NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF SIGNALS, CONTROL AND ROBOTICS
INTELLIGENT ROBOTICS AND AUTOMATION LABORATORY

Surgical Gesture Recognition in Robot-Assisted Surgery using Machine Learning Methods on Kinematic Data

DIPLOMA THESIS

of

ALEXANDROS DIMITRIADIS

Supervisor: Constantinos Tzafestas
Associate Professor

Athens, November 2024

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Intelligent Robotics & Automation Laboratory

Αναγνώριση χειρουργικών κινήσεων στη Ρομποτική Χειρουργική χρησιμοποιώντας μεθόδους Μηχανικής Μάθησης σε Κινηματικά Δεδομένα

DIPLOMA THESIS

of

ALEXANDROS DIMITRIADIS

Supervisor: Constantinos Tzafestas
Associate Professor

Approved by the examination committee on 06/11/2024.

.....	.....	.....
Constantinos Tzafestas, Associate Professor NTUA	Athanasios Rontogiannis Associate Professor NTUA	Haris Psillakis Lecturer NTUA

Athens, November 2024

.....
Alexandros Dimitriadis
Graduate of Electrical and Computer Engineering NTUA

Περίληψη

Στόχος της παρούσας διπλωματικής εργασίας ήταν η εκπαίδευση ενός μοντέλου μηχανικής μάθησης για την αναγνώριση χειρονομιών κατά τη διάρκεια ρομποτικών χειρουργικών επεμβάσεων σε πραγματικό χρόνο. Η αξιολόγηση έγινε χρησιμοποιώντας το dataset JIGSAWS, το οποίο περιλαμβάνει δεδομένα από τη χρήση του χειρουργικού ρομποτ Da Vinci. Συγκεκριμένα αξιοποιήσαμε τα δεδομένα από τις δοκιμές συρραφής. Στόχος ήταν η επίτευξη υψηλής απόδοσης ως προς την επιτυχή αναγνώριση χειρονομιών, συγκρίσιμη με τις βέλτιστες της διεθνούς βιβλιογραφίας, υπό τις ακόλουθες συνθήκες υπό τις ακόλουθες συνθήκες: α) το μοντέλο να μπορεί να λειτουργεί σε πραγματικό χρόνο μέσω ενός συρόμενου παραθύρου με μέγιστη καθυστέρηση 1 δευτερολέπτου, και β) η εκπαίδευση των μοντέλων να βασιστεί μόνο σε κινηματικά δεδομένα, χωρίς δηλαδή τη χρήση οπτικών (ενδοσκοπικών) δεδομένων. Το βασικό νευρωνικό δίκτυο που χρησιμοποιήθηκε ήταν το LSTM. Επιχειρήσαμε να βελτιώσουμε την απόδοση του μοντέλου με διάφορες τεχνικές όπως βελτιστοποίηση υπερπαραμέτρων, πρόωρη διακοπή, εισαγωγή drop out, κανονικοποίηση L2, και stratification.

Σε κάθε βήμα, έγινε προσπάθεια οπτικοποίησης των αποτελεσμάτων με διάφορα γραφήματα οπως confusion matrix, transition matrix και ακρίβεια ανά κλάση για να αξιολογήσουμε τις περιοχές που χρειάζονται βελτίωση. Πειραματιστήκαμε με τη χρήση διαφορετικών υποσυνόλων των διαθέσιμων χαρακτηριστικών (features), και παράλληλα έγινε μία προσπάθεια feature engineering, συνδυάζοντας τις γωνίες των δύο gripper σε μία νέα μεταβλητή τεσσάρων πιθανών καταστάσεων που ονομάσαμε Joint Gripper State.

Για την περαιτέρω βελτίωση της απόδοσης του συστήματος αναγνώρισης χειρονομιών σε πραγματικό χρόνο, προτείνονται και αξιολογούνται στην παρούσα διπλωματική εργασία δύο υβριδικές προσεγγίσεις της βασικής αρχιτεκτονικής ενός μοντέλου LSTM. Στην πρώτη προσέγγιση, εισάγεται ένα επιπλέον επίπεδο Attention, το οποίο επιλέχθηκε μετά από συγκριτική αξιολόγηση διαφόρων διατάξεων. Στη δεύτερη προσέγγιση, έγινε προσπάθεια εκμετάλλευσης της αραιούς κατανομής του transition matrix χρησιμοποιώντας ένα CRF το οποίο λαμβάνει ως είσοδο τις προβλέψεις του LSTM σε συνδυασμό με μέρος των κινηματικών δεδομένων. Βέλτιστη απόδοση με ακρίβεια 81.56% επετεύχθη τελικώς χρησιμοποιώντας ένα υβριδικό μοντέλο LSTM-Self Attention, βελτιώνοντας αντίστοιχες επιδόσεις που αναφέρονται στη διεθνή βιβλιογραφία, δεδομένων των δύο αυστηρών περιορισμών που ετέθησαν.

Λέξεις Κλειδιά

Αναγνώριση Ρομποτικών Χειρουργικών Κινήσεων, Ρομποτική Χειρουργική, LSTM, Κινηματικά Δεδομένα, Μηχανική Μάθηση, Attention Mechanism, CRF

Abstract

This diploma thesis focuses on training a machine learning model to recognize gestures during robot-assisted surgical procedures in real-time, using exclusively kinematic data from the patient-side manipulators. The JIGSAWS dataset, specifically the suturing tasks, serves as the evaluation benchmark. Our goal was to achieve state-of-the-art performance, ensuring the model operates in real-time with a maximum delay of 1 second and is trained solely on kinematic data.

We experimented with various neural network architectures, using an LSTM architecture as foundation, in order to effectively capture temporal dependencies within the data sequences. Visualization tools like graphs, confusion matrices, and transition matrices were employed to identify areas for improvement.

Challenges arising from imbalanced data led to difficulties in recognizing underrepresented classes. We expanded the feature set, creating a new feature based on gripper angles. To further enhance performance, we implemented two hybrid approaches: one integrating an attention layer and another combining an LSTM with a Conditional Random Field (CRF) to leverage the sparse transition matrix. Our efforts culminated in a hybrid LSTM - Self Attention model, achieving an accuracy of 81.56%, demonstrating improvements and meeting the constraints set for real-time operation and exclusive use of kinematic data.

Keywords

Surgical Gesture Recognition, Robotic Surgery, JIGSAWS, LSTM, Kinematic Data, Real-time, Machine Learning, Hybrid Model, Attention Mechanism, Self Attention, CRF

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή, κ. Κωνσταντίνο Τζαφέστα για την καθοδήγηση και για την εμπιστοσύνη που μου έδειξε καθ' όλη τη διάρκεια της συνεργασίας μας. Επίσης, τον Δρ. Γεώργιο Μούστρη για τις συμβουλές και την πολύτιμη βοήθειά του σε όλες τις δυσκολίες.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στους γονείς μου, Αθανάσιο και Χρυσάνθη, καθώς και στη σύζυγό μου Μαριάννα για την κατανόηση και την υποστήριξη σε κάθε βήμα.

Δημητριάδης Αλέξανδρος

Νοέμβριος 2024

List of Figures

List of Tables

Εκτεταμένη Ελληνική Περίληψη

Εισαγωγή

Τις τελευταίες δύο δεκαετίες, έχουμε δει σημαντική πρόοδο στην ανάπτυξη τεχνικών Τεχνητής Νοημοσύνης και Μηχανικής Μάθησης με εντυπωσιακά αποτελέσματα και εφαρμογές σε διάφορους τομείς. Η ταχεία εξέλιξη της επιστήμης δεδομένων και της βαθιάς μάθησης δεν έχει περάσει απαρατήρητη από τον κλάδο της ιατρικής, όπου υπάρχει ολοένα και μεγαλύτερη επιθυμία για ενσωμάτωσης της τεχνολογίας με στόχο την ενίσχυση της φροντίδας των ασθενών, τη βελτιστοποίηση των ροών εργασίας και την αύξηση της συνολικής αποτελεσματικότητας. Μία από τις πιθανές εφαρμογές είναι η ρομποτική χειρουργική. Η χρήση χειρουργικών ρομπότ παρέχει πολύτιμα δεδομένα που μπορούν να αξιοποιηθούν από μοντέλα βαθιάς μάθησης και να συμβάλλουν στην κατανόησή των χειρουργικών διεργασιών, την αξιολόγηση της εκτέλεσης αυτών, την παροχή βοήθειας κατά τη διάρκεια της διαδικασίας και γενικά αποτελεί ένα σημαντικό βήμα προς τον αυτοματισμό της χειρουργικής. Ιδιάιτερα κρίσιμο βήμα προς αυτή την κατεύθυνση, αποτελεί ο διαχωρισμός των χειρουργικών εργασιών σε ατομικές χειρονομίες και η ικανότητα αναγνώρισής τους. Αν και η κλινική υιοθέτηση της αυτόνομης ρομποτικής χειρουργικής φαίνεται ακόμα μακρινή, υπάρχει πλούσια βιβλιογραφία σχετικά με την αναγνώριση χειρονομιών, με σημαντικά ευρήματα. Αυτή η διπλωματική εργασία, παράλληλα με την βελτίωση της απόδοσης των μοντέλων που χρησιμοποιούν κινηματικά δεδομένα, στοχεύει επίσης να διερευνήσει βαθύτερα τις εγγενείς προκλήσεις και τους περιορισμούς των τρεχόντων μοντέλων, διεξάγοντας μια ανάλυση των υποκείμενων αδυναμιών κάθε προσέγγισης.

Σχετική βιβλιογραφία

Η βιβλιογραφία περιέχει πολυάριθμες εργασίες που παρουσιάζουν μια σειρά προσεγγίσεων για την ταξινόμηση χειρονομιών. Παρατηρούμε ότι παλαιότερες εργασίες επικεντρώθηκαν σε κινηματικά δεδομένα και προσπάθησαν να αντιμετωπίσουν το πρόβλημα χρησιμοποιώντας πιθανολογικά γραφικά μοντέλα, όπως Hidden Markov Models (HMM) ([21], [29] και [26]) ή Conditional Random Fields (CRFs) ([27], [13]). Μία επισκόπηση αυτών των προσεγγίσεων αποκαλύπτει έναν βασικό περιορισμό: περιορίζονται στη μοντελοποίηση μόνο μεταβάσεων καρέ προς καρέ ή τμήμα προς τμήμα. Αντίθετα, πρόσφατες προσπάθειες έχουν επιδιώξει να ξεπεράσουν αυτόν τον περιορισμό χρησιμοποιώντας τεχνικές βαθιάς μάθησης που έχουν την ικανότητα να καταγράφουν περίπλοκες μακροπρόθεσμες εξαρτήσεις σε ακολουθιακά δεδομένα. Τα Αναδρομικά Νευρωνικά Δίκτυα (RNN) είναι γνωστά για την ικανότητά τους να καταγράφουν μακροπρόθεσμες εξαρτήσεις, γεγονός που δικαιολογεί τη διάδοση της χρήσης τους σε διάφορες εφαρμογές. Στις περισσότερες περιπτώσεις, η προτιμώμενη παραλλαγή RNN είναι τα Δίκτυα Μακράς-Βραχυχρόνιας

Μνήμης (LSTM) [6], η οποία έχει αποδειχθεί ιδιαίτερα αποτελεσματική για την επεξεργασία ακολουθιακών κινηματικών δεδομένων. Επίσης, ορισμένες δημοσιεύσεις έχουν παρουσιάσει δίκτυα διπλής κατεύθυνσης (BiLSTM, BiGRU) [5], τα οποία έχουν αρκετά καλή απόδοση, αλλά με τον σημαντικό περιορισμό ότι δεν μπορούν να αξιοποιηθούν σε πραγματικό χρόνο, καθώς η αρχιτεκτονική τους, επιβάλει να έχουν πρόσβαση σε όλα τα δεδομένα της ακολουθίας, κάτι που δεν είναι εφικτό σε πραγματικό χρόνο. Τέλος, έχουν παρουσιαστεί ορισμένες αρχιτεκτονικές που επεξεργάζονται τόσο κινηματικά δεδομένα όσο και οπτικά. Τα δεύτερα τροφοδοτούνται συνήθως σε ένα Χρονικό Συνελικτικό Δίκτυο (TCN) [20]. Προσφάτως, έχουν χρησιμοποιηθεί μηχανισμοί προσοχής (Attention mechanisms) και η τεχνική της εξαγωγής χαρακτηριστικών που αντιπροσωπεύουν τις αλληλεπιδράσεις μεταξύ των χειρουργικών εργαλείων [31].

Συνεισφορά

Αυτή η διπλωματική εργασία παρουσιάζει μια προσπάθεια εφαρμογής αρχιτεκτονικών βαθιάς μάθησης για την αναγνώριση χειρονομιών χρησιμοποιώντας καταγραφές από την εκτέλεση χειρουργικών εργασιών συρραφής (suturing) του συνόλου δεδομένων JIGSAWS.

- Οι προσεγγίσεις μας υπακούουν σε δύο περιορισμούς:
    1. η εκπαίδευση των μοντέλων να βασίζεται μόνο σε κινηματικά δεδομένα, χωρίς δηλαδή τη χρήση οπτικών (ενδοσκοπικών) δεδομένων.
    2. το μοντέλο μπορεί να λειτουργεί σε πραγματικό χρόνο μέσω ενός συρόμενου παραθύρου, επομένως αποκλείουμε την εκμετάλευση μελλοντικών δεδομένων και επιτρέπουμε καθυστέρηση έως ένα δευτερόλεπτο.
- Η προσέγγισή μας περιελάμβανε την εκπαίδευση ενός δικτύου βασισμένου σε LSTM, το οποίο θα λειτουργήσει ως θεμέλιο.
- Προτείνονται δύο επεκτάσεις σε υβριδικά σχήματα που περιλαμβάνουν Conditional Random Fields (CRF) και μηχανισμούς προσοχής (attention mechanisms) αντίστοιχα.
- Πραγματοποιήθηκαν συγκρίσεις μεταξύ των διαφορετικών προτεινόμενων αρχιτεκτονικών και οπτικοποίση των αποτελεσμάτων.
- Εστίασαμε στις εγγενείς δυσκολίες του προβλήματος και την επίδραση των παραμέτρων του μοντέλου.
- Επετεύχθη Βελτίωση των κορυφαίων αποτελεσμάτων από τη διεθνή βιβλιογραφία τηρουμένων των περιορισμών που θέσαμε.

Θεωρητικό υπόβαθρο

Μηχανική Μάθηση

Η Μηχανική Μάθηση (Machine Learning) αποτελεί κλάδο της Τεχνητής Νοημοσύνης (AI) και περιγράφει ένα σύνολο αλγορίθμων που μαθαίνουν από δεδομένα αντί να ακολουθούν ρητές οδηγίες. Αυτοί οι αλγόριθμοι είναι ικανοί να αναλύουν στατιστικά τα δεδομένα για να εντοπίσουν μοτίβα και να κάνουν προβλέψεις ή να παίρνουν αποφάσεις. Τα μοντέλα μηχανικής μάθησης μαθαίνουν προσαρμόζοντας επαναληπτικά τις εσωτερικές τους παραμέτρους, για να ελαχιστοποιήσουν το σφάλμα. Ο στόχος είναι να χρησιμοποιηθεί η εμπειρία που αποκτήθηκε κατά την εκπαίδευση, σε νέα δεδομένα, τα οποία το μοντέλο δεν έχει επεξεργαστεί στο παρελθόν. Οι αλγόριθμοι μηχανικής μάθησης αντλούν έμπνευση από την ανθρώπινη γνωστική διαδικασία και για αυτό το λόγο, υπερέχουν σε εργασίες που απαιτούν αναγνώριση προτύπων και λήψη αποφάσεων, συχνά περιλαμβάνοντας άρρητες, διαισθητικές διεργασίες.

Τεχνητά Νευρωνικά Δίκτυα

Τυπικά, οι αρχιτεκτονικές βαθιάς μάθησης περιέχουν μια ιεραρχική δομή στρωμάτων που περιέχουν **νευρώνες**. Κάθε στρώμα παράγει μια αφηρημένη αναπαράσταση των δεδομένων εισόδου, περνώντας τα μέσα από μη γραμμικούς μετασχηματισμούς. Τα χαμηλότερα στρώματα συλλαμβάνουν απλούστερους συσχετισμούς, οι οποίοι στη συνέχεια χρησιμοποιούνται από τα ανώτερα στρώματα για να δημιουργήσουν σταδιακά μια πολύπλοκη αναπαράσταση. Συνήθως αποτελούνται από ένα **στρώμα εισόδου**, ένα **στρώμα εξόδου** και αρκετά **κρυφά στρώματα** ανάμεσά τους. Όταν κάθε νευρώνας ενός στρώματος είναι συνδεδεμένος με κάθε κόμβο του επόμενου στρώματος, αυτό ονομάζεται **πλήρως συνδεδεμένο** στρώμα και κατά συνέπεια ένα νευρωνικό δίκτυο του οποίου όλα τα στρώματα είναι πλήρως συνδεδεμένα, ονομάζεται αντίστοιχά πλήρως συνδεδεμένο νευρωνικό δίκτυο.

Τα δεδομένα εισόδου αποτελούνται από ένα σύνολο μετρήσιμων ιδιοτήτων που ονομάζονται χαρακτηριστικά, ενώ τα δεδομένα εξόδου παράγονται σε μορφή διανύσματος πιθανοτήτων. Για την εκπαίδευση των νευρωνικών δικτύων χρησιμοποιούμε **επισημειωμένα δεδομένα**. Κάθε στοιχείο εισόδου συνοδεύεται από μία ετικέτα η οποία περιλαμβάνει την επιθυμητή έξοδο. Η διαφορά των προβλέψεων από τις επιθυμητές εξόδους ορίζεται ως **κόστος** και υπολογίζεται από τη **συνάρτηση κόστους**. Τα αποτελέσματα της συνάρτησης κόστους για κάθε έξοδο του δικτύου αξιοποιούνται κατά τη διαδικασία του **backpropagation**, που περιλαμβάνει τη διάδοση του σφάλματος από την έξοδο προς την είσοδο του δικτύου. Κατά τη διάρκεια αυτής της διαδικασίας, οι παράγωγοι της συνάρτησης κόστους υπολογίζονται για κάθε βάρος και χρησιμοποιούνται για την ενημέρωση των βαρών με σκοπό τη μείωση του συνολικού σφάλματος.

Υπάρχουν δύο είδη τεχνητών νευρωνικών δικτύων, τα Νευρωνικά Δίκτυα Ευθείας

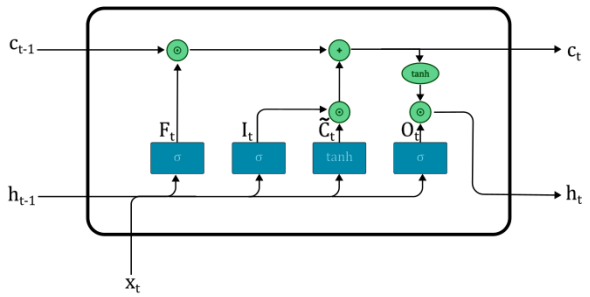

Figure 1: Αρχιτεκτονική ενός κυττάρου μνήμης, δικτύου LSTM

Τροφοδότησης (FNN) και τα Αναδρομικά Νευρωνικά Δίκτυα (RNN). Η διαφορά μεταξύ των δύο έγκειται στη ροή των πληροφοριών. Στα FNN κάθε νευρώνας μπορεί να λαμβάνει δεδομένα μόνο από τους νευρώνες των προηγούμενων στρωμάτων, ενώ τα RNN επιτρέπουν κυκλικές συνδέσεις που ονομάζονται **βρόχοι ανατροφοδότησης**.

Δίκτυα Μακράς Βραχυχρόνιας Μνήμης

Ο βρόχος ανατροφοδότησης που περιλαμβάνουν τα RNN, τους επιτρέπει να διατηρούν μια "μνήμη" προηγούμενων εισόδων, η οποία αντιπροσωπεύεται από την κρυφή κατάσταση, καθιστώντας τα ικανά να καταγράφουν χρονικές εξαρτήσεις εντός των δεδομένων. Ωστόσο, τα RNN πάσχουν από τα **προβλήματα εξαφάνισης και έκρηξης κλίσης**, τα οποία μπορούν να παρεμποδίσουν την ικανότητά τους να μαθαίνουν μακροπρόθεσμες εξαρτήσεις. Τα Δίκτυα Μακροπρόθεσμης Μνήμης (LSTM) σχεδιάστηκαν με στόχο την αντιμετώπιση των συγκεκριμένου προβλήματος. Τα LSTM χρησιμοποιούν μια περίπλοκη αρχιτεκτονική με τρεις πύλες: την **πύλη εισόδου**, την **πύλη λήθης** και την **πύλη εξόδου**. Αυτές οι πύλες ελέγχουν τη ροή των πληροφοριών και την κατάσταση μνήμης, επιτρέποντας στα LSTM να θυμούνται επιλεκτικά ή να ξεχνούν πληροφορίες με την πάροδο του χρόνου.

Τα κύτταρα μνήμης των LSTM δέχονται πληροφορίες από τρεις πηγές: την τρέχουσα είσοδο (x_t), η οποία αποτελείται από τα δεδομένα που εισάγονται στο κύτταρο κατά το τρέχον χρονικό βήμα, την προηγούμενη κρυφή κατάσταση (h_{t-1}) και την κατάσταση μνήμης του κυττάρου στο προηγούμενο χρονικό βήμα (C_{t-1}) , η οποία αποθηκεύει μακροπρόθεσμες πληροφορίες. Οι πληροφορίες αυτές περνούν από τρεις πύλες:

- Πύλη λήθης: Αποφασίζει ποιο μέρος της προηγούμενης κατάστασης μνήμης πρέπει να ξεχαστεί.

- Πύλη εισόδου: Αποφασίζει ποια νέα πληροφορία πρέπει να προστεθεί στην κατάσταση μνήμης.
- Πύλη εξόδου: Αποφασίζει ποιο μέρος της τρέχουσας κατάστασης μνήμης πρέπει να χρησιμοποιηθεί ως έξοδος.

Με αυτόν τον τρόπο, τα LSTM μπορούν να διατηρούν και να επεξεργάζονται πληροφορίες για μεγάλα χρονικά διαστήματα, καθιστώντας τα ιδιαίτερα αποτελεσματικά για την επεξεργασία ακολουθιακών δεδομένων.

Στατιστικά Μοντέλα

Ένα στατιστικό μοντέλο είναι ουσιαστικά ένα σύνολο κατανομών πιθανότητας στον χώρο δειγματοληψίας που περιγράφει την πιθανότητα διαφορετικών αποτελεσμάτων εντός ενός συνόλου δεδομένων. Μοντελοποιώντας τον τρόπο με τον οποίο τα δεδομένα κατανέμονται στο εκάστοτε πλαίσιο, μπορούμε να κάνουμε προβλέψεις για μελλοντικά δείγματα εντός του ίδιου πλαισίου.

Γραφικά Μοντέλα Πιθανότητας

Τα Γραφικά Μοντέλα Πιθανότητας (PGMs) χρησιμοποιούν δομές γραφημάτων για να απεικονίσουν πολύπλοκους συσχετισμούς μεταξύ μεταβλητών. Οι κόμβοι του γραφήματος αντιστοιχούν σε τυχαίες μεταβλητές και οι ακμές αντιπροσωπεύουν τις πιθανοτικές εξαρτήσεις μεταξύ των μεταβλητών που συνδέουν.

Markov Random Fields

Τα Markov Random Fields είναι μη κατευθυνόμενα γραφικά μοντέλα και οι κόμβοι τους αντιπροσωπεύουν τυχαίες μεταβλητές που ικανοποιούν την ιδιότητα Markov. Η ιδιότητα Markov δηλώνει ότι η επόμενη κατάσταση της στοχαστικής διαδικασίας εξαρτάται μόνο από την τρέχουσα κατάσταση και όχι από οποιαδήποτε προηγούμενη.

Conditional Random Fields (CRFs)

Τα Conditional Random Fields είναι ένα είδος μη κατευθυνόμενου γραφικού μοντέλου που βασίζεται στην αρχιτεκτονική των Markov Random Fields. Χρησιμοποιούνται συνήθως στη μηχανική μάθηση και ειδικά για εργασίες όπου οι γειτονικές καταστάσεις είναι κρίσιμες.

Τα CRF μοντελοποιούν την πιθανότητα μιας κλάσης σε μια συγκεκριμένη κορυφή του γραφήματος, ανάλογα με τις κλάσεις των γειτονικών κορυφών και την είσοδο X.

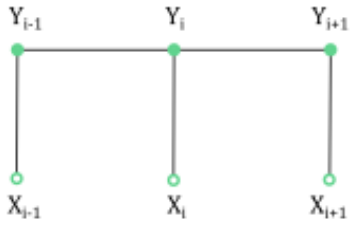

Figure 2: Δείγμα ενός γράφου CRF

Μηχανισμοί Προσοχής

Οι μηχανισμοί προσοχής είναι μια τεχνική μηχανικής μάθησης που δίνει στα μοντέλα τη δυνατότητα να εστιάζουν σε συγκεκριμένα μέρη των δεδομένων εισόδου, αυτά που θεωρούνται πιο επιδραστικά για την εκάστοτε εργασία, και να αγνοούν τα μέρη που θεωρούνται λιγότερο χρήσιμα. Αυτό συσχετίζεται με την ικανότητα του ανθρώπινου εγκεφάλου να φιλτράρει επιλεκτικά τον περιβαλλοντικό θόρυβο και τους περισπασμούς, ώστε να απομονώνει συγκεκριμένα ερεθίσματα. Ως εκ τούτου, γίνονται όλο και πιο δημοφιλείς για εργασίες όπως η επεξεργασία φυσικής γλώσσας, η αναγνώριση εικόνας και η μηχανική μετάφραση. Σε αντίθεση με τα RNNs, τα οποία χειρίζονται τα δεδομένα σειριακά, οι μηχανισμοί προσοχής επεξεργάζονται ολόκληρη την ακολουθία ταυτόχρονα, επιτρέποντας τον παράλληλους υπολογισμούς. Αυτή η προσέγγιση επιταχύνει σημαντικά τη διαδικασία εκπαίδευσης και αυξάνει την αποτελεσματικότητα.

Αυτο-Προσοχή (Self-Attention)

Η Αυτο-Προσοχή είναι μια θεμελιώδης παραλλαγή του μηχανισμού προσοχής που στοχεύει στην ανακάλυψη και μοντελοποίηση σχέσεων μεταξύ διαφορετικών τμημάτων μιας μεμονωμένης ακολουθίας εισόδου. Χρησιμοποιούν έναν μοναδικό τρόπο κωδικοποίησης πληροφοριών χρησιμοποιώντας τρεις διανύσματα:

- Ερώτημα (Query): Περιγράφει ποιο είδος πληροφορίας αναζητά αυτό το στοιχείο από άλλα στοιχεία της ακολουθίας.
- Κλειδί (Key): Αντιπροσωπεύει τη συνάφεια αυτού του στοιχείου με ένα δεδομένο ερώτημα.
- Τιμή (Value): Περιέχει τα πραγματικά δεδομένα που κατέχει το στοιχείο.

Το πρώτο βήμα για την εκπαίδευση ενός μοντέλου με αυτο-προσοχή είναι ο υπολογισμός αυτών των τριών διανυσμάτων για κάθε στοιχείο της ακολουθίας εισόδου. Στη συνέχεια, υπολογίζουμε επίσης μια βαθμολογία προσοχής για κάθε

ζεύγος στοιχείων εισόδου. Αυτή η τιμή αντιπροσωπεύει το βαθμό συσχέτισης μεταξύ των δύο στοιχείων.

Μηχανισμοί Προσοχής Πολλαπλών Κεφαλών (Multihead Attention Mechanisms)

Η παραλλαγή αυτή επεκτείνει την έννοια της αυτο-προσοχής εφαρμόζοντας τον μηχανισμό προσοχής ταυτόχρονα σε πολλαπλές περιπτώσεις. Αντί να μαθαίνει ένα μόνο σύνολο μετασχηματισμών, η προσοχή πολλαπλών κεφαλών έχει τη δυνατότητα να αναπτύσσει ανεξάρτητες ομάδες μετασχηματισμών που ονομάζονται κεφαλές. Η ιδέα πίσω από αυτήν την εφαρμογή είναι ότι κάθε κεφαλή μπορεί να επικεντρωθεί σε συγκεκριμένα μοτίβα και να καταγράφει διαφορετικούς τύπους συσχετισμών μεταξύ των στοιχείων εισόδου. Αυτές οι ξεχωριστές αναπαραστάσεις συνδυάζονται για να σχηματίσουν την τελική έξοδο.

Κάθε κεφαλή έχει το δικό της σύνολο διανυσμάτων Query, Key και Value και εφαρμόζει τους δικούς της γραμμικούς μετασχηματισμούς σε αυτά. Επειδή αυτοί οι μετασχηματισμοί μαθαίνονται ξεχωριστά, κάθε κεφαλή μπορεί, ενδεχομένως, να εκπαιδευτεί να εστιάζει σε διαφορετικά πράγματα.

Μεθοδολογία

Σύνολο δεδομένων JIGSAWS

Για την εκπαίδευση και αξιολόγηση του προτεινόμενου μοντέλου αναγνώρισης χειρονομιών, χρησιμοποιήθηκε το σύνολο δεδομένων JIGSAWS. Το JIGSAWS έχει αναπτυχθεί με τη χρήση ρομποτικών χειρουργικών συστημάτων, και χρησιμοποιείται κυρίως για τη μελέτη της αναγνώρισης χειρουργικών ενεργειών και της αξιολόγησης δεξιοτήτων. Η έρευνα επικεντρώνεται στην εργασία της συρραφής, η οποία περιλαμβάνει μια σειρά ενεργειών όπως το πιάσιμο μιας βελόνας, η εισαγωγή της στον δερματικό ιστό και το τράβηγμα του ράμματος.

Η συρραφή εκτελείται από 8 χρήστες, χερουργούς με ποικίλα επίπεδα εμπειρίας στη χρήση χειρουργικών ρομπότ, καθένας από τους οποίους έχει ολοκληρώσει 5 δοκιμές. Τα δεδομένα από ένα δοκιμαστικό του χρήστη Η δεν είναι διαθέσιμα, επομένως, το σύνολο δεδομένων περιλαμβάνει συνολικά 39 δοκιμές. Το JIGSAWS αποτελείται από κινηματικά δεδομένα που συνοδεύονται από βίντεο, το οποίο σε αυτή την εργασία δε θα χρησιμοποιηθεί. Τα κινηματικά δεδομένα περιλαμβάνουν πληροφορίες σχετικά με τη θέση, την ταχύτητα και τον προσανατολισμό των ρομποτικών εργαλείων. Αρχικά, αξιοποιήσαμε μόνο ένα υποσύνολο 14 χαρακτηριστικών, αλλά στη συνέχεια τα αυξήσαμε σε 21. Κάθε χρονική στιγμή, στα δεδομένα συνδέεται μια ετικέτα που αντιπροσωπεύει μια συγκεκριμένη χειρουργική χειρονομία. Το σύνολο δεδομένων JIGSAWS ορίζει 15 χειρονομίες, από τις οποίες οι 10 είναι σχετικές με την εργασία της συρραφής.

Προεπεξεργασία

Πριν από την εισαγωγή των κινηματικών δεδομένων σε ένα μοντέλο μηχανικής μάθησης, συνήθως είναι απραίτητο ένα επιπλέον βήμα που περιλαμβάνει τη μετατροπή των ακατέργαστων δεδομένων σε μορφή κατάλληλη, σύμφωνα με την αρχιτεκτονική του μοντέλου ώστε να μπορεί να αξιοποιηθεί πιο αποτελεσματικά από αυτό. Βασικό κομμάτι αυτής της διαδικασίας είναι η **κανονικοποίηση**, η οποία προσαρμόζει τα μεγέθη των χαρακτηριστικών σε μια κοινή κλίμακα, διασφαλίζοντας ότι είναι συγκρίσιμα. Ιδιαίτερα σημαντικό μέρος της διαδικασίας είναι η κανονικοποίηση, η οποία πραγματοποιήθηκε υπολογίζοντας τη **μέση τιμή** και την **τυπική απόκλιση** για τα δεδομένα κάθε δοκιμής, ξεχωριστά. Το σύνολο των δεδομένων χωρίζεται σε τρία μέρη: το **training set**, το οποίο χρησιμοποιείται για την εκπαίδευση, το **validation set**, το οποίο χρησιμοποιείται για την επιμέρους αξιολόγηση κατα τη διαδικασία της εκπαίδευσης, και το **test set**, το οποίο αξιοποιείται μόνο αφού έχει ολοκληρωθεί η εκπαίδευση του δικτύου, με σκοπό την τελική αξιολόγηση της επίδοσής του και της ικανότητας γενίκευσης. Η τελική επικύρωση της ακρίβειας του μοντέλου θα γίνει με την τεχνική **Leave-one-out**, όπου κάθε φορά όλες οι δοκιμές ενός χρήστη χρησιμοποιούνται ως test set, και η διαδικασία επαναλαμβάνεται N φορές, όπου N είναι ο αριθμός των χρηστών. Το τελικό αποτέλεσμα προκύπτει ως μέσος όρος των επιμέρους.

Αρχιτεκτονική LSTM

Το δίκτυο LSTM αποτελείται από ενα στρώμα μάσκας το οποίο εξαιρεί τις τιμές padding (-1), δύο στρώματα LSTM με 256 και 32 νευρώνες αντίστοιχα. Κάθε στρώμα LSTM περιλαμβάνει κανονικοποίηση L2, η οποία προσθέτει έναν όρο ποινής για την αποφυγή υπερπροσαρμογής (overfitting), διατηρώντας τα βάρη μικρά. Επίσης, κάθε στρώμα LSTM ακολουθείται από ένα στρώμα drop out, το οποίο απομακρύνει τυχαία μερικούς νευρώνες κατά την εκπαίδευση με σκοπό και πάλι την αποφυγή overfitting. Επιλέξαμε τη συνάρτηση Categorical Cross Entropy ως συνάρτηση απώλειας, σε συνδυασμό με την ενεργοποίηση softmax. Ο αλγόριθμος βελτιστοποίησης AdamW, μια παραλλαγή του Adamptive Movement Estimation (Adam), ενσωματώνει την ορμή και την αποσύνθεση βάρους για καλύτερη σύγκλιση.

$$L = -\sum_{i=1}^{10} t_i log(p_i)$$

Μετά την εξέταση της κατανομής των ετικετών στο σύνολο δεδομένων, βρήκαμε ότι είναι έντονα ανισοβαρής. Αναθέτοντας ένα **βάρος ανάλογο με το αντίστροφο της κατανομής των κλάσεων** σε κάθε κλάση και περνώντας ένα λεξικό των εν λόγω βαρών στη συνάρτηση εκπαίδευσης, διασφαλίζουμε ότι η υποεκπροσωπούμενη κλάση δεν αγνοείται.

Προτεινόμενα μοντέλα ενίσχυσης του LSTM

Feature engineering: η διαδικασία δημιουργίας ενός νέου χαρακτηριστικού μέσω της μετατροπής ενός υποσυνόλου των διαθέσιμων μεταβλητών εισόδου σε ένα πιο αποτελεσματικό σύνολο εισόδων. Προσπαθήσαμε να συνδυάσουμε δύο υπάρχοντα χαρακτηριστικά, τις γωνίες των αριστερών και δεξιών λαβίδων, σε μια μεταβλητή με τέσσερις δυνατές καταστάσεις που μπορούν να περιγραφούν ως "Κοινή κατάσταση λαβίδων":

- Και οι δύο κλειστές
- Αριστερή ανοιχτή - Δεξιά κλειστή
- Αριστερή κλειστή - Δεξιά ανοιχτή
- Και οι δύο ανοιχτές

Υβριδικό μοντέλο LSTM - CRF: Μετά την εξέταση του πίνακα μετάβασης των πραγματικών ετικετών του συνόλου δεδομένων και την επισήμανση ότι παρουσιάζεται ιδιαίτερα αραιός, προσπαθήσαμε να ενσωματώσουμε στο δίκτυό μας ένα μοντέλο που θα μπορούσε να επωφεληθεί από τις περιορισμένες πιθανότητες μετάβασης του συνόλου δεδομένων. Δεδομένου ότι το LSTM έχει σχεδιαστεί για να καταγράφει μακροχρόνιες εξαρτήσεις διατηρώντας μνήμη ενώ αναλύει ακολουθιακά δεδομένα, καταλήξαμε σε έναν συνδυασμό με ένα CRF, αναμένοντας ότι μπορεί να εστιάσει σε διαφορετικές πτυχές των δεδομένων και να μάθει μοτίβα συμπληρωματικά προς το LSTM που μπορούν να αξιοποιήσουν τη διακριτή κατανομή πιθανοτήτων μετάβασης. Στην προτεινόμενη αρχιτεκτονική μας, τοποθετούμε το CRF μετά το βασικό μέρος των 2 στρωμάτων LSTM. Η πρόβλεψη του LSTM για την προηγούμενη κατάσταση συνδυάζεται με την αρχική είσοδο και τροφοδοτείται στο CRF. Για την επιλογή της τελικής εξόδου εφαρμόσαμε έναν πολύ απλό μηχανισμό απόφασης που την έξοδο του CRF ως εναλλακτική λύση, όταν επίπεδο αβεβαιότητας του LSTM ξεπερνάει ένα κατώφλι.

Συνδυασμός LSTM με μηχανισμό προσοχής (Attention Mechanism): Μία ακόμα προσέγγιση που εξετάστηκε για τη βελτίωση της συνολικής απόδοσης του μοντέλου LSTM, περιλάμβανε την ενσωμάτωση ενός μηχανισμού MultiHead Attention, που μπορεί να δώσει στο μοντέλο την ικανότητα να εστιάζει σε διαφορετικά μέρη της ακολουθίας δεδομένων. Πειραματιστήκαμε με διαφορετικές διατάξεις του δομικού μπλοκ αυτού του μοντέλου και βρήκαμε πιο αποτελεσματική την τοποθέτηση του μηχανισμού προσοχής με 4 κεφαλές ανάμεσα στα δύο επίπεδα LSTM. Η ακολουθία εισόδου επεξεργάζεται από το πρώτο επίπεδο LSTM, το οποίο καταγράφει τις χρονικές εξαρτήσεις και εξάγει μια ακολουθία κρυφών καταστάσεων, πάνω στην οποία λειτουργεί το MultiHead Attention, προορισμένο να καταγράψει πιο σύνθετες εξαρτήσεις. Η ίδια ακολουθία χρησιμοποιείται ως query και value vector, οδηγώντας σε Self Attention.

Χαρακτηριστικά	Ακρίβεια (%)	F1 score
LSTM με 14 χαρακτ.	78.82	0.597
LSTM με 20 χαρακτ.	79.47.5	0.6
LSTM με 20 χαρακτ. + Κοινή κατάσταση λαβίδων	80.58	0.606

Table 1: Συγκριτικός πίνακας επιλογής μεταβλητών για το LSTM. Στα 14 χαρακτηριστικά περιλαμβάνονται μεταβλητές θέσης, γραμμικής ταχύτητας και γωνίας λαβίδας. Στα 20 προστίθενται και οι γωνιακές ταχύτητες

Πειραματικά αποτελέσματα

Αρχικά παραθέτουμε ένα συγκριτικό πίνακα των αποτελεσμάτων της LSTM αρχιτεκτονικής όπου παρουσιάζονται τα αποτελέσματα με τη χρήση διαφορετικών υποσυνόλων των διαθέσιμων κινηματικών μεταβλητών αλλά και της εξαγόμενης μεταβλητής που ονομάσαμε "Κοινή κατάσταση λαβίδων".

Βλέπουμε ότι η προσθήκη των γωνιακών ταχυτήτων και της μεταβλητής κοινής κατάστασης ενισχύουν την επίδοση του δικτύου.

Για να αποκτήσουμε καλύτερη κατανόηση των προκλήσεων στην αναγνώριση χειρονομιών, αναλύσαμε περαιτέρω κάθε δοκιμή. Με μια πρώτη ματιά, είναι πολύ σαφές ότι το μοντέλο δεν μπορεί να γενικεύσει εξίσου καλά για κάποιους χρήστες σε σύγκριση με άλλους. Μπορούμε να συμπεράνουμε ότι αν η συντριπτική πλειοψηφία των δειγμάτων μιας κλάσης είναι συγκεντρωμένη μεταξύ λίγων δοκιμών, η ικανότητα του μοντέλου να γενικεύει και να αναγνωρίζει τις περιπτώσεις αυτής της κλάσης επηρεάζεται. Συγκρίνοντας άμεσα την ακρίβεια ανά κλάση με τον αριθμό των περιπτώσεων ανά κλάση, μπορούμε να επιβεβαιώσουμε ότι στις περισσότερες περιπτώσεις υπάρχει ισχυρή συσχέτιση μεταξύ της αντιπροσώπευσης της κλάσης και του ποσοστού αναγνώρισης της. Είναι πιο εμφανές στις κλάσεις G9 ("Χρήση δεξιού χεριού για βοήθεια στη σύσφιξη ράμματος"), G10 ("Χαλάρωση ράμματος") και G11 ("Τέλος συρραφής και μετάβαση σε τελικά σημεία").

Η εισαγωγή του CRF, σε μεμονωμένα σημεία καταφέρνει να διορθώσει τα λανθασμένα αποτελέσματα του LSTM, ωστόσο δε βελτίωσε τα συνολικά αποτελέσματα, ενώ σε κάποια σημεία της εξόδου του CRF παρατηρείται υπερτμηματισμός. Παρόλα αυτά, η εξέταση του αποτελέσματος ανά δοκιμαστικό, μας οδηγεί στο συμπέρασμα ότι η ανάπτυξη μίας μεθόδου στρατηγικής συγχόνευσης των δύο εξόδων έχει την προοπτική να παράξει καλύτερα αποτελέσματα.

Τέλος, το επίπεδο προσοχής κατάφερε να βελτιώσει την ικανότητα γενίκευσης του μοντέλου, ακόμα και σε δοκιμές που περιλαμβάνουν κλάσεις με μικρότερη εκπροσώπηση. Αυτή η αρχιτεκτονική έφερε τα καλύτερα αποτελέσματα από όλες τις προσεγγίσεις που δοκιμάσαμε με μέση ακρίβεια **81.56%**, βελτιώνοντας, μάλιστα, αντίστοιχες επιδόσεις που αναφέρονται στη διεθνή βιβλιογραφία, δεδομένων των περιορισμών που ετέθησαν, όπως φαίνεται στον πίνακα 2.

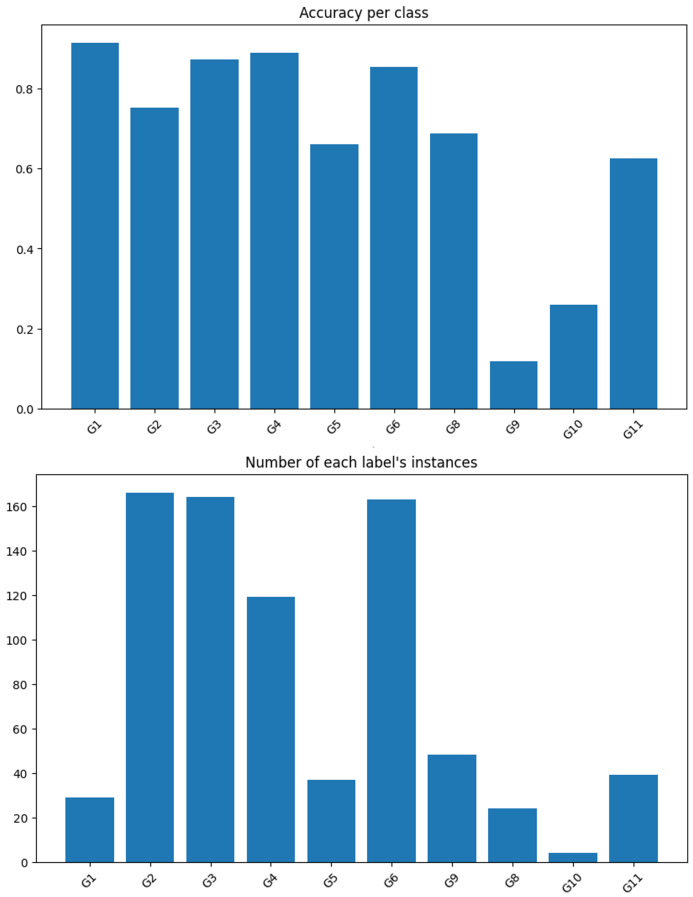

Figure 3: Επάνω: Ακρίβεια ανά ετικέτα. Κάτω: Αριθμός εμφανίσεων ανά ετικέτα. Τα αποτέλεσμα αντιστοιχούν στο μοντέλο LSTM με 20 χαρακτηριστικά + Κοινή κατάσταση λαβίδων

Μέθοδος	Ακρίβεια (%)	Χρονιά
Skip-Chain CRF [13]	80.29	2015
Forward LSTM [6]	80.5	2016
Μοντέλα που παρουσιάζονται στη διπλωματική εργασία		
LSTM με 14 χαρακτ.	78.82	2024
LSTM με 20 χαρακτ.	79.47	2024
LSTM με 20 χαρακτ. + Κοινή κατάσταση λαβίδων	80.58	2024
Υβριδικό μοντέλο LSTM με Self-Attention	**81.56**	2024

Table 2: Συγκριτικά αποτελέσματα διαφορετικών προσεγγίσεων από τη διπλωματική εργασία και τη διεθνή βιβλιογραφία, για το dataset JIGSAWS, με τη χρήση κινηματικών δεδομένων σε μοντέλα που μπορούν να αξιοποιηθούν σε πραγματικό χρόνο.

Contents

Chapter 1

Introduction

1.1 Motivation

During the last two decades, we have seen remarkable progress in the development of Artificial Intelligence and Machine Learning techniques that managed to revolutionize various fields. The fast-paced evolution of data science techniques and Deep Learning, hasn't gone unnoticed by the healthcare industry. Medical professionals increasingly recognize the potential of technology integration to enhance patient care, streamline workflows, and improve overall efficiency [18]. One of the most promising applications that can harness these technological advancements to refine medical procedures is robotic surgery. The usage of surgical robots can provide valuable data, which can be utilized by deep learning models and contribute to our understanding of the surgical tasks, assess the execution, provide assistance during the procedure and generally advance towards the automation of surgery. These data have been leveraged to construct datasets like JIGSAWS [8], thereby facilitating a more systematic evaluation of diverse methodologies. An integral part of this process is the segmentation of surgical tasks in atomic gestures and the ability to recognize them.

Automated surgical gesture recognition aims at automatically identifying meaningful action units within surgical tasks that constitute a surgical intervention [28]. Although the clinical adoption of autonomous robotic surgery still seems distant, a significant body of research has been published on gesture recognition, presenting substantial findings. The majority of this work primarily focuses on improving accuracy metrics for gesture recognition, which is categorized as a "classification" task. This thesis, alongside the improvements of the performance of kinematic data models also aims to delve deeper into the inherent challenges and limitation of current models by conducting an analysis of the underlying weaknesses of each approach.

1.2 Related work

The literature contains numerous works presenting a range of approaches to gesture classification. We notice that earlier works focused on kinematic data and tried to tackle the problem using probabilistic graphical models, like Hidden Markov Models (HMM) ([21], [29] and [26]) or Conditional Random Fields (CRFs) ([27], [13]). Lin et. al [17] used a Bayes Classifier for gesture classification and they also presented a feature-processing technique, aiming to reduce the dimensionality of the data and extract relevant features for the task. Chen et. al [3] after processing raw motion data and extracting relevant features that can be used to distinguish between different gestures, they employed Support Vector Machine (SVM) algorithm for hand gesture recognition. The used a database of dynamic hand gestures including 3D motion trajectories of the numbers and the alphabet. Their system aims to recognize the gesture as early as possible in the motion sequence, improving the responsiveness of the interaction.

A review of these approaches reveals a key limitation: they are restricted to modeling only frame-to-frame or segment-to-segment transitions [28]. In contrast, recent efforts have sought to overcome this constraint by utilizing deep learning techniques that have the ability to capture complex long-term dependencies in sequential data. Recurrent Neural Networks (RNNs) are known for their ability to learn temporal characteristics, facilitating their widespread use. In most cases, the preferred RNN variant is the Long Short Term Memory (LSTM) network. DiPietro et. al [6] trained a Forward LSTM for recognizing not only gestures, but also maneuvers, which they defined as longer, higher-level activities that consist of multiple gestures. They assessed the effectiveness of their method on two benchmark datasets: JIGSAWS for gesture recognition and MISTIC-SL for maneuver recognition.

In some cases bidirectional variants of LSTM and Gated Recurrent Unit (GRU) ([5]) were used. These models have the ability to processes data sequences both ways, but at a cost of a constraint, this method cannot be used in real time because their predictions are based on both past and future information. The study found that a single set of hyperparameters can lead to strong performance across both gesture and maneuver recognition tasks.

Finally, some multimodal architectures have been presented that utilize a combination of vision and kinematics. The visual data are typically fed to a Temporal Convolutional Network (TCN). The unified model Fusion-KV [20] demonstrates robustness in complex and realistic surgical scenarios, including dry-lab, cadaveric, and in-vivo experiments. Keshara Weerasinghe et. al [31] presented various architectures, which not only utilized the video data through Temporal Convolutional Networks (TCN) but also extracted features that

represent the interactions between surgical instruments by applying a tool segmentation mask on video data. Their approach also uses a Mulithead attention encoder.

1.3 Contribution

This thesis attempts to present a deep learning framework for real-time Gesture Recognition tailored for surgical procedures. All the presented models are trained and evaluated using the suturing task trials of the JIGSAWS dataset, which contains a variety of kinematic variable captures.

- We train and analyze different architectures under two constraints:
  - We utilize **kinematic** data only, as input.
  - We propose networks that **can be used online**, meaning that they should be able to process data in a streaming fashion, using a sliding window, with a maximum one-second delay.
- We begin by establishing a strong baseline using a Long Short-Term Memory (LSTM) network, a well-suited architecture for sequential data, given its ability to capture long-term relationships.
- To further improve performance, two hybrid extensions were proposed, incorporating Conditional Random Fields (CRFs) and attention mechanisms, respectively.
- Through rigorous experiments, we compare the performance of different architectures by using metrics such as accuracy and recall, along with visualizations of model predictions compared to ground truth, attempting to gain insights into their strengths and weaknesses.
- We delve into the inherent complexities of the problem and explore the impact of various model parameters on performance.
- Our proposed methods improve the state-of-the-art results on the JIGSAWS dataset, given imposed constraints.

Chapter 2

Theoretical foundation

In this chapter, machine learning concepts relevant to gesture recognition are discussed, focusing on techniques suitable for processing sequential data that are integral to our considered approaches, such as Long Short-Term Memory (LSTM) networks, Conditional Random Fields (CRFs), and attention mechanisms.

2.1 Machine Learning

Machine Learning belongs to the field of Artificial Intelligence (AI) and describes a set of algorithms that learn from data rather than following explicit instructions. These algorithms are able to statistically analyze data to identify patterns and make predictions or decisions. Machine learning models learn by iteratively adjusting their internal parameters to minimize error, based on the patterns observed in training data. [10]. The purpose is to use the experience gained from training to generalize to new, unseen data.
Machine learning algorithms draw inspiration from the human cognitive process. Akin to human learning, machine learning algorithms excel at tasks that require pattern recognition and decision-making, often involving implicit, intuitive processes. These tasks, such as natural language understanding or motor control, are challenging to formalize into explicit rules.

2.1.1 Deep Learning

Deep learning is a subfield of Machine learning that leverages the power of artificial neural networks. Typically, deep learning architectures contain a hierarchical structure of stacked layers. Each layer generates an abstract representation of the given data, by passing them through nonlinear transformations. The layers at the bottom capture simpler, more primitive information, which are then used by the layers built on top of them to gradually create a complex representation. They often consist of an input layer, an output

layer, and several hidden layers between them. There is no standard method for deciding the exact number of hidden states to use in a neural network, but generally it is dictated by the complexity of the task at hand.

There are two kinds of artificial neural networks, **feedforward neural networks**(FNN) and **recurrent neural networks** (RNNs) The difference between the two lies in the flow of information. In FFNs each neuron can only receive data from the neurons of the previous layers, while RNNs allow cyclical connections called feedback loops. When each neuron of a layer is connected to every node of the next layer, it is called a **fully connected layer** and a neural network consisting of fully connected layers is called a **fully connected neural network** [2]

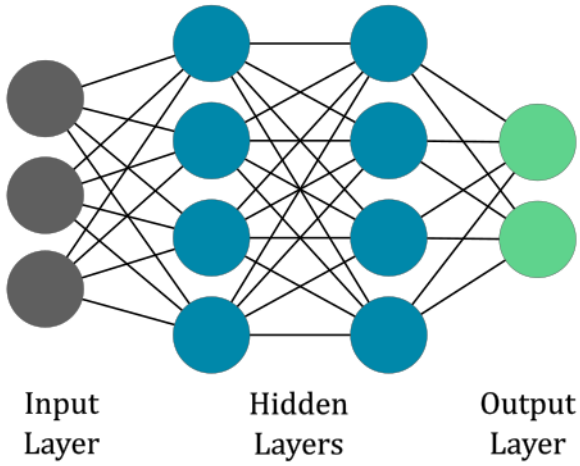

Figure 2.1: Example of a fully connected FFN with two hidden layers

2.1.2 Supervised Learning

Regularly, the input data consist of a set of measurable properties called **features**. Supervised machine learning is a common training technique that uses **labeled data**. The features are paired with the desired output, and then fed to the model as input. During the training process, the model, given the input features, generates an output in the form of a probability vector. This output is then compared to the target output, and the model's parameters are iteratively adjusted to minimize the discrepancy between them. [2].

2.2 Gradient-Based Optimization

Optimization is a critical part of deep learning algorithms, involving the process of searching for an objective function's f(x) **optimal value** (maximum or minimum) [10]. Our goal is to train the model, so that for an input (x), it makes a prediction as close to the ground truth (y) as possible, so we usually express this problem as the optimization task of finding the **minimum** of a function f(x) that depicts the difference between the predicted and the true label. This function is often called **loss function**, **cost function** or **error function**

$$x^* = argmin f(x)$$

A widely used optimization algorithm is **gradient descent** (GD). Given a function $y = f(x)$ we first calculate it's derivative $f'(x) = \frac{dy}{dx}$ which represents the slope of $f(x)$ at point x. By following the negative derivative of using a small step size ϵ, we decrease y.

$$x_{n+1} = x_n - \epsilon f'(x_n), \tag{2.1}$$
$$f(x_n) \geq f(x_{n+1}) \tag{2.2}$$

If we repeat this process until we reach a point where $f(x)' = 0$ meaning that small steps cannot improve (minimize) y any further. This point is known as **local minimum**. The optimal value of f(x) is the **global minimum**, but locating it using GD can be challenging, especially when there are many local mimima, so when finding one, we have to decide whether it is sufficiently good or if a more exhaustive search for the global minimum is necessary [2].

However, deep learning models mostly involve functions with multiple variables, and since derivative only works for functions with single variables we have to extend the same concept to multi-dimensional space by using the **gradient** of the function.

$$\nabla f(x_1, x_2, \ldots, x_n) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x_1, x_2, \ldots, x_n) \\ \frac{\partial f}{\partial x_2}(x_1, x_2, \ldots, x_n) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x_1, x_2, \ldots, x_n) \end{bmatrix} \tag{2.3}$$

$$a_{n+1} = a_n - \epsilon \nabla F(a_n), \tag{2.4}$$
$$F(a_n) \geq F(a_{n+1}) \tag{2.5}$$

2.2.1 Weights and Biases

Each neuron of a neural network receives data from all the other neurons it is connected to, as stated in 2.1.1. These incoming data signals are not passed to

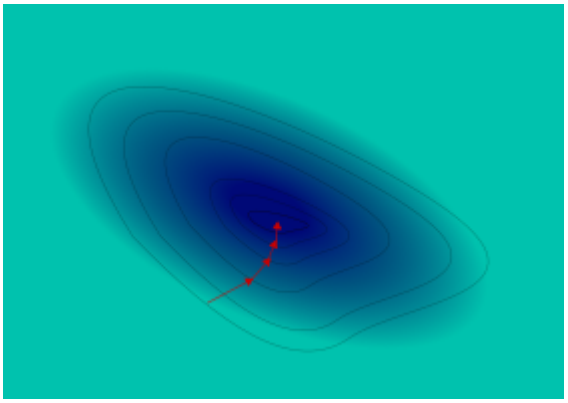

Figure 2.2: Gradient descent depiction in a 2-dimensional space

next neurons unaltered, instead, at each neuron they are **weighted** and **biased**. The weight W_{ij} and the bias b_i together form the **preactivation** z_i of a neuron:

$$z_i(x) = b_i + \sum_{j=1}^{n} W_{ij} x_j \tag{2.6}$$

Weights and biases form the **adjustable properties** of the model, and they are iteratively modified in order to improve the model's overall performance. [4] The weights determine the strength of the connection between the neurons and the impact of the received data, so they can adjust the importance of specific information. Biases, on the other hand, add an extra layer of flexibility to the network and act as a threshold, allowing the neuron to activate even when the weighted sum of the inputs is not enough to activate it on its own.

2.2.2 Activation function

The preactivation result is then fed into a scalar-valued function called **activation function**. This function produces the final output of the neuron and decides whether it should be activated. The activation function plays a crucial role in shaping the model's ability to learn complex relations by introducing **non-linearity**. Some of the most commonly used activation functions are the following:

1. Sigmoid: continuous function with output range of [0,1]. For x near zero, it is very steep and almost linear, but as the input moves away from zero the output approaches 0 (for negative values) and 1 (for positive values).

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

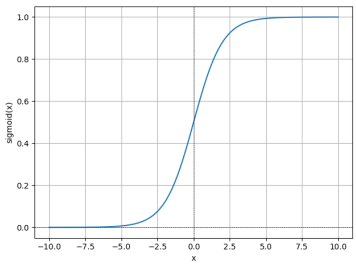

Figure 2.3: Sigmoid function

2. **Tanh**: Hyperbolic tangent, output range of [-1,1]. It is a shifted and scaled version of the sigmoid function.

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

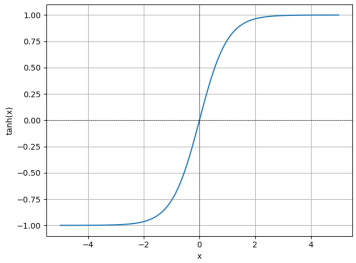

Figure 2.4: Tanh function

3. **ReLU**: Rectified Linear Unit. It is very widely used in the hidden layers of deep neural networks. It's output ranges in $[0, \infty]$. Scale-invariant

$$ReLU(x) = \max(0, x)$$

4. **Softmax**: Ideal for creating probability distribution, since the sum of it's output values equals 1. Mainly used in the output layer.

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.7}$$

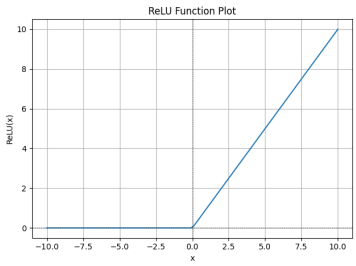

Figure 2.5: ReLU function

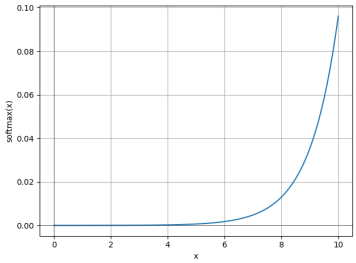

Figure 2.6: Softmax function

2.2.3 Backpropagation

In chapter 2.2.1 we analyzed the trainable parameters of the model. Usually, the weights and biases are initialized randomly. Subsequently, an iterative algorithm known as **backpropagation** is employed to determine which weights and biases should be adjusted and by what magnitude. This algorithm iteratively updates the model parameters to optimize performance.

After propagating the input data through every layer, with each neuron applying its weights and biases and passing the result through an activation function, a final output is shaped. This procedure is called **forward pass**. After a forward pass is completed, the **loss function** quantifies the error between the predicted and actual outputs. This error is transmitted backwards layer by layer and by calculating the gradient of the loss function with respect to the weights, we determine how to adjust these parameters in order to minimize the error. Biases are updated too, in a similar manner. Backpropagation algorithm takes advantage of the chain rule of derivatives that allows us to find the derivative of

a composite function $f(g(x))$ [15]:

$$\Delta g = \frac{\partial g}{\partial x}\Delta x \tag{2.8}$$

$$\Delta f = \frac{\partial f}{\partial g}\Delta g \tag{2.9}$$

$$\frac{\partial f(g(x))}{\partial g} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} \tag{2.10}$$

$$\tag{2.11}$$

In the context of backpropagation, firstly we compute the derivatives of the loss function L with respect to the output y, and then we apply the chain rule to find the gradient of the loss with respect to the preactivation function z (as given in equation 2.12).

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial y} \tag{2.12}$$

Ultimately, we compute the gradient of the loss with respect to the parameters, weight W and bias b and using the computed gradients, we update the parameters.

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial W} = x \cdot \frac{\partial L}{\partial z} \tag{2.13}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial b} = \frac{\partial L}{\partial z} \tag{2.14}$$

$$\tag{2.15}$$

2.2.4 Training parameters

Before training a Neural Network, we must adjust a few crucial parameters that determine the model's performance, it's ability to generalize to unseen data, and also affect its computational requirements. These are also referred to as **hyperparameters**. Each hyperparameter affects the model differently, and it is important to find a good balance between accuracy and efficiency.

- **Optimizer** : it is essentially an algorithm that uses as input the gradients calculated by backpropagation and decides how to adjust the network's weights and biases at each training step, in order to reduce loss. The most widely used optimizers are Stochastic Gradient Descent (SGD) and Adaptive Movement Estimation (Adam).
- **Learning rate**: this hyperparameter controls the magnitude of weight adjustments during training. Selecting a large learning rate can expedite the model's convergence, but it carries the risk of overshooting the optimal solution. Conversely, when the learning rate is too small, the model may become trapped in a local minimum, hindering its ability to find the global optimum.

- **Number of epoch**: an epoch represents a complete pass through the entire training dataset. The optimal number of epochs depends on the dataset's size and complexity. Making too many passes on a relatively small dataset can lead to **overfitting**, which means that the model starts to increase its accuracy by directly correlating the input values to the desired output, so instead of learning the task it just memorizes the dataset. On the other hand, a number of epochs can stop the training prematurely, preventing the model from reaching its full learning potential. This is called **underfitting**.
- **Batch size**: the number of input elements utilized in each forward pass. The batch size influences the accuracy of gradient estimation and the convergence speed of the model.

2.3 Long Short Term Memory

2.3.1 Recurrent neural network

As discussed in 2.1.1, what sets apart RNNs from FFNs is the flow of information within the network, which contains feedback loops (cyclical connections). These directed cycles enable the model to incorporate information from past states into the calculation of the current output. This creates a form of memory, allowing the RNN to learn and utilize temporal dependencies within the data. The hidden state of the RNN is often referred to as the **Memory State** for this reason.

While the recurrent nature of RNNs makes them very efficient in tasks that involve sequential data, they are also computationally expensive compared to traditional machine learning models.

2.3.2 Basic Architecture

The fundamental unit of an RNN is called **Recurrent Unit**. It is designed to handle sequential data by propagating data from earlier time steps to the current processing step [24]. At each time step, the recurrent units update their hidden state based on the current input and the previous hidden state, which allows them to capture sequential dependencies.

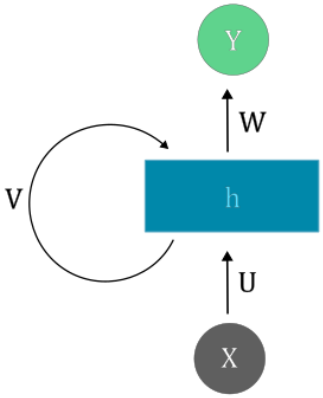

Figure 2.7: RNN example

Figure 2.7 shows the anatomy of a basic RNN, where X is the input, U, W and V are the parameters of the networks which are shared across timesteps. h is the hidden state that is part of the feedback loop in order to retain the memory of the network, and Y is the output.

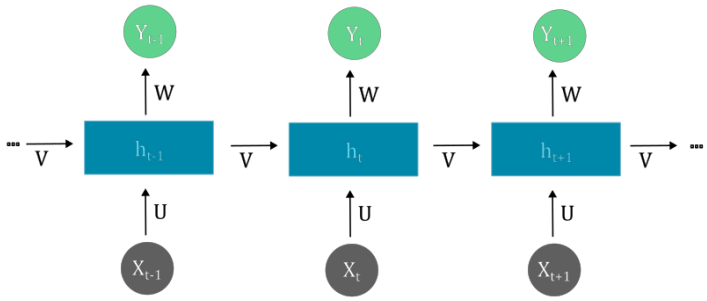

Figure 2.8: RNN unfolded example

By unfolding the RNN blueprint (fig.2.8) we can get a better understanding of it's underlying architecture and it's sequential nature.

2.3.3 Vanishing Gradient Problem

As explained in section 2.2 Gradient Descent is a very common optimization algorithm in neural networks. Although it is effective at finding the local minimum of a cost function, it is also susceptible to a problem called the **Vanishing Gradient Problem**. During backpropagation the derivative of the activation function is used for calculating the weight updates for the previous layer, but as this process continues through the layers, the gradients are getting progressively smaller. If multiple layers are involved, the gradient can completely vanish, impeding the learning process. Conversely, the **Exploding Gradient Problem** occurs when gradients grow exponentially, leading to unstable training and potential divergence.

The unique architecture of the RNNs, makes them particularly prone to those two issues, and since they are more prevalent when the number of layers is large, the network's ability to learn longer dependencies can be limited [24].

The vanishing and exploding gradient problems are often attributed to the saturation regions of activation functions like sigmoid and hyperbolic tangent. When inputs to these functions become very large or very small, the output approaches a constant value, resulting in small derivatives and consequently small gradients during backpropagation.

2.3.4 LSTM Architecture

Long Short Term Memory is an RNN variant, first introduced in 1997 by Hochreiter & Schmidhuber [9], designed to mitigate the vanishing gradient issue [23]. This goal was achieved by implementing a unique architecture that contains

three separate gates, each with a distinct task, in the networks fundamental unit called **memory cell**. Instead of maintaining and propagating through the network a single hidden state, LSTMs use the three gate, **input**, **forget** and **output**, as illustrated in Fig.2.9 in order to decide what information is added and removed from the cell and which part of it will form it's output. These adjustments to the memory state allow the network to retain longer and meaningful relationships [24].

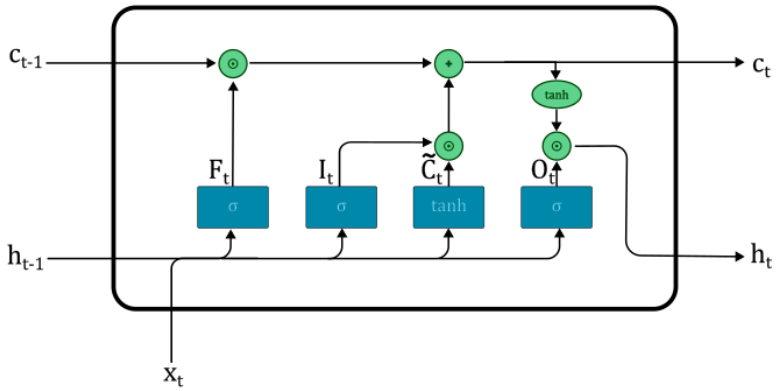

Figure 2.9: LSTM memory cell architecture

Each memory cell receives data from three sources:

- Input x_{t-1}
- Hidden State h_{t-1}
- Memory c_{t-1}

Afterward, they pass through the gates so the new memory and output will be produced.

Forget Gate: This gate is responsible for deciding which part of the memory is no longer useful and should be forgotten.
Its value arises from the previous hidden state h_{t-1} and the input x_t, which are concatenated and put through a sigmoid $/sigma$ activation function:

$$F_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

- W_f: Forget weight matrix
- b_f: Forget weight bias

- $[h_{t-1}, x_t]$: Concatenation of the previous hidden state and the current input

Input Gate and Candidate Memory: Those two elements are responsible for deciding what new information is going to be added to the memory. They act as a filter, allowing only the useful parts of the input to pass. The input gate, similar to the forget gate, uses the previous hidden state h_{t-1} and the input x_t and a sigmoid function $/sigma$. The Candidate Memory Gate, $\tilde{C}_t$ on the other hand, employs $tanh$ as the activation function. Each gate has its own weight and bias

$$I_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

After calculating all the necessary gate results, the previous memory state is multiplied by the forget gate value F_t, while the candidate memory $\tilde{C}_t$ is multiplied by the Input gate value I_t. The sum of these two products is then passed through a sigmoid function, forming the updated memory state.

$$C_t = \sigma(I_t\tilde{C}_t + F_tC_{t-1})$$

Output Gate: Finally, the output gate's task is to decide which part of the current state will make up the output of the cell. Similarly to the forget and input gates:

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o$$

Using the gate's result, we calculate the new hidden state.

$$h_t = O_t \tanh(C_t)$$

2.4 Conditional Random Fields

2.4.1 Statistical Models

A statistical model is essentially a set of **probability distributions** on the sample space that describes the probability of different outcomes within a dataset [7]. By modeling the way the data are distributed in the context of the sample source, we can make predictions for future samples within the same context.

- **Sample space $\mathcal{S}$** : a set of all the possible outcomes of the examined task.
- **Probability Distribution** : a function that outputs the probability of every possible outcome of the dataset.
- **Parameter Set Θ** : a collection of variables that define the specific characteristics of the statistical model. It is the part of the model that can be adjusted in order to improve its performance for the specific task.

- **Parameterized Statistical Model**: it is defined by the parameter set $\boldsymbol{\Theta}$ and the probability distribution function $P : \Theta \longrightarrow \mathcal{PS}$, which assigns a probability distribution P_θ on the sample space $\mathcal{S}$. Once the parameters are estimated, the model can be used to make predictions about unseen observations.

2.4.2 Probabilistic Graphical Models

Probabilistic Graphical Models (PGMs) use graph structures in order to represent complex probabilistic relationships between variables. The nodes of the graph correspond to random variables, and the edges represent the probabilistic dependencies between the variables they connect [1]. They are often used for tasks like computer vision, natural language processing and classification.

A key feature of PGMs is the usage of **conditional independence** among their variables, which allows the joint distribution of numerous variables to remain compact and efficient [25]. The core concept of conditional independence is that assuming we have three random variables X, Y and Z, if the probability distribution of X given Y and Z is the same as the probability distribution of X given Z, then **X and Y are conditionally independent given Z**:

$$P(X, Y|Z) = P(X|Z) \Rightarrow X \perp\!\!\!\perp Y|Z$$

PGMs are divided in two main categories: **Bayesian Networks** and **Markov Networks**.

2.4.3 Bayesian Network

Bayesian Networks are structured as **Directed Acyclic Graphs (DAGs)**, where the nodes represent random variables and the directed edges signify the conditional dependencies between these variables.

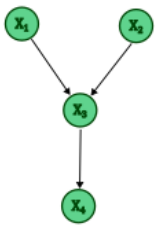

Figure 2.10: An example of a Directed Acyclic Graph

The **joint probability distribution** of a DAG is factorized into a product of the conditional probabilities of each variable given it's parents in the network.

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Parents(X_i))$$

2.4.4 Markov Random Fields

Markov Random Fields, on the other hand, are **undirected graphical models** and their nodes represent random variables that satisfy the **Markov property**. The Markov property asserts that the next state of a stochastic process depends solely on the current state, not on any prior states. Consequently, the probability of an event occurring in the future is determined by the present conditions, rather than by historical events.

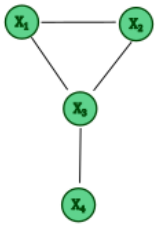

Figure 2.11: An example of a Markov Random Fields network

It can be formally described as:

$$P(X_{t+1}|X_t, X_{t-1}, \ldots., X_0) = P(X_{t+1}|X_t)$$

To make the model more compact and computationally efficient in expressing the joint probability distribution, a product of simpler, local distributions is employed.These subsets, known as **cliques**, are parts of the graph where each node is directly connected to every other node within that subset.

The joint probability of MRFs over cliques is the following:

$$P(X_1, X_2, \ldots, X_n) = \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \psi_C(X_C)$$

where $\mathcal{C}(\mathbf{G})$ is the set of all cliques in the graph G and $\psi_{\mathbf{C}}$ are non-negative potential functions over cliques. $\mathbf{Z}$ is the partition function :

$$Z = \sum_{X_1, X_2, \ldots, X_n} \prod_{C \in \mathcal{C}} \psi_C(X_C)$$

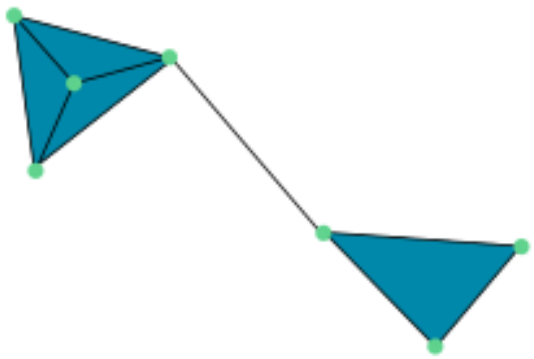

Figure 2.12: Undirected graph example, with its cliques highlighted. Both are maximal cliques, meaning they cannot be expanded to contain more nodes. The left clique can also form several non-maximal cliques if any trio of nodes is selected. Additionally, the left clique is characterized as maximum, as it is the clique with the most nodes in the network.

2.4.5 Conditional Random Field Model

Conditional Random field is a type of discriminative undirected graph model that is built upon the Markov Random Field architecture. They are commonly used in machine learning and especially for tasks where neighboring states are critical. **Discriminative models** are a broad category of statistical models focused on modeling the conditional probability of the output given the input, $P(y|x)$, in contrast to **generative models** whose aim is to model the joint probability distribution of the input and output variables, $P(x, y)$. They are primarily concerned with predicting the correct output category for a given input, rather than modeling the underlying probability distribution of the data.

Given X, a random variable over a data sequence, and Y, a random variable over the corresponding label sequences. For $y_i \in Y, y_i$ has a value range inside a finite class dictionary. The random variables X and Y are jointly distributed, but in a discriminative framework we construct a conditional model p(Y—X) from paired observation and label sequences, and do not explicitly model the marginal p(X).

A CRF is a model where the probability of a label at a particular vertex in the graph depends only on the labels of its neighboring vertices and the global input X. This means that the label at a vertex is conditionally independent of the labels at non-neighboring vertices, given the labels of its neighbors and the global input. This can be formally described with the following definition:

In a graph $G = (V, E)$ where $Y = (Y_v)_{v \in V})$ so that Y is indexed by vertices of G, (X, Y) is a CRF if, when conditioned on X, the random variables $Y_{\mathcal{V}}$ obey the Markov property with respect to the graph:

$$P(Y_v)|X, \{Y_w : w \neq v\}) = P(Y_v|X, \{Y_w : w \sim v\})$$

where $w \sim v$ means that w and v are neighbors [12].

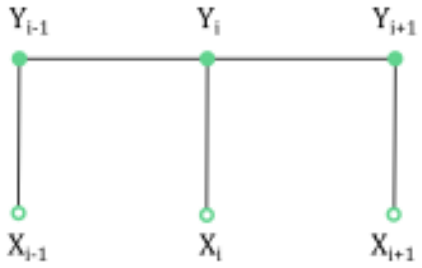

Figure 2.13: An example of a Conditional Random Fields graph

The conditional probability of the output sequence given the input sequence is described by the following equation:

$$P(y|x) = \frac{\exp\left(\sum_i w_i f_i(y, x)\right)}{Z(x)}$$

where:

- $\mathbf{x}, \mathbf{y}$: input and output sequences
- $\mathbf{f_i}(\mathbf{y}, \mathbf{x})$: feature functions that compute the contribution of each feature to the overall score of the output sequence
- $\mathbf{w_i}$: the weights associated with feature functions.
- $\mathbf{Z}(\mathbf{X})$: normalization factor

During training, in order to find the optimal parameters, CRFs use maximum likelihood estimation, which involves adjusting the parameters of the model to maximize the probability of the correct output sequence given the input features. Similar to deep learning models, iterative algorithms like gradient descent are used.

2.5 Attention

2.5.1 Attention mechanisms

Attention mechanisms are a machine learning technique that aims to give models the ability to focus on specific parts of the input data, those that are considered more influential for the given task, and ignore the parts that are deemed less useful. This ability is akin to the human brain's capacity to selectively filter out background noise and distractions, such as in a noisy environment, and isolate specific stimuli. By prioritizing the most informative features, attention mechanisms not only enhance their performance but also reduce the computational resources required by excluding input-output mappings that contribute less significantly. Consequently, they are becoming increasingly popular for tasks like natural language processing, image recognition, and machine translation [30].

Unlike RNNs, which handle data sequentially, self-attention processes the entire sequence simultaneously, enabling parallel computation. This approach speeds up the learning process and enhances its efficiency, especially when working with extensive sequential data. "Nevertheless, attention mechanisms can still effectively cooperate with RNNs.

2.5.2 Self-Attention

Self-Attention is a fundamental attention mechanism variant that attempts to discover and model relationships between different parts of a single input sequence. Self attention mechanisms are essential elements of **transformers**. They use a unique way of encoding information By using three vectors:

- **Query**(**Q**) : Describes what kind of information this element is seeking to acquire from other elements of the sequence.
- **Key**(**K**) : Represents the relevance of this element to a given query.
- **Value**(**V**) : Contains the actual data that the element holds.

The first step for training a model with self-attention is calculating these three vectors for every element of the input sequence. Next, we also evaluate an **attention score** for each pair of input items. This value represents the strength of the association between the two elements. The attention score is often calculated as a dot product, and sometimes it is scaled by the square root of the dimension d_k in order to prevent numerical instability.

$$\begin{aligned} score(i,j) &= Q_i * K_j^T \\ scaled\_score(i,j) &= \frac{score(i,j)}{\sqrt{(d_k)}} \end{aligned} \tag{2.16}$$

The scaled scores are then passed through a softmax function to normalize them into probabilities:

$$\text{attention}(i,j) = \text{softmax}(\text{scaled\_score}(i,j)) \tag{2.17}$$

Finally, the attention weights are multiplied by the value vectors.

$$\text{context\_vector}_i = \sum_j \text{attention}(i,j) \cdot V_j \tag{2.18}$$

This weighted sum creates a context vector for each element, that contains an aggregation of the actual information the element holds based on their influence as it was estimated by the attention mechanism.

In matrix form, the entire self-attention operation can be represented as:

$$\text{attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{2.19}$$

2.5.3 Multi-head attention

Multi-head attention extends the concept of self attention by applying the attention mechanism concurrently in multiple instances. Instead of learning a single set of transformations, multi-head attention has the ability to learn multiple, independent groups of transformations called heads. The idea behind this implementation is that each head can focus on specific patterns and capture different kinds of relationships between data items. These separate representations are combined in order to form the final output.

Each head independently learns its own linear transformations to project input embeddings into distinct query, key, and value subspaces. This allows each head to specialize in different aspects of the input data.

$$\begin{aligned} Q_i &= W_Q^i \cdot X \\ K_i &= W_K^i \cdot X \\ V_i &= W_V^i \cdot X \end{aligned} \tag{2.20}$$

where W_Q^i, W_K^i, W_V^i are projection matrices for the i_{th} head. For each head, the attention mechanism is applied as described in the standard self-attention

$$attention_i(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i \cdot K_i^T}{\sqrt{d_k}}\right) \cdot V_i \tag{2.21}$$

The attention outputs from all heads are concatenated:

$$concat\_heads = [attention_1(Q_1, K_1, V_1), \ldots, attention_h(Q_h, K_h, V_h)] \tag{2.22}$$

The concatenated heads are projected back to the original dimension:

$$multi\_head(Q, K, V) = W_O \cdot concat\_heads \tag{2.23}$$

where W_O is the final projection matrix. In matrix form, the entire multi-head attention operation can be represented as:

$$multi\_head(Q, K, V) = W_O \cdot \text{concat}(head_1(Q, K, V), \ldots, head_h(Q, K, V)) \tag{2.24}$$

where $head_i(Q, K, V)$ is the output of the i^{th} attention head.

Chapter 3

Methodology

This chapter presents the used dataset's structure, the types of kinematic data used, the preprocessing steps employed, and the challenges encountered in transforming raw kinematic data into a suitable format for training and testing machine learning models. We delve into the architecture and configuration of various models, including LSTM and hybrid models, that combine multiple techniques to effectively handle complex gesture transitions.

3.1 Dataset

3.1.1 General Information

The JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) is a renowned dataset in the realm of robotic surgery, extensively utilized for research in robotics, computer vision, and machine learning. This dataset is pivotal for analyzing surgical activities, treating them as dexterous human motions, with the ultimate goal of enhancing the effectiveness and safety of surgical procedures. JIGSAWS is primarily focused on two critical areas of study: **surgical activity recognition** and **skill assessment**.

3.1.2 Surgical Tasks

JIGSAWS dataset contains captures from three basic surgical tasks performed by humans with varying robotic surgical experience. For our evaluation, we focus on the **suturing** task due to its diverse range of gestures and its widespread use as a benchmark in similar research. This allows us to present comparative results effectively. During each suturing trial, a soft, fiber-like material is used, with the incision line and the needle entry and exit points clearly marked on its surface. The user begins by picking up the needle and orienting it towards the first entry point on the right side of the incision line. They then insert the needle using the

left tool tip, guiding it through the material and exiting at the designated point across the incision line. Subsequently, the needle is passed back to the right tool tip, and this sequence is repeated until all marked points have been sutured.

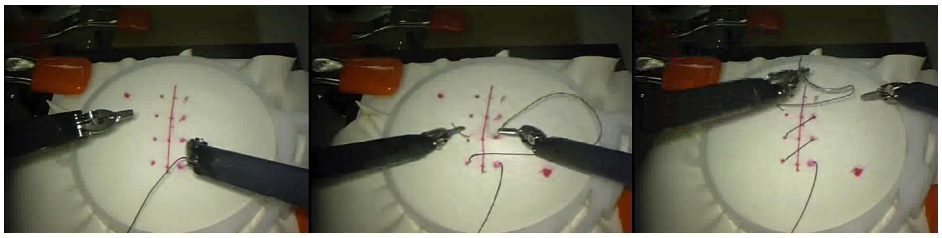

Figure 3.1: Suturing task footage from JIGSAWS

The task is performed by 8 users, indexed by letters

$$"B","C","D","E","F","G","H"$$

and each of them repeats the task 5 times. Since the second trial of user H is missing, we have 39 trials in total to work with.

3.1.3 Data Description

The dataset includes synchronized kinematic and video data, but this thesis focuses solely on the use of **kinematic** data. The tasks are performed using the Da Vinci Robotic Surgical System and the kinematic data are captured at 30Hz. This data encompasses kinematic variables from both the Master Tool Manipulators (MTMs) and Patient Side Manipulators (PSMs), though only the latter are pertinent to our task. For each manipulator, we have 19 variables, including Cartesian positions, linear and angular velocities, rotation matrix and gripper angle. Initially, we experimented with using only linear data (positions and velocities) and gripper angle, totaling 14 features (7 for each tooltip). Later, we increased them to 21 by adding angular velocities.

3.1.4 Labels

Each time step of the captures is associated with a label, which represents an atomic surgical activity segment that is concurrently being executed. These segments are also referred to as **gestures** or **surgemes**. Though there is no standardized method for defining gestures, and they can vary depending on the context, generally gestures are fundamental actions that are performed deliberately and have a meaningful outcome. For example, in the context of robotic surgery, the action of passing the needle from one end effector to the other is a distinguishable, intentional movement that has a specific purpose and can't be broken down any further without getting confused for other movements, thus it can form a gesture.

Label	Gesture Description
G1	Reaching for needle with right hand
G2	Positioning needle
G3	Pushing needle through tissue
G4	Transferring needle from left to right
G5	Moving to center with needle in grip
G6	Pulling suture with left hand
G7	Pulling suture with right hand
G8	Orienting needle
G9	Using right hand to help tighten suture
G10	Loosening more suture
G11	Dropping suture at the end and moving to end points

Table 3.1: Suturing task gestures

JIGSAWS dataset defines a vocabulary of 15 gestures. Table 3.1 contains 10 of those, that are used during the suturing task.

JIGSAWS dataset is **human annotated** with the help of a surgeon. The annotations were reviewed in the paper "Multi-Task Recurrent Neural Network for Surgical Gesture Recognition and Progress Prediction", [28] where 12 mistakes were identified and corrected. Consequently, we will be utilizing these revised annotations in our study.

3.2 Data Preprocessing

Before feeding the kinematic data to a machine learning model, we usually take an extra step that involves transforming the raw data in a format that is suitable according to the model architecture and can be more effectively utilized by it.

A fundamental part of this process is **normalization**. Normalization adjusts the features to a common scale, ensuring that they have comparable magnitudes. Having all features scaled to a common range can help gradient-based optimization algorithms 2.2 like gradient descent converge faster and more effectively. We standardized each trial separately by calculating the **mean** value and the standard deviation of each column. The normalized value is given by subtracting the mean and dividing by the standard deviation:

$$z_i = \frac{x_i - \mu}{\sigma}$$

Another important step when feeding categorical data to an RNN model, like the gesture labels in our case, is **one-hot encoding**. Essentially, after assigning an index to each label, we convert it to a binary vector with the same size as the number of label values.

Finally, since the data are sequential and each trial has a different duration, we cannot feed them directly to the model. Thus, we implement a **sliding window** approach by creating a subrange of fixed width that moves across the dataset, using a fixed step. The way the window moves from the start of the sequence towards the end simulates the way new data would arrive and fill a buffer in real time, and when the buffer is full, older data would be discarded. Since we wouldn't start with a buffer full of data, the first window contains only the first element of the sequence, as its last value, and the rest are filled with a padding value (-1). Similarly, padding would be used at the begging of the window until we reach $seq[window_size]$ element. The padding value will be masked out by the model.

3.3 Model Description

In this section, we describe our approach for online gesture recognition. Since we use sequential kinematic data, we decided to use an RNN, specifically an LSTM network. The LSTM model is the core component of our approach, providing a solid foundation for sequential data processing. Then we tried to further enhance the model's performance, by integrating other methods. One of them is multihead attention, which applies the attention mechanism in parallel across heads, capturing different aspects of the data. The other was the use of Conditional Random Fields (CRFs). The models are implemented using Google's tensorfow library and Keras framework.

3.3.1 Input

After completing all the preprocessing steps we discussed in 3.2, the kinematic data are in a suitable format for the LSTM model. Before starting the training, the dataset must be split in three parts: **training data**, **validation data** and **test data**.

Initially, we select the test data, which must be entirely excluded during the training process to ensure an unbiased evaluation. These are the unseen data on which the model is tested after it is fully trained, to estimate its generalization capability. Since the JIGSAWS dataset is already divided by user, it simplifies the implementation of the leave one out (LOO) evaluation technique, as we can easily use one user's trials as test data.

Next, we divide the remaining data into two parts. The training data, which constitutes the majority at 85%, is used in the actual training process of the network. The validation set, on the other hand, is used to assess the model's performance during training and aids in hyperparameter tuning or implementing early stopping.

The LSTM model expects two input vectors:

- X_Train: containing the kinematic data of shape (windows_count, window_size, number_of_feature)
- Y_Train: containing the corresponding one-hot encoded labels of shape (windows_count, number_of_labels)

3.3.2 Output

In a multi-class classification task, the output is an array with a length equal to the number of classes. This array represents the probability distribution across all classes, with each element indicating the likelihood that the model assigns to each respective class being the correct one. The output layer uses the softmax activation function 2.7 to ensure that all the output probabilities sum to 1.

3.3.3 LSTM architecture

This is the architecture of the core part of our proposed gesture recognition network. It's main components are two LSTM layers. Here is a breakdown of some of it's key characteristics.

- **Masking layer** : Uses masking value of -1. It is important to exclude from the training calculations, the padding values we inserted when sampling the sequence using the sliding window (section 3.2).
- **L2 regularization** : It is a technique commonly used in machine learning in order to prevent over-fitting. L2 regularization, also called ridge regression, adds a penalty term that is equal to the squared sum of coefficients as shown in eq. 3.1. Here we apply kernel regularization, which means that the penalty is applied to each LSTM layer's weights. The purpose of this penalty term is to keep the weights relatively small so it can generalize more accurately, since very big weights tend to make the model fit to the noise in the training data.

$$L_{2\_reg} = \lambda \sum_{i=1}^{n} w_i^2 \tag{3.1}$$

- **Dropout Layer**: During each iteration, a dropout layer will randomly select some neurons and temporarily remove them from the network. This also aims to prevent over-fitting.

We employed Hyperband [16] to optimize crucial hyperparameters for our LSTM model, including the number of units, L2 regularization, and dropout rate. Hyperband is an advanced hyperparameter optimization technique designed to efficiently allocate computational resources. It leverages the principles of random search and successive halving, dynamically distributing resources and

rapidly identifying the most promising configurations. Hyperband initiates by evaluating numerous configurations with minimal resources and progressively increases the resources for the best-performing ones, discarding less effective configurations early on. This approach effectively balances exploration and exploitation, accelerating the discovery of optimal hyperparameters while minimizing computational costs.

3.3.4 Training

As the **loss function** of the network, we chose **Categorical Cross-Entropy**, which is usually combined with Softmax output activation function. This loss function estimates the variation between the predicted probability distribution and ground truth's probability distribution. The loss for each class is calculated separately and then summed to determine the total loss:

$$L = -\sum_{i=1}^{10} t_i log(p_i)$$

where:

- i is an iterator over the classes
- t_i is the ground truth and
- p_i is the probability distribution produced by the softmax

The **optimization algorithm** of the network is a variant of the **Adaptive Moment Estimation (Adam)** algorithm, called **AdamW**. Adam is an adaptive learning rate optimization algorithm based upon stochastic gradient descent (SGD) and Root Mean Square Propagation (RMSProp) [10]. It incorporates momentum as an estimate of the average of the gradients in order to accelerate convergence. The **momentum** term is typically calculated as an exponentially weighted moving average of past gradients. This means that the optimizer gives more weight to recent gradients but still considers the history of past gradients: **RMSpop** keeps a moving average of the squared gradients over time and uses it for adjusting the learning rate for each parameter according to its changing pace, in order to avoid overshooting without slowing down the convergence too much. Adam incorporates elements from both approaches. The key difference between Adam and AdamW, is that the latter decouples weight decay from the gradient update by applying it directly to the parameters instead of the gradients:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \lambda \theta_t$$

where:

- θ_t are the parameters at step t.

- η is the learning rate.
- $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second moment estimates.
- ϵ is a small constant to prevent division by zero.
- λ is the weight decay coefficient.

After examining the labels' distribution in the dataset, we found that it is heavily imbalanced. As it is evident from Fig.3.3 that shows how many times each gesture is executed in the dataset, some gestures, especially G10 (Loosening more suture) are underrepresented. During the train-validation split, we use a technique called **stratification** that ensures the proportions of class labels are the same in both subsets. Furthermore, by assigning a **weight proportional to the inverse of the class distribution** to each class and passing a dictionary of said weights to the training function, we make sure that the underrepresented class is not ignored during the training process. Tensorflow uses these class weights to internally modify the textbfloss function so the loss is multiplied by the weight associated with the example's class. This increases the penalty for misclassifying examples from underrepresented classes.

We calculate each class's weight using the following formula:

$$W_i = \frac{1}{C_i} * \frac{\sum_{i=1}^{10} c_i}{2}$$

where:

- C_i : count of i^{th} label occurrences
- $\sum_{i=1}^{10} c_i$: total timesteps

Another technique we employed to mitigate overfitting is **Early stopping**. It involves monitoring the model's performance on the validation set during training and halting the training process once the performance no longer improves. This helps to ensure that the model does not learn noise or random fluctuations in the training data, which can lead to poor generalization on new, unseen data.

3.3.5 Considered approaches

Building on the foundation provided by the LSTM network, we sought to further enhance our model's performance by integrating additional techniques. These enhancements aim to capture more intricate patterns and dependencies within the sequential data.

Feature engineering : This is the process of creating a new feature by transforming a subset of the available input variables into a more effective set of inputs. As mentioned in section 1.2, Keshara Weerasinghe et. al [31] use the video data to extract a state vector representing the interactions between surgical instruments (e.g., graspers, scissors) and objects (e.g., needle, thread) in the surgical scene. We also tried to generate a state variable using kinematic data only. Specifically, we tried to combine two existing features, the left and right gripper angles, into a categorical context variable with four possible states that can be described as "Joint gripper state":

1. Both closed
2. Left open - Right closed
3. Left closed - Right open
4. Both open

The plot in figure 3.4 showcases the transformation of the two analog signals into a discrete state feature.

Hybrid LSTM - CRF In various domains of machine learning, particularly in natural language processing, we often see efforts to enhance the capabilities of an RNN by combining it with **statistical models**. We have seen related approaches that combine LSTM with Hidden Markov Models (HMM) in [11] and [22] or with other models like CRFs in [32] and Support Vector Machines (SVM) in [19].

After reviewing the transition matrix of the real labels of the dataset and noticing how sparse it is, we tried to integrate in our network, a model that could potentially take advantage of the limited transition probabilities of the dataset. As shown in Fig. 3.5 most pairs of states have a transition probability of zero, meaning transitions between these states are not permitted. Among the few allowed transitions, many have a heavily weighted transition that has significantly higher probability compared to others.

Since the LSTM is designed to capture long term dependencies by maintaining a memory while parsing sequential data we resorted to a combination with a CRF, expecting that it can focus on different aspects of the data, and learn patterns complementary to the LSTM that can utilize the distinctive transition probability distribution.

In our proposed architecture, we place the CRF after the basic 2 LSTM layer part we described in section 3.3.3. The LSTM network's prediction for the previous gesture is concatenated to the original input and fed to the CRF with the intention of the CRF to refine and improve the LSTM's performance.In order to refine the LSTM output without compromising its reliability, we implemented a very simple decision mechanism that employs the CRF only when the LSTM prediction has a high level of uncertainty. We set a max probability threshold (we found 77%

to be effective) for using the LSTM output or falling back to CRF. The whole architecture is illustrated in Fig. 3.6.

The CRF model uses both L1 and L2 regularization, with coefficient set to 0.1 for both. The optimization algorithm used is called Limited-memory BFGS (L-BFGS) which provides an efficient way of minimizing a differentiable scalar function f(x) over a real vector x. It uses an approximation of the inverse Hessian matrix of the objective function to find the search direction. The max iterations of the CRF model are 100.

Hybrid LSTM - Attention One more approach was considered for enhancing the overall performance of the LSTM model, that involved the integration of a MultiHead attention mechanism that can give the model the unique ability to focus on different parts of the data sequence simultaneously and model an estimation of the importance of each part of it.

We experimented with various configurations of the model's building blocks and found that placing the 4-head attention mechanism between the two LSTM layers was the most effective. The input sequence is processed by the first LSTM layer, which captures temporal dependencies and outputs a sequence of hidden states, on which the multihead attention operates intended to capture more complex dependencies. The same sequence is used as query and value vector, resulting in **Self Attention**. The produced output is concatenated with the output of the LSTM, and afterward it is being fed into the second LSTM layer for further processing.

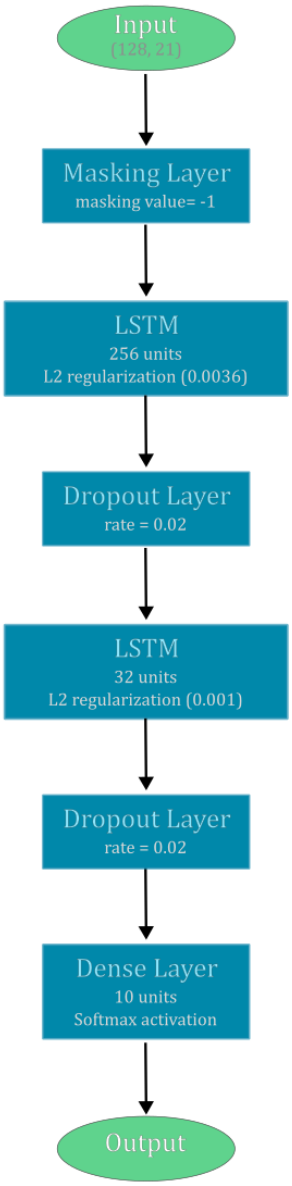

Figure 3.2: The architecture of the gesture recognition network, consisting of two LSTM layers and dropout layers between them.

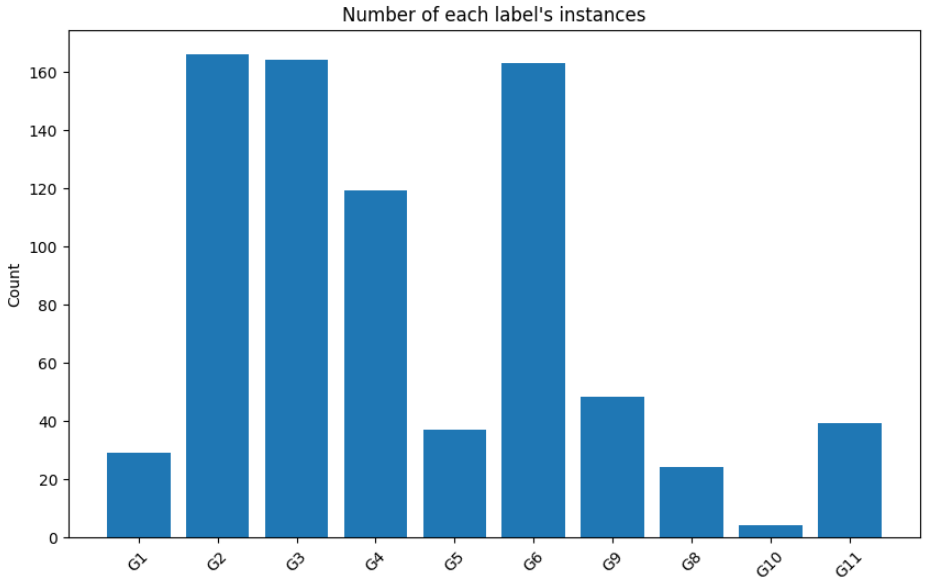

Figure 3.3: Number of instances per label

Figure 3.4: A plot containing the gripper angle signals and the created joint state's transitions throughout a single trial. Label transitions are also included.

Figure 3.5: Transition probability matrix of the ground truth.

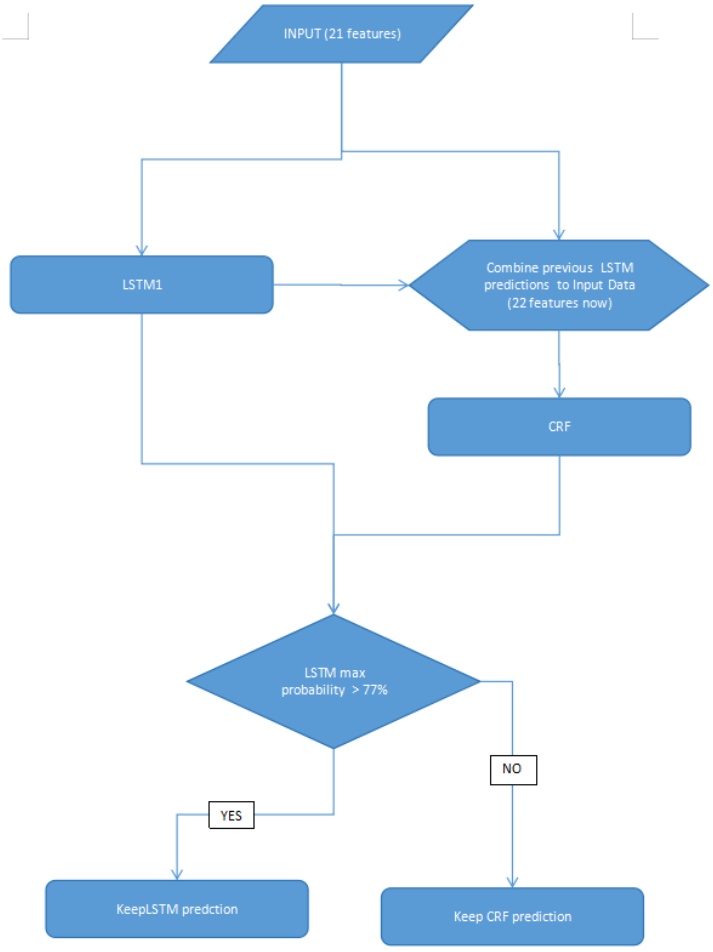

Figure 3.6: Hybrid LSTM - CRF architecture

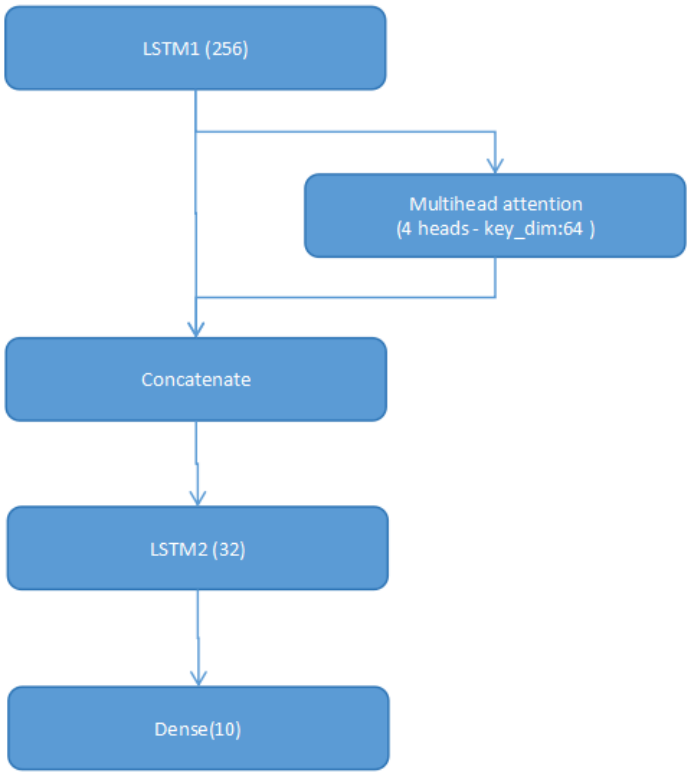

Figure 3.7: LSTM with MultiHead attention layer

Chapter 4

Results

This chapter presents and analyzes the results of our proposed approaches. We compare different model architectures, examine the results in detail, and gain insights into the gesture classification task. Finally, we evaluate our models using standard metrics and compare them to state-of-the-art methods in the field.

4.1 Leave One Out

As we mentioned in 3.3.1 we reserved a portion of the dataset that will remain unseen from the model during the training process and will be used for **cross validation**. Leave-One-Out Cross Validation (LOOCV) is the standard cross validation technique for the JIGSAWS dataset since the trials are already divided by user and there are eight users in total, enough for an unbiased estimation of the model performance and not too many to require multiple training-evaluation cycles. Furthermore, the fact that it is widely used gives us the chance to objectively compare our results to related work. LOOCV is a type of k-fold cross-validation where the number of folds equals the number of users. It requires training the model separately each time with one user's trial left out, which is then used to calculate the desired metrics of the model's. The process is repeated until every user has been left out once, and the final metrics are calculated as the average of all iterations.

4.2 LSTM Parameters

4.2.1 Feature Selection

As we explained in section 3.1, we trained our LSTM network using exclusively kinematic data of the patient side manipulators. The dataset offers various kinematic variables. We started using 14 of them, 7 for each one of the two tool tips:

Features	Accuracy (%)	F1 score
LSTM with 14 features	78.82	0.597
LSTM with 20 features	79.47.5	0.6
LSTM with 20 features + Joint Gripper State	80.58	0.606

Table 4.1: Feature selection ablation study

- linear position (x, y, z)
- linear velocity (x', y', z')
- gripper angle (θ)

Later we added 6 more variables, 3 for each tool tip:

- rotational velocity (α', β', γ')

Finally, we added one more variable, which this time does not come from the dataset raw data, but is one we constructed by combining the two gripper angles, as we explained in detail in chapter 3.3.5. We named this state variable **Joint Gripper State**. In table 4.1 we present an ablation study of the different choices in features to include. The metrics used for comparison are **accuracy** and **F1 score** and are both calculated on the evaluation data, with Leave One Out cross-validation (section 4.1).

We can see that the inclusion of angular velocities and Joint Gripper State played a role in boosting the model's performance. As we can see in the model's summary in Fig. (4.1), it has 965984 parameters.

Layer (type)	Output Shape	Param #	Connected to
input_layer_8 (InputLayer)	(None, 128, 21)	0	-
not_equal_8 (NotEqual)	(None, 128, 21)	0	input_layer_8[0][0]
masking_8 (Masking)	(None, 128, 21)	0	input_layer_8[0][0]
any_24 (Any)	(None, 128)	0	not_equal_8[0][0]
lstm_16 (LSTM)	(None, 128, 256)	284,672	masking_8[0][0], any_24[0][0]
dropout_16 (Dropout)	(None, 128, 256)	0	lstm_16[0][0]
lstm_17 (LSTM)	(None, 32)	36,992	dropout_16[0][0], any_24[0][0]
dropout_17 (Dropout)	(None, 32)	0	lstm_17[0][0]
dense_8 (Dense)	(None, 10)	330	dropout_17[0][0]

Total params: 965,984 (3.68 MB)
Trainable params: 321,994 (1.23 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 643,990 (2.46 MB)

Figure 4.1: Summary of the LSTM model with 21 features

Sliding Window Size	Accuracy (%)	F1 score
3.2 sec	76.7	0.5855
6.4 sec	77.12	0.5891
12.8 sec	79.26	0.588
25.6 sec	75.99	0.5337

Table 4.2: Sliding window size ablation study. Results obtained from LSTM model on JIGSAWS dataset

4.2.2 Sliding window size

Here we present an ablation study about the effect of varying sliding window sized on the model's accuracy and F1 score. The sliding window approach segments the continuous stream of input data into fixed-size chunks for processing. By experimenting with different window sizes, we aimed to identify the optimal value for capturing sufficient contextual information and maintaining real-time performance. As the data in Table 4.2 show, initially, as the window size increases, the model gains more contextual information, leading to more accurate predictions. The added context helps the model better understand the sequence of gestures. But after a certain point this relation changes. For larger values, the window may include too much data, introducing noise and irrelevant information. This excessive information can overwhelm the model, making it harder to isolate the relevant features for each gesture.

4.2.3 Delay

This ablation study is focused on the delay parameter, which represents the time lag allowed between the occurrence of a gesture and its recognition by the model. This parameter is crucial for online applications where timely responses are essential. In Table 4.3 we investigated how different delay values influenced the model's ability to correctly classify gestures in a timely manner. A smaller delay enhances the real-time capabilities of the model but may reduce accuracy due to rushed decision-making. Conversely, a larger delay might improve accuracy by allowing more thorough analysis but at the cost of responsiveness. We chose to permit a delay of up to 1 second to ensure that the model remains viable for real-time applications.

Delay	Accuracy (%)	F1 score
0 sec	74.96	0.5697
0.2 sec	74.5	0.5738
0.5 sec	76.49	0.5867
1 sec	79.26	0.588

Table 4.3: Delay ablation study. Results obtained from LSTM model on JIGSAWS dataset

4.3 LSTM output analysis

4.3.1 Individual trials examination

To gain a deeper understanding of the challenges in online gesture recognition, we conducted a detailed analysis of each trial. We began by plotting the predicted labels and the ground truth for each trial. Additionally, since the output of the LSTM network is generated using a softmax activation function, we also plotted the maximum probability of each output as a measure of the model's confidence in its predictions. Each user's trials were evaluated using the iteration in which they were excluded from the training set.

Upon initial inspection, it becomes evident that the model does not generalize equally well across different users. For instance, examining the best and worst results reveals an average accuracy of 88.6%for user F, while for user D, the average accuracy is constrained to 68.5%. Analyzing each trial for user B reveals a recurring pattern: **a single gesture is getting regularly mislabeled**.This gesture, G9, corresponds to "Using right hand to help tighten the suture." User B frequently utilizes this gesture, whereas it is almost absent in other users' trials. In fact, G9 does not appear even once in any trial of user F.

So we can infer that if the vast majority of a class samples are concentrated in a few trials, the model's ability to generalize and accurately recognize instances from this class is affected. Especially when these trials happen to be in the same evaluation fold, this class is severely underrepresented in the remaining data, leaving too little information for the network to learn patterns and relations regarding this gesture. We also notice a sharp decline in the network's confidence while this gesture is executed. The results for this user are even worse when using fewer features. Specifically, with 14 features, the average accuracy drops to 55.47%.

4.3.2 Accuracy per class

By directly comparing the accuracy per class to the number of instances per class in Fig.(4.4) we can confirm that in most cases there is a strong correlation between the representation of the class and its recognition rate. It is more apparent

Figure 4.2: User out: B to E. On top of each subplot we see the predicted labels, and directly below, the ground truth. The bottom chart shows the confidence of the network for each output. Results obtained from LSTM model on JIGSAWS dataset

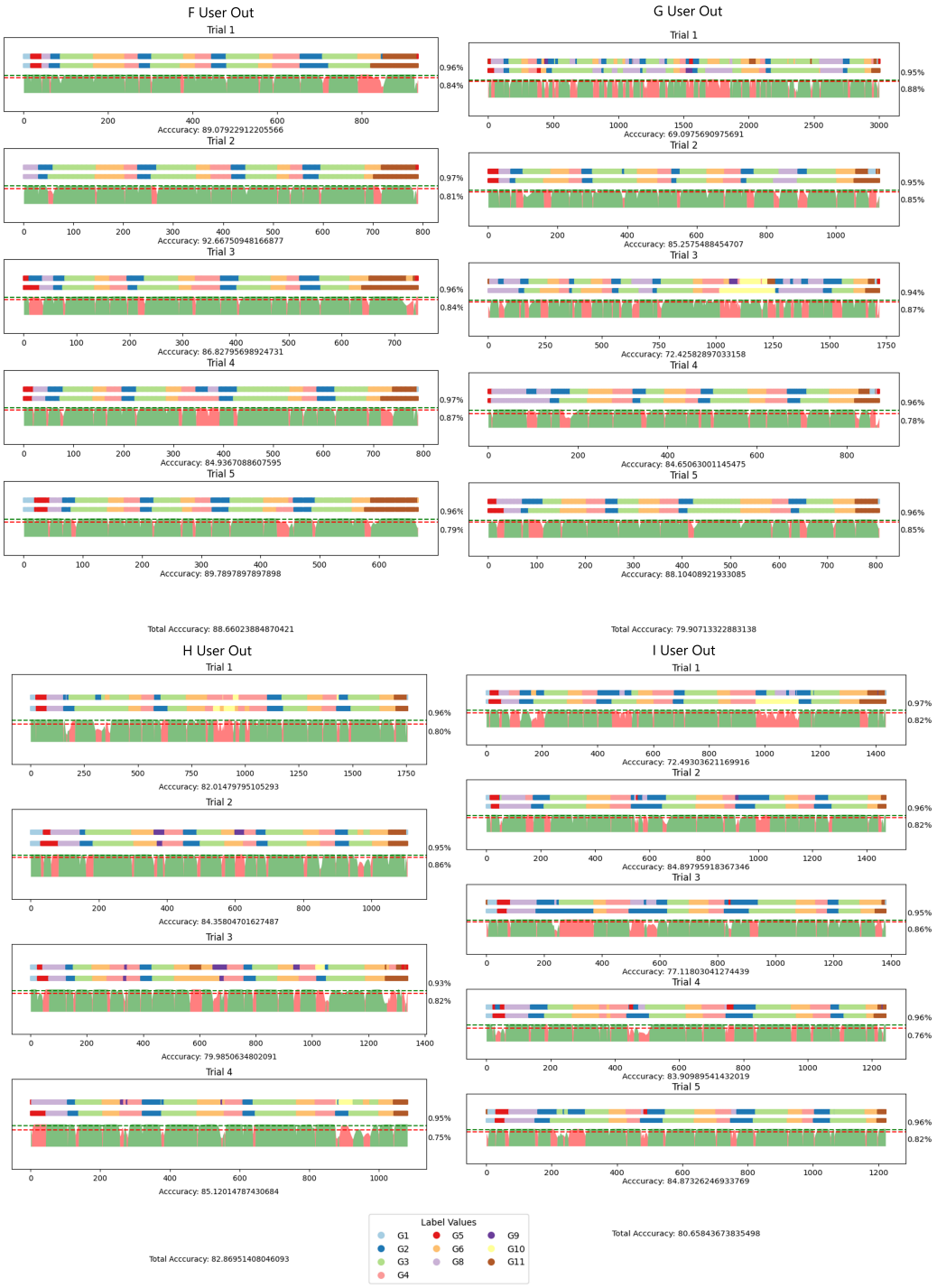

Figure 4.3: User out: F to G. On top of each subplot we see the predicted labels, and directly below, the ground truth. The bottom chart shows the confidence of the network for each output. Results obtained from LSTM model on JIGSAWS dataset

in classes G9 ("Using right hand to help tighten suture"), G10 ("Loosening more suture") and G11 ("Dropping suture at end and moving to end points"). Though there are exceptions like G1 ("Reaching for needle with right hand") that can possibly be explained by the fact that G1 is always the starting gesture, which is characteristic that the network could potentially manage to pick up, or it simply contains some more distinctive and recognizable patterns.

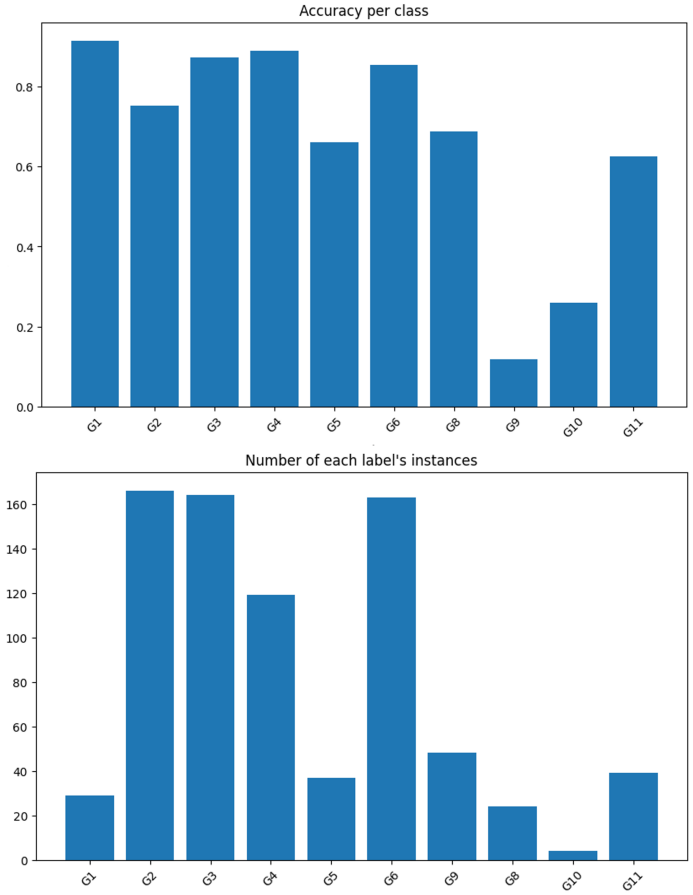

Figure 4.4: Top: Accuracy per label. Bottom: The number of instances per label. Results obtained from LSTM model on JIGSAWS dataset

4.3.3 Confusion matrix

Another plot that provides helpful intuition about the network is the confusion matrix that is presented in Fig. 4.5. Here we don't see any pair of gestures getting regularly confused for each other both ways. The only one where it is noticeable is G2 ("Positioning needle")- G3 ("Pushing needle through tissue") with

G2 mislabeled for G3 at 12.4% percentage and G2 for G3 at 8.4%. This shows that the network, in most cases, can sufficiently detect distinguishable patterns for each gesture. When examining the results for G9, G10, and G11, we observe that the false predictions are **dispersed across nearly all classes**. This indicates that the network misclassifies the minority classes as almost any other class, suggesting it struggles to establish clear decision boundaries, resulting in outputs that appear almost random.

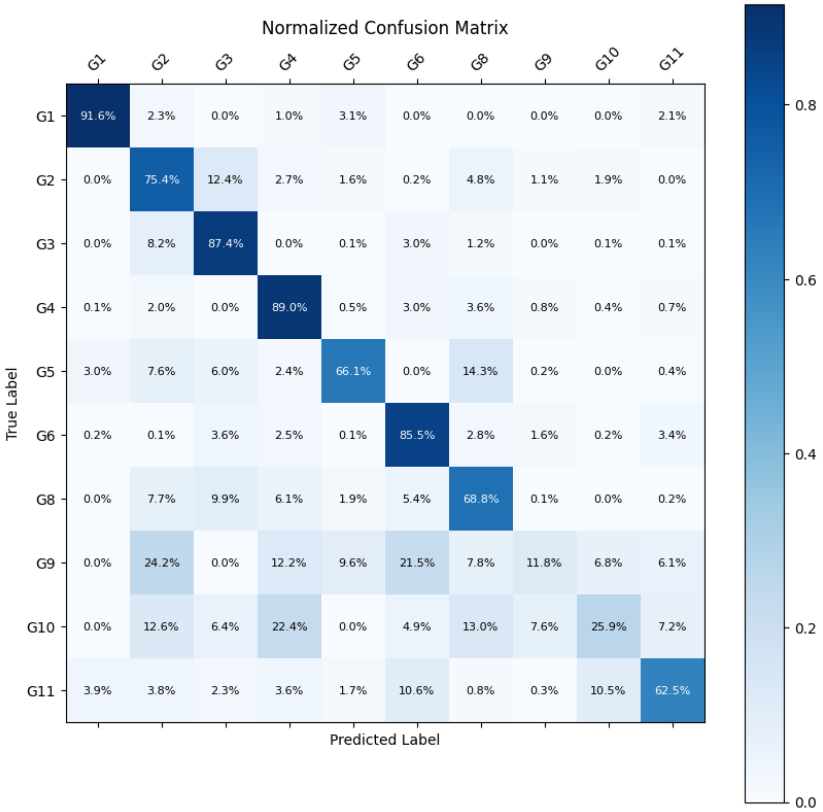

Figure 4.5: Confusion matrix. Results obtained from LSTM model on JIGSAWS dataset

4.4 LSTM with CRF

As we mentioned in 3.3.5 the intention of the choice to combine the LSTM using a CRF model was to take advantage of the concentrated **transition probabilities** that are presented in the model's transition matrix. For most of the classes, there are only two or three possible next states, and in many cases, there is one transition with a very high probability.

6

Figure 4.6: Transition matrix. Results obtained from LSTM model on JIGSAWS dataset

By comparing the transition probabilities of the ground truth ((Fig. 3.5)) to those of our LSTM network (Fig. 4.6) we see that the predicted output contains

a few illegal transitions. We trained a CRF model that takes the LSTM output as input, aiming to correct some of the LSTMs misclassifications by utilizing the transition probabilities.

We tried to find an adequate way to combine the two outputs by using a threshold in LSTM network's confidence in order to decide whether to keep its prediction or use the one by the CRF. Although we weren't able to improve the overall accuracy, we will still examine some outputs of this approach.

Fig. 4.7 contains the output of the hybrid model for user B, in a format similar to Fig.4.2 and 4.3. Both the individual and the combined models' results. We have also highlighted the parts where the CRF output was used and provided a correct or incorrect result. The CRF model, on its own, performs poorly and in some cases like trial 1 it leads to over-segmentation. Nevertheless, in trials 2 and 3 we can see that if it is strategically employed it can slightly improve the output of the LSTM. Perhaps the simple decision mechanism of a universal threshold is not enough, and more sophisticated voting mechanisms can provide better performance.

4.5 LSTM with attention

Our final attempt at enhancing the model included the addition of an attention layer. After experimenting with various layouts, we got the best results by placing a 4 Head attention layer that uses the 1st dropout layer's output as both query and value vector, essentially acting as a self attention layer.

The attention layer managed to improve the generalization ability of the model, even at trials that include minority gestures (user B accuracy increased from 68.52 to 74.58). This architecture brought the best results out of all the approaches we tried with an average accuracy of **81.56%**

It is notable from both the accuracy per class chart and the confusion matrix (Fig.4.9) that this hybrid model provides more balanced and less biased predictions.

4.6 Comparison with state of the art

Numerous approaches have been published in the field of gesture recognition, with many evaluated on the suturing task of the JIGSAWS dataset. However, not all of these approaches meet the criteria established for this thesis, **using only kinematic data** and suggesting an architecture that **can be used online** When comparing only to work that adheres to these limitations, our network has surpassed the state of the art, achieving an accuracy of **81.56%**, as shown in (Table 4.4).

For the sake of completeness, we also present the best results achieved by bidirectional models, which are not suitable for online use (Table 4.6). Bidirectional models, in order to make a prediction for a given time step, need to

Method	Accuracy (%)	year
Skip-Chain CRF [13]	80.29	2015
Forward LSTM [6]	80.5	2016
Our work		
Forward LSTM - 14 Features	**78.82**	2024
Forward LSTM - 20 Features	**79.47**	2024
Forward LSTM - 20 Features + Gripper State	**80.58**	2024
LSTM with self attention	**81.56**	2024

Table 4.4: Results on JIGSAWS with kinematic data that can be used in real time

Method	Accuracy (%)	year
LC-SC-CRF [14]	83.5	2016
$C + V_{Spatial}$ [31]	84.4	2024
Fusion-KV [20]	86.3	2020
$K_{14} + C + V_{Spatial}$ [31]	**87.1**	2024

Table 4.5: Results on JIGSAWS with both kinematic and visual data

have access to the entire sequence, making them impossible to function effectively in real-time application.

Additionally, we include the top results obtained using visual data too (Table 4.5), which encompass the overall state-of-the-art performance on the JIGSAWS dataset, as reported in the paper "Multimodal Transformers for Real-Time Surgical Activity Prediction" [31].

Method	Accuracy (%)	year
Bidir LSTM [6]	83.3	2016
Bidir MIST [5]	84.7	2019
Bidir LSTM [5]	84.7	2019
Bidir GRU [5]	**84.8**	2019

Table 4.6: Results on JIGSAWS with kinematic data that can't be used in real time

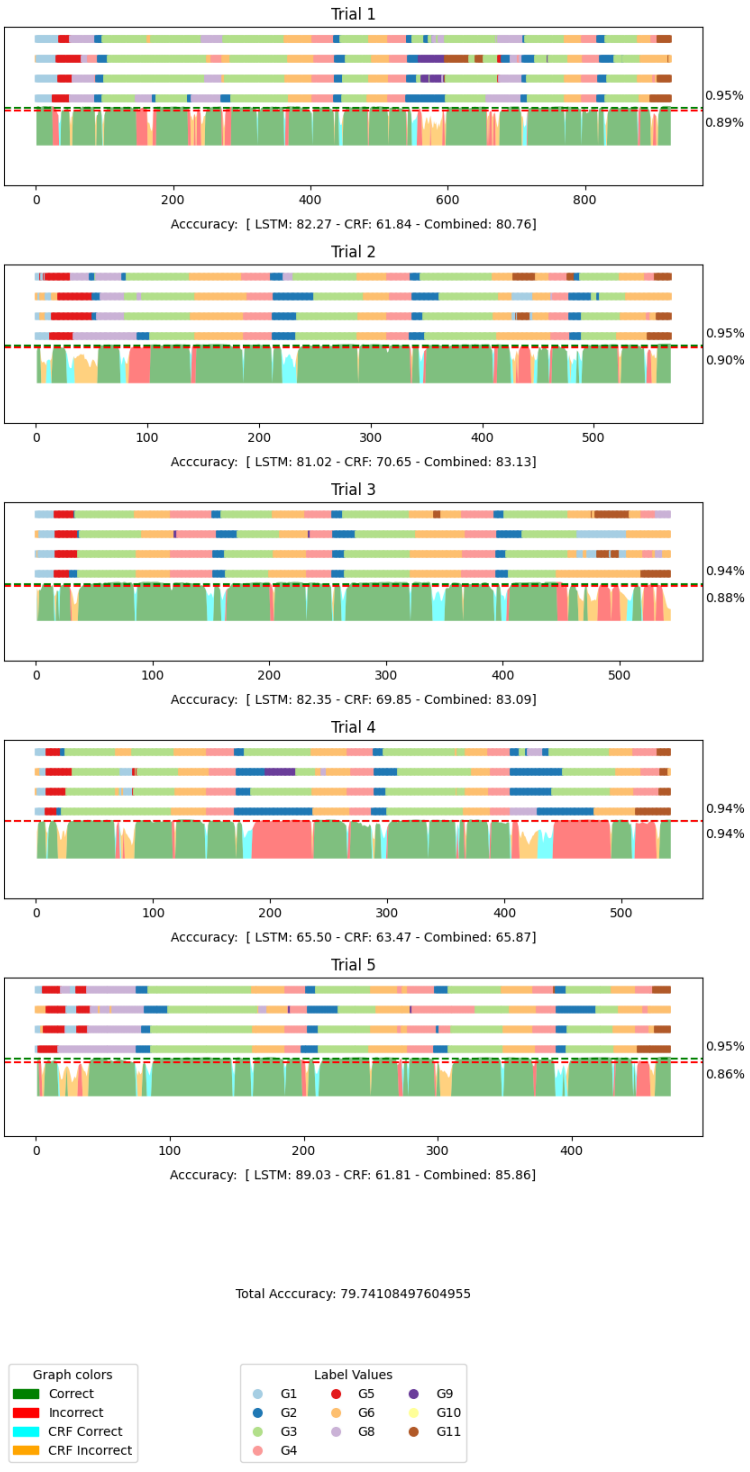

Figure 4.7: Results of user B using the Hybrid LSTM - CRF model. Each subplot contains from top to bottom: LSTM output, CRF output, Hybrid LSMT-CRF, output, ground truth - confidence curve.

```
Model: "model_5"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_6 (InputLayer)        [(None, 128, 21)]         0         []

 masking_5 (Masking)         (None, 128, 21)           0         ['input_6[0][0]']

 lstm_10 (LSTM)              (None, 128, 256)          284672    ['masking_5[0][0]']

 dropout_10 (Dropout)        (None, 128, 256)          0         ['lstm_10[0][0]']

 multi_head_attention_5 (Mu  (None, 128, 256)          263168    ['dropout_10[0][0]',
 ltiHeadAttention)                                               'dropout_10[0][0]']

 concatenate_5 (Concatenate  (None, 128, 512)          0         ['dropout_10[0][0]',
 )                                                               'multi_head_attention_5[0][0]
                                                                 ']

 lstm_11 (LSTM)              (None, 32)                69760     ['concatenate_5[0][0]']

 dropout_11 (Dropout)        (None, 32)                0         ['lstm_11[0][0]']

 dense_5 (Dense)             (None, 10)                330       ['dropout_11[0][0]']

=================================================================================================
Total params: 617930 (2.36 MB)
Trainable params: 617930 (2.36 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________________________________________
```

Figure 4.8: LSTM with self attention model summary

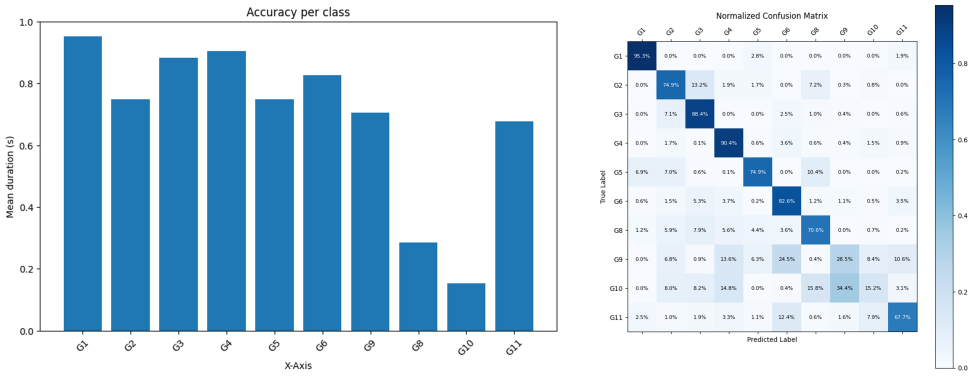

Figure 4.9: Hybrid LSTM - Self Attention on JIGSAWS. Left: Accuracy per class chart. Right: Confusion matrix

Chapter 5

Conclusion and Future work

Our objective was to train a machine learning model to recognize gestures during surgical procedures in real-time. Evaluation was done using the JIGSAWS dataset, specifically with the suturing tasks. Our goal was to achieve state-of-the-art performance under the following conditions: the model to be used must be able to operate in real-time through a sliding window with a maximum delay of 1 second and be trained only with kinematic data. The basic neural network used was the LSTM. Initially, we experimented with using one and two LSTM layers and then attempted to improve the model's performance with various trivial techniques such as hyperparameter tuning, early stopping, introduction of dropout layers, L1 and L2 regularization, validation split. At each step, we tried to visualize the results with various graphs, confusion matrix, transition matrix, etc. to evaluate the areas that needed improvement. After observing a weakness in recognizing classes with the smallest representation in the dataset, we applied stratification to the training-validation data split, as well as a larger penalty for misclassification of these classes. Subsequently, we tried to increase the number of features by adding angle velocities and performing feature engineering by creating a new feature with 4 discrete states based on the values of the gripper angles (left close-right open, both open, etc.). Finally, to further improve performance, we tried 2 hybrid approaches of the LSTM model. In the first, we compared the results of introducing a multihead attention layer at different points in the architecture. In the second, we tried to exploit the sparse transition matrix of the data by using a CRF which receives as input the LSTM predictions in combination with part of the kinematic data. We managed to improve the state-of-the-art performance given the two constraints we set with an accuracy of 81.56% with a hybrid LSTM - Self Attention model.

Based on the work that was presented in this thesis, many suggestions for future work can be made. Although we explored the idea of combining LSTM with CRF there is definitely room for improvement, in the way the two model outputs are merged. A potential solution could be a more complex voting mechanism with separate class weights, based on the success rate for each class prediction by each model. This idea could be expanded in other hybrid models that might further

enhance the LSTM abilities, such as Transformers. Our model based solely on kinematic data, but future research could build upon it by utilizing visual data too, which opens up a wide variety of machine learning models to try and evaluate. Additionally, the performance of the examined models can be assessed in more publicly available surgical datasets. A detailed list can be found in [18].

Lastly, the gesture recognition network could be a part of a bigger system that aims to monitor the procedure and even assist the surgeon in executing the desired task. Such a system could contain a library of movement primitives for each gesture, and after recognizing the currently executed gesture, help the operator make safe, efficient and precise movements by providing haptic feedback.

Bibliography

[1] Edoardo M Airoldi. Getting started in probabilistic graphical models. *PLoS Computational Biology*, 3(12):e252, December 2007.

[2] Oswald Campesato. *Artificial Intelligence, Machine Learning, and Deep Learning*. Mercury Learning and Information, 2020.

[3] Yanmei Chen, Zeyu Ding, Yen-Lun Chen, and Xinyu Wu. Rapid recognition of dynamic hand gestures using leap motion. In *2015 IEEE International Conference on Information and Automation*, pages 1419–1424, 2015.

[4] Sho Yaida Daniel A. Roberts. *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks*.

[5] Robert DiPietro, Narges Ahmidi, Anand Malpani, Madeleine Waldram, Gyusung Lee, Mija Lee, Sunil Vedula, and Gregory Hager. Segmenting and classifying activities in robot-assisted surgery with recurrent neural networks. *International Journal of Computer Assisted Radiology and Surgery*, 14:1–16, 04 2019.

[6] Robert DiPietro, Colin Lea, Anand Malpani, Narges Ahmidi, S. Swaroop Vedula, Gyusung I. Lee, Mija R. Lee, and Gregory D. Hager. Recognizing surgical activities with recurrent neural networks, 2016.

[7] D. A. S. Fraser and N. Reid. [what is a statistical model?]: Discussion. *The Annals of Statistics*, 30(5):1283–1286, 2002.

[8] Yixin Gao, S. Swaroop Vedula, Carol E. Reiley, Narges Ahmidi, Balakrishnan Varadarajan, Henry C. Lin, Lingling Tao, Luca Zappella, Benjamín Béjar, David D. Yuh, Chi Chiung Grace Chen, René Vidal, Sanjeev Khudanpur, and Gregory Hager. Jhu-isi gesture and skill assessment working set (jigsaws) : A surgical activity dataset for human motion modeling. 2014.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[10] Bengio Yoshua; Courville Aaron; Goodfellow Ian J. *Deep learning: adaptive computation and machine learning*. Adaptive Computation and Machine Learning series. The MIT Press, 2016.

[11] Viktoriya Krakovna and Finale Doshi-Velez. Increasing the interpretability of recurrent neural networks using hidden markov models, 2016.

[12] John Lafferty, Andrew Mccallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289, 01 2001.

[13] Colin Lea, Gregory D. Hager, and Rene Vidal. An improved model for segmentation and recognition of fine-grained activities with application to surgical training tasks. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 1123–1129, 2015.

[14] Colin Lea, René Vidal, and Gregory D. Hager. Learning convolutional action primitives for fine-grained action recognition. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1642–1649, 2016.

[15] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[16] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization, 2018.

[17] Henry Lin, Izhak Shafran, David Yuh, and Gregory Hager. Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions. *Computer aided surgery : official journal of the International Society for Computer Aided Surgery*, 11:220–30, 10 2006.

[18] Lena Maier-Hein, Matthias Eisenmann, Duygu Sarikaya, Keno März, Toby Collins, Anand Malpani, Johannes Fallert, Hubertus Feussner, Stamatia Giannarou, Pietro Mascagni, Hirenkumar Nakawala, Adrian Park, Carla Pugh, Danail Stoyanov, Swaroop S. Vedula, Kevin Cleary, Gabor Fichtinger, Germain Forestier, Bernard Gibaud, Teodor Grantcharov, Makoto Hashizume, Doreen Heckmann-Nötzel, Hannes G. Kenngott, Ron Kikinis, Lars Mündermann, Nassir Navab, Sinan Onogur, Tobias Roß, Raphael Sznitman, Russell H. Taylor, Minu D. Tizabi, Martin Wagner, Gregory D. Hager, Thomas Neumuth, Nicolas Padoy, Justin Collins, Ines Gockel, Jan Goedeke, Daniel A. Hashimoto, Luc Joyeux, Kyle Lam, Daniel R. Leff, Amin Madani, Hani J. Marcus, Ozanan Meireles, Alexander Seitel, Dogu Teber, Frank Ückert, Beat P. Müller-Stich, Pierre Jannin, and Stefanie Speidel. Surgical data science – from concepts toward clinical translation. *Medical Image Analysis*, 76:102306, 2022.

[19] Farrokh Mehryary, Jari Björne, Tapio Salakoski, and Filip Ginter. Combining support vector machines and lstm networks for chemical-protein relation extraction. 2018.

[20] Yidan Qin, Sahba Aghajani Pedram, Seyedshams Feyzabadi, Max Allan, A. Jonathan McLeod, Joel W. Burdick, and Mahdi Azizian. Temporal segmentation of surgical sub-tasks through deep learning with multiple data sources, 2020.

[21] Jacob Rosen, Blake Hannaford, Christina G. Richards, and Mika N. Sinanan. Markov modeling of minimally invasive surgery based on tool/tissue interaction and force/torque signatures for evaluating surgical skills. *IEEE Transactions on Biomedical Engineering*, 48:579–591, 2001.

[22] Agnimitra Sengupta, Adway Das, and S. Ilgin Guler. Hybrid hidden markov lstm for short-term traffic flow prediction, 2023.

[23] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, March 2020.

[24] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding lstm - a tutorial into long short-term memory recurrent neural networks. *ArXiv*, abs/1909.09586, 2019.

[25] Charles Sutton and Andrew McCallum. An introduction to conditional random fields, 2010.

[26] Lingling Tao, Ehsan Elhamifar, Sanjeev Khudanpur, and René Hager, Gregory D.and Vidal. Sparse hidden markov models for surgical gesture classification and skill evaluation. In Purang Abolmaesumi, editor, *Information Processing in Computer-Assisted Interventions*, pages 167–177, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[27] Lingling Tao, Luca Zappella, Gregory D. Hager, and René Vidal. Surgical gesture segmentation and recognition. In Kensaku Mori, Ichiro Sakuma, Yoshinobu Sato, Christian Barillot, and Nassir Navab, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*, pages 339–346, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[28] Beatrice van Amsterdam, Matthew J. Clarkson, and Danail Stoyanov. Multi-task recurrent neural network for surgical gesture recognition and progress prediction, 2020.

[29] Balakrishnan Varadarajan, Carol Reiley, Henry Lin, Sanjeev Khudanpur, and Gregory Hager. Data-derived models for segmentation with application to surgical assessment and training. In Guang-Zhong Yang, editor, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009*, pages 426–434, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[31] Keshara Weerasinghe, Seyed Hamid Reza Roodabeh, Kay Hutchinson, and Homa Alemzadeh. Multimodal transformers for real-time surgical activity prediction, 2024.

[32] Renzo M. Rivera Zavala, Paloma Martínez, and Isabel Segura-Bedmar. A hybrid bi-lstm-crf model for knowledge recognition from ehealth documents. In *TASS@SEPLN*, 2018.