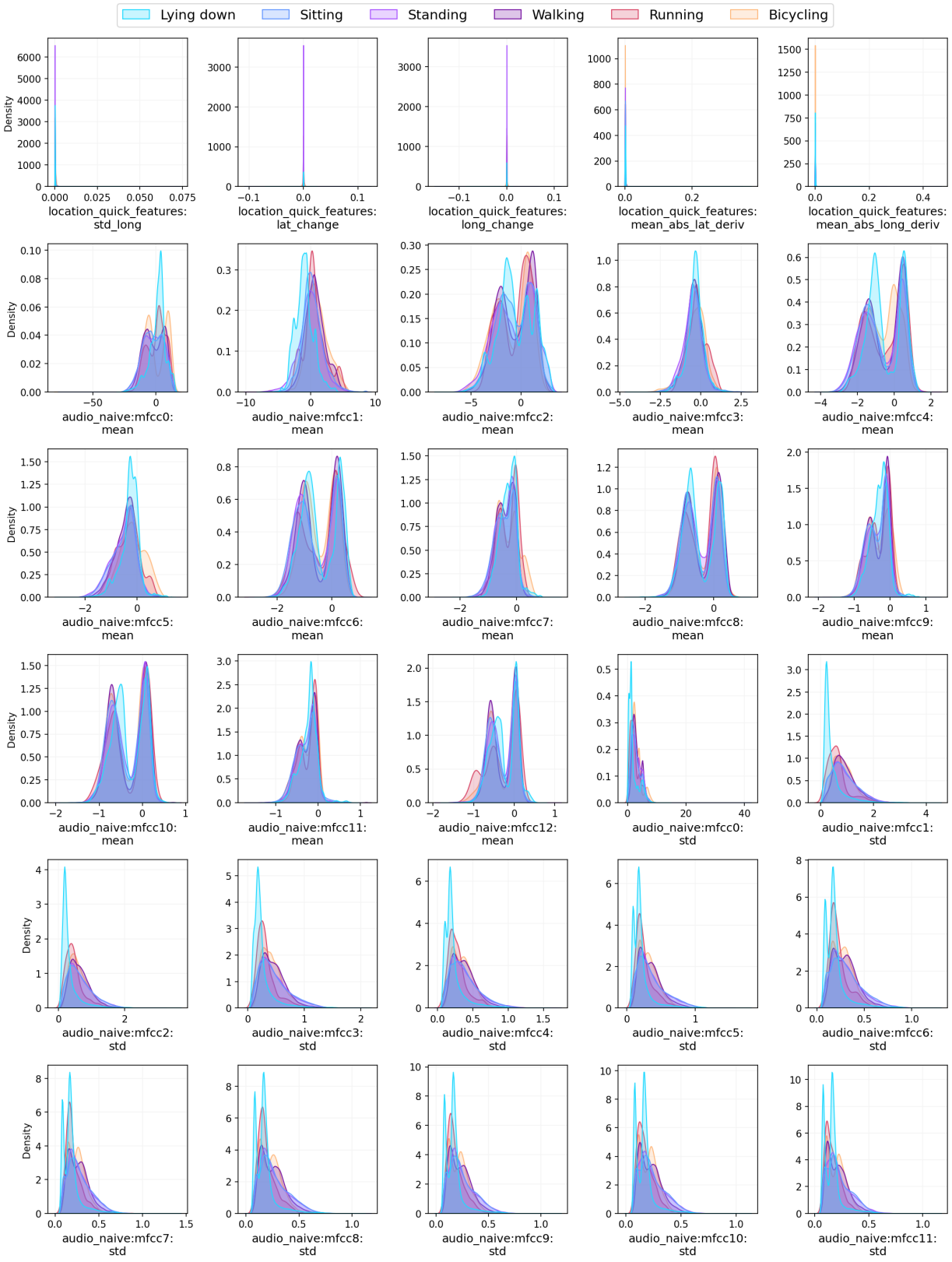
(c) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for all users - Part 3



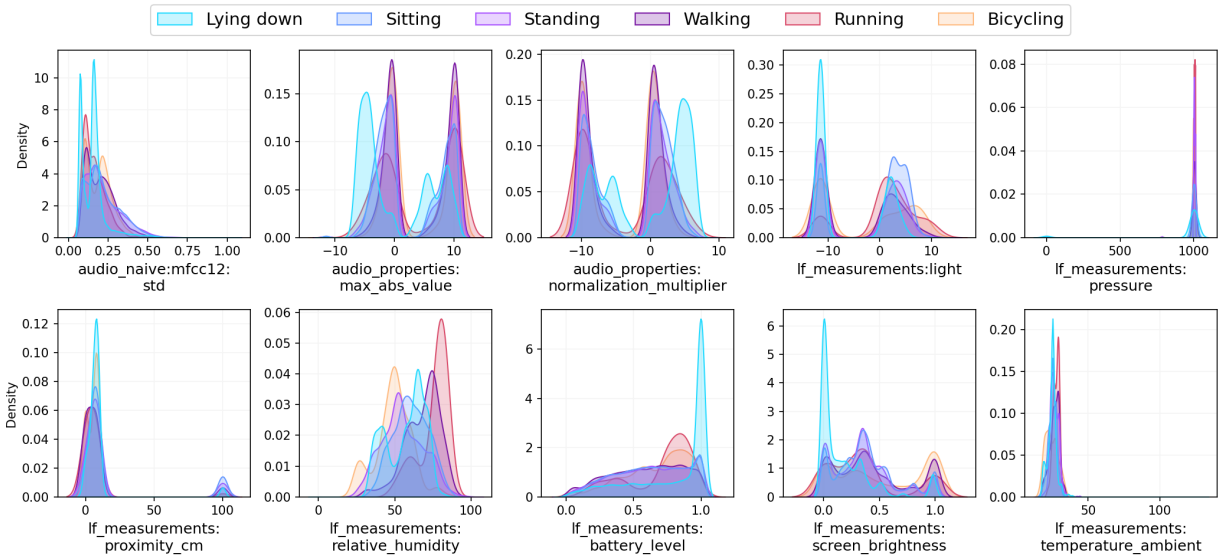
(d) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for all users - Part 4



(e) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for all users - Part 5



(f) KDE plot visualizing the distribution of each feature across the "Posture/Movement" activities subset, for all users - Part 6



(g) KDE plot visualizing the distribution of each feature
across the "Posture/Movement" activities subset, for all users - Part 7

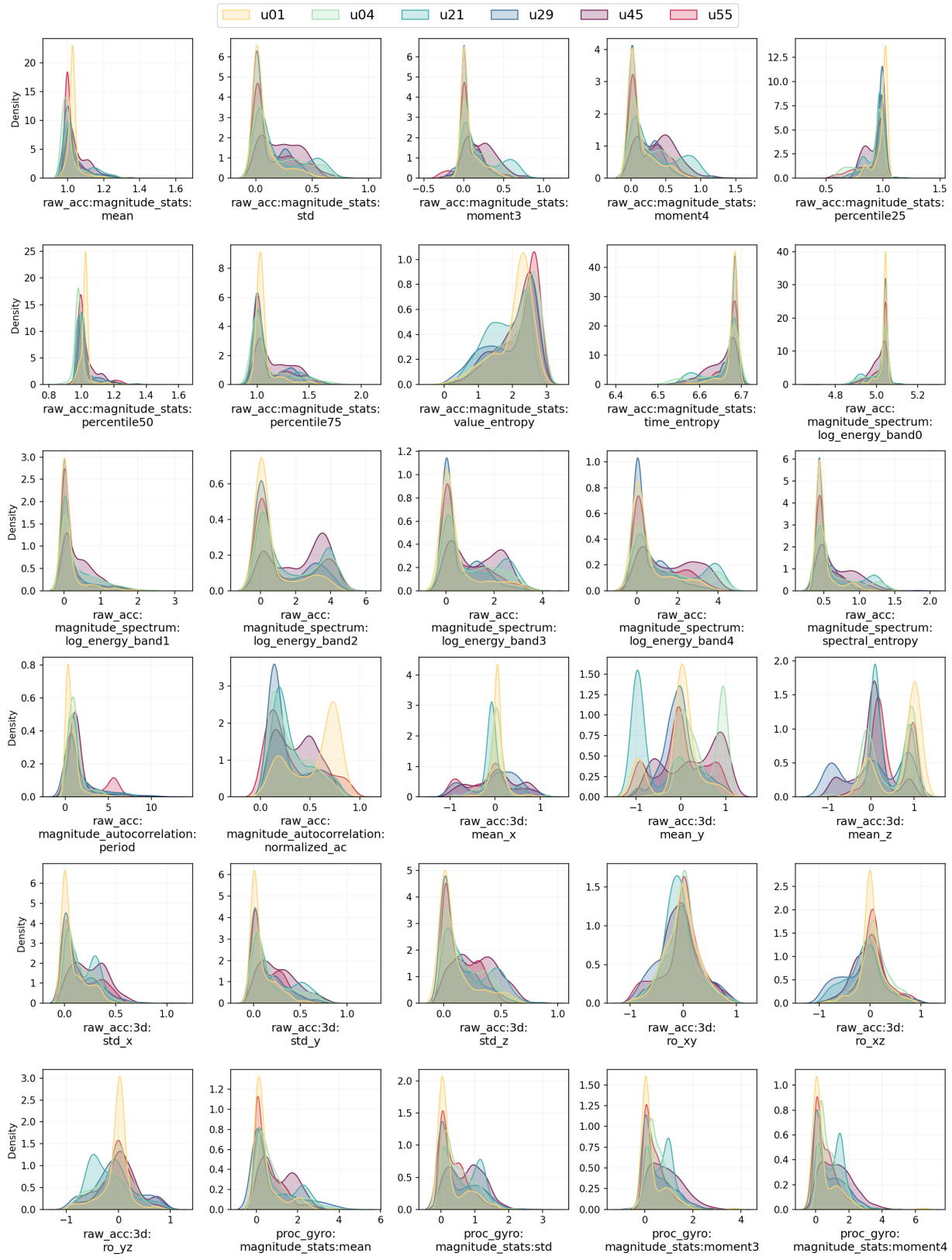
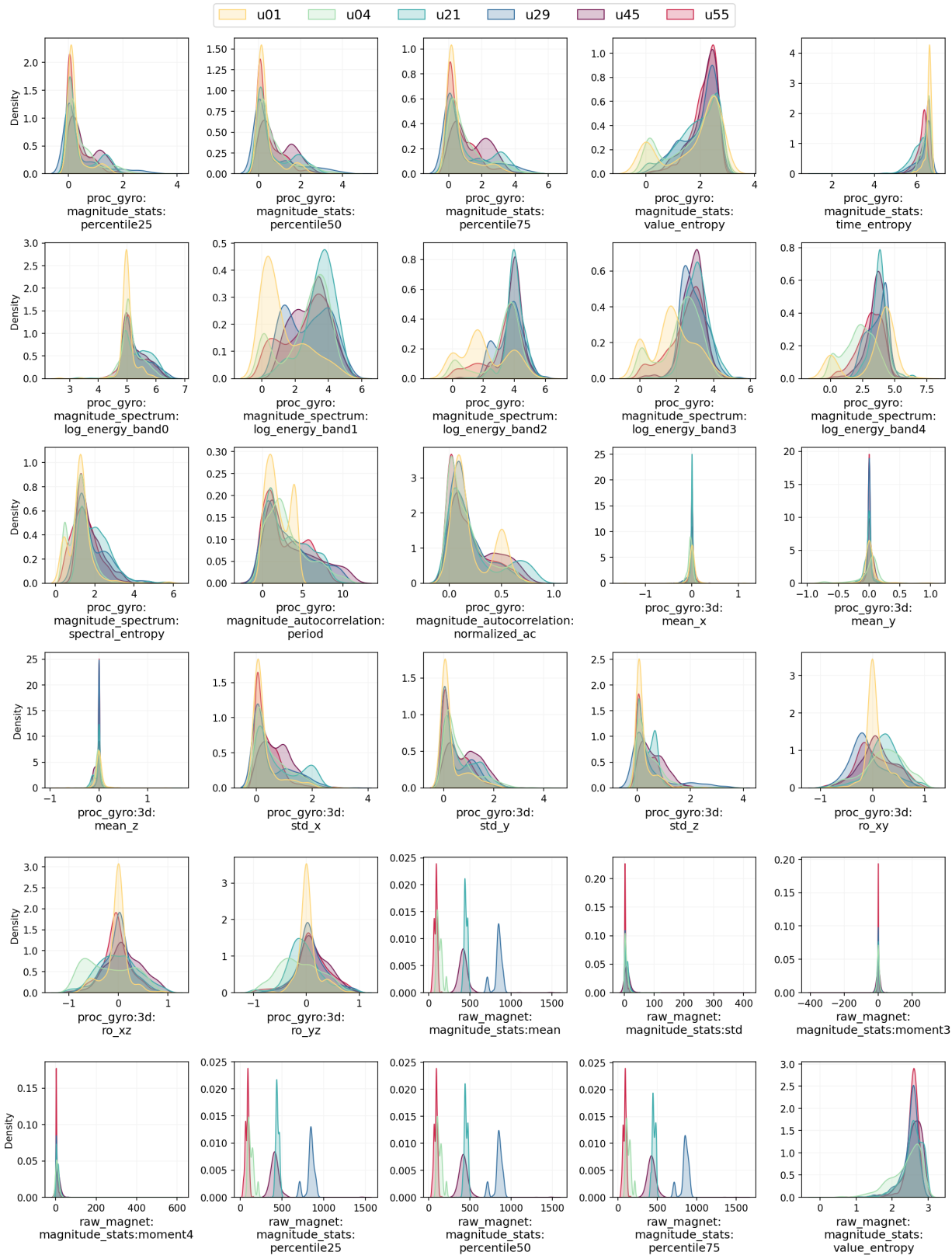
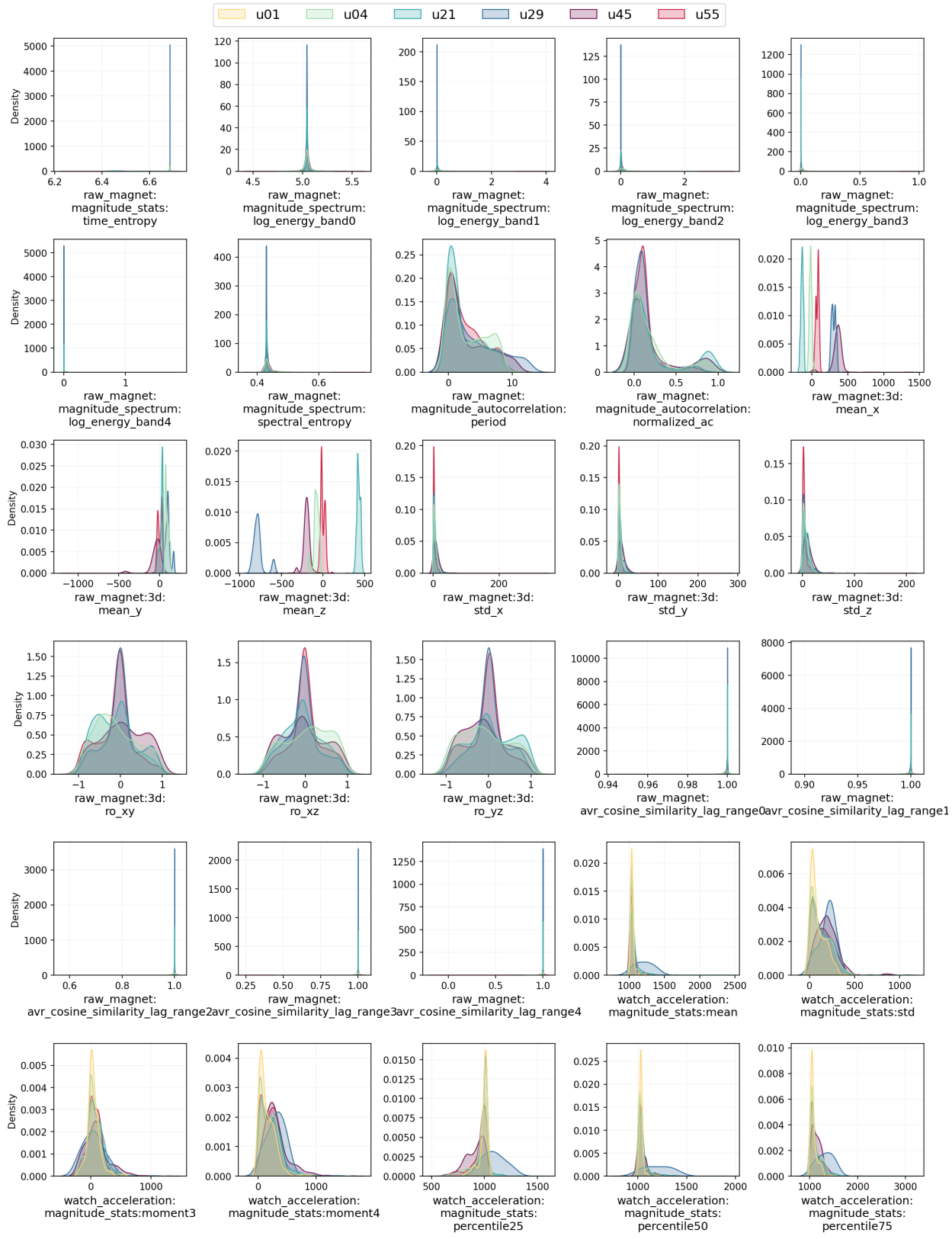


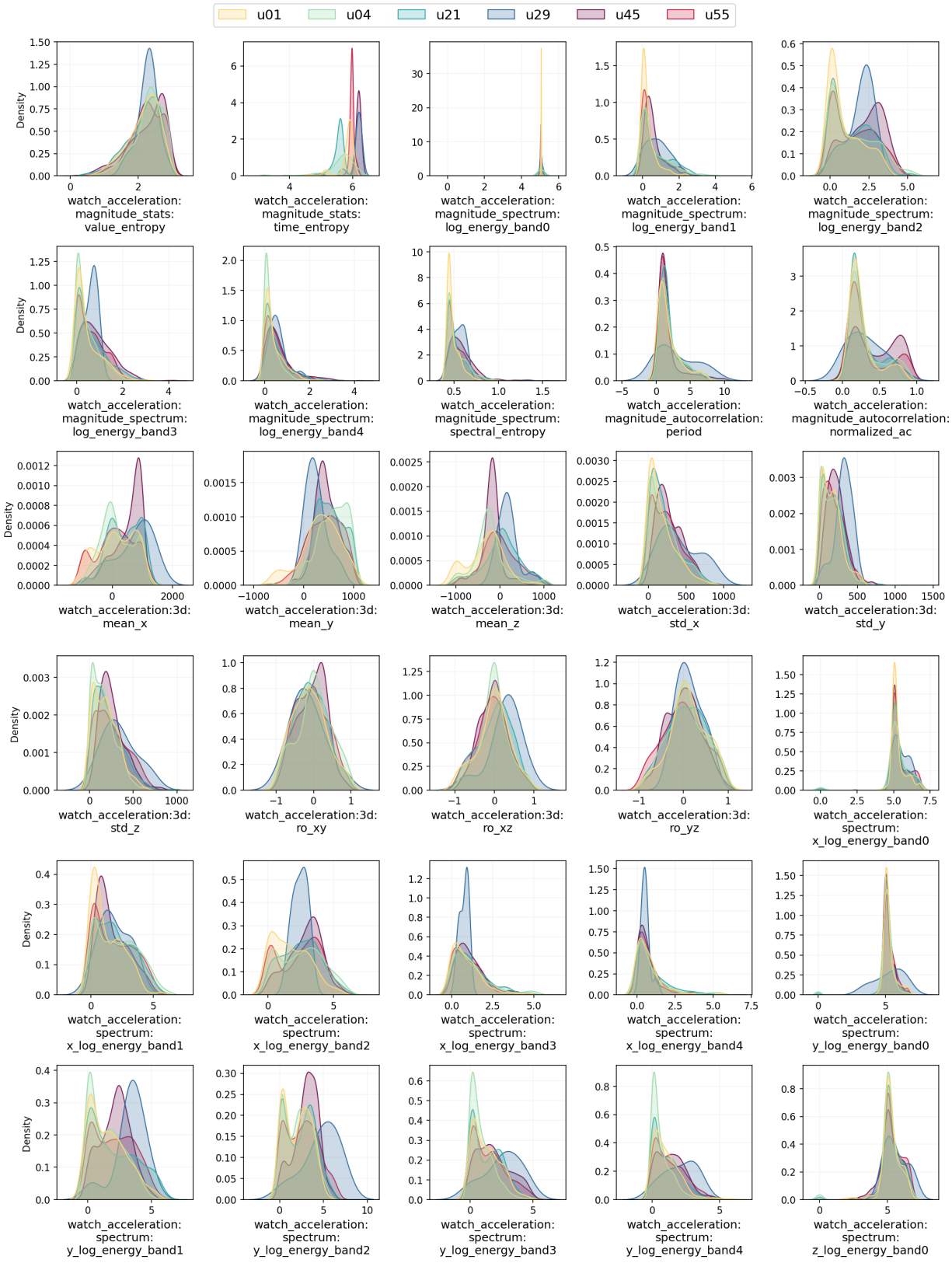
Figure 5.2.5: (a) KDE plot visualizing the distribution of each feature across five users: u01, u04, u21, u29, u45, u55, for the "Walking" activity - Part 1



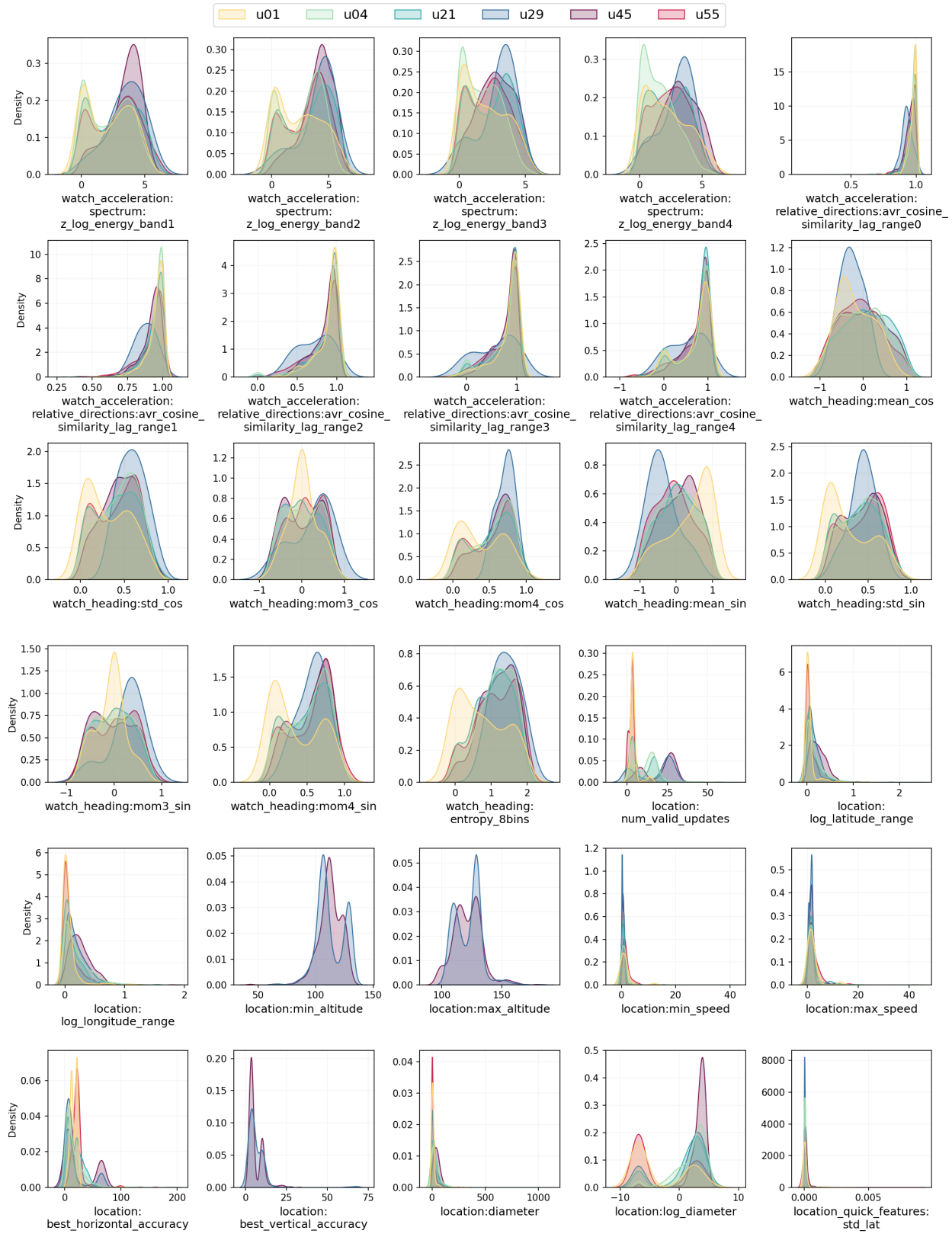
(b) KDE plot visualizing the distribution of each feature across five users: u01, u04, u21, u29, u45, u55, for the "Walking" activity - Part 2



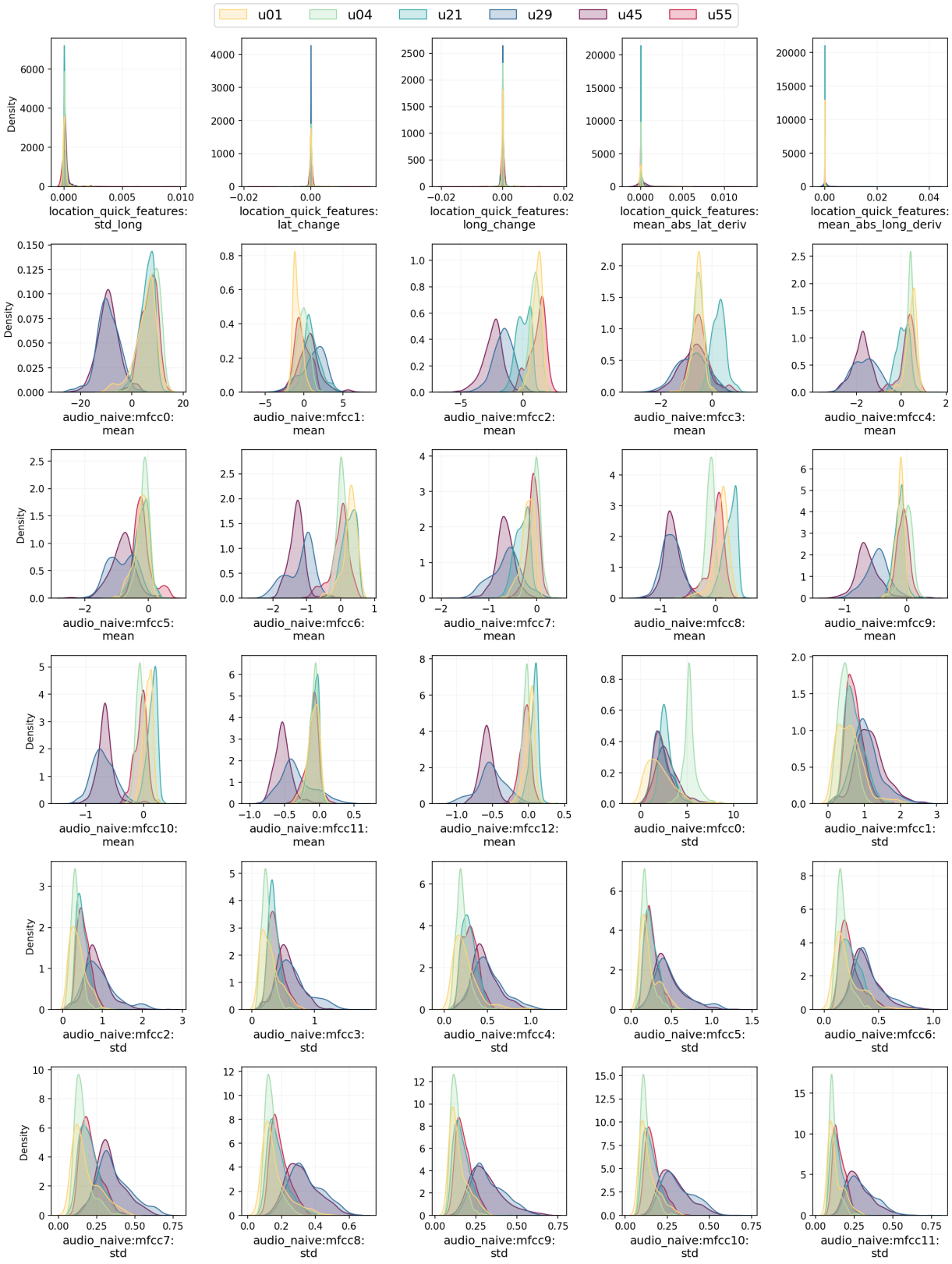
(c) KDE plot visualizing the distribution of each feature across five users: u01, u04, u21, u29, u45, u55, for the "Walking" activity - Part 3



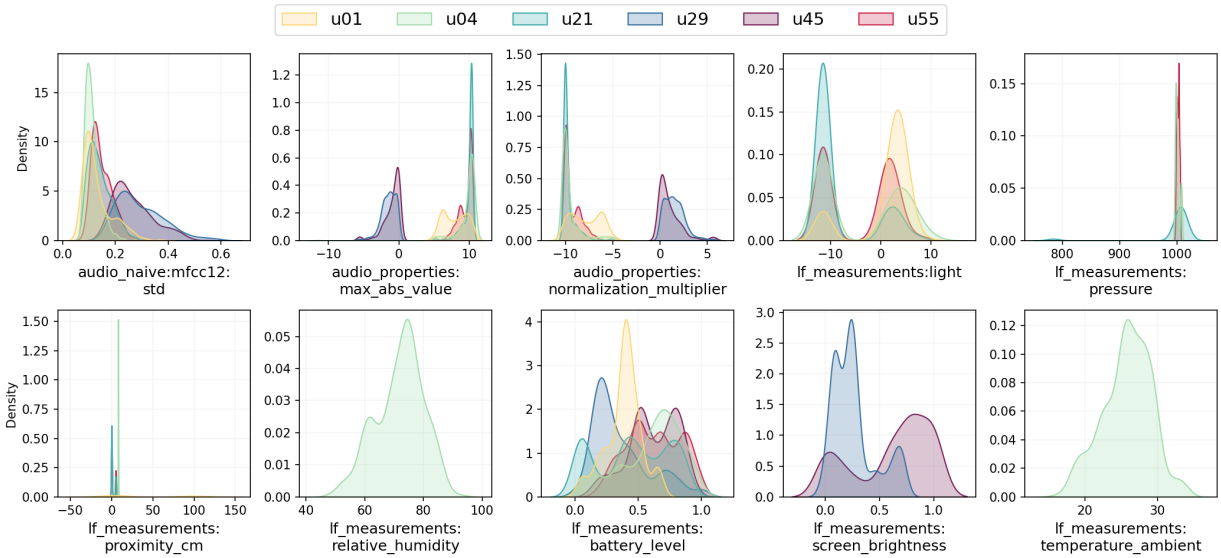
(d) KDE plot visualizing the distribution of each feature across five users: u01, u04, u21, u29, u45, u55, for the "Walking" activity - Part 4



(e) KDE plot visualizing the distribution of each feature
across five users: u01, u04, u21, u29, u45, u55, for the “Walking” activity - Part 5



(f) KDE plot visualizing the distribution of each feature across five users: u01, u04, u21, u29, u45, u55, for the "Walking" activity - Part 6

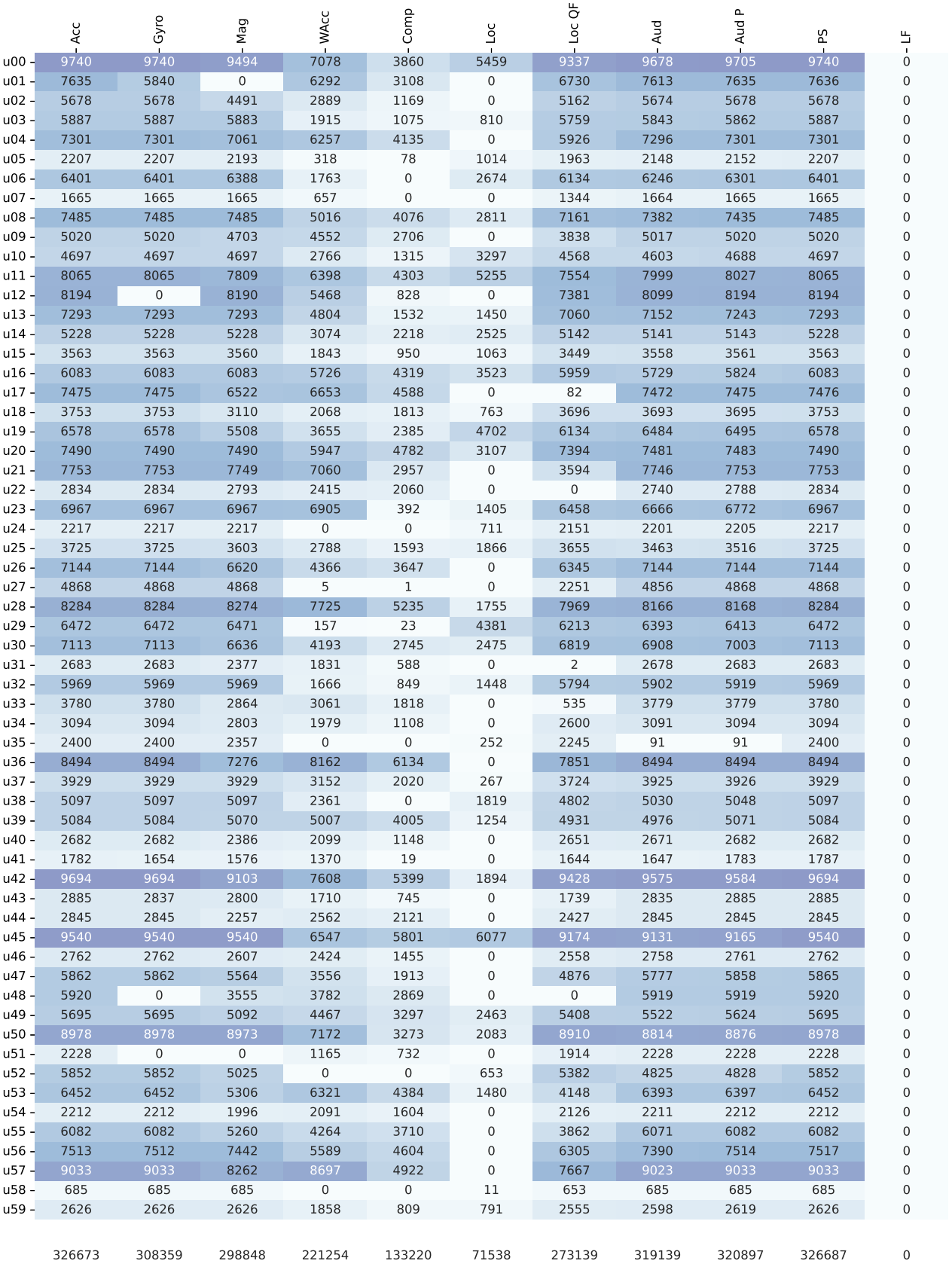


(g) KDE plot visualizing the distribution of each feature
across five users: u01, u04, u21, u29, u45, u55, for the "Walking" activity - Part 7

5.2.3 Missing Sensors

Another factor that should be taken into account in order to successfully train activity recognition models is the aspect of missing sensors. In Figure 5.2.6, we visualize the number of data samples for each sensor, for each user and in total, calculated from the already extracted features of the dataset as provided by the original creators in [VEL17]. This Figure provides a user-level analysis on the data provided in Table 4.2. Some discrepancies between that Table and this Figure can be attributed to the following factors:

- Regarding the total number of examples for each sensor, the discrepancy is due to a particular difference in counting. In the Table 4.2 provided in [VEL17], only the examples that have been annotated with a label of primary (main) activity, that is, an activity of the "Posture/Movement" subset, are counted as labeled and that results in about 308K labeled examples, while in our analysis, all examples that have at least one label (any label) are counted as labeled, totaling at about 326K labeled examples.
- Regarding the difference in counting sensor examples for each sensor, we have adopted a strategy for simplifying the visualization, since it was impossible to visualize the number of examples for each of the 225 sensor features. For each sensor we have counted all the examples that have no `NaN` values for all features derived from the specific sensor. This approach comes handy for visualization, but cannot provide thorough insights for specific sensors, because of the differences between smartphone types, since some sensors and/or sensor features are not the same across both Android and iOS.
  - For the "Location" sensor (features named `location:*`) or "Loc" in the Table, the number of examples appears to be low due to specific missing location features in 27 out of 60 users. More specifically, by analyzing the number of examples for each sensor feature separately, we can see that the features: `location:min_altitude`, `location:max_altitude`, `location:min_speed`, `location:max_speed`, and `location:best_vertical_accuracy` are missing for a lot of users, and that is why the number of examples with all location sensor features available is totaling at only around 71K at the Table.
  - Regarding "Low frequency measurements" (features named `lf_measurements:*`), through our analysis it was shown that while most examples have some values for this type of sensor, there weren't any examples of any user that had all such features available, and that is why all the last column in the Figure is blank. The number of examples for each low frequency sensor feature separately are presented in Table 5.2.



	Acc	Gyro	Mag	WAcc	Comp	Loc	Loc QF	Aud	Aud P	PS	LF
u00	9740	9740	9494	7078	3860	5459	9337	9678	9705	9740	0
u01	7635	5840	0	6292	3108	0	6730	7613	7635	7636	0
u02	5678	5678	4491	2889	1169	0	5162	5674	5678	5678	0
u03	5887	5887	5883	1915	1075	810	5759	5843	5862	5887	0
u04	7301	7301	7061	6257	4135	0	5926	7296	7301	7301	0
u05	2207	2207	2193	318	78	1014	1963	2148	2152	2207	0
u06	6401	6401	6388	1763	0	2674	6134	6246	6301	6401	0
u07	1665	1665	1665	657	0	0	1344	1664	1665	1665	0
u08	7485	7485	7485	5016	4076	2811	7161	7382	7435	7485	0
u09	5020	5020	4703	4552	2706	0	3838	5017	5020	5020	0
u10	4697	4697	4697	2766	1315	3297	4568	4603	4688	4697	0
u11	8065	8065	7809	6398	4303	5255	7554	7999	8027	8065	0
u12	8194	0	8190	5468	828	0	7381	8099	8194	8194	0
u13	7293	7293	7293	4804	1532	1450	7060	7152	7243	7293	0
u14	5228	5228	5228	3074	2218	2525	5142	5141	5143	5228	0
u15	3563	3563	3560	1843	950	1063	3449	3558	3561	3563	0
u16	6083	6083	6083	5726	4319	3523	5959	5729	5824	6083	0
u17	7475	7475	6522	6653	4588	0	82	7472	7475	7476	0
u18	3753	3753	3110	2068	1813	763	3696	3693	3695	3753	0
u19	6578	6578	5508	3655	2385	4702	6134	6484	6495	6578	0
u20	7490	7490	7490	5947	4782	3107	7394	7481	7483	7490	0
u21	7753	7753	7749	7060	2957	0	3594	7746	7753	7753	0
u22	2834	2834	2793	2415	2060	0	0	2740	2788	2834	0
u23	6967	6967	6967	6905	392	1405	6458	6666	6772	6967	0
u24	2217	2217	2217	0	0	711	2151	2201	2205	2217	0
u25	3725	3725	3603	2788	1593	1866	3655	3463	3516	3725	0
u26	7144	7144	6620	4366	3647	0	6345	7144	7144	7144	0
u27	4868	4868	4868	5	1	0	2251	4856	4868	4868	0
u28	8284	8284	8274	7725	5235	1755	7969	8166	8168	8284	0
u29	6472	6472	6471	157	23	4381	6213	6393	6413	6472	0
u30	7113	7113	6636	4193	2745	2475	6819	6908	7003	7113	0
u31	2683	2683	2377	1831	588	0	2	2678	2683	2683	0
u32	5969	5969	5969	1666	849	1448	5794	5902	5919	5969	0
u33	3780	3780	2864	3061	1818	0	535	3779	3779	3780	0
u34	3094	3094	2803	1979	1108	0	2600	3091	3094	3094	0
u35	2400	2400	2357	0	0	252	2245	91	91	2400	0
u36	8494	8494	7276	8162	6134	0	7851	8494	8494	8494	0
u37	3929	3929	3929	3152	2020	267	3724	3925	3926	3929	0
u38	5097	5097	5097	2361	0	1819	4802	5030	5048	5097	0
u39	5084	5084	5070	5007	4005	1254	4931	4976	5071	5084	0
u40	2682	2682	2386	2099	1148	0	2651	2671	2682	2682	0
u41	1782	1654	1576	1370	19	0	1644	1647	1783	1787	0
u42	9694	9694	9103	7608	5399	1894	9428	9575	9584	9694	0
u43	2885	2837	2800	1710	745	0	1739	2835	2885	2885	0
u44	2845	2845	2257	2562	2121	0	2427	2845	2845	2845	0
u45	9540	9540	9540	6547	5801	6077	9174	9131	9165	9540	0
u46	2762	2762	2607	2424	1455	0	2558	2758	2761	2762	0
u47	5862	5862	5564	3556	1913	0	4876	5777	5858	5865	0
u48	5920	0	3555	3782	2869	0	0	5919	5919	5920	0
u49	5695	5695	5092	4467	3297	2463	5408	5522	5624	5695	0
u50	8978	8978	8973	7172	3273	2083	8910	8814	8876	8978	0
u51	2228	0	0	1165	732	0	1914	2228	2228	2228	0
u52	5852	5852	5025	0	0	653	5382	4825	4828	5852	0
u53	6452	6452	5306	6321	4384	1480	4148	6393	6397	6452	0
u54	2212	2212	1996	2091	1604	0	2126	2211	2212	2212	0
u55	6082	6082	5260	4264	3710	0	3862	6071	6082	6082	0
u56	7513	7512	7442	5589	4604	0	6305	7390	7514	7517	0
u57	9033	9033	8262	8697	4922	0	7667	9023	9033	9033	0
u58	685	685	685	0	0	11	653	685	685	685	0
u59	2626	2626	2626	1858	809	791	2555	2598	2619	2626	0
	326673	308359	298848	221254	133220	71538	273139	319139	320897	326687	0

Figure 5.2.6: Quantitative visualization of number of sensor examples per user

Low frequency sensor features' total examples		
Feature	**Examples**	**Users**
`lf_measurements:light`	123,961	25
`lf_measurements:pressure`	92,620	18
`lf_measurements:proximity_cm`	125,351	26
`lf_measurements:proximity`	194,083	34
`lf_measurements:relative_humidity`	19,608	4
`lf_measurements:battery_level`	326,659	60
`lf_measurements:screen_brightness`	194,083	34
`lf_measurements:temperature_ambient`	20,010	4

Table 5.2: Low frequency sensor features and number of examples, separately per feature

5.3 Performance Evaluation

In order to evaluate multi-task, multi-label, in-the-wild HAR performance, we apply a five-fold cross validation scheme, with 48 users in the training set and 12 users in the test set in each iteration. The cross validation (CV) procedure for each iteration can be described as follows:

1. Hold out the selected fold to act as the test set.
2. Train a classifier on the remaining four folds.
3. Apply the classifier to the held out test set.

For each fold and for each label, we counted the numbers of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) of the prediction results over the test set. The numbers of TP, TN, FP and FN are summed over the five-folds, and the following evaluation metrics are calulcated:

- *Accuracy* is the proportion of correctly classified examples out of all the examples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.3.1}$$

- *Precision* (Prec) is the proportion of correctly classified examples out of the examples that the classifier declared as positive:

$$Precision = \frac{TP}{TP + FP} \tag{5.3.2}$$

- *True Positive Rate* (TPR), also called *Sensitivity* or *Recall*, is the proportion of positive examples that were correctly classified as positive:

$$Sensitivity = Recall = TPR = \frac{TP}{TP + FN} \tag{5.3.3}$$

- *True Negative Rate* (TNR), also called *Specificity*, is the proportion of negative examples that were correctly classified as negative:

$$Specificity = TNR = \frac{TN}{TN + FP} \tag{5.3.4}$$

- *F1-score* (F1) is calculated as the harmonic mean of precision and recall:

$$F1 = \frac{2 * TPR * Prec}{TPR + Prec} \tag{5.3.5}$$

- *Balanced Accuracy* (BA) is a measure that accounts for the trade-off between TPR and TNR:

$$BA = \frac{TPR + TNR}{2} \tag{5.3.6}$$

5.4 Baselines

In the following experiments, the "Core" subset of the dataset, which includes all examples that have measurements from all six core sensors, which are the accelerometer (Acc), the gyroscope (Gyro), the location sensor (Loc & Loc QF), the audio recordings (Aud) and the phone state (PS) of the smartphone and the accelerometer (WAcc) of the smartwatch, totaling 169,001 examples, is used. These sensors correspond to 175 out of the 225 precomputed features that are included in the dataset. All the examples of the "Core" set have at least some of the features corresponding to each core sensor. The `NaN` feature values, if existent, are zero-imputated after standardizing the features in each iteration of the five-fold cross validation using the mean and std values of the training set. In this way, the `NaN` feature values are replaced by the mean feature value across the training set.

Python's `NumPy` library [Har+20a] is used to generate random predictions for the random classifier. The majority class classifier and logistic regression models are implemented using `scikit-learn` [Ped+11], a standard Python Machine Learning library. The multilayer perceptron is implemented using `PyTorch` [Pas+19], a popular Machine Learning and Deep Learning framework. Also, `sklearn.metrics` is used to calculate the evaluation metrics for all models' predictions.

5.4.1 Random Chance

Our first step in order to establish baselines for comparison purposes, is to calculate the performance metrics of some dummy classifiers. Python's `numpy.random.Generator.integers` has been used to imitate a random classifier, that declares "relevant" with probability 0.5 independently of the example and the ground-truth label. The evaluation metrics over 20 iterations are presented in Table 5.4 for all context labels, and in Table 5.3 averaged over the labels of each label subset presented in Table 5.1. As mentioned before, chance level of BA is 0.5 for every label, while for the F1-score, the chance level for each label is dependent on the proportion of positive and negative examples in the dataset. We also notice that Accuracy, Sensitivity and Specificity are about 0.5, while Precision is very low, about 0.1, which can also be attributed to the very low proportion of positive examples.

Random Classifier recognition scores averaged for each label subset						
Label	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Posture/Movement	0.500	0.171	0.500	0.500	0.215	0.500
Special Movement	0.500	0.008	0.505	0.500	0.015	0.503
Phone Location	0.500	0.268	0.500	0.500	0.288	0.500
Work-related	0.500	0.087	0.500	0.500	0.142	0.500
Location-based	0.500	0.119	0.501	0.500	0.152	0.501
Transportation	0.500	0.029	0.499	0.500	0.054	0.499
Chores	0.500	0.013	0.498	0.500	0.026	0.499
Self-care	0.500	0.061	0.499	0.500	0.087	0.500
Leisure Time	0.500	0.065	0.500	0.500	0.111	0.500
Companion	0.500	0.103	0.500	0.500	0.167	0.500
Environment	0.500	0.510	0.499	0.502	0.396	0.500
Average	0.500	0.110	0.500	0.500	0.137	0.500

Table 5.3: Recognition scores reported for Random Classifier, averaged for each label subset. 20 simulations of five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

Random Classifier recognition scores for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.500	0.309	0.499	0.500	0.381	0.500
Sitting	79368	0.500	0.469	0.500	0.499	0.484	0.500
Standing	22071	0.500	0.130	0.500	0.500	0.207	0.500
Walking	11715	0.500	0.069	0.499	0.500	0.121	0.499
Running	661	0.500	0.007	0.501	0.500	0.014	0.500
Bicycling	3504	0.499	0.043	0.503	0.499	0.080	0.501
Strolling	339	0.501	0.015	0.500	0.501	0.028	0.501
Stairs - Going up	399	0.500	0.007	0.502	0.500	0.014	0.501
Stairs - Going down	390	0.500	0.007	0.498	0.500	0.013	0.499
Elevator	123	0.499	0.003	0.520	0.499	0.006	0.509
Phone in pocket	14074	0.499	0.180	0.499	0.500	0.264	0.499
Phone in hand	7313	0.500	0.089	0.501	0.500	0.151	0.501
Phone in bag	5031	0.501	0.086	0.499	0.501	0.147	0.500
Phone on table	65979	0.500	0.715	0.500	0.500	0.589	0.500
In class	2852	0.500	0.057	0.500	0.500	0.103	0.500
Lab work	2898	0.501	0.110	0.500	0.501	0.180	0.500
Computer work	22536	0.500	0.159	0.500	0.500	0.241	0.500
In a meeting	2837	0.500	0.023	0.500	0.500	0.044	0.500
At home	80044	0.500	0.479	0.499	0.501	0.489	0.500
At school	25342	0.500	0.181	0.499	0.500	0.266	0.500
At main workplace	19235	0.500	0.201	0.499	0.500	0.287	0.500
At a restaurant	1275	0.500	0.015	0.499	0.500	0.029	0.499
At a bar	520	0.500	0.027	0.508	0.500	0.051	0.504
At a party	404	0.498	0.016	0.496	0.498	0.031	0.497
At the gym	897	0.500	0.028	0.504	0.500	0.053	0.502
At the beach	116	0.501	0.005	0.508	0.500	0.010	0.504
In a car	3550	0.500	0.034	0.497	0.500	0.064	0.499
On a bus	1179	0.499	0.012	0.502	0.499	0.024	0.500
Drive - Driver	4879	0.501	0.052	0.500	0.501	0.094	0.500
Drive - Passenger	1650	0.500	0.018	0.495	0.500	0.035	0.498
Shopping	809	0.500	0.010	0.501	0.500	0.020	0.501
Cooking	2212	0.500	0.018	0.492	0.500	0.034	0.496
Cleaning	1813	0.500	0.021	0.501	0.500	0.040	0.501
Doing laundry	471	0.500	0.009	0.494	0.500	0.017	0.497
Washing dishes	829	0.500	0.010	0.502	0.500	0.019	0.501
Bathing - Shower	1120	0.500	0.010	0.498	0.500	0.019	0.499
Toilet	1558	0.500	0.012	0.498	0.500	0.024	0.499
Grooming	1775	0.501	0.017	0.504	0.501	0.033	0.502
Dressing	1248	0.500	0.011	0.496	0.500	0.021	0.498
Sleeping	40869	0.500	0.256	0.500	0.500	0.339	0.500
Exercise	5191	0.500	0.036	0.498	0.500	0.068	0.499
Eating	9668	0.500	0.060	0.500	0.500	0.107	0.500
Drinking alcohol	859	0.499	0.021	0.502	0.499	0.041	0.501
Watching TV	8945	0.501	0.086	0.500	0.501	0.147	0.500
Surfing the internet	10668	0.500	0.103	0.499	0.500	0.170	0.499
Talking	18477	0.500	0.122	0.499	0.500	0.196	0.500
Singing	384	0.500	0.024	0.502	0.500	0.046	0.501
With co-workers	3972	0.500	0.064	0.502	0.500	0.113	0.501
With friends	12686	0.500	0.143	0.499	0.500	0.222	0.499
Indoors	102510	0.500	0.938	0.500	0.503	0.652	0.501
Outside	6793	0.501	0.081	0.499	0.501	0.140	0.500
Average		0.500	0.110	0.500	0.500	0.137	0.500

Table 5.4: Recognition scores reported for Random Classifier, for all context labels. 20 simulations of five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

5.4.2 Majority Class Classifier

Another dummy classifier which was tested to be used as reference, is the majority class classifier which outputs the majority class seen during training, for all the examples of the test set. In other words, if the majority of the examples seen during training were annotated as "relevant", the classifier predicts "relevant" for all the test set examples. If the majority of the examples seen during training were annotated as "not relevant", the classifier predicts "not relevant" for all the test set examples. We use `sklearn.dummy.DummyClassifier(strategy='most_frequent')` to apply this classifier to the dataset. The evaluation metrics of the five-fold cross validation iterations are presented in Table 5.6 for all context labels, and in Table 5.5 averaged over the labels of each label subset.

As we can see in the Tables, for all labels, in most iterations of the five-fold cross validation procedure, the majority class is "not relevant" and thus all test set examples are labeled as "not relevant". So, for most labels, we do not have any TP or FP predictions, thus Precision cannot be defined and is denoted as `NaN`. From Table 5.6, it is observed that only for "Phone on table" and "Indoors" the majority class is "relevant" in all cross validation iterations, and for "At home" the majority class is "relevant" for some iterations, leading to non-zero Sensitivity values and high F1-score values, which however are not indicative of the high prediction performance of the classifier. Also, we can see that for the remaining 48 labels for which the majority class classifier always predicts "non relevant", Accuracy has high values, which are correlated with the rareness of each label. For labels that are very rare and have very few ground-truth positive examples, the Accuracy of the classifier is very high, but again this is not indicative of high recogntion performance. Since very few positive examples are predicted in total, average F1-score is very low, and BA is again about 0.5 which indicates poor performance.

Majority Class Classifier recognition scores averaged for each label subset

Label	Accuracy	Precision	Sensitivity	Specificity	F1-score	BA
Posture/Movement	0.828	NaN	0.000	1.000	0.000	0.500
Special Movement	0.992	NaN	0.000	1.000	0.000	0.500
Phone Location	0.840	NaN	0.250	0.750	0.209	0.500
Work-related	0.913	NaN	0.000	1.000	0.000	0.500
Location-based	0.874	NaN	0.005	0.981	0.009	0.493
Transportation	0.971	NaN	0.000	1.000	0.000	0.500
Chores	0.987	NaN	0.000	1.000	0.000	0.500
Self-care	0.939	NaN	0.000	1.000	0.000	0.500
Leisure Time	0.935	NaN	0.000	1.000	0.000	0.500
Companion	0.897	NaN	0.000	1.000	0.000	0.500
Environment	0.928	NaN	0.500	0.500	0.484	0.500
Average	0.915	NaN	0.040	0.958	0.037	0.499

Table 5.5: Recognition scores reported for Majority Class Classifier, averaged for each label subset. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

Majority Class Classifier recognition scores for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.691	NaN	0.000	1.000	0.000	0.500
Sitting	79368	0.530	NaN	0.000	1.000	0.000	0.500
Standing	22071	0.869	NaN	0.000	1.000	0.000	0.500
Walking	11715	0.931	NaN	0.000	1.000	0.000	0.500
Running	661	0.993	NaN	0.000	1.000	0.000	0.500
Bicycling	3504	0.957	NaN	0.000	1.000	0.000	0.500
Strolling	339	0.985	NaN	0.000	1.000	0.000	0.500
Stairs - Going up	399	0.993	NaN	0.000	1.000	0.000	0.500
Stairs - Going down	390	0.993	NaN	0.000	1.000	0.000	0.500
Elevator	123	0.997	NaN	0.000	1.000	0.000	0.500
Phone in pocket	14074	0.820	NaN	0.000	1.000	0.000	0.500
Phone in hand	7313	0.911	NaN	0.000	1.000	0.000	0.500
Phone in bag	5031	0.914	NaN	0.000	1.000	0.000	0.500
Phone on table	65979	0.715	0.715	1.000	0.000	0.834	0.500
In class	2852	0.943	NaN	0.000	1.000	0.000	0.500
Lab work	2898	0.890	NaN	0.000	1.000	0.000	0.500
Computer work	22536	0.841	NaN	0.000	1.000	0.000	0.500
In a meeting	2837	0.977	NaN	0.000	1.000	0.000	0.500
At home	80044	0.461	0.204	0.043	0.845	0.071	0.444
At school	25342	0.819	NaN	0.000	1.000	0.000	0.500
At main workplace	19235	0.799	NaN	0.000	1.000	0.000	0.500
At a restaurant	1275	0.985	NaN	0.000	1.000	0.000	0.500
At a bar	520	0.974	NaN	0.000	1.000	0.000	0.500
At a party	404	0.984	NaN	0.000	1.000	0.000	0.500
At the gym	897	0.972	NaN	0.000	1.000	0.000	0.500
At the beach	116	0.995	NaN	0.000	1.000	0.000	0.500
In a car	3550	0.965	NaN	0.000	1.000	0.000	0.500
On a bus	1179	0.988	NaN	0.000	1.000	0.000	0.500
Drive - Driver	4879	0.948	NaN	0.000	1.000	0.000	0.500
Drive - Passenger	1650	0.982	NaN	0.000	1.000	0.000	0.500
Shopping	809	0.990	NaN	0.000	1.000	0.000	0.500
Cooking	2212	0.982	NaN	0.000	1.000	0.000	0.500
Cleaning	1813	0.979	NaN	0.000	1.000	0.000	0.500
Doing laundry	471	0.991	NaN	0.000	1.000	0.000	0.500
Washing dishes	829	0.990	NaN	0.000	1.000	0.000	0.500
Bathing - Shower	1120	0.990	NaN	0.000	1.000	0.000	0.500
Toilet	1558	0.987	NaN	0.000	1.000	0.000	0.500
Grooming	1775	0.983	NaN	0.000	1.000	0.000	0.500
Dressing	1248	0.989	NaN	0.000	1.000	0.000	0.500
Sleeping	40869	0.744	NaN	0.000	1.000	0.000	0.500
Exercise	5191	0.964	NaN	0.000	1.000	0.000	0.500
Eating	9668	0.940	NaN	0.000	1.000	0.000	0.500
Drinking alcohol	859	0.979	NaN	0.000	1.000	0.000	0.500
Watching TV	8945	0.914	NaN	0.000	1.000	0.000	0.500
Surfing the internet	10668	0.897	NaN	0.000	1.000	0.000	0.500
Talking	18477	0.878	NaN	0.000	1.000	0.000	0.500
Singing	384	0.976	NaN	0.000	1.000	0.000	0.500
With co-workers	3972	0.937	NaN	0.000	1.000	0.000	0.500
With friends	12686	0.857	NaN	0.000	1.000	0.000	0.500
Indoors	102510	0.938	0.938	1.000	0.000	0.968	0.500
Outside	6793	0.918	NaN	0.000	1.000	0.000	0.500
Average		0.915	NaN	0.040	0.958	0.037	0.499

Table 5.6: Recognition scores reported for Majority Class Classifier, for all context labels. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

5.4.3 Reproduce Baseline: Logistic Regression

Our next step is to reproduce the baseline models by Vaizman *et al.* referred in Subsections 4.3.5 and 4.3.6. Regarding Logistic Regression, we train binary LR classifiers for each context label. We follow the preprocessing and validation process described in Subsection 4.3.5:

- We standardize each sensor feature (mean and standard deviation are calculated using the training set). After standardization, missing feature values are zero-imputated.
- We implement the *Early Fusion* configuration.
- Five-fold cross validation is used, with each fold containing 48 users in the training set and 12 users in the test set (using the same partition as the original work).
- The training set of each CV iteration is further partitioned using another round of five-fold cross validation (split into 80% internal training subset and 20% internal validation subset), to determine the hyperparameter C out of $\{0.001, 0.01, 0.1, 1, 10, 100\}$ for each LR classifier.
- The selected value of the hyperparameter C for each context label is then used to train the model using the whole training set and evaluate it on the test set, for all CV iterations.
- The total number of learnable parameters is $8976 = 176 \times 51$, since the LR model for each label with 175 inputs (sensor features) has 176 learnable parameters and the number of context labels that we use is 51.
- In the testing phase, the continuous predictions of the model are converted to binary outputs using a threshold of 0.5.
- We calculate the evaluation metrics over non missing entries for each label.

The `scikit-learn` toolkit is used for all the experiments and evaluation pipeline. More specifically, we use `sklearn.linear_model.LogisticRegression(class_weight='balanced')` for Logistic Regression per-label modeling with instance weighting, and `sklearn.model_selection.GridSearchCV` to optimize the hyperparameter C.

Logistic Regression recognition scores averaged for each label subset						
Label	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Posture/Movement	0.832	0.394	0.708	0.832	0.474	0.770
Special Movement	0.898	0.023	0.284	0.903	0.042	0.593
Phone Location	0.784	0.456	0.713	0.789	0.525	0.751
Work-related	0.835	0.289	0.735	0.842	0.402	0.788
Location-based	0.880	0.316	0.601	0.887	0.387	0.744
Transportation	0.876	0.175	0.829	0.877	0.276	0.853
Chores	0.852	0.045	0.508	0.857	0.082	0.682
Self-care	0.833	0.176	0.571	0.836	0.213	0.703
Leisure Time	0.789	0.149	0.531	0.802	0.229	0.666
Companion	0.741	0.196	0.519	0.767	0.280	0.643
Environment	0.885	0.685	0.872	0.864	0.733	0.868
Average	0.839	0.246	0.612	0.844	0.314	0.728

Table 5.7: Recognition scores reported for Logistic Regression (LR) models per label, averaged for each label subset. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

In Table 5.8 we see the recognition scores reported for the LR models per label, for all context labels. Our results are similar to the results of [VEL17] (where descriptive tables with F1-score and BA for all context labels are included) and [VWL18] (where average evaluation metrics over all context labels are compared to the MLP results). Minor differences in the number of positive ground-truth examples (support) per label are attributed to minor differences in examples' filtering to be used in the final dataset for this experiment, while small differences in some evaluation metrics are attributed to discarding the "missing label" examples (according to the rules described in Subsection 4.3.6) which results in a substantially smaller number of

negative examples for rare labels (since a lot of examples that were previously considered as negative, are now discarded as "missing label").

In Table 5.7 we see the recognition scores reported for the LR models per label, averaged over the labels of each label subset. As we can see, the evaluation metrics substantially differ across different types of labels, since some types of labels are easier to predict based on smartphone and smartwatch sensor data, while other are harder.

Logistic Regression recognition scores for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.878	0.773	0.859	0.887	0.814	0.873
Sitting	79368	0.758	0.724	0.784	0.735	0.753	0.759
Standing	22071	0.686	0.246	0.679	0.687	0.361	0.683
Walking	11715	0.832	0.258	0.757	0.838	0.385	0.797
Running	661	0.917	0.030	0.336	0.921	0.056	0.629
Bicycling	3504	0.920	0.331	0.834	0.924	0.474	0.879
Strolling	339	0.851	0.045	0.460	0.856	0.083	0.658
Stairs - Going up	399	0.880	0.018	0.306	0.884	0.034	0.595
Stairs - Going down	390	0.902	0.017	0.238	0.907	0.032	0.573
Elevator	123	0.961	0.010	0.130	0.964	0.018	0.547
Phone in pocket	14074	0.782	0.441	0.785	0.781	0.564	0.783
Phone in hand	7313	0.726	0.190	0.639	0.735	0.293	0.687
Phone in bag	5031	0.832	0.286	0.630	0.851	0.393	0.741
Phone on table	65979	0.795	0.905	0.796	0.791	0.847	0.794
In class	2852	0.896	0.303	0.631	0.912	0.410	0.772
Lab work	2898	0.840	0.397	0.885	0.834	0.548	0.859
Computer work	22536	0.745	0.343	0.663	0.760	0.452	0.712
In a meeting	2837	0.859	0.115	0.761	0.861	0.199	0.811
At home	80044	0.758	0.746	0.748	0.766	0.747	0.757
At school	25342	0.753	0.400	0.721	0.761	0.514	0.741
At main workplace	19235	0.835	0.559	0.835	0.835	0.670	0.835
At a restaurant	1275	0.903	0.103	0.715	0.906	0.180	0.810
At a bar	520	0.952	0.324	0.750	0.958	0.452	0.854
At a party	404	0.923	0.108	0.517	0.930	0.179	0.723
At the gym	897	0.927	0.172	0.428	0.941	0.245	0.685
At the beach	116	0.992	0.117	0.095	0.996	0.105	0.546
In a car	3550	0.897	0.231	0.846	0.899	0.363	0.873
On a bus	1179	0.858	0.068	0.818	0.858	0.125	0.838
Drive - Driver	4879	0.893	0.305	0.843	0.895	0.448	0.869
Drive - Passenger	1650	0.855	0.095	0.810	0.856	0.170	0.833
Shopping	809	0.854	0.043	0.628	0.856	0.080	0.742
Cooking	2212	0.838	0.067	0.629	0.841	0.122	0.735
Cleaning	1813	0.834	0.054	0.424	0.843	0.096	0.633
Doing laundry	471	0.876	0.032	0.456	0.880	0.061	0.668
Washing dishes	829	0.858	0.028	0.402	0.862	0.052	0.632
Bathing - Shower	1120	0.826	0.026	0.454	0.830	0.049	0.642
Toilet	1558	0.783	0.030	0.512	0.787	0.056	0.649
Grooming	1775	0.814	0.044	0.482	0.820	0.080	0.651
Dressing	1248	0.846	0.036	0.511	0.850	0.068	0.680
Sleeping	40869	0.894	0.744	0.895	0.894	0.812	0.894
Exercise	5191	0.878	0.194	0.744	0.883	0.307	0.813
Eating	9668	0.676	0.114	0.649	0.678	0.194	0.663
Drinking alcohol	859	0.876	0.089	0.530	0.883	0.152	0.706
Watching TV	8945	0.778	0.203	0.538	0.801	0.294	0.670
Surfing the internet	10668	0.721	0.181	0.487	0.748	0.264	0.617
Talking	18477	0.676	0.222	0.658	0.678	0.332	0.668
Singing	384	0.921	0.043	0.109	0.941	0.062	0.525
With co-workers	3972	0.779	0.153	0.549	0.794	0.239	0.672
With friends	12686	0.704	0.239	0.488	0.740	0.321	0.614
Indoors	102510	0.894	0.989	0.896	0.851	0.941	0.874
Outside	6793	0.876	0.381	0.847	0.878	0.526	0.862
Average		0.839	0.246	0.612	0.844	0.314	0.728

Table 5.8: Recognition scores reported for Logistic Regression (LR) models per label, for all context labels. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

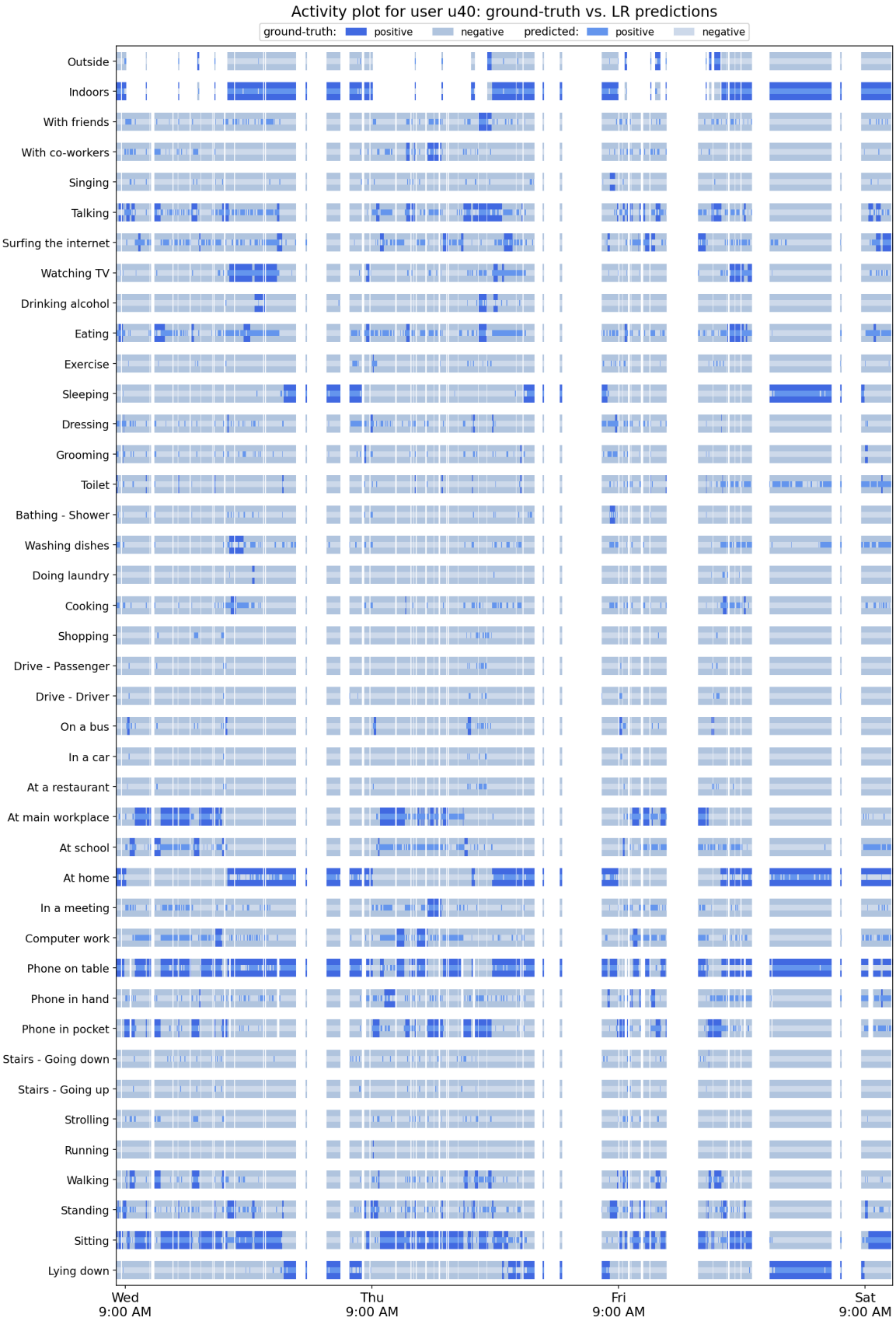


Figure 5.4.1: Activity plot including LR predictions for user u40

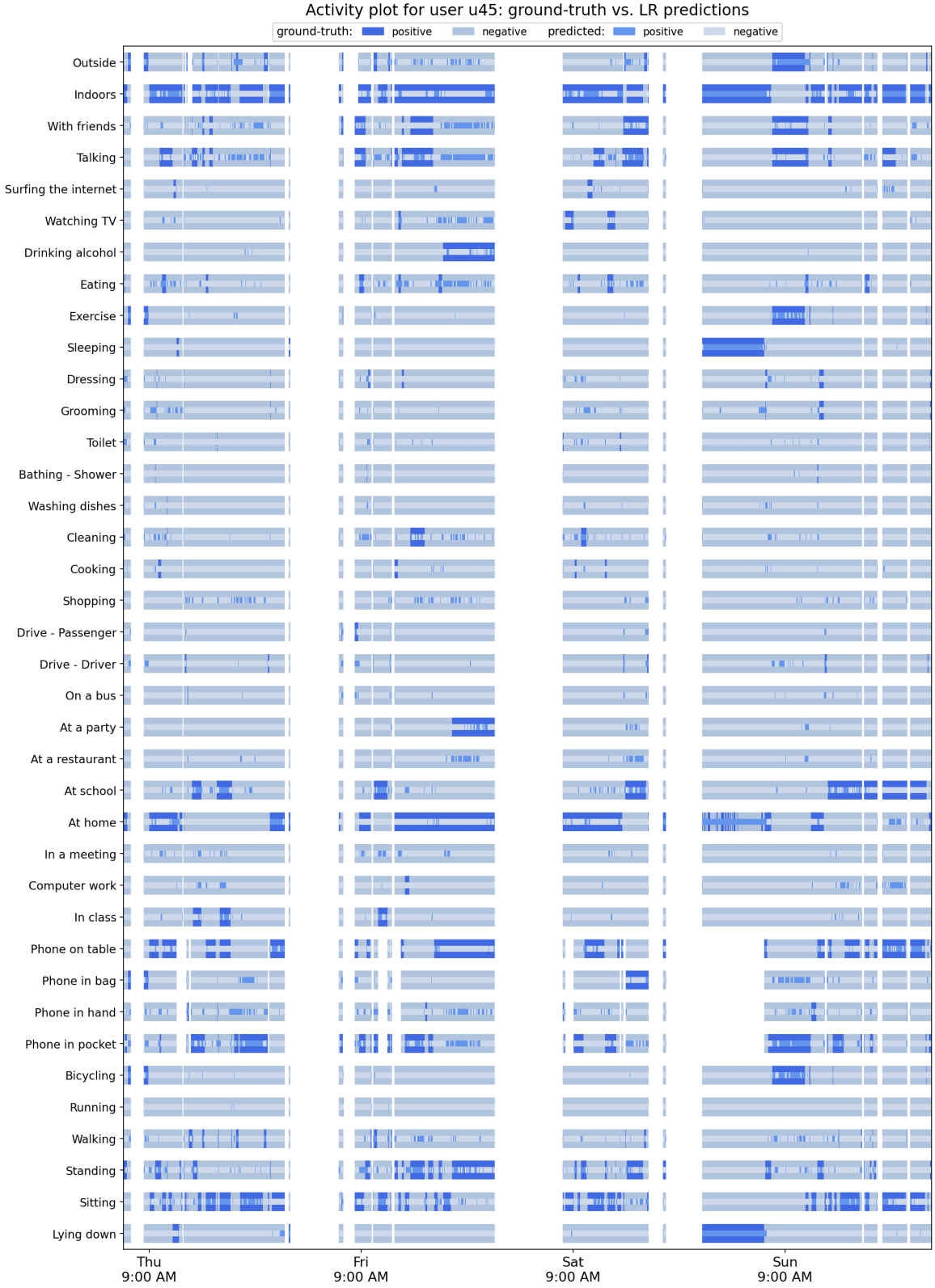


Figure 5.4.2: Activity plot including LR predictions for user u45

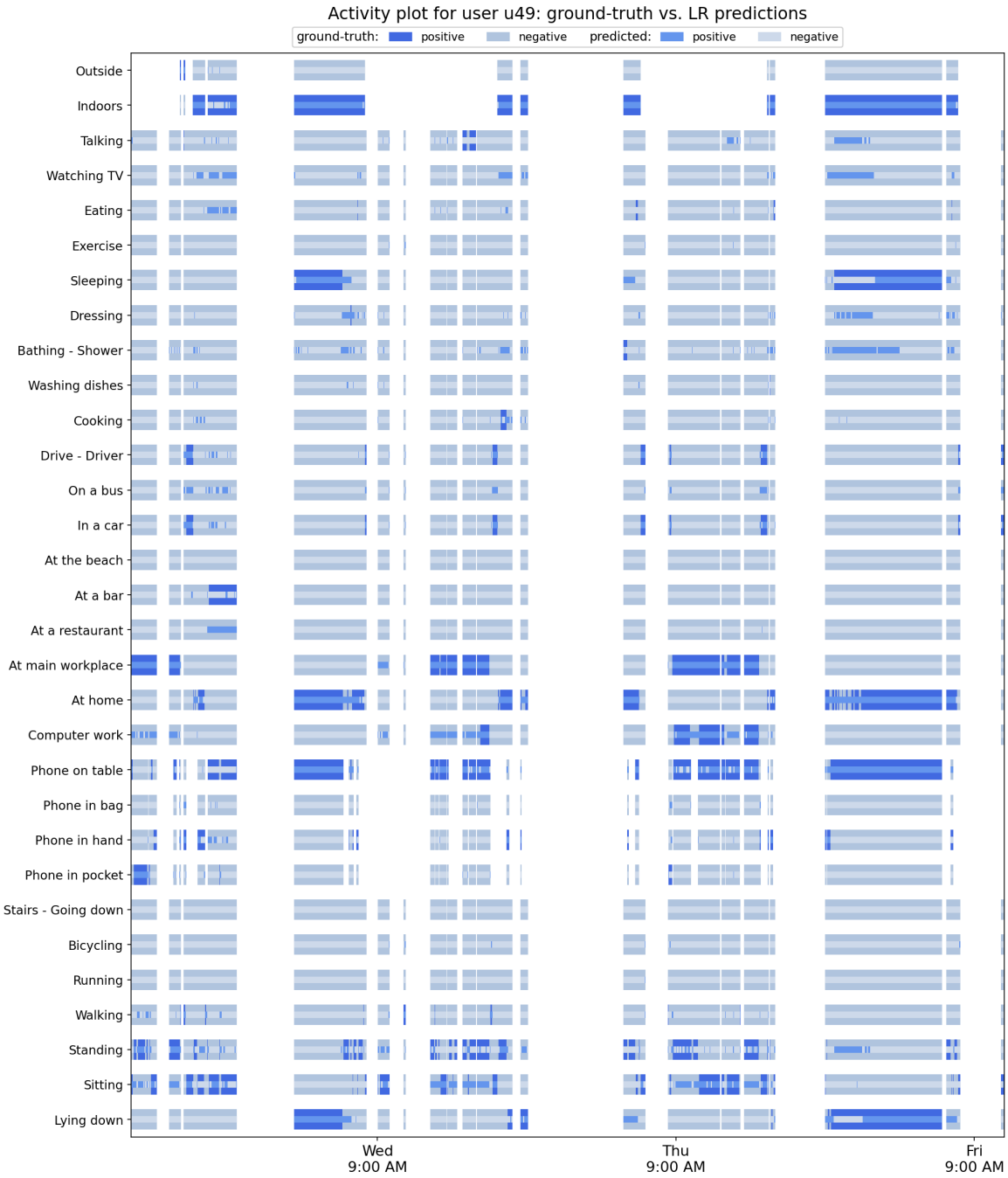


Figure 5.4.3: Activity plot including LR predictions for user u49

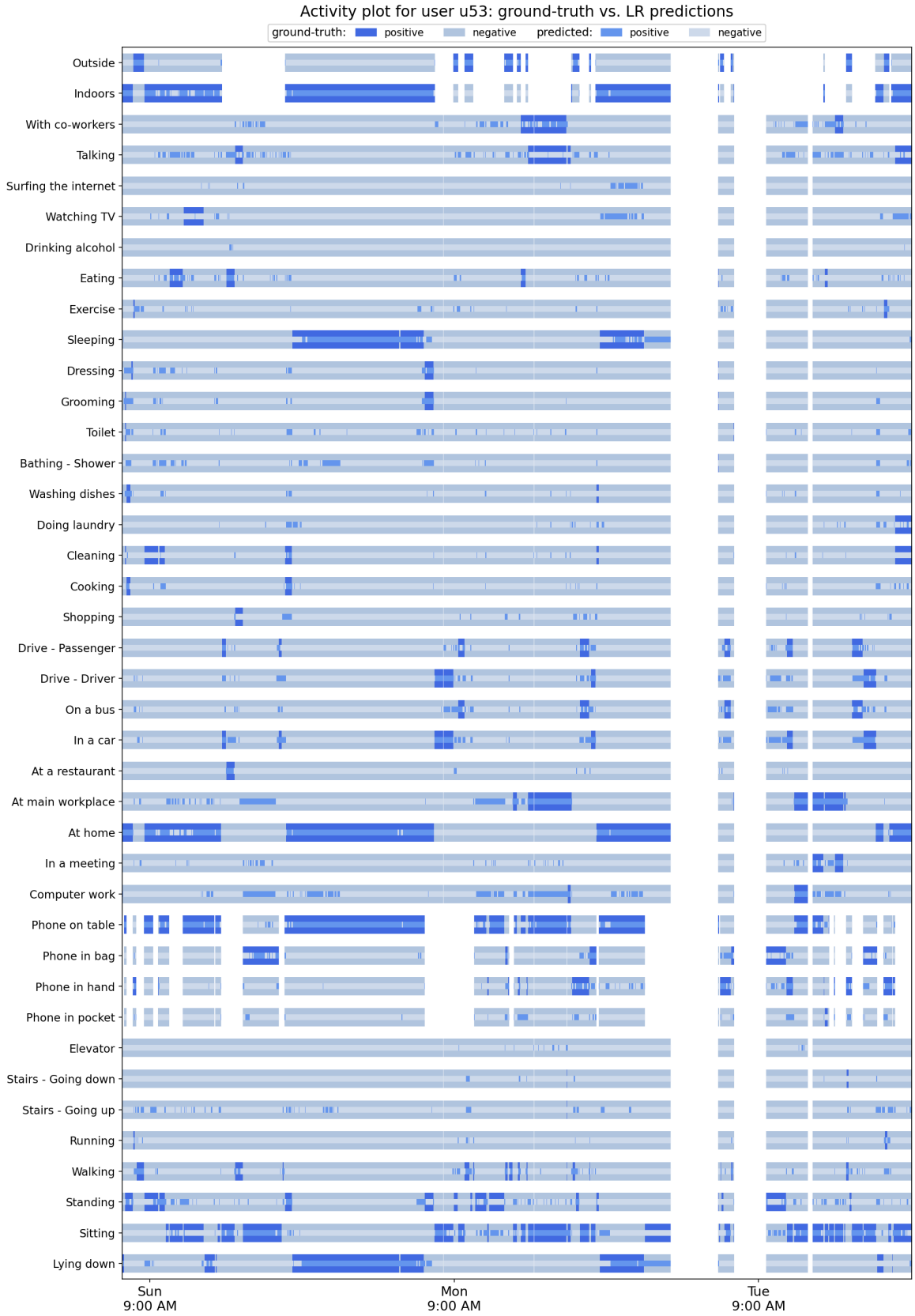


Figure 5.4.4: Activity plot including LR predictions for user u53

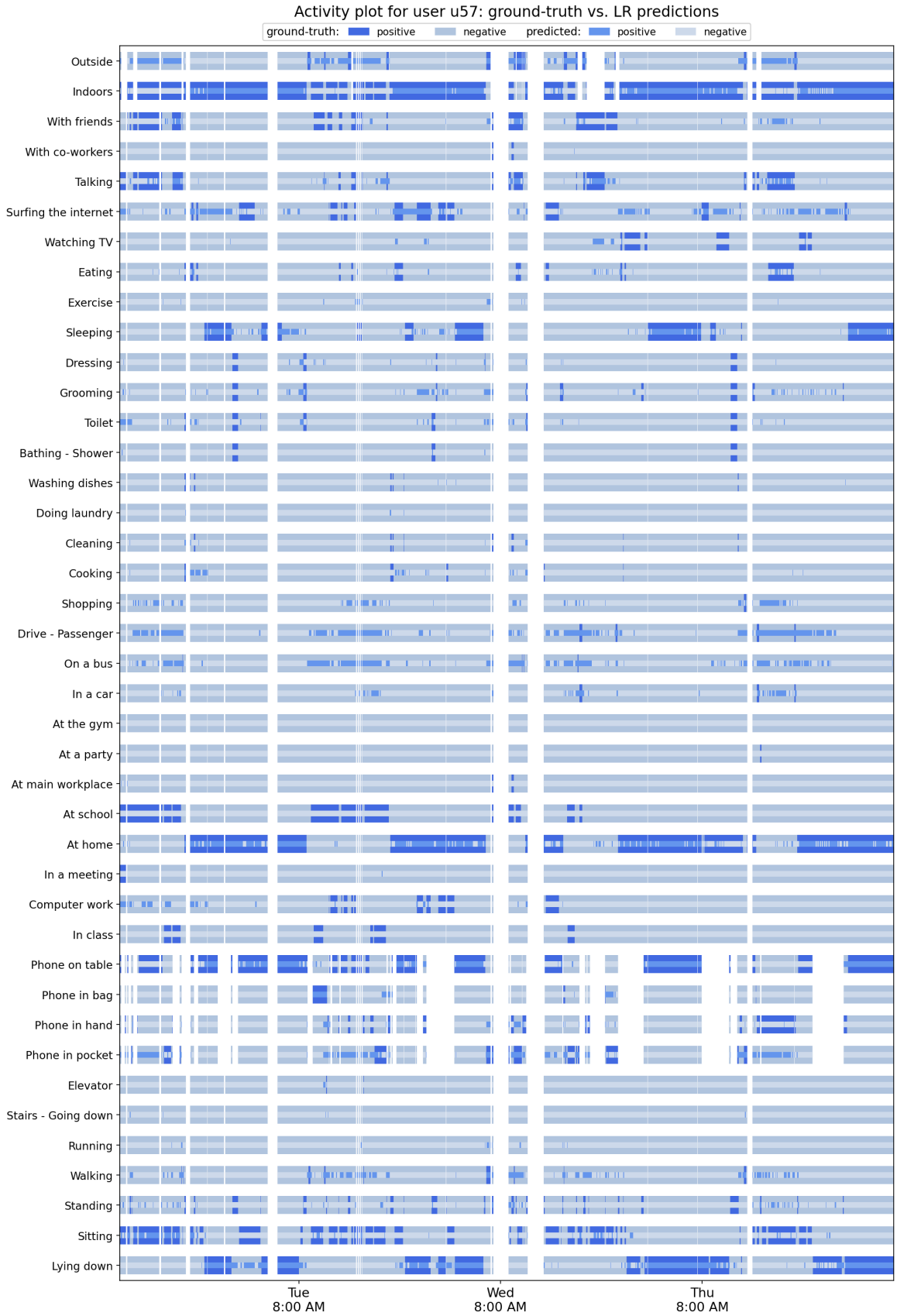


Figure 5.4.5: Activity plot including LR predictions for user u57

5.4.4 Reproduce Baseline: Multilayer Perceptron

In our next experiment, we reproduce a baseline MLP according to [VWL18], which is presented in Subsection 4.3.6. More specifically, we implement, train and test the the MLP(16,16) configuration, which has the best recognition performance in the reference paper. In general, we follow the process as originally described. The key points for this experiment are summarized below:

- Five-fold cross validation is used, with each fold containing 48 users in the training set and 12 users in the test set (using the same partition as the original work).
- The training set of each CV iteration is further partitioned in a training subset and a validation subset, as follows: for each of the 48 users of the training set, 80% of their data is used for training, and 20% is used for validation, to decide which epoch's checkpoint will be selected for the test inference.
- We standardize each sensor feature (mean and standard deviation are calculated using the training subset). After standardization, missing feature values are zero-imputated.
- The MLP(16,16) configuration includes two layers of 16 hidden nodes each, followed by an output layer of 51 nodes which correspond to the 51 context labels. As for the activation function, we use Leaky ReLU as in the original work, and we have added Batch Normalization and Dropout layers, as we found them to facilitate convergence and help avoid overfitting. The complete architecture is depicted in Figure 5.4.6.
- The total number of trainable parameters is 4019, out of which 3955 belong to the Linear layers and 64 belong to the Batch Normalization layers.
- Regarding the loss, we implement a custom loss based on `torch.nn.BCEWithLogitsLoss`, slightly modified to allow dynamic per-batch, per-element masking in the loss matrices (to mask the loss elements corresponding to missing ground-truth labels for each batch, and not use them in the loss computation). Also, the `pos_weight` is used for instance-weighting to account for the imbalance in the number of positive examples per context label, by multiplying the term of the positive examples in the loss, with the ratio of negative to positive examples for this label in the training set.

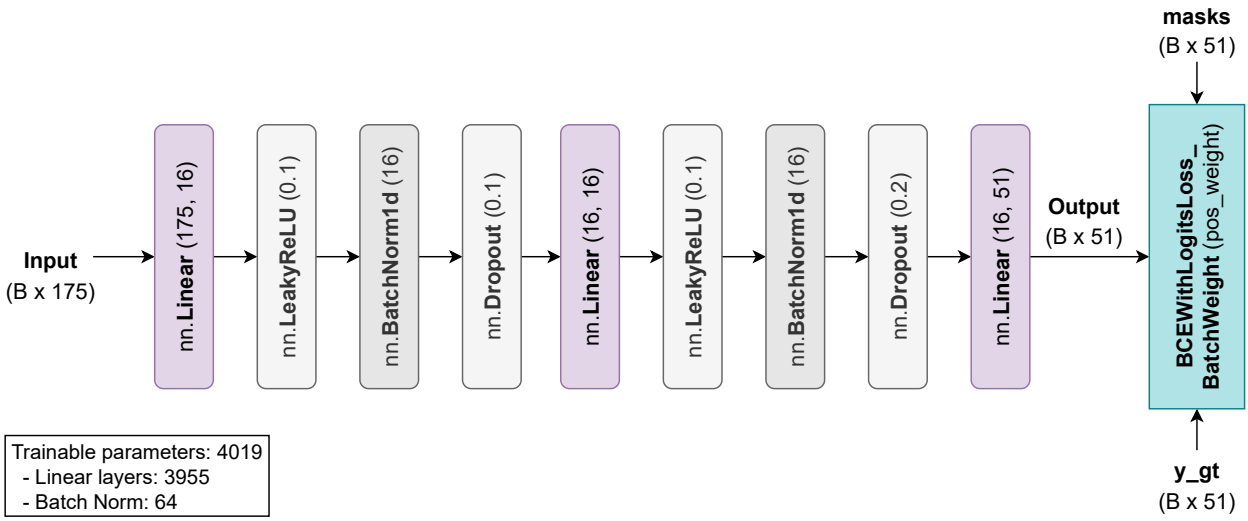


Figure 5.4.6: The MLP(16, 16) model architecture used as baseline, similar to the one used in [VWL18]. Two hidden layers of 16 nodes are followed by an output layer of 51 nodes corresponding to the 51 context labels. Leaky ReLU is used as an activation function, followed by Batch Normalization and Dropout layers, to facilitate convergence and avoid overfitting. A custom loss based on `torch.nn.BCEWithLogitsLoss` is used, but with a slight modification to allow per-batch, per-element masking in the loss matrices (to leverage missing labels information). Also, the `pos_weight` argument is used for instance-weighting to account for the imbalance in the number of positive examples per context label.

- We use a `batch size` of 32 to train the model.
- We use the Adam [KB15] optimizer with `learning rate = 0.01`, instead of Stochastic Gradient Descent, to train the model.
- In the testing phase, the continuous predictions of the model pass through a sigmoid activation function, and then they are converted to binary outputs using a threshold of 0.5.
- We calculate the evaluation metrics for each label over its non-missing ground-truth examples.

Since `scikit-learn` does not provide instance-weighting and loss masking for MLPs, we use `PyTorch` [Pas+19] to build the model, the supervisor and the training scripts. The `torch.nn` modules that are used, can be seen in detail in Figure 5.4.6.

In Table 5.10 we see the recognition scores reported for the MLP, for all context labels, and in Table 5.9 the same results are averaged over the labels of each label subset. Our result of 0.772 average BA is comparable to the original paper's value of 0.773, while we also provide the full set of metrics over each context label and in total, while in the original paper the only metric reported is BA.

In comparison with the results of the LR models per label, the multi-label MLP achieves an important improvement in BA, using far less parameters in total (4019 in the MLP vs. 8976 for the LR per label models). In the LR experiments, we created a single LR model per label, thus each label is modeled independently, while in the MLP, all labels are modeled together, sharing the hidden layers' parameters. This results in improved Sensitivity, which means that the model is capable of predicting correctly more positive examples per label. In LR, we noticed an imbalance between Sensitivity and Specificity; Sensitivity was 0.612, while Specificity was 0.844. In MLP, this imbalance is mitigated and the values are much closer: 0.757 and 0.786, respectively. This effect can be attributed to the instance-weighting applied in the loss function, that equalizes the contribution of positive and negative examples for each label in the loss function. We notice that when we train the same model, but without instance-weighting, the classifier optimizes raw Accuracy, becomes almost trivial, and predicts only the most common labels, while ignoring the rare ones.

Regarding the detailed results per context label and per label subsets for the MLP, the increase in Sensitivity values compared to LR, is mostly observed in the rarer labels, resulting also in higher BA for those labels, compared to the previous baseline.

MLP recognition scores averaged for each label subset

Label	Accuracy	Precision	Sensitivity	Specificity	F1-score	BA
Posture/Movement	0.792	0.374	0.803	0.787	0.450	0.795
Special Movement	0.804	0.027	0.719	0.805	0.052	0.762
Phone Location	0.726	0.427	0.782	0.730	0.498	0.756
Work-related	0.795	0.248	0.793	0.793	0.365	0.793
Location-based	0.822	0.265	0.789	0.817	0.340	0.803
Transportation	0.852	0.151	0.862	0.852	0.247	0.857
Chores	0.797	0.041	0.653	0.799	0.077	0.726
Self-care	0.805	0.186	0.711	0.808	0.222	0.759
Leisure Time	0.711	0.144	0.709	0.712	0.231	0.711
Companion	0.687	0.190	0.672	0.688	0.295	0.680
Environment	0.852	0.663	0.873	0.868	0.703	0.870
Average	0.786	0.228	0.757	0.786	0.298	0.772

Table 5.9: Recognition scores reported for the simple MLP architecture, averaged for each label subset. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

MLP recognition scores for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.888	0.813	0.829	0.915	0.821	0.872
Sitting	79368	0.767	0.730	0.799	0.739	0.763	0.769
Standing	22071	0.619	0.222	0.767	0.597	0.345	0.682
Walking	11715	0.778	0.213	0.823	0.774	0.339	0.799
Running	661	0.829	0.029	0.699	0.830	0.056	0.764
Bicycling	3504	0.871	0.238	0.900	0.870	0.377	0.885
Strolling	339	0.774	0.048	0.776	0.774	0.091	0.775
Stairs - Going up	399	0.810	0.024	0.667	0.811	0.047	0.739
Stairs - Going down	390	0.824	0.024	0.636	0.826	0.047	0.731
Elevator	123	0.809	0.012	0.797	0.809	0.023	0.803
Phone in pocket	14074	0.753	0.410	0.844	0.733	0.552	0.789
Phone in hand	7313	0.630	0.164	0.770	0.617	0.270	0.693
Phone in bag	5031	0.747	0.220	0.758	0.746	0.341	0.752
Phone on table	65979	0.776	0.915	0.756	0.824	0.828	0.790
In class	2852	0.803	0.190	0.755	0.806	0.304	0.780
Lab work	2898	0.809	0.354	0.891	0.799	0.506	0.845
Computer work	22536	0.735	0.345	0.744	0.733	0.471	0.739
In a meeting	2837	0.835	0.101	0.784	0.836	0.180	0.810
At home	80044	0.786	0.762	0.805	0.768	0.783	0.787
At school	25342	0.718	0.370	0.793	0.701	0.504	0.747
At main workplace	19235	0.835	0.558	0.862	0.828	0.677	0.845
At a restaurant	1275	0.854	0.076	0.798	0.855	0.139	0.827
At a bar	520	0.870	0.153	0.873	0.870	0.261	0.871
At a party	404	0.785	0.053	0.733	0.786	0.100	0.759
At the gym	897	0.852	0.119	0.679	0.857	0.202	0.768
At the beach	116	0.873	0.030	0.767	0.874	0.057	0.820
In a car	3550	0.862	0.180	0.843	0.862	0.297	0.853
On a bus	1179	0.827	0.060	0.878	0.826	0.112	0.852
Drive - Driver	4879	0.871	0.268	0.859	0.872	0.408	0.866
Drive - Passenger	1650	0.846	0.095	0.868	0.846	0.171	0.857
Shopping	809	0.779	0.036	0.820	0.779	0.070	0.799
Cooking	2212	0.780	0.055	0.695	0.782	0.102	0.738
Cleaning	1813	0.771	0.056	0.630	0.774	0.102	0.702
Doing laundry	471	0.854	0.028	0.469	0.858	0.053	0.663
Washing dishes	829	0.801	0.031	0.650	0.802	0.060	0.726
Bathing - Shower	1120	0.806	0.033	0.663	0.807	0.063	0.735
Toilet	1558	0.714	0.031	0.730	0.714	0.060	0.722
Grooming	1775	0.783	0.049	0.640	0.786	0.091	0.713
Dressing	1248	0.815	0.038	0.649	0.817	0.071	0.733
Sleeping	40869	0.904	0.779	0.874	0.915	0.824	0.894
Exercise	5191	0.846	0.169	0.819	0.848	0.280	0.833
Eating	9668	0.619	0.109	0.747	0.611	0.190	0.679
Drinking alcohol	859	0.807	0.089	0.885	0.805	0.162	0.845
Watching TV	8945	0.781	0.224	0.631	0.795	0.331	0.713
Surfing the internet	10668	0.690	0.174	0.538	0.708	0.263	0.623
Talking	18477	0.613	0.207	0.765	0.592	0.326	0.678
Singing	384	0.624	0.036	0.581	0.625	0.069	0.603
With co-workers	3972	0.757	0.170	0.732	0.759	0.276	0.745
With friends	12686	0.616	0.211	0.613	0.617	0.314	0.615
Indoors	102510	0.857	0.992	0.855	0.893	0.918	0.874
Outside	6793	0.847	0.335	0.890	0.843	0.487	0.867
Average		0.786	0.228	0.757	0.786	0.298	0.772

Table 5.10: Recognition scores reported for the simple MLP architecture, for all context labels. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

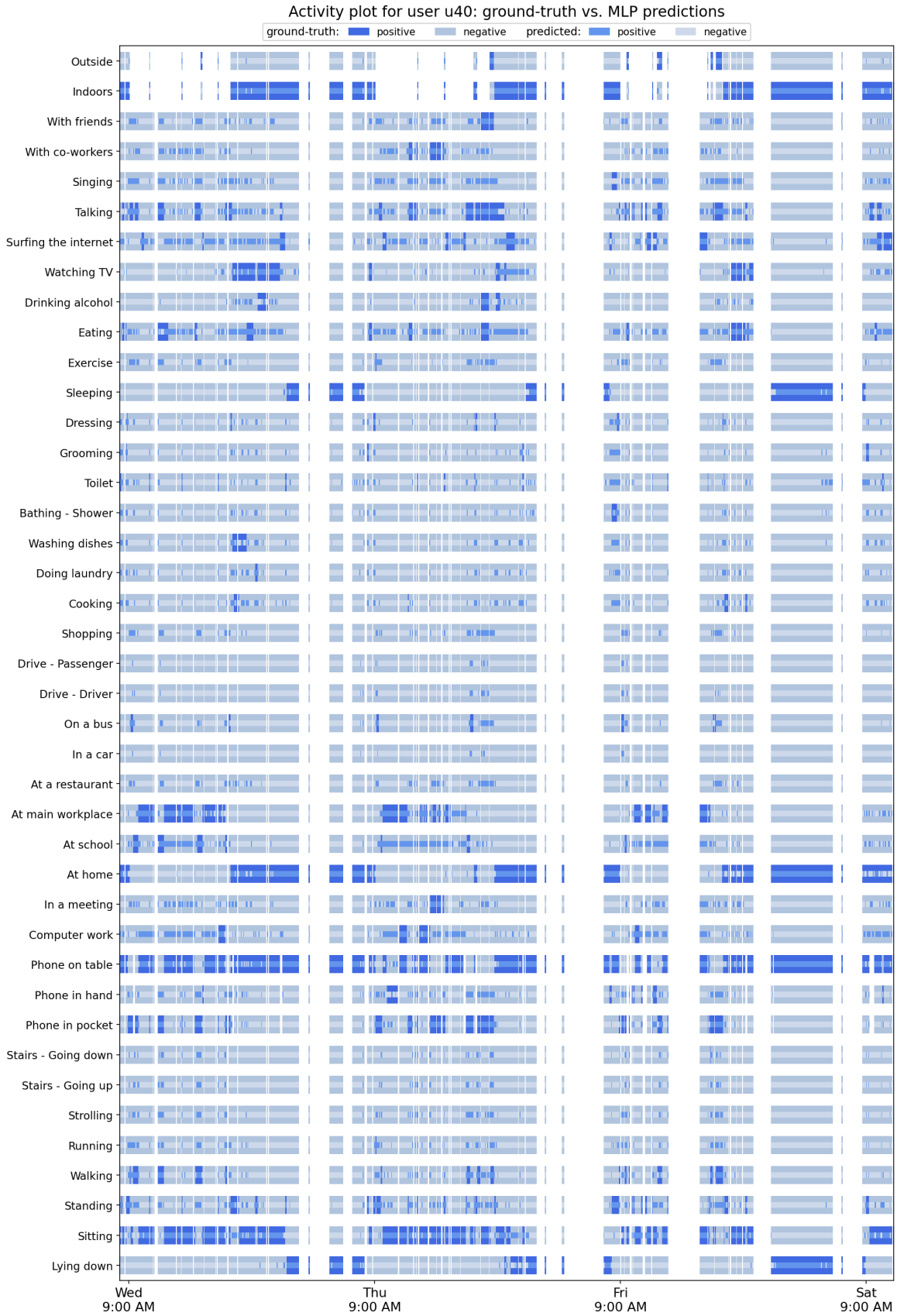


Figure 5.4.7: Activity plot including MLP predictions for user u40

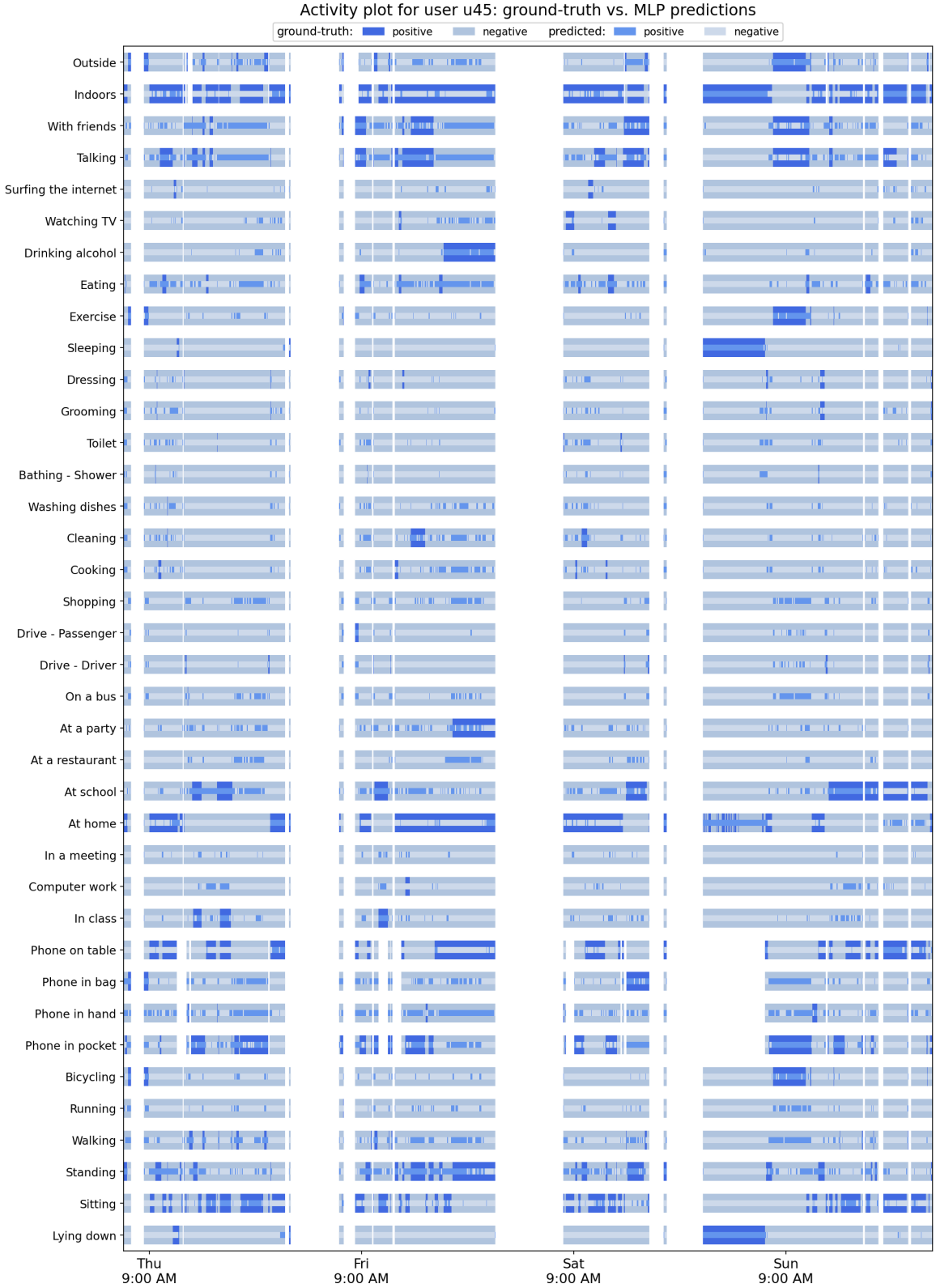


Figure 5.4.8: Activity plot including MLP predictions for user u45

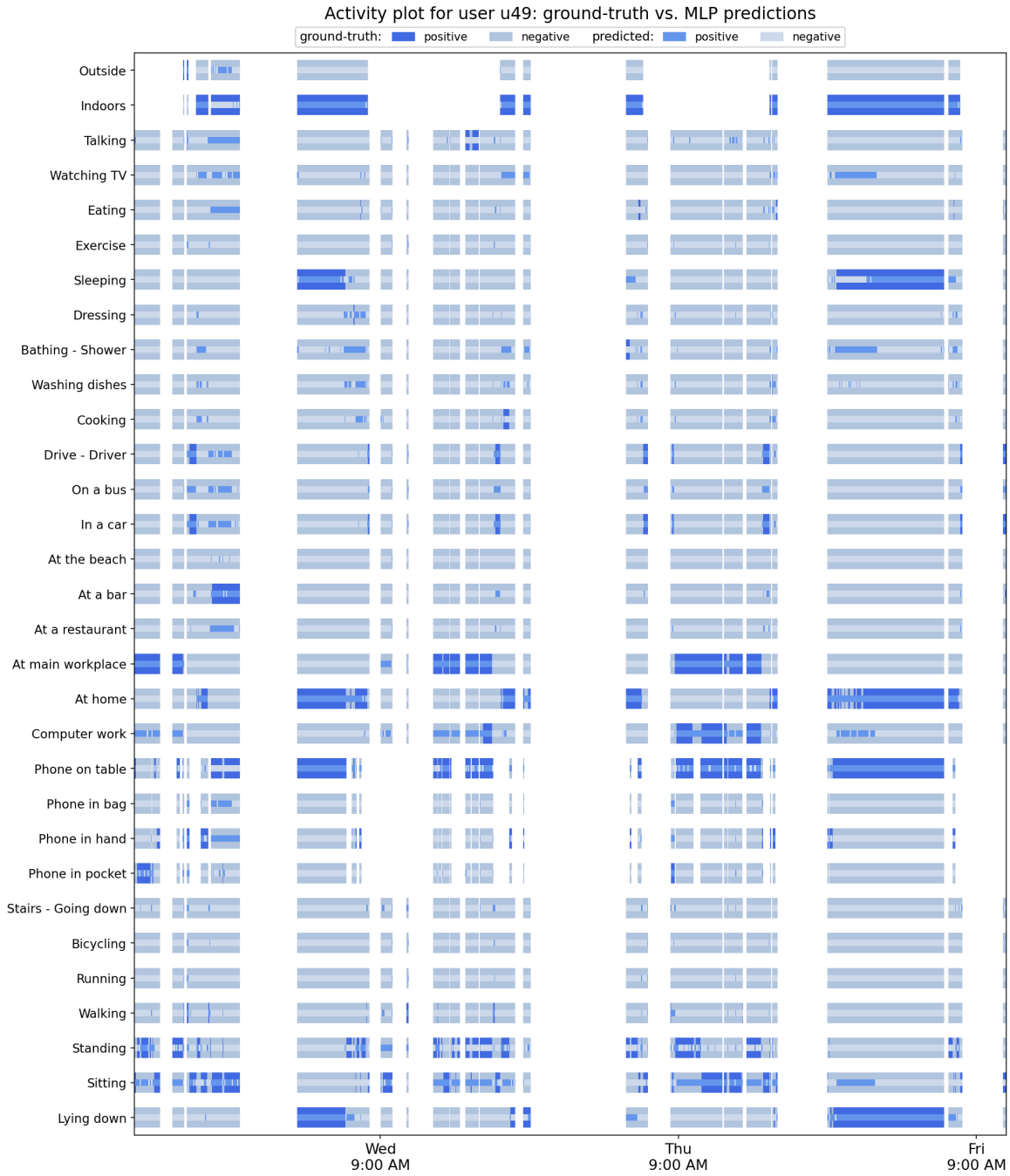


Figure 5.4.9: Activity plot including MLP predictions for user u49

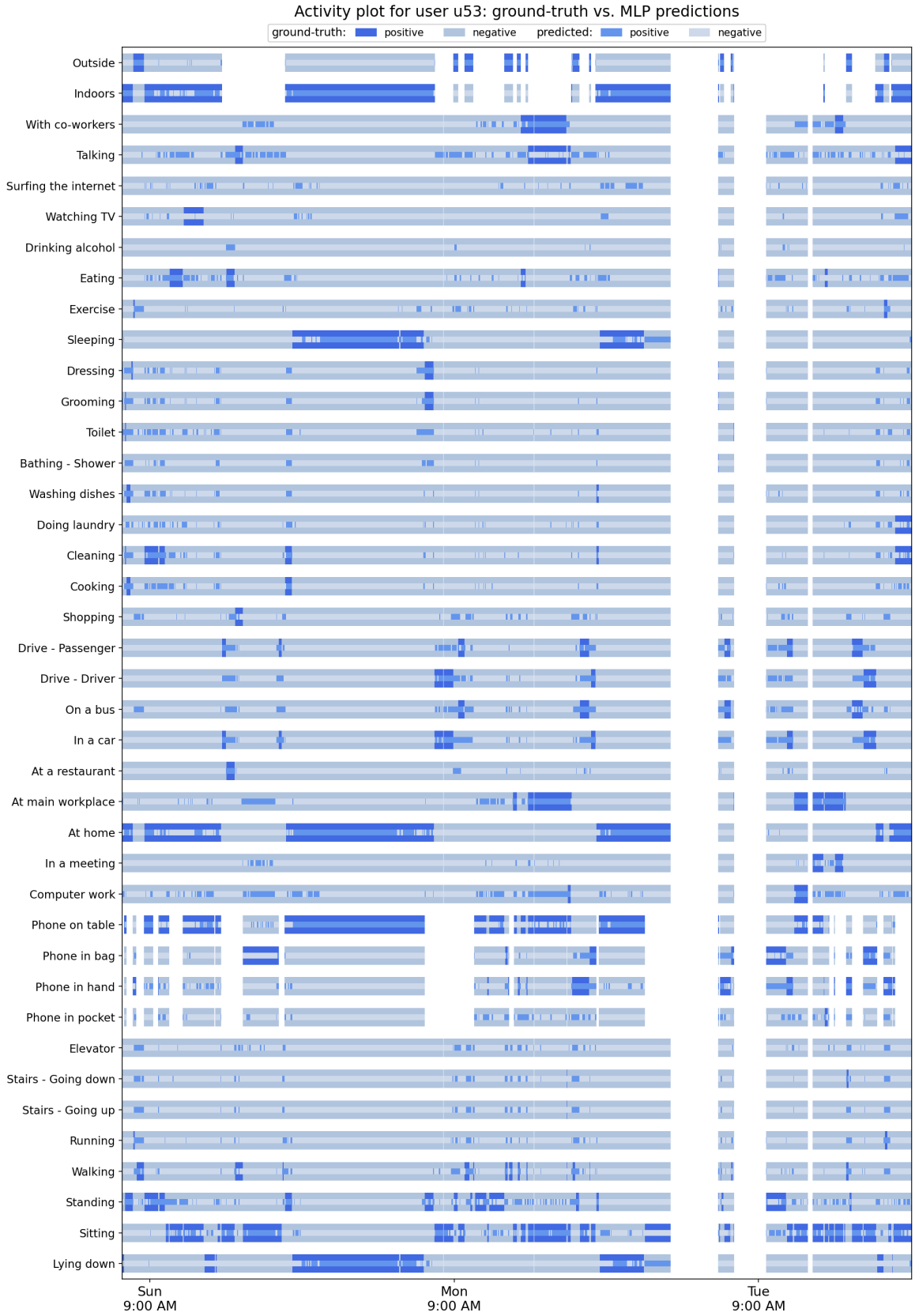


Figure 5.4.10: Activity plot including MLP predictions for user u53

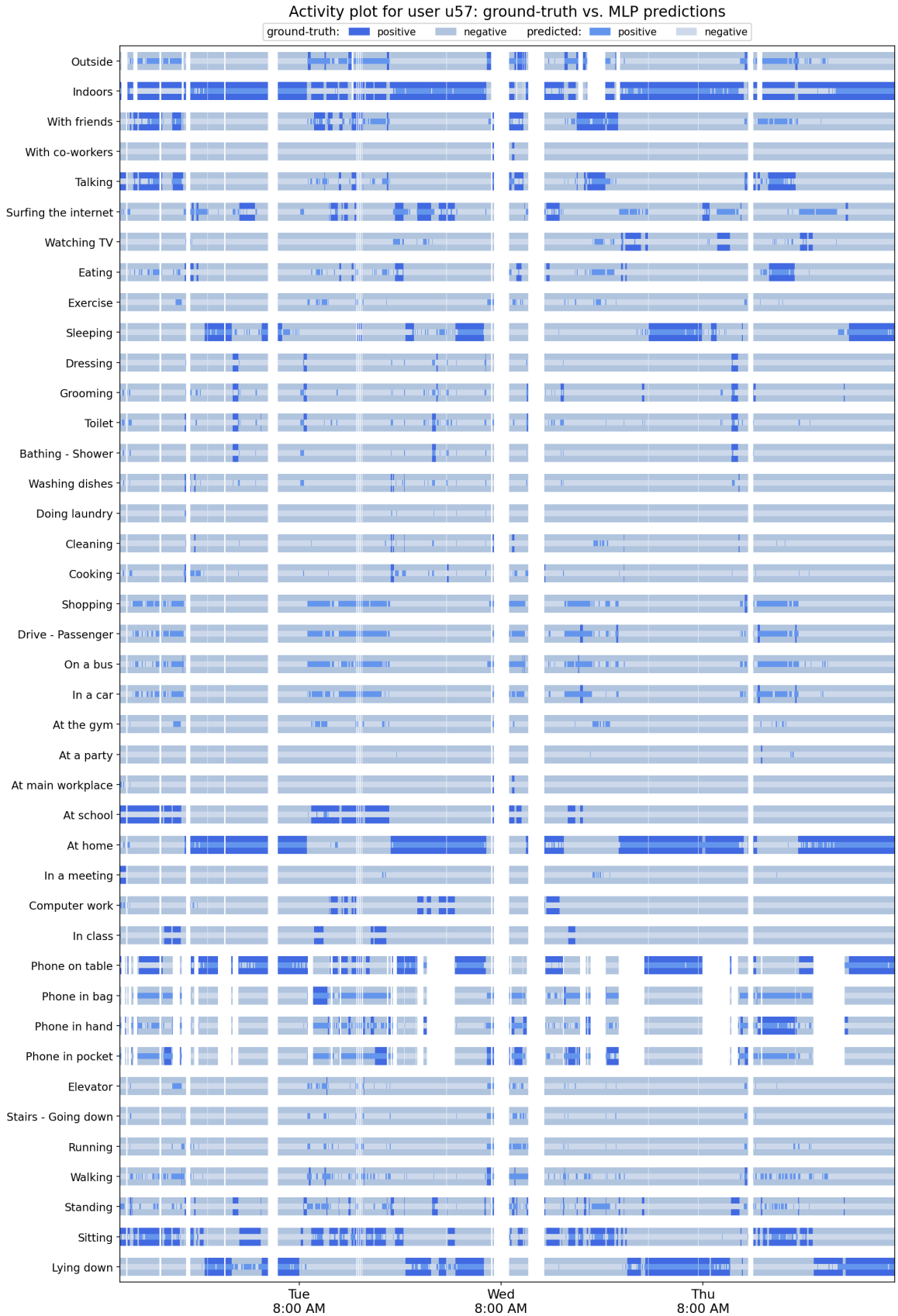


Figure 5.4.11: Activity plot including MLP predictions for user u57

5.5 Bidirectional LSTM

The most profound way to leverage the sequential format of the data samples is to train a recurrent architecture using the already extracted sensor features. We use the bidirectional Long Short-Term Memory (BiLSTM) to implement two HAR models, combining a two-layer BiLSTM with a Linear layer.

In these experiments we again use the aforementioned "Core" subset of the dataset, which includes all examples that have measurements from all six core sensors (Acc, Gyro, WAcc, Loc & Loc QF, Aud, PS), totaling 169,001 examples. These sensors correspond to 175 out of the 225 precomputed features that are included in the dataset. We follow the same preprocessing steps as described in Section 5.4. All the examples of the "Core" set have at least some of the features corresponding to each core sensor. The `NaN` feature values, if existent, are zero-imputated after standardizing the features in each iteration of the five-fold cross validation using the mean and std values of the training set. In this way, the `NaN` feature values are replaced by the mean feature value across the training set.

However, to abide by the recurrent logic introduced here, we now need to provide sequences of examples as input to our models, instead of the single examples we provided in our previous, baseline models. Thus, we can approach our data as time series of examples, and we need to select a `N`-length sequence of consecutive data points as our models' input. Since each of the examples of the dataset corresponds to a single minute for a specific user, it is intuitive to select `N` consecutive examples to create our model input sequence. However, the ExtraSensory dataset is characterized by an intrinsic difficulty for this task, since there are missing examples due to the *in-the-wild* nature of the dataset and the recording conditions (the participants could start and stop recording their daily routines as preferable and convenient). In addition, the "Core" subset which is used, is characterized by a second intrinsic difficulty, since we filter out examples with missing core sensors, and as a result, we end up with even more missing examples.

For these reasons, we devise a strategy to solve the missing examples issue when trying to create `N`-length sequences of consecutive examples. Each example in the dataset, containing 225 features in total, and 175 features in the "Core" subset, is accompanied by the `user_id` of the corresponding user, and the `timestamp` which contains all the date-time information for the specific example, with second precision. First, we convert these timestamps to minute precision, to obtain the exact minute corresponding to each example. Then, for each dataset example at time `t` for user `u`, we attempt to create a `N`-length sequence of consecutive examples by gathering the examples of user `u` for minutes {`t-N+1`, ..., `t`}. If the example corresponding to a specific past minute is missing, the next example is repeated, and this process is repeated recursively until the sequence is created.

We choose to include only past examples in the time sequence of each example, in order not to impose any constraints on real-time use, where we only have data for the current minute and the past minutes, we do not have access to data in the future. When filling the missing examples in the sequence, we choose to repeat the next example and not the previous example, because we want to pay attention more to the examples closer to the current minute, and thus, if an example must be repeated to fill the missing example, it makes more sense to repeat the next one, than to repeat the previous one or to fill the missing example with zeros.

In Figure 5.5.1 we can see an example of generation of {5, 10, 15}-length sequences of consecutive examples, in cases where the examples for some timestamps are missing. It can be seen in the left, that the examples for minute precision timestamps 24077947, 24077949 and 24077950 are missing for this user. In the right, it is shown that when generating an examples sequence for the selected timestamp (minute) 24077959, in the 10-length sequence and 15-length sequence cases, the example corresponding to the next minute is repeated to fill the missing example 24077951 is repeated for timestamps 24077950 and 24077949 which are missing, 24077948 is repeated for timestamp 24077947 which is also missing.

Therefore, it becomes possible to create examples sequences of equal length `N` for all examples of the dataset, even in cases where some or all examples in the immediate past of an example are missing. This way, it is much easier to handle sequential data to be used as input for recurrent NN architectures, since they can be combined in batches and stored in `Numpy arrays` and `PyTorch Tensors`. Otherwise, if we split the examples sequence every time missing examples appear, we would end up with examples sequences of very variable length, that would require a lot of padding and elaborate handling to be used efficiently in recurrent NN architectures.

Figure 5.5.1: An example of generation of {5, 10, 15}-length sequences of consecutive examples, in cases where the examples for some timestamps are missing. It can be seen in the left, that the examples for minute precision timestamps 24077947, 24077949 and 24077950 are missing for this user. In the right, it is shown that when generating an examples sequence for the selected timestamp (minute) 24077959, in the 10-length sequence and 15-length sequence cases, the example corresponding to the next minute is repeated to fill the missing example.

5.5.1 Using BiLSTM Final Hidden States

At first, we implement a model including a two-layer BiLSTM, whose `h_n` output, which contains a concatenation of the final forward and reverse hidden states of each BiLSTM layer, is then fed to a Linear layer, to predict the probabilities for each of the 51 context labels. This model is conceptually presented in Figure 5.5.2, and the key points of the experiments are summarized as follows:

- Five-fold cross validation is used, with each fold containing 48 users in the training set and 12 users in the test set (using the same partition as the original work).
- The training set of each CV iteration is further partitioned in a training subset and a validation subset, as follows: for each of the 48 users of the training set, 80% of their data is used for training, and 20% is used for validation, to decide which epoch's checkpoint will be selected for the test inference.
- We standardize each sensor feature (mean and standard deviation are calculated using the training subset). After standardization, missing feature values are zero-imputated.
- The model configuration includes a two-layer BiLSTM followed by a Linear layer. The `h_n` output of the BiLSTM, which contains a concatenation of the final forward and reverse hidden states of the two BiLSTM layers, is fed to the Linear layer of 51 nodes which correspond to the 51 context labels. We use Dropout after the first BiLSTM layer, with `dropout rate = 0.5` to help avoid overfitting.
- By using the `h_n` output of the BiLSTM, whose size is independent of the examples sequence length, as input for the output Linear layer, the resulting model's size is independent of the examples sequence length. Since all other hyperparameters are fixed (input features size, bidirectional, number of layers, output size), the model's size depends only on the BiLSTM layers' `hidden_size`.
- We experiment with two hyperparameters: the input sequence length, denoted as `window_len`, and the BiLSTM layers' `hidden_size`. Regarding `window_len`, we experiment using lengths of 5, 10, 15, and 30 examples (minutes). Regarding `hidden_size`, we experiment using sizes of 16, 32, and 64. The results of our investigation are thorougly presented in Table 5.11, along with the number of parameters for each model.

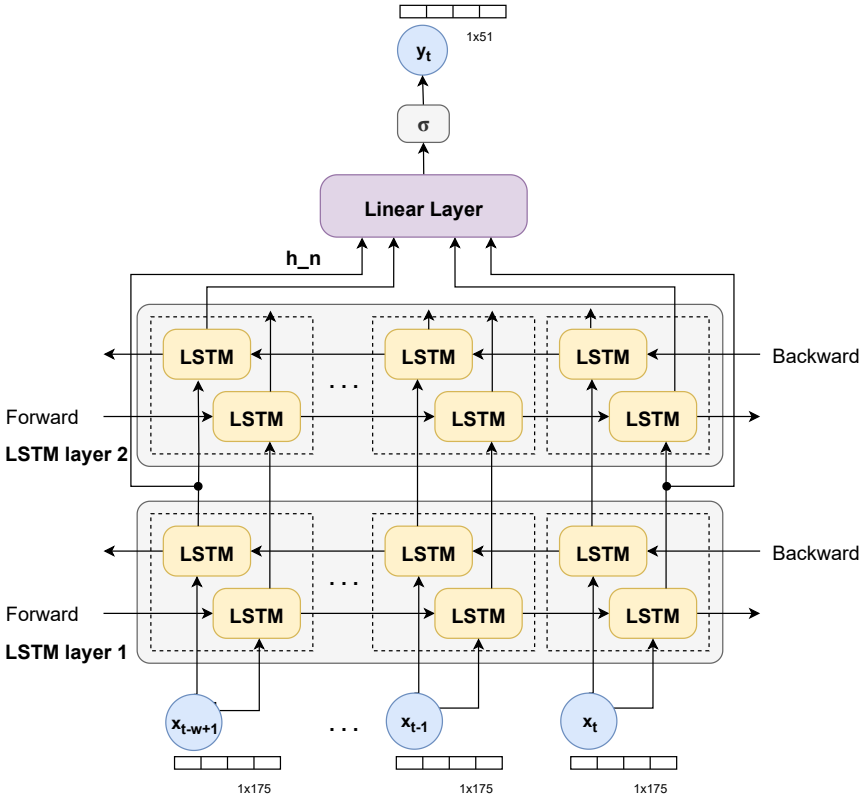


Figure 5.5.2: A simple two-layer BiLSTM model is presented conceptually. An examples sequence of length `w (window_len)` is given as input, and the target is to predict the context labels of the last example of the sequence. The `h_n` output of the BiLSTM, which contains a concatenation of the final forward and reverse hidden states of each BiLSTM layer, passes through a Linear layer, followed by a sigmoid function to convert the model's outputs to labels' probabilities.

- Regarding the loss, we again use the custom loss that was first described in Subsection 5.4.4, based on `torch.nn.BCEWithLogitsLoss`. This loss is slightly modified to allow dynamic per-batch, per-element masking in the loss matrices (to mask the loss elements corresponding to missing ground-truth labels for each batch, and not use them in the loss computation). Also, the `pos_weight` is used for instance-weighting to account for the imbalance in the number of positive examples per context label, by multiplying the term of the positive examples in the loss, with the ratio of negative to positive examples for this label in the training set.
- We use a `batch size` of 32 to train the model.
- We use the Adam optimizer to train the model. The `learning rate` hyperparameter is adapted according to the model size and to the input sequence length (`window_len`). The optimal values of `learning rate` for each combination of `window_len` and BiLSTM layers' `hidden_size`, as determined by our experiments, are denoted in Table 5.11.
- In the testing phase, the continuous predictions of the model pass through a sigmoid activation function, and then they are converted to binary outputs using a threshold of 0.5.
- We calculate the evaluation metrics for each label over its non-missing ground-truth examples.

We use `PyTorch` to build the model, the supervisor and the training scripts. The `torch.nn` modules and `torch` functions that are used, can be seen in detail in Figure 5.5.3.

In Table 5.11 we present the recognition scores for the BiLSTM model using the `h_n` output, averaged for all labels, for different values of the hyperparameters: input sequence length `window_len` = {5, 10, 15, 30} and BiLSTM layers' `hidden_size` = {16, 32, 64}. As it can be seen, for `window_len` = {5, 10, 15} we get

Performance metrics averaged over all labels for the BiLSTM model using only the final hidden states Hyperparameter tuning: BiLSTM input sequence length & BiLSTM layers' hidden size								
Hidden size	**No. of parameters**	**Learning rate**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Window length: 5								
16	34419	0.0001	0.798	0.233	0.756	0.798	0.303	0.777
32	85171	0.00005	0.800	0.236	0.760	0.800	0.308	0.780
64	235827	0.00002	0.813	0.243	0.753	0.814	0.316	0.784
Window length: 10								
16	34419	0.0001	0.797	0.232	0.748	0.799	0.302	0.773
32	85171	0.00005	0.805	0.237	0.755	0.805	0.310	0.780
64	235827	0.00002	0.814	0.241	0.743	0.815	0.314	0.779
Window length: 15								
16	34419	0.0001	0.790	0.230	0.767	0.790	0.300	0.779
32	85171	0.00005	0.813	0.240	0.732	0.814	0.311	0.773
64	235827	0.00002	0.810	0.240	0.746	0.810	0.313	0.778
Window length: 30								
16	34419	0.0001	0.799	0.231	0.723	0.801	0.299	0.762
32	85171	0.00005	0.817	0.239	0.702	0.819	0.310	0.761
64	235827	0.00002	0.814	0.240	0.733	0.814	0.313	0.774

Table 5.11: Recognition scores for the BiLSTM model using the `h_n` output, averaged for all labels, for different values of the hyperparameters: input sequence length (`window_len`) and BiLSTM layers' `hidden_size`. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

results consistently better than the results of the baseline MLP, while for for `window_len` = 30, the results are worse than or similar to the baseline MLP. This means that providing past examples in an interval of about 5-15 minutes helps the model better predict the context labels of the current minute, while providing past context of a longer interval spanning 30 minutes does not seem to help and even deteriorates total performance. We also observe that regarding `hidden_size`, in most cases, a larger hidden size improves the recognition performance, but comes with a cost of a lot more model parameters. We achieve the best BA of 0.784 with the combination of hyperparameters `window_len` = 5 and BiLSTM layers' `hidden_size` = 64. We also notice that when increasing `hidden_size`, we need to decrease the training's `learning rate` for best results, which reflect better model convergence.

For the best-performing model configuration, with `window_len` = 5 and BiLSTM layers' `hidden_size` = 64, we present the detailed model architecture in Figure 5.5.3, and the recognition scores for all context labels in Table 5.13 and averaged over the labels of each label subset in Table 5.12. Regarding the recognition scores for all context labels, we observe an improvement in BA for almost all context labels in comparison with the MLP baseline. Regarding the recognition scores averaged over the labels of each label subset, we observe an improvement in BA for almost all label subsets in comparison with the MLP baselines. Regarding the averaged recognition scores for all labels, there is an increase in all metrics compared with the MLP baseline, except for Sensitivity whose value is similar to that of the MLP. Overall we notice that this model has similar performance with the MLP regarding Sensitivity, which means that the number of positive examples for each label which are correctly recognized as positive is similar to that of the MLP, and a higher Specificity (0.814 compared with the MLP value of 0.786), which means that the number of negative examples for each label which are correctly recognized as negative is higher compared with that of the MLP, which means that we have on average less False Positives. Overall we achieve an average BA of 0.784, which improves the 0.772 BA value of the baseline MLP.

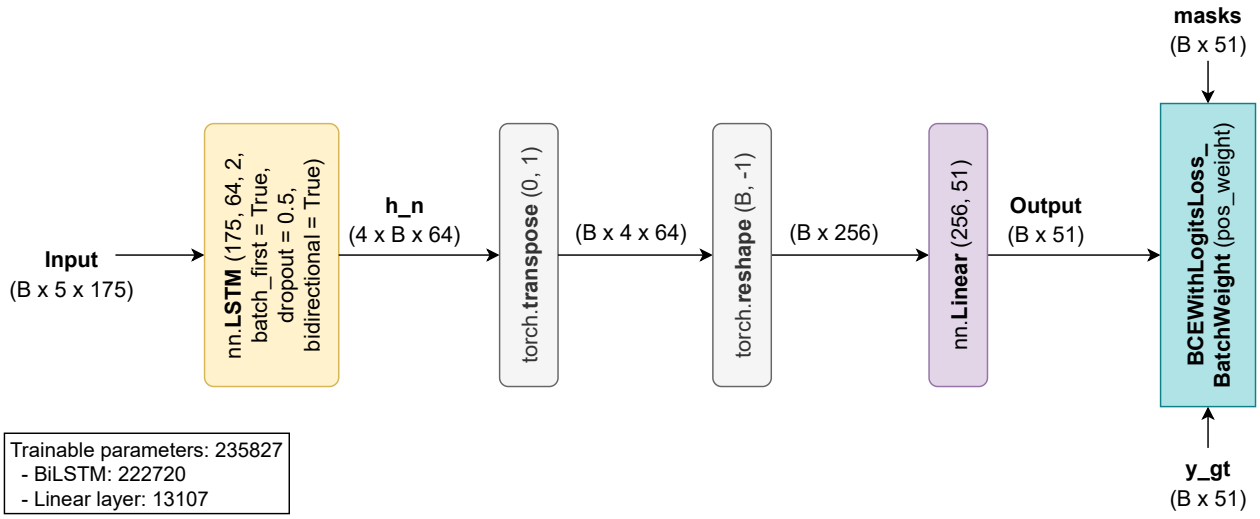


Figure 5.5.3: The best-performing BiLSTM model architecture using output `h_n`. A 5-length examples sequence (`window_len = 5`) is given as input to the two-layer BiLSTM with `hidden_size = 64` and `dropout = 0.5`. The `h_n` output of the BiLSTM (which contains a concatenation of the final forward and reverse hidden states of each BiLSTM layer), after being transposed and reshaped, passes through an output Linear layer of 51 nodes corresponding to the 51 context labels. A custom loss based on `torch.nn.BCEWithLogitsLoss` is used to account for missing labels information and label weights in loss computation.

Recognition scores of the BiLSTM model using only the final hidden states, averaged for each label subset

Label	Accuracy	Precision	Sensitivity	Specificity	F1-score	BA
Posture/Movement	0.829	0.402	0.796	0.829	0.482	0.813
Special Movement	0.806	0.028	0.734	0.806	0.053	0.770
Phone Location	0.774	0.452	0.769	0.769	0.531	0.769
Work-related	0.807	0.260	0.820	0.804	0.384	0.812
Location-based	0.818	0.271	0.804	0.817	0.343	0.810
Transportation	0.869	0.174	0.883	0.869	0.279	0.876
Chores	0.842	0.053	0.637	0.845	0.097	0.741
Self-care	0.864	0.202	0.698	0.869	0.246	0.783
Leisure Time	0.742	0.159	0.667	0.747	0.248	0.707
Companion	0.708	0.191	0.614	0.721	0.287	0.668
Environment	0.885	0.694	0.894	0.891	0.743	0.892
Average	0.813	0.243	0.753	0.814	0.316	0.784

Table 5.12: Recognition scores reported for the BiLSTM model using only the final hidden states (`h_n` output), averaged for each label subset. Model configuration with `window_len` = 5 and `hidden_size` = 64. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

Recognition scores of the BiLSTM model using only the final hidden states, for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.893	0.819	0.840	0.917	0.829	0.878
Sitting	79368	0.778	0.741	0.809	0.750	0.774	0.780
Standing	22071	0.708	0.267	0.707	0.708	0.387	0.707
Walking	11715	0.825	0.259	0.817	0.826	0.393	0.821
Running	661	0.871	0.040	0.731	0.872	0.076	0.801
Bicycling	3504	0.901	0.288	0.873	0.902	0.434	0.888
Strolling	339	0.741	0.045	0.829	0.739	0.085	0.784
Stairs - Going up	399	0.839	0.028	0.652	0.841	0.054	0.746
Stairs - Going down	390	0.823	0.025	0.659	0.824	0.048	0.742
Elevator	123	0.821	0.012	0.797	0.821	0.024	0.809
Phone in pocket	14074	0.786	0.448	0.825	0.777	0.581	0.801
Phone in hand	7313	0.698	0.187	0.716	0.697	0.297	0.706
Phone in bag	5031	0.803	0.264	0.717	0.811	0.386	0.764
Phone on table	65979	0.810	0.908	0.817	0.792	0.860	0.804
In class	2852	0.828	0.220	0.793	0.830	0.345	0.812
Lab work	2898	0.771	0.315	0.928	0.752	0.471	0.840
Computer work	22536	0.766	0.380	0.751	0.769	0.505	0.760
In a meeting	2837	0.863	0.123	0.807	0.864	0.213	0.835
At home	80044	0.794	0.772	0.807	0.781	0.789	0.794
At school	25342	0.762	0.414	0.756	0.763	0.535	0.760
At main workplace	19235	0.838	0.564	0.846	0.836	0.677	0.841
At a restaurant	1275	0.877	0.091	0.814	0.878	0.164	0.846
At a bar	520	0.856	0.143	0.894	0.855	0.247	0.875
At a party	404	0.808	0.062	0.762	0.808	0.114	0.785
At the gym	897	0.809	0.100	0.734	0.811	0.176	0.772
At the beach	116	0.801	0.020	0.819	0.801	0.040	0.810
In a car	3550	0.888	0.219	0.877	0.888	0.351	0.883
On a bus	1179	0.837	0.064	0.885	0.836	0.119	0.860
Drive - Driver	4879	0.892	0.310	0.889	0.892	0.460	0.891
Drive - Passenger	1650	0.859	0.104	0.882	0.858	0.186	0.870
Shopping	809	0.801	0.038	0.778	0.801	0.073	0.789
Cooking	2212	0.837	0.073	0.691	0.840	0.132	0.766
Cleaning	1813	0.828	0.075	0.640	0.832	0.134	0.736
Doing laundry	471	0.865	0.030	0.452	0.869	0.056	0.661
Washing dishes	829	0.880	0.050	0.626	0.882	0.093	0.754
Bathing - Shower	1120	0.865	0.050	0.699	0.867	0.093	0.783
Toilet	1558	0.818	0.042	0.624	0.821	0.079	0.722
Grooming	1775	0.844	0.064	0.612	0.848	0.117	0.730
Dressing	1248	0.884	0.063	0.692	0.886	0.115	0.789
Sleeping	40869	0.907	0.793	0.864	0.922	0.827	0.893
Exercise	5191	0.878	0.202	0.796	0.881	0.322	0.839
Eating	9668	0.690	0.122	0.669	0.691	0.206	0.680
Drinking alcohol	859	0.866	0.114	0.790	0.868	0.199	0.829
Watching TV	8945	0.802	0.249	0.648	0.817	0.360	0.732
Surfing the internet	10668	0.722	0.182	0.485	0.750	0.264	0.617
Talking	18477	0.634	0.209	0.715	0.623	0.323	0.669
Singing	384	0.601	0.033	0.562	0.602	0.063	0.582
With co-workers	3972	0.756	0.165	0.706	0.759	0.268	0.733
With friends	12686	0.660	0.216	0.523	0.683	0.305	0.603
Indoors	102510	0.891	0.993	0.890	0.904	0.939	0.897
Outside	6793	0.879	0.394	0.898	0.878	0.548	0.888
Average		0.813	0.243	0.753	0.814	0.316	0.784

Table 5.13: Recognition scores reported for the BiLSTM model using only the final hidden states (`h_n` output), for all context labels. Model configuration with `window_len` = 5 and `hidden_size` = 64. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

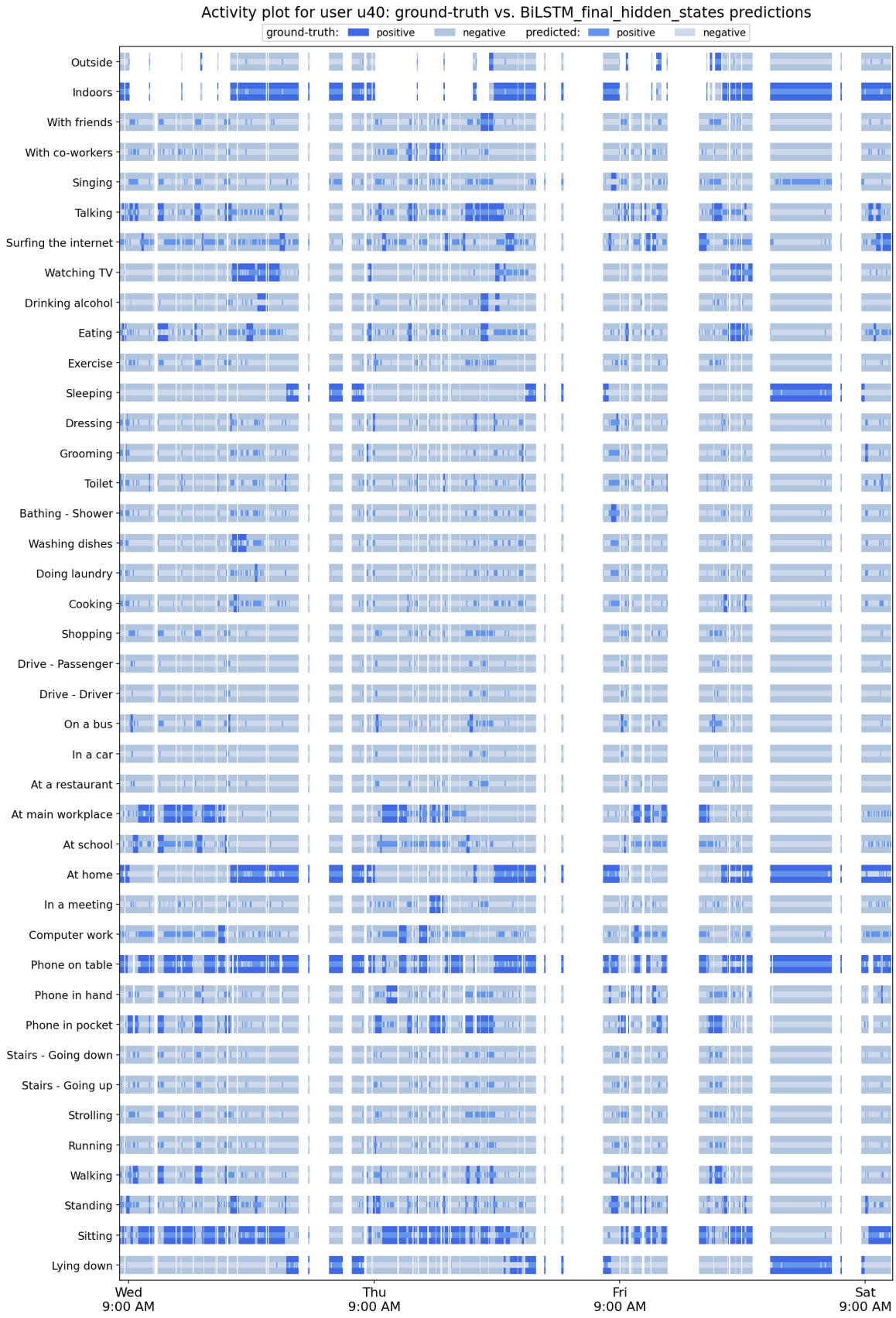


Figure 5.5.4: Activity plot including the predictions of the BiLSTM model using final hidden states for user u40

Figure 5.5.5: Activity plot including the predictions of the BiLSTM model using final hidden states for user u45

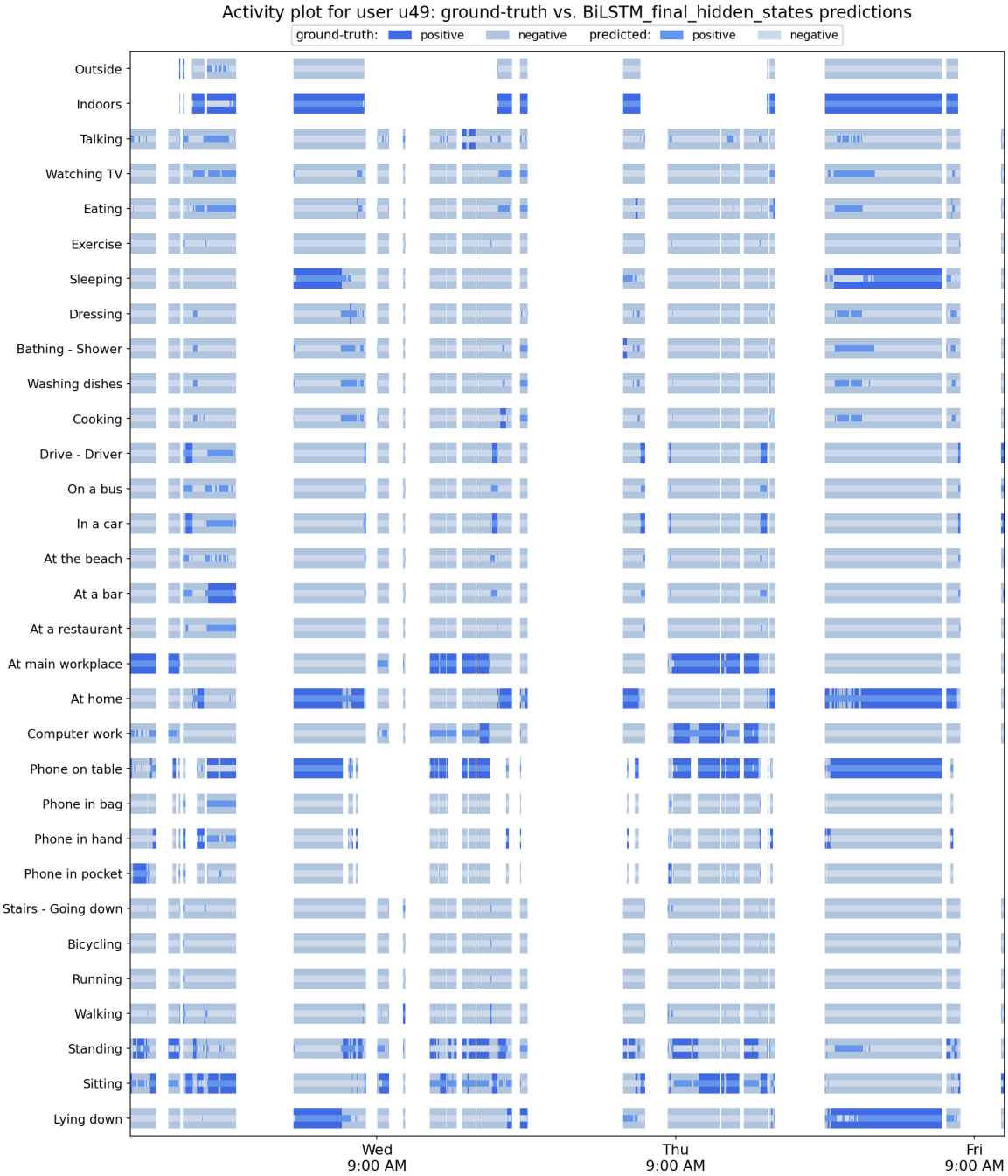


Figure 5.5.6: Activity plot including the predictions of the BiLSTM model using final hidden states for user u49

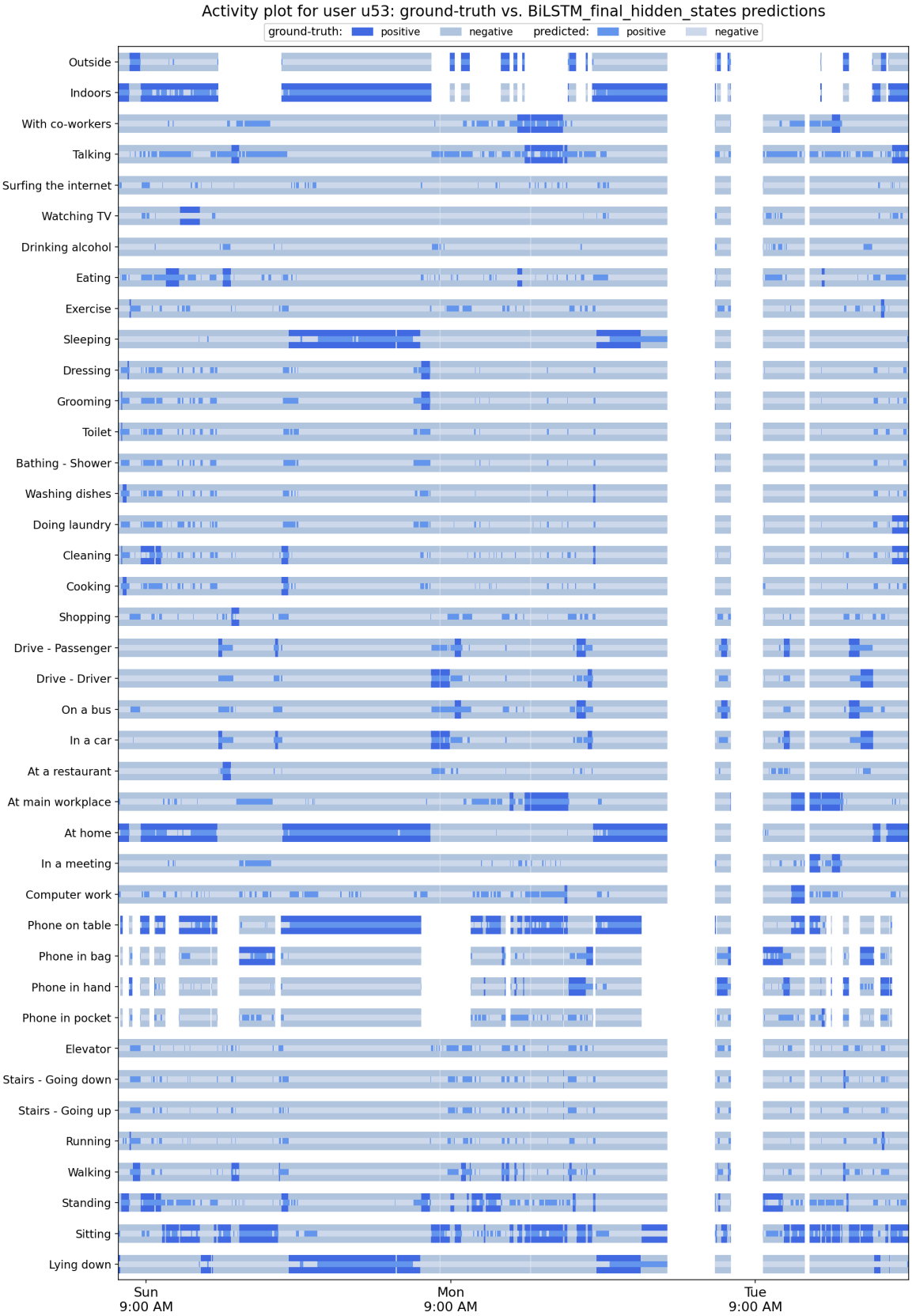


Figure 5.5.7: Activity plot including the predictions of the BiLSTM model using final hidden states for user u53

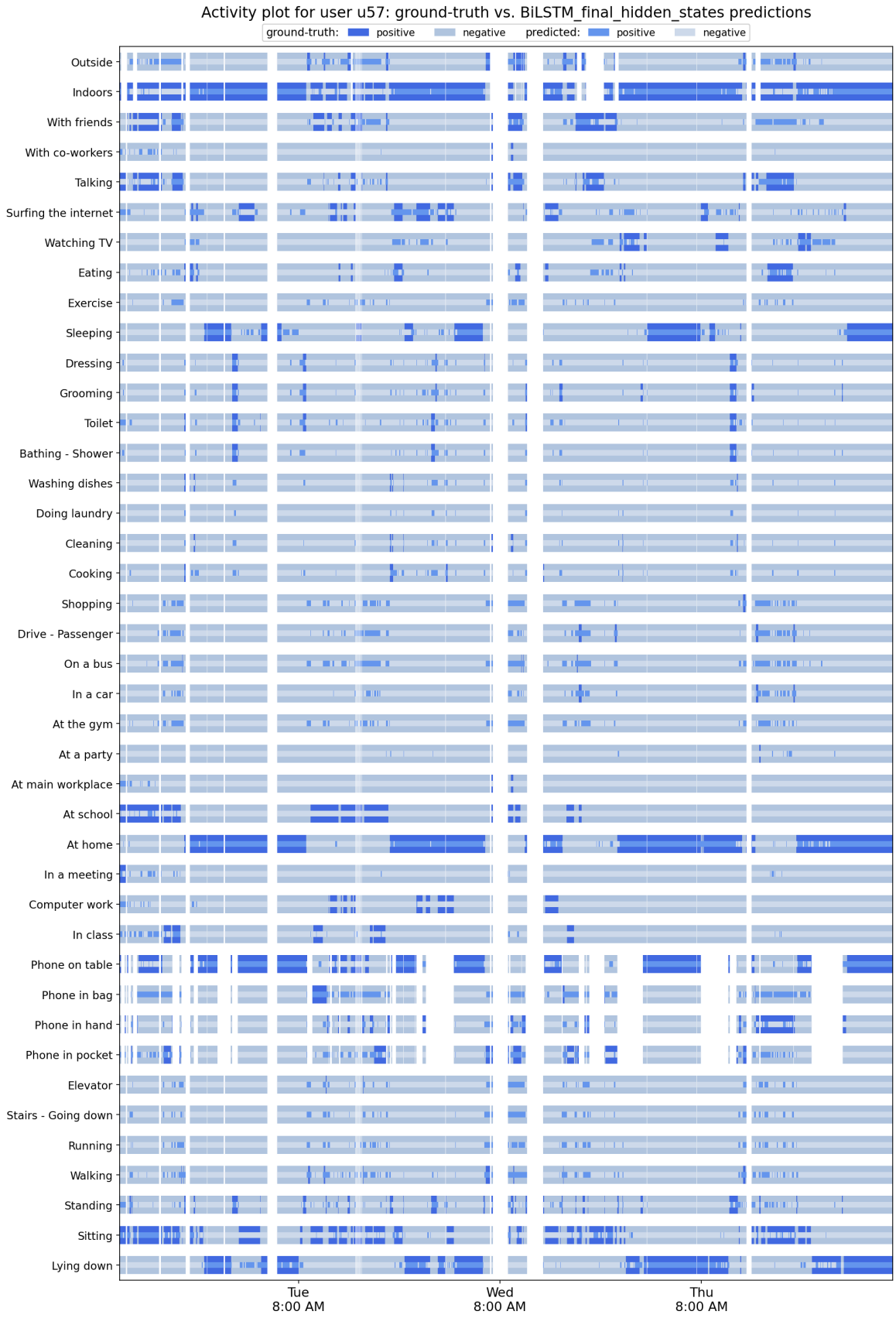


Figure 5.5.8: Activity plot including the predictions of the BiLSTM model using final hidden states for user u57

5.5.2 Using BiLSTM Output for all Timesteps

We also implement a model including a two-layer BiLSTM, whose full `output`, which contains the output features (`h_t`) from the last layer of the BiLSTM for all timesteps, is then fed to a Linear layer, to predict the probabilities for each of the 51 context labels. This model is conceptually presented in Figure 5.5.9.

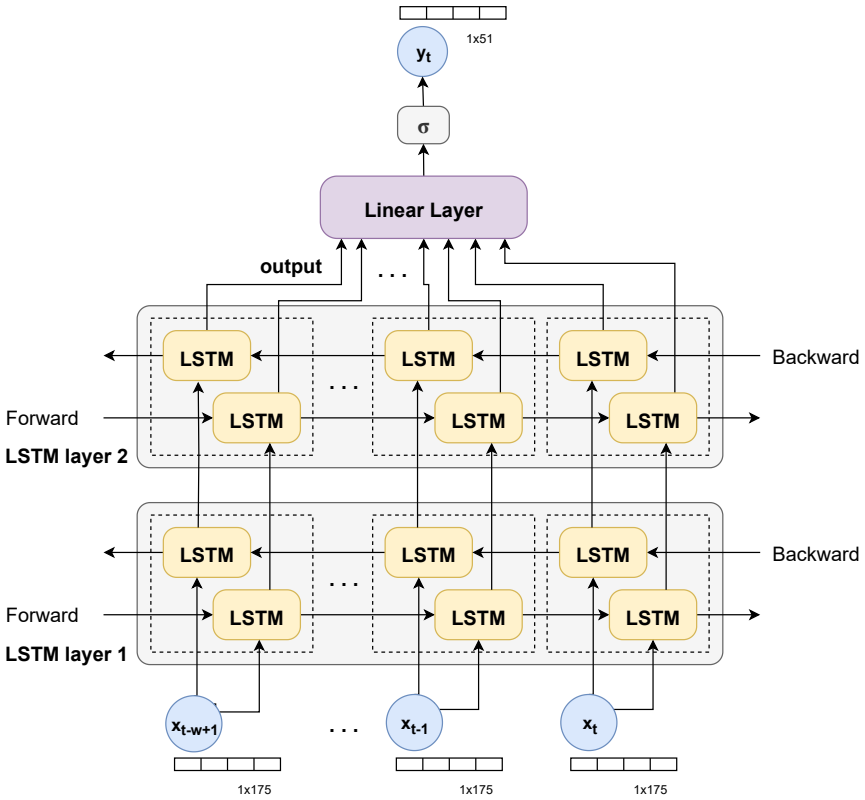


Figure 5.5.9: A simple two-layer BiLSTM model is presented conceptually. An examples sequence of length `w (window_len)` is given as input, and the target is to predict the context labels of the last example of the sequence. The `output` of the BiLSTM, which contains the output features (`h_t`) from the last layer of the BiLSTM for each timestep, passes through a Linear layer, followed by a sigmoid function to convert the model's outputs to labels' probabilities.

The key points of the experiments are summarized below:

- Five-fold cross validation is used, with each fold containing 48 users in the training set and 12 users in the test set (using the same partition as the original work).
- The training set of each CV iteration is further partitioned in a training subset and a validation subset, as follows: for each of the 48 users of the training set, 80% of their data is used for training, and 20% is used for validation, to decide which epoch's checkpoint will be selected for the test inference.
- We standardize each sensor feature (mean and standard deviation are calculated using the training subset). After standardization, missing feature values are zero-imputated.
- The model configuration includes a two-layer BiLSTM followed by a Linear layer. The `output` of the BiLSTM, which contains the output features (`h_t`) from the last layer of the BiLSTM for each timestep, is fed to the Linear layer of 51 nodes which correspond to the 51 context labels. We use Dropout after the first BiLSTM layer, with `dropout rate = 0.5` to help avoid overfitting.
- Since the `output` size depends on the examples sequence length, the resulting model's size grows as the length of the input sequence grows, since the Linear layer's size is proportional to the size of its input. Since all other hyperparameters are fixed (input features size, bidirectional, number of layers, output size), the model's size depends on both the examples sequence length `window_len` and the BiLSTM

layers' `hidden_size`.

- We experiment with these two hyperparameters: the input sequence length, denoted as `window_len`, and the BiLSTM layers' `hidden_size`. Regarding `window_len`, we experiment using lengths of 5, 10, 15, and 30 examples (minutes). Regarding `hidden_size`, we experiment using sizes of 16, 32, and 64. The results of our investigation are thorougly presented in Table 5.14, along with the number of parameters for each model.
- Regarding the loss, we again use the custom loss that was first described in Subsection 5.4.4, based on `torch.nn.BCEWithLogitsLoss`. This loss is slightly modified to allow dynamic per-batch, per-element masking in the loss matrices (to mask the loss elements corresponding to missing ground-truth labels for each batch, and not use them in the loss computation). Also, the `pos_weight` is used for instance-weighting to account for the imbalance in the number of positive examples per context label, by multiplying the term of the positive examples in the loss, with the ratio of negative to positive examples for this label in the training set.
- We use a `batch size` of 32 to train the model.
- We use the Adam optimizer to train the model. The `learning rate` hyperparameter is adapted according to the model size and to the input sequence length (`window_len`). The optimal values of `learning rate` for each combination of `window_len` and BiLSTM layers' `hidden_size`, as determined by our experiments, are denoted in Table 5.14.
- In the testing phase, the continuous predictions of the model pass through a sigmoid activation function, and then they are converted to binary outputs using a threshold of 0.5.
- We calculate the evaluation metrics for each label over its non-missing ground-truth examples.

We use `PyTorch` to build the model, the supervisor and the training scripts. The `torch.nn` modules and `torch` functions that are used, can be seen in detail in Figure 5.5.10.

Performance metrics averaged over all labels for the BiLSTM model using all timesteps' outputs
Hyperparameter tuning: BiLSTM input sequence length & BiLSTM layers' hidden size

Hidden size	No. of parameters	Learning rate	Accuracy	Precision	Sensitivity	Specificity	F1-score	BA
Window length: 5								
16	39315	0.00005	0.791	0.232	0.764	0.791	0.302	0.778
32	94963	0.00001	0.795	0.234	0.764	0.795	0.305	0.779
64	255411	0.00001	0.810	0.241	0.761	0.811	0.314	0.786
Window length: 10								
16	47475	0.00001	0.775	0.222	0.761	0.775	0.288	0.768
32	111283	0.00001	0.789	0.229	0.751	0.789	0.298	0.770
64	288051	0.00001	0.804	0.241	0.759	0.805	0.312	0.782
Window length: 15								
16	55635	0.00001	0.782	0.223	0.742	0.782	0.290	0.762
32	127603	0.00001	0.807	0.240	0.754	0.808	0.311	0.781
64	320691	0.000005	0.814	0.242	0.746	0.814	0.315	0.780
Window length: 30								
16	80115	0.00001	0.783	0.221	0.724	0.782	0.288	0.753
32	176563	0.000005	0.778	0.223	0.722	0.780	0.288	0.751
64	418611	0.000002	0.802	0.233	0.725	0.803	0.302	0.764

Table 5.14: Recognition scores for the BiLSTM model using all outputs, averaged for all labels, for different values of the hyperparameters: input sequence length (`window_len`) and BiLSTM layers' `hidden_size`. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

In Table 5.14 we present the recognition scores for the BiLSTM model using all timesteps' `output`, averaged for all labels, for different values of the hyperparameters: input sequence length `window_len` = {5, 10, 15, 30} and BiLSTM layers' `hidden_size` = {16, 32, 64}. As it can be seen, for `window_len` = 5, we get results

consistently better than the results of the baseline MLP, for `window_len` = {10, 15} we get results better that those of the MLP only for larger `hidden_size` values, while for `window_len` = 30, the results are worse than the baseline MLP. This means that providing past examples in an interval of 5 minutes is helpful for this model configuration, independently of `hidden_size` among the tested values. Providing past context of a longer interval of about 10-15 minutes is helpful only when the `hidden_size` of the BiLSTM is large enough to capture adequate past information, while providing past context of a larger interval spanning 30 minutes reduces total performance. We also observe that regarding `hidden_size`, again, in most cases, a larger hidden size improves the recognition performance, but comes with a cost of a lot more model parameters. We achieve the best BA of 0.786 with the combination of hyperparameters `window_len` = 5 and BiLSTM layers' `hidden_size` = 64. We also notice again that when increasing `hidden_size`, we need to decrease the training's `learning rate` for better model convergence.

For the best-performing model configuration, with `window_len` = 5 and BiLSTM layers' `hidden_size` = 64, we present the detailed model architecture in Figure 5.5.10, and the recognition scores for all context labels in Table 5.16 and averaged over the labels of each label subset in Table 5.15. In general, in the recognition scores we observe trends similar to those of the previous experiment using only final hidden states. More specifically, regarding the recognition scores for all context labels, we observe an improvement in BA for almost all context labels in comparison with the MLP baseline. Regarding the recognition scores averaged over the labels of each label subset, we observe an improvement in BA for almost all label subsets in comparison with the MLP baselines. Regarding the averaged recognition scores for all labels, there is an increase in all metrics compared with the MLP baseline. In detail, we notice that this model has similar performance regarding Sensitivity and improved performance regarding Specificity (0.811 compared to the MLP value of 0.786), which again means that we have a decrease on False Positives. Finally, overall we achieve an average BA of 0.786, which is an improvement over the 0.772 BA value of the baseline MLP.

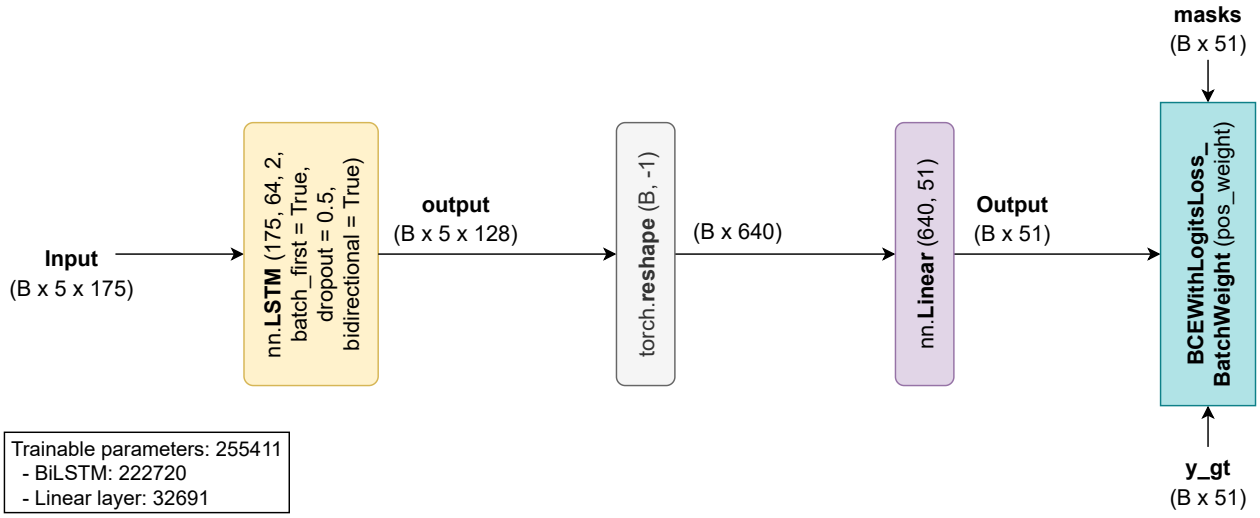


Figure 5.5.10: The best-performing BiLSTM model architecture using all timesteps' outputs. A 5-length examples sequence (`window_len = 5`) is given as input to the two-layer BiLSTM with `hidden_size = 64` and `dropout = 0.5`. The `output` of the BiLSTM (which contains the output features `h_t` from the last layer of the BiLSTM for each timestep), after being reshaped, passes through an output Linear layer of 51 nodes corresponding to the 51 context labels. A custom loss based on `torch.nn.BCEWithLogitsLoss` is used to account for missing labels information and label weights in loss computation.

Recognition scores of the BiLSTM model using all timesteps' outputs averaged for each label subset						
Label	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Posture/Movement	0.825	0.397	0.792	0.823	0.476	0.808
Special Movement	0.798	0.028	0.744	0.799	0.053	0.771
Phone Location	0.770	0.453	0.783	0.766	0.533	0.774
Work-related	0.802	0.253	0.797	0.800	0.372	0.799
Location-based	0.814	0.267	0.804	0.812	0.338	0.808
Transportation	0.866	0.169	0.877	0.866	0.271	0.871
Chores	0.826	0.053	0.699	0.828	0.097	0.764
Self-care	0.856	0.201	0.700	0.861	0.243	0.780
Leisure Time	0.758	0.160	0.688	0.761	0.253	0.724
Companion	0.706	0.191	0.625	0.720	0.287	0.672
Environment	0.884	0.694	0.894	0.892	0.744	0.893
Average	0.810	0.241	0.761	0.811	0.314	0.786

Table 5.15: Recognition scores reported for the BiLSTM model using all timesteps' outputs, averaged for each label subset. Model configuration with `window_len` = 5 and `hidden_size` = 64. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

Recognition scores of the BiLSTM model using all timesteps' outputs for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.895	0.825	0.838	0.921	0.832	0.879
Sitting	79368	0.780	0.738	0.824	0.741	0.778	0.782
Standing	22071	0.697	0.257	0.700	0.696	0.376	0.698
Walking	11715	0.814	0.247	0.824	0.813	0.380	0.818
Running	661	0.872	0.039	0.696	0.874	0.074	0.785
Bicycling	3504	0.894	0.273	0.872	0.895	0.416	0.883
Strolling	339	0.772	0.051	0.823	0.772	0.096	0.797
Stairs - Going up	399	0.828	0.026	0.657	0.829	0.051	0.743
Stairs - Going down	390	0.815	0.023	0.644	0.816	0.045	0.730
Elevator	123	0.778	0.011	0.854	0.777	0.021	0.815
Phone in pocket	14074	0.782	0.444	0.839	0.769	0.581	0.804
Phone in hand	7313	0.684	0.183	0.736	0.679	0.293	0.707
Phone in bag	5031	0.807	0.274	0.746	0.813	0.401	0.779
Phone on table	65979	0.808	0.912	0.811	0.802	0.858	0.807
In class	2852	0.824	0.204	0.718	0.830	0.317	0.774
Lab work	2898	0.781	0.326	0.932	0.762	0.483	0.847
Computer work	22536	0.762	0.375	0.742	0.766	0.498	0.754
In a meeting	2837	0.842	0.107	0.797	0.843	0.189	0.820
At home	80044	0.788	0.768	0.800	0.777	0.784	0.789
At school	25342	0.759	0.407	0.728	0.766	0.522	0.747
At main workplace	19235	0.835	0.558	0.861	0.828	0.677	0.845
At a restaurant	1275	0.864	0.086	0.849	0.865	0.156	0.857
At a bar	520	0.831	0.122	0.875	0.830	0.214	0.852
At a party	404	0.832	0.075	0.829	0.832	0.138	0.831
At the gym	897	0.817	0.100	0.701	0.820	0.175	0.761
At the beach	116	0.782	0.018	0.784	0.782	0.035	0.783
In a car	3550	0.884	0.212	0.870	0.884	0.341	0.877
On a bus	1179	0.832	0.063	0.894	0.831	0.117	0.862
Drive - Driver	4879	0.887	0.295	0.864	0.888	0.440	0.876
Drive - Passenger	1650	0.860	0.105	0.879	0.860	0.187	0.869
Shopping	809	0.789	0.037	0.802	0.789	0.071	0.796
Cooking	2212	0.837	0.073	0.688	0.840	0.131	0.764
Cleaning	1813	0.808	0.070	0.672	0.811	0.127	0.742
Doing laundry	471	0.825	0.032	0.641	0.826	0.060	0.734
Washing dishes	829	0.872	0.051	0.692	0.873	0.095	0.783
Bathing - Shower	1120	0.849	0.042	0.649	0.851	0.078	0.750
Toilet	1558	0.807	0.041	0.647	0.809	0.077	0.728
Grooming	1775	0.839	0.063	0.618	0.843	0.114	0.730
Dressing	1248	0.875	0.060	0.712	0.877	0.111	0.794
Sleeping	40869	0.912	0.800	0.874	0.925	0.835	0.899
Exercise	5191	0.874	0.195	0.786	0.878	0.313	0.832
Eating	9668	0.667	0.115	0.681	0.666	0.197	0.674
Drinking alcohol	859	0.854	0.108	0.811	0.855	0.190	0.833
Watching TV	8945	0.801	0.248	0.648	0.815	0.359	0.732
Surfing the internet	10668	0.703	0.182	0.540	0.722	0.272	0.631
Talking	18477	0.632	0.212	0.739	0.617	0.329	0.678
Singing	384	0.772	0.062	0.607	0.776	0.113	0.691
With co-workers	3972	0.743	0.163	0.738	0.743	0.267	0.741
With friends	12686	0.670	0.220	0.512	0.697	0.307	0.604
Indoors	102510	0.888	0.993	0.887	0.906	0.937	0.896
Outside	6793	0.880	0.396	0.902	0.878	0.550	0.890
Average		0.810	0.241	0.761	0.811	0.314	0.786

Table 5.16: Recognition scores reported for the BiLSTM model using all timesteps' outputs, for all context labels. Model configuration with `window_len` = 5 and `hidden_size` = 64. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

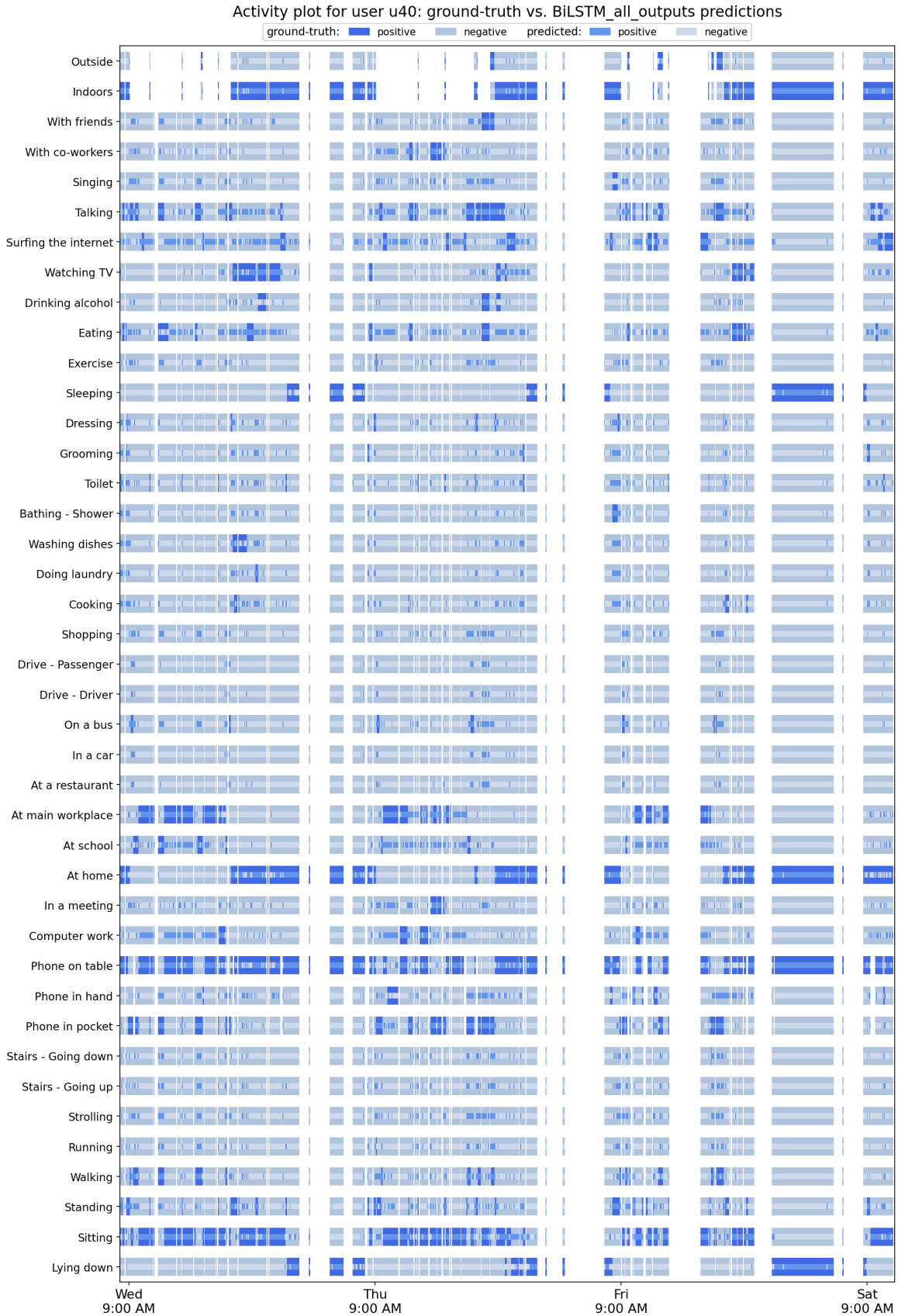


Figure 5.5.11: Activity plot including the predictions of the BiLSTM model using all outputs for user u40

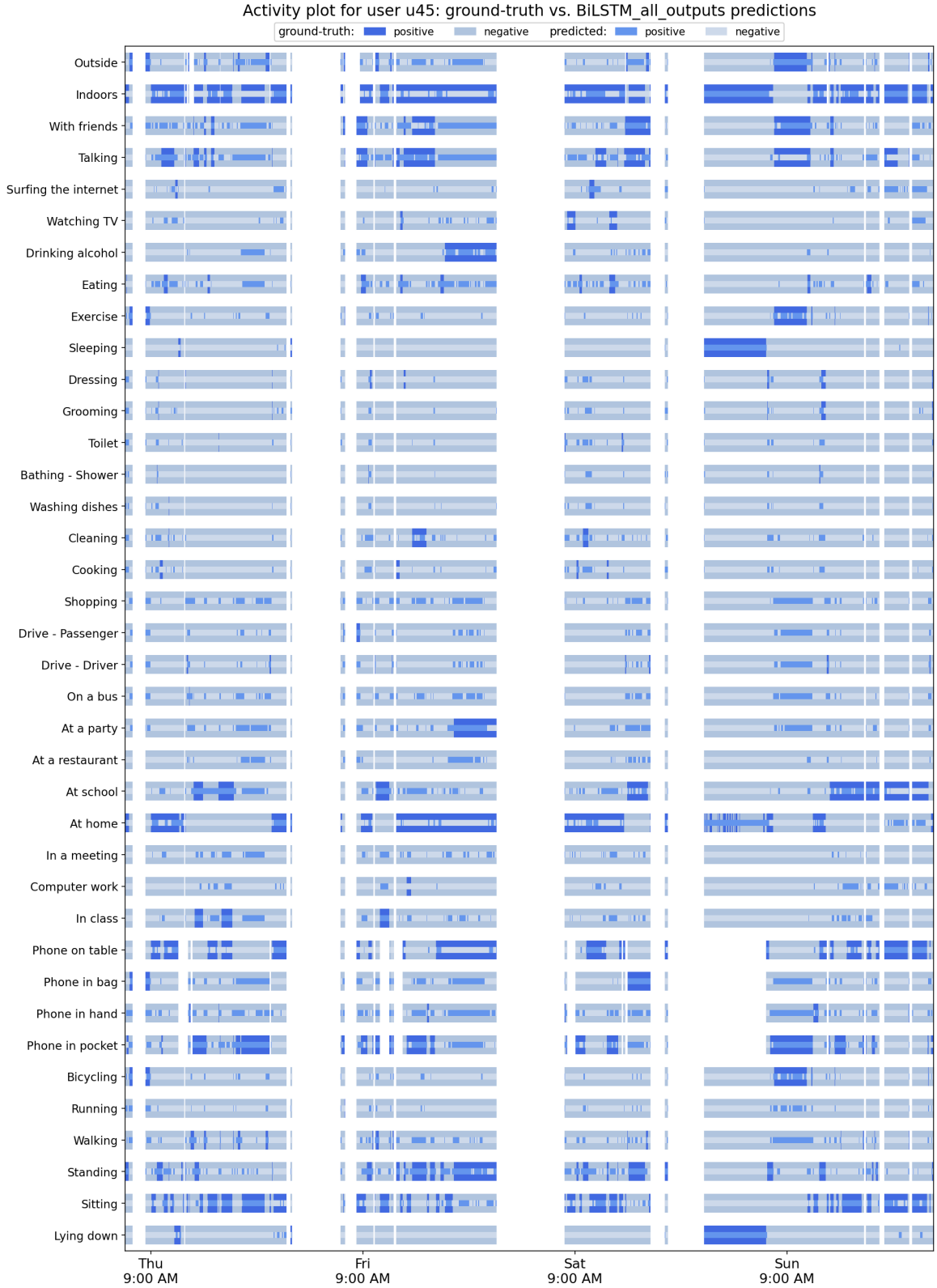


Figure 5.5.12: Activity plot including the predictions of the BiLSTM model using all outputs for user u45

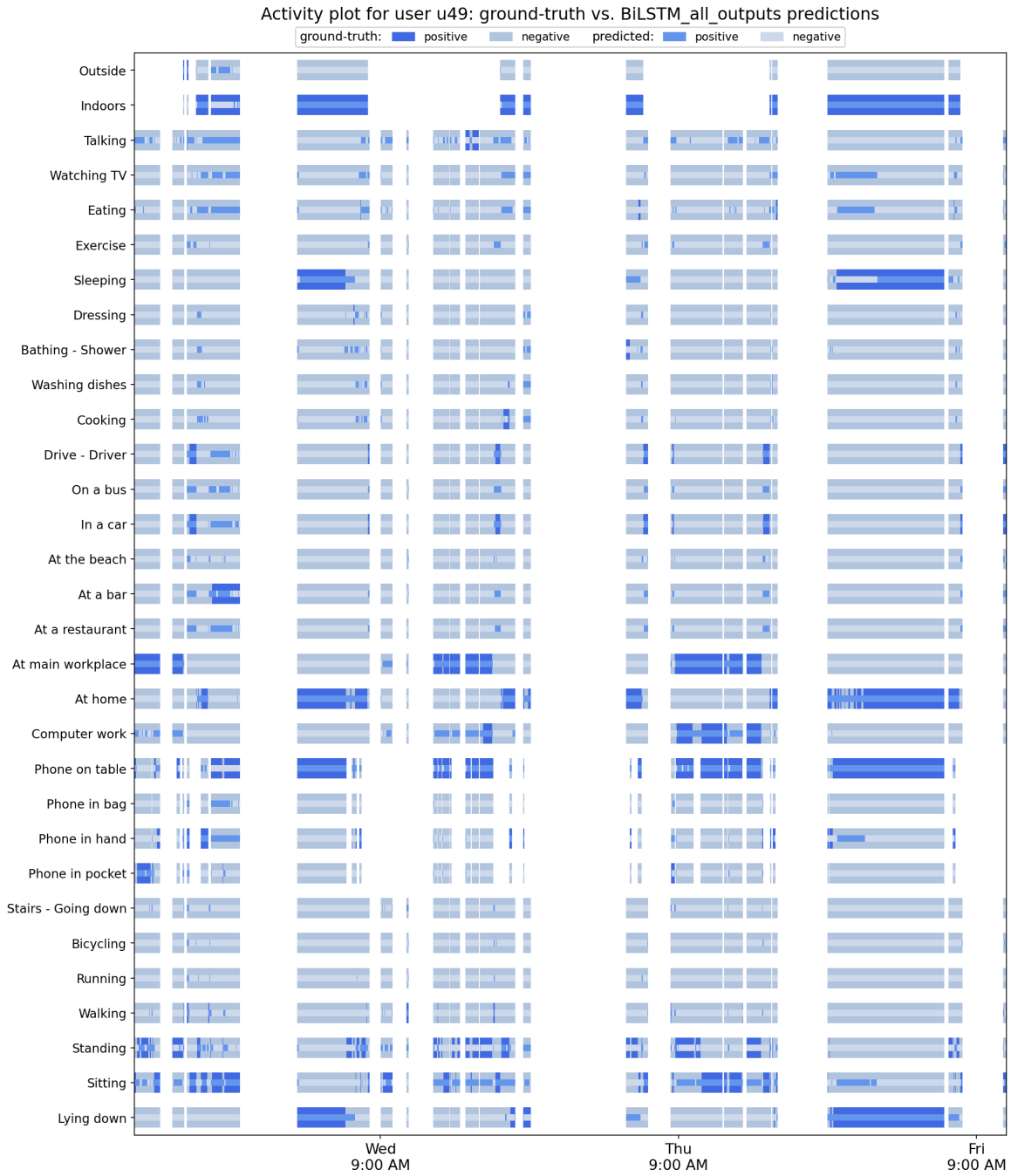


Figure 5.5.13: Activity plot including the predictions of the BiLSTM model using all outputs for user u49

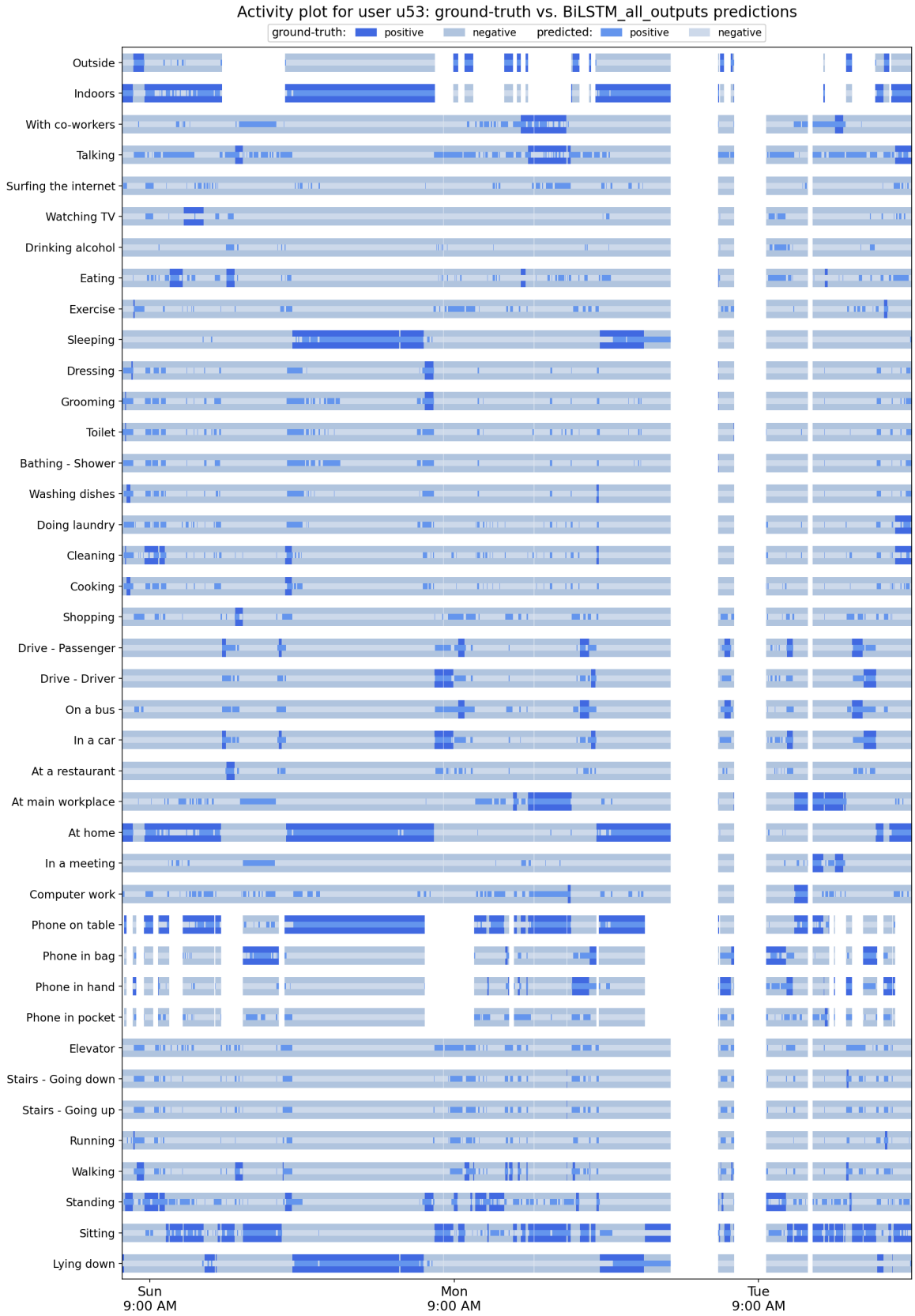


Figure 5.5.14: Activity plot including the predictions of the BiLSTM model using all outputs for user u53

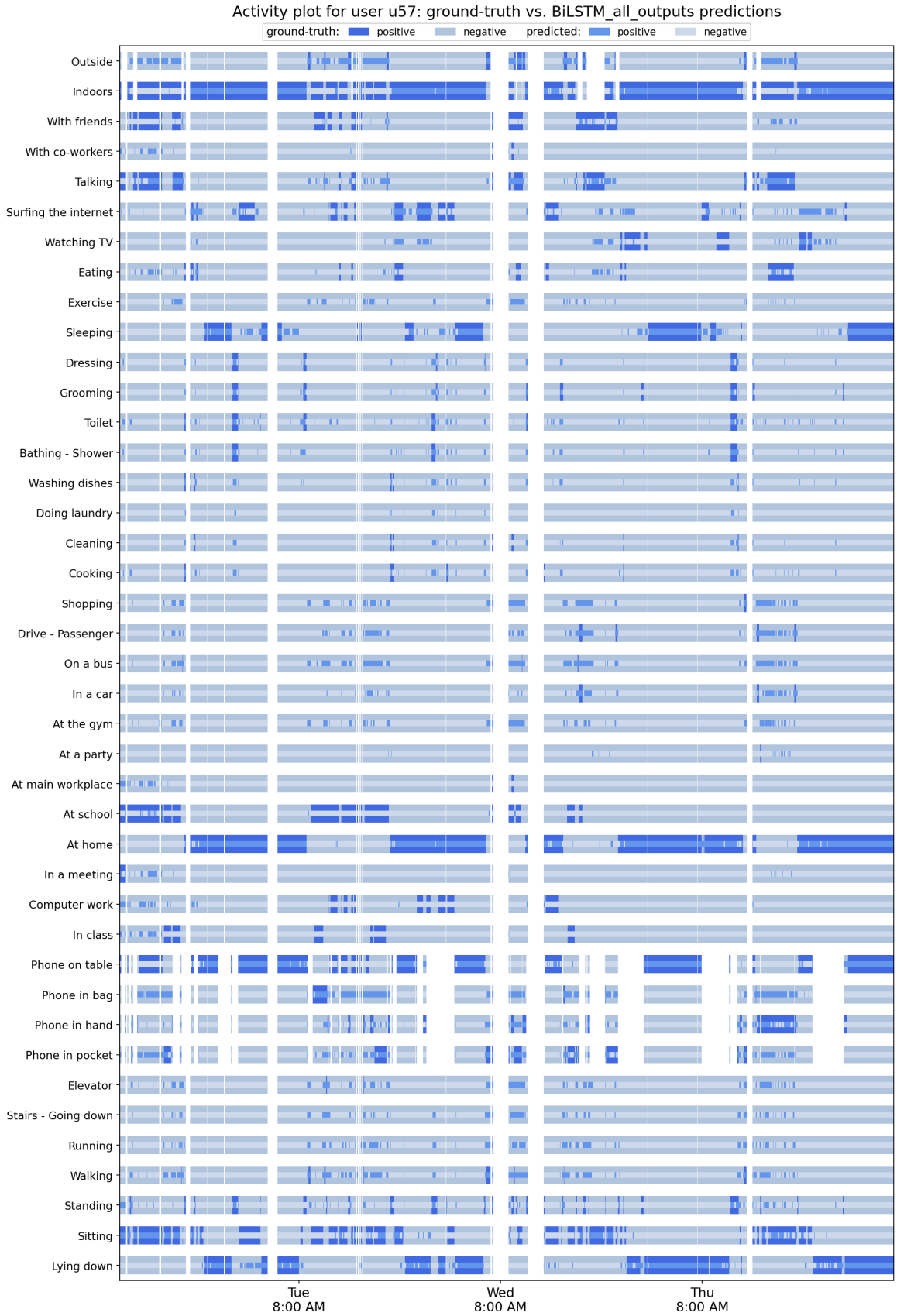


Figure 5.5.15: Activity plot including the predictions of the BiLSTM model using all outputs for user u57

5.6 Bidirectional LSTM with Attention

In this Section, we investigate if the performance of the BiLSTM model for our task can be improved with the addition of a self-attention or cross-attention mechanism. We augment the aforementioned BiLSTM-based architectures of Section 5.5 in two different ways. In the first experiment, we add a self-attention module before the two-layer BiLSTM whose final hidden states output `h_n` is used in the output Linear layer. In the second experiment, we add a cross-attention module after the two-layer BiLSTM whose output for all timesteps is used as key in the cross-attention module while the input features are used as query and value.

In these experiments we again use the aforementioned "Core" subset of the dataset, exactly as described and processed in Section 5.5. We follow the same standardization and missing values imputation techniques, and the same cross-validation scheme. We also follow the same logic as previously described, for the creation of `N`-length sequences of consecutive examples to be used as input for our models.

5.6.1 Self-Attention preceding BiLSTM using Final Hidden States

In this experiment, we expand the model of Subsection 5.5.1 by adding a self-attention module before the BiLSTM. This way, the input sequence passes through the self-attention before being fed to the two-layer BiLSTM, whose `h_n` output (final forward and reverse hidden states of the two BiLSTM layers), passes through a Linear layer to predict the probabilities for each of the 51 context labels. This model is conceptually presented in Figure 5.6.1, and the key points of the experiments are summarized as follows:

- Five-fold cross validation is used, with each fold containing 48 users in the training set and 12 users in the test set (using the same partition as the original work).
- The training set of each CV iteration is further partitioned in a training subset and a validation subset, as follows: for each of the 48 users of the training set, 80% of their data is used for training, and 20% is used for validation, to decide which epoch's checkpoint will be selected for the test inference.
- We standardize each sensor feature (mean and standard deviation are calculated using the training subset). After standardization, missing feature values are zero-imputated.
- The model configuration includes a Multi-head Self-Attention module, and a two-layer BiLSTM followed by a Linear layer. The `N`-length examples sequence is given as input to the Multi-head Self-Attention and its output is fed to the BiLSTM. The `h_n` output of the BiLSTM, which contains a concatenation of the final forward and reverse hidden states of the two BiLSTM layers, is fed to the Linear layer of 51 nodes which correspond to the 51 context labels. We use Dropout after the first BiLSTM layer, with `dropout rate = 0.5` to help avoid overfitting.
- Regarding the Self-Attention, we use Multi-head Attention, with Scaled Dot-Product Attention in each head. The `N`-length examples sequence is fed in all three: queries `Q`, keys `K` and values `V` of the Self-Attention. We use Multi-head Attention, with `num_heads = 5`, since single-head Attention did not give as good results, and as number of heads we use the smallest integer greater than 1, which is a divisor of the input features size (175). Also, after experimenting, we end up using `dropout rate = 0.2` in the Self-Attention, which was found to give the best results.
- We experiment with two hyperparameters of our model: the input sequence length, denoted as `window_len`, and the BiLSTM layers' `hidden_size`. Regarding `window_len`, we experiment using lengths of 5, 10, 15, and 30 examples (minutes). Regarding `hidden_size`, we experiment using sizes of 16, 32, and 64. The results of our investigation are thorougly presented in Table 5.17, along with the number of parameters for each model.
- Regarding the loss, we again use the custom loss that was first described in Subsection 5.4.4, based on `torch.nn.BCEWithLogitsLoss`. This loss is slightly modified to allow dynamic per-batch, per-element masking in the loss matrices (to mask the loss elements corresponding to missing ground-truth labels for each batch, and not use them in the loss computation). Also, the `pos_weight` is used for instance-weighting to account for the imbalance in the number of positive examples per context label, by multiplying the term of the positive examples in the loss, with the ratio of negative to positive examples for this label in the training set.

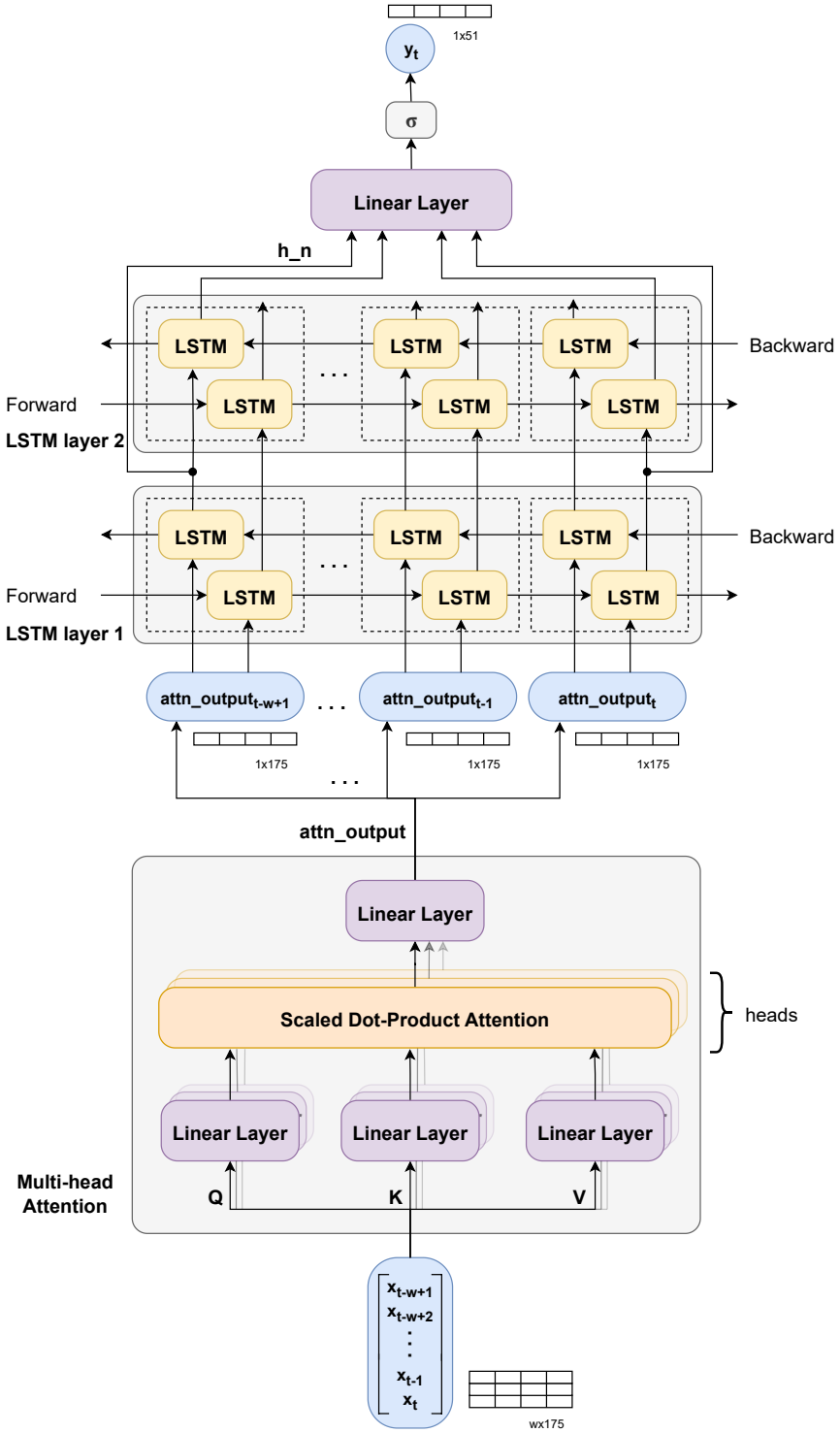


Figure 5.6.1: A two-layer BiLSTM model preceded by a Multi-head Self-Attention module is presented conceptually. An examples sequence of length `w (window_len)` is given as input, and the target is to predict the context labels of the last example of the sequence. The input is first fed to the Self-Attention module and its output passes to the BiLSTM. The `h_n` output of the BiLSTM, which contains a concatenation of the final forward and reverse hidden states of each BiLSTM layer, passes through a Linear layer, followed by a sigmoid function to convert the model's outputs to labels' probabilities.

- We use a `batch size` of 32 to train the model.
- We use the Adam optimizer to train the model. The `learning rate` hyperparameter is adapted according to the model size and to the input sequence length (`window_len`). The optimal values of `learning rate` as determined by our experiments, are denoted in Table 5.17.
- In the testing phase, the continuous predictions of the model pass through a sigmoid activation function, and then they are converted to binary outputs using a threshold of 0.5.
- We calculate the evaluation metrics for each label over its non-missing ground-truth examples.

We use `PyTorch` to build the model, the supervisor and the training scripts. The `torch.nn` modules and `torch` functions that are used, can be seen in detail in Figure 5.6.2.

Performance metrics averaged over all labels
for the Multi-head Self-Attention & BiLSTM model using only the final hidden states
Hyperparameter tuning: model input sequence length & BiLSTM layers' hidden size

Hidden size	No. of parameters	Learning rate	Accuracy	Precision	Sensitivity	Specificity	F1-score	BA
Window length: 5								
16	157619	0.00005	0.805	0.237	0.757	0.805	0.309	0.781
32	208371	0.00002	0.810	0.242	0.760	0.811	0.315	0.786
64	359027	0.00005	0.818	0.248	0.756	0.819	0.323	0.788
Window length: 10								
16	157619	0.0001	0.798	0.234	0.745	0.799	0.305	0.772
32	208371	0.00005	0.822	0.247	0.734	0.823	0.321	0.779
64	359027	0.00002	0.816	0.246	0.751	0.816	0.320	0.783
Window length: 15								
16	157619	0.0001	0.792	0.230	0.754	0.791	0.300	0.773
32	208371	0.0001	0.810	0.239	0.732	0.812	0.311	0.772
64	359027	0.00002	0.812	0.241	0.738	0.813	0.313	0.776
Window length: 30								
16	157619	0.0001	0.790	0.224	0.729	0.789	0.293	0.759
32	208371	0.0001	0.803	0.233	0.732	0.804	0.303	0.768
64	359027	0.00005	0.810	0.235	0.723	0.809	0.306	0.766

Table 5.17: Recognition scores for the Multi-head Self-Attention & BiLSTM model using the `h_n` output, averaged for all labels, for different values of the hyperparameters: input sequence length (`window_len`) and BiLSTM layers' `hidden_size`. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

In Table 5.17 we present the recognition scores for the Multi-head Self-Attention & BiLSTM model using the `h_n` output, averaged for all labels, for different values of the hyperparameters: input sequence length `window_len` = {5, 10, 15, 30} and BiLSTM layers' `hidden_size` = {16, 32, 64}. Compared to the results of the plain BiLSTM model using the `h_n` output, without the Self-Attention module (Table 5.11), we observe that for `window_len` = 5 we get results consistently better, while for `window_len` = {10, 15} the results are similar, and for `window_len` = 30, the results are similar or worse. A possible interpretation for this observation is that the Self-Attention module preceding the BiLSTM, is useful to compute more meaningful representations of the input sequences only in cases of shorter input sequences, e.g., spanning 5 minutes, which are more relevant to the current timestep labels, compared to longer input sequences. In total, we achieve the best BA of 0.788 with the combination of hyperparameters `window_len` = 5 and BiLSTM layers' `hidden_size` = 64, improved over the respective best BA of 0.784 of the plain BiLSTM model using only final hidden states.

For the best-performing model configuration, with `window_len` = 5 and BiLSTM layers' `hidden_size` = 64, we present the detailed model architecture in Figure 5.6.2, and the recognition scores for all context labels in Table 5.19 and averaged over the labels of each label subset in Table 5.18. Compared to the plain BiLSTM model using the `h_n` output, we can see that the Multi-head Self-Attention & BiLSTM model using the `h_n` output offers a slight improvement in all performance metrics averaged for all labels. The improvement in

both Sensitivity and Specificity means that on average, we achieve a slight decrease in the number of both False Positives and False Negatives.

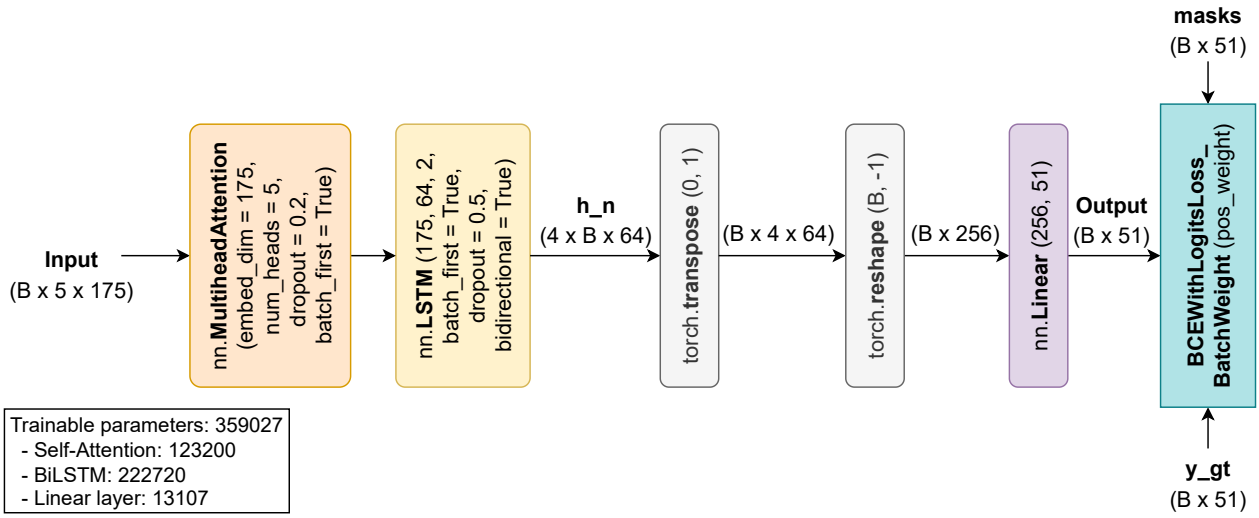


Figure 5.6.2: The best-performing Self-Attention & BiLSTM model architecture using output `h_n`. A 5-length examples sequence (`window_len = 5`) is given as input to the Multi-head Self-Attention module with `num_heads = 5` and `dropout = 0.2`, and its output passes through the two-layer BiLSTM with `hidden_size = 64` and `dropout = 0.5`. The `h_n` output of the BiLSTM (which contains a concatenation of the final forward and reverse hidden states of each BiLSTM layer), after being transposed and reshaped, passes through an output Linear layer of 51 nodes corresponding to the 51 context labels. A custom loss based on `torch.nn.BCEWithLogitsLoss` is used to account for missing labels information and label weights in loss computation.

Recognition scores of the Multi-head Self-Attention & BiLSTM model using only the final hidden states, averaged for each label subset

Label	Accuracy	Precision	Sensitivity	Specificity	F1-score	BA
Posture/Movement	0.836	0.408	0.776	0.836	0.488	0.806
Special Movement	0.815	0.029	0.732	0.816	0.056	0.774
Phone Location	0.774	0.454	0.774	0.765	0.534	0.770
Work-related	0.821	0.272	0.797	0.820	0.396	0.809
Location-based	0.830	0.279	0.797	0.827	0.354	0.812
Transportation	0.881	0.186	0.877	0.882	0.295	0.879
Chores	0.827	0.053	0.698	0.828	0.098	0.763
Self-care	0.864	0.204	0.698	0.869	0.246	0.783
Leisure Time	0.751	0.162	0.678	0.755	0.256	0.716
Companion	0.707	0.198	0.656	0.716	0.299	0.686
Environment	0.890	0.701	0.893	0.893	0.751	0.893
Average	0.818	0.248	0.756	0.819	0.323	0.788

Table 5.18: Recognition scores reported for the Multi-head Self-Attention & BiLSTM model using only the final hidden states (BiLSTM `h_n` output), averaged for each label subset. Model configuration with `window_len` = 5 and BiLSTM `hidden_size` = 64. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

Recognition scores of the Multi-head Self-Attention & BiLSTM model using only the final hidden states, for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.895	0.824	0.838	0.920	0.831	0.879
Sitting	79368	0.779	0.740	0.814	0.747	0.776	0.781
Standing	22071	0.713	0.272	0.710	0.714	0.393	0.712
Walking	11715	0.822	0.253	0.808	0.823	0.386	0.815
Running	661	0.893	0.042	0.625	0.895	0.079	0.760
Bicycling	3504	0.915	0.320	0.862	0.917	0.467	0.889
Strolling	339	0.784	0.053	0.814	0.784	0.099	0.799
Stairs - Going up	399	0.849	0.029	0.642	0.850	0.056	0.746
Stairs - Going down	390	0.820	0.023	0.618	0.822	0.044	0.720
Elevator	123	0.808	0.012	0.854	0.808	0.024	0.831
Phone in pocket	14074	0.785	0.447	0.821	0.777	0.579	0.799
Phone in hand	7313	0.676	0.179	0.736	0.670	0.288	0.703
Phone in bag	5031	0.820	0.284	0.713	0.830	0.406	0.771
Phone on table	65979	0.815	0.906	0.827	0.784	0.865	0.806
In class	2852	0.847	0.229	0.711	0.855	0.346	0.783
Lab work	2898	0.803	0.351	0.933	0.787	0.510	0.860
Computer work	22536	0.765	0.380	0.763	0.765	0.507	0.764
In a meeting	2837	0.872	0.128	0.783	0.874	0.220	0.828
At home	80044	0.799	0.772	0.823	0.777	0.797	0.800
At school	25342	0.773	0.430	0.777	0.772	0.554	0.775
At main workplace	19235	0.848	0.582	0.863	0.844	0.695	0.854
At a restaurant	1275	0.888	0.095	0.773	0.889	0.169	0.831
At a bar	520	0.881	0.164	0.862	0.881	0.275	0.871
At a party	404	0.835	0.072	0.770	0.836	0.132	0.803
At the gym	897	0.826	0.097	0.632	0.832	0.168	0.732
At the beach	116	0.788	0.020	0.879	0.788	0.040	0.834
In a car	3550	0.903	0.246	0.872	0.904	0.384	0.888
On a bus	1179	0.847	0.067	0.882	0.846	0.125	0.864
Drive - Driver	4879	0.892	0.309	0.883	0.893	0.458	0.888
Drive - Passenger	1650	0.883	0.122	0.869	0.884	0.215	0.876
Shopping	809	0.803	0.039	0.786	0.803	0.074	0.795
Cooking	2212	0.841	0.079	0.734	0.843	0.142	0.789
Cleaning	1813	0.809	0.070	0.671	0.812	0.127	0.741
Doing laundry	471	0.814	0.028	0.594	0.816	0.053	0.705
Washing dishes	829	0.866	0.050	0.704	0.868	0.093	0.786
Bathing - Shower	1120	0.866	0.050	0.698	0.868	0.093	0.783
Toilet	1558	0.816	0.042	0.630	0.819	0.079	0.724
Grooming	1775	0.842	0.065	0.621	0.846	0.117	0.733
Dressing	1248	0.885	0.063	0.681	0.887	0.115	0.784
Sleeping	40869	0.909	0.799	0.860	0.926	0.828	0.893
Exercise	5191	0.900	0.232	0.759	0.905	0.355	0.832
Eating	9668	0.695	0.125	0.681	0.696	0.211	0.688
Drinking alcohol	859	0.884	0.133	0.820	0.885	0.229	0.852
Watching TV	8945	0.786	0.234	0.653	0.799	0.344	0.726
Surfing the internet	10668	0.686	0.167	0.514	0.706	0.252	0.610
Talking	18477	0.613	0.199	0.716	0.598	0.312	0.657
Singing	384	0.693	0.046	0.604	0.695	0.086	0.650
With co-workers	3972	0.746	0.167	0.756	0.746	0.274	0.751
With friends	12686	0.667	0.228	0.557	0.686	0.324	0.621
Indoors	102510	0.894	0.993	0.894	0.900	0.941	0.897
Outside	6793	0.887	0.410	0.892	0.886	0.562	0.889
Average		0.818	0.248	0.756	0.819	0.323	0.788

Table 5.19: Recognition scores reported for the Multi-head Self-Attention & BiLSTM model using the final hidden states, for all context labels. Configuration with `window_len` = 5 and BiLSTM `hidden_size` = 64.

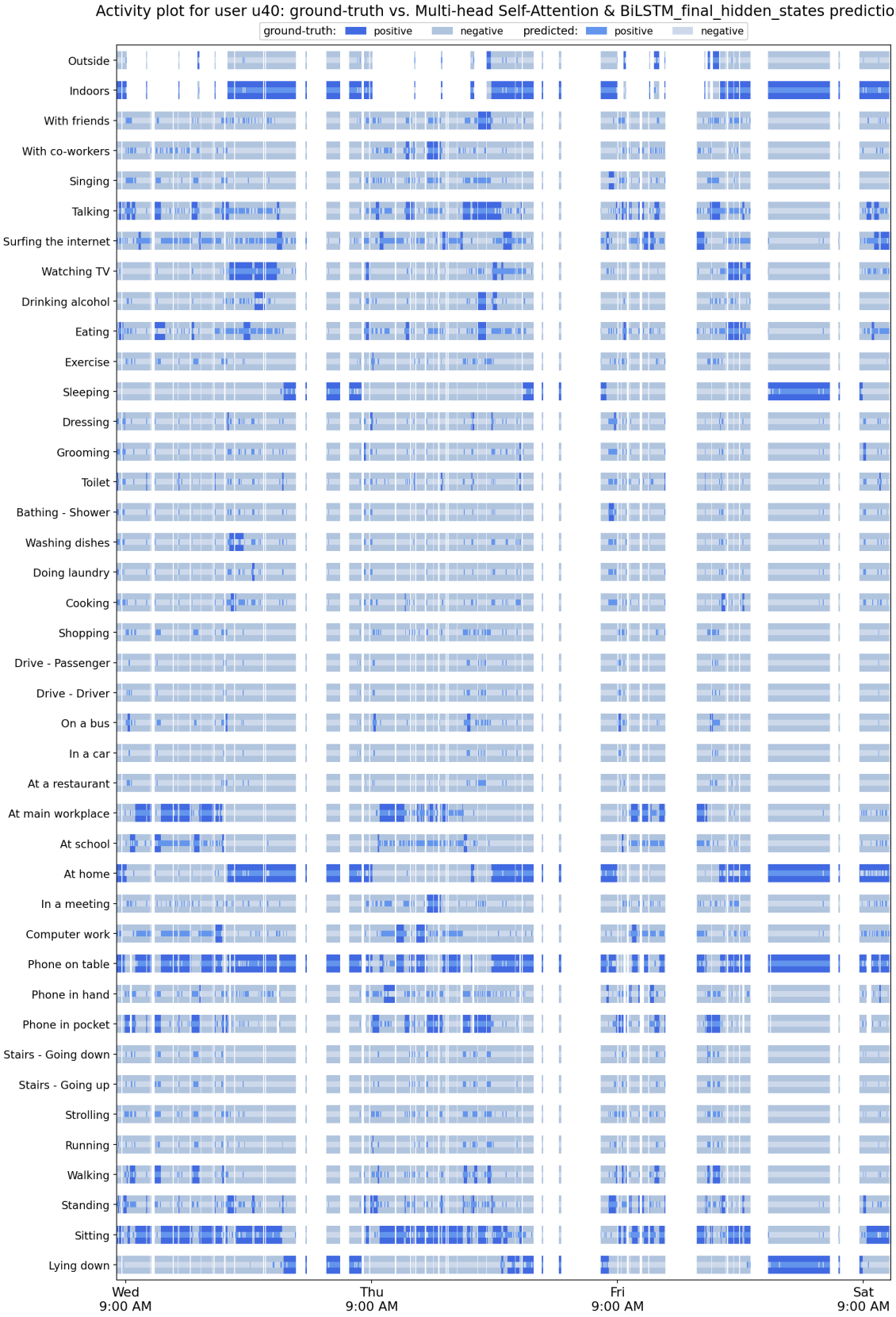


Figure 5.6.3: Activity plot including the predictions of the Multi-head Self-Attention & BiLSTM model using final hidden states for user u40

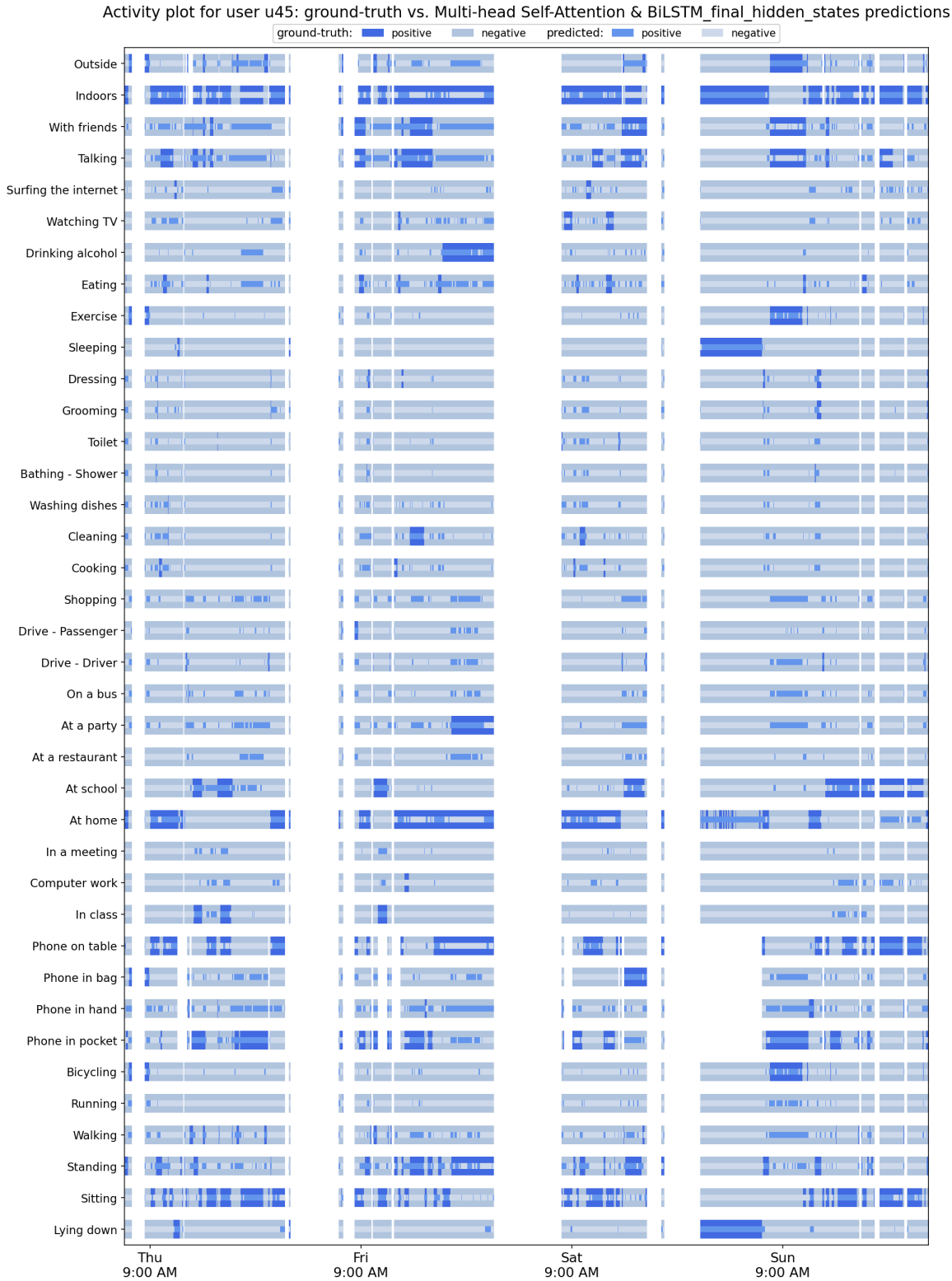


Figure 5.6.4: Activity plot including the predictions of the Multi-head Self-Attention & BiLSTM model using final hidden states for user u45

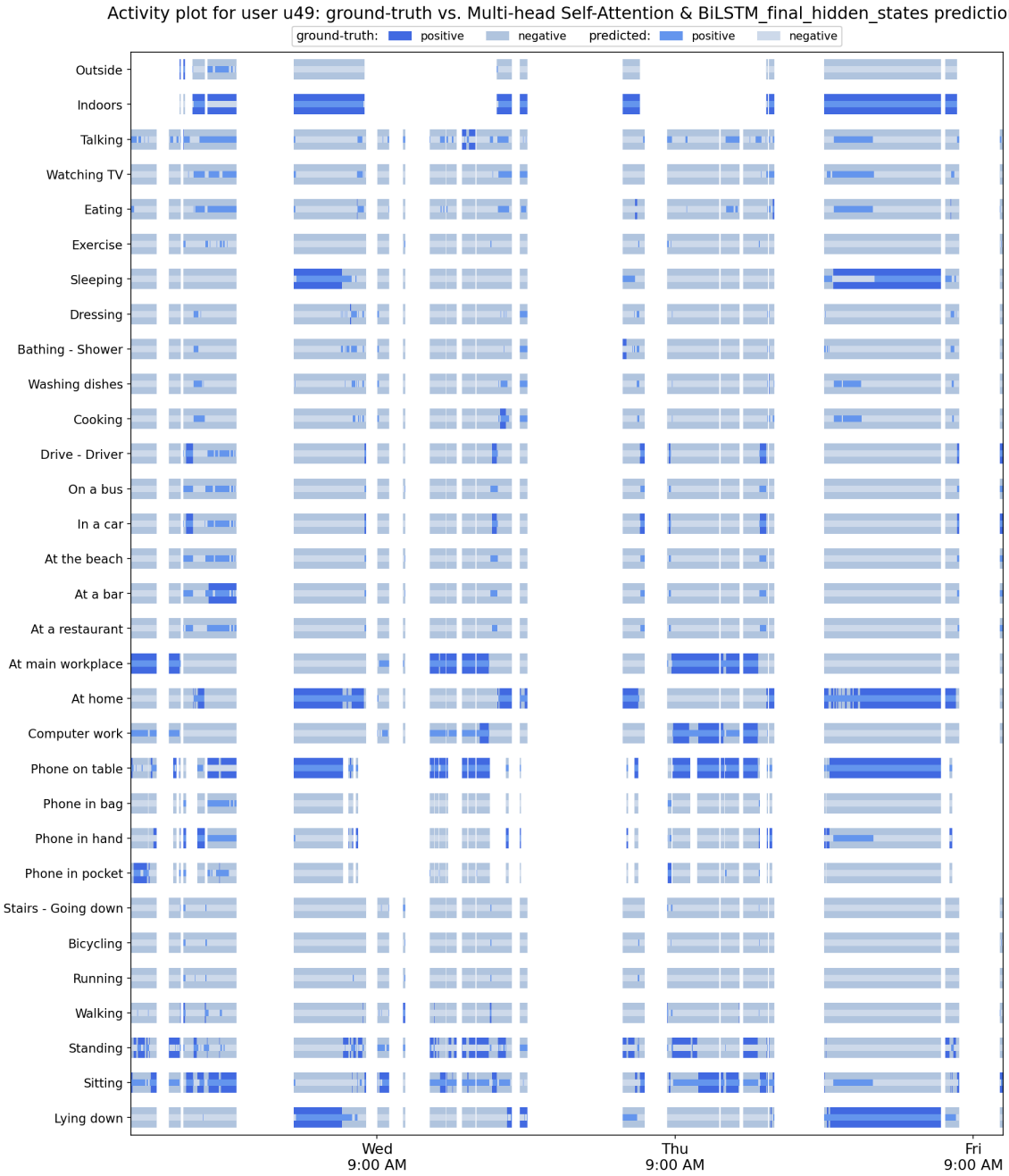


Figure 5.6.5: Activity plot including the predictions of the Multi-head Self-Attention & BiLSTM model using final hidden states for user u49

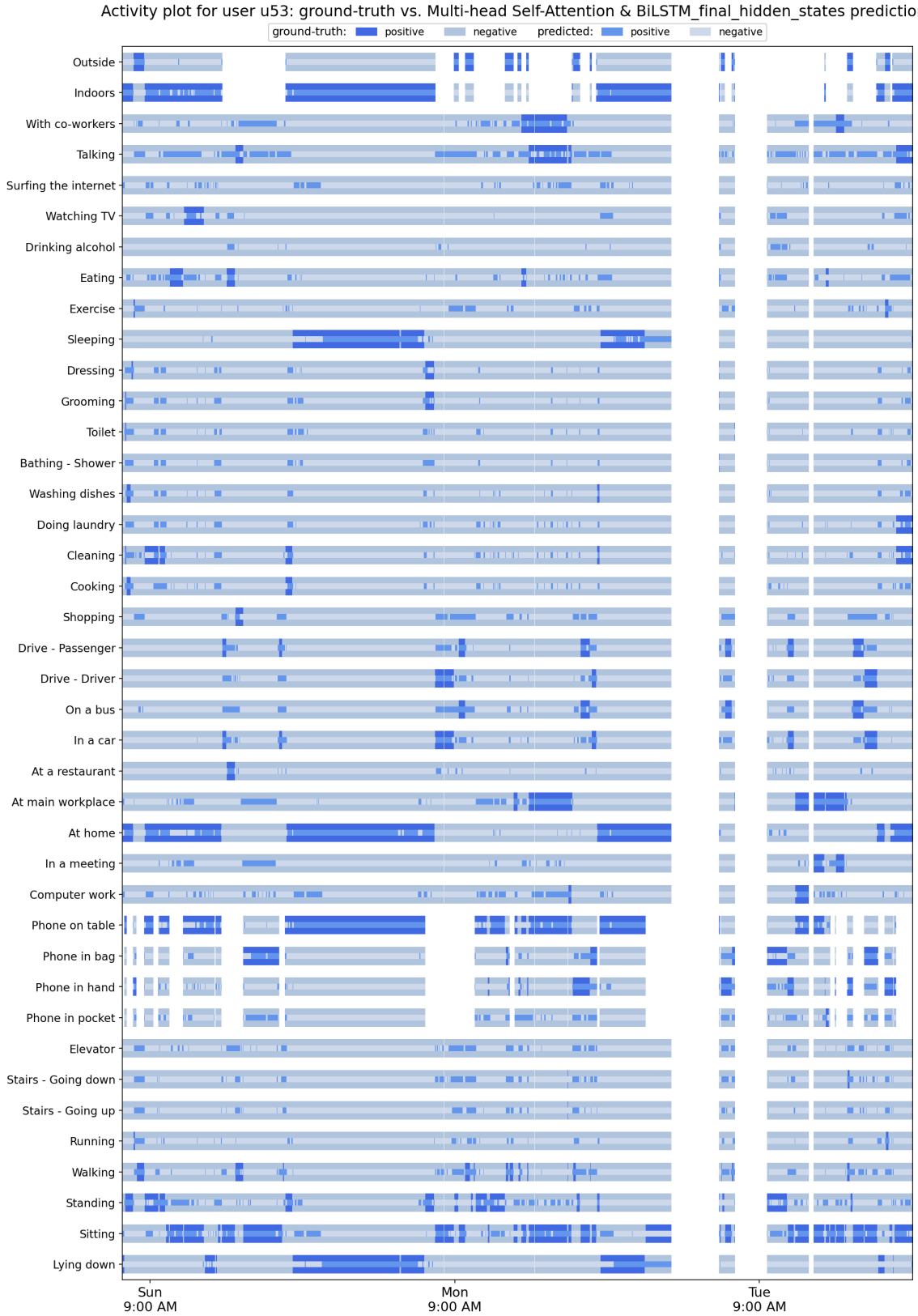


Figure 5.6.6: Activity plot including the predictions of the Multi-head Self-Attention & BiLSTM model using final hidden states for user u53

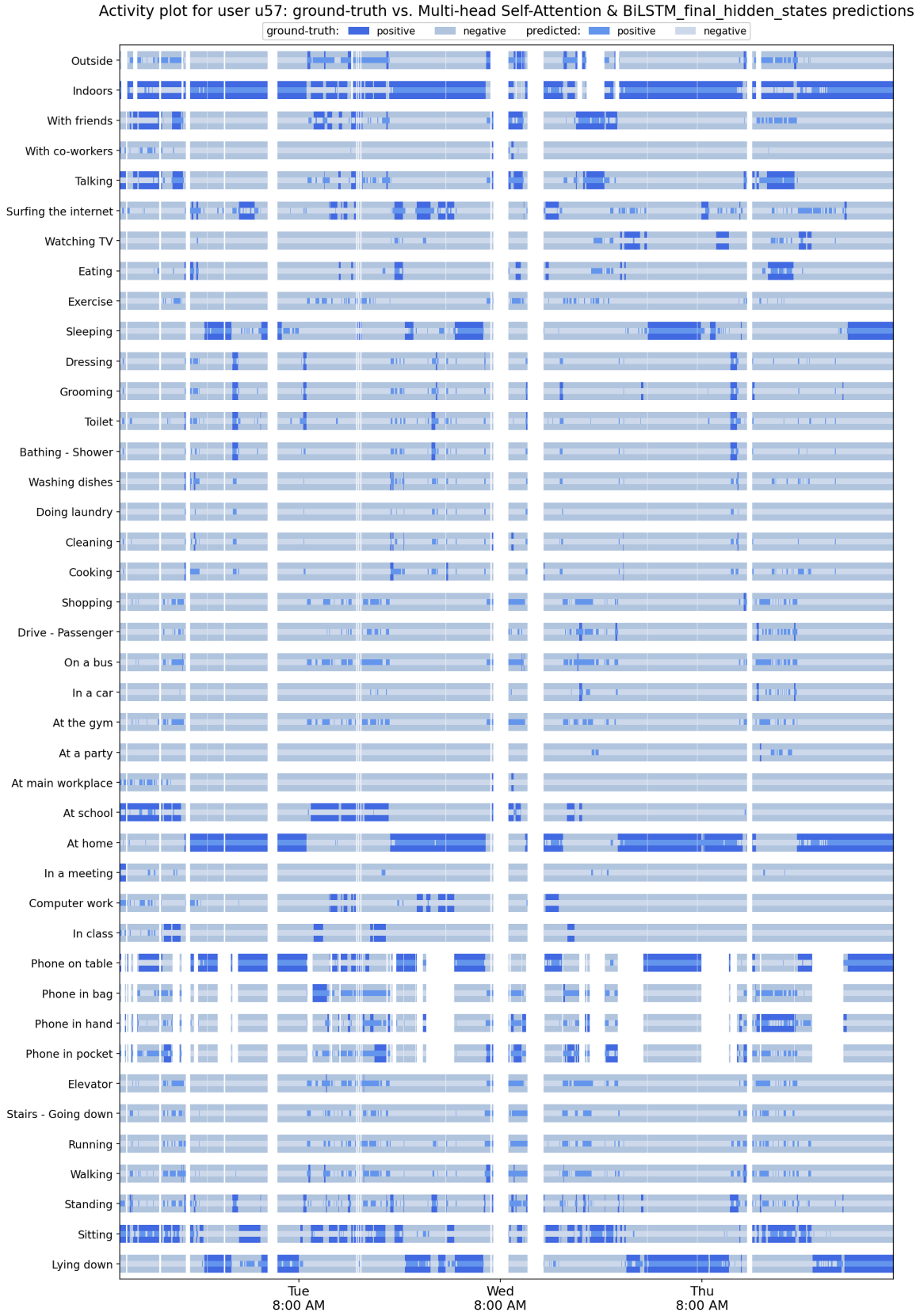


Figure 5.6.7: Activity plot including the predictions of the Multi-head Self-Attention & BiLSTM model using final hidden states for user u57

5.6.2 BiLSTM using Output for all Timesteps followed by Features' Cross-Attention

In this experiment, we expand the model of Subsection 5.5.2 by adding a cross-attention module after the BiLSTM. The full `output` of the BiLSTM, which contains the output features (`h_t`) from the last layer of the BiLSTM for all timesteps is used to produce the query `Q` in the cross-attention module, while the input features, after being transposed, are used to produce both key `K` and value `V` in the cross-attention module. The output of the cross-attention module is concatenated with the BiLSTM `output` and then they pass through a Linear layer to predict the probabilities for each of the 51 context labels. This model is conceptually presented in Figure 5.6.8, and the key points of this experiment are summarized as follows:

- Five-fold cross validation is used, with each fold containing 48 users in the training set and 12 users in the test set (using the same partition as the original work).
- The training set of each CV iteration is further partitioned in a training subset and a validation subset, as follows: for each of the 48 users of the training set, 80% of their data is used for training, and 20% is used for validation, to decide which epoch's checkpoint will be selected for the test inference.
- We standardize each sensor feature (mean and standard deviation are calculated using the training subset). After standardization, missing feature values are zero-imputated.
- The model configuration includes a two-layer BiLSTM and a Multi-head Cross-Attention module, followed by a Linear layer. The `N`-length examples sequence is given as input to the BiLSTM, whose full `output`, which contains the output features (`h_t`) from the last layer of the BiLSTM for each timestep, is then used to produce the query `Q` in the Cross-Attention module, while the input examples sequence, after being transposed, is used to produce both key `K` and value `V` in the Cross-Attention module. The `attn_output` of the Cross-Attention module is concatenated with the BiLSTM `output` and then they pass through a Linear layer of 51 nodes which correspond to the 51 context labels.
- Regarding the BiLSTM, we use Dropout after the first layer, with `dropout rate = 0.5` to help avoid overfitting. In this model, we do not try to optimize the two hyperparameters, `window_len` and `hidden_size`, as we did in previous experiments. Instead, we use the best configuration of these hyperparameters, `window_len = 5` and BiLSTM layers' `hidden_size = 64`, as determined in our previous experiments, to achieve performance comparable to them, and we mostly focus on visualizing the Attention weights, to unveil and better understand the input features' importance on our model's predictions.
- Regarding the Cross-Attention, we use Multi-head Attention, with Scaled Dot-Product Attention in each head. The hyperparameters `num_heads = 2` and `dropout rate = 0.5` were found to produce the best results.
- Regarding the loss, we again use the custom loss that was first described in Subsection 5.4.4, based on `torch.nn.BCEWithLogitsLoss`. This loss is slightly modified to allow dynamic per-batch, per-element masking in the loss matrices (to mask the loss elements corresponding to missing ground-truth labels for each batch, and not use them in the loss computation). Also, the `pos_weight` is used for instance-weighting to account for the imbalance in the number of positive examples per context label, by multiplying the term of the positive examples in the loss, with the ratio of negative to positive examples for this label in the training set.
- We use a `batch size` of 32 to train the model.
- We use the Adam optimizer to train the model, with `learning rate = 0.00005`.
- In the testing phase, the continuous predictions of the model pass through a sigmoid activation function, and then they are converted to binary outputs using a threshold of 0.5.
- We calculate the evaluation metrics for each label over its non-missing ground-truth examples.

We use `PyTorch` to build the model, the supervisor and the training scripts. The `torch.nn` modules and `torch` functions that are used, can be seen in detail in Figure 5.6.9.

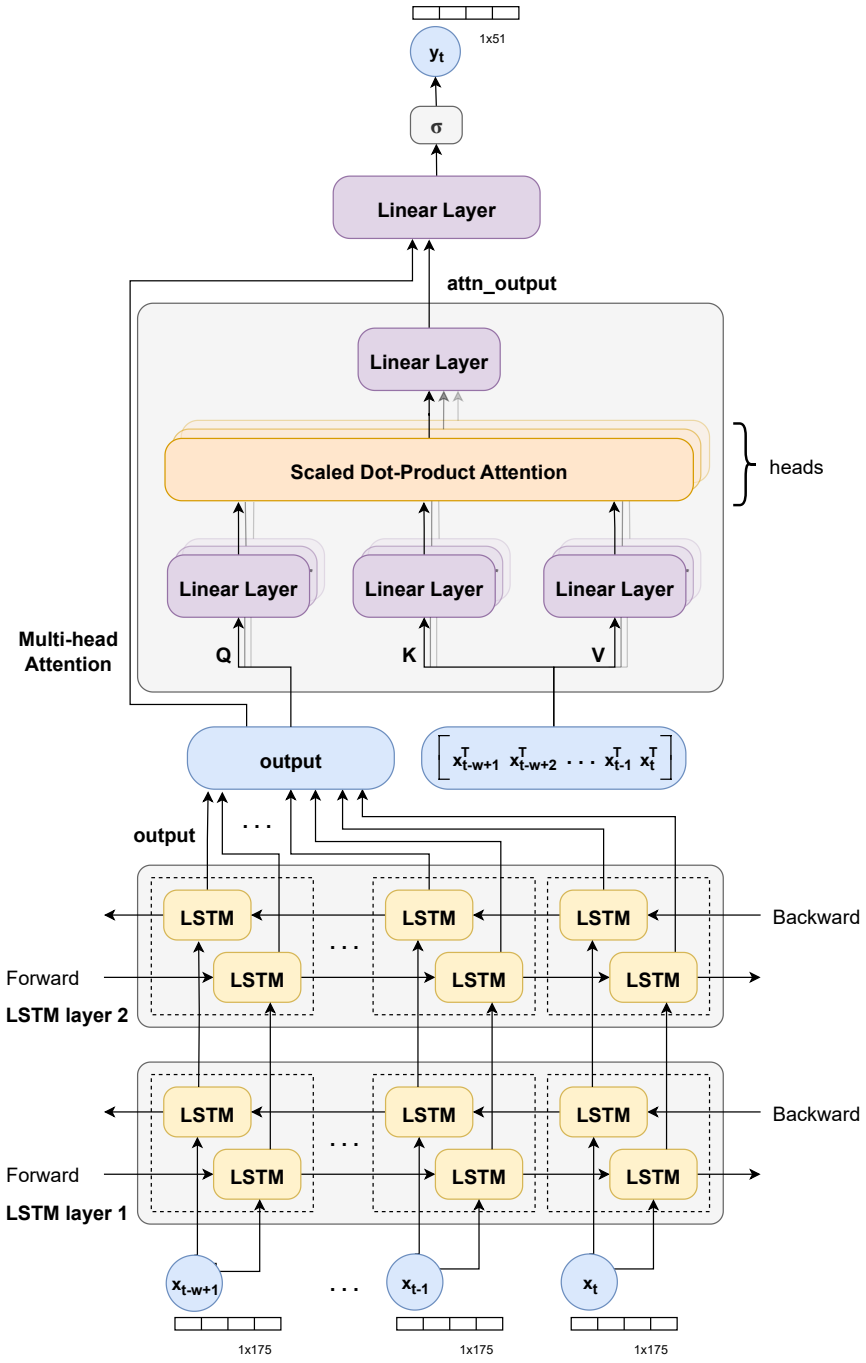


Figure 5.6.8: A two-layer BiLSTM model followed by a Multi-head Cross-Attention module is presented conceptually. An examples sequence of length `w (window_len)` is given as input, and the target is to predict the context labels of the last example of the sequence. The `output` of the BiLSTM is used to produce the query `Q`, while the input features, after being transposed, are used to produce both key `K` and value `V` in the Cross-Attention module. The `attn_output` is concatenated with the BiLSTM `output` and then they pass through a Linear layer, followed by a sigmoid function to convert the model's outputs to labels' probabilities.

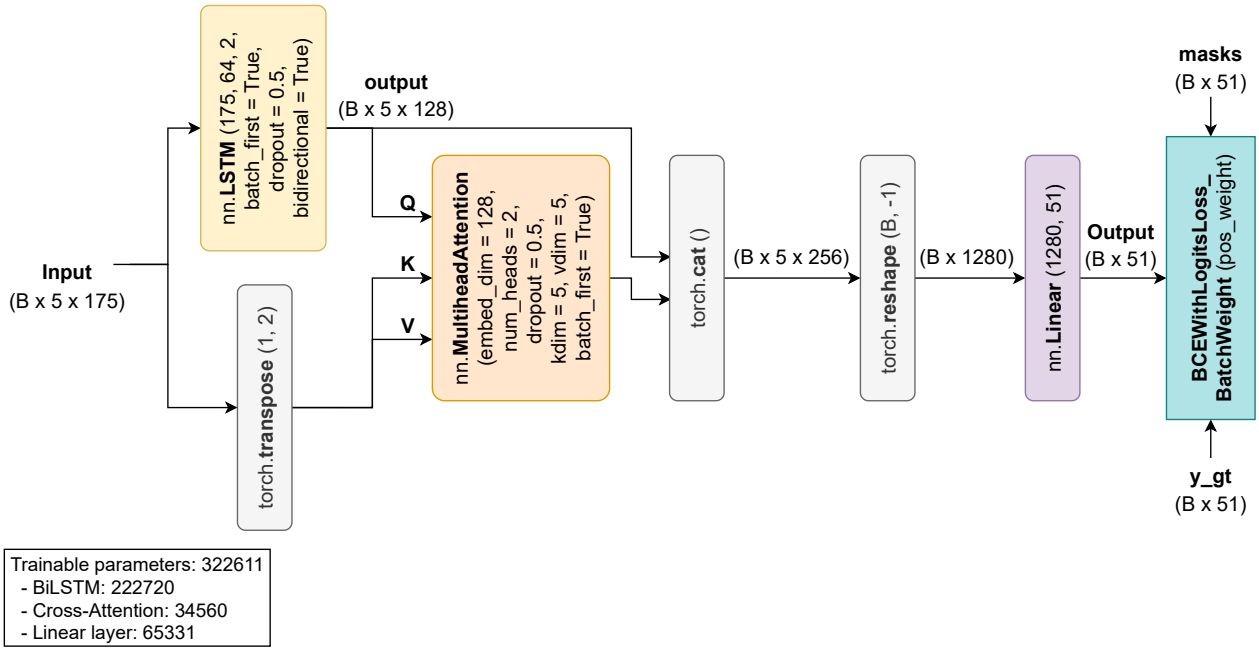


Figure 5.6.9: A two-layer BiLSTM model followed by a Multi-head Cross-Attention module. A 5-length examples sequence (`window_len = 5`) is given as input to the two-layer BiLSTM with `hidden_size = 64` and `dropout = 0.5`. The `output` of the BiLSTM (which contains the output features `h_t` from the last layer of the BiLSTM for each timestep) is used to produce the query `Q`, while the input features, after being transposed, are used to produce both key `K` and value `V` in the Cross-Attention module with `num_heads = 2` and `dropout = 0.5`. The `attn_output` is concatenated with the BiLSTM `output` and then, after being reshaped, they pass through an output Linear layer of 51 nodes corresponding to the 51 context labels. A custom loss based on `torch.nn.BCEWithLogitsLoss` is used to account for missing labels information and label weights in loss computation.

Recognition scores of the BiLSTM model using all outputs & features' Cross-Attention, averaged for each label subset

Label	Accuracy	Precision	Sensitivity	Specificity	F1-score	BA
Posture/Movement	0.817	0.390	0.799	0.813	0.469	0.806
Special Movement	0.795	0.028	0.748	0.796	0.053	0.772
Phone Location	0.756	0.440	0.766	0.755	0.515	0.761
Work-related	0.803	0.259	0.819	0.800	0.380	0.809
Location-based	0.817	0.272	0.809	0.815	0.341	0.812
Transportation	0.861	0.166	0.881	0.861	0.267	0.871
Chores	0.799	0.049	0.766	0.799	0.092	0.783
Self-care	0.845	0.199	0.692	0.850	0.238	0.771
Leisure Time	0.729	0.144	0.669	0.731	0.231	0.700
Companion	0.715	0.197	0.617	0.728	0.293	0.672
Environment	0.885	0.695	0.891	0.889	0.744	0.890
Average	0.800	0.238	0.767	0.801	0.309	0.784

Table 5.20: Recognition scores reported for the BiLSTM model using all outputs & features' Multi-head Cross-Attention, averaged for each label subset. Model configuration with `window_len` = 5 and BiLSTM `hidden_size` = 64. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

Recognition scores of the BiLSTM model using all outputs & features' Cross-Attention, for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.899	0.833	0.844	0.924	0.838	0.884
Sitting	79368	0.782	0.740	0.827	0.742	0.781	0.785
Standing	22071	0.677	0.249	0.730	0.669	0.371	0.700
Walking	11715	0.803	0.236	0.825	0.801	0.367	0.813
Running	661	0.864	0.036	0.675	0.866	0.068	0.770
Bicycling	3504	0.878	0.247	0.892	0.877	0.387	0.884
Strolling	339	0.787	0.052	0.796	0.787	0.098	0.792
Stairs - Going up	399	0.820	0.026	0.692	0.821	0.051	0.756
Stairs - Going down	390	0.793	0.022	0.692	0.794	0.043	0.743
Elevator	123	0.782	0.010	0.813	0.781	0.020	0.797
Phone in pocket	14074	0.768	0.426	0.833	0.753	0.564	0.793
Phone in hand	7313	0.671	0.177	0.739	0.665	0.286	0.702
Phone in bag	5031	0.789	0.246	0.697	0.797	0.363	0.747
Phone on table	65979	0.797	0.911	0.794	0.805	0.849	0.800
In class	2852	0.834	0.221	0.759	0.838	0.342	0.798
Lab work	2898	0.783	0.326	0.914	0.766	0.480	0.840
Computer work	22536	0.769	0.386	0.772	0.768	0.515	0.770
In a meeting	2837	0.827	0.102	0.830	0.827	0.181	0.828
At home	80044	0.799	0.778	0.813	0.786	0.795	0.800
At school	25342	0.782	0.442	0.769	0.785	0.561	0.777
At main workplace	19235	0.842	0.571	0.857	0.838	0.685	0.847
At a restaurant	1275	0.849	0.078	0.858	0.848	0.144	0.853
At a bar	520	0.859	0.140	0.848	0.859	0.240	0.854
At a party	404	0.786	0.065	0.906	0.784	0.121	0.845
At the gym	897	0.785	0.079	0.635	0.789	0.141	0.712
At the beach	116	0.831	0.023	0.784	0.831	0.044	0.808
In a car	3550	0.876	0.202	0.874	0.876	0.328	0.875
On a bus	1179	0.833	0.063	0.892	0.832	0.117	0.862
Drive - Driver	4879	0.889	0.301	0.871	0.890	0.447	0.880
Drive - Passenger	1650	0.847	0.097	0.886	0.846	0.175	0.866
Shopping	809	0.784	0.039	0.865	0.783	0.075	0.824
Cooking	2212	0.799	0.067	0.785	0.800	0.123	0.792
Cleaning	1813	0.796	0.069	0.712	0.797	0.126	0.755
Doing laundry	471	0.783	0.027	0.673	0.784	0.051	0.728
Washing dishes	829	0.833	0.045	0.796	0.834	0.085	0.815
Bathing - Shower	1120	0.839	0.037	0.604	0.841	0.069	0.723
Toilet	1558	0.793	0.037	0.628	0.795	0.071	0.711
Grooming	1775	0.823	0.059	0.640	0.826	0.109	0.733
Dressing	1248	0.855	0.052	0.705	0.857	0.096	0.781
Sleeping	40869	0.917	0.812	0.881	0.930	0.845	0.905
Exercise	5191	0.852	0.169	0.785	0.854	0.278	0.820
Eating	9668	0.651	0.117	0.732	0.646	0.201	0.689
Drinking alcohol	859	0.824	0.091	0.816	0.825	0.164	0.820
Watching TV	8945	0.771	0.230	0.707	0.777	0.347	0.742
Surfing the internet	10668	0.688	0.164	0.495	0.710	0.246	0.602
Talking	18477	0.622	0.204	0.722	0.608	0.319	0.665
Singing	384	0.693	0.034	0.427	0.699	0.062	0.563
With co-workers	3972	0.745	0.155	0.680	0.750	0.253	0.715
With friends	12686	0.684	0.238	0.553	0.705	0.333	0.629
Indoors	102510	0.890	0.993	0.889	0.899	0.938	0.894
Outside	6793	0.881	0.397	0.893	0.880	0.550	0.887
Average		0.800	0.238	0.767	0.801	0.309	0.784

Table 5.21: Recognition scores reported for the BiLSTM using all outputs & features' Multi-head Cross-Attention, for all context labels. Configuration with `window_len` = 5 & BiLSTM `hidden_size` = 64.

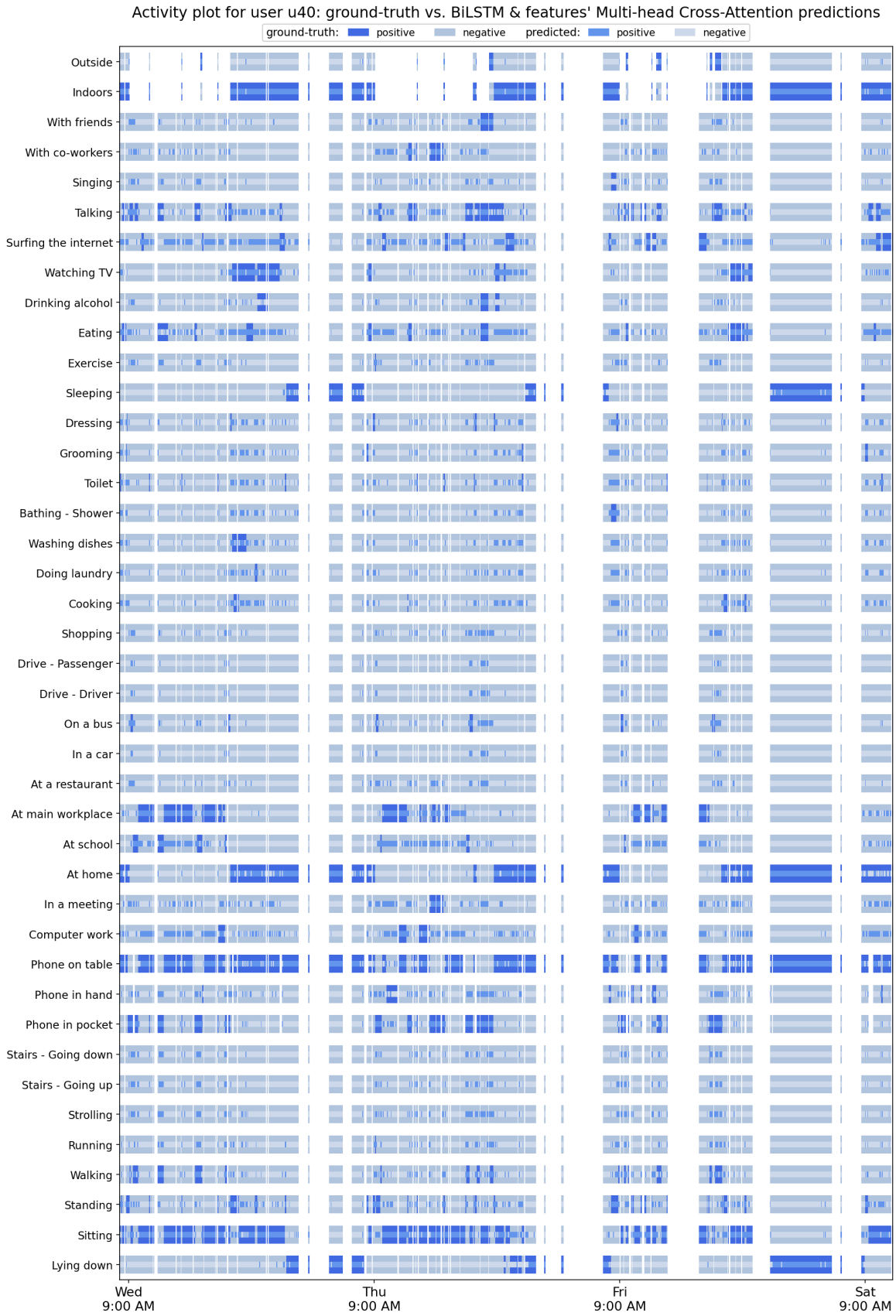


Figure 5.6.10: Activity plot including the predictions of the BiLSTM model using all outputs & features' Multi-head Cross-Attention for user u40

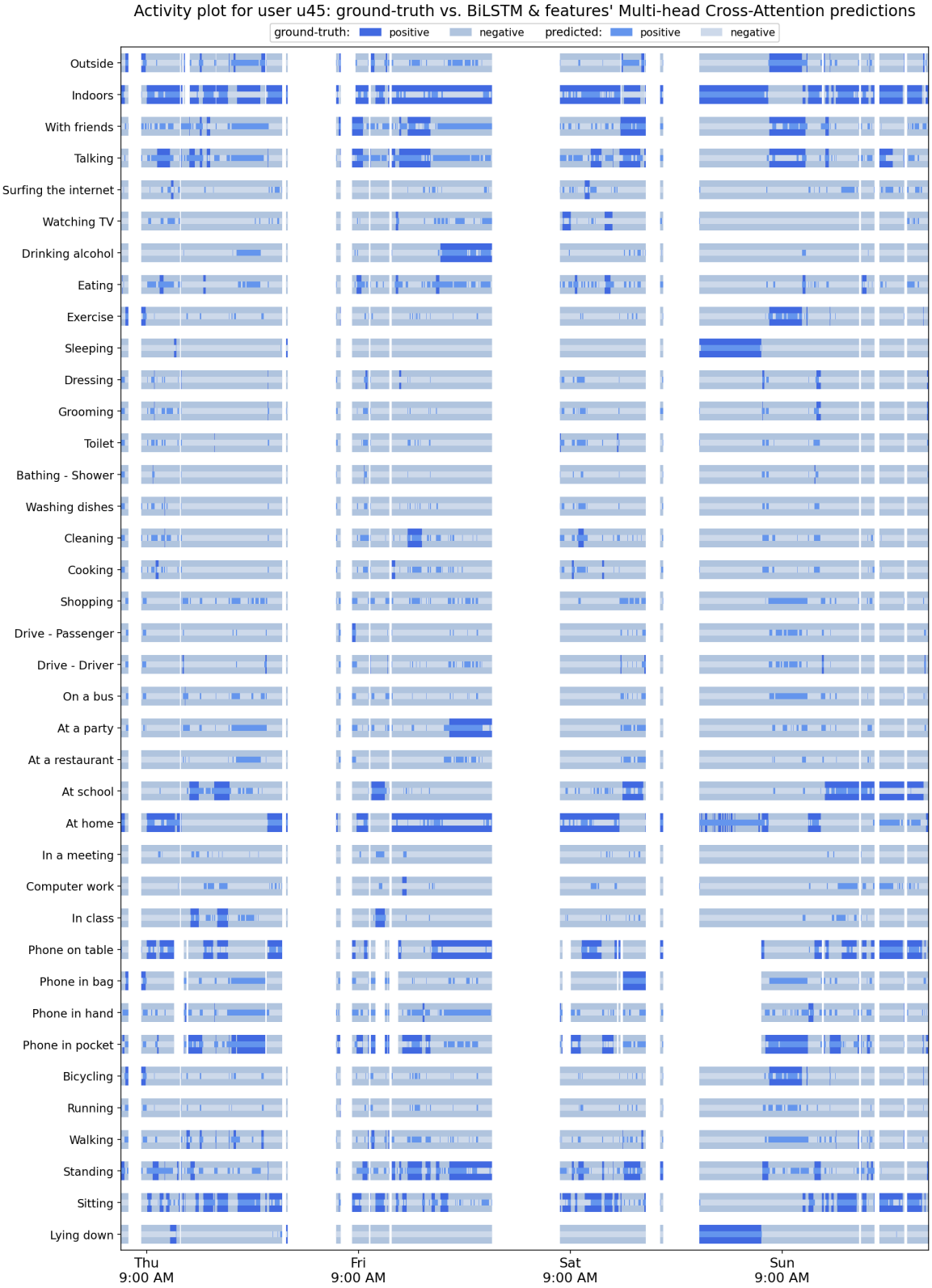


Figure 5.6.11: Activity plot including the predictions of the BiLSTM model using all outputs & features' Multi-head Cross-Attention for user u45

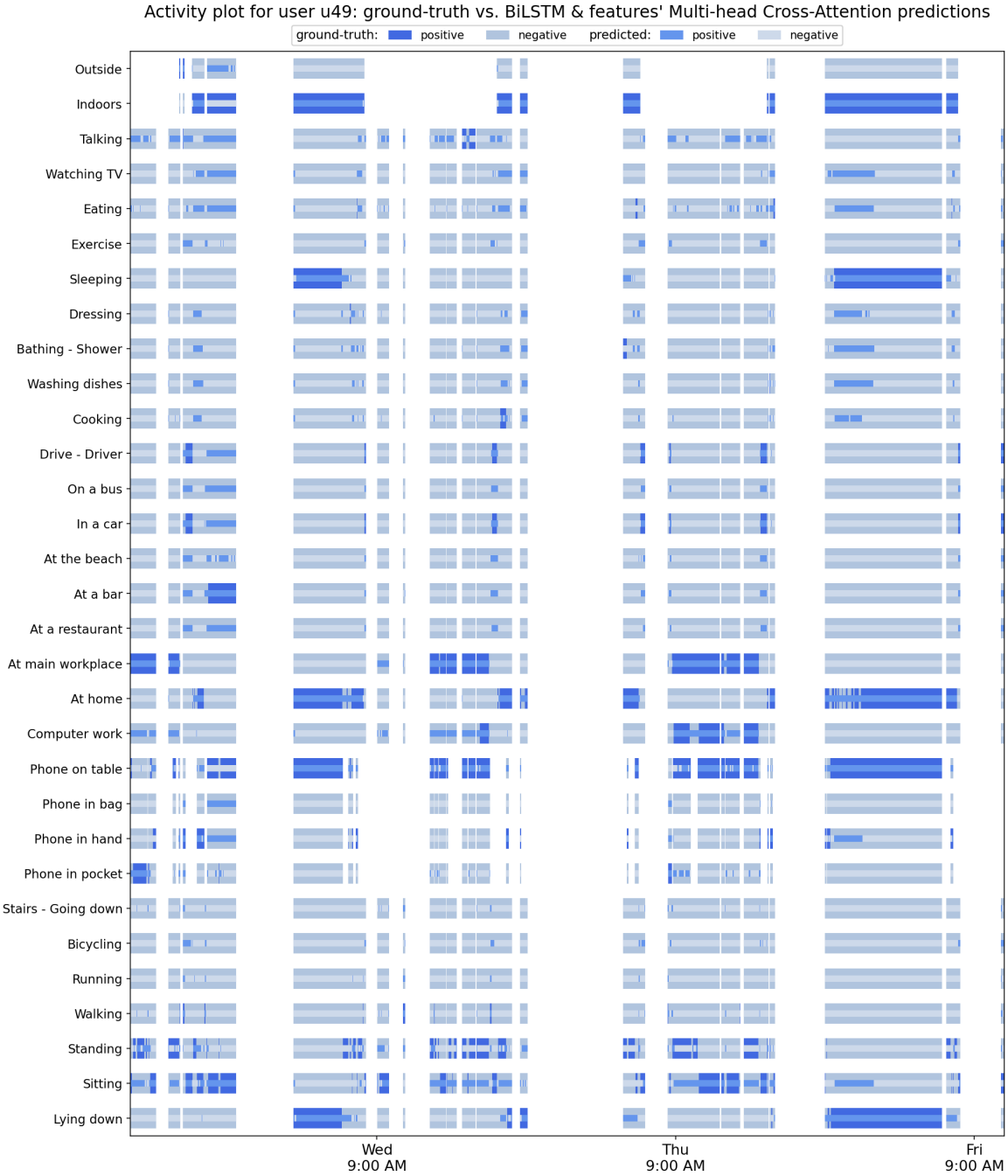


Figure 5.6.12: Activity plot including the predictions of the BiLSTM model using all outputs & features' Multi-head Cross-Attention for user u49

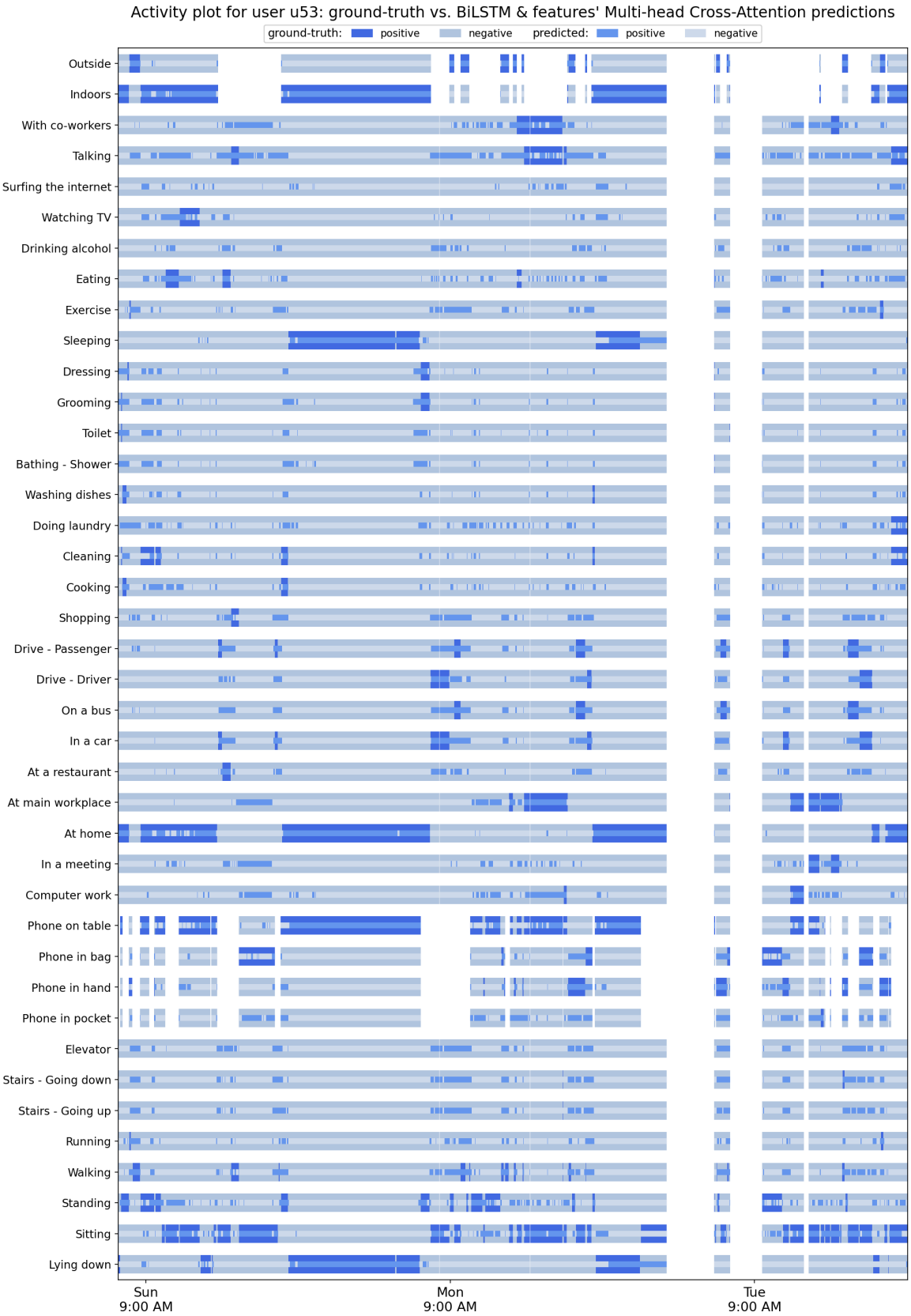


Figure 5.6.13: Activity plot including the predictions of the BiLSTM model using all outputs & features' Multi-head Cross-Attention for user u53

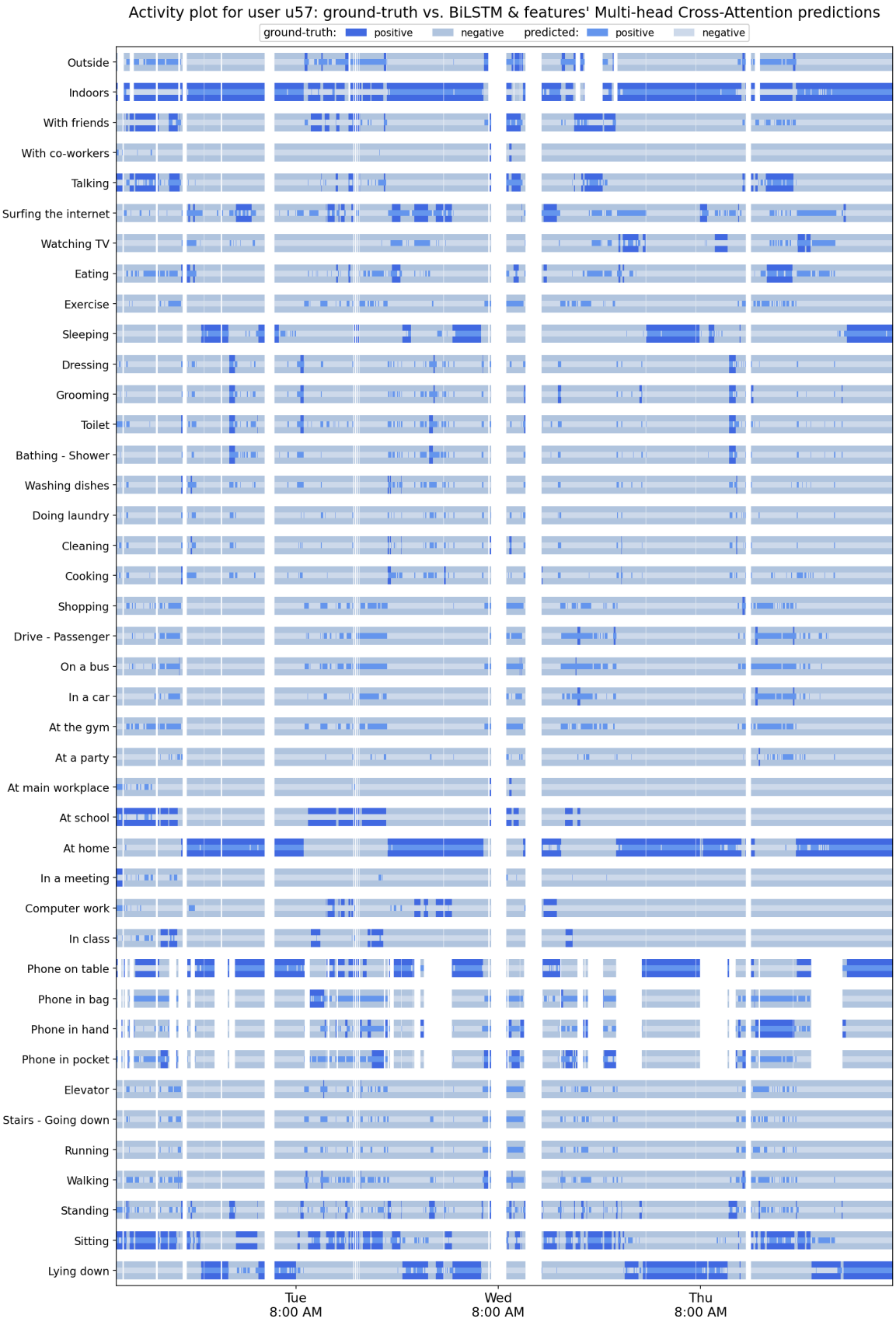


Figure 5.6.14: Activity plot including the predictions of the BiLSTM model using all outputs & features' Multi-head Cross-Attention for user u57

For the model configuration with `window_len` = 5 and BiLSTM layers' `hidden_size` = 64, we present the detailed model architecture in Figure 5.6.9, and the recognition scores for all context labels in Table 5.21 and averaged over the labels of each label subset in Table 5.20. Compared to the previous BiLSTM and Self-Attention & BiLSTM models, this model has comparable or a little worse performance, as it can be observed from the averaged recognition scores. More specifically, compared to the previous Self-Attention & BiLSTM using final hidden states only (Subsection 5.6.1), the BiLSTM using all outputs & features' Cross-Attention has slightly worse performance regarding all metrics (a BA of 0.784 compared to the respective best BA of 0.788 of the Self-Attention & BiLSTM model), except for Sensitivity, which is 0.767, where we notice an improvement compared to the 0.756 value of the previous model. Overall, taking into consideration that the BiLSTM & features' Cross-Attention model has the highest Sensitivity and the lowest Specificity among all BiLSTM and Attention & BiLSTM models that were implemented, we can conclude that on average this model produces the less False Negatives and the most False Positives so far.

5.6.3 Visualizing Cross-Attention Weights for Interpretability

The BiLSTM & features' Cross-Attention model of Subsection 5.6.2 is included in this thesis not only for its performance regarding Sensitivity, but also and mainly as a study on Neural Network Interpretability. Similarly to [CR19; Che+20], we visualize the weights of the Cross-Attention module between the BiLSTM `output` for all timesteps and the features of the input examples sequence, to investigate whether they can provide useful insights about which features are more impactful in the model's predictions for various labels, and also whether these predictions are meaningful or if the model is misdirected by false correlations between input features and context labels.

During inference time, for each input example sequence, we save the `attn_output_weights` of the Cross-Attention module after converting the Tensor to a `NumPy array`. We plot examples of Cross-Attention weights and try to understand the model's predictions. In the following plots of Cross-Attention weights, the x-axis corresponds to the 175 input features of the core sensors (Acc, Gyro, WAcc, Loc & Loc QF, Aud, PS) which are locally grouped by sensor. The y-axis corresponds to the 5 BiLSTM output timesteps (the current minute and the previous four minutes). Darker color corresponds to larger weights, which indicate that the specific feature probably plays a more important role in the prediction.

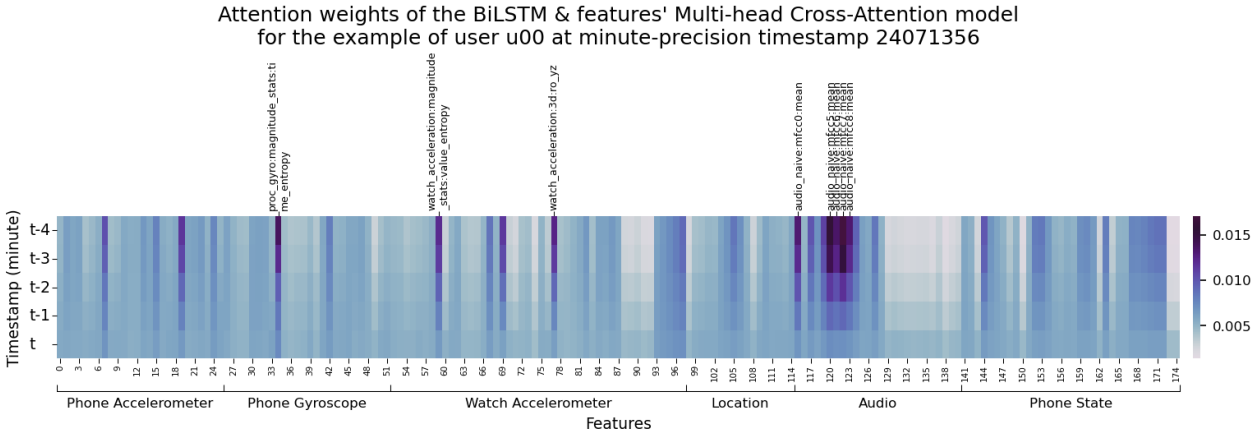


Figure 5.6.15: Attention weights of the BiLSTM & features' Cross-Attention model for u00 and t24071356
Ground-truth labels: Sitting, Indoors, At home, Watching TV, Eating, Phone on table
Predicted labels: Sitting, Indoors, At home, Watching TV, Surfing the internet, Talking, Eating, Phone on table, With friends

In Figure 5.6.15, we visualize the Attention weights for the example of user u00 and minute-precision timestamp 24071356. We observe that the features that seem most important are the MFCC mean values, and also the time-entropy magnitude statistic (entropy calculated from normalizing the magnitude signal and treating it as a probability distribution, which is designed to detect peakiness in time—sudden bursts of magnitude) from the smartphone gyroscope signal, the value-entropy magnitude statistic (entropy calculated from a histogram of quantization of the magnitude values to 20 bins) and the ro_yz 3D feature (y-z inter-

axis correlation coefficient) from the watch accelerometer signal. The importance on the specific watch accelerometer features can explain the recognition of "Eating", and also might lead to the false prediction of "Surfing the internet" which might include some similar hand movements. "Sitting", "Watching TV", "Surfing the internet", "Eating", and "Phone on table" all might possibly share the characteristic of phone immobility, and that may be why they are all predicted at the same time, given the importance to the time-entropy smartphone gyroscope magnitude feature. Also, the MFCC mean values are indicative of the sound levels and characteristics, since they capture the shape of the power spectrum of the received sound signal, and are useful to predict "Watching TV", which might could be confused with "Talking" and "With friends" that could be assigned in environments with similar sounds.

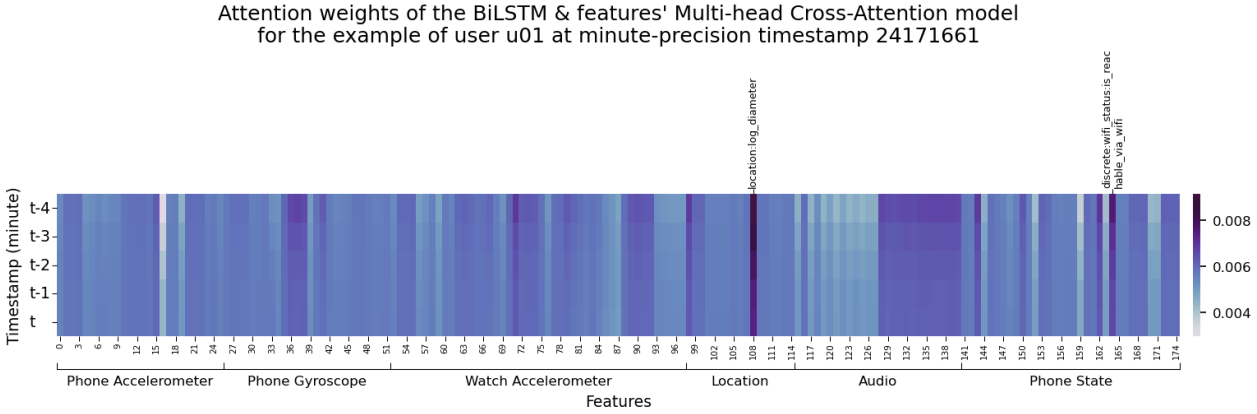


Figure 5.6.16: Attention weights of the BiLSTM & features' Cross-Attention model for u01 and t24171661
Ground-truth labels: Sitting, Indoors, At home
Predicted labels: Indoors, Toilet, Phone on table

In Figure 5.6.16, we visualize the Attention weights for the example of user u01 and minute-precision timestamp 24171661. We observe that a lot of features have comparatively large attention weights, but the most prevalent ones are two: the log_diameter location feature (log of the maximum distance between two locations in the recording session) and the reachable Wi-Fi phone state feature. The "Indoors" context label is predicted correctly, while our model also predicts "Toilet" and "Phone on table" instead of "Sitting" and "At home". "Sitting" and "Toilet" are pretty much alike and the feature importance of the aforementioned two features is justifiable, since they are activities that involve no movement (the same goes for the "Phone on table" label), and also are usually combined with being in Wi-Fi reach. However it remains unclear why the "At home" label was not predicted alongside "Indoors" and "Toilet".

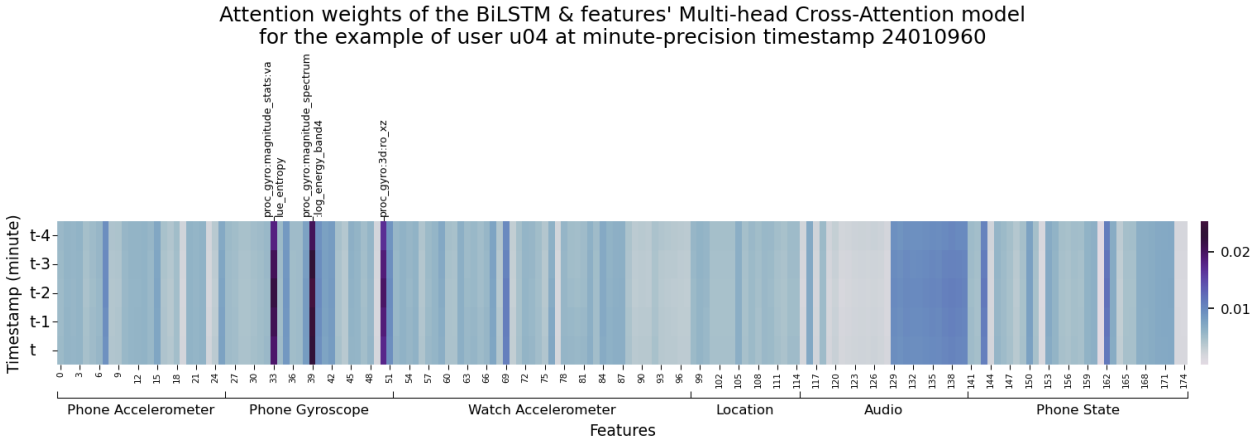


Figure 5.6.17: Attention weights of the BiLSTM & features' Cross-Attention model for u04 and t24010960
Ground-truth labels: Sitting, Indoors, At home, Computer work, Phone on table
Predicted labels: Sitting, Indoors, At home, Surfing the internet, Computer work, Eating, Phone on table

In Figure 5.6.17, we visualize the Attention weights for the example of user u04 and minute-precision timestamp 24010960. The features which appear more prevalent in the plot come from the smartphone gyroscope, and include the value-entropy magnitude statistic, the band4 log energy feature of the magnitude spectrum, and the ro_xz 3D feature (x-z inter-axis correlation coefficient). Supposing the phone is placed on a table, as denoted by the respective ground-truth label "Phone on table", it probably remains still, so these features have large weights to indicate the immobility and the orientation of the phone. However correctly predicted labels such as "Sitting" and "Computer work" cannot be directly predicted from the phone position since the phone does not participate in any way in these activities, but there might be a correlation between phone placement and activity, e.g., the phone is usually placed on the table, flat and still, during "Computer work". Moreover, some of the correctly predicted labels, as well as the falsely predicted labels "Surfing the internet" and "Eating" might be indicated also by the MFCC std features that appear with light blue in the plot (the right half of the features denoted as "Audio"), which express the standard deviation of each MFCC feature over time.

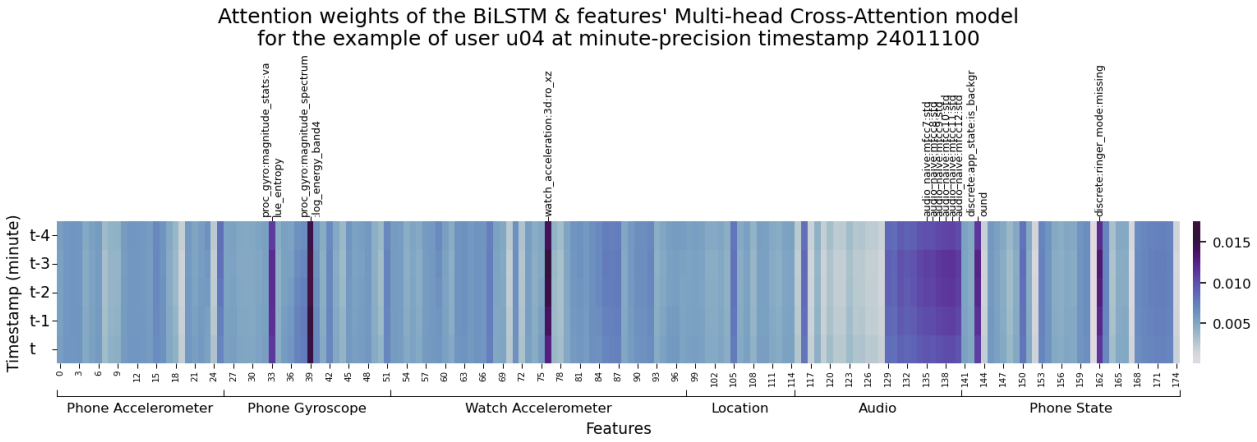


Figure 5.6.18: Attention weights of the BiLSTM & features' Cross-Attention model for u04 and t24011100
Ground-truth labels: Sitting, Indoors, At home
Predicted labels: Sitting, At main workplace, Indoors, At home, Cooking, Bathing - Shower, Washing dishes, Surfing the internet, Computer work, Eating, Toilet, Grooming, Standing, Phone on table

In Figure 5.6.18, we visualize the Attention weights for the example of user u04 and minute-precision timestamp 24011100. We observe that the features with the largest attention weights are again the smartphone's gyroscope value-entropy magnitude statistic and band4 log energy feature of the magnitude spectrum, and the MFCC std features, and also the ro_xz 3D feature (x-z inter-axis correlation coefficient) from the watch accelerometer signal and two phone state features regarding app state and ringer mode. In this example we notice the paradox of being annotated by three, very generic labels, "Sitting", "Indoors", "At home", while the set of predicted labels is a superset of the predicted ones, also including other more or less possible labels. Some predictions, such as "At main workplace", "Bathing - Shower" and "Standing" are completely odd, since they contradict other predictions, such as "At home" and "Sitting", but this probably happens because we do not use any constraints on joint context label predictions and there might be correlations of specific features with labels contradictory to each other.

Overall, out of the set of predicted labels, excluding the ground-truth labels, all of which are correctly predicted, some labels are absurd and wrongly predicted, such as the aforementioned "At main workplace", "Bathing - Shower" and "Standing" which are contradictory to the ground-truth labels, while some other predicted labels could also be labels for activities that actually took place then, but were unintentionally omitted by the user in the annotation process, such as "Cooking", "Surfing the internet", "Computer work", "Eating" or "Toilet". These predicted features are not contradictory to the ground-truth labels, and there is a possibility that the user performed some of them but forgot to use the corresponding label in the app. This possibility is strengthened when we observe the attention weights, where the large weight of the watch accelerometer's ro_xz 3D feature attributes some importance in arm movement, compatible with some of the aforementioned labels.

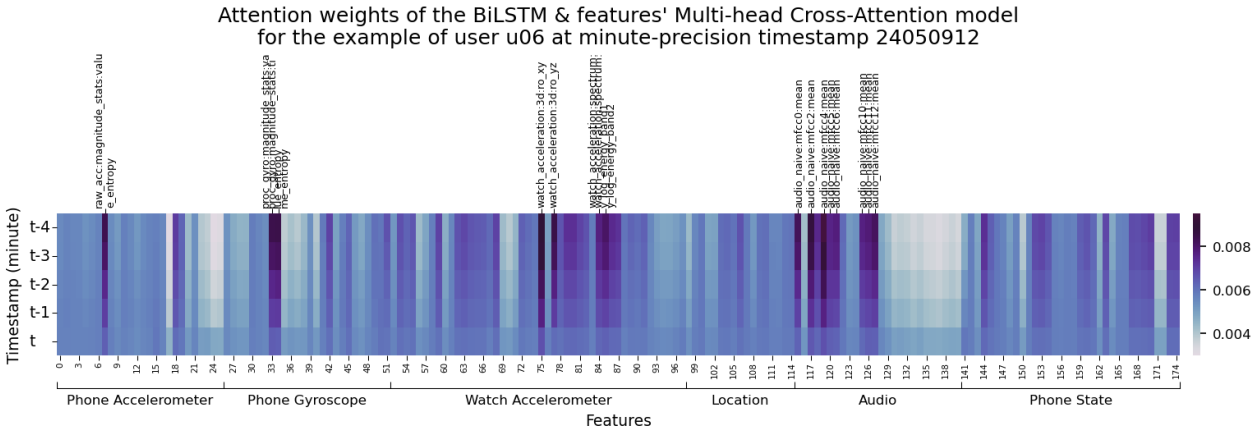


Figure 5.6.19: Attention weights of the BiLSTM & features' Cross-Attention model for u06 and t24050912
Ground-truth labels: Sitting, Indoors, At home, Talking, At school, Phone in hand
Predicted labels: Sitting, In class, In a meeting, At main workplace, Indoors, Phone in pocket, Singing, Talking, Eating, At school, Phone in hand, With co-workers, With friends

In Figure 5.6.19, we visualize the Attention weights for the example of user u06 and minute-precision timestamp 24050912. As we can see, in this example there is an error in the ground-truth labels, since "At home" and "At school" are both used by the user in the annotation process. Similar errors are common in the dataset, and hinder both model training when located in the training set and model evaluation when located in the test set. Regarding features' importance, we observe that a lot of features have comparatively large attention weights, but the most prevalent ones are: the value-entropy magnitude stat of the smartphone accelerometer, the value-entropy and the time-entropy magnitude stats of the smartphone gyroscope, the ro_xy and ro_yz 3D features as well as the band1 and band2 log energy of the y-axis signal spectrum of the watch accelerometer, and the MFCC mean features. Labels "Sitting", "Indoors", "Talking", "At school", "Phone in hand" are correctly predicted. More relevant or not so relevant labels are wrongly predicted, such as "In class", "In a meeting", "With co-workers" and "With friends" that might be predicted because of the MFCC features which capture voices talking, while other labels such as "Eating" might be resulting from hand movement or relevant sounds. Some of the wrongly predicted labels might have been actually unintentionally omitted by the user during annotation, while others, like "Singing", are most probably irrelevant.

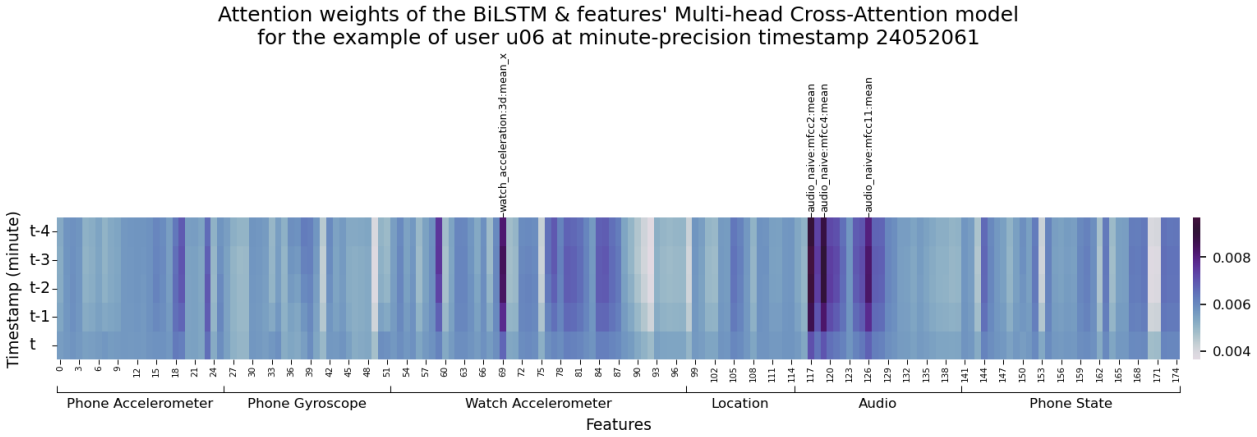


Figure 5.6.20: Attention weights of the BiLSTM & features' Cross-Attention model for u06 and t24052061
Ground-truth labels: Sitting, Eating, At school, Phone on table
Predicted labels: Sitting, Indoors, Watching TV, Surfing the internet, Talking, Eating, At school, Phone in hand

In Figure 5.6.20, we visualize the Attention weights for the example of user u06 and minute-precision timestamp 24052061. The features with the largest attention weights are the mean value of the x-axis signal

of the watch accelerometer, and the mean value of the second, fourth and eleventh MFCC. The ground-truth labels "Sitting", "Eating" and "At school" are predicted by our model, while "Watching TV" and "Talking" are wrongly predicted and might be attributed to relevant sounds captured by the MFCCs, and "Surfing the internet" and "Phone in hand" are also wrongly predicted and might be attributed to relevant arm or hand movement. However, again, it is unclear whether the wrongly predicted labels are actually wrong or were omitted or wrongly labeled by the user.

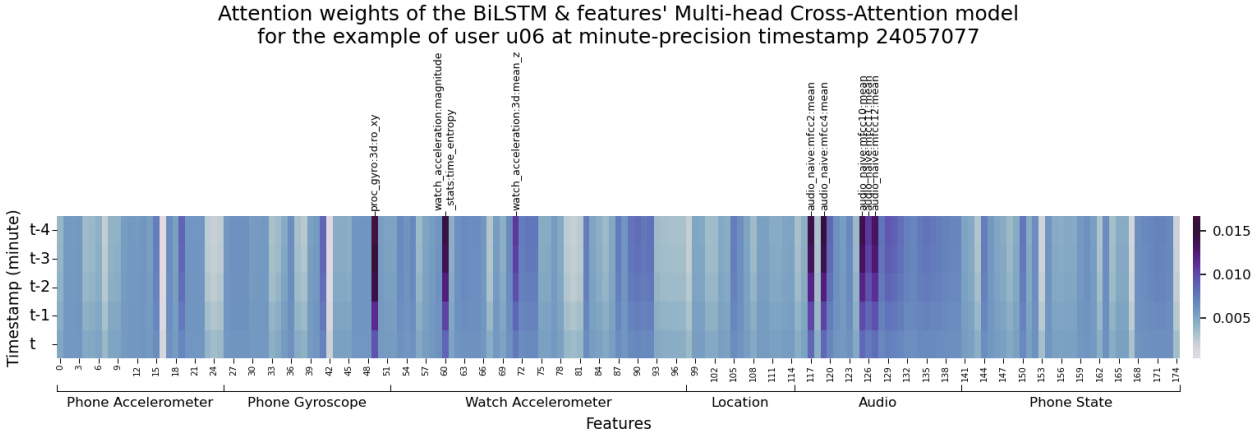


Figure 5.6.21: Attention weights of the BiLSTM & features' Cross-Attention model for u06 and t24057077
Ground-truth labels: Lying down, Sleeping, Indoors, At home, Phone on table
Predicted labels: Lying down, Sleeping, Indoors, At home, Phone on table

In Figure 5.6.21, we visualize the Attention weights for the example of user u06 and minute-precision timestamp 24057077. In this example, we have a rare case of perfect prediction accuracy, since all the true labels were predicted, and also no false labels were predicted for the specific example. We can see that the most prevalent features are the ro_xy 3D feature of the smartphone gyroscope, the time-entropy magnitude stat and the mean value of the z-axis signal of the watch accelerometer, and the mean value of the second, fourth, tenth, eleventh and twelfth MFCC. These features can justify body immobility, phone placement and also at home and sleeping conditions (probably low noise conditions). We should also note that, as we have also seen in the recognition scores' tables for our experiments, for our models it is easier to correctly predict sleeping and sleeping conditions than most other labels, since the sleeping body state is very typical (very low or no movement) and has low variation for a user or between users, compared to most other activity and context labels.

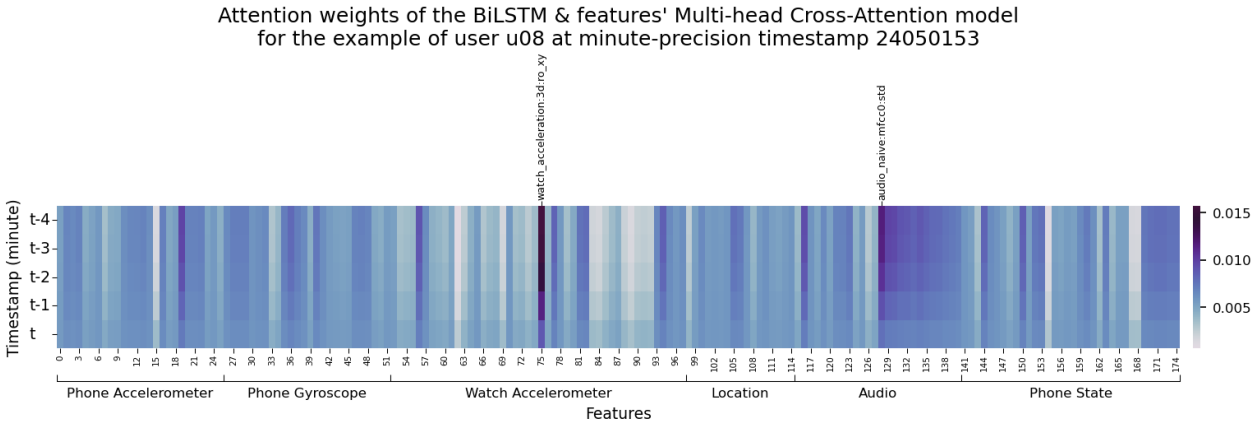


Figure 5.6.22: Attention weights of the BiLSTM & features' Cross-Attention model for u08 and t24050153
Ground-truth labels: Lying down, Sleeping, Indoors, At home, Phone on table
Predicted labels: Lying down, Sleeping, Indoors, At home, Bathing - Shower, Washing dishes, Toilet, Grooming, Dressing, Phone on table

In Figure 5.6.22, we visualize the Attention weights for the example of user u08 and minute-precision timestamp 24050153. This example has the same ground-truth labels as the previous one, centered around sleep, namely "Lying down", "Sleeping", "Indoors", "At home" and "Phone on table", but in the predicted labels, we notice that although all ground-truth labels are correctly predicted, there are also other labels, which are wrongly predicted. These labels are: "Bathing - Shower", "Washing dishes", "Toilet", "Grooming", "Dressing". We also notice that the features that are shown to be given the higher importance by the model are: the ro_xy 3D feature of the watch accelerometer, and the std of MFCC0, while more MFCC std features also have relatively large attention weights. As most wrongly predicted labels potentially include running water, a hypothesis is that a background noise of such type could maybe lead to the false prediction of those labels. Other than that, it is difficult to imagine what type of arm or hand movement would be suited to a person sleeping and also to a person bathing, washing dishes or grooming, to explain the additional, wrongly predicted labels.

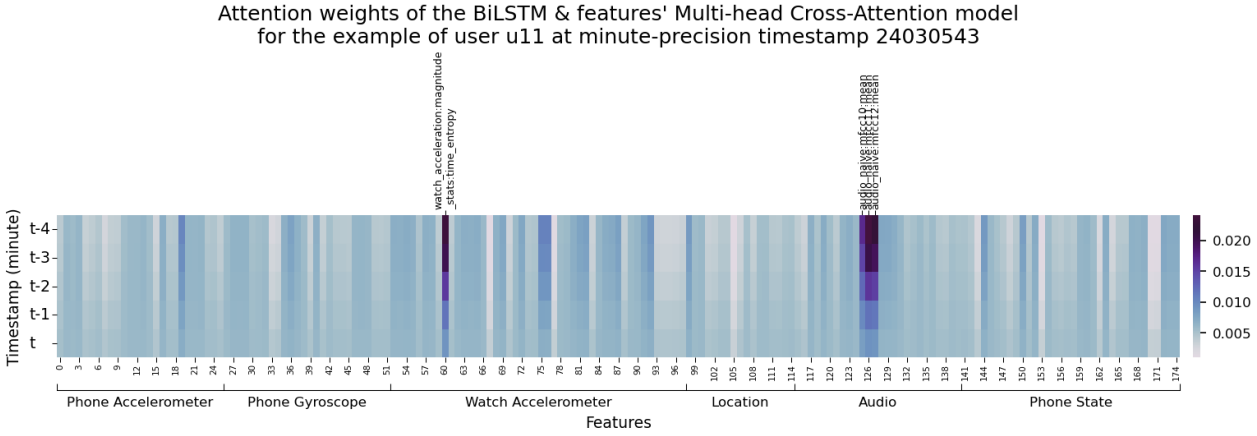


Figure 5.6.23: Attention weights of the BiLSTM & features' Cross-Attention model for u11 and t24030543
Ground-truth labels: Sitting, Lab work, Computer work, At school
Predicted labels: Sitting, Lab work, In class, At main workspace, Indoors, Surfing the internet, Computer work, At school, Phone on table

In Figure 5.6.23, we visualize the Attention weights for the example of user u11 and minute-precision timestamp 24030543. We observe that the features that are given the largest attention weights include the watch accelerometer's time-entropy magnitude stat and the mean value of the tenth, eleventh and twelfth MFCC. All ground-truth labels, namely "Sitting", "Lab work", "Computer work" and "At school" are correctly predicted, and also some other labels are predicted, including "In class", "At main workspace", "Indoors", "Surfing the internet", "Phone on table". Once again, the additional, wrongly predicted labels are somewhat conceptually related to the ground-truth ones, and it is unclear whether some of them were unintentionally omitted during the annotation process. We should also notice that after also cross-checking other examples with the ground-truth label "Lab work", it was observed that in many cases with this label existent, the mean value of the tenth, eleventh and twelfth MFCC features were attributed large attention weights, probably meaning that "Lab work" at least for this dataset's users, in many cases is associated with specific sounds which are recognized by our model.

In Figure 5.6.24, we visualize the Attention weights for the example of user u11 and minute-precision timestamp 24031660. This is a typical example of transportation by car. The ground-truth labels are "Sitting" and "In a car" and they are correctly predicted by our model, along with other labels which are wrongly predicted, some of which are directly related to transportation, like "On a bus", "Drive - Driver" and "Drive - Passenger", while others seem irrelevant, like "At a party" or "At the beach". As expected, the most prevalent features regarding the attention weight values, come from the location sensor, and include the log of latitude range, the log of longitude range and the diameter (maximum distance between two locations in the recording session, in meters), and also the mean value of MFCC1. The location features that have large attention weights, are indicative of the means of transportation, and have discriminative power since the latitude range, longitude range and diameter of an example is substantially different when the user is on a moving car or bus, compared to being on foot. However, we see that the model does not seem to

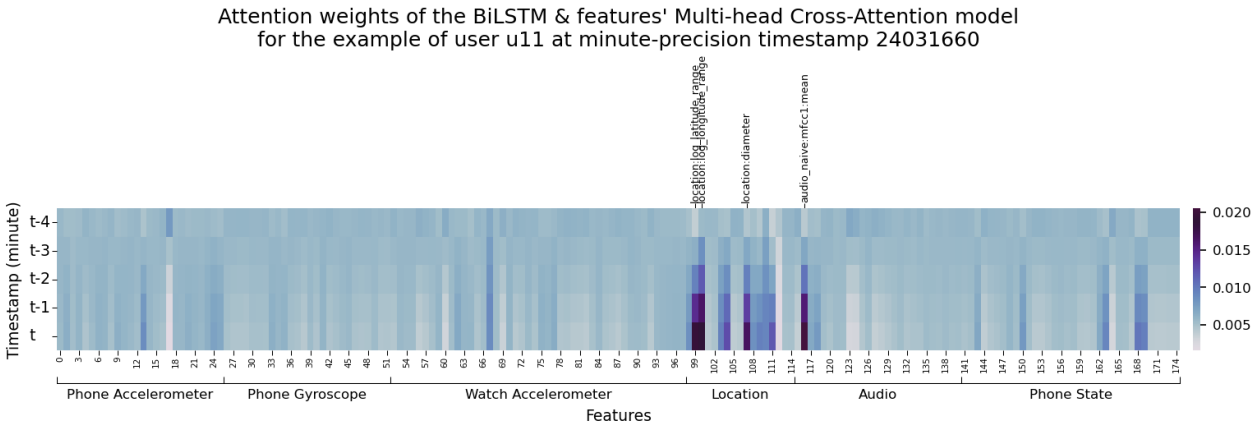


Figure 5.6.24: Attention weights of the BiLSTM & features' Cross-Attention model for u11 and t24031660
Ground-truth labels: Sitting, In a car
Predicted labels: Sitting, Outside, In a car, On a bus, Drive - Driver, Drive - Passenger, Phone in pocket, Shopping, At a party, At the beach, Phone in hand, Phone in bag, With friends

be able to discriminate between "In a car" and "On a bus", and also between "Drive - Driver" and "Drive - Passenger", and all those labels are predicted at the same time. In addition, the phone position, which is not included in the ground-truth labels, is predicted with three labels at the same time, namely "Phone in pocket", "Phone in hand", "Phone in bag", out of which one might be correct, but we cannot know. Also, the MFFC1 mean value feature attributes importance to the background sounds, e.g., the car radio might be on, and that might be the reason other labels which potentially include music are predicted, like "Shopping", "At a party", "At the beach". However, since these activities are perfomed when not being in a moving vehicle, they should not have been predicted. Finally, the label "With friends" could be an actual label that was omitted during annotation, or might be wrongly predicted due to talking or other speech or singing sounds in the background.

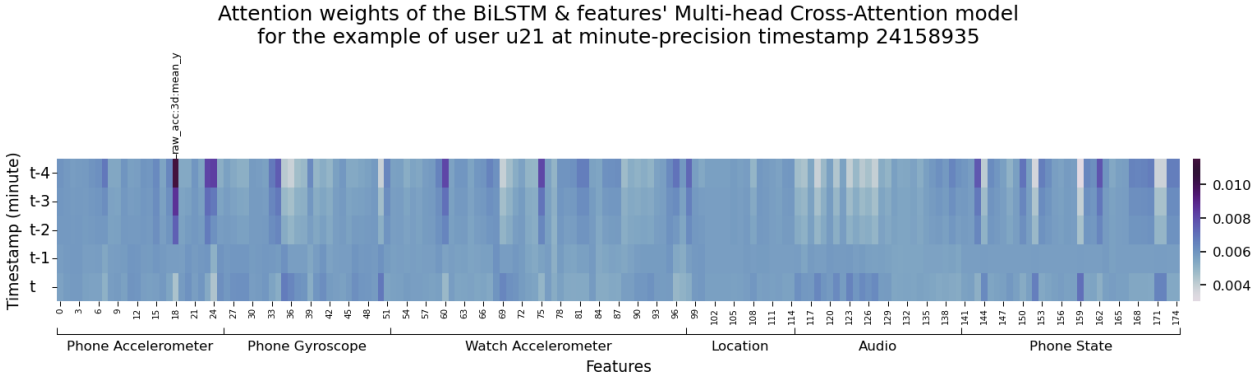


Figure 5.6.25: Attention weights of the BiLSTM & features' Cross-Attention model for u21 and t24158935
Ground-truth labels: Walking, Outside, Talking, With friends
Predicted labels: Sitting, Walking, In class, In a meeting, Outside, Phone in pocket, Talking, Eating, Standing, At school, Phone in hand, With co-workers, With friends

In Figure 5.6.25, we visualize the Attention weights for the example of user u21 and minute-precision timestamp 24158935. The most prevalent feature is the mean value of the y-axis signal of the smartphone accelerometer, which can explain correctly predicting the ground-truth "Walking" label. The remaining ground-truth labels, "Outside", "Talking" and "With friends" are also predicted by our models, although there are no large attention weights in the sound-related features, namely the MFCC. We also see that many other irrelevant labels are wrongly predicted, such as "Sitting" and "Standing" which contradict "Walking", and also "In class" and "In a meeting" which contradict "Outside". In this case the model does not focus in meaningful features for the specific example's context, leading in odd and irrelevant additional predictions.

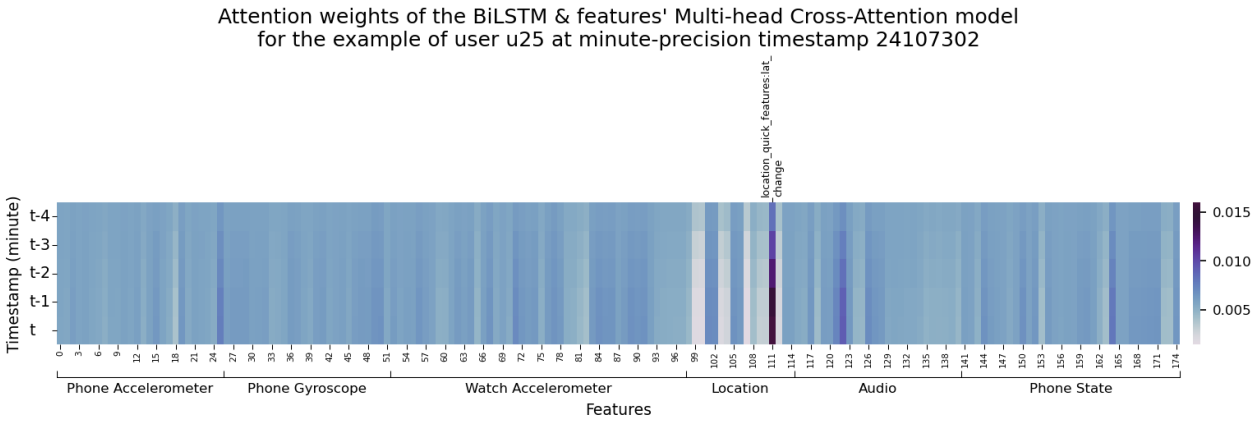


Figure 5.6.26: Attention weights of the BiLSTM & features' Cross-Attention model for u25 and t24107302
Ground-truth labels: Sitting, Drive - Driver, Phone on table
Predicted labels: Sitting, Outside, In a car, On a bus, Drive - Driver, Drive - Passenger, At a restaurant, Drinking alcohol, At a party, At a bar, Phone in hand, Phone in bag, With friends

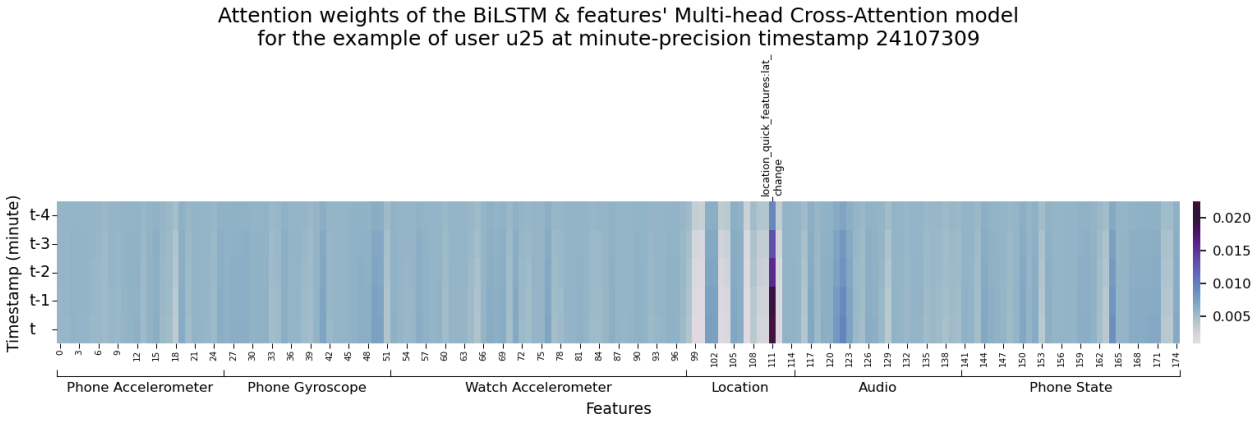


Figure 5.6.27: Attention weights of the BiLSTM & features' Cross-Attention model for u25 and t24107309
Ground-truth labels: Sitting, Drive - Driver, Phone on table
Predicted labels: Sitting, Outside, In a car, On a bus, Drive - Driver, Drive - Passenger, Phone in hand, Phone in bag, With friends

In Figure 5.6.26 and Figure 5.6.27, we visualize the Attention weights for the examples of user u25 and minute-precision timestamps 24107302 and 24107309, respectively. In essence, these two examples are seven minutes apart, and capture the same activity, driving, based on the same ground-truth labels "Sitting", "Drive - Driver", "Phone on table" of the two examples. The two examples also share a common subset of predicted labels, including "Sitting", "Outside", "In a car", "On a bus", "Drive - Driver", "Drive - Passenger", "Phone in hand", "Phone in bag" and "With friends". As previously mentioned in the commentary of Figure 5.6.24, in these cases the model mainly focuses on features from the location sensor, which can indicate being in a moving vehicle, but these features are not so useful to discriminate between "In a car" and "On a bus", and also between "Drive - Driver" and "Drive - Passenger". In Figures 5.6.26 and 5.6.27 the main focus is on the latitude change (latitude last value minus first value of the minute's recording session) location quick feature.

When comparing the two plots, we also notice a slight difference regarding the MFCC weights. It seems that in Figure 5.6.26, the model pays more attention to some MFCC mean features, compared to Figure 5.6.27, which might explain why in the first case, the model also predicts labels such as "At a restaurant", "Drinking alcohol", "At a party" and "At a bar". However, the difference in the attention weights is small, so our assumptions are uncertain and inconclusive.

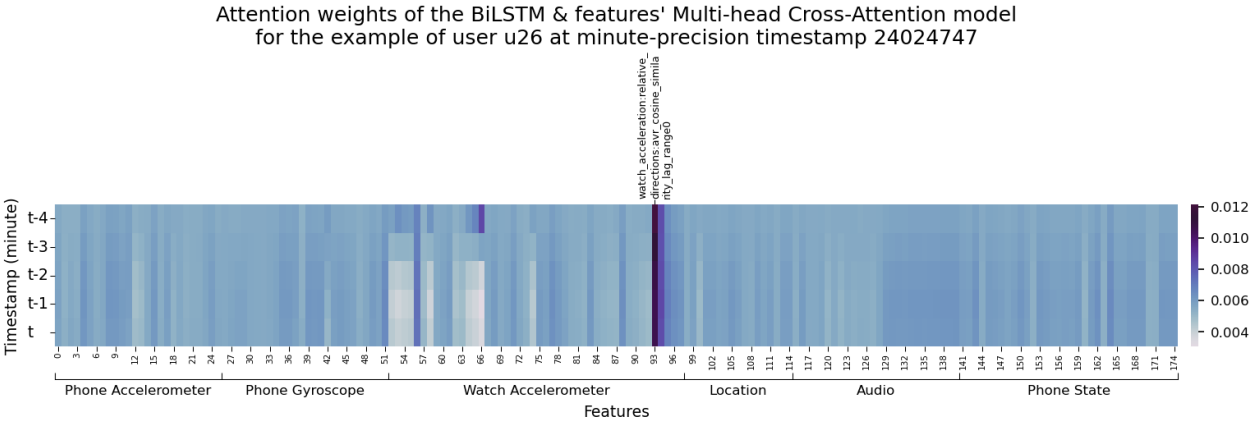


Figure 5.6.28: Attention weights of the BiLSTM & features' Cross-Attention model for u26 and t24024747
Ground-truth labels: Bicycling, Exercise
Predicted labels: Walking, Running, Bicycling, Outside, On a bus, Phone in pocket, Exercise, Shopping, Strolling, Singing, At the gym, Stairs - Going up, Elevator, Phone in bag

In Figure 5.6.28, we visualize the Attention weights for the example of user u26 and minute-precision timestamp 24024747. We observe that the feature with the largest attention weight is the `avr_cosine_similarity_lag_range0` relative-direction feature of the watch accelerometer, which is calculated as follows: first the cosine-similarity between the acceleration directions of any two time points in the time series is calculated (value of 1 meaning same direction, value of -1 meaning opposite directions and value of 0 meaning orthogonal directions) and then the cosine similarity values in the 0-0.5sec range of time-lag between the compared time points are averaged. After cross-checking other examples with the ground-truth label "Bicycling", we observe that it is common for the model to attribute large attention weights to the specific feature, which is related to rapid changes in watch orientation. We notice that the `avr_cosine_similarity_lag_range1` relative-direction feature of the watch accelerometer (averaging cosine similarity values in the 0.5-1sec range of time-lag between the compared time points) is also given relatively large attention weight. However, although all ground-truth labels, namely "Bicycling" and "Exercise", are predicted, the model also predicts a lot of other labels. Some of them are contextually relevant, such as "At the gym", while others are wrongly predicted, such as "Running", "Shopping", "Strolling" and "Elevator", which include different types of movement.

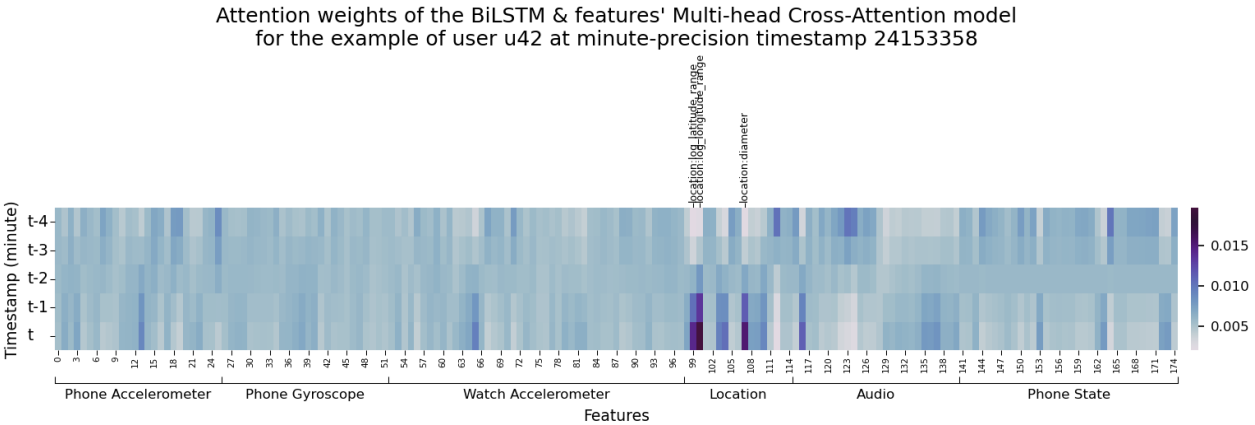


Figure 5.6.29: Attention weights of the BiLSTM & features' Cross-Attention model for u42 and t24153358
Ground-truth labels: Sitting, Drive - Driver
Predicted labels: Sitting, Outside, In a car, On a bus, Drive - Driver, Drive - Passenger, Shopping, At a party, Phone in bag, With friends

In Figure 5.6.29, we visualize the Attention weights for the example of user u42 and minute-precision timestamp 24153358. This is another case of being in a moving vehicle and driving, and the largest attention weights are given to features of the location sensor: log of latitude range, log of longitude range, and diameter. The features with the largest attention weights as well as the predictions and prediction errors of the model are similar to the ones of the example of Figure 5.6.24, although in the current example, we can see that less attention is paid to the MFCC1 mean feature.

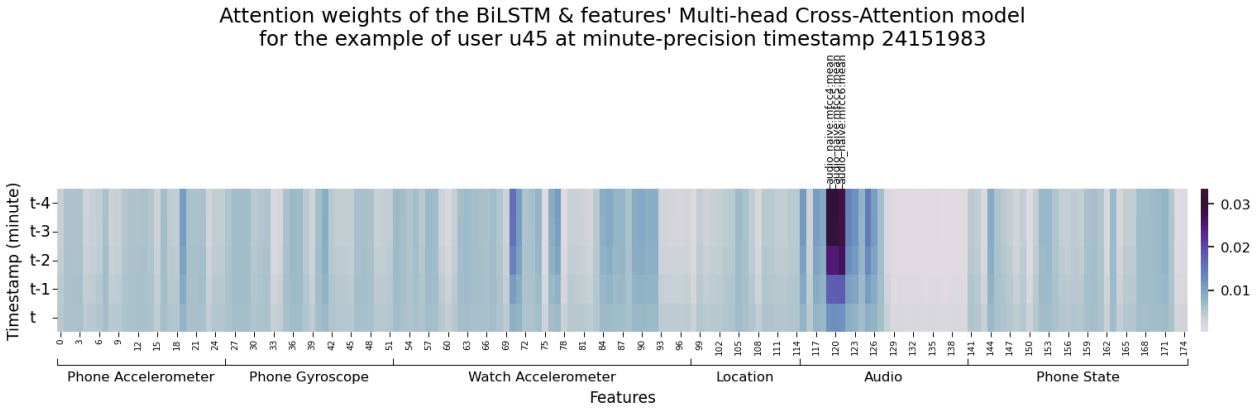


Figure 5.6.30: Attention weights of the BiLSTM & features' Cross-Attention model for u42 and t24151983
Ground-truth labels: Sitting, Indoors, At home, Watching TV, Talking, Phone on table
Predicted labels: Sitting, Indoors, At home, Watching TV, Talking, Eating, Phone on table, With friends

In Figure 5.6.30, we visualize the Attention weights for the example of user u45 and minute-precision timestamp 24151983. The most prevalent features are the mean values of the fourth, fifth and sixth MFCC. The model predicts all ground-truth labels: "Sitting", "Indoors", "At home", "Watching TV", "Talking" and "Phone on table", and also two other relevant labels, "Eating" and "With friends".

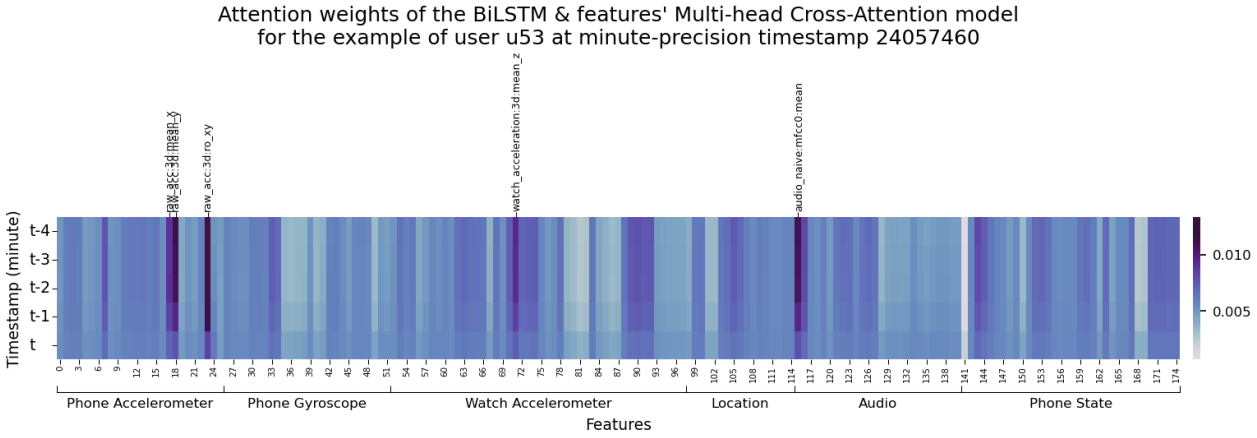


Figure 5.6.31: Attention weights of the BiLSTM & features' Cross-Attention model for u53 and t24057460
Ground-truth labels: Indoors, At home, Grooming, Dressing, Standing
Predicted labels: Indoors, At home, Cooking, Bathing - Shower, Cleaning, Doing laundry, Washing dishes, Toilet, Grooming, Dressing, Standing, Phone on table

In Figure 5.6.31, we visualize the Attention weights for the example of user u53 and minute-precision timestamp 24057460. In this example centered on self-care, the features with the largest attention weights are: the mean of the x-axis signal, the mean of the y-axis signal and the ro_xy 3D feature of the smartphone's accelerometer, the mean of the z-axis signal of the watch accelerometer, and the mean of MFCC0. The model predicts all ground-truth labels, "Indoors", "At home", "Grooming", "Dressing" and "Standing". However, the model also wrongly predicts many more labels for house activities with similar movements, including "Cooking", "Bathing - Shower", "Cleaning", "Doing laundry" and "Washing dishes".

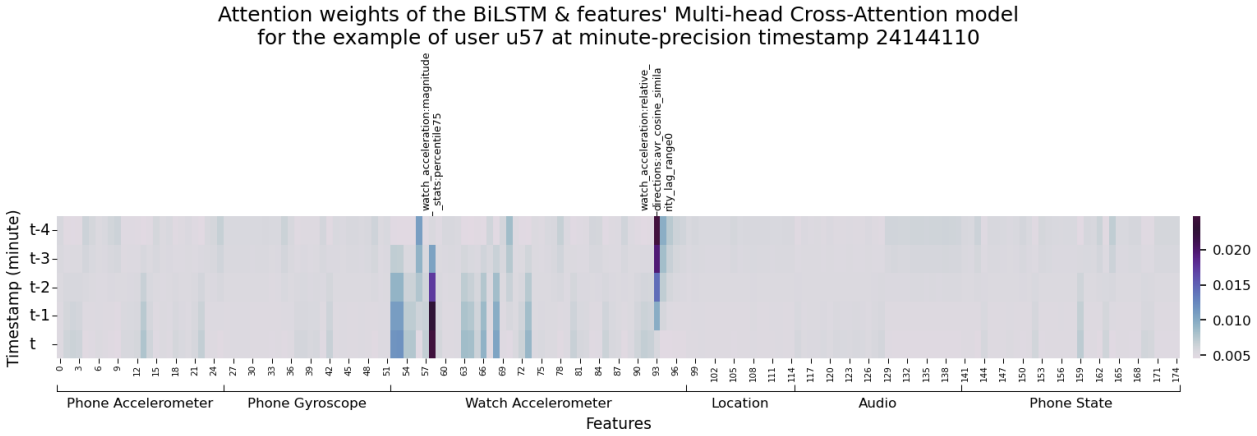


Figure 5.6.32: Attention weights of the BiLSTM & features' Cross-Attention model for u57 and t24144110
Ground-truth labels: Running, Indoors, Exercise, At the gym, Phone on table
Predicted labels: Walking, Running, Bicycling, Outside, Phone in pocket, Exercise, Shopping, Strolling, Doing laundry, At the beach, Singing, Stairs - Going up, Stairs - Going down, Elevator, Phone in hand, With friends

In Figure 5.6.32, we visualize the Attention weights for the example of user u57 and minute-precision timestamp 24144110. This is a rare example where only a few features have considerable attention weights, and the model focuses mainly on the 75th percentile magnitude stat and the `avr_cosine_similarity_lag_range0` relative-direction feature of the watch accelerometer. The model correctly predicts only two ground-truth labels, "Running" and "Exercise", and also predicts over ten wrong labels, some of which, such as "Outside", "At the beach" and "Phone in hand", directly contradict the ground-truth labels. When paying attention to only a very small subset of the input features, the model is not able to make sound decisions regarding activity and context labels, and as it can be seen in this example, most labels reflecting physical activity are predicted.

5.6.4 Other Model Architectures

In our efforts to integrate a Self-Attention or Cross-Attention module in the BiLSTM model to improve its performance, except for the aforementioned successful experiments including either a Self-Attention module preceding the BiLSTM whose final hidden states are used in the output Linear layer (Subsection 5.6.1), or a BiLSTM whose full output for all timesteps is used to produce the query in the Cross-Attention module that follows, with the input features used to produce key and value (Subsection 5.6.2), we have also tried other model architectures. They can be summarized as follows:

- Including a Self-Attention module preceding the BiLSTM whose full output for all timesteps is used in the output Linear layer.
- Including a Cross-Attention module following the BiLSTM, whose full output for all timesteps is used to produce the query in the Cross-Attention, while the model's input sequence is used to produce key and value, and the Attention's output is fed to the output Linear layer.
- Including a Cross-Attention module following the BiLSTM, whose full output for all timesteps is used to produce the query in the Cross-Attention, while the model's input sequence is used to produce key and value, and the Attention's output is fed to another BiLSTM and its output is fed to the output Linear layer.
- Including a Cross-Attention module following the BiLSTM, where the final hidden states of the BiLSTM are used to produce the query in the Cross-Attention, and the full output for all timesteps of the BiLSTM is used to produce key and value, and the Attention's output is fed to the output Linear layer.

According to our experiments, after training and evaluating these models, and experimenting with different hyperparameter values, the performance of these models on the ExtraSensory dataset, using five-fold cross

validation as in all our previous experiments, was similar to or worse than the performance of the simple BiLSTM models of Section 5.5, based on the recognition metrics. Thus, for reasons of brevity and simplicity, we won't provide more information or detailed tables of results regarding these models.

5.7 Deep Learning Feature Extraction using Raw Sensor Measurements

In all our previous experiments, we have used the pre-computed sensor features provided in the ExtraSensory dataset. In this Section, we attempt to leverage information directly from the collected sensor signals, by using Deep Learning models to extract meaningful representations of them, instead of using the hand-crafted features, as we did previously. In these experiments we again use the aforementioned "Core" subset of the dataset, which includes all examples that have measurements from all six core sensors (Acc, Gyro, WAcc, Loc & Loc QF, Aud, PS), totaling 169,001 examples. More specifically, instead of using the 175 hand-crafted features as we did in our previous experiments, we now use:

- The hand-crafted features for Loc & Loc QF (17 features) for privacy reasons, and also because there is no standard or obvious way to handle Location map coordinates as input to a neural network
- The hand-crafted features for PS (34 features) since the Phone State sensors have a single measurement for each example by nature, and the hand-crafted one-hot feature vectors are the intuitive way to use them
- The raw sensor signal for Acc, which is a time series of 3-axis vectors of acceleration along the standard axes of the smartphone, which is sampled at 40Hz during the 20sec recording window of each minute, producing a time series of approximately 800 time points
- The calibrated signal for Gyro, which is a time series of 3-axis vectors of rotation rate around each of the phone's standard axes, which is sampled at 40Hz during the 20sec recording window of each minute, producing a time series of approximately 800 time points (the calibrated version of the gyroscope signal, in which the estimated drift effects have been removed, is provided in the dataset)
- The raw sensor signal for WAcc, which is a time series of 3-axis vectors of acceleration along the standard axes of the smartwatch, which is sampled at 25Hz during the 20sec recording window of each minute, producing a time series of approximately 500 time points
- The MFCCs for audio (which was recorded at 22,050Hz), which are calculated using half-overlapping windows of 2048 samples, 40 Mel-scaled frequency bands and 13 cepstral coefficients including the 0th coefficient (raw audio or Mel spectrograms are not provided in the dataset for privacy reasons)

However, although we use the "Core" subset of the dataset which contains examples with sensor measurements for all six core sensors, since the dataset is collected *in-the-wild* there is no consistency in the lengths of time series of each sensor. We summarize some statistics regarding the lengths of the time series of each sensor's measurements in Table 5.22. These differences have occurred not because of differences in the sampling rate, but because of differences in the duration of the sensor recording window among examples of a sensor.

Statistics of the time-series lengths for each sensor					
Sensor	**Median**	**Mean**	**SD**	**Min**	**Max**
Acc (raw_acc)	800	800	0	800	800
Gyro (proc_gyro)	800	781	67.01	2	800
WAcc (watch_acc)	450	396	120.12	25	500
Aud (audio)	428	396	75.36	3	4894

Table 5.22: Statistics of the time-series lengths for each sensor, for the examples of the "Core" subset

In order to be able to use the raw sensor measurements with models that need consistency in the input dimensions, e.g., CNNs which need input of standard dimensions, we devise a strategy to create time series of equal length for all examples, for each sensor. We decide on the desired lengths of 800 for Acc and Gyro, and 500 for WAcc (which are in accordance with the expected lengths, according to the sampling rate of

each sensor) and 700 for Aud. This way, we have an input example size of (3, 800) for Acc and Gyro, (3, 500) for WAcc, and (13, 700) for Aud. For examples where the recorded sensor measurements have shorter length, we repeat them from the beginning, once or as many times as needed to reach the desired lengths (also, regarding Aud, in cases where the MFCC time series is longer than 700 timesteps, we only use the 700 first ones). This type of padding might introduce some artifacts in the signals, but it seems to be a better solution than zero padding or repeating only the last available sensor values.

5.7.1 CNN-based Feature Extraction for IMU and Audio Data

At first, we implement a CNN-based pipeline for feature extraction from the raw sensor signals (Acc, Gyro, WAcc, Aud), followed by an MLP with two hidden layers and an output layer, similar to the baseline MLP in Subsection 5.4.4. In essence, we want to replicate the baseline MLP experiment, but instead of using the pre-extracted features provided by the dataset as input, CNN layers are used to extract meaningful features from the sensor signals (for Loc and PS, the pre-extracted features are concatenated to the CNN-extracted features of the other sensors, before passing through the MLP). This model is conceptually presented in Figure 5.7.1, and the key points of the experiments are summarized as follows:

- Five-fold cross validation is used, with each fold containing 48 users in the training set and 12 users in the test set (using the same partition as the original work).
- The training set of each CV iteration is further partitioned in a training subset and a validation subset, as follows: for each of the 48 users of the training set, 80% of their data is used for training, and 20% is used for validation, to decide which epoch's checkpoint will be selected for the test inference.
- Regarding Loc and PS, for which we use the pre-extracted features provided in the dataset, we standardize each sensor feature (mean and standard deviation are calculated using the training subset). After standardization, missing feature values are zero-imputated. Regarding the rest of the sensors, for which we use the raw sensor measurements, we standardize the measurements of each sensor (mean and standard deviation are also calculated using the training subset).
- The model configuration includes a CNN pipeline for each of the raw signals' sensors (Acc, Gyro, WAcc, Aud) to extract features from each sensor measurements' time-series. The CNN pipeline for each of these sensors includes multiple layers that contain the following operations, in this order: 2D/1D Conv layer, leaky ReLU, Batch Normalization, and Dropout. The full CNN pipelines' configuration hyperparameters can be found in Table 5.23.
- The outputs of all sensors' CNN pipelines are concatenated, and also concatenated with the pre-extracted features from Loc and PS, and then they are fed to the baseline MLP configuration which includes two hidden layers, followed by an output layer of 51 nodes which correspond to the 51 context labels. Each of the hidden layers is followed by leaky ReLU activation, Batch Normalization, and Dropout, as described in Subsection 5.4.4.
- We experiment with two hyperparameters: the CNN layers' No. of filters, denoted as `out_channels`, and the hidden Linear layers' size, denoted as `out_features`. Regarding `out_channels`, we use the same number in all CNN layers of each sensor. Given that the number of `out_channels` for each CNN layer of the Acc and Gyro CNN pipelines is c_1 and the respective number of the WAcc and Aud CNN pipelines is c_2, we define the search space as $(c_1, c_2) \in \{(16, 32), (32, 64), (64, 128)\}$. Regarding `out_features`, we experiment using sizes of 8, 16, and 32. The results of our investigation are thoroughly presented in Table 5.24, along with the number of parameters for each model.
- Regarding the loss, we again use the custom loss that was first described in Subsection 5.4.4, based on `torch.nn.BCEWithLogitsLoss`. This loss is slightly modified to allow dynamic per-batch, per-element masking in the loss matrices (to mask the loss elements corresponding to missing ground-truth labels for each batch, and not use them in the loss computation). Also, the `pos_weight` is used for instance-weighting to account for the imbalance in the number of positive examples per context label, by multiplying the term of the positive examples in the loss, with the ratio of negative to positive examples for this label in the training set.
- We use a `batch size` of 32 to train the model.

- We use the Adam optimizer to train the model. The `learning rate` hyperparameter is adapted to get the best results. The optimal values of `learning rate` for each combination of CNN layers' No. of filters (`out_channels`) and hidden Linear layers' size (`out_features`), as determined by our experiments, are denoted in Table 5.24.
- In the testing phase, the continuous predictions of the model pass through a sigmoid activation function, and then they are converted to binary outputs using a threshold of 0.5.
- We calculate the evaluation metrics for each label over its non-missing ground-truth examples.

We use `PyTorch` to build the model, the supervisor and the training scripts. The `torch.nn` modules and `torch` functions that are used, can be seen in detail in Figure 5.7.2.

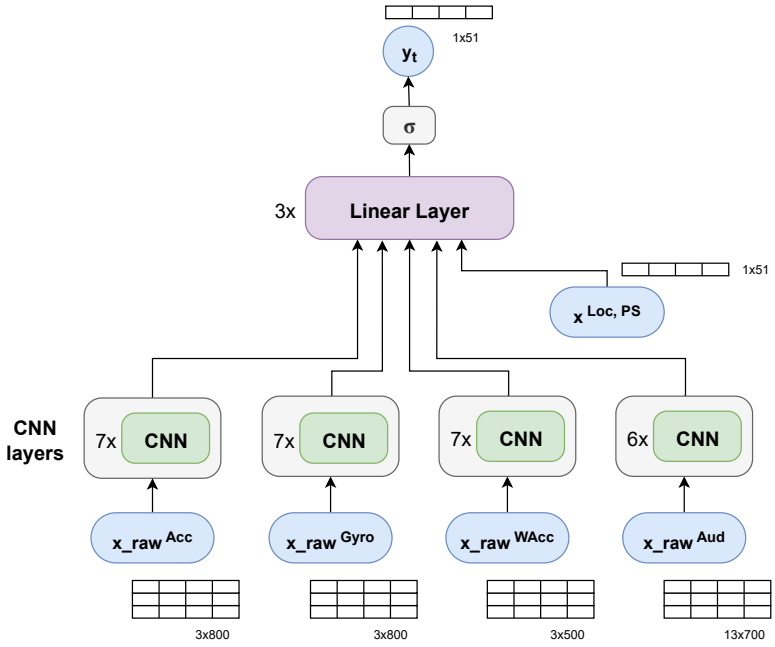


Figure 5.7.1: CNN-based model for feature extraction from sensor measurements, presented conceptually. For raw sensor measurements for a single example from Acc, Gyro, WAcc and Aud sensors, CNN layers are used to extract features. The outputs of all sensors' CNN pipelines are concatenated, and also concatenated with the pre-extracted features from Loc and PS, and then they are fed to an MLP that includes two hidden layers and an output layer, followed by a sigmoid function to convert the model's outputs to labels' probabilities.

In Table 5.24 we present the recognition scores for the CNN-based model using raw sensor measurements for Acc, Gyro, WAcc and Aud, averaged for all labels, for different values of the hyperparameters: CNN layers' `out_channels` $(c_1, c_2) \in \{(16, 32), (32, 64), (64, 128)\}$ and hidden Linear layers' size `out_features` $= \{8, 16, 32\}$. We notice that regardless of the values of the specific hyperparameters we have experimented on, the BA of the model is consistently around 0.760-0.770. We get the highest BA of 0.772 with model configuration `out_channels` $(c_1, c_2) = (32, 64)$, `out_features` $= 16$, and with model configuration `out_channels` $(c_1, c_2) = (64, 128)$, `out_features` $= 16$. As the best-performing model, we select the first of the two since it provides the best BA with far less model parameters, compared to the second one. By observing the Table, we can see that increasing the CNN layers' `out_channels` with constant hidden Linear layers' size, or increasing the hidden Linear layers' size with constant CNN layers' `out_channels`, lead to an increase in Specificity and a decrease in Sensitivity, and we achieve the highest BA when the trade-off between Sensitivity and Specificity is balanced. Overall, we notice that increasing the CNN layers' size or the MLP hidden layers size does not improve the resulting BA of the model in general, and that, regardless of how much we have tried to fine-tune other hyperparameters of the model, mostly the CNN layers' number, kernel size, stride etc., we did not achieve to get higher recognition BA.

CNN model hyperparameters per sensor
No. of filters search space: $(c_1, c_2) \in \{(16, 32), (32, 64), (64, 128)\}$

Smartphone accelerometer (**Acc**)

Input shape: $(B, 1, 3, 800)$

Layer type	No. of filters	Kernel size	Stride	Normalization	Activation	Dropout
Conv2d	c_1	(1, 5)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 15)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(3, 5)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 15)	3	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 15)	3	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 31)	3	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 17)	1	Batch	LeakyReLU (0.1)	0.2

Output shape: $(B, c_1, 1, 1)$

Smartphone gyroscope (**Gyro**)

Input shape: $(B, 1, 3, 800)$

Layer type	No. of filters	Kernel size	Stride	Normalization	Activation	Dropout
Conv2d	c_1	(1, 5)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 15)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(3, 5)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 15)	3	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 15)	3	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 31)	3	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_1	(1, 17)	1	Batch	LeakyReLU (0.1)	0.2

Output shape: $(B, c_1, 1, 1)$

Smartwatch accelerometer (**WAcc**)

Input shape: $(B, 1, 3, 500)$

Layer type	No. of filters	Kernel size	Stride	Normalization	Activation	Dropout
Conv2d	c_2	(1, 5)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(1, 15)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(3, 5)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(1, 15)	3	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(1, 15)	2	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(1, 31)	2	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(1, 21)	1	Batch	LeakyReLU (0.1)	0.2

Output shape: $(B, c_2, 1, 1)$

Smartphone audio (**Aud**)

Input shape: $(B, 1, 13, 700)$

Layer type	No. of filters	Kernel size	Stride	Normalization	Activation	Dropout
Conv2d	c_2	(3, 3)	1	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(3, 5)	2	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(3, 9)	2	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(1, 15)	3	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(1, 15)	3	Batch	LeakyReLU (0.1)	0.2
Conv2d	c_2	(1, 13)	1	Batch	LeakyReLU (0.1)	0.2

Output shape: $(B, c_2, 1, 1)$

Table 5.23: CNN model architecture hyperparameters for Acc, Gyro, WAcc and Audio CNN encoders and search space regarding CNN layers' No. of filters (`out_channels`)

Performance metrics averaged over all labels for the CNN model using raw sensor measurements for Acc, Gyro, WAcc and Aud Hyperparameter tuning: No. of CNN layers' output channels & hidden Linear layers' size								
Hidden size	**No. of parameters**	**Learning rate**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
No. of CNN layers' output channels: Acc: 16, Gyro: 16, WAcc: 32, Aud: 32								
8	261299	0.001	0.772	0.222	0.760	0.773	0.288	0.767
16	263123	0.001	0.774	0.222	0.761	0.775	0.289	0.768
32	267155	0.001	0.803	0.231	0.722	0.803	0.301	0.762
No. of CNN layers' output channels: Acc: 32, Gyro: 32, WAcc: 64, Aud: 64								
8	1035667	0.001	0.768	0.220	0.769	0.767	0.287	0.768
16	1038259	0.0005	0.782	0.228	0.762	0.781	0.296	0.772
32	1043827	0.0005	0.802	0.232	0.728	0.803	0.303	0.766
No. of CNN layers' output channels: Acc: 64, Gyro: 64, WAcc: 128, Aud: 128								
8	4126547	0.001	0.760	0.218	0.763	0.758	0.284	0.760
16	4130675	0.0005	0.790	0.230	0.754	0.790	0.299	0.772
32	4139315	0.001	0.801	0.232	0.727	0.801	0.303	0.764

Table 5.24: Recognition scores for the CNN model using raw sensor measurements for Acc, Gyro, WAcc and Aud, averaged for all labels, for different values of the hyperparameters: CNN layers' `out_channels` and hidden Linear layers' size `out_features`. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

For the best-performing model configuration, with CNN layers' `out_channels` $(c_1, c_2) = (32, 64)$ and hidden Linear layers' size `out_features` $= 16$, we present the detailed model architecture in Figure 5.7.2, and the recognition scores for all context labels in Table 5.26 and averaged over the labels of each label subset in Table 5.25. As we can see, our CNN-based model outperforms similar models in the literature [Sae+19; Cru+20a] which use CNN-based models for feature extraction from ExtraSensory dataset's raw sensor measurements, reporting BA scores of up to 0.750. However, the BA of 0.772 that our model achieves, is on par with the BA score of the baseline MLP(16,16) using the hand-crafted features of Subsection 4.3.6, which means that the inclusion of raw sensor data from Acc, Gyro, WAcc and Aud in our model via CNN-based feature extraction did not lead to an improvement in BA. Moreover, we can see that our CNN-based model performs on par with the baseline MLP(16,16) in all averaged recognition metrics, including Accuracy, Sensitivity and Specificity. Furthermore, we notice that the per-label BA of both models is similar in most cases.

Recognition scores of the CNN model, averaged for each label subset						
Label	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Posture/Movement	0.797	0.374	0.799	0.790	0.451	0.795
Special Movement	0.793	0.027	0.737	0.794	0.051	0.765
Phone Location	0.757	0.446	0.792	0.755	0.525	0.774
Work-related	0.787	0.243	0.816	0.783	0.362	0.799
Location-based	0.793	0.264	0.821	0.789	0.332	0.805
Transportation	0.835	0.141	0.890	0.834	0.235	0.862
Chores	0.774	0.039	0.682	0.776	0.074	0.729
Self-care	0.826	0.190	0.653	0.831	0.224	0.742
Leisure Time	0.709	0.140	0.700	0.708	0.227	0.704
Companion	0.689	0.181	0.621	0.702	0.274	0.662
Environment	0.857	0.667	0.878	0.873	0.708	0.875
Average	0.782	0.228	0.762	0.781	0.296	0.772

Table 5.25: Recognition scores reported for the CNN model, averaged for each label subset. Model configuration using CNN layers' `out_channels = 32` for Acc, Gyro and `64` for WAcc, Aud. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

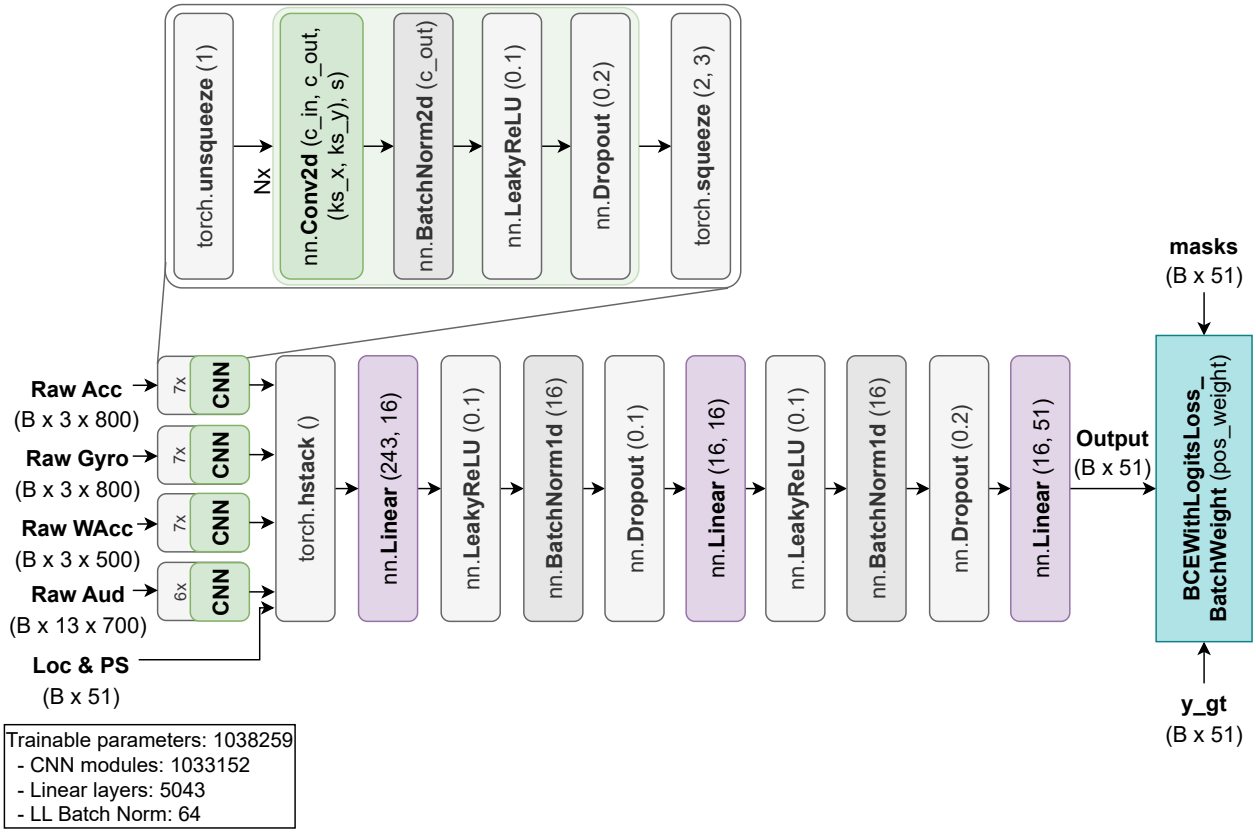


Figure 5.7.2: The selected CNN model architecture, using `out_channels = 32` for Acc, Gyro and 64 for WAcc, Aud. For raw sensor measurements for a single example, a feature extraction pipeline including CNN layers is used. The hyperparameters for each CNN layer are thoroughly listed in Table 5.23. The outputs of all sensors' CNN pipelines are concatenated, and also concatenated with the pre-extracted features from Loc and PS, and they are fed to the baseline MLP(16,16) architecture of Subsection 4.3.6. The aforementioned custom loss based on `torch.nn.BCEWithLogitsLoss` is used to train the model.

Recognition scores of the CNN model, for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.888	0.822	0.815	0.921	0.819	0.868
Sitting	79368	0.762	0.714	0.823	0.708	0.765	0.765
Standing	22071	0.627	0.225	0.756	0.608	0.346	0.682
Walking	11715	0.788	0.220	0.809	0.787	0.347	0.798
Running	661	0.853	0.034	0.694	0.854	0.065	0.774
Bicycling	3504	0.866	0.230	0.899	0.864	0.367	0.881
Strolling	339	0.760	0.048	0.829	0.759	0.092	0.794
Stairs - Going up	399	0.817	0.024	0.639	0.819	0.047	0.729
Stairs - Going down	390	0.819	0.024	0.651	0.820	0.046	0.736
Elevator	123	0.777	0.010	0.829	0.777	0.020	0.803
Phone in pocket	14074	0.769	0.428	0.852	0.750	0.570	0.801
Phone in hand	7313	0.654	0.168	0.730	0.646	0.273	0.688
Phone in bag	5031	0.795	0.267	0.781	0.797	0.398	0.789
Phone on table	65979	0.812	0.921	0.806	0.825	0.860	0.816
In class	2852	0.825	0.214	0.772	0.828	0.335	0.800
Lab work	2898	0.767	0.310	0.913	0.749	0.463	0.831
Computer work	22536	0.745	0.354	0.741	0.745	0.479	0.743
In a meeting	2837	0.811	0.095	0.836	0.811	0.170	0.823
At home	80044	0.795	0.768	0.818	0.773	0.793	0.796
At school	25342	0.753	0.404	0.767	0.750	0.529	0.758
At main workplace	19235	0.850	0.588	0.857	0.849	0.697	0.853
At a restaurant	1275	0.833	0.072	0.864	0.833	0.133	0.849
At a bar	520	0.803	0.114	0.952	0.799	0.203	0.876
At a party	404	0.680	0.041	0.827	0.678	0.077	0.752
At the gym	897	0.837	0.108	0.670	0.842	0.185	0.756
At the beach	116	0.789	0.019	0.810	0.789	0.037	0.799
In a car	3550	0.843	0.164	0.867	0.842	0.276	0.855
On a bus	1179	0.815	0.058	0.906	0.814	0.109	0.860
Drive - Driver	4879	0.863	0.258	0.882	0.862	0.399	0.872
Drive - Passenger	1650	0.821	0.085	0.904	0.820	0.156	0.862
Shopping	809	0.769	0.036	0.865	0.768	0.070	0.816
Cooking	2212	0.778	0.056	0.722	0.779	0.104	0.751
Cleaning	1813	0.750	0.047	0.569	0.754	0.086	0.662
Doing laundry	471	0.760	0.021	0.567	0.762	0.040	0.664
Washing dishes	829	0.815	0.036	0.688	0.816	0.068	0.752
Bathing - Shower	1120	0.823	0.031	0.563	0.826	0.059	0.695
Toilet	1558	0.762	0.032	0.621	0.764	0.061	0.693
Grooming	1775	0.813	0.055	0.624	0.816	0.101	0.720
Dressing	1248	0.822	0.036	0.583	0.825	0.067	0.704
Sleeping	40869	0.909	0.794	0.872	0.922	0.831	0.897
Exercise	5191	0.843	0.162	0.796	0.845	0.270	0.820
Eating	9668	0.633	0.109	0.710	0.628	0.188	0.669
Drinking alcohol	859	0.797	0.083	0.860	0.796	0.152	0.828
Watching TV	8945	0.749	0.208	0.681	0.756	0.318	0.718
Surfing the internet	10668	0.683	0.182	0.597	0.693	0.279	0.645
Talking	18477	0.603	0.201	0.753	0.582	0.317	0.667
Singing	384	0.651	0.034	0.500	0.655	0.064	0.577
With co-workers	3972	0.712	0.147	0.734	0.711	0.244	0.722
With friends	12686	0.667	0.216	0.508	0.693	0.303	0.601
Indoors	102510	0.863	0.992	0.860	0.899	0.922	0.880
Outside	6793	0.850	0.341	0.896	0.846	0.494	0.871
Average		0.782	0.228	0.762	0.781	0.296	0.772

Table 5.26: Recognition scores reported for the CNN model, for all context labels. Model configuration using CNN layers' `out_channels = 32` for Acc, Gyro and `64` for WAcc, Aud. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

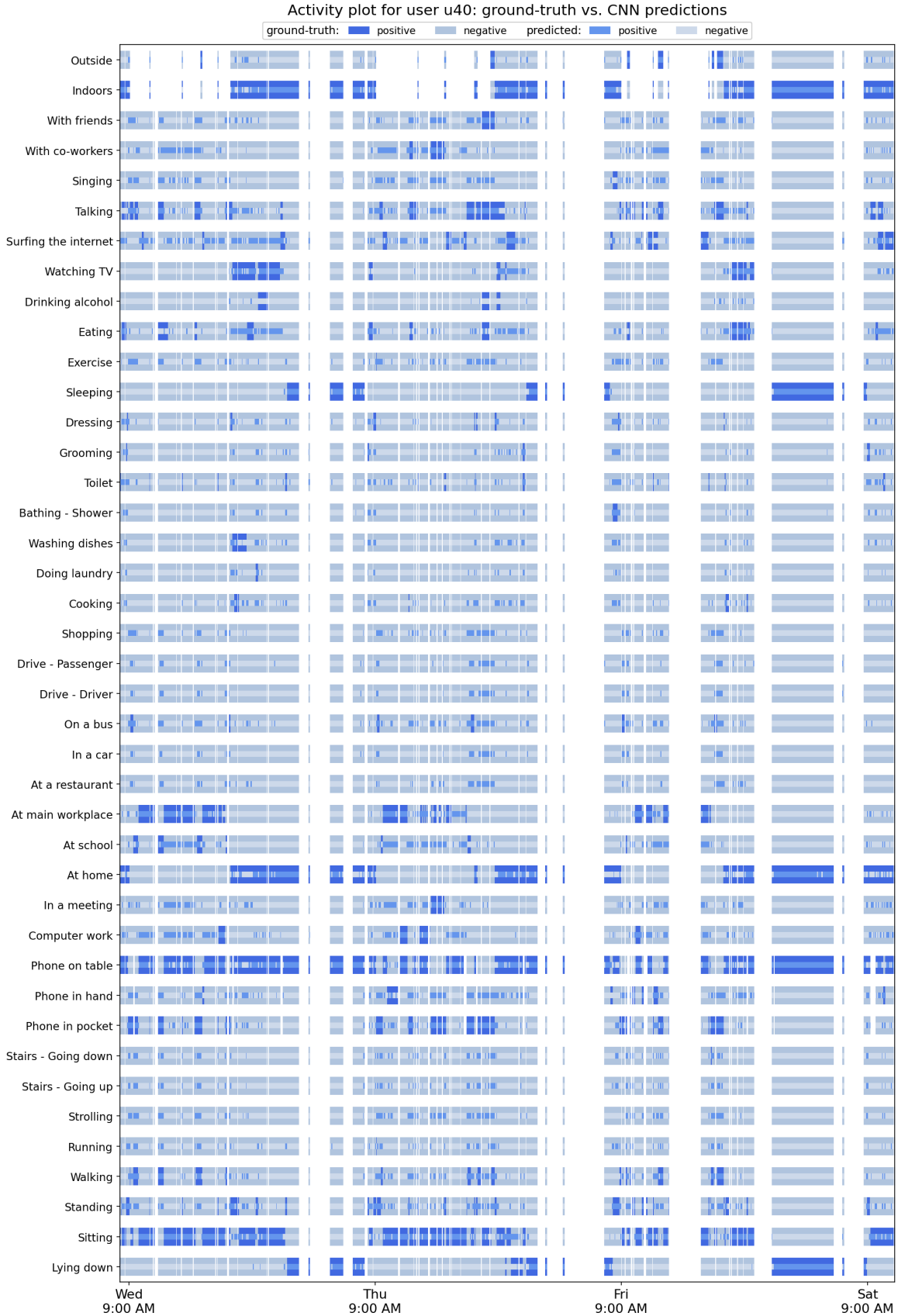


Figure 5.7.3: Activity plot including the predictions of the CNN model for user u40

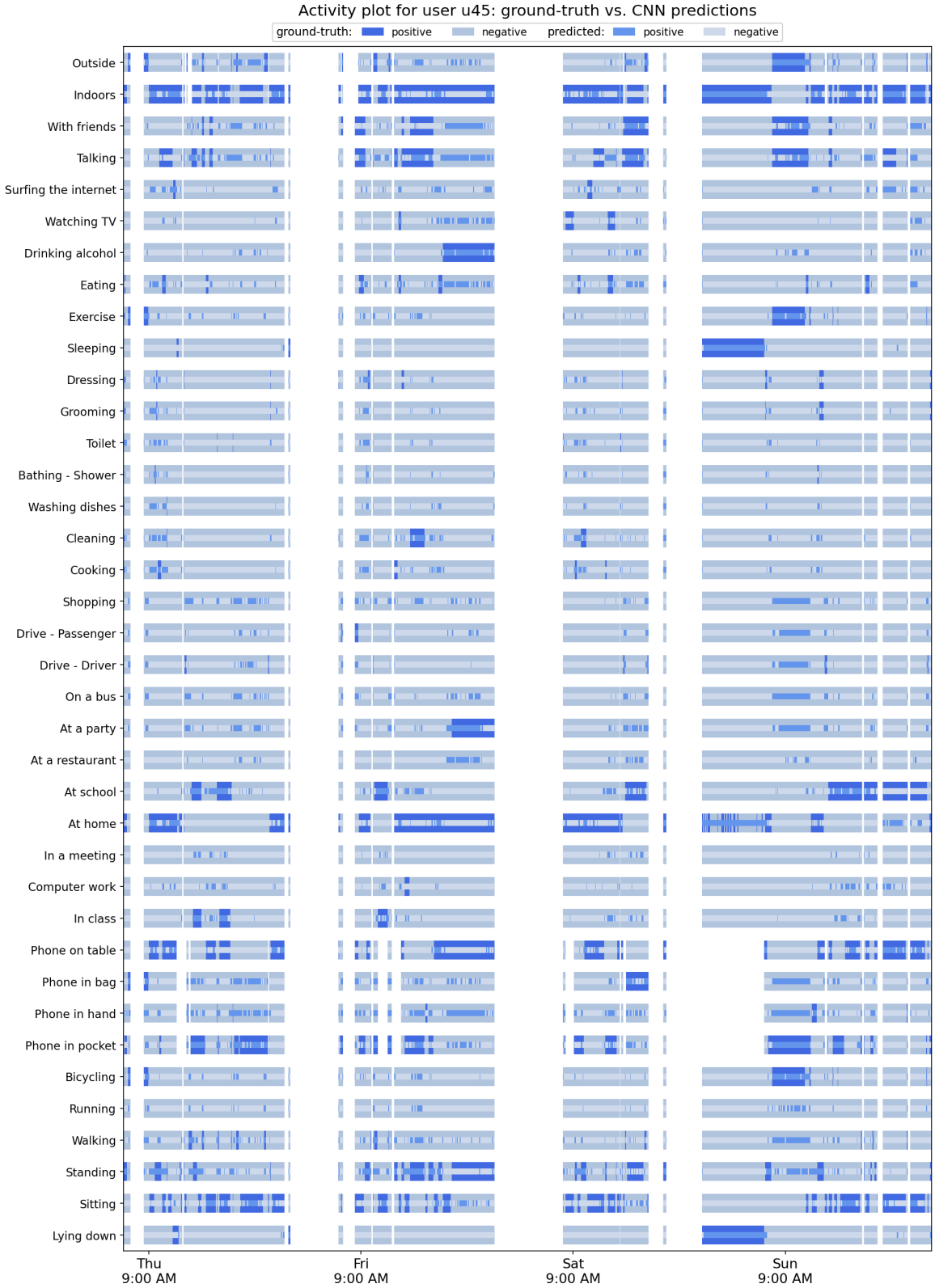


Figure 5.7.4: Activity plot including the predictions of the CNN model for user u45

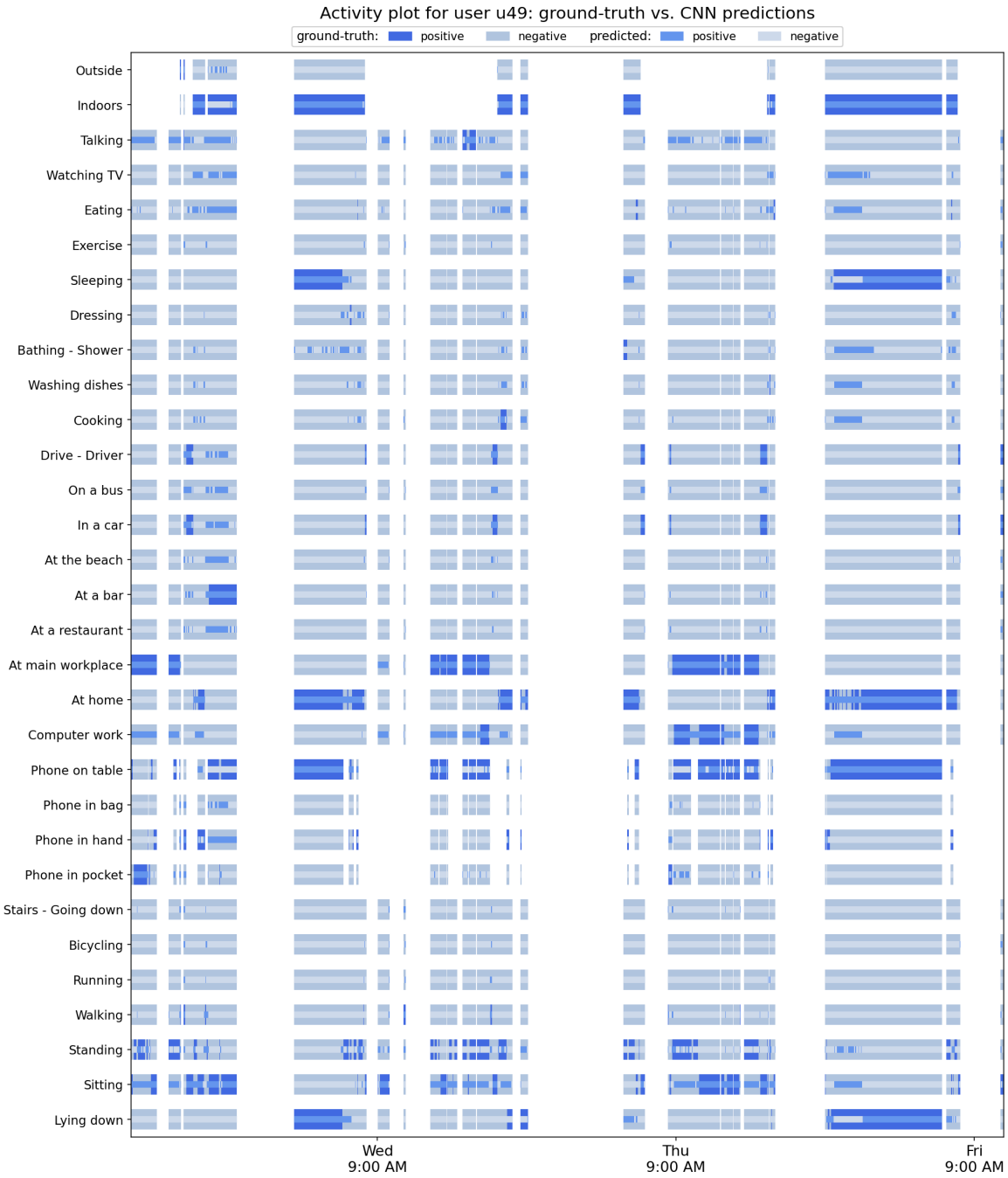


Figure 5.7.5: Activity plot including the predictions of the CNN model for user u49

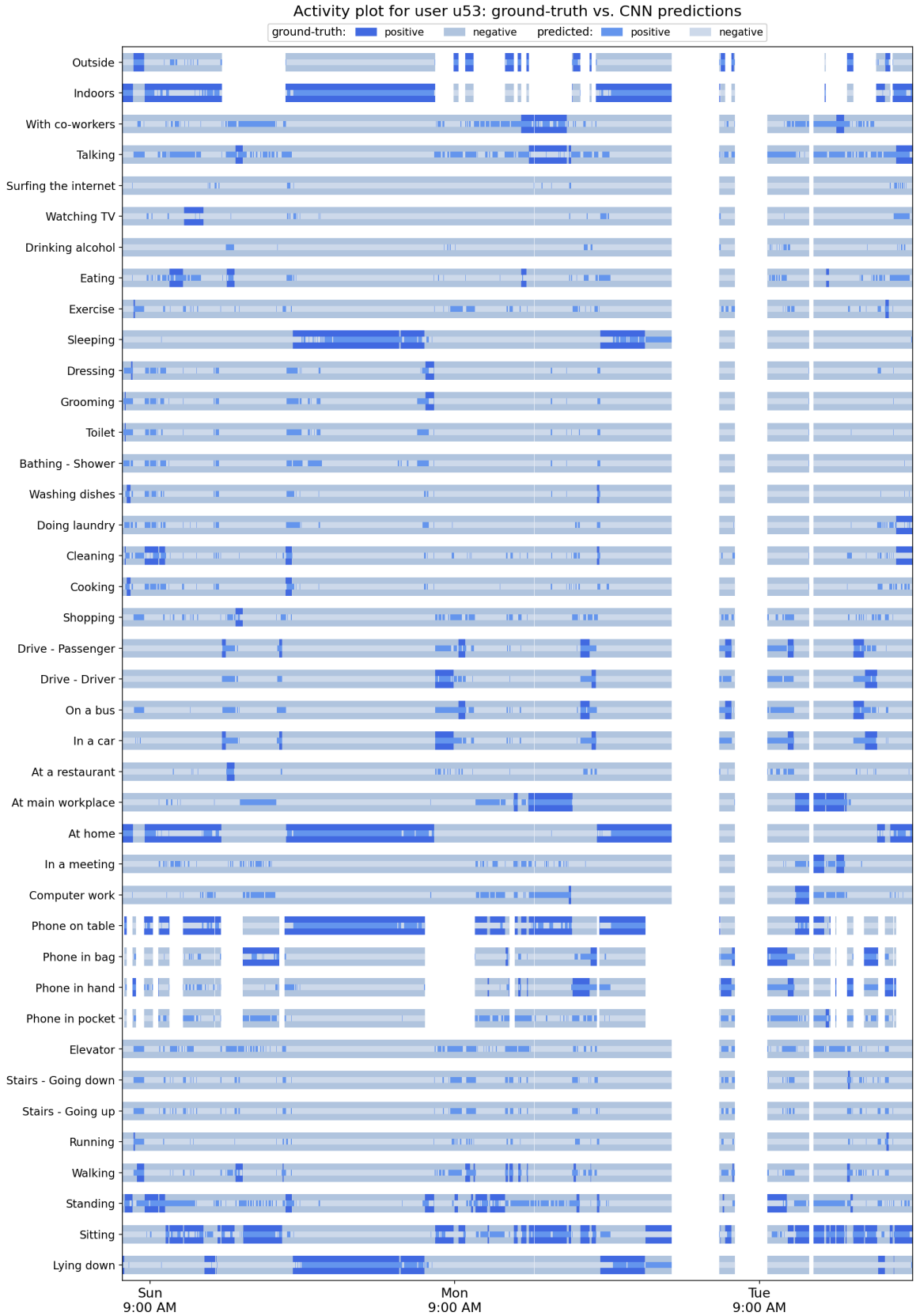


Figure 5.7.6: Activity plot including the predictions of the CNN model for user u53

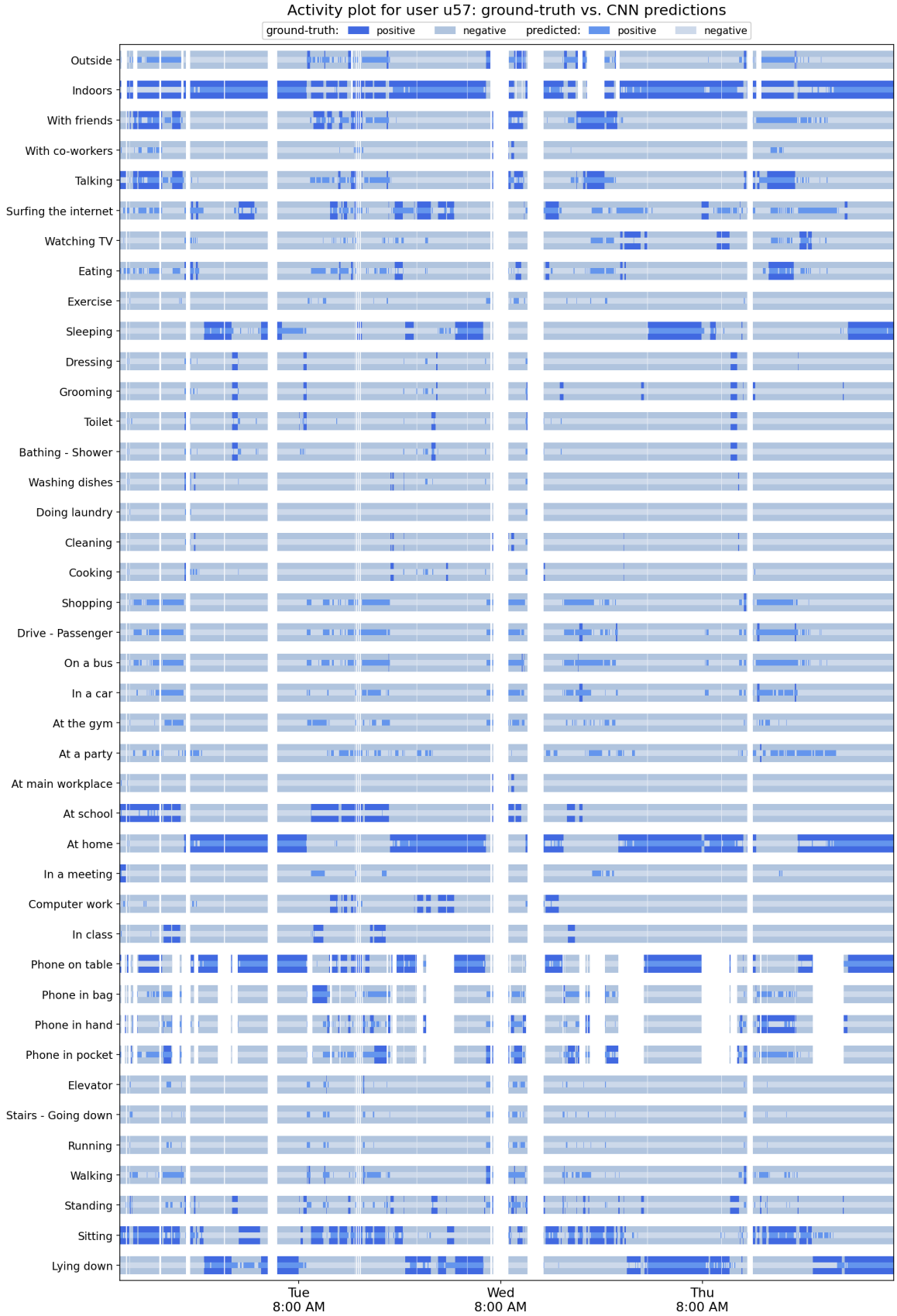


Figure 5.7.7: Activity plot including the predictions of the CNN model for user u57

5.7.2 CNN-Transformer-based Feature Extraction for IMU and Audio Data

Since the CNN pipeline for feature extraction does not take into account the temporal and sequential nature of the raw sensor data, we attempt to build feature extraction pipelines leveraging the benefits of recurrent layers, based on BiLSTM layers or CNN layers followed by BiLSTM layers. However, these attempts were unsuccessful since the resulting models' BA were far lower than the BA reported for the baseline MLP(16, 16) and the CNN-based model of the previous Subsection. Subsequently, our next thought was to test a Transformer-based model. Since the Transformer's intended use is for sequence-to-sequence problems, while our problem is a many-to-one classification problem, we use only Transformer Encoder layers to leverage their sequence modeling capabilities, inspired by [SK21]. The Transformer Encoder layers are preceded by CNN layers to transform the input data time-series in a shorter one, embedded in a higher dimension. This model is conceptually presented in Figure 5.7.8, and the key points of the experiments are summarized below:

- Five-fold cross validation is used, with each fold containing 48 users in the training set and 12 users in the test set (using the same partition as the original work).
- The training set of each CV iteration is further partitioned in a training subset and a validation subset, as follows: for each of the 48 users of the training set, 80% of their data is used for training, and 20% is used for validation, to decide which epoch's checkpoint will be selected for the test inference.
- Regarding Loc and PS, for which we use the pre-extracted features provided in the dataset, we standardize each sensor feature (mean and standard deviation are calculated using the training subset). After standardization, missing feature values are zero-imputated. Regarding the rest of the sensors, for which we use the raw sensor measurements, we standardize the measurements of each sensor (mean and standard deviation are also calculated using the training subset).
- The model configuration includes a pipeline based on CNN layers and Transformer Encoder layer for each of the raw signals' sensors (Acc, Gyro, WAcc, Aud) to extract features from each sensor measurements' time-series. The pipeline for each of these sensors first includes four CNN layers that contain the following operations, in this order: 2D/1D Conv layer, leaky ReLU, Batch Normalization, and Dropout.
- The feature extraction pipeline for each of the raw data sensors also includes a Transformer Encoder module, which includes two Transformer Encoder layers, preceded by prepending the [CLS] token and adding a trainable positional encoding in the input sequence.
- The outputs of all sensors' CNN-Transformer pipelines are concatenated, and also concatenated with the pre-extracted features from Loc and PS, and then they are fed to the baseline MLP configuration which includes two hidden layers, followed by an output layer of 51 nodes which correspond to the 51 context labels. Each of the hidden layers is followed by leaky ReLU activation, Batch Normalization, and Dropout, as described in Subsection 5.4.4.
- Regarding the hyperparameters that we use in this experiment, for the CNN layers we use `out_channels = 48` for all raw sensor modalities, and detailed information on the CNN layers' configuration can be found in Table 5.27. For the Transformer Encoder, the hyperparameters which produced the best results in our experiments are: Transformer Encoder `num_layers = 2`, and for each Transformer Encoder layer: `nhead = 4` for the Multi-head Attention, `dim_feedforward = 64` for the Linear layer, `dropout = 0.2` and GELU activation.
- Regarding the loss, we again use the custom loss that was first described in Subsection 5.4.4, based on `torch.nn.BCEWithLogitsLoss`. This loss is slightly modified to allow dynamic per-batch, per-element masking in the loss matrices (to mask the loss elements corresponding to missing ground-truth labels for each batch, and not use them in the loss computation). Also, the `pos_weight` is used for instance-weighting to account for the imbalance in the number of positive examples per context label, by multiplying the term of the positive examples in the loss, with the ratio of negative to positive examples for this label in the training set.
- We use a `batch size` of 32 to train the model.
- We use the Adam optimizer to train the model, with `learning rate = 0.0005`.

- In the testing phase, the continuous predictions of the model pass through a sigmoid activation function, and then they are converted to binary outputs using a threshold of 0.5.
- We calculate the evaluation metrics for each label over its non-missing ground-truth examples.

We use `PyTorch` to build the model, the supervisor and the training scripts. The `torch.nn` modules and `torch` functions that are used, can be seen in detail in Figure 5.7.9.

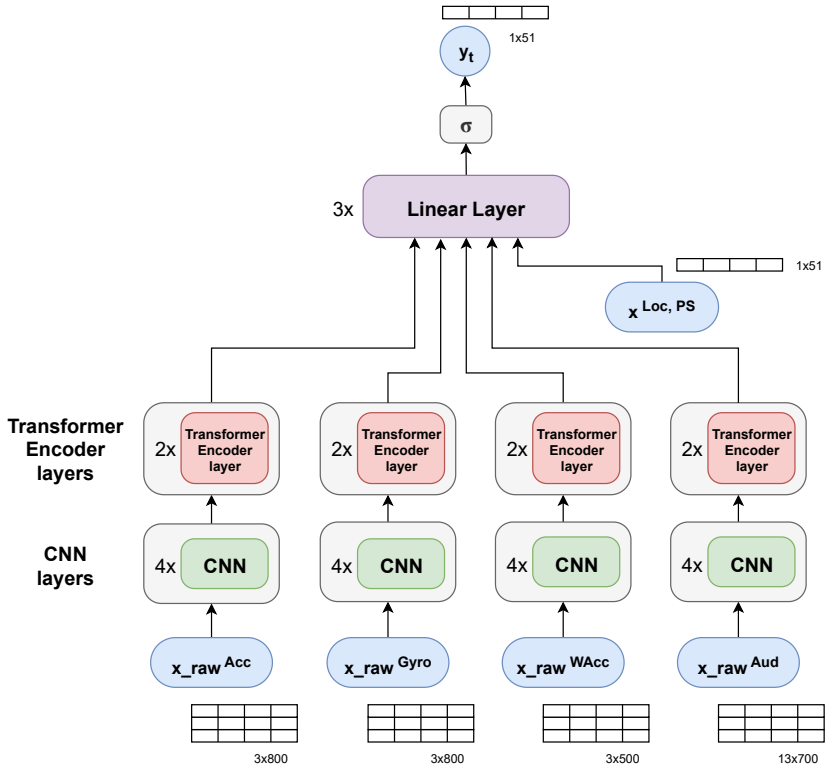


Figure 5.7.8: CNN-Transformer model for feature extraction from sensor measurements, presented conceptually. For raw sensor measurements for a single example from Acc, Gyro, WAcc and Aud sensors, four CNN layers followed by two Transformer Encoder layers are used to extract features. The outputs of all sensors' CNN-Transformer pipelines are concatenated, and also concatenated with the pre-extracted features from Loc and PS, and then they are fed to an MLP that includes two hidden layers and an output layer, followed by a sigmoid function to convert the model's outputs to labels' probabilities.

For the aforementioned model configuration, we present the recognition scores for all context labels in Table 5.29 and averaged over the labels of each label subset in Table 5.28. Although the combination of CNN layers and Transformer Encoder layers is promising, based on previous works applying CNN-Transformer pipelines for feature extraction in HAR tasks, such as [SK21], in which the CNN-Transformer model outperforms a CNN baseline in HAR tasks for activities performed in a controlled environment, in our case, the integration of two Transformer Encoder layers in the feature extraction pipeline does not offer any substantial improvement in the resulting BA score (0.774) compared to the BA of the CNN-based feature extraction pipeline (0.772). We have experimented extensively with the hyperparameters of this CNN-Transformer architecture, varying the CNN number of layers, number of filters, kernel size and stride, and also investigating the impact of the Transformer Encoder layers' number, Attention heads' number and Linear layer size, but we were not able to achieve higher recognition scores. There could be multiple reasons why the Transformer Encoder layers are not compatible with this task: beyond the fact that Transformers are mainly oriented to and have been proved to be well-suited mainly for sequence-to-sequence tasks [Vas+17] which is not our case, we could not find any guarantees regarding the performance of Transformers on noisy data, and, in addition, we could not satisfy the "data-hungry" property of Transformers, since in our dataset, especially for rare labels, the available data are scarce.

CNN architecture hyperparameters for the CNN-Transformer model						
Smartphone accelerometer (**Acc**)						
Input shape: $(B, 1, 3, 800)$						
Layer type	**No. of filters**	**Kernel size**	**Stride**	**Normalization**	**Activation**	**Dropout**
Conv2d	48	(1, 5)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(1, 7)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(3, 11)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(1, 15)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Output shape: $(B, 48, 3, 4)$						
Smartphone gyroscope (**Gyro**)						
Input shape: $(B, 1, 3, 800)$						
Layer type	**No. of filters**	**Kernel size**	**Stride**	**Normalization**	**Activation**	**Dropout**
Conv2d	48	(1, 5)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(1, 7)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(3, 11)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(1, 15)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Output shape: $(B, 48, 3, 4)$						
Smartwatch accelerometer (**WAcc**)						
Input shape: $(B, 1, 3, 500)$						
Layer type	**No. of filters**	**Kernel size**	**Stride**	**Normalization**	**Activation**	**Dropout**
Conv2d	48	(1, 5)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(1, 7)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(3, 11)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(1, 15)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Output shape: $(B, 48, 3, 1)$						
Smartphone audio (**Aud**)						
Input shape: $(B, 1, 13, 700)$						
Layer type	**No. of filters**	**Kernel size**	**Stride**	**Normalization**	**Activation**	**Dropout**
Conv2d	48	(1, 5)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(1, 7)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(3, 11)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Conv2d	48	(1, 15)	(1, 3)	Batch	LeakyReLU (0.1)	0.2
Output shape: $(B, 48, 13, 3)$						

Table 5.27: CNN architecture hyperparameters for the CNN-Transformer model, for Acc, Gyro, WAcc and Audio CNN encoders, with CNN layers' No. of filters `out_channels = 48`

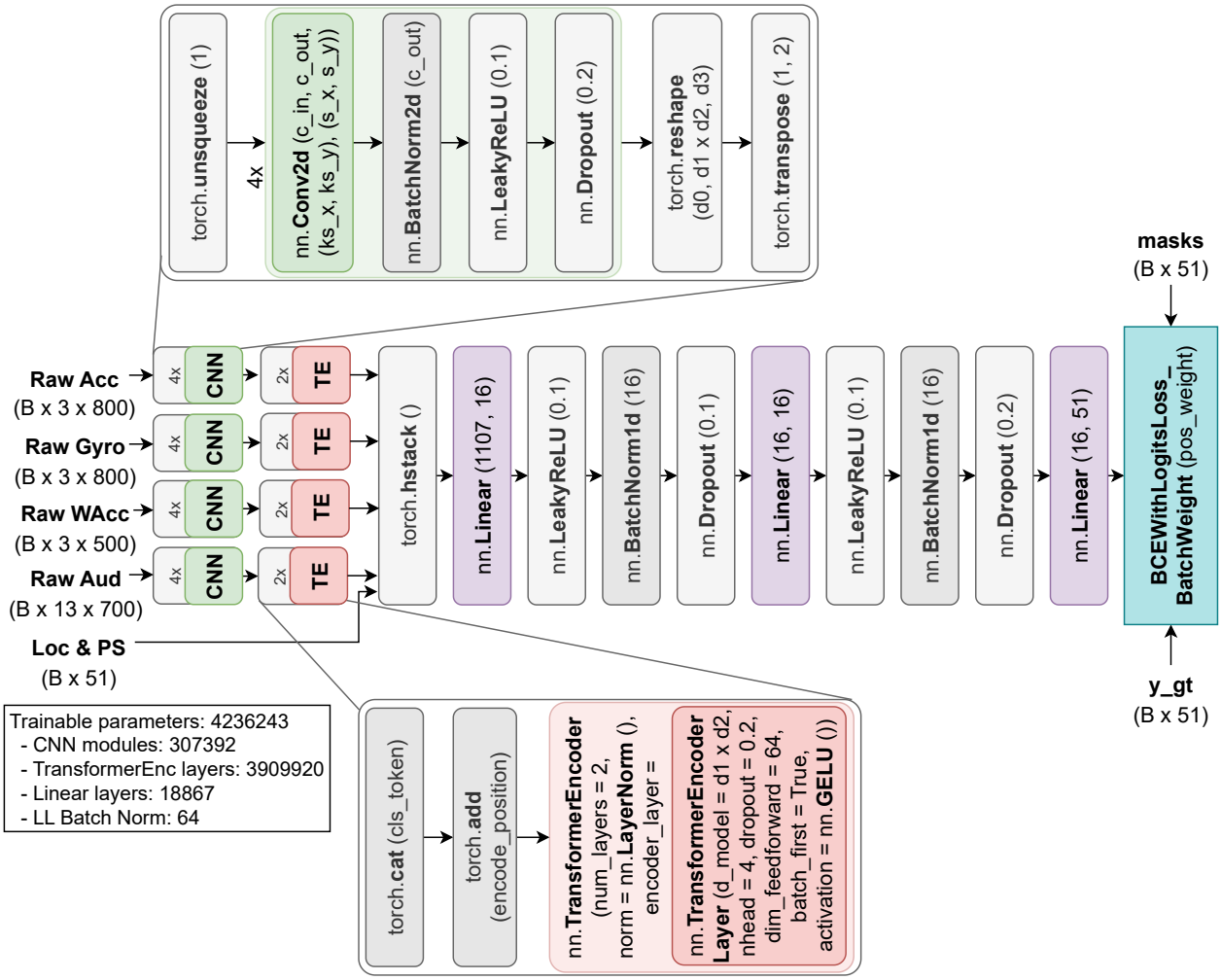


Figure 5.7.9: CNN-Transformer model architecture, using `out_channels = 48` for Acc, Gyro, WAcc, Aud. For raw sensor measurements for a single example, a feature extraction pipeline including CNN and Transformer Encoder layers is used. The hyperparameters for each CNN layer are thoroughly listed in Table 5.27. The outputs of all raw sensors' pipelines are concatenated, and also concatenated with the pre-extracted features from Loc and PS, and they are fed to the baseline MLP(16,16) architecture of Subsection 4.3.6. The aforementioned custom loss based on `torch.nn.BCEWithLogitsLoss` is used to train the model.

Recognition scores of the CNN-Transformer model, averaged for each label subset						
Label	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Posture/Movement	0.807	0.377	0.802	0.798	0.459	0.800
Special Movement	0.827	0.029	0.690	0.828	0.055	0.759
Phone Location	0.769	0.455	0.794	0.760	0.535	0.777
Work-related	0.792	0.247	0.814	0.788	0.368	0.801
Location-based	0.818	0.265	0.792	0.814	0.336	0.803
Transportation	0.832	0.140	0.893	0.830	0.233	0.861
Chores	0.782	0.039	0.665	0.784	0.073	0.724
Self-care	0.813	0.185	0.689	0.817	0.221	0.753
Leisure Time	0.719	0.142	0.701	0.717	0.232	0.709
Companion	0.694	0.193	0.674	0.698	0.296	0.686
Environment	0.862	0.668	0.874	0.867	0.710	0.870
Average	0.792	0.229	0.759	0.790	0.300	0.774

Table 5.28: Recognition scores reported for the CNN-Transformer model, averaged for each label subset. Model configuration using CNN layers' `out_channels = 48` for Acc, Gyro, WAcc, Aud. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

Recognition scores of the CNN-Transformer model, for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.885	0.802	0.834	0.908	0.818	0.871
Sitting	79368	0.768	0.716	0.838	0.706	0.772	0.772
Standing	22071	0.661	0.238	0.729	0.650	0.359	0.690
Walking	11715	0.786	0.217	0.801	0.785	0.342	0.793
Running	661	0.860	0.037	0.731	0.861	0.071	0.796
Bicycling	3504	0.881	0.251	0.880	0.881	0.390	0.880
Strolling	339	0.793	0.054	0.794	0.793	0.101	0.793
Stairs - Going up	399	0.835	0.023	0.551	0.837	0.044	0.694
Stairs - Going down	390	0.843	0.026	0.610	0.845	0.050	0.727
Elevator	123	0.838	0.014	0.805	0.838	0.027	0.821
Phone in pocket	14074	0.801	0.469	0.818	0.797	0.597	0.808
Phone in hand	7313	0.653	0.171	0.754	0.643	0.279	0.699
Phone in bag	5031	0.795	0.265	0.771	0.797	0.394	0.784
Phone on table	65979	0.825	0.914	0.834	0.803	0.872	0.819
In class	2852	0.822	0.213	0.784	0.825	0.335	0.804
Lab work	2898	0.795	0.336	0.882	0.784	0.486	0.833
Computer work	22536	0.729	0.344	0.782	0.719	0.478	0.750
In a meeting	2837	0.822	0.097	0.807	0.823	0.173	0.815
At home	80044	0.787	0.749	0.833	0.744	0.789	0.788
At school	25342	0.774	0.428	0.749	0.779	0.545	0.764
At main workplace	19235	0.838	0.566	0.841	0.838	0.677	0.839
At a restaurant	1275	0.852	0.077	0.814	0.853	0.140	0.833
At a bar	520	0.793	0.106	0.923	0.789	0.190	0.856
At a party	404	0.830	0.071	0.782	0.831	0.130	0.807
At the gym	897	0.835	0.101	0.630	0.841	0.174	0.735
At the beach	116	0.839	0.023	0.767	0.839	0.045	0.803
In a car	3550	0.834	0.158	0.885	0.832	0.269	0.858
On a bus	1179	0.811	0.055	0.882	0.810	0.104	0.846
Drive - Driver	4879	0.865	0.263	0.898	0.863	0.407	0.881
Drive - Passenger	1650	0.818	0.084	0.905	0.816	0.154	0.861
Shopping	809	0.781	0.037	0.823	0.781	0.070	0.802
Cooking	2212	0.766	0.055	0.746	0.766	0.102	0.756
Cleaning	1813	0.760	0.046	0.541	0.765	0.085	0.653
Doing laundry	471	0.804	0.025	0.563	0.806	0.048	0.684
Washing dishes	829	0.800	0.031	0.650	0.801	0.060	0.726
Bathing - Shower	1120	0.805	0.032	0.646	0.807	0.061	0.727
Toilet	1558	0.756	0.034	0.683	0.757	0.066	0.720
Grooming	1775	0.787	0.047	0.603	0.790	0.087	0.697
Dressing	1248	0.816	0.037	0.639	0.818	0.071	0.728
Sleeping	40869	0.901	0.772	0.873	0.911	0.819	0.892
Exercise	5191	0.866	0.178	0.741	0.871	0.287	0.806
Eating	9668	0.597	0.108	0.786	0.584	0.190	0.685
Drinking alcohol	859	0.814	0.091	0.873	0.813	0.165	0.843
Watching TV	8945	0.729	0.201	0.722	0.730	0.314	0.726
Surfing the internet	10668	0.643	0.163	0.599	0.649	0.257	0.624
Talking	18477	0.619	0.210	0.764	0.599	0.329	0.681
Singing	384	0.767	0.044	0.422	0.776	0.080	0.599
With co-workers	3972	0.753	0.176	0.785	0.751	0.287	0.768
With friends	12686	0.635	0.210	0.564	0.646	0.306	0.605
Indoors	102510	0.870	0.991	0.869	0.882	0.926	0.876
Outside	6793	0.854	0.344	0.879	0.851	0.495	0.865
Average		0.792	0.229	0.759	0.790	0.300	0.774

Table 5.29: Recognition scores reported for the CNN-Transformer model, for all context labels. Model configuration using CNN layers' `out_channels = 48` for Acc, Gyro, WAcc, Aud. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

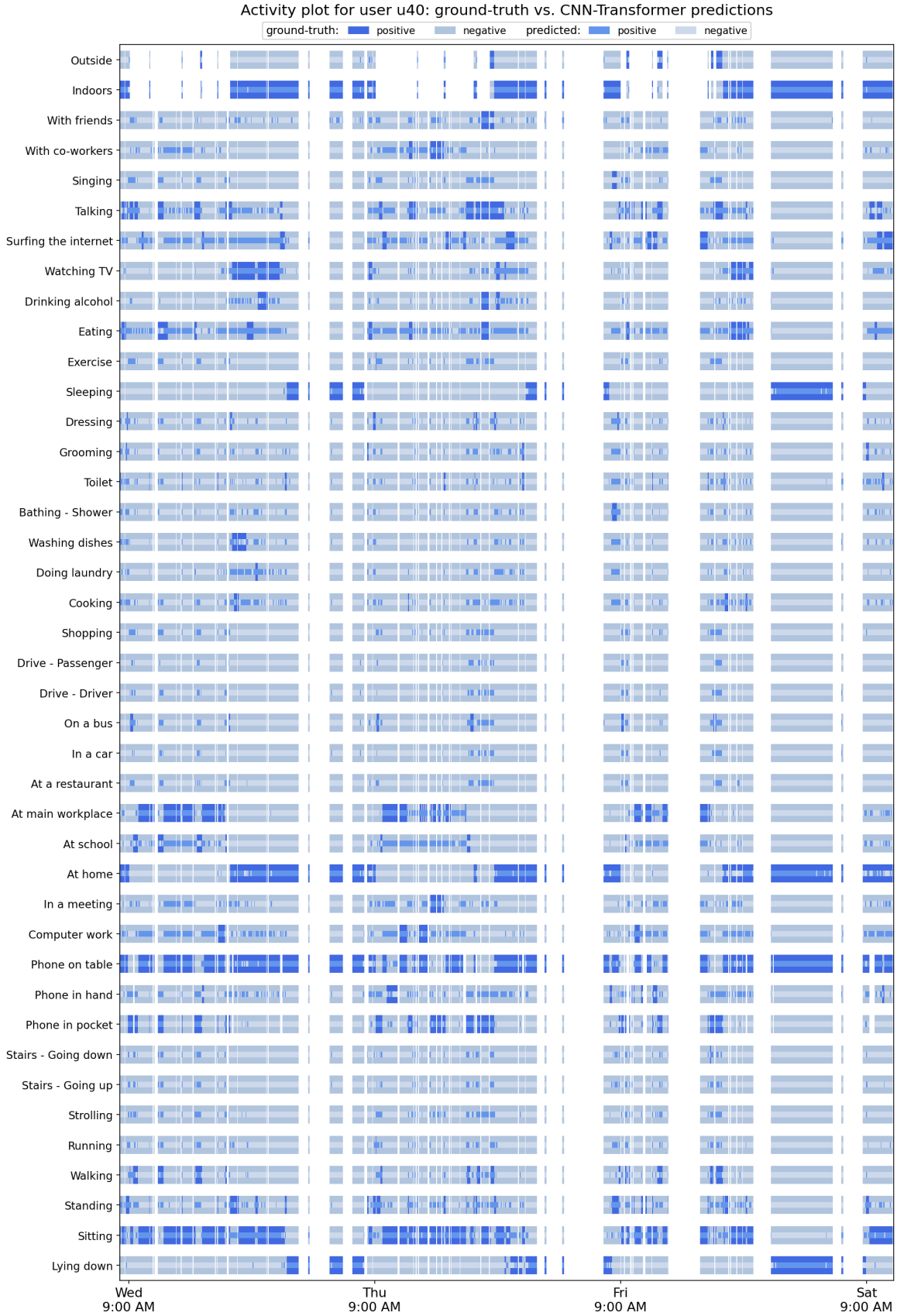


Figure 5.7.10: Activity plot including the predictions of the CNN-Transformer model for user u40

Figure 5.7.11: Activity plot including the predictions of the CNN-Transformer model for user u45

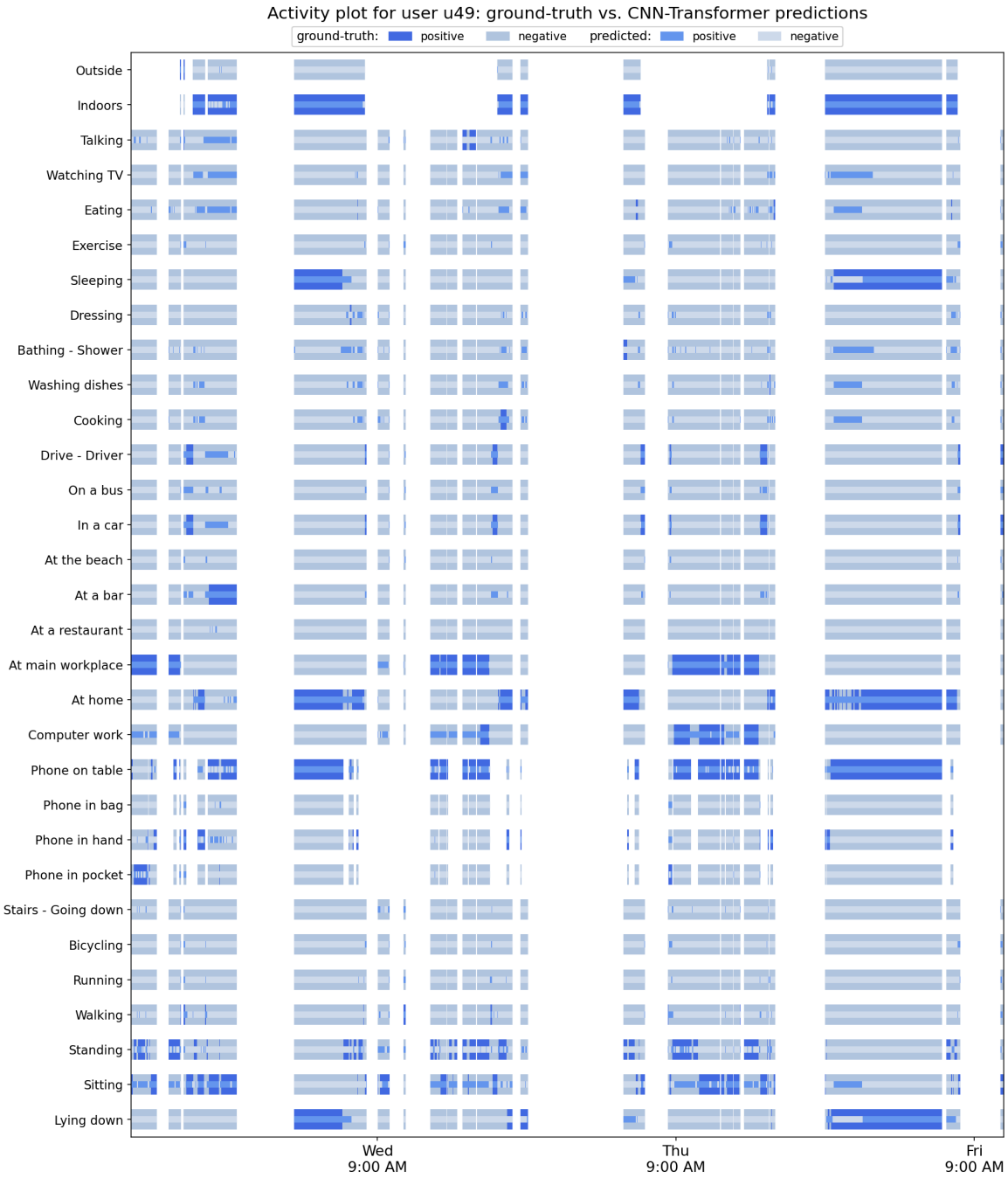


Figure 5.7.12: Activity plot including the predictions of the CNN-Transformer model for user u49

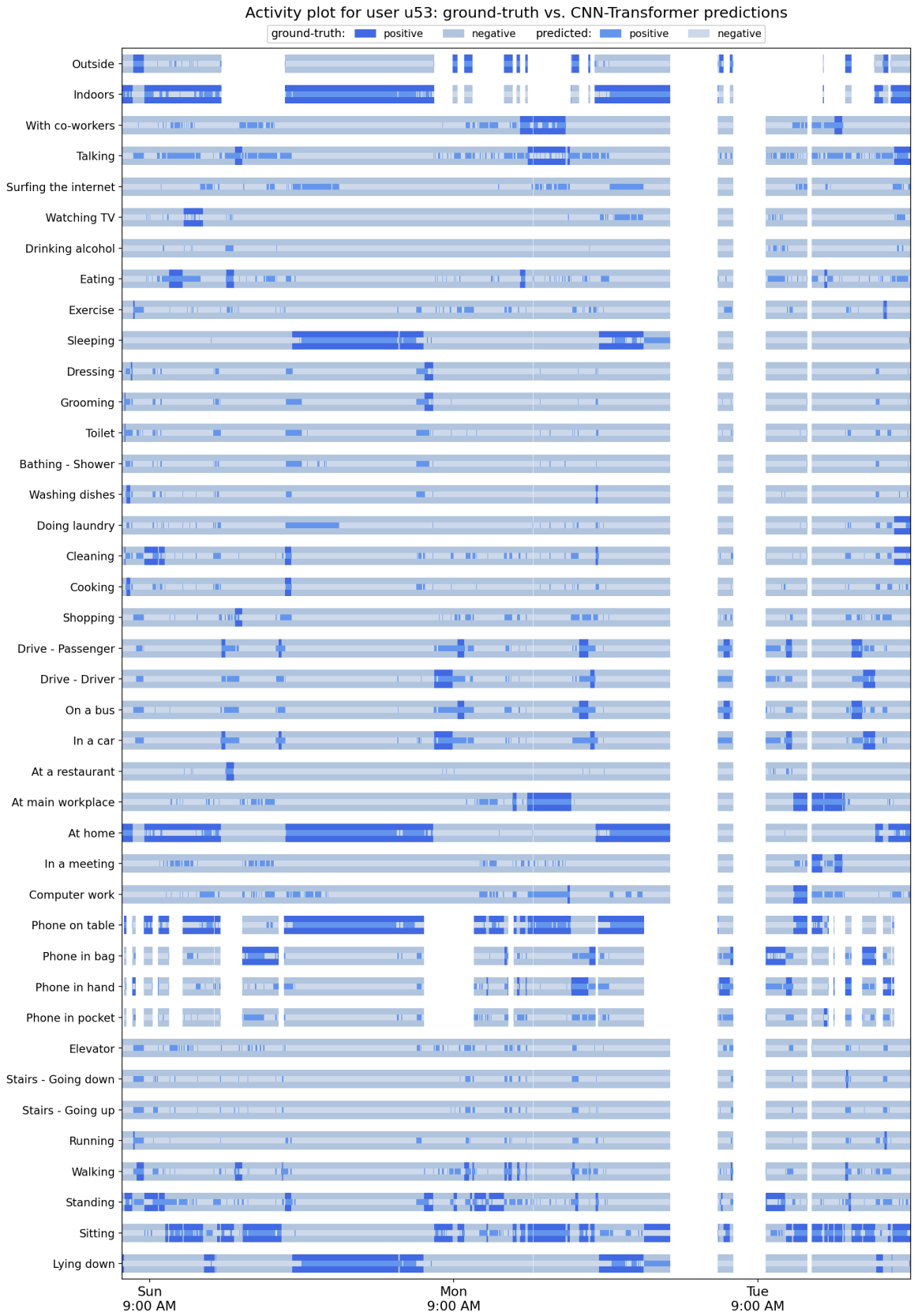


Figure 5.7.13: Activity plot including the predictions of the CNN-Transformer model for user u53

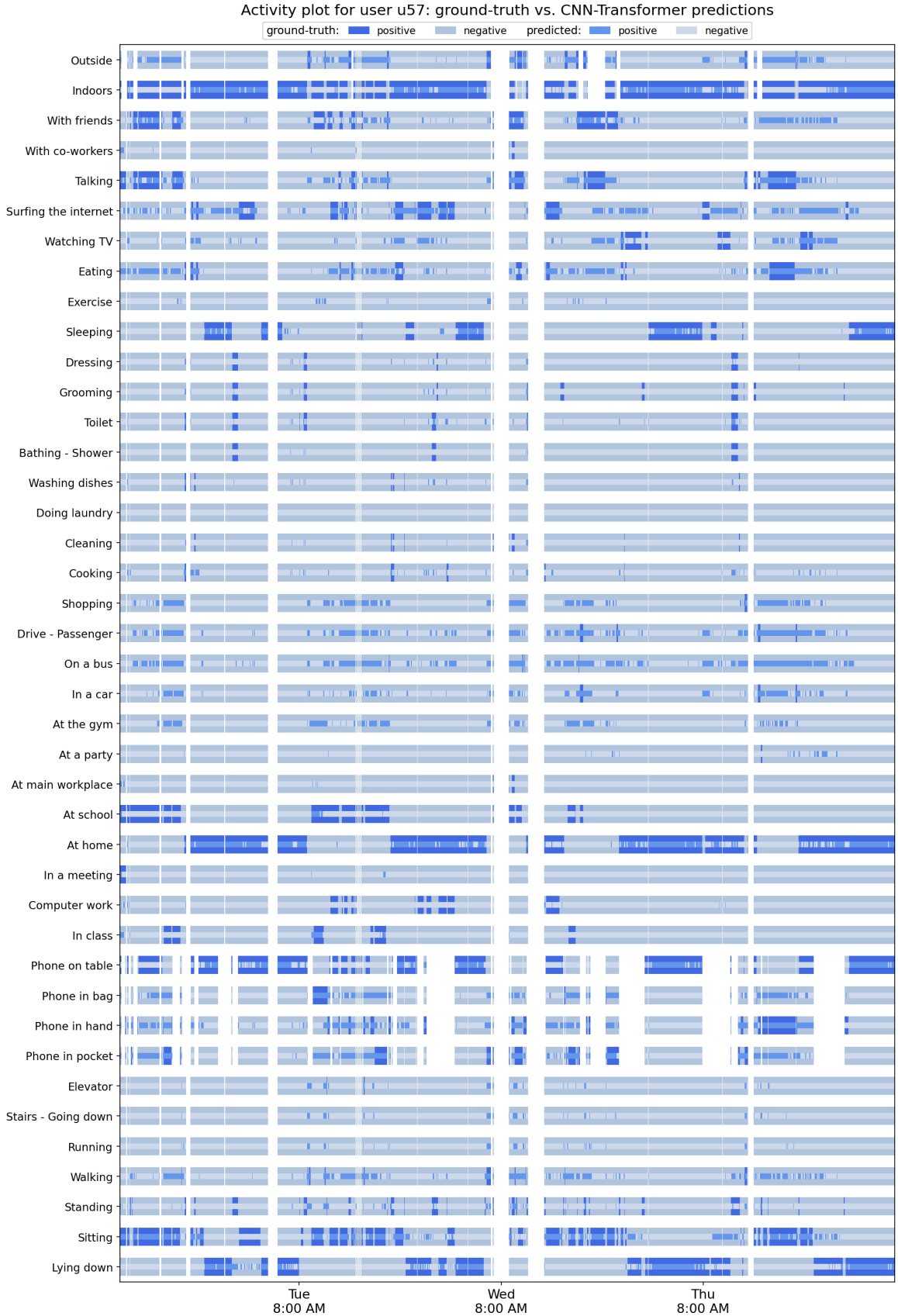


Figure 5.7.14: Activity plot including the predictions of the CNN-Transformer model for user u57

5.8 Deep Learning Feature Extraction combined with BiLSTM Sequence Modeling

In this Section, we present the last experiments that we conducted for this thesis, which combine ideas from the previous Sections. More specifically, we combine Deep Learning-based feature extraction from raw sensor measurements (Section 5.7) with the examples sequence modeling (Section 5.5), so as to give the features extracted by CNN layers as input to the sequence modeling BiLSTM, instead of the hand-crafted features that were given as input in previous BiLSTM experiments.

5.8.1 CNN-based Feature Extraction combined with BiLSTM Sequence Modeling

Our goal is to combine the CNN-based feature extraction from raw sensor data of Subsection 5.7.1 with the BiLSTM sequence modeling of an input sequence of consecutive examples of Section 5.5 and more specifically the BiLSTM model using only final hidden states presented in Subsection 5.5.1. In essence, we implement a feature extraction CNN pipeline very similar to that of Subsection 5.7.1, but we give a sequence of examples as input and we use Time-Distributed CNN layers, so as for all examples in the sequence to be processed in parallel by the CNN layers. When given a sequence of input examples, the Time-Distributed CNN layers produce a sequence of extracted features, which is then fed to a two-layer BiLSTM, whose `h_n` output, which contains a concatenation of the final forward and reverse hidden states of each BiLSTM layer, is then fed to a Linear layer, to predict the probabilities for each of the 51 context labels. We visualize this model in Figure 5.8.1, and the key points of our implementation are summarized below:

- Five-fold cross validation is used, with each fold containing 48 users in the training set and 12 users in the test set (using the same partition as the original work).
- The training set of each CV iteration is further partitioned in a training subset and a validation subset, as follows: for each of the 48 users of the training set, 80% of their data is used for training, and 20% is used for validation, to decide which epoch's checkpoint will be selected for the test inference.
- Regarding Loc and PS, for which we use the pre-extracted features provided in the dataset, we standardize each sensor feature (mean and standard deviation are calculated using the training subset). After standardization, missing feature values are zero-imputated. Regarding the rest of the sensors, for which we use the raw sensor measurements, we standardize the measurements of each sensor (mean and standard deviation are also calculated using the training subset).
- The model configuration includes a CNN pipeline for each of the raw signals' sensors (Acc, Gyro, WAcc, Aud) to extract features from each sensor measurements' time-series. The CNN pipeline for each of these sensors includes multiple layers that contain the following operations, in this order: 2D/1D Conv layer, leaky ReLU, Batch Normalization, and Dropout. We use `out_channels = 64` for all CNN layers of all raw sensor modalities, and detailed information on the CNN layers' configuration hyperparameters can be found in Table 5.31.
- Instead of a single example, an examples sequence of length `window_len = 5` is given as input. Since the Time-Distributed wrapper which allows to apply a layer to every temporal slice of an input that exists in `Keras`, is not available in `PyTorch` as far as we know, we implement it manually, by reshaping the CNN layers' input of shape `(batch_size, window_len, input_channels, timeseries_length)` to `(batch_size` $\times$ `window_len, 1, input_channels, timeseries_length)` in order for the CNN layers to process all the examples of the input sequence of length in parallel. We reshape back the CNN layers' output, in order to acquire a sequence of extracted features corresponding to the input examples sequence.
- The outputs of all sensors' CNN pipelines are concatenated, and also concatenated with the pre-extracted features from Loc and PS, and then they pass through a two-layer BiLSTM with `hidden_size = 64` followed by a Linear layer, as described in Subsection 5.5.1. The `h_n` output of the BiLSTM, which contains a concatenation of the final forward and reverse hidden states of the two BiLSTM layers, is fed to the Linear layer of 51 nodes which correspond to the 51 context labels. We use Dropout after each BiLSTM layer, with `dropout rate = 0.5` to help avoid overfitting.

- Regarding the loss, we again use the custom loss that was first described in Subsection 5.4.4, based on `torch.nn.BCEWithLogitsLoss`. This loss is slightly modified to allow dynamic per-batch, per-element masking in the loss matrices (to mask the loss elements corresponding to missing ground-truth labels for each batch, and not use them in the loss computation). Also, the `pos_weight` is used for instance-weighting to account for the imbalance in the number of positive examples per context label, by multiplying the term of the positive examples in the loss, with the ratio of negative to positive examples for this label in the training set.
- We use a `batch size` of 32 to train the model.
- We use the Adam optimizer to train the model, with `learning rate = 0.00005`. We notice that when modeling a sequence of examples and using a BiLSTM, smaller learning rates are required for better model convergence, which is also in accordance with the learning rates of Section 5.5.
- In the testing phase, the continuous predictions of the model pass through a sigmoid activation function, and then they are converted to binary outputs using a threshold of 0.5.
- We calculate the evaluation metrics for each label over its non-missing ground-truth examples.

`PyTorch` is used to build the model, the supervisor and the training scripts. The `torch.nn` modules and `torch` functions that we use are presented in Figure 5.8.2.

For the aforementioned model configuration, we present the recognition scores for all context labels in Table 5.32 and averaged over the labels of each label subset in Table 5.30. With an average BA of 0.775, we notice that this CNN-BiLSTM model leveraging both raw sensor measurements from Acc, Gyro, WAcc and Aud, and a 5-length examples sequence as input, does not offer any substantial improvement over the previous CNN-based and CNN-Transformer-based models that used raw sensor measurements from a single example as input. Contrary to the previous experiments of Section 5.5 using hand-crafted features, where providing a sequence of examples led in a substantial improvement on the recognition metrics, here, when we extract features from the raw sensor data directly in the model, providing a sequence of examples does not seem to help the model, at least in the extent of the experiments that it was possible to conduct in our study. Although we have extensively experimented with the model's hyperparameters, regarding both the CNN layers (number of layers, number of filters, kernel size, stride) and the BiLSTM (number of layers, hidden size), it hasn't been possible to produce better results than the ones reported here.

Recognition scores of the CNN-BiLSTM model, averaged for each label subset

Label	Accuracy	Precision	Sensitivity	Specificity	F1-score	BA
Posture/Movement	0.826	0.391	0.788	0.829	0.472	0.808
Special Movement	0.856	0.029	0.570	0.858	0.055	0.714
Phone Location	0.782	0.461	0.770	0.777	0.538	0.773
Work-related	0.817	0.260	0.735	0.821	0.377	0.778
Location-based	0.816	0.276	0.821	0.814	0.343	0.817
Transportation	0.861	0.163	0.887	0.861	0.265	0.874
Chores	0.824	0.047	0.644	0.826	0.087	0.735
Self-care	0.868	0.191	0.626	0.871	0.238	0.749
Leisure Time	0.765	0.158	0.654	0.773	0.246	0.713
Companion	0.725	0.208	0.650	0.736	0.312	0.693
Environment	0.885	0.693	0.887	0.883	0.742	0.885
Average	0.819	0.241	0.728	0.821	0.313	0.775

Table 5.30: Recognition scores reported for the CNN-BiLSTM model, averaged for each label subset. Model configuration using CNN layers' `out_channels = 64` for Acc, Gyro, WAcc, Aud. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

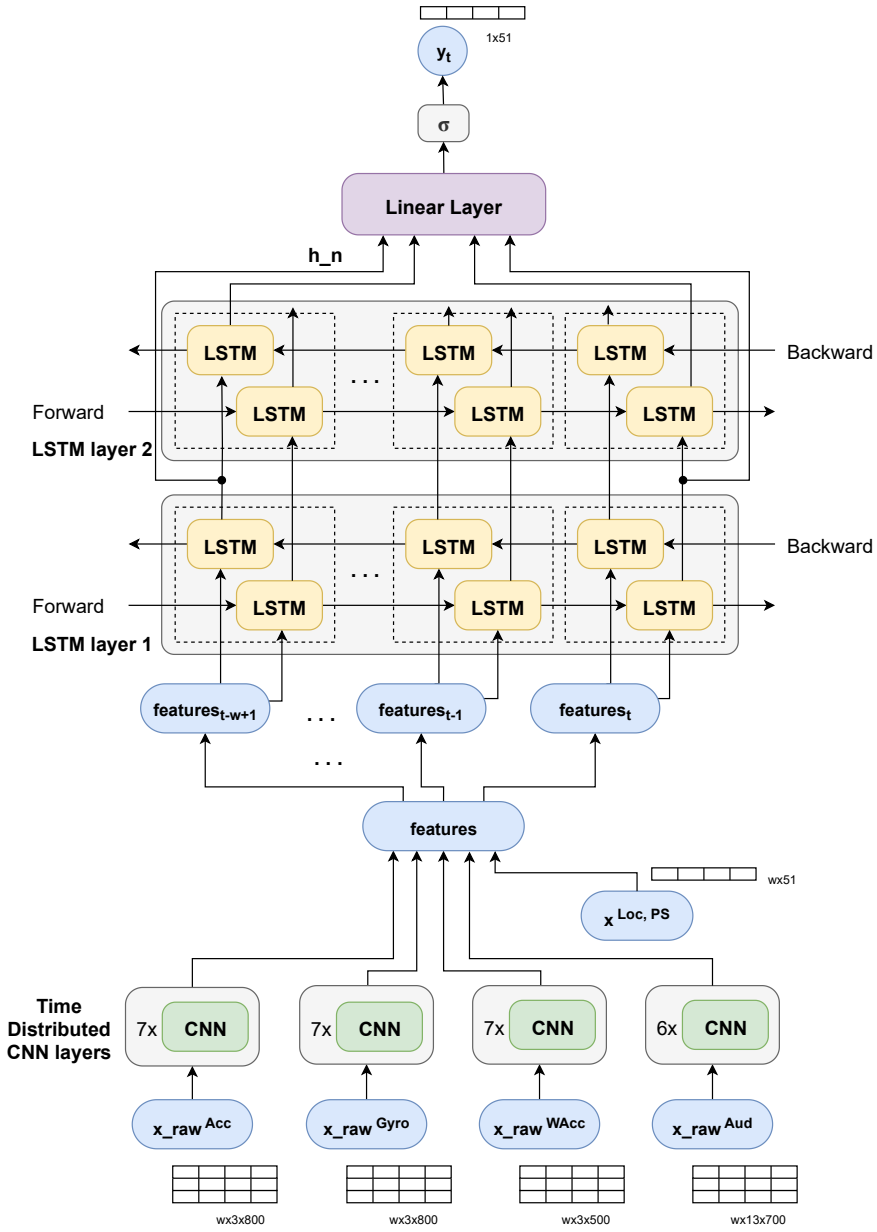


Figure 5.8.1: CNN-based model for feature extraction from sensor measurements, combined with BiLSTM examples sequence modeling, presented conceptually. An examples sequence of length `w (window_len)` is given as input, and the target is to predict the context labels of the last example of the sequence. For raw sensor measurements from Acc, Gyro, WAcc and Aud sensors, Time-Distributed CNN layers are used to extract features. The outputs of all sensors' CNN pipelines are concatenated, and also concatenated with the pre-extracted features from Loc and PS, and then they are fed to a two-layer BiLSTM. The `h_n` output of the BiLSTM, which contains a concatenation of the final forward and reverse hidden states of each BiLSTM layer, passes through a Linear layer, followed by a sigmoid function to convert the model's outputs to labels' probabilities.

CNN architecture hyperparameters for the CNN-BiLSTM model

Smartphone accelerometer (**Acc**)

Input shape: $(B \times w, 1, 3, 800)$

Layer type	**No. of filters**	**Kernel size**	**Stride**	**Normalization**	**Activation**	**Dropout**
Conv2d	64	(1, 5)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(3, 5)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	3	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	3	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 31)	3	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 17)	1	Batch	LeakyReLU (0.1)	0.4

Output shape: $(B \times w, 64, 1, 1)$

Smartphone gyroscope (**Gyro**)

Input shape: $(B \times w, 1, 3, 800)$

Layer type	**No. of filters**	**Kernel size**	**Stride**	**Normalization**	**Activation**	**Dropout**
Conv2d	64	(1, 5)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(3, 5)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	3	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	3	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 31)	3	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 17)	1	Batch	LeakyReLU (0.1)	0.4

Output shape: $(B \times w, 64, 1, 1)$

Smartwatch accelerometer (**WAcc**)

Input shape: $(B \times w, 1, 3, 500)$

Layer type	**No. of filters**	**Kernel size**	**Stride**	**Normalization**	**Activation**	**Dropout**
Conv2d	64	(1, 5)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(3, 5)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	3	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	2	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 31)	2	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 21)	1	Batch	LeakyReLU (0.1)	0.4

Output shape: $(B \times w, 64, 1, 1)$

Smartphone audio (**Aud**)

Input shape: $(B \times w, 1, 13, 700)$

Layer type	**No. of filters**	**Kernel size**	**Stride**	**Normalization**	**Activation**	**Dropout**
Conv2d	64	(3, 3)	1	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(3, 5)	2	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(3, 9)	2	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	3	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 15)	3	Batch	LeakyReLU (0.1)	0.4
Conv2d	64	(1, 13)	1	Batch	LeakyReLU (0.1)	0.4

Output shape: $(B \times w, 64, 1, 1)$

Table 5.31: CNN architecture hyperparameters for the CNN-BiLSTM model, for Acc, Gyro, WAcc and Audio CNN encoders, with CNN layers' No. of filters `out_channels = 64`. In our experiments, we use an input examples sequence of length `w = 5`.

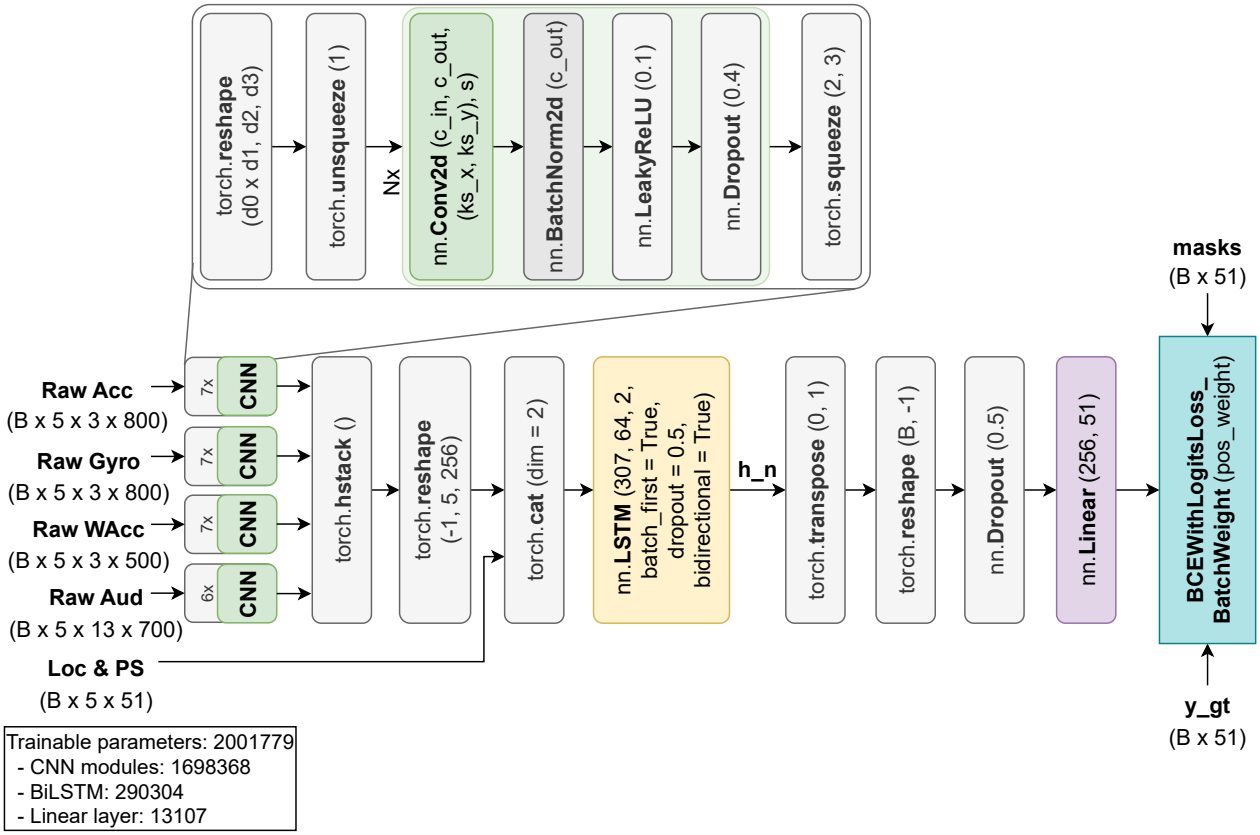


Figure 5.8.2: The selected CNN-BiLSTM architecture, with CNN layers' `out_channels = 64` for Acc, Gyro, WAcc, Aud. For the raw sensor measurements of a 5-length examples sequence (`window_len = 5`), a feature extraction pipeline including Time-Distributed CNN layers is used. The hyperparameters for each CNN layer are thoroughly listed in Table 5.31. The outputs of all sensors' CNN pipelines are concatenated, and also concatenated with the pre-extracted features from Loc and PS, and they are given as input to the two-layer BiLSTM with `hidden_size = 64` and `dropout = 0.5` of Subsection 5.5.1. The aforementioned custom loss based on `torch.nn.BCEWithLogitsLoss` is used to train the model.

Recognition scores of the CNN-BiLSTM model, for all context labels							
Label	**Support**	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1-score**	**BA**
Lying down	51682	0.874	0.764	0.859	0.881	0.808	0.870
Sitting	79368	0.775	0.753	0.775	0.774	0.764	0.775
Standing	22071	0.715	0.267	0.678	0.720	0.383	0.699
Walking	11715	0.819	0.251	0.808	0.820	0.383	0.814
Running	661	0.886	0.045	0.719	0.887	0.085	0.803
Bicycling	3504	0.888	0.264	0.888	0.888	0.408	0.888
Strolling	339	0.794	0.053	0.782	0.794	0.100	0.788
Stairs - Going up	399	0.870	0.026	0.481	0.873	0.049	0.677
Stairs - Going down	390	0.882	0.024	0.423	0.885	0.046	0.654
Elevator	123	0.877	0.013	0.593	0.878	0.026	0.736
Phone in pocket	14074	0.809	0.481	0.804	0.810	0.602	0.807
Phone in hand	7313	0.687	0.184	0.736	0.683	0.295	0.709
Phone in bag	5031	0.803	0.262	0.702	0.813	0.381	0.757
Phone on table	65979	0.827	0.915	0.836	0.804	0.874	0.820
In class	2852	0.842	0.213	0.655	0.853	0.321	0.754
Lab work	2898	0.743	0.269	0.775	0.739	0.399	0.757
Computer work	22536	0.779	0.397	0.751	0.785	0.520	0.768
In a meeting	2837	0.905	0.163	0.759	0.908	0.269	0.834
At home	80044	0.808	0.774	0.846	0.773	0.808	0.809
At school	25342	0.789	0.448	0.719	0.804	0.552	0.762
At main workplace	19235	0.863	0.615	0.850	0.866	0.714	0.858
At a restaurant	1275	0.858	0.080	0.818	0.859	0.146	0.838
At a bar	520	0.773	0.100	0.952	0.769	0.181	0.860
At a party	404	0.748	0.054	0.884	0.746	0.102	0.815
At the gym	897	0.824	0.105	0.715	0.827	0.183	0.771
At the beach	116	0.868	0.029	0.784	0.868	0.056	0.826
In a car	3550	0.871	0.197	0.884	0.871	0.322	0.877
On a bus	1179	0.844	0.067	0.891	0.843	0.125	0.867
Drive - Driver	4879	0.882	0.290	0.887	0.882	0.437	0.884
Drive - Passenger	1650	0.848	0.098	0.886	0.848	0.176	0.867
Shopping	809	0.819	0.044	0.820	0.819	0.084	0.820
Cooking	2212	0.830	0.068	0.669	0.833	0.124	0.751
Cleaning	1813	0.780	0.051	0.548	0.785	0.094	0.667
Doing laundry	471	0.832	0.030	0.590	0.834	0.058	0.712
Washing dishes	829	0.856	0.040	0.592	0.859	0.075	0.726
Bathing - Shower	1120	0.896	0.055	0.587	0.899	0.100	0.743
Toilet	1558	0.810	0.036	0.542	0.814	0.067	0.678
Grooming	1775	0.871	0.069	0.531	0.877	0.122	0.704
Dressing	1248	0.869	0.047	0.577	0.872	0.088	0.724
Sleeping	40869	0.895	0.747	0.892	0.896	0.813	0.894
Exercise	5191	0.874	0.197	0.805	0.876	0.317	0.841
Eating	9668	0.708	0.120	0.609	0.715	0.201	0.662
Drinking alcohol	859	0.831	0.094	0.818	0.831	0.169	0.825
Watching TV	8945	0.799	0.238	0.607	0.817	0.342	0.712
Surfing the internet	10668	0.725	0.178	0.464	0.755	0.257	0.609
Talking	18477	0.657	0.218	0.697	0.652	0.332	0.674
Singing	384	0.762	0.057	0.576	0.766	0.103	0.671
With co-workers	3972	0.785	0.194	0.759	0.786	0.309	0.773
With friends	12686	0.665	0.223	0.541	0.685	0.316	0.613
Indoors	102510	0.890	0.992	0.890	0.886	0.938	0.888
Outside	6793	0.880	0.395	0.883	0.880	0.546	0.882
Average		0.819	0.241	0.728	0.821	0.313	0.775

Table 5.32: Recognition scores reported for the CNN-BiLSTM model, for all context labels. Model configuration using CNN layers' `out_channels = 64` for Acc, Gyro, WAcc, Aud. Five-fold cross validation with 48 users in the training set and 12 users in the test set for each iteration.

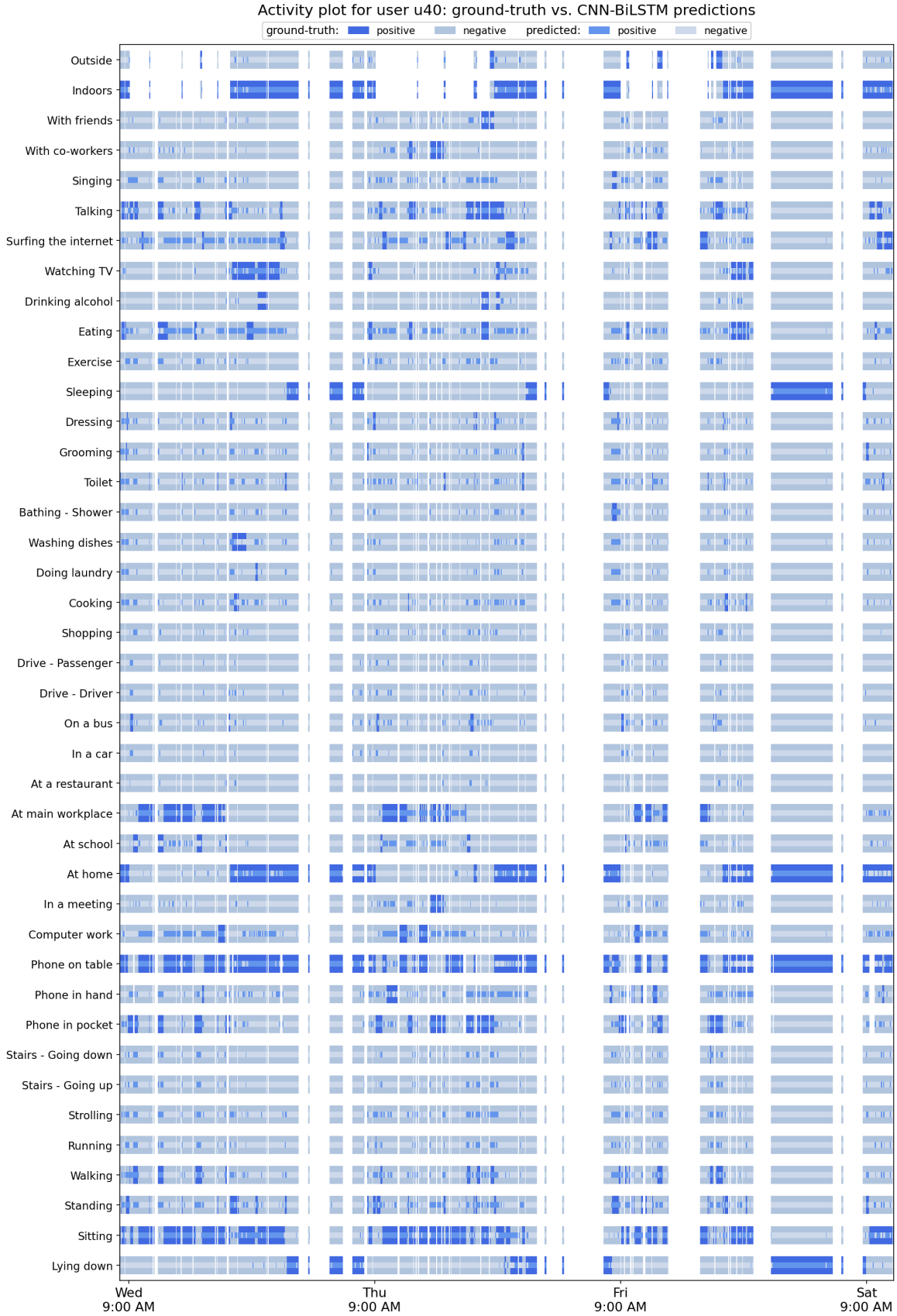


Figure 5.8.3: Activity plot including the predictions of the CNN-BiLSTM model for user u40

Figure 5.8.4: Activity plot including the predictions of the CNN-BiLSTM model for user u45

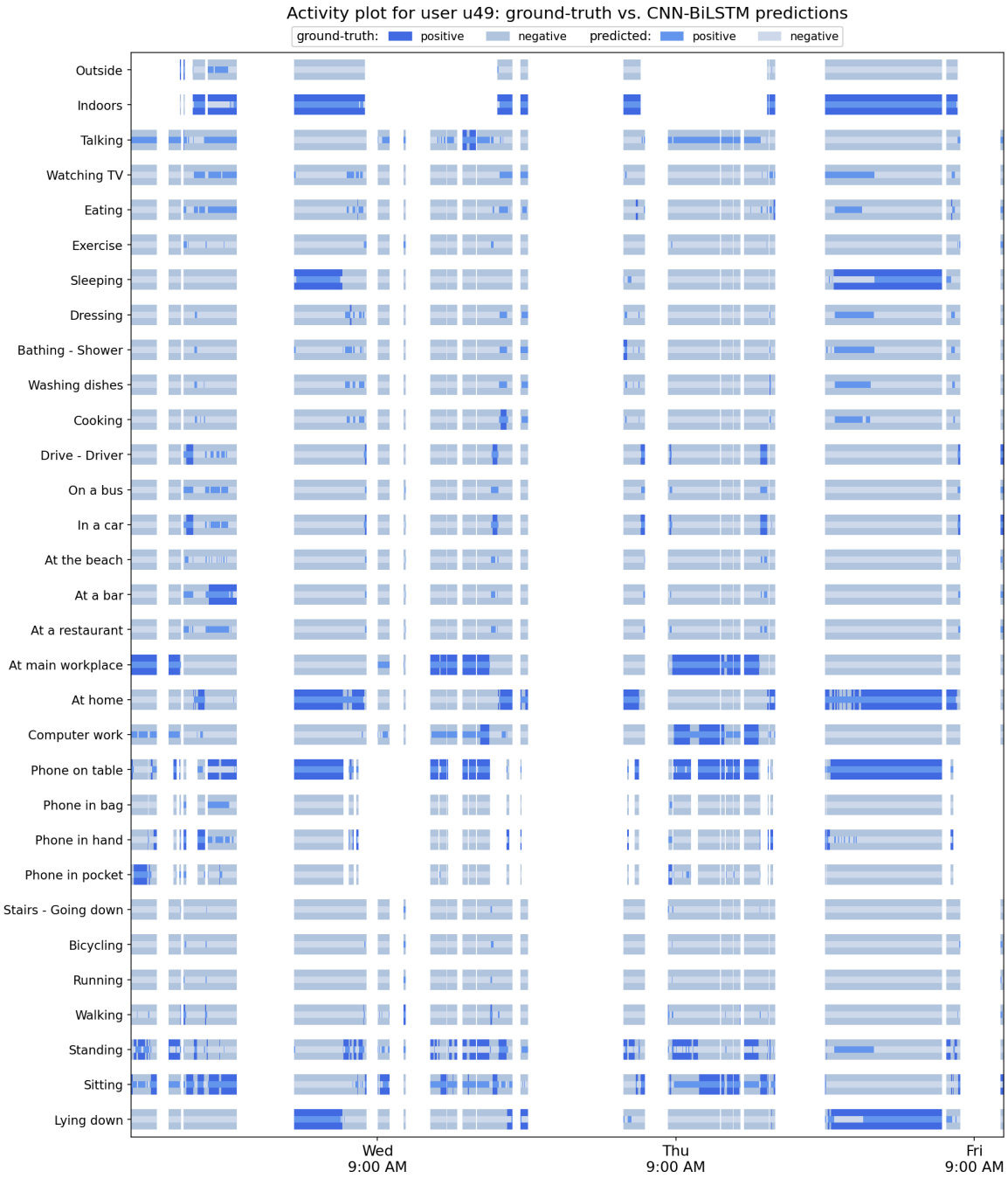


Figure 5.8.5: Activity plot including the predictions of the CNN-BiLSTM model for user u49

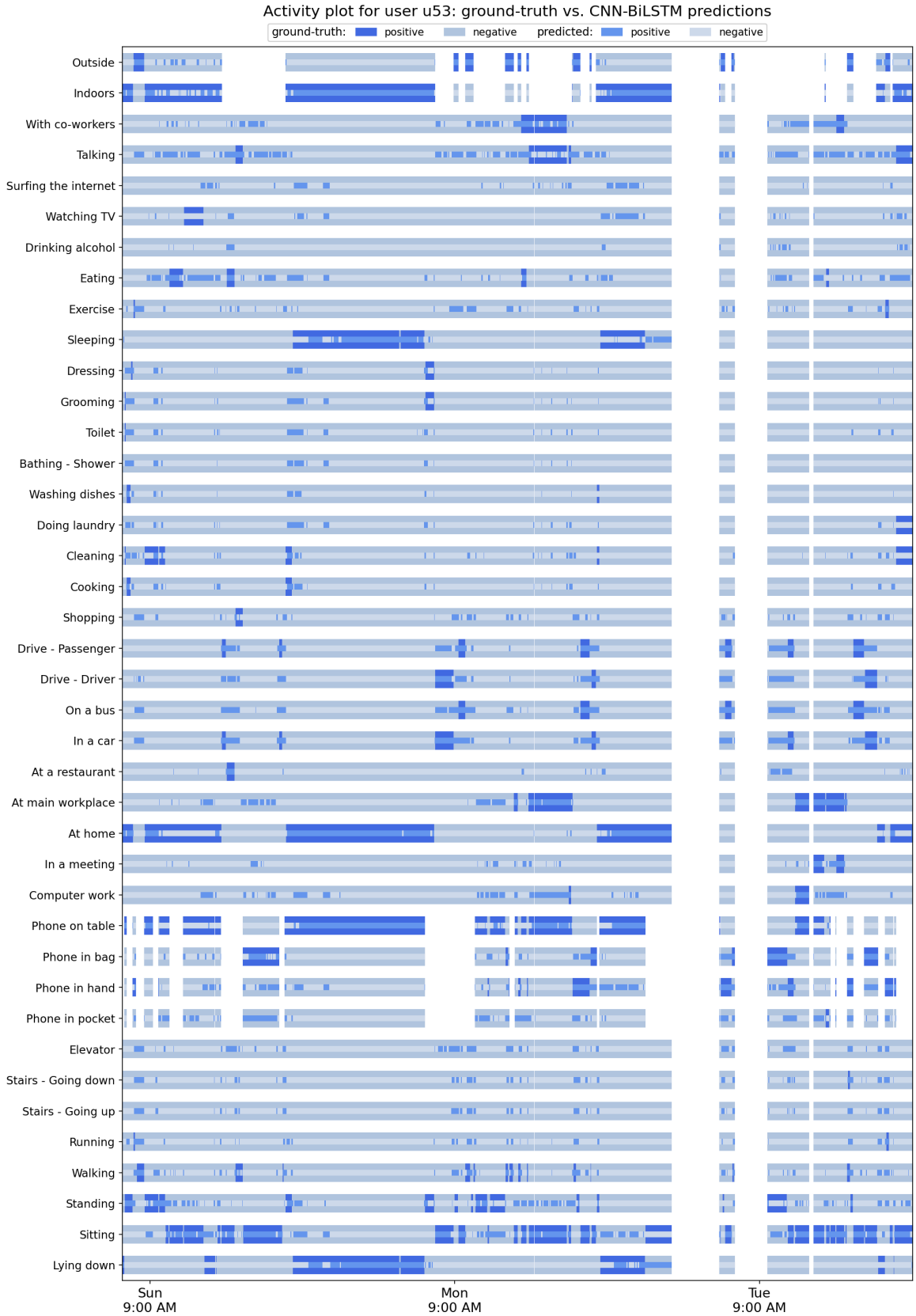


Figure 5.8.6: Activity plot including the predictions of the CNN-BiLSTM model for user u53

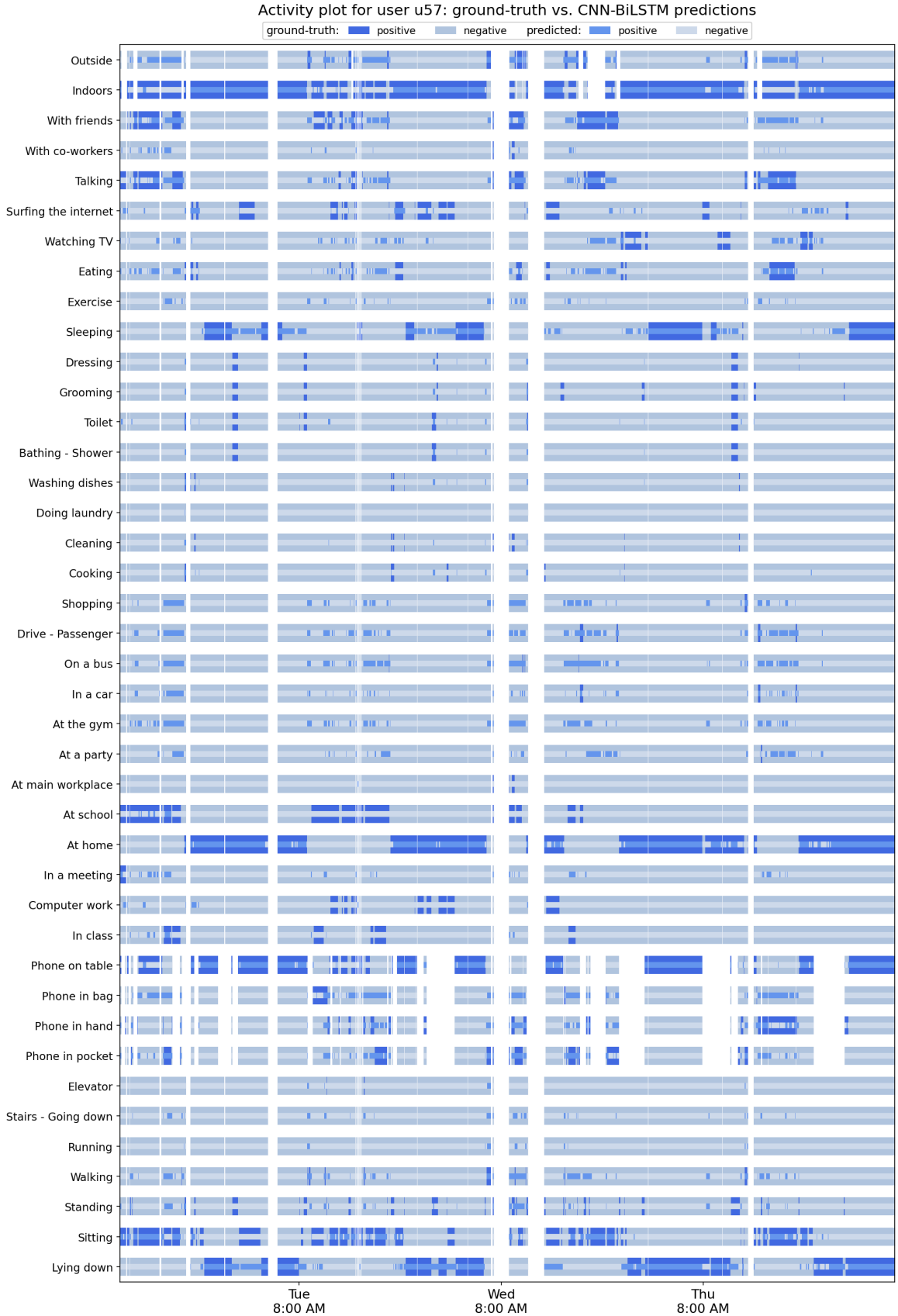


Figure 5.8.7: Activity plot including the predictions of the CNN-BiLSTM model for user u57

5.9 Results Overview

In this Section, in Table 5.33, we gather all the best performance metrics for each of the models we have implemented and tested in this thesis, to summarize our results and extract useful remarks. We include two trivial classifiers to prove that the models we built are significantly better than random guessing or a majority class classifier. We also include the two baselines provided by the researchers who created the ExtraSensory dataset [VEL17; VWL18], which are directly comparable to our work, since we use the same dataset subset, the same dataset split and cross validation scheme, and the same set of 51 target activity and context labels.

We have referenced all the publications which perform HAR tasks using the same dataset in Table 4.4. However, to the best of our knowledge, none of those papers includes an investigation on ML/DL architectures which is directly comparable to ours (which would require the same dataset subset and the same cross validation scheme or test subset in general and the same set of 51 target labels) and also produces results similar or better than the results provided by the original researchers. In many of those papers, a different, smaller test subset of specific users is used (while in our experiments we use cross-validation and eventually all users are included in the test set in one of the five folds), and, also, in many of those papers, HAR is performed using a much smaller subset of target labels.

Comparative results overview of the performance metrics averaged over all labels for the best-performing model configuration for each subsection's architecture

Input	Time series modeling	Model	Accuracy	Precision	Sensitivity	Specificity	F1-score	BA
		Random classifier	0.500	0.110	0.500	0.500	0.137	0.500
		Majority class classifier	0.915	NaN	0.040	0.958	0.037	0.499
Extracted features	Single example (minute)	LR [VWL18]	0.832	-	0.597	0.838	-	0.718
		LR	0.839	0.246	0.612	0.844	0.314	0.728
		MLP [VWL18]	0.773	-	0.773	0.773	-	0.773
		MLP	0.786	0.228	0.757	0.786	0.298	0.772
Extracted features	Sequence of examples (minutes)	BiLSTM (last output)	0.813	0.243	0.753	0.814	0.316	0.784
		BiLSTM (all outputs)	0.810	0.241	0.761	0.811	0.314	0.786
		Self-Attention & BiLSTM	0.818	0.248	0.756	0.819	0.323	0.788
		BiLSTM & Cross-Attention	0.800	0.238	0.767	0.801	0.309	0.784
Raw sensor data	Single example (minute)	CNN	0.782	0.228	0.762	0.781	0.296	0.772
		CNN & Transfomer	0.792	0.229	0.759	0.790	0.300	0.774
Raw sensor data	Sequence of examples (minutes)	CNN + BiLSTM	0.819	0.241	0.728	0.821	0.313	0.775

Table 5.33: An overview of the recognition scores of all models, averaged for all labels

In Table 5.33, we include the best-performing model configuration for each of the models we have implemented and presented in the previous Sections, grouped according to the input type and according to whether we model a single example or a sequence of examples. First, we used the hand-crafted, pre-extracted features provided in the ExtraSensory dataset for each example, and we model a single example in order to predict its activity and context labels, as it was also done in the baselines. In Section 5.4, we have reproduced the baselines, Logistic Regression models per-label, and a universal MLP(16,16) which includes two hidden layers of 16 nodes.

Subsequently, we have continued using the hand-crafted features but we now model a sequence of examples, in order to predict the labels of the last example by providing context of the immediate past. In Section 5.5, we have used a two-layer BiLSTM to model the input sequence of examples, and we have tested two models, one using only the final hidden states of the two BiLSTM layers, and one using the outputs of the last BiLSTM layer for all timesteps. We got the best results for both models, when providing an input sequence of length `window_len = 5` and when using BiLSTM layers' `hidden_size = 64`. We notice that the respective model BA scores of 0.784 and 0.786 are an improvement over the baseline scores, which means that providing a sequence of 5 examples (minutes) including the current example for which we want to predict the labels, is helpful for our model. We have also experimented with longer sequences of examples, and it was found that as we increase the sequence length up to 30 examples, the performance of the model decreases.

Our next experiments, in Section 5.6, were attempts to further enhance the BiLSTM model. By preceding the two-layer BiLSTM with a Self-Attention module, we get a further improvement on the BA, which reaches 0.788 when using the best-performing two-layer BiLSTM using only final hidden states, and a Multi-head Self-Attention module with `num_heads = 5`. Another model that we have implemented, includes a Multi-head Cross-Attention module after the two-layer BiLSTM. The output of the BiLSTM for all timesteps is used to produce the query `Q`, while the input features are used to produce both key `K` and value `V` in the Cross-Attention module. This model has similar performance to the simple BiLSTM model, but we use it to visualize and study the attention weights during inference, for model interpretability.

The next step along the way is to start using the raw sensor measurements for each example, instead of the hand-crafted features. In Section 5.7 we propose two pipelines based on Deep Learning for feature extraction from raw sensor data. The first is a CNN-based pipeline for each of the raw sensor modalities, that contains stacked CNN layers (which include: 2D/1D Conv layer, leaky ReLU, Batch Normalization, and Dropout). The second one is a CNN-Transformer-based pipeline for each of the raw sensor modalities, that contains four stacked CNN layers followed by two Transformer Encoder layers. In both cases, an MLP(16,16) takes the extracted features as input to predict the context labels. Although these feature extraction pipelines produced the best results among all model architectures and configurations that we tested for Deep Learning-based feature extraction on this dataset, they still do not outperform the results of the baseline MLP(16,16) using the hand-crafted features.

Our last experiment, in Section 5.8, is based again on using the raw sensor measurements, but now for a sequence of examples. We use Time-Distributed CNN pipelines to extract features for all the input sequence examples, and we model the sequence of extracted features with a two-layer BiLSTM, as previously. However, in this case, we get no substantial improvement in the resulting BA of 0.775 compared to the BA of 0.772 the CNN-based model for a single example. Again, we have extensively experimented with the hyperparameters of this model, but we did not succeed in producing better results than the ones reported here.

We notice that, when using the raw sensor data and Deep Learning feature extraction pipelines integrated in our models, we did not succeed to build models that significantly outperform the baseline MLP(16,16). This struggle can possibly be attributed to at least two factors we can think of. The first factor is that it is much more strenuous, and as a result, much harder, to fine-tune the hyperparameters of deep models which contain numerous layers, with a lot of configurable parameters each. Optimizing these hyperparameters from scratch is very time-consuming and resource-consuming, and it is not straightforward how to do it in a smart and efficient way. Thus, one reason for not getting very improved results might be that, although we did an extensive search to tune the hyperparameter values, there might have been cues that we missed and we might not have found the best hyperparameter values for the specific combination of model and dataset. In the models using hand-crafted features, where the number of hyperparameters to be optimized was smaller, it was more straightforward to tune them and we got better results much easier.

The second factor is that there is a conundrum on whether hand-crafted features or deep neural representations are better suited for HAR tasks, especially for data collected out of the lab, and also in cases of using unseen users during inference, for which the data distribution's distance from the distribution of the training data might be greater. Research on this topic has been conducted by Bento *et al.* [Ben+22] and it was found that when testing on unseen users or datasets, the models using hand-crafted features as input, tend to perform better. Our findings are in accordance with these results. Moreover, since we use a dataset which includes many more labels and is very unbalanced, for a lot of the labels we have very few examples and it is intuitive to think that the representations extracted by the deep neural pipelines for these

labels won't be robust. Since in our experiments all labels have the same weight when calculating the average metrics for all labels, irrespectively of the number of samples which have been annotated with each label, it is important for the model to predict both common labels and rare labels as accurately as possible, to produce better average recognition metrics.

Overall, by observing Table 5.33, we can see that there is a trade-off between Sensitivity and Specificity, that was also mentioned in previous commentary of model results, earlier in this thesis. We notice that although models better suited for this task might produce higher values in both metrics, compared to baseline or weaker models, when we train a specific model with different hyperparameter configurations or when we train similar models, when the resulting recognition Sensitivity is higher, the resulting Specificity is usually lower. In addition, we notice that for all models, the Sensitivity values are not significantly improved, and range in values up to 0.773. However, the Specificity values, especially in the cases where we use BiLSTM models and sequences of consecutives examples as input, rise up to 0.810-0.820, which means that on average, the False Positive predictions of these models are significantly decreased, compared to the baseline MLP(16,16). In general, since we have tested a wide range of models and we could not get any BA values greater than a value of about 0.788, we conclude that either the task is inherently flawed because of all the dataset's noise and imperfections and the margin for improvement is relatively small by default, or a radical change in our approach to the specific HAR task is required to produce greater improvements in the recognition metrics, or both. We present ideas for future research in Section 6.2.

Chapter 6

Conclusion

6.1 Contributions

In this Thesis, we have conducted an extensive study on neural network architectures for Human Activity Recognition using data collected *in-the-wild* from wearable devices. We have used the ExtraSensory dataset which includes data collected from smartphone and smartwatch sensors, from 60 users totaling over 300k examples (minutes) accompanied with 51 activity and context labels, while each example is annotated with all relevant labels. We have implemented and trained multiple general-puprose HAR models, in a user-agnostic setup, and we always test our models in users unseen during training, which makes our task even more challenging. The contributions of our work are summarized below:

- We have extensively studied the relevant literature on Human Activity Recognition based on non-visual, wearable sensors, and we have presented a thorough literature review on the Activity Recognition Chain, a standard pipeline which includes sensor data preprocessing, segmentation, feature extraction and classification. We have listed the available open-source datasets, and we have reported major HAR challenges, trade-offs and open problems. We have also studied the architecture and application of popular Deep Learning models, including DNNs, CNNs, RNNs (LSTM, GRU), Attention and Transformers, on HAR tasks.
- We have delved into HAR *in-the-wild*, which transfers the data collection process from the scripted and controlled scenarios in a lab environment, to totally uncontrolled real-life settings. We have familiarized ourselves with data collected *in-the-wild*, namely the ExtraSensory dataset, which is very rich in content and labels, and better reflects the activities of daily living, but leads to HAR tasks of increased complexity because of its inherent characteristics, which include extreme label imbalance, missing sensor data, missing labels, noisy sensor data, wrongly-annotated data, inter-personal and intra-personal variability.
- We have proceeded to conduct a systematic and widespread exploration of Machine Learning and Deep Learning models for general-purpose multi-label, multi-task HAR *in-the-wild* using the ExtraSensory dataset and the aforementioned 51 activity and context labels. A detailed overview of all our experiments and results can be found in Section 5.9. Concisely, we have reproduced the best available baselines, which use hand-crafted features, and we have proposed several models for which we report improvements on the balanced accuracy (BA), compared to the baselines. These models are based on a two-layer BiLSTM which takes as input a sequence of consecutive examples, again using the pre-extracted, hand-crafted features. A Self-Attention module before the two-layer BiLSTM was also found to be useful and further improved the BA metric. We have also experimented with using the raw sensor measurements instead of the hand-crafted features and extracting features via a CNN-based or CNN-Transformer-based pipeline integrated in our model, for a single example, or combined with a BiLSTM to model a sequence of examples. However, these experiments did not produce much better results than our baselines.
- We have tried to shed some light on the functionality of our models, by implementing a model with this specific purpose. We have added a Cross-Attention mechanism after the two-layer BiLSTM model, and we use the BiLSTM outputs for all timesteps to produce the Attention queries and the input sequnce features to produce the Attention keys and values. At inference time, we save the Attention weights for each example and the respective predictions of the model, while we also have the ground-truth labels of each example available. In Subsection 5.6.3, we have visualized the Attention weights and we have thorougly investigated whether they can provide useful insights about which features are more impactful in the model's predictions for various labels, and also whether these predictions are meaningful or if the model is misdirected by false correlations between input features and context labels.
- Along the course of studying for this thesis, exploring the dataset and implementing the aforementioned models, we have created the Github repository `alexvioni/ExtraSensory-functionality` which includes and supports all modules from dataloading, data management, feature exploration and visualization to the design and implementation of ML/DL models and all the required scripts to train, test and evaluate them, including metrics and plots. This repository is private as of October 2023, but might be released to the public at a later date, and it is a ready-to-use framework that can be used for a wide range of tasks, enabling quick integration of more features and models and fast expansion to more datasets, domains or tasks.

6.2 Future Work

We present some ideas for future research based on the insights acquired during studying and conducting experiments for this Thesis:

- As previously mentioned, in our work we have implemented and trained multiple general-purpose HAR models, which can be used in any case of multi-label, multi-task HAR, with arbitrary number of labels, since we do not impose any constraints on label prediction. However, in many HAR tasks we can take advantage of specific label connections, such as taxonomies of activities or contexts, hierarchical relations between labels, or labels that cannot coexist, to better define the loss that is used to train the model and improve model performance. For example, if we know that some labels are mutually exclusive, we can use the Softmax activation function since we only need to predict one label, the predominant one, out of the mutually exclusive ones.
- In addition, our models have been user-agnostic, a.k.a. not conditioned on the user, and we always test our models on users unseen during training to ensure that they can generalize adequately. However, since each user has his/her own habits and routines and might perform activities in his/her own way, user personalization could be beneficial for our model and could possibly improve model performance. Considering that it is difficult to collect annotated human activity data from each new user that we want to personalize in order to train the model from scratch using only his/her data, we can use a small amount of data to fine-tune an already trained model, on the specific user. This approach combines improved HAR performance and the need for only a small amount of annotated data from each new user, and the resulting model will be adapted to a specific user, and available in his/her wearable device.
- Another interesting direction would be to find unsupervised or semi-supervised ways, e.g., clustering, to determine which ground-truth labels of the dataset are correct and which examples have been wrongly-annotated accidentally. This process would enormously reduce the need for manual cleaning and model training would benefit from a "cleaner" or better curated dataset. Also, instead of discarding the examples or labels that were found to be unreliable, the loss could be adapted using weights according to the confidence of each label for the specific example.
- Our next idea is based on the fact that although we mitigate the fact that many of the dataset's labels are rare by using inverse label weights in the loss function and we manage to guide the model to predict all labels and not only the most common ones, the scarcity of examples annotated with the rare labels results in systems that struggle to model these labels. We propose to incorporate modules which can assist in this task, by using more labeled datasets to increase the examples annotated by rare labels, or by leveraging unlabeled datasets using self-supervised learning to extract meaningful representations. Furthermore, we could expand the aforementioned conundrum on whether hand-crafted features or deep neural representations are better suited for HAR on unseen users, by including representations extracted from models based on self-supervised learning, e.g., [Yua+23], in the discussion.
- Another proposal for future work would be to take advantage of this dataset which is very rich in sensor sources, data and annotations, and transfer the knowledge to models that will use a single sensor modality during inference time. It would be very interesting to see if we could build and train models that would achieve similar recognition performance using less sensors or even only a single sensor. In the same direction, it would also be very interesting to investigate whether we can apply a model trained on the ExtraSensory dataset on a very different dataset from a different domain, e.g., the E-Prevention dataset [Zla+22], and to evaluate its activity recognition potential on out-of-domain datasets without ground-truth activity labels.
- Finally, in order for the model to be used in real-life applications, real-time inference is required. Still, our model uses sensor recordings of about 20sec in order to predict labels for each minute of daily living. It remains to be investigated whether the model can predict labels with similar accuracy using much shorter sensor recordings, or to explore other options for activity recognition on the fly. Additionally, the model's accuracy must be improved in order to be used seamlessly in daily life. Since we have seen that it is very hard to improve the BA, and there is a trade-off between Sensitivity and Specificity, in some cases we could choose to optimize Sensitivity, if we care more about eliminating the False Negatives, e.g., in a healthcare application, or Specificity, if we prefer to eliminate the False Positives.

Bibliography

[Ach+12] Achumba, I. E. et al. "On time series sensor data segmentation for fall and activity classification". In: *2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE, Oct. 2012. DOI: 10.1109/healthcom.2012.6379453.

[AT19] Adaimi, R. and Thomaz, E. "Leveraging active learning and conditional mutual information to minimize data annotation in human activity recognition". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3.3 (2019), pp. 1–23.

[Agh+07] Aghajan, H. et al. "Distributed Vision-Based Accident Management for Assisted Living". In: *Pervasive Computing for Quality of Life Enhancement*. Vol. 4541. Springer Berlin Heidelberg, 2007, pp. 196–205. DOI: 10.1007/978-3-540-73035-4_21.

[AFB23] Akram, A., Farhan, A. A., and Basharat, A. "Less is more: Efficient behavioral context recognition using Dissimilarity-Based Query Strategy". In: *Plos one* 18.6 (2023), e0286919.

[Ale+22] Alexiadis, A. et al. "A Sensor-Independent Multimodal Fusion Scheme for Human Activity Recognition". In: *International Conference on Pattern Recognition and Artificial Intelligence*. Springer. 2022, pp. 28–39.

[Ali+15] Ali, R. et al. "KARE: a hybrid reasoning approach for promoting active lifestyle". In: *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication - IMCOM'15*. ACM Press, 2015. DOI: 10.1145/2701126.2701156.

[AAA17] Almaslukh, B., AlMuhtadi, J., and Artoli, A. "An Effective Deep Autoencoder Approach for Online Smartphone-Based Human Activity Recognition". In: *International Journal of Computer Science and Network Security* 17 (Apr. 2017).

[Als+15] Alsheikh, M. A. et al. *Deep Activity Recognition Models with Triaxial Accelerometers*. 2015. arXiv: 1511.04664v2 [cs.LG].

[Als+13] Alshurafa, N. et al. "Robust human intensity-varying activity recognition using Stochastic Approximation in wearable sensors". In: *2013 IEEE International Conference on Body Sensor Networks*. IEEE, May 2013. DOI: 10.1109/bsn.2013.6575515.

[ABT10] Altun, K., Barshan, B., and Tunçel, O. "Comparative study on classifying human activities with miniature inertial and magnetic sensors". In: *Pattern Recognition* 43.10 (Oct. 2010), pp. 3605–3620. DOI: 10.1016/j.patcog.2010.04.019.

[AT14] Ann, O. C. and Theng, L. B. "Human activity recognition: A review". In: *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*. IEEE, Nov. 2014. DOI: 10.1109/iccsce.2014.7072750.

[Ard+20] Ardywibowo, R. et al. *Dynamic Feature Selection for Efficient and Interpretable Human Activity Recognition*. 2020.

[ACB23] Arrotta, L., Civitarese, G., and Bettini, C. *Neuro-Symbolic Approaches for Context-Aware Human Activity Recognition*. 2023. arXiv: 2306.05058 [cs.LG].

[Arr+23] Arrotta, L. et al. "DOMINO: A Dataset for Context-Aware Human Activity Recognition using Mobile Devices". In: *Proceedings of the 24th International Conference on Mobile Data Management (MDM)*. IEEE Computer Society. 2023.

[AS03] Ashbrook, D. and Starner, T. "Using GPS to learn significant locations and predict movement across multiple users". In: *Personal and Ubiquitous Computing* 7.5 (Oct. 2003), pp. 275–286. DOI: 10.1007/s00779-003-0240-0.

[Asi+20] Asim, Y. et al. "Context-Aware Human Activity Recognition (CAHAR) in-the-Wild using Smartphone Accelerometer". In: *IEEE Sensors Journal* 20.8 (2020), pp. 4361–4371.

[Avc+10] Avci, A. et al. "Activity Recognition Using Inertial Sensing for Healthcare, Wellbeing and Sports Applications: A Survey." In: *ARCS Workshops*. Ed. by M. Beigl and F. J. Cazorla-Almeida. VDE Verlag, 2010, pp. 167–176. ISBN: 978-3-8007-3222-7.

[AS17] Azmi, M. S. M. and Sulaiman, M. N. "Accelerator-Based Human Activity Recognition Using Voting Technique with NBTree and MLP Classifiers". In: *International Journal on Advanced Science, Engineering and Information Technology* 7.1 (Feb. 2017), p. 146. DOI: 10.18517/ijaseit.7.1.1790.

[Bac+10] Bachlin, M. et al. "Wearable Assistant for Parkinson's Disease Patients With the Freezing of Gait Symptom". In: *IEEE Transactions on Information Technology in Biomedicine* 14.2 (Mar. 2010), pp. 436–446. DOI: 10.1109/titb.2009.2036165.

[BCB15] Bahdanau, D., Cho, K. H., and Bengio, Y. "Neural machine translation by jointly learning to align and translate". In: *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

[BGS14] Baños, O., Garcia, R., and Saez, A. *MHEALTH Dataset Data Set (UCI Machine Learning Repository)*. 2014.

[BTA12] Baños, O., Toth, M. A., and Amft, O. *REALDISP Activity Recognition Dataset Data Set (UCI Machine Learning Repository)*. 2012.

[Bañ+12] Baños, O. et al. "A benchmark dataset to evaluate sensor displacement in activity recognition". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*. ACM Press, 2012. DOI: 10.1145/2370216.2370437.

[Bañ+14a] Baños, O. et al. "mHealthDroid: A Novel Framework for Agile Development of Mobile Health Applications". In: *Ambient Assisted Living and Daily Activities*. Springer International Publishing, 2014, pp. 91–98. DOI: 10.1007/978-3-319-13105-4_14.

[Bañ+14b] Baños, O. et al. "Window Size Impact in Human Activity Recognition". In: *Sensors* 14.4 (Apr. 2014), pp. 6474–6499. DOI: 10.3390/s140406474.

[BI04] Bao, L. and Intille, S. S. "Activity Recognition from User-Annotated Acceleration Data". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 1–17. DOI: 10.1007/978-3-540-24646-6_1.

[BA10] Barshan, B. and Altun, K. *Daily and Sports Activities Data Set (UCI Machine Learning Repository)*. 2010.

[Ben09] Bengio, Y. "Learning Deep Architectures for AI". In: *Foundations and Trends® in Machine Learning* 2.1 (2009), pp. 1–127. DOI: 10.1561/2200000006.

[BSF94] Bengio, Y., Simard, P., and Frasconi, P. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (Mar. 1994), pp. 157–166. DOI: 10.1109/72.279181.

[Ben+22] Bento, N. et al. "Comparing Handcrafted Features and Deep Neural Representations for Domain Generalization in Human Activity Recognition". In: *Sensors* 22.19 (2022). ISSN: 1424-8220. DOI: 10.3390/s22197324.

[BAT22] Bhattacharya, S., Adaimi, R., and Thomaz, E. "Leveraging sound and wrist motion to detect activities of daily living with commodity smartwatches". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6.2 (2022), pp. 1–28.

[BL16] Bhattacharya, S. and Lane, N. D. "From smart to deep: Robust activity recognition on smartwatches using deep learning". In: *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, Mar. 2016. DOI: 10.1109/percomw.2016.7457169.

[Bha+14] Bhattacharya, S. et al. "Using unlabeled data in a sparse-coding framework for human activity recognition". In: *Pervasive and Mobile Computing* 15 (Dec. 2014), pp. 242–262. DOI: 10.1016/j.pmcj.2014.05.006.

[BS08] Blanke, U. and Schiele, B. "Sensing Location in the Pocket". In: *In Proceedings of Ubiquitous Computing 2008, Seoul, Korea*. 2008, pp. 21–24.

[Bra+20] Brahim, G. B. et al. "A New Semantic-based Multi-Level Classification Approach for Activity Recognition Using Smartphones". In: *International Journal of Software Engineering and Knowledge Engineering* 30.08 (2020), pp. 1051–1078.

[BBS14] Bulling, A., Blanke, U., and Schiele, B. "A tutorial on human activity recognition using body-worn inertial sensors". In: *ACM Computing Surveys* 46.3 (Jan. 2014), pp. 1–33. DOI: 10.1145/2499621.

[BWG13] Bulling, A., Weichel, C., and Gellersen, H. "EyeContext: Recognition of high-level contextual cues from human visual behaviour". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, 2013. DOI: 10.1145/2470654.2470697.

[CD21] Campana, M. G. and Delmastro, F. "COMPASS: Unsupervised and online clustering of complex human activities from smartphone sensors". In: *Expert Systems with Applications* 181 (2021), p. 115124. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115124.

[CPH05] Carreira-Perpiñán, M. Á. and Hinton, G. E. "On contrastive divergence learning". In: *Proceedings of the tenth international workshop on artificial intelligence and statistics (AISTATS)*. 2005, pp. 33–40.

[Cha+17] Chatzaki, C. et al. "Human Daily Activity and Fall Recognition Using a Smartphone's Acceleration Sensor". In: *Communications in Computer and Information Science*. Springer International Publishing, 2017, pp. 100–118. DOI: 10.1007/978-3-319-62704-5_7.

[Cha+13] Chavarriaga, R. et al. "The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition". In: *Pattern Recognition Letters* 34.15 (Nov. 2013), pp. 2033–2042. DOI: 10.1016/j.patrec.2012.12.014.

[CJK15] Chen, C., Jafari, R., and Kehtarnavaz, N. "UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor". In: *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2015. DOI: 10.1109/icip.2015.7350781.

[Che+12] Chen, L. et al. "Sensor-Based Activity Recognition". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (Nov. 2012), pp. 790–808. DOI: 10.1109/tsmcc.2012.2198883.

[CR19] Chen, Y. and Rangwala, H. "Attention-based multi-task learning for sensor analytics". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 2187–2196.

[Che+20] Chen, Y. et al. "Federated multi-task learning with hierarchical attention for sensor data analytics". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[CX15] Chen, Y. and Xue, Y. "A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer". In: *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, Oct. 2015. DOI: 10.1109/smc.2015.263.

[Cho+14] Cho, K. et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014. DOI: 10.3115/v1/d14-1179.

[Chu+23] Chui, K. T. et al. "A Convolutional Neural Network-Based Feature Extraction and Weighted Twin Support Vector Machine Algorithm for Context-Aware Human Activity Recognition". In: *Electronics* 12.8 (2023), p. 1915.

[Chu+22] Chung, S. et al. "Real-world multimodal lifelog dataset for human behavior study". In: *ETRI Journal* 44.3 (2022), pp. 426–437.

[Cru+19] Cruciani, F. et al. "A public domain dataset for human activity recognition in free-living conditions". In: *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE. 2019, pp. 166–171.

[Cru+20a] Cruciani, F. et al. "Feature learning for Human Activity Recognition using Convolutional Neural Networks: A case study for Inertial Measurement Unit and audio data". In: *CCF Transactions on Pervasive Computing and Interaction* 2.1 (2020), pp. 18–32.

[Cru+20b] Cruciani, F. et al. "Personalizing activity recognition with a clustering based semi-population approach". In: *IEEE Access* 8 (2020), pp. 207794–207804.

[Csá01] Csáji, B. C. "Approximation with Feed-Forward Artificial Neural Networks". (thesis written at the Eindhoven University of Technology). MA thesis. Budapest, Hungary: (Computer Science with Mathematics major), Faculty of Science, Eötvös Loránd University, 2001.

[Den14] Deng, L. "Deep Learning: Methods and Applications". In: *Foundations and Trends® in Signal Processing* 7.3-4 (2014), pp. 197–387. DOI: 10.1561/2000000039.

[DeO+22] DeOliveira, J. et al. "HAR-CTGAN: A Mobile Sensor Data Generation Tool for Human Activity Recognition". In: *2022 IEEE International Conference on Big Data (Big Data)*. IEEE. 2022, pp. 5233–5242.

[Der+12] Dernbach, S. et al. "Simple and Complex Activity Recognition through Smart Phones". In: *2012 Eighth International Conference on Intelligent Environments*. IEEE, June 2012. DOI: 10.1109/ie.2012.39.

[DLKP22] Dirgová Luptáková, I., Kubovčík, M., and Pospíchal, J. "Wearable Sensor-Based Human Activity Recognition with Transformer Model". In: *Sensors* 22.5 (2022). ISSN: 1424-8220. DOI: 10.3390/s22051911.

[EK16] Edel, M. and Koppe, E. "Binarized-BLSTM-RNN based Human Activity Recognition". In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, Oct. 2016. DOI: 10.1109/ipin.2016.7743581.

[Erm+08] Ermes, M. et al. "Detection of Daily Activities and Sports With Wearable Sensors in Controlled and Uncontrolled Conditions". In: *IEEE Transactions on Information Technology in Biomedicine* 12.1 (Jan. 2008), pp. 20–26. DOI: 10.1109/titb.2007.899496.

[Fat21] Fatima, S. "Activity recognition in older adults with training data from younger adults: preliminary results on in vivo smartwatch sensor data". In: *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. 2021, pp. 1–4.

[Faw06] Fawcett, T. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (June 2006), pp. 861–874. DOI: 10.1016/j.patrec.2005.10.010.

[Faz+21] Fazli, M. et al. "HHAR-net: Hierarchical Human Activity Recognition using Neural Networks". In: *Intelligent Human Computer Interaction: 12th International Conference, IHCI 2020, Daegu, South Korea, November 24–26, 2020, Proceedings, Part I 12*. Springer. 2021, pp. 48–58.

[Fig+10] Figo, D. et al. "Preprocessing techniques for context recognition from accelerometer data". In: *Personal and Ubiquitous Computing* 14.7 (Mar. 2010), pp. 645–662. DOI: 10.1007/s00779-010-0293-9.

[Fin14] Fink, G. A. *Markov Models for Pattern Recognition*. Springer London, 2014. DOI: 10.1007/978-1-4471-6308-4.

[FSF99] Foerster, F., Smeja, M., and Fahrenberg, J. "Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring". In: *Computers in Human Behavior* 15.5 (Sept. 1999), pp. 571–583. DOI: 10.1016/s0747-5632(99)00037-0.

[FHT00] Friedman, J., Hastie, T., and Tibshirani, R. "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)". In: *The Annals of Statistics* 28.2 (Apr. 2000), pp. 337–407. DOI: 10.1214/aos/1016218223.

[GA22] Ge, W. and Agu, E. O. "QCRUFT: Quaternion Context Recognition under Uncertainty using Fusion and Temporal Learning". In: *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*. IEEE. 2022, pp. 41–50.

[Ge+23] Ge, W. et al. "Heterogeneous Hyper-Graph Neural Networks for Context-aware Human Activity Recognition". In: *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE. 2023, pp. 350–354.

[GSC00] Gers, F. A., Schmidhuber, J., and Cummins, F. "Learning to Forget: Continual Prediction with LSTM". In: *Neural Computation* 12.10 (Oct. 2000), pp. 2451–2471. DOI: 10.1162/089976600300015015.

[Giu+22] Giunchiglia, F. et al. *A survey on students' daily routines and academic performance at the University of Trento*. 2022.

[GBC16] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

[Goo+10] Goodwin, M. S. et al. "Automated Detection of Stereotypical Motor Movements". In: *Journal of Autism and Developmental Disorders* 41.6 (Sept. 2010), pp. 770–782. DOI: 10.1007/s10803-010-1102-z.

[Gre+17] Greff, K. et al. "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (Oct. 2017), pp. 2222–2232. DOI: 10.1109/tnnls.2016.2582924.

[GP17] Guan, Y. and Plötz, T. "Ensembles of Deep LSTM Learners for Activity Recognition using Wearables". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1.2 (June 2017), pp. 1–28. DOI: 10.1145/3090076.

[Gue+09] Guenterberg, E. et al. "An Automatic Segmentation Technique in Body Sensor Networks based on Signal Energy". In: *Proceedings of the 4th International ICST Conference on Body Area Networks*. ICST, 2009. DOI: 10.4108/icst.bodynets2009.6036.

[Guo+14] Guo, J. et al. "Structural health monitoring by using a sparse coding-based deep learning algorithm with wireless sensor networks". In: *Personal and Ubiquitous Computing* 18.8 (Aug. 2014), pp. 1977–1987. DOI: 10.1007/s00779-014-0800-5.

[GD14] Gupta, P. and Dallas, T. "Feature Selection and Activity Recognition System Using a Single Triaxial Accelerometer". In: *IEEE Transactions on Biomedical Engineering* 61.6 (June 2014), pp. 1780–1786. DOI: 10.1109/tbme.2014.2307069.

[HC16] Ha, S. and Choi, S. "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2016. DOI: 10.1109/ijcnn.2016.7727224.

[HYC15] Ha, S., Yun, J.-M., and Choi, S. "Multi-modal Convolutional Neural Networks for Activity Recognition". In: *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, Oct. 2015. DOI: 10.1109/smc.2015.525.

[Hal99] Hall, M. A. "Correlation-based Feature Selection for Machine Learning". PhD Thesis. Department of Computer Science, The University of Waikato, 1999.

[HHP16] Hammerla, N. Y., Halloran, S., and Ploetz, T. *Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables*. 2016. arXiv: 1604.08880v1 [cs.CV].

[Ham+15] Hammerla, N. Y. et al. "PD Disease State Assessment in Naturalistic Environments Using Deep Learning". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 1742–1748. ISBN: 0262511290.

[HWZ23] Hao, Y., Wang, B., and Zheng, R. "VALERIAN: Invariant Feature Learning for IMU Sensor-based Human Activity Recognition in the Wild". In: *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*. 2023, pp. 66–78.

[HA20] Haq, M. Ehatisham-ul and Azam, M. A. "Opportunistic sensing for inferring in-the-wild human contexts based on activity pattern recognition using smart computing". In: *Future Generation Computer Systems* 106 (2020), pp. 374–392.

[Har14] Harasimowicz, A. "Comparison of Data Preprocessing Methods and the Impact on Autoencoder's Performance in Activity Recognition Domain". In: *ICT Young 2014*. Sept. 2014.

[Har+20a] Harris, C. R. et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[Har+20b] Hartvigsen, T. et al. "Recurrent halting chain for early multi-label classification". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 1382–1392.

[Hay09] Haykin, S. S. *Neural Networks and Learning Machines*. Third. Upper Saddle River, NJ: Pearson Education, 2009.

[HJ09] He, Z. and Jin, L. "Activity recognition from acceleration data based on discrete consine transform and SVM". In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, Oct. 2009. DOI: 10.1109/icsmc.2009.5346042.

[Hei+06] Heinz, E. A. et al. "Using Wearable Sensors for Real-Time Recognition Tasks in Games of Martial Arts - An Initial Experiment". In: *2006 IEEE Symposium on Computational Intelligence and Games*. IEEE, May 2006. DOI: 10.1109/cig.2006.311687.

[HOT06] Hinton, G. E., Osindero, S., and Teh, Y.-W. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18.7 (July 2006), pp. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527.

[HS97] Hochreiter, S. and Schmidhuber, J. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[Huo+22] Huo, Z. et al. "Dynimp: Dynamic Imputation for Wearable Sensing Data through Sensory and Temporal Relatedness". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 3988–3992.

[HFS08] Huynh, T., Fritz, M., and Schiele, B. "Discovery of activity patterns using topic models". In: *Proceedings of the 10th international conference on Ubiquitous computing - UbiComp '08*. ACM Press, 2008. DOI: 10.1145/1409635.1409638.

[IIN16] Inoue, M., Inoue, S., and Nishida, T. *Deep Recurrent Neural Network for Mobile Human Activity Recognition with High Throughput*. 2016. arXiv: 1611.03607v1 [cs.CV].

[JY15] Jiang, W. and Yin, Z. "Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks". In: *Proceedings of the 23rd ACM international conference on Multimedia - MM '15*. ACM Press, 2015. DOI: 10.1145/2733373.2806333.

[JZS15] Józefowicz, R., Zaremba, W., and Sutskever, I. "An Empirical Exploration of Recurrent Network Architectures". In: *International Conference on Machine Learning (ICML)*. 2015, pp. 2342–2350.

[Kas+08] Kasteren, T. van et al. "Accurate activity recognition in a home setting". In: *Proceedings of the 10th international conference on Ubiquitous computing - UbiComp '08*. ACM Press, 2008. DOI: 10.1145/1409635.1409637.

[Kat+70] Katz, S. et al. "Progress in Development of the Index of ADL". In: *The Gerontologist* 10.1 Part 1 (Mar. 1970), pp. 20–30. DOI: 10.1093/geront/10.1_part_1.20.

[Kat83] Katz, S. "Assessing Self-maintenance: Activities of Daily Living, Mobility, and Instrumental Activities of Daily Living". In: *Journal of the American Geriatrics Society* 31.12 (Dec. 1983), pp. 721–727. DOI: 10.1111/j.1532-5415.1983.tb03391.x.

[KA76] Katz, S. and Akpom, C. A. "A Measure of Primary Sociobiological Functions". In: *International Journal of Health Services* 6.3 (July 1976), pp. 493–508. DOI: 10.2190/uurl-2ryu-wryd-ey3k.

[Keo+01] Keogh, E. et al. "An online algorithm for segmenting time series". In: *Proceedings 2001 IEEE International Conference on Data Mining*. IEEE Comput. Soc, 2001. DOI: 10.1109/icdm.2001.989531.

[KSL13] Khan, A., Siddiqi, M., and Lee, S.-W. "Exploratory Data Analysis of Acceleration Signals to Select Light-Weight and Accurate Features for Real-Time Activity Recognition on Smartphones". In: *Sensors* 13.10 (Sept. 2013), pp. 13099–13122. DOI: 10.3390/s131013099.

[KT17] Khan, S. S. and Taati, B. "Detecting unseen falls from wearable devices using channel-wise ensemble of autoencoders". In: *Expert Systems with Applications* 87 (Nov. 2017), pp. 280–290. DOI: 10.1016/j.eswa.2017.06.011.

[KK18] Kim, W. and Kim, M. "On-Line Detection and Segmentation of Sports Motions Using a Wearable Sensor". In: *Sensors* 18.3 (Mar. 2018), p. 913. DOI: 10.3390/s18030913.

[KB15] Kingma, D. and Ba, J. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. San Diego, CA, USA, 2015.

[KC14] Krishnan, N. C. and Cook, D. J. "Activity recognition on streaming sensor data". In: *Pervasive and Mobile Computing* 10 (Feb. 2014), pp. 138–154. DOI: 10.1016/j.pmcj.2012.07.003.

[KP08] Krishnan, N. C. and Panchanathan, S. "Analysis of low resolution accelerometer data for continuous human activity recognition". In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Mar. 2008. DOI: 10.1109/icassp.2008.4518365.

[KWM11] Kwapisz, J. R., Weiss, G. M., and Moore, S. A. "Activity recognition using cell phone accelerometers". In: *ACM SIGKDD Explorations Newsletter* 12.2 (Mar. 2011), p. 74. DOI: 10.1145/1964897.1964918.

[LL13] Lara, O. D. and Labrador, M. A. "A Survey on Human Activity Recognition using Wearable Sensors". In: *IEEE Communications Surveys & Tutorials* 15.3 (2013), pp. 1192–1209. DOI: 10.1109/surv.2012.110112.00192.

[LB69] Lawton, M. P. and Brody, E. M. "Assessment of Older People: Self-Maintaining and Instrumental Activities of Daily Living". In: *The Gerontologist* 9.3 Part 1 (Sept. 1969), pp. 179–186. DOI: 10.1093/geront/9.3_part_1.179.

[LBH15] LeCun, Y., Bengio, Y., and Hinton, G. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: 10.1038/nature14539.

[Leo78] Leontev, A. N. *Activity, Consciousness and Personality*. Prentice Hall, 1978. ISBN: 9780130035332.

[Les+05] Lester, J. et al. "A Hybrid Discriminative/Generative Approach for Modeling Human Activities". In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. Edinburgh, Scotland: Morgan Kaufmann Publishers Inc., 2005, pp. 766–772.

[Li+19] Li, L. et al. "A maximal correlation embedding method for multilabel human context recognition". In: *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*. 2019, pp. 305–306.

[LLL11] Li, X., Lee, T. S., and Liu, Y. "Hybrid generative-discriminative classification using posterior divergence". In: *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2011. DOI: 10.1109/cvpr.2011.5995584.

[Li+14] Li, Y. et al. "Unsupervised Feature Learning for Human Activity Recognition Using Smartphone Sensors". In: *Mining Intelligence and Knowledge Exploration*. Springer International Publishing, 2014, pp. 99–107. DOI: 10.1007/978-3-319-13817-6_11.

[LFK05] Liao, L., Fox, D., and Kautz, H. A. "Location-Based Activity Recognition using Relational Markov Networks." In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. 2005, pp. 773–778. ISBN: 0938075934.

[Liu+16] Liu, C. et al. "Lasagna: towards deep hierarchical understanding and searching over mobile sensing data". In: *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, Oct. 2016. DOI: 10.1145/2973750.2973752.

[LT14] Liu, H. and Taniguchi, T. "Feature Extraction and Pattern Recognition for Human Motion by a Deep Sparse Autoencoder". In: *2014 IEEE International Conference on Computer and Information Technology*. IEEE, Sept. 2014. DOI: 10.1109/cit.2014.144.

[Llo+15] Lloret, J. et al. "A smart communication architecture for ambient assisted living". In: *IEEE Communications Magazine* 53.1 (Jan. 2015), pp. 26–33. DOI: 10.1109/mcom.2015.7010512.

[Log+21] Logacjov, A. et al. "HARTH: a human activity recognition dataset for machine learning". In: *Sensors* 21.23 (2021), p. 7853.

[Lu+10] Lu, H. et al. "The Jigsaw continuous sensing engine for mobile phone applications". In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems - SenSys '10*. ACM Press, 2010. DOI: 10.1145/1869983.1869992.

[Luk+07] Lukowicz, P. et al. "WearIT@work: Toward Real-World Industrial Wearable Computing". In: *IEEE Pervasive Computing* 6.4 (Oct. 2007), pp. 8–13. DOI: 10.1109/mprv.2007.89.

[Man+19] Mansoor, H. et al. "Delfi: Mislabelled human context detection using multi-feature similarity linking". In: *2019 IEEE Visualization in Data Science (VDS)*. IEEE. 2019, pp. 11–19.

[May+16] Mayr, H. C. et al. "HCM-L: Domain-Specific Modeling for Active and Assisted Living". In: *Domain-Specific Conceptual Modeling*. Springer International Publishing, 2016, pp. 527–552. DOI: 10.1007/978-3-319-39417-6_24.

[MMN17] Micucci, D., Mobilio, M., and Napoletano, P. "UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data from Smartphones". In: *Applied Sciences* 7.10 (Oct. 2017), p. 1101. DOI: 10.3390/app7101101.

[Mit07] Mitchell, H. B. "Sensor Value Normalization". In: *Multi-Sensor Data Fusion*. Springer, Berlin, Heidelberg, 2007, pp. 97–112. DOI: 10.1007/978-3-540-71559-7_7.

[Moh+22] Mohamed, A. et al. "HAR-GCNN: Deep Graph CNNs for Human Activity Recognition From Highly Unlabeled Mobile Sensor Data". In: *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE. 2022, pp. 335–340.

[Mol+22] Mollyn, V. et al. "SAMoSA: Sensing Activities with Motion and Subsampled Audio". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6.3 (2022), pp. 1–19.

[MR16] Morales, F. J. O. and Roggen, D. "Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations". In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers - ISWC '16*. ACM Press, 2016. DOI: 10.1145/2971763.2971764.

[Mor+15] Moran, K. et al. "Detection of Running Asymmetry Using a Wearable Sensor System". In: *Procedia Engineering* 112 (2015), pp. 180–183. DOI: 10.1016/j.proeng.2015.07.196.

[MM13] Muaaz, M. and Mayrhofer, R. "An Analysis of Different Approaches to Gait Recognition Using Cell Phone Based Accelerometers". In: *Proceedings of International Conference on Advances in Mobile Computing & Multimedia - MoMM '13*. ACM Press, 2013. DOI: 10.1145/2536853.2536895.

[NSS05] Neumann, J., Schnörr, C., and Steidl, G. "Combined SVM-Based Feature Selection and Classification". In: *Machine Learning* 61.1-3 (July 2005), pp. 129–150. DOI: 10.1007/s10994-005-1505-9.

[NJ02] Ng, A. Y. and Jordan, M. I. "On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes". In: *Advances in Neural Information Processing Systems 14*. Ed. by T. G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, 2002, pp. 841–848.

[NPG18] Nguyen, K. T., Portet, F., and Garbay, C. "Dealing with imbalanced data sets for human activity recognition using mobile phone sensors". In: *3rd International Workshop on Smart Sensing Systems*. 2018.

[NHC15] Ni, Q., Hernando, A. G., and Cruz, I. de la. "The Elderly's Independent Living in Smart Homes: A Characterization of Activities and Sensing Infrastructure Survey to Facilitate Services Development". In: *Sensors* 15.5 (May 2015), pp. 11312–11362. DOI: 10.3390/s150511312.

[Nwe+18] Nweke, H. F. et al. "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges". In: *Expert Systems with Applications* 105 (Sept. 2018), pp. 233–261. DOI: 10.1016/j.eswa.2018.03.056.

[OCW14] Okeyo, G., Chen, L., and Wang, H. "Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes". In: *Future Generation Computer Systems* 39 (Oct. 2014), pp. 29–43. DOI: 10.1016/j.future.2014.02.014.

[OF96] Olshausen, B. A. and Field, D. J. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583 (June 1996), pp. 607–609. DOI: 10.1038/381607a0.

[OR16] Ordóñez, F. and Roggen, D. "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition". In: *Sensors* 16.1 (Jan. 2016), p. 115. DOI: 10.3390/s16010115.

[Pas+19] Paszke, A. et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[Pat+05] Patterson, D. J. et al. "Fine-Grained Activity Recognition by Aggregating Abstract Object Usage". In: *Ninth IEEE International Symposium on Wearable Computers (ISWC'05)*. IEEE, 2005. DOI: 10.1109/iswc.2005.22.

[Ped+11] Pedregosa, F. et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[PLD05] Peng, H., Long, F., and Ding, C. "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8 (Aug. 2005), pp. 1226–1238. DOI: 10.1109/tpami.2005.159.

[Pir+22] Pires, I. M. et al. "Daily motionless activities: A dataset with accelerometer, magnetometer, gyroscope, environment, and GPS data". In: *Scientific Data* 9.1 (2022), p. 105.

[PR12] Pirsiavash, H. and Ramanan, D. "Detecting activities of daily living in first-person camera views". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 2847–2854. DOI: 10.1109/CVPR.2012.6248010.

[PHO11] Plötz, T., Hammerla, N. Y., and Olivier, P. "Feature Learning for Activity Recognition in Ubiquitous Computing". In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*. Barcelona, Catalonia, Spain: AAAI Press, 2011, pp. 1729–1734. ISBN: 9781577355144.

[PW17] Press, O. and Wolf, L. "Using the Output Embedding to Improve Language Models". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 157–163.

[PY19] Putra, D. N. S. and Yulita, I. N. "Multilayer Perceptron for Activity Recognition Using a Batteryless Wearable Sensor". In: *IOP Conference Series: Earth and Environmental Science* 248 (Apr. 2019), p. 012039. DOI: 10.1088/1755-1315/248/1/012039.

[Que+15] Quesada, F. J. et al. "Generation of a Partitioned Dataset with Single, Interleave and Multioccupancy Daily Living Activities". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2015, pp. 60–71. DOI: 10.1007/978-3-319-26401-1_6.

[Rab89] Rabiner, L. R. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286. DOI: 10.1109/5.18626.

[Rad+16] Radu, V. et al. "Towards multimodal deep learning for activity recognition on mobile devices". In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. ACM, Sept. 2016. DOI: 10.1145/2968219.2971461.

[RMM16] Ranasinghe, S., Machot, F. A., and Mayr, H. C. "A review on applications of activity recognition systems with regard to performance and evaluation". In: *International Journal of Distributed Sensor Networks* 12.8 (Aug. 2016), p. 155014771666552. DOI: 10.1177/1550147716665520.

[Rav+16] Ravi, D. et al. "Deep learning for human activity recognition: A resource efficient implementation on low-power devices". In: *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE, June 2016. DOI: 10.1109/bsn.2016.7516235.

[Raz+21] Raza, A. et al. *Lightweight Transformer in Federated Setting for Human Activity Recognition*. 2021. arXiv: 2110.00244 [cs.CV].

[Rei12] Reiss, A. *PAMAP2 Physical Activity Monitoring Data Set (UCI Machine Learning Repository)*. 2012.

[RS12] Reiss, A. and Stricker, D. "Introducing a New Benchmarked Dataset for Activity Monitoring". In: *2012 16th International Symposium on Wearable Computers*. IEEE, June 2012. DOI: 10.1109/iswc.2012.13.

[RO+15] Reyes-Ortiz, J.-L. et al. *Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set (UCI Machine Learning Repository)*. 2015.

[RO+16] Reyes-Ortiz, J.-L. et al. "Transition-Aware Human Activity Recognition Using Smartphones". In: *Neurocomputing* 171 (Jan. 2016), pp. 754–767. DOI: 10.1016/j.neucom.2015.07.085.

[RPH10] Roggen, D., Plotnik, M., and Hausdorff, J. *Daphnet Freezing of Gait Data Set (UCI Machine Learning Repository)*. 2010.

[Rog+10] Roggen, D. et al. "Collecting complex activity datasets in highly rich networked sensor environments". In: *2010 Seventh International Conference on Networked Sensing Systems (INSS)*. IEEE, June 2010. DOI: 10.1109/inss.2010.5573462.

[SOL18] Saeed, A., Ozcelebi, T., and Lukkien, J. "Synthesizing and reconstructing missing sensory modalities in behavioral context recognition". In: *Sensors* 18.9 (2018), p. 2967.

[Sae+19] Saeed, A. et al. "End-to-end multi-modal behavioral context recognition in a real-life setting". In: *2019 22th International Conference on Information Fusion (FUSION)*. IEEE. 2019, pp. 1–8.

[SS18] Sánchez, V. G. and Skeie, N.-O. "Decision Trees for Human Activity Recognition Modelling in Smart House Environments". In: *SNE Simulation Notes Europe* 28.4 (Dec. 2018), pp. 177–184. DOI: 10.11128/sne.28.tn.10447.

[Sco92] Scott, D. W. *Multivariate Density Estimation*. John Wiley & Sons, Aug. 1992. DOI: 10.1002/9780470316849.

[SK21] Shavit, Y. and Klein, I. "Boosting Inertial-Based Human Activity Recognition With Transformers". In: *IEEE Access* 9 (2021), pp. 53540–53547. DOI: 10.1109/ACCESS.2021.3070646.

[She+22] Shen, Q. et al. "Federated Multi-Task Attention for Cross-Individual Human Activity Recognition". In: *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI'22)*. 2022, pp. 3423–3429.

[She+23a] Shen, Q. et al. "Federated Meta-Learning with Attention for Diversity-Aware Human Activity Recognition". In: *Sensors* 23.3 (2023), p. 1083.

[She+23b] Shen, Q. et al. "To Transfer or Not to Transfer and Why? Meta-Transfer Learning for Explainable and Controllable Cross-Individual Activity Recognition". In: *Electronics* 12.10 (2023), p. 2275.

[Sho+15] Shoaib, M. et al. "Towards detection of bad habits by fusing smartphone and smartwatch sensors". In: *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, Mar. 2015. DOI: 10.1109/percomw.2015.7134104.

[Sho+16] Shoaib, M. et al. "Complex Human Activity Recognition Using Smartphone and Wrist-Worn Motion Sensors". In: *Sensors* 16.4 (Mar. 2016), p. 426. DOI: 10.3390/s16040426.

[SR12] Siirtola, P. and Röning, J. "User-Independent Human Activity Recognition Using a Mobile Phone: Offline Recognition vs. Real-Time on Device Recognition". In: *Advances in Intelligent and Soft Computing*. Springer Berlin Heidelberg, 2012, pp. 617–627. DOI: 10.1007/978-3-642-28765-7_75.

[SN21] Sikder, N. and Nahid, A.-A. "KU-HAR: An open dataset for heterogeneous human activity recognition". In: *Pattern Recognition Letters* 146 (2021), pp. 46–54. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2021.02.024.

[Smo86] Smolensky, P. "Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. Ed. by D. E. Rumelhart and J. L. McClelland. MIT Press, 1986, pp. 194–281.

[Sou+23] Souza, P. de et al. "A Study on the Influence of Sensors in Frequency and Time Domains on Context Recognition". In: *Sensors* 23.12 (2023), p. 5756.

[Sti+11] Stikic, M. et al. "Weakly Supervised Recognition of Daily Life Activities with Wearable Sensors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.12 (Dec. 2011), pp. 2521–2537. DOI: 10.1109/tpami.2011.36.

[Sti+08] Stikic, M. et al. "ADL recognition based on the combination of RFID and accelerometer sensing". In: *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*. IEEE, Jan. 2008. DOI: 10.1109/pcthealth.2008.4571084.

[Sti+15a] Stisen, A. et al. *Heterogeneity Activity Recognition Data Set (UCI Machine Learning Repository)*. 2015.

[Sti+15b] Stisen, A. et al. "Smart Devices are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition". In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems - SenSys '15*. ACM Press, 2015. DOI: 10.1145/2809695.2809718.

[Sto+23] Stojchevska, M. et al. "From Lab to Real World: Assessing the Effectiveness of Human Activity Recognition and Optimization through Personalization". In: *Sensors* 23.10 (2023), p. 4606.

[Sun+19] Sun, B. et al. "Attention-based LSTM Network for Wearable Human Activity Recognition". In: *2019 Chinese Control Conference (CCC)*. IEEE, July 2019. DOI: 10.23919/chicc.2019.8865360.

[Tam+17] Tamamori, A. et al. "An investigation of recurrent neural network for daily activity recognition using multi-modal signals". In: *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, Dec. 2017. DOI: 10.1109/apsipa.2017.8282239.

[TB21] Tarafdar, P. and Bose, I. "Recognition of human activities for wellness management using a smartphone and a smartwatch: a boosting approach". In: *Decision Support Systems* 140 (2021), p. 113426.

[Tee+22] Tee, W. Z. et al. "Human Activity Recognition models using Limited Consumer Device Sensors and Machine Learning". In: *2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML)*. IEEE. 2022, pp. 456–461.

[UB16] Uslu, G. and Baydere, S. "On the activity detection with incomplete acceleration data using iterative KNN classifier". In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Oct. 2016. DOI: 10.1109/smc.2016.7844779.

[VEL17] Vaizman, Y., Ellis, K., and Lanckriet, G. "Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches". In: *IEEE Pervasive Computing* 16.4 (Oct. 2017), pp. 62–74. DOI: 10.1109/mprv.2017.3971131.

[VWL18] Vaizman, Y., Weibel, N., and Lanckriet, G. "Context Recognition In-the-Wild: Unified Model for Multi-Modal Sensors and Multi-Label Classification". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1.4 (Jan. 2018), pp. 1–22. DOI: 10.1145/3161192.

[Vai+18] Vaizman, Y. et al. "ExtraSensory App: Data Collection In-the-Wild with Rich User Interface to Self-Report Behavior". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, 2018. DOI: 10.1145/3173574.3174128.

[Var+18] Varamin, A. A. et al. *Deep Auto-Set: A Deep Auto-Encoder-Set Network for Activity Recognition Using Wearables*. 2018. arXiv: 1811.08127v1 [cs.LG].

[Vas+17] Vaswani, A. et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. 2017, pp. 5998–6008.

[Vel18] Veličković, P. *Collection of PGF/TikZ figures*. 2018.

[Vel+96] Veltink, P. H. et al. "Detection of static and dynamic activities using uniaxial accelerometers". In: *IEEE Transactions on Rehabilitation Engineering* 4.4 (1996), pp. 375–385. DOI: 10.1109/86.547939.

[Vep+15] Vepakomma, P. et al. "A-Wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities". In: *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE, June 2015. DOI: 10.1109/bsn.2015.7299406.

[WDT16] Walse, K. H., Dharaskar, R. V., and Thakare, V. M. "PCA Based Optimal ANN Classifiers for Human Activity Recognition Using Mobile Sensors Data". In: *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 1*. Springer International Publishing, 2016, pp. 429–436. DOI: 10.1007/978-3-319-30933-0_43.

[Wan+19] Wang, J. et al. "Deep learning for sensor-based activity recognition: A survey". In: *Pattern Recognition Letters* 119 (Mar. 2019), pp. 3–11. DOI: 10.1016/j.patrec.2018.02.010.

[WL19] Wang, L. and Liu, R. "Human Activity Recognition Based on Wearable Sensor Using Hierarchical Deep LSTM Networks". In: *Circuits, Systems, and Signal Processing* 39.2 (Apr. 2019), pp. 837–856. DOI: 10.1007/s00034-019-01116-y.

[Wan+11] Wang, W. zhong et al. "Analysis of filtering methods for 3D acceleration signals in body sensor network". In: *International Symposium on Bioelectronics and Bioinformations 2011*. IEEE, Nov. 2011. DOI: 10.1109/isbb.2011.6107697.

[War+06] Ward, J. et al. "Activity Recognition of Assembly Tasks Using Body-Worn Microphones and Accelerometers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10 (Oct. 2006), pp. 1553–1567. DOI: 10.1109/tpami.2006.197.

[WGY18] Weiss, G., Goldberg, Y., and Yahav, E. *On the Practical Computational Power of Finite Precision RNNs for Language Recognition*. 2018. arXiv: 1805.04908v1 [cs.LG].

[Wei19] Weiss, G. *WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set (UCI Machine Learning Repository)*. 2019.

[WYH19] Weiss, G. M., Yoneda, K., and Hayajneh, T. "Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living". In: *IEEE Access* 7 (2019), pp. 133190–133202. DOI: 10.1109/access.2019.2940729.

[WX08] Wu, G. and Xue, S. "Portable Preimpact Fall Detector With Inertial Sensors". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 16.2 (Apr. 2008), pp. 178–183. DOI: 10.1109/tnsre.2007.916282.

[Yan+12] Yan, Z. et al. "Energy-Efficient Continuous Activity Recognition on Mobile Phones: An Activity-Adaptive Approach". In: *2012 16th International Symposium on Wearable Computers*. IEEE, June 2012. DOI: 10.1109/iswc.2012.23.

[Yan+15] Yang, J. B. et al. "Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition". In: *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. Buenos Aires, Argentina: AAAI Press, 2015, 3995–4001. ISBN: 9781577357384.

[Yao+16] Yao, S. et al. *DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing*. 2016. arXiv: 1611.01942v2 [cs.LG].

[Yfa+22] Yfantidou, S. et al. "LifeSnaps, a 4-month multi-modal dataset capturing unobtrusive snapshots of our lives in the wild". In: *Scientific Data* 9.1 (2022), p. 663.

[YTO13] Yoshizawa, M., Takasaki, W., and Ohmura, R. "Parameter exploration for response time reduction in accelerometer-based activity recognition". In: *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication - UbiComp '13*. ACM Press, 2013. DOI: 10.1145/2494091.2495986.

[YQY18] Yu, S., Qin, L., and Yin, Q. "A C-LSTM Neural Network for Human Activity Recognition Using Wearables". In: *2018 International Symposium in Sensing and Instrumentation in IoT Era (ISSI)*. IEEE, Sept. 2018. DOI: 10.1109/issi.2018.8538129.

[Yua+23] Yuan, H. et al. *Self-supervised Learning for Human Activity Recognition Using 700,000 Person-days of Wearable Data*. 2023. arXiv: 2206.02909 [eess.SP].

[Zap+07] Zappi, P. et al. "Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness". In: *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. IEEE, 2007. DOI: 10.1109/issnip.2007.4496857.

[Zen+14] Zeng, M. et al. "Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors". In: *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services*. ICST, 2014. DOI: 10.4108/icst.mobicase.2014.257786.

[ZWL15] Zhang, L., Wu, X., and Luo, D. "Real-Time Activity Recognition on Smartphones Using Deep Neural Networks". In: *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*. IEEE, Aug. 2015. DOI: 10.1109/uic-atc-scalcom-cbdcom-iop.2015.224.

[ZS12] Zhang, M. and Sawchuk, A. A. "USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*. ACM Press, 2012. DOI: 10.1145/2370216.2370438.

[ZS13] Zhang, M. and Sawchuk, A. A. "Human Daily Activity Recognition With Sparse Representation Using Wearable Sensors". In: *IEEE Journal of Biomedical and Health Informatics* 17.3 (May 2013), pp. 553–560. DOI: 10.1109/jbhi.2013.2253613.

[Zha+18] Zhao, Y. et al. "Deep Residual Bidir-LSTM for Human Activity Recognition Using Wearable Sensors". In: *Mathematical Problems in Engineering* 2018 (Dec. 2018), pp. 1–13. DOI: 10.1155/2018/7316954.

[ZHY09] Zheng, V. W., Hu, D. H., and Yang, Q. "Cross-domain activity recognition". In: *Proceedings of the 11th international conference on Ubiquitous computing*. ACM, Sept. 2009. DOI: 10.1145/1620545.1620554.

[ZGW15] Zhou, X., Guo, J., and Wang, S. "Motion Recognition by Using a Stacked Autoencoder-Based Deep Learning Algorithm with Smart Phones". In: *Wireless Algorithms, Systems, and Applications*. Springer International Publishing, 2015, pp. 778–787. DOI: 10.1007/978-3-319-21837-3_76.

[Zla+22] Zlatintsi, A. et al. "E-Prevention: Advanced Support System for Monitoring and Relapse Prevention in Patients with Psychotic Disorders Analyzing Long-Term Multimodal Data from Wearables and Video Captures". In: *Sensors* 22.19 (2022). ISSN: 1424-8220. DOI: 10.3390/s22197544.