NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF INDUSTRIAL ELECTRIC DEVICES AND DECISION SYSTEMS

MMSleepNet: A Multimodal Model for Automatic Sleep Staging

DIPLOMA THESIS

Dimitrios K. Xynogalas

Supervisor: Dimitrios Askounis
Professor NTUA

Athens, February 2024

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF INDUSTRIAL ELECTRIC DEVICES AND DECISION SYSTEMS

MMSleepNet: A Multimodal Model for Automatic Sleep Staging

DIPLOMA THESIS

Dimitrios K. Xynogalas

Supervisor: Dimitrios Askounis
Professor NTUA

Approved by the examination committee on 29 February 2024

_____	_____	_____
Ioannis Psarras	Dimitrios Askounis	Vangelis Marinakis
Professor NTUA	Professor NTUA	Assistant Professor NTUA

Athens, February 2024

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF INDUSTRIAL ELECTRIC DEVICES AND DECISION SYSTEMS

Dimitrios Xynogalas
Electrical and Computer Engineer, NTUA

Περίληψη

Η αυτόματη ταξινόμηση των σταδίων ύπνου είναι ζωτικής σημασίας για την αξιολόγηση της ποιότητας του ύπνου και τη διάγνωση των διαταραχών του ύπνου. Αν και έχουν αναπτυχθεί πολυάριθμες προσεγγίσεις για τον σκοπό αυτό, πολλές βασίζονται αποκλειστικά σε μονοκάναλα σήματα ηλεκτροεγκεφαλογραφήματος (EEG) ή εξετάζουν εισροές από ένα μόνο πεδίο. Η πολυσωμνογραφία (PSG) προσφέρει μια πιο ολοκληρωμένη λύση παρέχοντας πολλαπλά κανάλια καταγραφής σήματος, επιτρέποντας στα μοντέλα να εξάγουν και να ενσωματώνουν πληροφορίες από διάφορα κανάλια για βελτιωμένη απόδοση ταξινόμησης. Επιπλέον, πολύτιμες πληροφορίες μπορούν να αντληθούν τόσο από τις αναπαραστάσεις του πεδίου του χρόνου όσο και από τις αναπαραστάσεις στο πεδίο της συχνότητας των σημάτων PSG. Η παρούσα μελέτη εισάγει το MMSleepNet, ένα μοντέλο βαθιάς μάθησης που εξάγει και συγχωνεύει επιδέξια πληροφορίες από πολλαπλά κανάλια PSG τόσο στο πεδίο του χρόνου όσο και στο πεδίο της συχνότητας για να επιτύχει την αυτόματη κατηγοριοποίηση των σταδίων του ύπνου. Αναγνωρίζοντας τη δυνατότητα των εργαστηρίων ύπνου να παράγουν τεράστιες ποσότητες μη επισημασμένων δεδομένων για τη κατηγοριοποίηση του ύπνου και την εγγενή δυσκολία της επισήμανσης αυτών των δεδομένων, η μελέτη διερευνά την αποτελεσματικότητα διαφόρων αλγορίθμων αυτοεπιβλεπόμενης μάθησης για την αντιμετώπιση της έλλειψης ετικετών και τη βελτίωση της απόδοσης του προτεινόμενου μοντέλου.

Λέξεις κλειδιά: Μηχανική Μάθηση, Πολυτροπική, Αυτόματη Κατηγοριοποίηση Ύπνου

Abstract

Automatic sleep stage classification is crucial for assessing sleep quality and diagnosing sleep disorders. While numerous approaches have been developed for this purpose, many rely solely on single-channel electroencephalogram (EEG) signals or consider inputs from a single domain. Polysomnography (PSG) offers a more comprehensive solution by providing multiple channels of signal recording, enabling models to extract and integrate information from diverse channels for enhanced classification performance. Furthermore, valuable insights can be derived from both the time and frequency domain representations of PSG signals. This study introduces MMSleepNet, a deep learning model that adeptly extracts and fuses information from multiple PSG channels across both the time and frequency domains to facilitate automatic sleep stage scoring. Recognizing the potential of sleep labs to generate vast amounts of unlabeled data for sleep scoring and the inherent difficulty of labeling this data, the study explores the efficacy of various self-supervised learning algorithms to address label scarcity and enhance the proposed model's performance.

Keywords: Machine Learning, Multimodal, Automatic Sleep Staging

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου και την εκτίμησή μου προς τον Καθηγητή Δημήτρη Ασκούνη για την ευκαιρία που μου παρείχε να ασχοληθώ με το πρόβλημα ανίχνευσης σταδίων του ύπνου στο εργαστήριό του. Επίσης, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Λουκά Ηλία, ο οποίος με καθοδήγησε και με υποστήριξε κατά την διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την υποστήριξη τους καθ' όλη την διάρκεια των σπουδών μου.

Δημήτρης Ξυνογαλάς,
Αθήνα, Ιανουάριος 2024

Contents

Chapter 0. Εκτεταμένη Περίληψη στα Ελληνικά

0.1 Εισαγωγή

Ο ύπνος έχει κρίσιμο ρόλο για την υγεία και ευεξία του κάθε ανθρώπου, με τις διάφορες διαταραχές του ύπνου να συνδέονται συχνά με πολλές ασθένειες. Η αναγνώριση των σταδίων του ύπνου είναι σημαντική για την κατανόηση των προτύπων ύπνου, τη διάγνωση των διαταραχών και την βελτίωση της περίθαλψής τους. Κλινικά, η πολυυπνογραφία (PSG) είναι μια σημαντική τεχνική για την ανάλυση των σταδίων του ύπνου και τη διάγνωση διαταραχών του ύπνου. Το σύστημα PSG μετράει συνήθως την εγκεφαλική δραστηριότητα (ηλεκτροεγκεφαλογράφημα, EEG), τις οφθαλμικές κινήσεις (ηλεκτροοφθαλμογράφημα, EOG), τη μυϊκή δραστηριότητα ή την ενεργοποίηση των σκελετικών μυών (ηλεκτρομυογράφημα, EMG) και τον καρδιακό ρυθμό (ηλεκτροκαρδιογράφημα, ECG). Συνήθως, εμπειρογνώμονες χειροκίνητα ταξινομούν αυτά τα σήματα σε έξι στάδια ύπνου ακολουθώντας διάφορα εγχειρίδια, αλλά αυτές οι μέθοδοι είναι χρονοβόρες και υποκειμενικές.

Πολλές μελέτες έχουν χρησιμοποιήσει τεχνικές μηχανικής μάθησης για την αυτόματη ταξινόμηση των σταδίων του ύπνου, κυρίως με παραδοσιακές μεθόδους επεξεργασίας σημάτων ή χρησιμοποιώντας μοντέλα βαθιάς μηχανικής μάθησης. Οι παραδοσιακές μέθοδοι απαιτούν την επιλογή των χαρακτηριστικών που πρέπει να εξαχθούν από τα σήματα της πολυυπνογραφίας, ενώ οι μέθοδοι βαθιάς μάθησης αναγνωρίζουν αυτόματα τα χρησιμότερα χαρακτηριστικά. Ωστόσο, η επισήμανση των δεδομένων που απαιτείτε να πραγματοποιηθεί πρώτα από ειδικούς πριν μπορούν να χρησιμοποιηθούν οι παραπάνω τεχνικές είναι κοστοβόρα και περιορίζει τη δημιουργία μεγάλων συνόλων δεδομένων που θα μπορούσαν να αξιοποιηθούν για να βελτιώσουν την απόδοση των εν λόγω μοντέλων.

Επιπλέον, πολλές πρόσφατες μελέτες χρησιμοποιούν πολύπλοκα μοντέλα "ακολουθίας προς ακολουθία" για την ανάλυση δεδομένων και την ταξινόμηση των σταδίων του ύπνου. Αυτή η προσέγγιση αποτυπώνει τις χρονικές σχέσεις μεταξύ των εποχών ύπνου, αλλά απαιτεί μεγάλα μοντέλα και χρόνους εκπαίδευσης. Το AttnSleep [1], που χρησιμοποιεί ένα μοντέλο μιας εποχής, επιτυγχάνει καλά αποτελέσματα με ταχύτερη εκπαίδευση, υποδηλώνοντας τις δυνατότητες απλούστερων, αρχιτεκτονικών ένα προς ένα στην ανάλυση του ύπνου.

Η παρούσα διατριβή παρουσιάζει ένα νέο μοντέλο μηχανικής μάθησης, το MMSleepNet, το οποίο ενσωματώνει πληροφορίες από πολλαπλά φυσιολογικά σήματα που έχουν εξαχθεί με χρήση της τεχνικής PSG για την βελτίωση της ταξινόμησης των σταδίων του ύπνου. Το μοντέλο δοκιμάστηκε σε δεδομένα με πλήρη επισήμανση για να μπορέσει να γίνει σύγκριση με άλλες μεθόδους που χρησιμοποιούνται στην βιβλιογραφία για κατηγοριοποίηση των σταδίων του ύπνου, αλλά και σε δεδομένα με μειωμένη επισήμανση, όπου αφαιρέθηκαν οι επισημάνσεις από ένα μεγάλο ποσοστό των δειγμάτων του συνόλου εκπαίδευσης, έτσι ώστε να συγκριθεί η απόδοση του μοντέλου σε αυτό το σενάριο όταν εκπαιδεύεται εξαρχής με αυτά τα δεδομένα έναντι στο όταν αξιοποιούνται πρώτα αλγόριθμοι αυτο-εποπτευόμενης μάθησης για την προεκπαίδευση του μοντέλου.

Οι ερευνητές στην βιβλιογραφία αναγνωρίζουν τη σημασία της αυτόματης κατηγοριοποίηση του ύπνου με τη χρήση μηχανικής μάθησης και δίνουν έμφαση στη χρήση πολυτροπικών και πολυκαναλικών μοντέλων. Η μετάβαση από τις παραδοσιακές μεθόδους

στις μεθόδους βαθιάς μάθησης προσφέρει τεράστιες δυνατότητες για τον τομέα και έχει παρουσιάσει σημαντική βελτίωση στην ακρίβεια και ευρωστία των προβλέψεων. Ωστόσο, υπάρχουν ακόμα προκλήσεις που πρέπει να αντιμετωπίσουν, όπως η δυσκολία στην απόκτηση δεδομένων με επισήμανση από ειδικούς και η αποτελεσματική εξαγωγή και μίξη πληροφορίας που προέρχεται από πολλαπλά σήματα. Μελλοντικές ερευνητικές εργασίες μπορούν να επικεντρωθούν σε αυτές τις προκλήσεις για να βελτιώσουν περαιτέρω την αυτόματη κατηγοριοποίηση των σταδίων του ύπνου.

0.2 Θεωρητικό Υπόβαθρο

0.2.1 Στάδια Ύπνου

Η αυτόματη ταξινόμηση σταδίων ύπνου (ASSC) είναι ένας σημαντικός τομέας της βιοϊατρικής μηχανικής που αποσκοπεί στην αυτόματη αναγνώριση των σταδίων του ύπνου μέσω της ανάλυσης καταγραφών πολυυπνογραφίας (PSG). Η PSG περιλαμβάνει την καταγραφή πολλαπλών φυσιολογικών σημάτων, όπως ηλεκτροεγκεφαλογράφημα (EEG), ηλεκτροοφθαλμογράφημα (EOG), ηλεκτρομυογράφημα (EMG) και αναπνευστικές παράμετροι. Το ASSC έχει τη δυνατότητα να βελτιώσει την κλινική πρακτική και την έρευνα, μειώνοντας το χειροκίνητο φόρτο εργασίας και επιτρέποντας την παρακολούθηση του ύπνου σε πραγματικό χρόνο.

Ο ανθρώπινος ύπνος περιλαμβάνει πέντε διαφορετικά στάδια, τα οποία είναι: εγρήγορση, ύπνος NREM και ύπνος REM. Ο ύπνος NREM αποτελείται από τέσσερα στάδια (N1, N2, N3, N4), και οι φυσιολογικές παράμετροι και τα μοτίβα EEG διαφέρουν για κάθε στάδιο. Ο ύπνος NREM αποτελεί το 75% του συνολικού ύπνου, με το στάδιο N2 να είναι το πιο κοινό. Μία τυπική περίοδος ύπνου περιλαμβάνει 4 έως 5 κύκλους, με τη σειρά N1, N2, N3, N2, REM. Κάθε περίοδος REM γίνεται μεγαλύτερη καθώς προχωρά η νύχτα, ενώ ο βαθύς ύπνος (NREM) μειώνεται. Ο πλήρης κύκλος ύπνου διαρκεί περίπου 90 έως 110 λεπτά.

0.2.2 Μηχανική Μάθηση

Η μηχανική μάθηση είναι ένα πεδίο έρευνας που ασχολείται με την ανάπτυξη μεθόδων που μπορούν να "μάθουν" από δεδομένα προκειμένου να βελτιώσουν την απόδοσή τους σε διάφορες εργασίες. Οι αλγόριθμοι μηχανικής μάθησης χρησιμοποιούν δεδομένα εκπαίδευσης για να δημιουργήσουν μαθηματικά μοντέλα και να κάνουν προβλέψεις ή να πάρουν αποφάσεις. Η μηχανική μάθηση θεωρείται υποσύνολο της τεχνητής νοημοσύνης, μιας επιστήμης που ασχολείται με την ανάπτυξη και την μελέτη έξυπνων μηχανών.

Τα νευρωνικά δίκτυα αποτελούν τον πιο δημοφιλή κλάδο μοντέλων μηχανικής μάθησης. Τα νευρωνικά δίκτυα αποτελούνται από συνδεδεμένες μονάδες, γνωστές ως τεχνητοί νευρώνες, που μοντελοποιούν τους αληθινούς νευρώνες ενός εγκεφάλου. Οι νευρώνες λαμβάνουν σήματα, τα επεξεργάζονται και μεταδίδουν σήματα σε άλλους νευρώνες μέσω των συνδέσεών τους. Η έξοδος κάθε νευρώνα υπολογίζεται από μια μη γραμμική συνάρτηση του αθροίσματος των εισόδων του. Οι νευρώνες έχουν βάρη που προσαρμόζονται κατά τη διάρκεια της μάθησης, επηρεάζοντας τη δύναμη των σημάτων στις συνδέσεις. Οι νευρώνες μπορεί επίσης να έχουν ένα κατώφλι, καθορίζοντας ότι θα παράξουν σήμα μόνο όταν το συνολικό σήμα στην είσοδο υπερβαίνει αυτό το κατώφλι.

Οι αλγόριθμοι μηχανικής μάθησης χρησιμοποιούνται σε πολλούς τομείς, όπως η όραση υπολογιστών, η αναγνώριση ομιλίας και το φιλτράρισμα ηλεκτρονικού ταχυδρομείου. Η υπολογιστική μοντελοποίηση αυτών των εργασιών είναι πολύπλοκη και η ανάπτυξη

συμβατικών αλγορίθμων είναι συχνά δύσκολη ή αδύνατη. Η μηχανική μάθηση παρέχει μια αποτελεσματική προσέγγιση για την ανάπτυξη αλγορίθμων για την επίλυση αυτών των προβλημάτων.

Υπάρχουν τρεις κύριες πρακτικές εκπαίδευσης στη μηχανική μάθηση: η μάθηση με επίβλεψη, η μάθηση χωρίς επίβλεψη (ή μάθηση με αυτοεπίβλεψη) και η μάθηση με ενίσχυση. Η μάθηση με επίβλεψη απαιτεί τη χρήση ετικετών για την εκπαίδευση των μοντέλων, η μάθηση χωρίς επίβλεψη επιτρέπει στα μοντέλα να ανακαλύπτουν τη δομή των δεδομένων χωρίς οδηγίες, και η μάθηση με ενίσχυση ανταμείβει τα μοντέλα για τις σωστές ενέργειες και τιμωρεί τις λανθασμένες. Υπάρχει επίσης η ημιεπιτηρούμενη μάθηση, που συνδυάζει στοιχεία από την επιβλεπόμενη και την ανεπίβλεπτη μάθηση.

Η μάθηση με αυτοεπίβλεψη είναι μια μορφή μηχανικής μάθησης όπου το μοντέλο μαθαίνει από τα δεδομένα χωρίς επίβλεψη από ανθρώπους ή την χρήση ετικετών. Αντί να βασίζεται σε επισημειωμένα παραδείγματα, το μοντέλο παράγει τα δικά του σήματα εποπτείας από τα δεδομένα εισόδου. Συχνά, η μάθηση με αυτοεπίβλεψη στην βαθιά μάθηση περιλαμβάνει το σχεδιασμό εργασιών που αξιοποιούν τις δομές και τις σχέσεις μεταξύ των δεδομένων. Μέσω αυτών των εργασιών, το μοντέλο μαθαίνει αναπαραστάσεις που μπορούν να χρησιμοποιηθούν σε άλλες εργασίες. Η αυτοεπιβλεπόμενη μάθηση είναι δημοφιλής γιατί μπορεί να αξιοποιεί μεγάλες ποσότητες δεδομένων χωρίς ετικέτες.

0.2.3 Βαθιά Νευρωνικά Δίκτυα

Ο όρος Τεχνητό Νευρωνικό Δίκτυο (ANN) αναφέρεται σε ένα υπολογιστικό σύστημα επεξεργασίας δεδομένων, το οποίο πρόκειται στην ουσία για ένα δίκτυο διασυνδεδεμένων δομικών μονάδων που ονομάζονται νευρώνες οι οποίοι είναι τοπολογικά οργανωμένοι σε επίπεδα. Κάθε τεχνητός νευρώνας λαμβάνει ένα σύνολο αριθμητικών εισόδων από άλλους κόμβους του δικτύου, τις οποίες μετασχηματίζει βάσει ενός γραμμικού συνδυασμού με ανάλογα βάρη και παράγει την τελική έξοδο ύστερα από την εφαρμογή μιας μη-γραμμικής συνάρτησης ενεργοποίησης, η οποία τροφοδοτείται στην συνέχεια σε άλλους νευρώνες του δικτύου. Το βασικό χαρακτηριστικό των τεχνητών νευρωνικών δικτύων έγκειται στην δυνατότητα εκπαίδευσής τους μέσα από μια διαδικασία η οποία στοχεύει στην σταδιακή βελτίωση της ικανότητάς τους να επιλύουν κάποιο συγκεκριμένο πρόβλημα μέσο της αξιοποίησης μιας συλλογής δεδομένων που χρησιμοποιούνται για την εκπαίδευση του δικτύου. Σε επίπεδο υλοποίησης, αυτό επιτυγχάνεται μέσω ενός αλγορίθμου ακολουθιακού υπολογισμού των μεταβολών των βαρών κάθε νευρώνα του δικτύου, ο οποίος καλείται Backpropagation.

Τα Συνελικτικά Νευρωνικά Δίκτυα (CNNs) αποτελούν μία κατηγορία τεχνητών νευρωνικών δικτύων ειδικά σχεδιασμένων για την επεξεργασία και την ανάλυση δεδομένων που χαρακτηρίζονται από κάποιου είδους χωρική τοπολογία πλέγματος και αναπαρίστανται συνήθως με την μορφή γενικευμένων πινάκων. Παραδείγματα περιλαμβάνουν δεδομένα χρονοσειρών, τα οποία μπορούν να θεωρηθούν ως ένα πλέγμα μίας διάστασης που λαμβάνει δείγματα σε τακτά χρονικά διαστήματα, και δεδομένα εικόνας, τα οποία μπορούν να θεωρηθούν ως ένα πλέγμα δισδιάστατων εικονοστοιχείων. Η κύρια διαφορά μεταξύ των CNNs και ενός απλού νευρωνικού δικτύου είναι ότι, σε τουλάχιστον ένα στρώμα ενός CNN ο συμβατικός πολλαπλασιασμός πινάκων αντικαθίσταται από ένα στρώμα συνέλιξης.

Τα επαναλαμβανόμενα νευρωνικά δίκτυα, γνωστά και ως RNN, είναι μια κατηγορία τεχνητών νευρωνικών δικτύων που έχουν σχεδιαστεί για την επεξεργασία ακολουθιών

δεδομένων. Είναι ιδιαίτερα κατάλληλα για εργασίες όπως η επεξεργασία φυσικής γλώσσας, η αναγνώριση ομιλίας, η πρόβλεψη χρονοσειρών και άλλα. Τα RNNs χαρακτηρίζονται από την ικανότητά τους να διατηρούν μια μορφή μνήμης που τους επιτρέπει να συλλαμβάνουν και να επεξεργάζονται πληροφορίες από προηγούμενα χρονικά βήματα μέσα σε μια ακολουθία, καθιστώντας τα ιδιαίτερα ισχυρά για εργασίες που περιλαμβάνουν χρονικές εξαρτήσεις.

Οι μηχανισμοί προσοχής είναι ένα θεμελιώδες στοιχείο στη βαθιά μάθηση που επιτρέπει στα μοντέλα να εστιάζουν σε συγκεκριμένα τμήματα της εισόδου όταν κάνουν προβλέψεις ή κωδικοποιούν πληροφορίες. Έχουν χρησιμοποιηθεί ευρέως σε διάφορες εφαρμογές, συμπεριλαμβανομένης της επεξεργασίας φυσικής γλώσσας, της όρασης υπολογιστών και της αναγνώρισης ομιλίας. Οι μηχανισμοί προσοχής είναι ιδιαίτερα χρήσιμοι όταν έχουμε να κάνουμε με ακολουθίες ή όταν θέλουμε να δώσουμε διαφορετική σημασία σε διαφορετικά μέρη των δεδομένων εισόδου.

0.3 Μεθοδολογία και Πειράματα

0.3.1 Μεθοδολογία

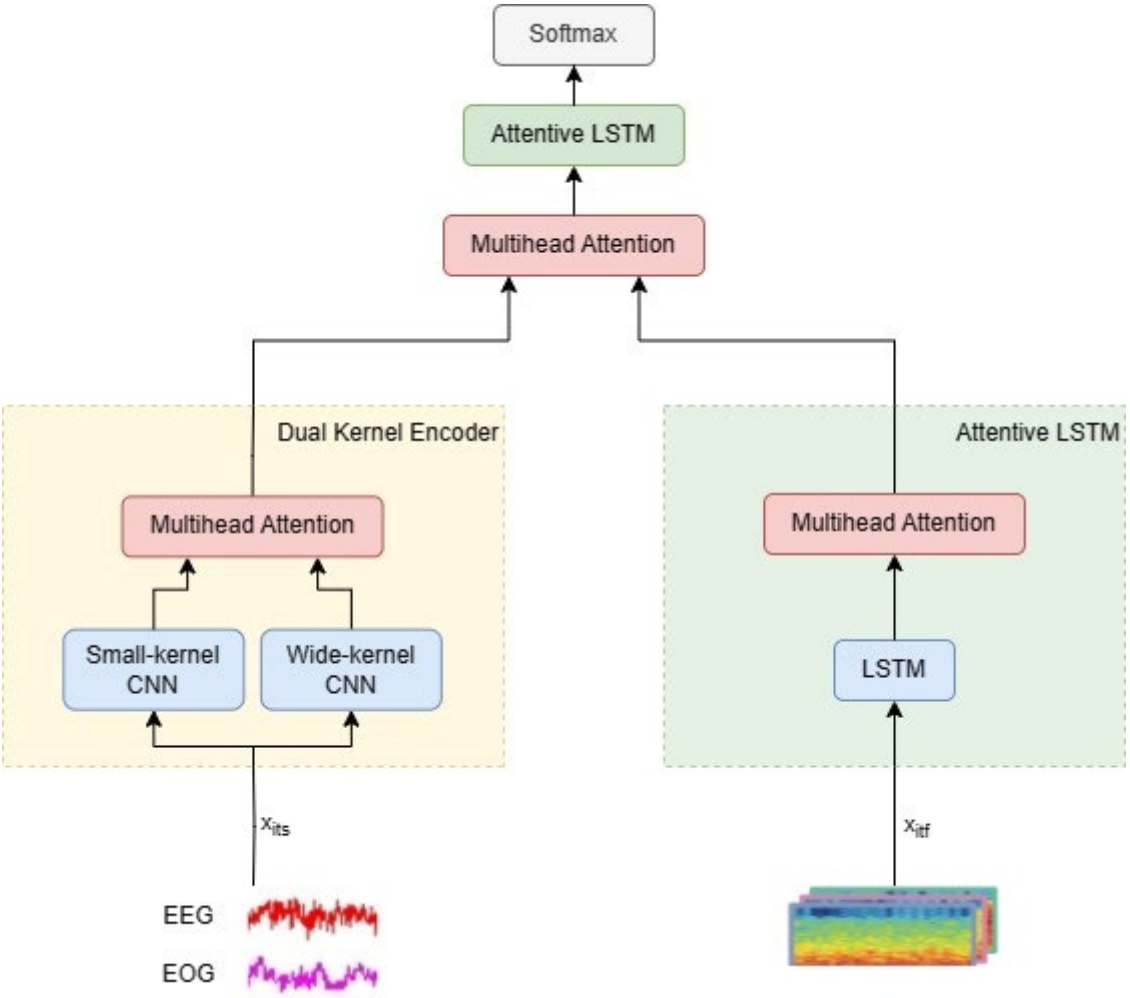
Σε αυτή την εργασία προτείνουμε ένα ένα-προς-ένα, πολυτροπικό μοντέλο μηχανικής μάθησης, το MMSleepNet, για την επίλυση του προβλήματος της αυτόματης κατηγοριοποίησης των σταδίων του ύπνου. Επιπλέον, εξετάζουμε την απόδοση του μοντέλου όταν αυτό εκπαιδεύεται εξαρχής χρησιμοποιώντας μικρή ποσότητα επισημασμένων δεδομένων έναντι στην περίπτωση όπου έχει προηγηθεί προεκπαίδευσή του με την χρήση μεγαλύτερου συνόλου δεδομένων χωρίς ετικέτες μέσο της χρήσης αλγορίθμων αυτοεπιβλεπόμενης μάθησης.

Η αρχιτεκτονική του προτεινόμενου μοντέλου φαίνεται στην Εικόνα 1 και αποτελείται από δύο κλάδους. Ο ένας κλάδος εξάγει χαρακτηριστικά από χρονοσειρές 30 δευτερολέπτων EEG και EOG ενός πολυυπνογραφήματος, ενώ ο άλλος κλάδος εξάγει χαρακτηριστικά από τα αντίστοιχα σπεκτρογράμματα. Τα εξαγόμενα χαρακτηριστικά στην συνέχεια ενοποιούνται μέσο ενός μηχανισμού προσοχής και δίνονται ως είσοδοι σε ένα RNN από το οποίο εξάγεται η τελική πρόβλεψη του μοντέλου.

Για την εκτίμηση της απόδοσης του μοντέλου χρησιμοποιήθηκε το σύνολο δεδομένων SleepEDF-20 που αποτελείται από πολυυπνογραφήματα 20 ανθρώπων με ηλικίες από 25 έως 34 ετών. Από το παραπάνω σύνολο δεδομένων χρησιμοποιήθηκαν είτε μόνο τα δεδομένα που αντιστοιχούσαν σε περίοδο ύπνου είτε προστέθηκαν και 30 επιπλέον λεπτά πριν και μετά την έναρξη και την λήξη του ύπνου αντίστοιχα, για να ανακουφιστεί σε κάποιον βαθμό το πρόβλημα της μη ισορροπημένης κατανομής ετικετών το οποίο εμφανίζεται τόσο στο συγκεκριμένο σύνολο δεδομένων, όσο και συνολικά στο πρόβλημα της αυτόματης κατηγοριοποίησης σταδίων του ύπνου.

Για να γίνει σωστή αξιολόγηση της απόδοσης του μοντέλου χρησιμοποιήθηκαν για σύγκριση τρία μοντέλα αναφοράς από την βιβλιογραφία, τα AttnSleep [1], XSleepNet [2] και DeepSleepNet [3].

Ως μητρικές για την αξιολόγηση της συνολικής επίδοσης των μοντέλων χρησιμοποιήθηκαν η ακρίβεια (Acc), το macro-averaged F1-score (MF1) και το Cohen Kappa (κ), ενώ το F1-score χρησιμοποιήθηκε στην περίπτωση αξιολόγησης της απόδοσης ανά ετικέτα.



Εικόνα 1: Αρχιτεκτονική του MMSleepNet

0.3.2 Εκπαίδευση με Όλες τις Ετικέτες

Τα αποτελέσματα για την ανά ετικέτα επίδοση του μοντέλου δίνονται στους Πίνακας 1 και Πίνακας 2 για το σύνολο δεδομένων SleepEDF-20 με και χωρίς την προσθήκη των 30 λεπτών πριν και μετά τον ύπνο, αντίστοιχα. Από τους προαναφερθέντες πίνακες διαπιστώνουμε ότι η προσθήκη των 30 λεπτών πριν και μετά τον ύπνο οδηγούν σε σημαντικές βελτιώσεις στην απόδοση του μοντέλου για τις περισσότερες μητρικές και ετικέτες. Επιπλέον, μπορούμε να παρατηρήσουμε ότι το MMSleepNet προσφέρει ως επί το πλείστον καλύτερα αποτελέσματα σε σχέση με τις δύο προηγούμενες μεθόδους, το AttnSleep [1] και DeepSleepNet [3].

Στον Πίνακας 3 δίνεται η συνολική απόδοση του μοντέλου σε σχέση με τις μεθόδους αναφοράς. Από τον δοθείσα πίνακα μπορούμε να συμπεράνουμε ότι το προτεινόμενο μοντέλο επιδεικνύει ανταγωνιστικά αποτελέσματα βάση των μητρικών που χρησιμοποιήθηκαν σε σύγκριση με τα παλαιότερα μοντέλα αναφοράς που υπάρχουν στην βιβλιογραφία.

Πίνακας 1: Αποτελέσματα κατηγοριοποίησης ανά ετικέτα στο SleepEDF-20

	MMSleepNet		
	PR	*RE*	*F1*
W	81.9	83.2	81.9
N1	54.3	39.1	44.4
N2	86.9	87.6	86.6
N3	82.5	86.0	83.1
REM	76.8	83.9	79.2

Πίνακας 2: Αποτελέσματα κατηγοριοποίησης ανά ετικέτα στο SleepEDF-20 (±30 min)

	MMSleepNet			AttnSleep [1]			DeepSleepNet [3]		
	PR	RE	F1	PR	RE	F1	PR	RE	F1
W	**94.3**	**92.0**	**93.0**	89.6	89.7	89.7	86.0	83.4	84.7
N1	**57.9**	**51.5**	**53.4**	47.1	39.1	42.8	43.5	50.1	46.6
N2	89.3	88.2	88.6	89.1	**88.6**	**88.8**	**90.5**	81.7	85.9
N3	**85.9**	81.2	82.1	80.7	89.8	**90.2**	77.1	**94.2**	84.8
REM	76.1	**87.3**	81.0	76.1	82.2	79.0	**80.9**	83.9	**82.4**

Πίνακας 3: Σύγκριση αποτελεσμάτων με μοντέλα αναφοράς

		Overall metrics			Per-class F1-Score				
Dataset	Model	Acc	κ	MF1	W	N1	N2	N3	N4
SleepEDF-20	**MMSleepNet**	81.6	0.73	75.1	81.9	44.4	86.6	83.1	79.2
	AttnSleep [1]	-	-	-	-	-	-	-	-
	XSleepNet [2]	**83.3**	**0.76**	**77.3**	-	-	-	-	-
	DeepSleepNet [3]	80.8	0.74	74.2	-	-	-	-	-
SleepEDF-20 (±30 min)	**MMSleepNet**	85.7	**0.82**	79.6	**93.0**	**53.4**	88.6	82.1	81.0
	AttnSleep [1]	84.4	0.79	78.1	89.7	42.6	**88.8**	**90.2**	79.0
	XSleepNet [2]	**86.4**	0.81	**80.9**	-	-	-	-	-
	DeepSleepNet [3]	81.9	0.76	76.6	86.7	45.5	85.1	83.3	**82.6**

0.3.3 Εκπαίδευση με Μειωμένες Ετικέτες

Για να αξιολογηθεί η απόδοση του μοντέλου, με και χωρίς την χρήση αλγορίθμων αυτοεπιβλεπόμενης μάθησης, στην περίπτωση που έχουμε πρόσβαση σε μικρό αριθμό επισημασμένων δειγμάτων σε σχέση με το συνολικό μέγεθος του συνόλου δεδομένων αξιολογήσαμε την απόδοση του μοντέλου βάση της ακρίβειας (Acc), του macro-averaged F1-score (MF1) και του Cohen Kappa (κ) καθώς και του ανά κατηγορίας F1-score τόσο για το μοντέλο χωρίς προεκπαίδευση, όσο και για την περίπτωση του προεκπαιδευμένου μοντέλου με χρήση κάποιου από τους επιλεγμένους αλγορίθμους μη επιβλεπόμενης μάθησης.

Τα αποτελέσματα που δίνονται στον Πίνακας 4 καταδεικνύουν ότι η επιλογή του αλγορίθμου SSL έχει τεράστια σημασία. Όταν χρησιμοποιείται ο λάθος αλγόριθμος για το πρόβλημα, μπορεί να οδηγήσει σε σημαντική υποβάθμιση των επιδόσεων του μοντέλου,

ενώ όταν εφαρμόζεται ένας πιο κατάλληλος αλγόριθμος μπορεί να οδηγήσει σε σημαντικές βελτιώσεις στις προγνωστικές ικανότητες του μοντέλου που προκύπτει.

Σε αυτό το σύνολο δεδομένων μειωμένης ετικέτας τα καλύτερα αποτελέσματα προκύπτουν όταν γίνεται προεκπαίδευση με τον αλγόριθμο CPC. Ο πιθανός λόγος για αυτό το αποτέλεσμα είναι ότι αυτός ο αλγόριθμος βασίζεται στην πρόβλεψη των μελλοντικών χρονικών βημάτων στον λανθάνοντα χώρο, γεγονός που του επιτρέπει να μάθει για τα χρονικά χαρακτηριστικά στα δεδομένα EEG και EOG, τα οποία είναι χρήσιμα για τη βελτίωση της απόδοσης του μοντέλου.

Πίνακας 4: Σύγκριση επίδοσης μοντέλου με και χωρίς προεκπαίδευση με SSL αλγορίθμους στο SleepEDF-20 (±30 min) σύνολο δεδομένων με χρήση μόνο του 10% των ετικετών

	Overall metrics			Per-class F1-Score				
Algorithm	Acc	κ	MF1	W	N1	N2	N3	REM
Supervised	65.7	0.56	**57.4**	82.5	23.9	73.3	**71.3**	49.8
ClsTran	60.6	0.48	49.9	73.8	16.6	72.7	68.5	36.5
SimCLR	63.2	0.53	54.8	81.1	21.2	74.0	60.2	47.4
CPC	**71.4**	**0.62**	**57.4**	**84.7**	**30.6**	**79.0**	67.6	**56.1**
TS-TCC	56.4	0.34	31.6	60.0	5.8	64.4	49.2	28.5

Chapter 1. Introduction

1.1 Motivation

Sleep is an essential aspect of human life which plays a pivotal role in maintaining overall health and well-being. Studies have suggested that sleep disorders may lead to a range of physical and mental illnesses [4]. Lack of sleep is associated with increased mortality and conditions, such as Alzheimer's disease [5]. Therefore, the study of sleep architecture and its various stages has long been of interest to researchers and healthcare professionals alike. Sleep stage recognition, a critical component of sleep analysis, involves categorizing the different phases of sleep, such as wakefulness, rapid eye movement (REM), and non-REM (NREM) stages. Accurate identification of these stages is paramount for understanding sleep patterns, diagnosing sleep disorders, and improving the quality of healthcare interventions.

Clinically, polysomnography (PSG) is an important technique to analyse sleep stage and sleep disorder diagnosis. The PSG system generally consists of brain activity (electroencephalogram, EEG), eye movements (electrooculogram, EOG), muscle activity or skeletal muscle activation (electromyogram, EMG), and heart rhythm (electrocardiogram, ECG) [6]. Traditional methods of sleep stage recognition primarily rely on manual scoring of PSG recordings by trained experts who follow predetermined guidelines to identify the distinct stages. Based on the Rechtschaffen and Kales (R&K) rules [7], PSG signals are typically split into segments of 30 seconds and classified into six sleep stages, which are wakefulness (Wake), four non-rapid eye movement stages (S1, S2, S3 and S4), and rapid eye movement (REM). Due to limited evidence of a difference between the S3 and S4, the American Academy of Sleep Medicine (AASM) later modified these rules with only three non-rapid eye movement (N1, N2 and N3) stages being defined, where the S3 and S4 stages in R&K were merged into N3 [8].

The main issue with having human experts performing manual sleep stage classification is that this process can be significantly time-consuming and error prone, while at the same time many decisions rely heavily on the subjective judgment of the individual expert and thus results can vary arbitrarily. To solve these problems, many studies have adopted machine learning methods for automatic sleep staging, which usually either use traditional signal processing methods or deep learning to extract features from raw signals. When traditional machine learning techniques, such as naive Bayes, k-nearest neighbour, support vector machines or random forests, are used researchers must choose what features to extract from the raw signal to use for classification. This means that they are highly dependent on the researchers' knowledge of sleep medicine when selecting appropriate features to distinguish among stages. To avoid this limitation many researchers have applied deep learning models, such as convolutional neural networks (CNNs) or LSTMs, to perform sleep stage classification.

Despite the strides made in recent deep learning-based sleep stage models, they often focus on single-channel time-domain data, primarily EEG or EOG signals. In contrast, clinicians routinely employ multichannel and multimodal data, incorporating signals like EEG, EOG, and EMG for comprehensive sleep stage scoring. Recognizing the potential insights and improvements gained from multimodal data, both in the time and frequency domains, becomes paramount for enhancing prediction accuracy and robustness.

Many recent studies use complex "sequence-to-sequence" models for analyzing EEG data and classifying sleep stages. This approach captures temporal relationships between sleep epochs but requires large models and training times. AttnSleep [1], uses a single-epoch model, achieves good results with faster training, suggesting the potential of simpler, one-to-one architectures in sleep analysis.

However, the development of deep learning models for automated sleep staging encounters a significant hurdle— the reliance of supervised learning models on massive, labelled datasets. Labelling such datasets is labour-intensive and expensive, limiting the feasibility of large-scale data creation. Although sleep labs accumulate extensive overnight recordings, the challenge of labelling restricts the potential of these models, making them cost-prohibitive. To address this, the exploration of self-supervised and semi-supervised learning techniques, requiring only a fraction or even no part of the dataset to be labelled, emerges as a highly desirable avenue, mitigating the high cost associated with obtaining labelled data.

This thesis introduces a novel single-epoch, multimodal and multichannel machine learning model designed to enhance the accuracy and robustness of sleep stage classification. By incorporating information from multiple physiological signals, the proposed model, MMSleepNet, aims to align more closely with the comprehensive approach employed by clinicians in scoring sleep stages. Furthermore, in recognizing the challenges associated with acquiring large, labelled datasets, the thesis explores the applicability of SSL algorithms in the context of automated sleep staging. By testing the model's performance under both full label and reduced label scenarios, where only a fraction of the dataset is labelled, this research seeks to shed light on the feasibility and efficacy of SSL techniques in mitigating the data labelling burden.

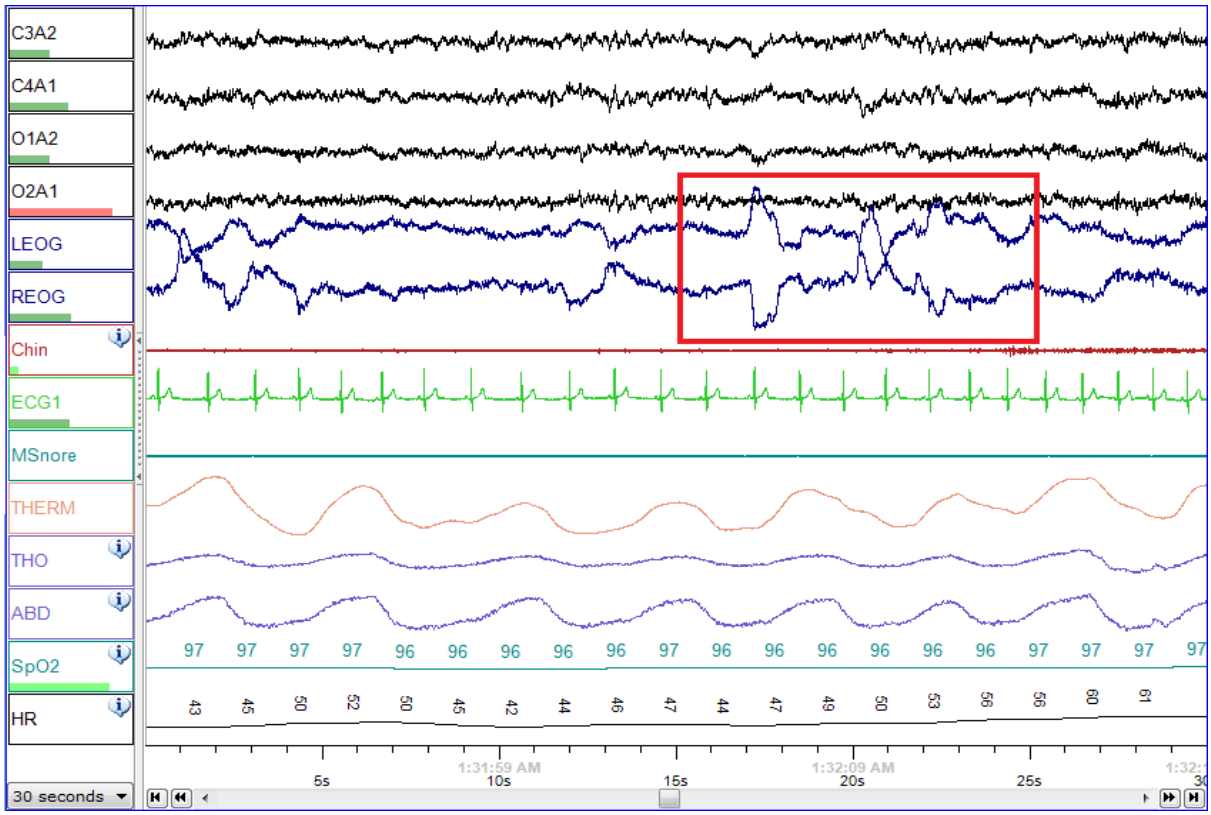


Figure 1: Screenshot of a PSG of a person in REM sleep [9]

1.2 Thesis Outline

In Chapter 2 of this thesis we present the theoretical background necessary to fully understand the motivation, methodology and results that are presented. Specifically, we first explore some theoretical aspects of sleep stage classification, then we present fundamental concepts in the field of machine learning in general and deep neural networks in particular. Finally, we discuss the most popular evaluation metrics used in the literature concerning automatic sleep staging.

In Chapter 3 we present and discuss related research present in the literature that explores similar issues to those addressed in this thesis. In particular, we explore previous methods for automatic sleep staging using both unimodal and multimodal methods and present works that utilize and develop self-supervised learning approaches which are relevant to the work done in this thesis.

In Chapter 4 we discuss the methodological approach that was followed to produce the results demonstrated and to tackle the problem at hand. To be precise, we focus on the dataset chosen to conduct our evaluation, the architecture of the proposed model, and the SSL algorithms tested in the reduced-label scenario.

In Chapter 5 we discuss the technical implementation details of our approach and present and evaluate the results of our experiments.

Finally, in Chapter 6 we discuss the overall conclusions resulting from the work conducted in this thesis and present ideas for related future experimentation and research.

Chapter 2. Theoretical Background

2.1 Sleep Stage Classification

2.1.1 Introduction

Automatic sleep stage classification (ASSC) has emerged as a significant domain in biomedical engineering, aiming to automate the identification of sleep stages from polysomnography (PSG) recordings. PSG is a comprehensive assessment of sleep architecture, involving the recording of multiple physiological signals, including electroencephalography (EEG), electrooculography (EOG), electromyography (EMG), and respiratory parameters. ASSC holds immense potential for enhancing clinical practice and research by reducing the manual workload of sleep experts, enabling real-time sleep monitoring, and facilitating large-scale sleep studies.

2.1.2 Sleep Stages

The human sleep cycle comprises five distinct stages: Wakefulness (W), Non-REM sleep (NREM), and Rapid Eye Movement (REM) sleep. NREM sleep can further be categorized into four stages (N1, N2, N3, and N4). Each sleep stage exhibits unique characteristics in terms of EEG patterns, eye movements, muscle activity, and physiological parameters. Approximately 75% of sleep is spent in the NREM stages, with the majority spent in the N2 stage [10]. A typical night's sleep consists of 4 to 5 sleep cycles, with the progression of sleep stages in the following order: N1, N2, N3, N2, REM [11]. A complete sleep cycle takes roughly 90 to 110 minutes. The first REM period is short, and, as the night progresses, longer periods of REM and decreased time in deep sleep (NREM) occur [12].

- **Wake/Alert**

  EEG recording: beta waves - highest frequency, lowest amplitude (alpha waves are seen during quiet/relaxed wakefulness)

  The first stage is the wake stage or stage W, which further depends on whether the eyes are open or closed. During eye-open wakefulness, beta waves predominate. As individuals become drowsy and close their eyes, alpha waves become the predominant pattern [13].

- **N1 (Stage 1) - Light Sleep (5%)**

  EEG recording: theta waves - low voltage

  This is the lightest stage of sleep and begins when more than 50% of the alpha waves are replaced with low-amplitude mixed-frequency (LAMF) activity. Muscle tone is present in the skeletal muscle, and breathing tends to occur at a regular rate. This stage lasts around 1 to 5 minutes, consisting of 5% of total sleep time.

- **N2 (Stage 2) - Deeper Sleep (45%)**

  EEG recording: sleep spindles and K complexes

  This stage represents deeper sleep as your heart rate and body temperate drop. It is characterized by the presence of sleep spindles, K-complexes, or both. Sleep spindles are brief, powerful bursts of neuronal firing in the superior temporal gyri, anterior cingulate,

insular cortices, and thalamus, inducing calcium influx into cortical pyramidal cells. This mechanism is believed to be integral to synaptic plasticity. Numerous studies suggest that sleep spindles play an important role in memory consolidation, specifically procedural and declarative memory [14].

K-complexes are long delta waves that last for approximately one second and are known to be the longest and most distinct of all brain waves. K-complexes have been shown to function in maintaining sleep and memory consolidation [15]. Stage 2 sleep lasts around 25 minutes in the first cycle and lengthens with each successive cycle, eventually consisting of about 45% of total sleep. This stage of sleep is when bruxism (teeth grinding) occurs.

- **N3 (Stage 3) - Deepest Non-REM Sleep (25%)**

EEG recording: delta waves - lowest frequency, highest amplitude

N3 is also known as slow-wave sleep (SWS). This is considered the deepest stage of sleep and is characterized by signals with much lower frequencies and higher amplitudes, known as delta waves. This stage is the most difficult to awaken from, and, for some people, even loud noises (> 100 decibels) will not awaken them. As people age, they tend to spend less time in this slow, delta wave sleep and more time in stage N2 sleep. Although this stage has the greatest arousal threshold, if someone is awoken during this stage, they will have a transient phase of mental fogginess, known as sleep inertia. Cognitive testing shows that individuals awakened during this stage tend to have moderately impaired mental performance for 30 minutes to an hour [16]. This is the stage when the body repairs and regrows tissues, builds bone and muscle, and strengthens the immune system. This is also the stage when sleepwalking, night terrors, and bedwetting occurs [17].

- **REM (25%)**

EEG recording: beta waves - similar to brain waves during wakefulness

REM is associated with dreaming and is not considered a restful sleep stage. While the EEG is similar to an awake individual, the skeletal muscles are atonic and without movement, except for the eyes and diaphragmatic breathing muscles, which remain active. However, the breathing rate becomes more erratic and irregular. This stage usually starts 90 minutes after you fall asleep, with each of your REM cycles getting longer throughout the night. The first period typically lasts 10 minutes, with the final one lasting up to an hour [18]. REM is when dreaming, nightmares, and penile/clitoral tumescence occur.

Important characteristics of REM:

- Associated with dreaming and irregular muscle movements as well as rapid movements of the eyes
- A person is more difficult to arouse by sensory stimuli than during SWS
- People tend to awaken spontaneously in the morning during an episode of REM sleep
- Loss of motor tone, increased brain O2 use, increased and variable pulse and blood pressure

- Increased levels of ACh
- The brain is highly active throughout REM sleep, increasing brain metabolism by up to 20% [19] [12]

2.1.3 Polysomnography

Polysomnography (PSG) is a comprehensive assessment of sleep architecture, involving the recording of multiple physiological signals. The four primary PSG signals are:

1. **Electroencephalography (EEG)**

   EEG measures electrical activity in the brain using electrodes placed on the scalp. EEG signals provide insights into brain activity during sleep, allowing for the identification of sleep stages and the assessment of sleep disorders.

2. **Electrooculography (EOG)**

   EOG measures eye movements using electrodes placed near the eyes. EOG signals are used to detect rapid eye movements (REM) and eye movements during NREM sleep.

3. **Electromyography (EMG)**

   EMG measures muscle activity using electrodes placed on muscles. EMG signals are used to assess muscle tone during sleep and detect arousals and awakenings.

4. **Respiratory Parameters**

   Respiratory parameters include respiratory rate, airflow, and oxygen saturation. They are measured using specialized respiratory sensors and provide information about breathing patterns during sleep, which can be crucial for detecting sleep-disordered breathing (SDB) [9].

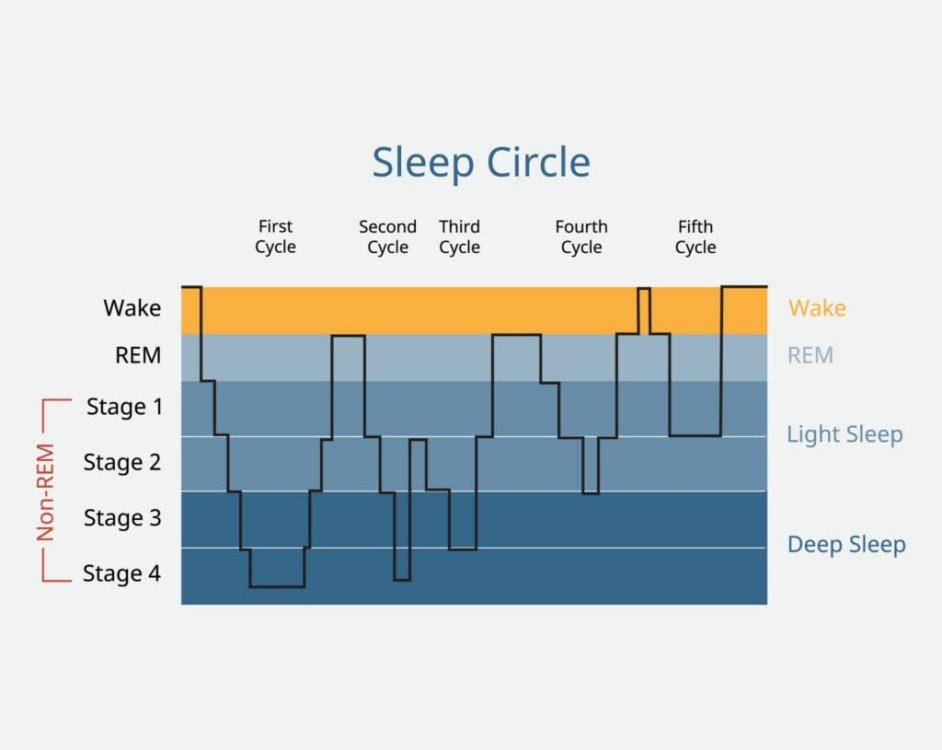


Figure 2: The sleep cycle [20]

2.2 Machine Learning

2.2.1 Introduction

According to Mitchell [21] “machine learning (ML) is a field of inquiry devoted to understanding and building methods that ’learn’, that is, methods that leverage data to improve performance on some set of tasks”. Machine learning algorithms automatically build a mathematical model from data alone to make predictions or decisions. The data used to train these models is called training data and the set of parameters that can be adjusted through this procedure, along with their underlying structure is often described as a machine learning model.

Machine learning is a subset of Artificial Intelligence, a field of study in computer science that develops and studies intelligent machines. Earlier efforts in Artificial Intelligence focused on expert knowledge systems which, mainly based on logical inference rules, derived new fragments of knowledge, or reasoned over statements. Machine learning bypasses several limitations of these earlier approaches, such as the need to formally describe all possible knowledge for a given task, and in many cases achieves significantly better results in real-world applications.

The most popular branch of machine learning models are neural networks. A neural network is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives signals then processes them and can signal neurons connected to it. The signal at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold [22].

Machine learning algorithms are used to solve a wide variety of tasks, in areas such as computer vision, speech recognition and email filtering. The computational modeling of such tasks is overly complex for humans, making it difficult or even unfeasible to develop conventional algorithms to perform the needed processes. Because of this, it turns out to be much more effective to help the machine develop its own algorithm in order to solve such problems.

2.2.2 Types of Machine Learning

The variety of different Machine Learning approaches can be categorized by the presence or absence of human influence on raw data, regarding the inclusion of a reward, the utilization of feedback or the existence of labels. There are three primary training practices, Supervised Learning, Unsupervised Learning, and Reinforcement Learning. We discuss those in the section below, along with Semi-Supervised Learning, which is a combination of Supervised and Unsupervised Learning.

2.2.2.1 Supervised Learning

In supervised learning the dataset is required to comprise pairs $(x, y), x \in X, y \in Y$, where X is some input space and Y is some output space. These pairs are samples, perhaps affected

by noise, from a mapping and the aim is to learn this mapping. If Y is discrete, we refer to the task as classification, in which case y is called the label and the dataset is said to be annotated, otherwise, if y is a continuous variable we refer to the task as regression. A point x of the input space is often called a feature vector, X often is a simple multidimensional vector space, but may also be something more complex such as a space of variable length sequences. This structure of training examples provides the algorithm with the ability to analyze the corresponding dataset, capture correlations and associations among samples and exploit this kind of information in order to produce an inferred function which is able to predict output y from a new input x that was not in the training set.

2.2.2.2 Unsupervised Learning

In unsupervised learning, the dataset is comprised only from the input variables $x, x \in X$, without having an output variable that would declare the label or target value of the mappings output. The goal for the unsupervised learning algorithms is often then to learn patterns exclusively from the unlabeled data, so as to be able to adequately group the samples into distinct classes and assign them a label or a value based on their class. This type of unsupervised learning is called clustering.

Another class of unsupervised learning models are known as generative models. Generative models are models that, given a dataset of examples $x_1, \ldots, x_n$, which are samples from a data distribution $p(x)$, aim at approximating this underlying probability distribution of the data. Synthetic data can then be generated by sampling from the approximated distribution.

One of the most important benefits of unsupervised learning techniques concerns the structure of the utilized dataset. Unlabeled data do not require any kind of human intervention or annotation, which is often a significantly time-consuming procedure, as such, they constitute the most common type of dataset for most real-world applications. Furthermore, such methods can be used for modeling more complex tasks compared to supervised learning, since they have the ability to automatically detect and reveal hidden patterns and intrinsic features of the underlying data distribution.

2.2.2.3 Semi-Supervised Learning

In semi-supervised learning, there are the input variables $x, x \in X$, only a subset of which have a corresponding output variable y, $y \in Y$. It is therefore a combination of supervised and unsupervised learning. Semi-supervised learning algorithms usually use the labelled subset of the dataset to label the remaining unlabeled samples. They can be used in various cases, where it is infeasible or too expensive to label every sample of the dataset.

2.2.2.4 Reinforcement learning

Reinforcement Learning differs greatly from the aforementioned categories. It is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize some notion of a cumulative reward.

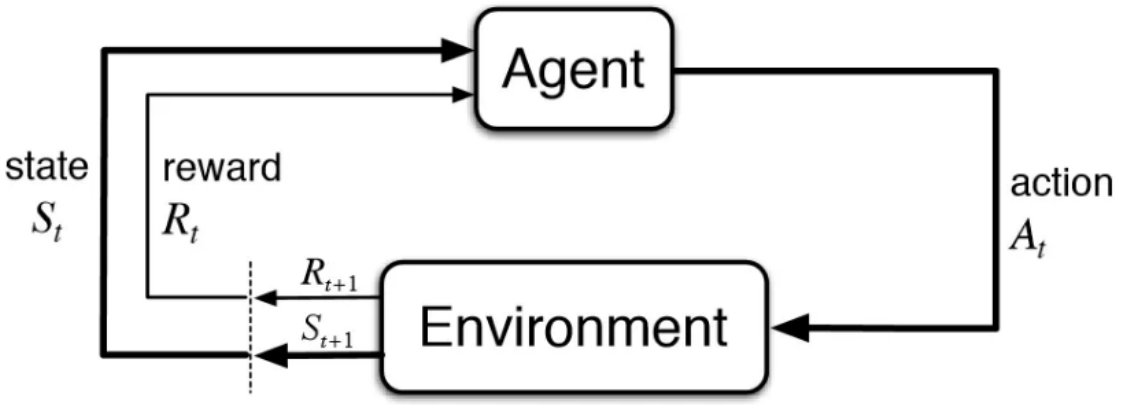


Figure 3: Action-reward feedback loop [23]

Usually, the aim is to find an algorithm for choosing actions based on the agent's and the environment's current state. This is called policy-based RL. Another, RL implementation is value-based RL, where the agent tries to maximize an arbitrary value function. Finally, model-based RL creates a virtual model for a certain environment and the agent learns to act within the constraints set by said model.

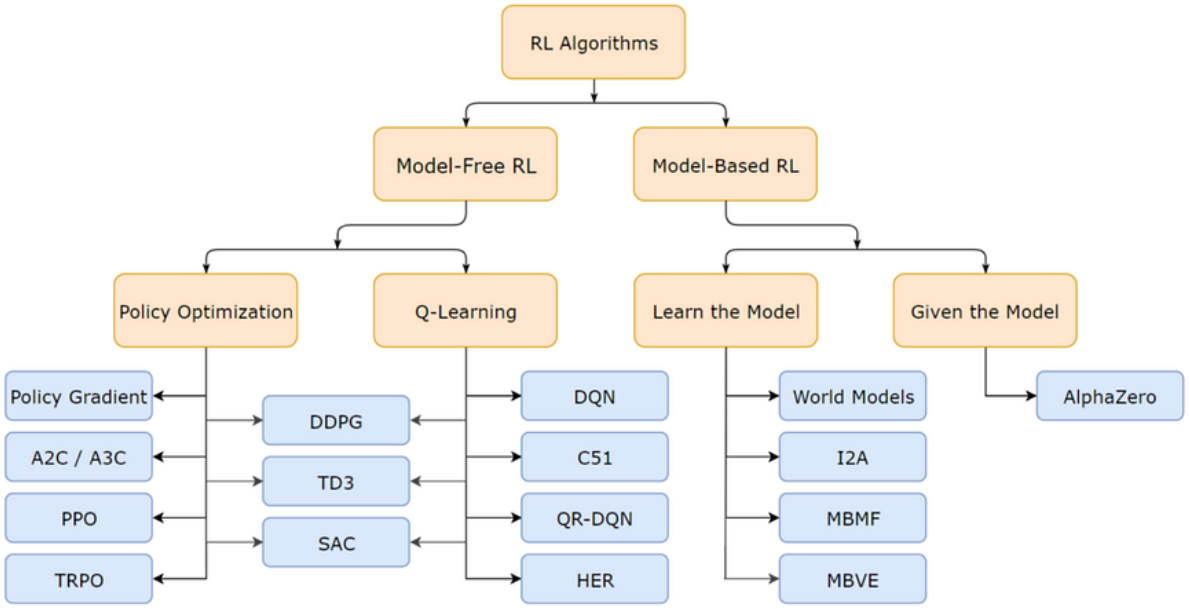


Figure 4: A Reinforcement Learning taxonomy as defined by OpenAI [24]

2.2.3 Deep Learning

2.2.3.1 Introduction

Deep learning is a machine learning technique that uses multiple layers to progressively extract higher-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces. Most deep learning methods use multi-layered neural network architectures, such as convolutional neural networks and transformers, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines.

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own. This does not eliminate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction.[25]

2.2.3.2 Concepts

2.2.3.2.1 Underfitting, Overfitting and Regularization

A central problem in machine learning is how to make an algorithm that will perform well not just on the training data, but also when given new inputs. Many strategies used in machine learning are explicitly designed to reduce the test error, possibly at the expense of increased training error. These strategies are known collectively as **regularization**.

Two related problems often encountered when developing machine learning models that regularization can alleviate are underfitting and overfitting.

Underfitting is a phenomenon in machine learning where a model is too simplistic to capture the underlying patterns in the data. This results in poor performance on both the training and testing data because the model is overly biased and cannot effectively learn from the training data. Underfit models are too simple or lack the necessary features or complexity to represent the true data distribution, leading to inadequate generalization to new, unseen data. Addressing underfitting typically involves increasing model complexity, adding relevant features, fine-tuning hyperparameters, expanding the training dataset, or applying regularization techniques to strike a better balance between simplicity and complexity, ultimately enabling the model to make more accurate predictions.

Overfitting is a phenomenon in machine learning where a model learns the training data so well that it essentially memorizes it, capturing noise and random fluctuations rather than the actual underlying patterns. As a result, an overfit model performs exceptionally well on the training data but poorly on new, unseen data because it has become too specialized and fails to generalize. Overfitting occurs when a model is excessively complex or has too many parameters relative to the amount of training data, allowing it to fit even the smallest details of the training set. To mitigate overfitting, techniques like reducing model complexity,

increasing the amount of training data, or applying regularization methods are employed to make the model more robust and better at generalizing to real-world data.

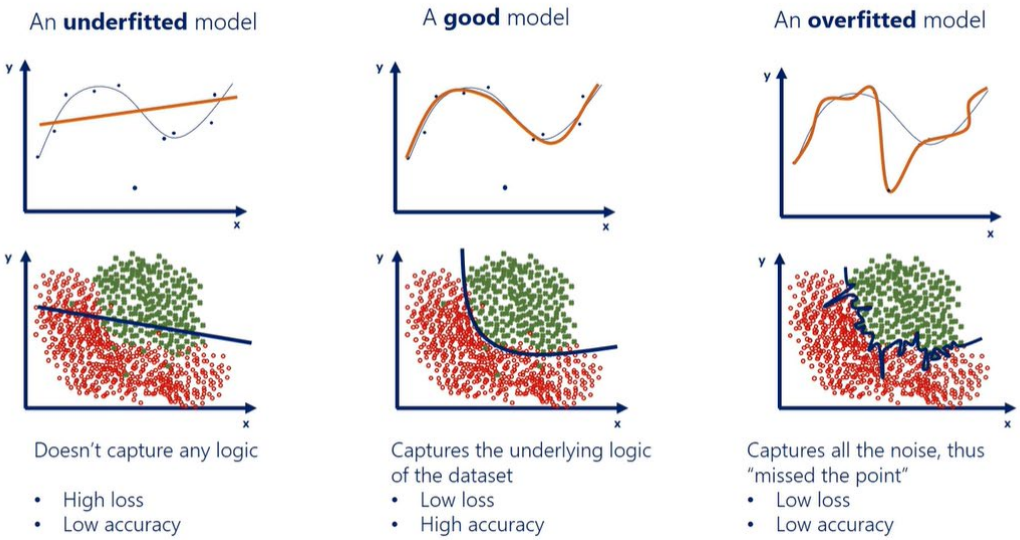


Figure 5: Overfitting and Underfitting [26]

There are various regularization techniques that can be used to alleviate the problems mentioned above. Some of the most important ones are:

- **L1 Regularization** (**Lasso**):
  L1 regularization adds a penalty term based on the absolute values of the model parameters to the loss function. It encourages some parameters to become exactly zero, effectively selecting a subset of the most important features, which can help with feature selection and to reduce model complexity.
- **L2 Regularization** (**Ridge**):
  L2 regularization adds a penalty term based on the square of the model parameters to the loss function. It reduces the magnitudes of all parameters without forcing any to become exactly zero. This encourages a more balanced reduction in parameter values, preventing large weights that might lead to overfitting.
- **Elastic Net Regularization**:
  Elastic Net combines L1 and L2 regularization. It adds a linear combination of both L1 and L2 penalty terms to the loss function. This provides a balance between feature selection (like L1) and parameter magnitude reduction (like L2).
- **Dropout**:
  Dropout is a regularization technique commonly used in neural networks. During training, dropout randomly deactivates (sets to zero) a fraction of neurons in each layer, making the network more robust and preventing it from relying too heavily on any specific set of neurons.

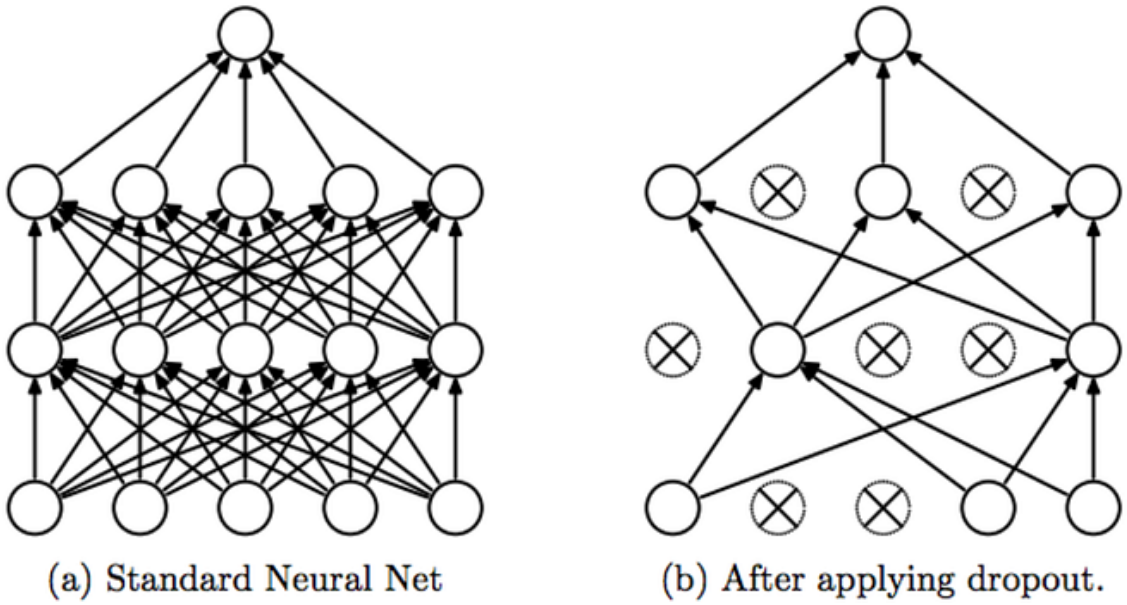


Figure 6: Dropout [27]

- **Early Stopping**:
  Early stopping is a simple technique where training is stopped when the model's performance on a validation dataset starts to degrade. This prevents the model from becoming overly complex and overfitting the training data.
- **Data Augmentation**:
  Data augmentation is a technique used to artificially increase the size of the training dataset by applying random transformations to the data, like rotation, cropping, or adding noise. This helps the model generalize better.
- **Batch Normalization**:
  Batch normalization normalizes the input to each layer in a neural network by adjusting the mean and variance of each mini batch of data. This stabilizes training, prevents vanishing/exploding gradients, and acts as a form of regularization.

These regularization techniques are essential tools for improving the generalization performance of machine learning models in various domains and algorithms. The choice of the appropriate technique depends on the specific problem and the characteristics of the dataset and model.

2.2.3.2.2 Activation Function

Activation functions are critical components in artificial neural networks and machine learning models. They introduce non-linearity to the model, allowing it to learn complex patterns and relationships within the data. Each node in a neural network typically applies an activation function to its weighted inputs and produces an output. Some common activation functions are explained below:

- **Sigmoid Function:**
  The sigmoid function (logistic function) maps input values to a range between 0 and 1. It's particularly useful in binary classification problems, as it resembles a step function and squashes input values to a probability-like output. However, it can suffer from vanishing gradient problems when used in deep networks.

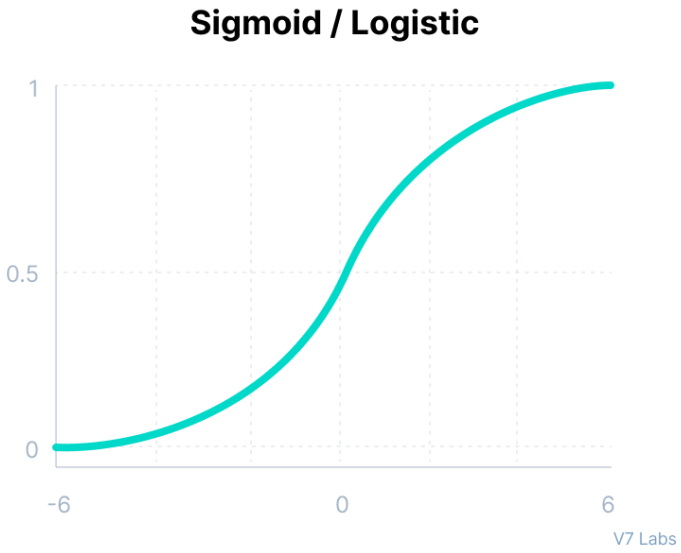


Figure 7: Sigmoid/Logistic Activation Function [28]

- **Hyperbolic Tangent (tanh):**
  The tanh function maps input values to a range between -1 and 1. It shares some characteristics with the sigmoid function but is zero-centered, which can help with training deep networks by mitigating the vanishing gradient problem to some extent.

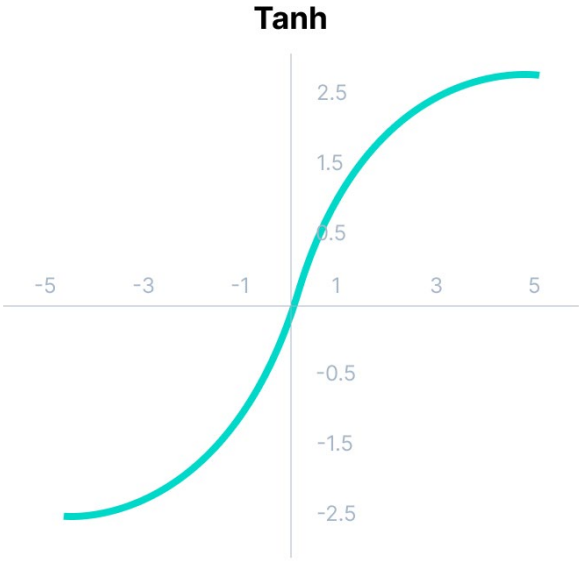


Figure 8: Tanh Function (Hyperbolic Tangent) [28]

- **Rectified Linear Unit (ReLU):**
  The ReLU activation function is one of the most widely used activation functions. It's simple and computationally efficient. It outputs the input as-is if it's positive, and zero if it's negative. ReLU can speed up training by mitigating the vanishing gradient problem and is particularly effective in deep networks.

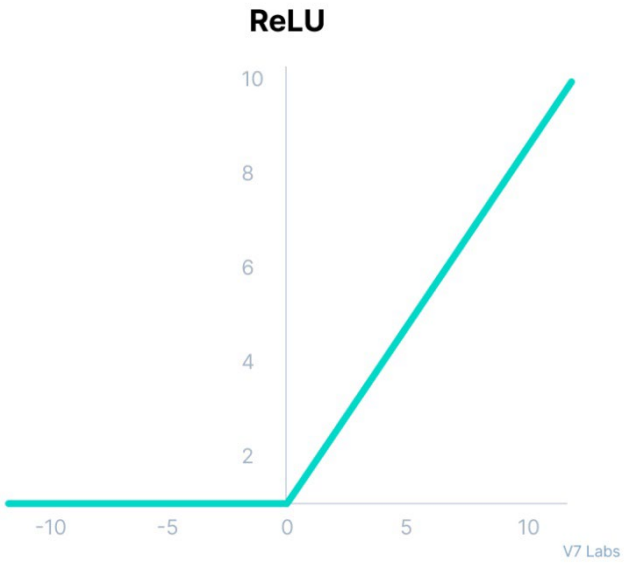


Figure 9: ReLU Activation Function [28]

- **Leaky ReLU:**
  Leaky ReLU is a variation of the ReLU function that allows a small, non-zero gradient for negative input values, preventing "dead" neurons that don't update their weights during training.

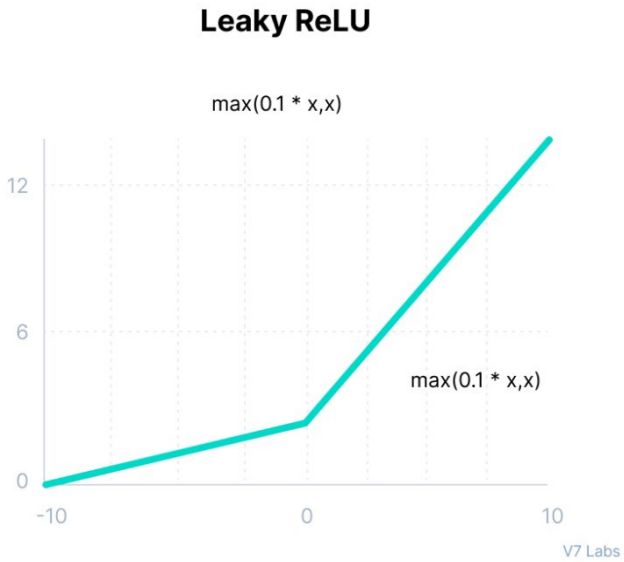


Figure 10: Leaky ReLU [28]

- **Parametric ReLU (PReLU):**
  PReLU extends Leaky ReLU by allowing the slope for negative values to be learned during training, making it more adaptive to the data.

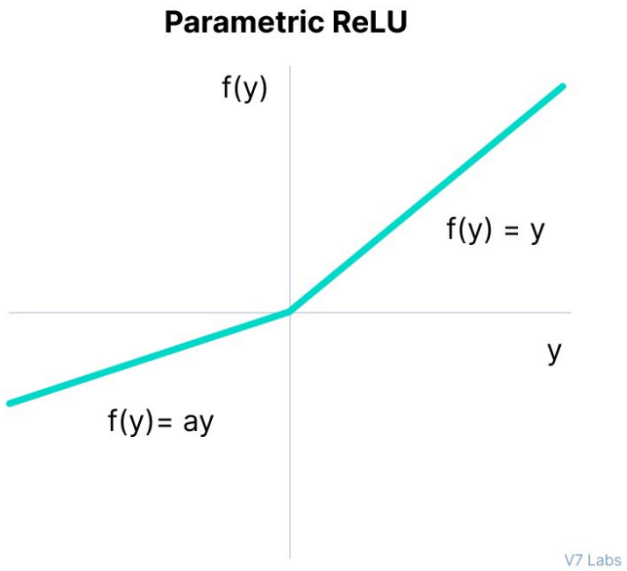


Figure 11: Parametric ReLU [28]

The choice of activation function often depends on the specific problem, architecture, and data characteristics. Experimentation is key to determine which activation function works best for a given task. Different activation functions have different properties and can significantly impact the training and performance of a neural network or machine learning model.

2.2.3.2.3 Loss Functions

Loss functions, also known as **cost functions**, are a fundamental component of neural networks used to measure the disparity between the predicted outputs of the model and the actual target values (ground truth). They play a pivotal role in training neural networks by quantifying how well or poorly the network is performing. The choice of an appropriate loss function depends on the specific type of machine learning task, whether it's classification, regression, or something else. Here are some common types of loss functions used in neural networks:

- **Mean Squared Error (MSE):**
  MSE is a common loss function for regression tasks. It measures the average squared difference between the predicted values and the actual target values. It is sensitive to outliers and penalizes larger errors more heavily.
- **Mean Absolute Error (MAE):**
  MAE is another loss function for regression that computes the average absolute differences between the predictions and the actual values. It is less sensitive to outliers compared to MSE.
- **Binary Cross-Entropy (Log Loss):**
  Binary cross-entropy is used in binary classification tasks. It quantifies the dissimilarity between the predicted probabilities and the true binary labels.

- **Categorical Cross-Entropy:**
  Categorical cross-entropy is used in multi-class classification tasks, where there are more than two classes. It measures the divergence between the predicted class probabilities and the true class labels.
- **Kullback-Leibler Divergence (KLD):**
  KLD is often used in probabilistic models and measures the difference between two probability distributions.

Selecting the appropriate loss function is crucial for training a neural network effectively, as it directly impacts the optimization process. Different loss functions emphasize different aspects of the prediction error, and choosing the right one can lead to better model performance and faster convergence during training.

2.2.3.2.4 Gradient Descent

Gradient descent is a fundamental optimization algorithm used in machine learning and deep learning to minimize a loss function and find the optimal values of a model's parameters (weights and biases). It operates by first initializing the parameters and then iteratively calculating the gradient of the loss function with respect to each parameter. The gradient indicates the direction and magnitude of the steepest increase in the loss, serving as a guide for adjusting the parameters to reduce the error between the model's predictions and the actual target values.

These parameter updates are made in the direction opposite to the gradient, with the step size controlled by a user-defined parameter known as the **learning rate**. Smaller learning rates lead to more precise but slower convergence, while larger rates can expedite convergence but may risk overshooting the minimum.

Variants of gradient descent, such as Stochastic Gradient Descent (SGD), Mini-Batch Gradient Descent, Momentum, and adaptive methods like Adagrad, RMSprop, and Adam, offer different trade-offs in terms of computational efficiency and the ability to escape local minima.

Overall, gradient descent is a crucial tool for training machine learning models, ensuring they learn the best parameters through iterative optimization.

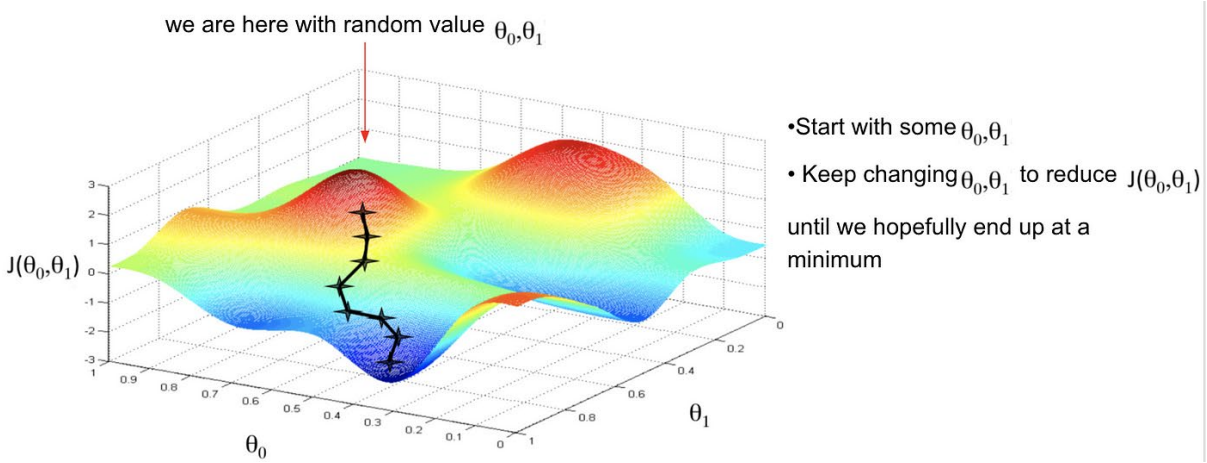


Figure 12: Gradient descent with 2 parameters θ_0, θ_1 and loss function J [29]

2.2.3.2.5 Backpropagation

Backpropagation, is a fundamental algorithm used for training artificial neural networks, particularly feedforward neural networks. It's the key process through which a neural network learns by updating its parameters (weights and biases) to minimize the error between its predictions and the actual target values. Here's a step-by-step explanation of how backpropagation works:

1. **Forward Pass:**
   In the forward pass, input data is fed into the neural network, and it propagates through the layers, one layer at a time, to produce predictions. Each neuron in a layer computes a weighted sum of its inputs, adds a bias term, and applies an activation function to generate an output. This output is then used as input for the next layer.
2. **Compute Loss:**
   After the forward pass, the network's predictions are compared to the actual target values, and a loss (or cost) function is calculated. The loss function quantifies how far off the predictions are from the true values, providing a measure of the network's performance.
3. **Backward Pass (Backpropagation):**
   Backpropagation is the process of propagating the error backward through the network to compute the gradients of the loss with respect to the model's parameters. This involves applying the chain rule of calculus to calculate how much each parameter contributed to the error. The gradients represent the direction and magnitude of change needed for each parameter to minimize the loss.
4. **Update Parameters:**
   The computed gradients are used to adjust the model's parameters through an optimization algorithm, typically gradient descent or one of its variants. The parameters are updated in the direction that minimizes the loss. The learning rate controls the size of the steps in this parameter space.
5. **Repeat:**
   Steps 1 to 4 are repeated for a fixed number of iterations (epochs) or until a convergence criterion is met. During each iteration, the neural network gets better at making predictions as the parameters are refined to minimize the loss.

Backpropagation enables neural networks to learn from data by iteratively fine-tuning their parameters. It's a crucial process for supervised learning tasks, allowing the model to adjust its weights and biases to produce more accurate predictions over time. To prevent issues like vanishing gradients, various activation functions and regularization techniques are often used in conjunction with backpropagation. Additionally, deep learning frameworks and libraries have made it easier to implement backpropagation in practice, making it a foundational tool for building and training neural networks.

2.2.3.3 Models

2.2.3.3.1 Neural Networks

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks (ANNs) are a collection of interconnected units or nodes, called artificial neurons, which are organized in multiple layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.[30]

The goal then of a neural network, also often called a feedforward network, is to approximate some function f^*. For example, for a classifier, $y = f^*(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = f\ (x;\ \theta)$ and learns the value of the parameters θ that result in the best function approximation.

These models are called feedforward because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself. When feedforward neural networks are extended to include feedback connections, they are called recurrent neural networks, presented in section 2.2.3.3.3.

Feedforward neural networks are called networks because they are typically represented by composing together many different functions. The model is associated with a directed acyclic graph describing how the functions are composed together. For example, we might have three functions $f(1)$, $f(2)$, and $f(3)$ connected in a chain, to form $f(x) = f(3)(f(2)(f(1)(x)))$. These chain structures are the most commonly used structures of neural networks. In this case, $f(1)$ is called the first layer of the network, $f(2)$ is called the second layer, and so on. The overall length of the chain gives the depth of the model. The final layer of a feedforward network is called the output layer. During neural network training, we drive $f(x)$ to match $f^*(x)$. The training data provides us with noisy, approximate examples of $f^*(x)$ evaluated at different training points.[31]

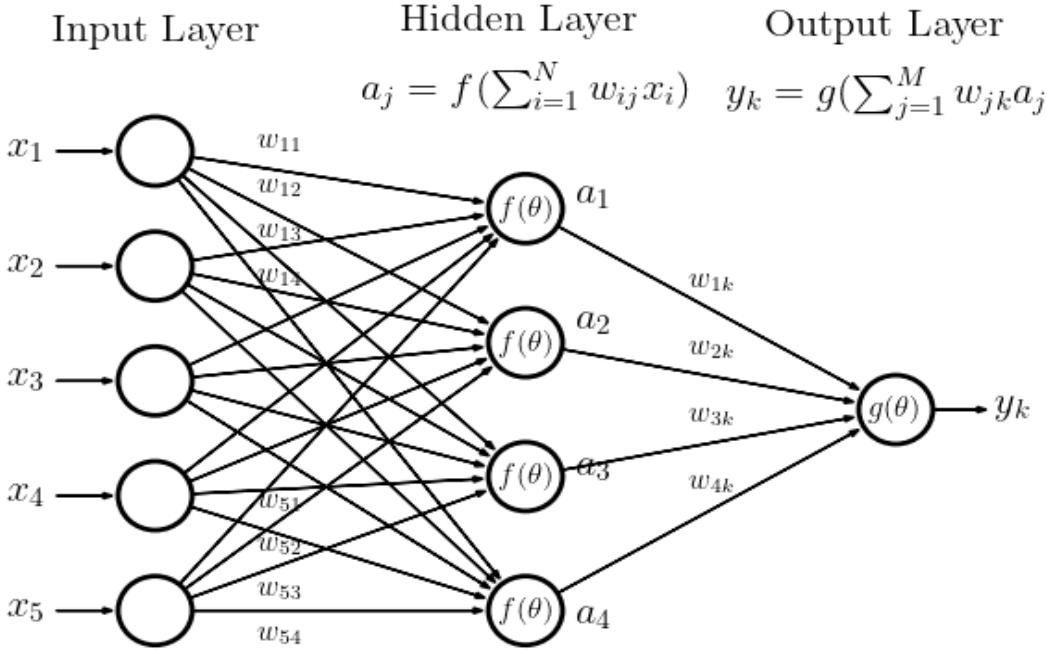


Figure 13: A NN with 1 hidden layer. Where, w_{ij} is the weight that multiplies the activation from the i-th neuron of the previous layer to the j-th neuron of the current layer, f is the activation function of the hidden layer and g is the activation function of the output layer [32]

2.2.3.3.2 Convolutional neural network

Convolutional neural networks, also known as **CNNs**, are a specialized kind of neural network that excels at processing data which has a known, grid-like topology. Examples include time-series data, which can be thought of as a 1D grid taking samples at regular time intervals, and image data, which can be thought of as a 2D grid of pixels. The main difference between CNNs and a simple feedforward neural network is that, in at least one layer of an CNNs the conventional matrix multiplication is replaced by a convolutional layer.

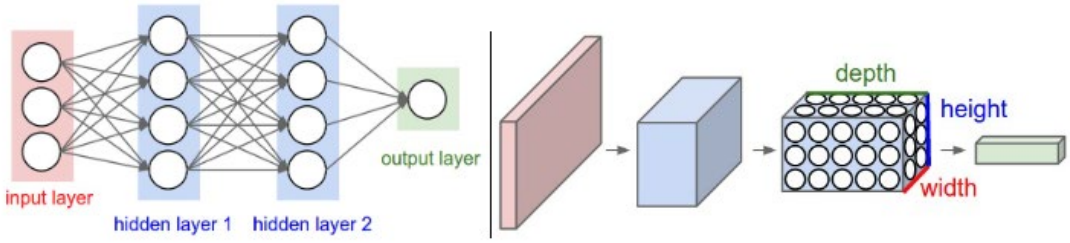


Figure 14: CNN architecture in comparison with a typical MLP model [33]

The **convolutional layer** is the fundamental building block of a CNN model. This layer applies grid-shaped feature detector filters to the input image, through a mathematical operation, called Convolution. This procedure is graphically displayed in Figure 15 for the case of a 2-dimensional grid. Some parameters of a convolutional layer, like the weight values, adjust during training through the process of backpropagation and gradient descent. However, there are three hyperparameters which affect the volume size of the output that need to be set before the training of the neural network begins. These include:

- The number of filters affects the depth of the output. For example, three distinct filters would yield three different feature maps, creating a depth of three.
- Stride is the distance, or number of pixels, that the kernel moves over the input matrix. While stride values of two or greater are rare, a larger stride yields a smaller output.
- Zero-padding is usually used when the filters do not fit the input image. This sets all elements that fall outside of the input matrix to zero, producing a larger or equally sized output. There are three types of padding:
    - Valid padding:
      This is also known as no padding. In this case, the last convolution is dropped if dimensions do not align.
    - Same padding:
      This padding ensures that the output layer is the same size as the input layer.
    - Full padding:
      This type of padding increases the size of the output by adding zeros to the border of the input.[34]

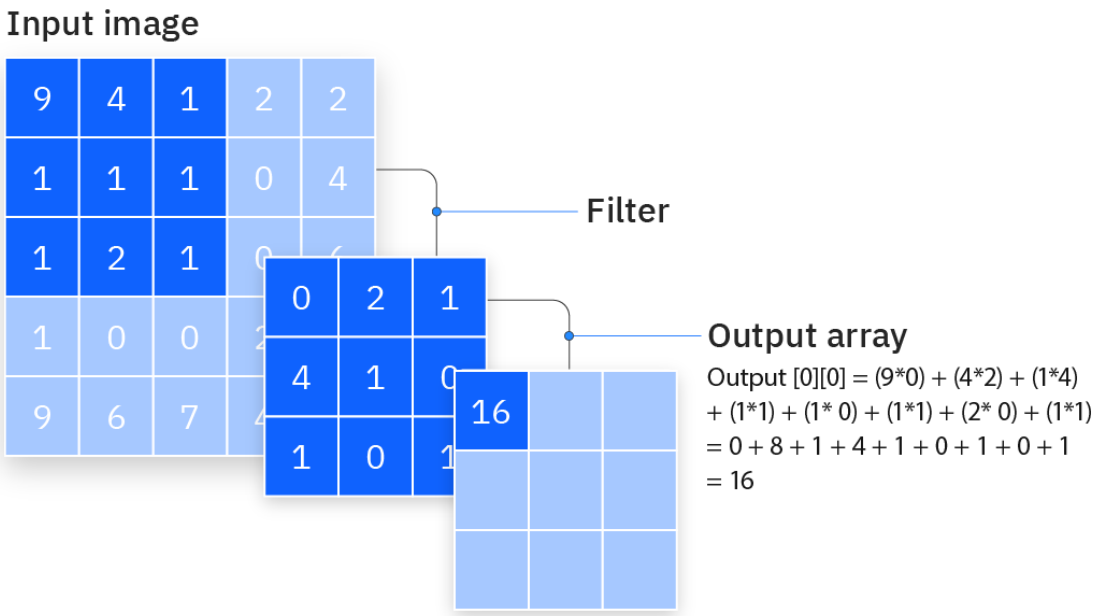


Figure 15: Illustration of the Convolution Operation for the 2 dimensional case [34]

In Figure 15, it is important to note that during convolution the filter's weights remain fixed as it moves across the image, which is also known as parameter sharing. This means that we need to store fewer parameters, when compared to a simple neural network where every output unit interacts with every input unit, which both reduces the memory requirements of the model and improves its statistical efficiency. It also means that computing the output requires fewer operations. Due to parameter sharing, the layers of convolution neural network will also have a property of equivariance to translation. It says that if we changed the input in a way, the output would also get changed in the same way.

A typical layer of a convolutional network consists of three stages. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations. In the second stage, each linear activation is run through a nonlinear activation function, such as the rectified linear activation function. In the third stage, we use a pooling function to modify the output of the layer further. The third stage is performed by the pooling layer.

The **pooling layer** replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. Pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change. This can be a very useful property if we care more about whether some feature is present than exactly where it is. For example, when determining whether an image contains a face, we need not know the location of the eyes with pixel-perfect accuracy, we just need to know that there is an eye on the left side of the face and an eye on the right side of the face.

Pooling over spatial regions produces invariance to translation, but if we pool over the outputs of separately parametrized convolutions, the features can learn which transformations to become invariant to, as shown in Figure 16.

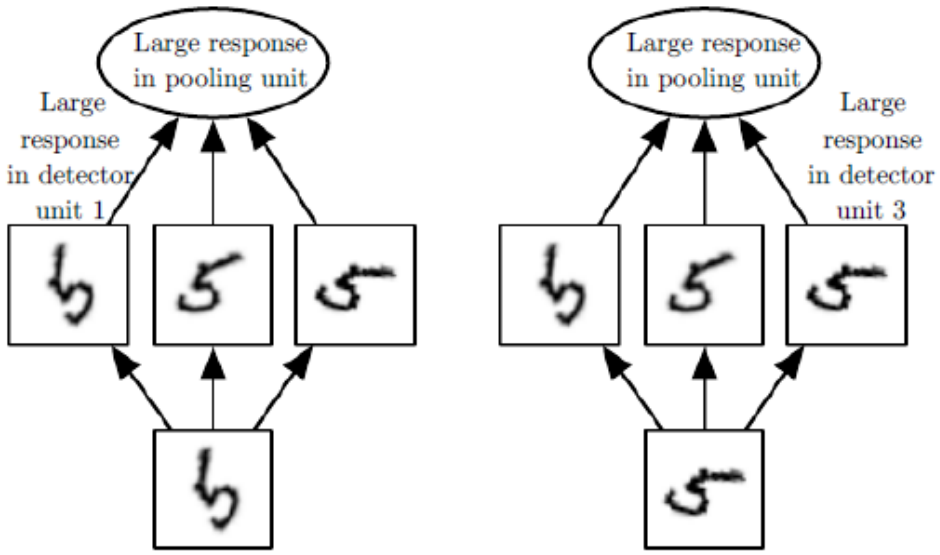


Figure 16: Example of how three learned filters and a max pooling unit can learn to become invariant to rotation [31]

There are several pooling functions such as the average of the rectangular neighborhood, L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel. However, the most popular process is max pooling, which reports the maximum output from the neighborhood.[31], [35]

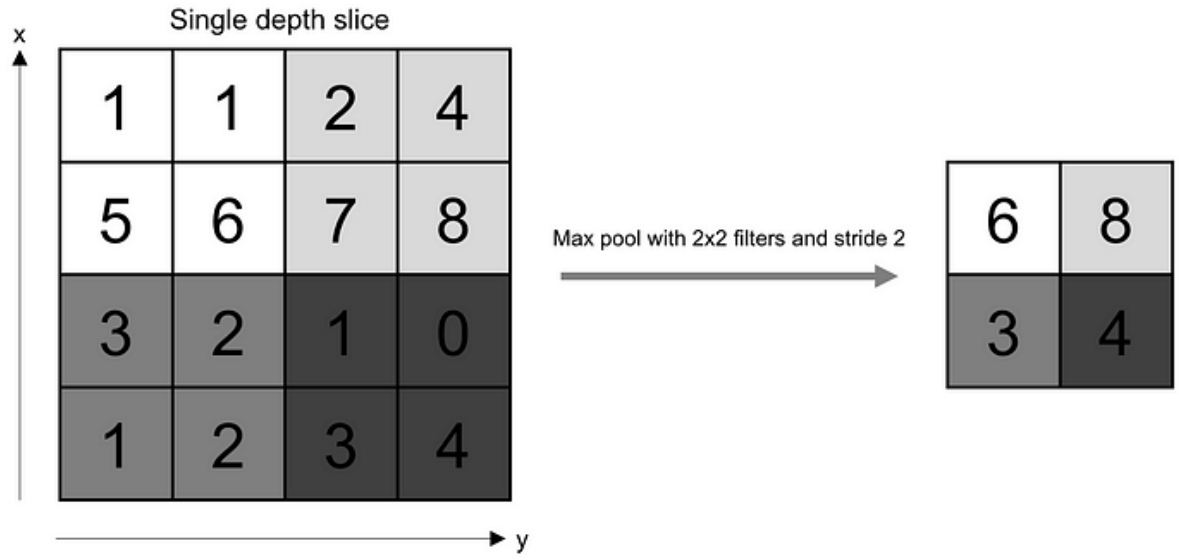


Figure 17: Example of max pooling operation [35]

2.2.3.3.3 Recurrent Neural Networks

Recurrent Neural Networks, also known as **RNNs**, are a class of artificial neural networks designed for processing sequences of data. They are particularly well-suited for tasks such as natural language processing, speech recognition, time series prediction, and more. RNNs are characterized by their ability to maintain a form of memory that enables them to capture and process information from previous time steps within a sequence, making them especially powerful for tasks involving temporal dependencies.

Just as convolutional networks can readily scale to images with large width and height, and some convolutional networks can process images of variable size, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length.[31]

RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations. At its core, an RNN is a type of neural network with an internal hidden state that is updated at each time step as it processes a sequence of input data. The architecture of a simple RNN, which is shown in Figure 18, can be described as follows:

- **Input:** $x(t)$ is taken as the input to the network at time step t. For example, $x(1)$, could be a one-hot vector corresponding to a word of a sentence.
- **Hidden State**: $h(t)$ represents a hidden state at time t and acts as "memory" of the network. $h(t)$ is calculated based on the current input and the previous time step's hidden state: $h(t) = f(U * x(t) + W * h(t-1))$. The function f is taken to be a non-linear transformation such as tanh, ReLU.
- **Weights**: The RNN has input to hidden connections parameterized by a weight matrix U, hidden-to-hidden recurrent connections parameterized by a weight matrix W, and hidden-to-output connections parameterized by a weight matrix V and all these weights (U, V, W) are shared across time.
- **Output**: $o(t)$ illustrates the output of the network. At each time step, the RNN produces an output based on the current hidden state. This output can be used for various tasks, such as predicting the next item in a sequence or providing a summary of the entire sequence.[36]

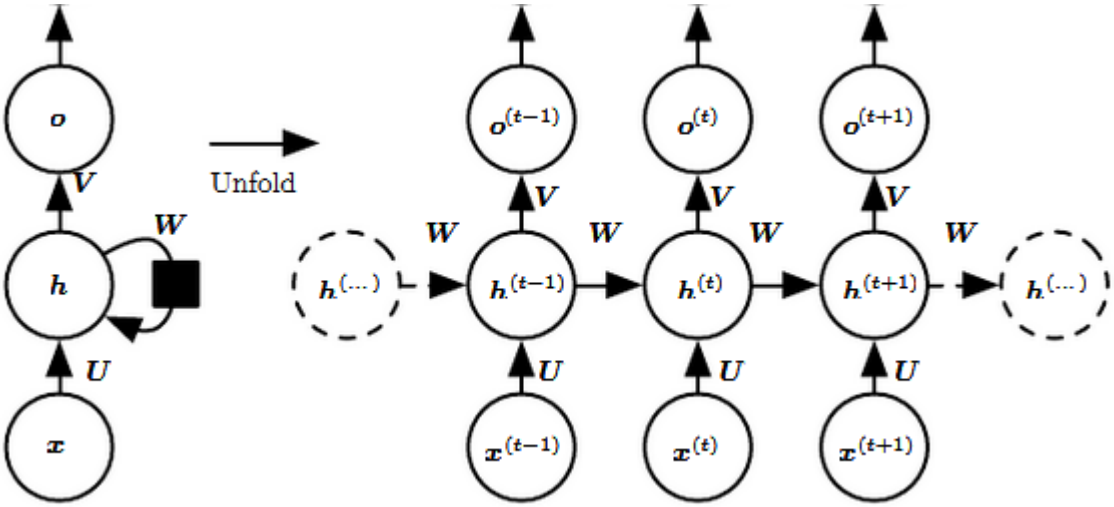


Figure 18: RNN architecture. (Left) Circuit diagram. The black square indicates a delay of a single time step. (Right) The same network seen as an unfolded computational graph, where each node is now associated with one particular time instance [36]

To train an RNN, the network's parameters (i.e., the weight matrices) are optimized to minimize a specified loss function. This is typically done using backpropagation through time (BPTT) or gradient descent.

Backpropagation Through Time, also known as **BPTT**, is a crucial training technique for Recurrent Neural Networks (RNNs), extending the backpropagation algorithm to handle sequential data. During the forward pass, the RNN processes the input sequence step by step, computing hidden states and outputs at each time step. Losses are calculated at each time step by comparing the predicted outputs with target values. In the backward pass, starting from the last time step, gradients with respect to model parameters are computed using the chain rule of calculus and then aggregated over all time steps. These gradients are used to update the model's parameters with optimization algorithms like gradient descent, minimizing the average loss across the entire sequence. To manage the computational burden and address gradient issues, such as the vanishing and exploding gradient problem, sequences are often truncated, and gradient clipping is employed. While effective for many sequential tasks, BPTT has limitations in capturing long-range dependencies, for which more advanced RNN variants like LSTMs and GRUs with gating mechanisms are designed.

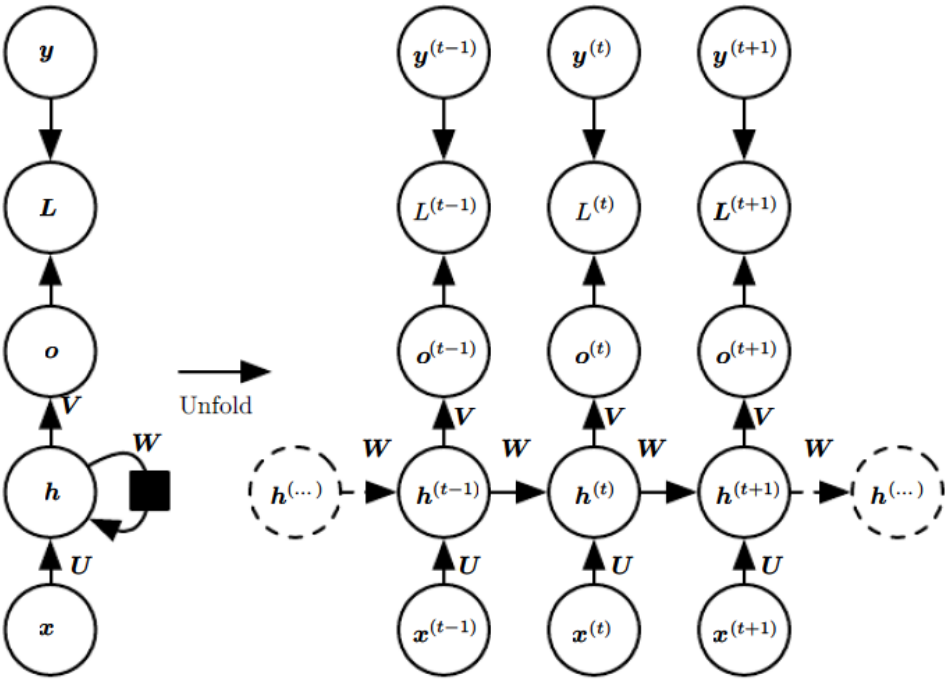


Figure 19: The computational graph used to compute the training loss of a RNN [31]

Long Short-Term Memory networks, also known as **LSTMs**, are a type of recurrent neural network (RNN) architecture, introduced by Hochreiter and Schmidhuber in 1997 [37], specifically designed to address the vanishing gradient problem and effectively capture long-range dependencies in sequential data. LSTMs comprise memory cells and gated mechanisms, including the forget (f_t), input (i_t), and output (o_t) gates, which control the flow of information. The forget gate determines what to retain or discard from the previous cell state (c_t), the input gate calculates candidate values for updating the cell state, and the output gate governs how much information should be used to compute the hidden state (h_t) at the current time step. LSTMs cell state acts as long-term memory, enabling them to excel in tasks requiring memory of distant events, such as natural language processing and time series analysis.

Given a sequence $x_1, x_2, \ldots, x_t, \ldots, x_n$ of vectors of an input sequence of length n, for vector x_t, with inputs h_{t-1} and c_{t-1}, h_t and c_t are computed as follows:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right)$$

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right)$$

$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right)$$

$$u_t = tanh(W_u x_t + U_u h_{t-1} + b_u)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot u_t$$

$$h_t = o_t \odot tanh(c_t)$$

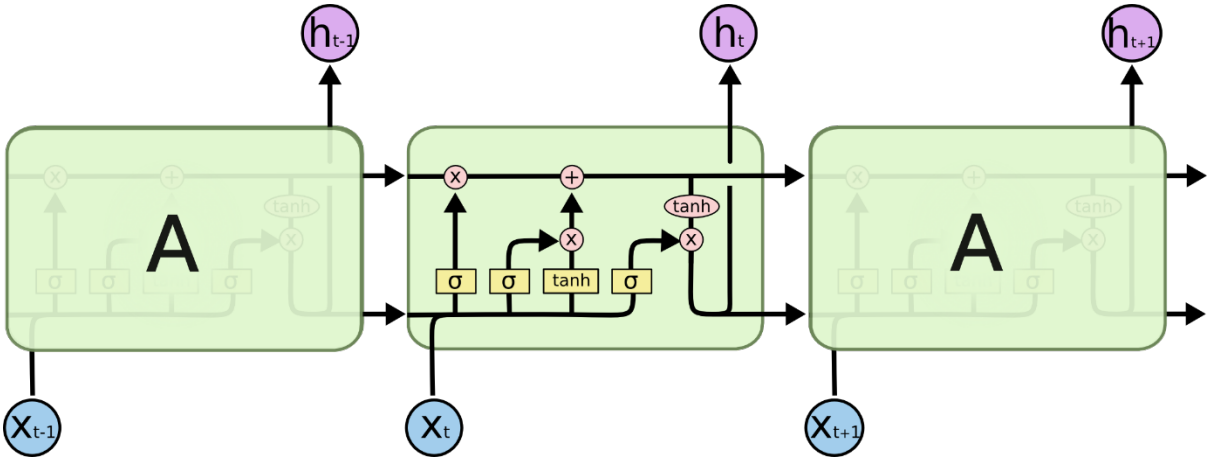


Figure 20: LSTM architecture [38]

Gated Recurrent Units, also known as **GRUs**, are a type of recurrent neural network (RNN) architecture that, like LSTMs, address the vanishing gradient problem while simplifying the model. GRUs feature gating mechanisms, including an update gate (z_t) and a reset gate (r_t), which allow them to control the flow of information within the network. The update gate determines how much of the previous hidden state to carry forward, while the reset gate regulates the extent to which the current input should influence the hidden state. This architecture combines short-term and long-term memory capabilities within a more streamlined structure, making it computationally efficient and often suitable for tasks involving sequential data processing, where capturing both recent and distant dependencies is crucial.

The equations that describe the function of a GRU are the following:

$$z_t = \sigma\left(W_z x_t \; + \; U_z x_t \; + \; b_z\right)$$

$$r_t = \sigma\left(W_r x_t \; + \; U_r x_t \; + \; b_r\right)$$

$$u_t = tanh(W_h x_t \; + \; U_h(r_t \odot h_{t-1}) \; + \; b_h)$$

$$h_t = (1 \; - \; z_t) \odot h_{t\,-1} \; + \; z_t \odot u_t$$

Although LSTMs and GRUs partially solved the vanishing gradient problem, the passing of the information through a series of recurrent connections evokes information loss. Moreover, the inherently sequential nature of recurrent networks makes it hard to do computation in parallel. A solution to these problems was proposed by Bahdanau et al [39] in the form of Attention Mechanisms, which will be discussed in section 2.2.3.3.4.

2.2.3.3.4 Attention Mechanisms

Attention mechanisms in deep learning are a fundamental component that enable models to focus on specific parts of the input when making predictions or encoding information. They have been widely used in various applications, including natural language processing, computer vision, and speech recognition. Attention mechanisms are particularly useful when dealing with sequences or when you want to give different importance to different parts of the input data.

An attention mechanism usually has three key components, called the Query (Q), the Key (K) and the value (V). The query represents what part of the input we are currently interested in. For each element in the input, we associate a key and a value. The key is used to measure the similarity between the query and that element, while the value holds the information we want to focus on. The key/value/query concept is analogous to ideas used in retrieval systems.

The core idea of an attention mechanism is to calculate a weighted sum of values based on the compatibility (similarity) between the query and keys. This weighted sum is used to generate an attended context, which is then combined with the query to produce an output.

In the following table several popular techniques for computing attention are presented. The symbol s_t denotes the predictions, while different W indicate trainable matrices:

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Figure 21: Ways to compute attention [40]

Self-attention is an attention mechanism that is used within a sequence, where for each element in the input sequence, the model calculates an attention score with respect to all other elements in the same sequence. These attention scores determine how much focus the model should place on each element when producing an output. In this way, it enables the model to understand contextual information and relationships.

Rather than only computing the attention once, the **multi-head attention** mechanism runs through the scaled dot-product attention multiple times in parallel. The independent attention outputs are simply concatenated and linearly transformed into the expected dimensions. Each head operates in parallel and learns distinct patterns and dependencies, enhancing the model's expressiveness and enabling it to capture complex relationships, as it allows the model to jointly attend to information from different representation subspaces at different positions. Whereas, with a single attention head, averaging inhibits this.

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$

$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices W_i^Q, W_i^K, W_i^V.

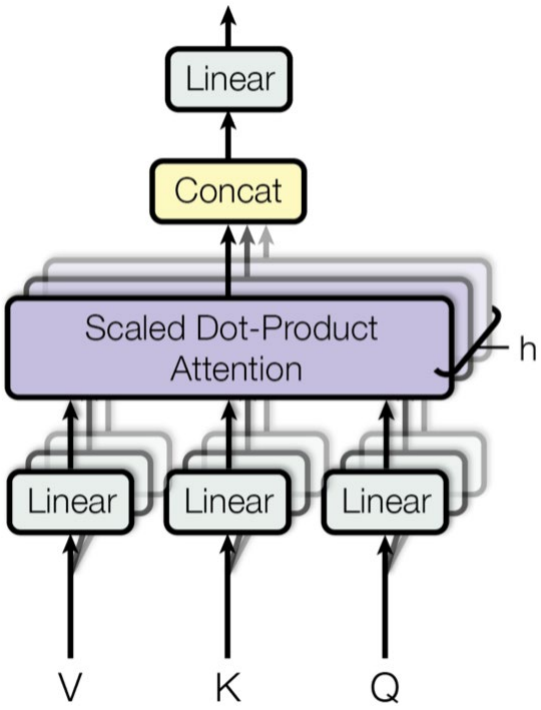


Figure 22: Multi-head scaled dot-product attention mechanism [41]

2.2.4 Evaluation Metrics

Evaluating the performance of a machine learning model is crucial to ensure its effectiveness and reliability. Different types of machine learning models require different evaluation metrics to assess their performance accurately. This section delves into a comprehensive overview of common evaluation metrics used for classification.

Most metrics rely on the definition of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) which are terms used to evaluate the performance of a machine learning model in distinguishing between multiple distinct categories.

- **True Positive (TP)**
  A true positive occurs when the model correctly predicts that an instance belongs to a particular class (positive class). For instance, in a multiclass classification task of identifying different types of flowers, a TP would be a case where the model correctly identifies a rose as a rose.
- **True Negative (TN)**
  A true negative occurs when the model correctly predicts that an instance does not belong to a particular class (negative class). In the flower classification example, a TN would be a case where the model correctly identifies a sunflower as not being a rose.
- **False Positive (FP)**
  A false positive occurs when the model incorrectly predicts that an instance belongs to a particular class (positive class) when it actually belongs to a different class. In the flower classification scenario, an FP would be a case where the model mistakenly identifies a tulip as a rose.
- **False Negative (FN)**
  A false negative occurs when the model incorrectly predicts that an instance does not belong to a particular class (negative class) when it actually belongs to that class. In the flower classification example, an FN would be a case where the model wrongly classifies a rose as a tulip.

Some of the most common metrics used for multiclass classification are:

- **Accuracy**
  Accuracy measures the proportion of correct predictions made by the model. It is calculated by dividing the number of correctly predicted instances by the total number of instances.
  $$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Confusion Matrix**

  The confusion matrix shows the number of observations belong to each class in actual data and predictions.

Figure 23: Example of a multiclass confusion matrix [42]

- **Precision**
  Precision measures the proportion of correct positive predictions among all predicted positives. It is calculated by dividing the number of true positives by the sum of true positives and false positives. Precision focuses on avoiding false positives, ensuring the model's high confidence when predicting positive instances.

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

- **Recall**
  Recall measures the proportion of actual positives that were correctly identified by the model. It is calculated by dividing the number of true positives by the sum of true positives and false negatives. Recall focuses on avoiding false negatives, ensuring the model captures the majority of actual positive instances.

$$Recall = \frac{TP}{TP + FN}$$

- **F1 score**

  The F1 score is a harmonic mean of precision and recall, providing a balanced measure of both. It represents the overall classification accuracy considering both the precision and recall aspects. F1 score is particularly useful when both false positives and false negatives have significant consequences.

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

Chapter 3. Related Work

The classification of sleep stages is a critical task in the field of sleep medicine, as it provides valuable insights into a person's sleep patterns and overall health. Accurate and efficient sleep stage classification is essential for diagnosing sleep disorders and optimizing treatment plans. In recent years, there has been a growing interest in applying machine learning techniques to automate and enhance the accuracy of sleep stage classification. This related work section provides a comprehensive overview of the existing research in this domain, highlighting the key approaches, methodologies, and advancements that have contributed to the development of automated sleep stage classification systems.

3.1 Unimodal Approaches

A large part of the literature on automatic sleep stage scoring utilizes only a single modality to perform the classification task. Usually, these approaches perform the classification task based only on raw single-channel EEG timeseries as input.

In Supratak et al., 2017 [3] the authors proposed DeepSleepNet, a model for automatic sleep stage scoring based on raw single-channel EEG. DeepSleepNet utilizes Convolutional Neural Networks to extract time invariant features, and bidirectional-Long Short-Term Memory to learn transition rules among sleep stages automatically from EEG epochs. In order to effectively train the model and to prevent it from suffering due to the class imbalance problem often present in a large sleep dataset. The algorithm first pre-trains the representation learning part of the model and then fine-tunes the whole model using two different learning rates. They evaluated their model using different single-channel EEGs (F4-EOG(Left), Fpz-Cz and Pz-Oz) from two public sleep datasets, that have different properties (e.g., sampling rate) and scoring standards (AASM and R&K). The results showed that DeepSleepNet achieved similar overall accuracy and macro F1-score (MASS [43]: 86.2%-81.7, SleepEDF [44]: 82.0%-76.9) when compared to the contemporary state-of-the-art methods on both datasets. This demonstrated that, without changing the model architecture and the training algorithm, the model could automatically learn features for sleep stage scoring from different raw single-channel EEGs and from different datasets without needing any hand-engineered features.

In Phan et al., 2018 [45] the authors used time-frequency image features extracted from the raw EEG signals after applying STFT and then frequency smoothing them though a learned DNN filter. Their model consists of a simple CNN whose convolutional layer is able to support convolutional kernels with different sizes, and is therefore, capable of learning features at multiple temporal resolutions. In addition, the 1-max pooling strategy is employed at the pooling layer to better capture the shift-invariance property of EEG signals. They demonstrated that their proposed 1-max pooling CNN performs comparably with the very deep CNNs in the literature on the SleepEDF dataset [44] and that by preprocessing the time-frequency image features with the learned filter bank before presenting them to the CNN leads to significant improvements on the classification accuracy.

In Phan et al., 2019 [46] the authors tackle the task of automatic sleep staging as a sequence-to-sequence classification problem that receives a sequence of multiple epochs as input and classifies all of their labels at once. To achieve this, they proposed a hierarchical recurrent neural network named SeqSleepNet. The network consists of a filter bank layer

tailored to learn frequency-domain filters, an attention-based recurrent layer designed for short term sequential modeling and a recurrent layer placed on top of the learned epoch-wise features for long-term modeling of sequential epochs. The classification is then carried out on the output vectors at every time step of the top recurrent layer to produce the sequence of output labels. Their proposed method achieved a macro F1-score of 83.3 on the MASS dataset [43].

In Eldele et al., 2021 [1] the authors proposed an attention-based architecture called AttnSleep to classify sleep stages using single channel EEG signals. The core of the architecture forms a multi-resolution convolutional neural network which is used to extract low and high frequency features. They also use a temporal context encoder that leverages a multi-head attention mechanism to capture the temporal dependencies among the extracted features. AttnSleep achieved a macro F1 score of 78.1 on the SleepEDF-78 dataset [44].

In Phan et al., 2022 [47] the authors proposed SleepTransformer, a sequence-to-sequence sleep-staging model based on the transformer backbone that offers interpretability of the model's decisions at both the epoch and sequence level. They also proposed a simple entropy-based method to quantify uncertainty in the model's decisions. SleepTransformer achieved a macro F1-score of 74.3 on the SleepEDF-78 dataset [44].

3.2 Multivariate and Multimodal Approaches

In Chambon et al., 2018 [48] the authors introduced the first deep learning approach for sleep stage classification, that exploits all multivariate and multimodal polysomnography (PSG) signals (EEG, EMG, and EOG). For each modality, the first layer learns linear spatial filters and the last layer feeds the learnt representation to a softmax classifier. Their study reveals that a good tradeoff for optimal classification performance measured with balanced accuracy is to use 6 EEG with 2 EOG (left and right) and 3 EMG chin channels.

In Jia et al., 2020 [49] the authors proposed SleepPrintNet to capture the SleepPrint in physiological time-series, which represents the complementarity among different features of EEG and discriminative features from other modalities in different sleep stages. In order to achieve this, it considers information from EEG signals in the time-domain, frequency-domain, and spatial-domain, while also having different feature extractors for EEG, EOG and EMG signals. SleepPrintNet thus consists of an EEG temporal feature extraction module, an EEG spectral-spatial feature extraction module for the temporal-spectral-spatial representation of EEG signals, and two multimodal feature extraction modules including EOG and EMG feature extraction module. SleepPrintNet achieved a macro F1-score of 84.3 in the MASS-SS3 dataset [43].

In Duan et al., 2021 [50] the authors proposed an automatic sleep staging network model based on data adaptation and multimodal feature fusion using EEG and electrooculogram (EOG) signals. 3D-CNN is used to extract the time-frequency features of EEG at different time scales, and LSTM is used to learn the frequency evolution of EOG. The nonlinear relationship between the high-level features of EEG and EOG is fitted by a deep probabilistic network. This approach achieved a macro F1-score of 86.0 on the SleepEDF-20 dataset [44].

In Yubo et al., 2022 [51] the authors introduced MMASleepNet a multimodal attention network to extract, perceive and fuse multimodal features of electrophysiological signals.

The MMASleepNet has a multi-branch feature extraction (MBFE) module followed by an attention-based feature fusing (AFF) module. In the MBFE module, branches are designed to extract multimodal signals' temporal and spectral features. Each branch has two-stream convolutional networks with a unique kernel to perceive features of different time scales. The AFF module contains a modal-wise squeeze and excitation (SE) block to adjust the weights of modalities with more discriminative features and a Transformer encoder (TE) to generate attention matrices and extract the inter-dependencies among multimodal features. MMASleepNet achieved a macro F1-score of 77.6 on the SleepEDF-78 dataset [44].

3.3 Self-supervised Approach

In Banville et al., 2021 [52] the authors investigated self-supervised learning (SSL) to learn representations of EEG signals. Specifically, they explored two tasks based on temporal context prediction as well as contrastive predictive coding on two clinically relevant problems: EEG-based sleep staging and pathology detection. Their results suggest that self-supervision may pave the way to a wider use of deep learning models on EEG data, and that linear classifiers trained on SSL-learned features consistently outperformed purely supervised deep neural networks in low-labelled data regimes while reaching competitive performance when all labels were available.

In Xiao et al. 2021 [53] the authors propose SleepDPC, a novel sleep stage classification algorithm based on SSL. By incorporating two dedicated predictive and discriminative learning principles, SleepDPC discovers underlying semantics from raw EEG signals in a more efficient manner. The experimental results show that the proposed SleepDPC method not only learns meaningful representations but also produces superior performance versus various competing methods despite limited access of labeled data.

In Eldele et al., 2023 [54] the authors evaluate the efficacy of SSL to boost the performance of existing SSC models in the few-labels regime. This paper conducts a thorough study on three SSC datasets, and finds that, in many cases, fine-tuning the pretrained SSC models with only 5% of labeled data can achieve competitive performance to the supervised training with full labels.

Chapter 4. Methodology

4.1 Data

4.1.1 Dataset

To evaluate the proposed model in both the full and the reduced-label regimes experiments were conducted using the publicly available SleepEDF dataset [44]. The SleepEDF-78 dataset is the 2018 version of the Sleep-EDF Expanded dataset, it contains 153 Sleep Cassettes which were obtained in a 1987-1991 study of age effects on sleep in healthy Caucasians aged 25-101, without any sleep-related medication. Two PSGs of about 20 hours each were recorded during two subsequent day-night periods at the subjects' homes, except subjects 13, 36, and 52 whose one recording was lost due to device failure. Subjects continued their normal activities but wore a modified Walkman-like cassette-tape recorder. Manual scoring was done by sleep experts according to the R&K standard [7], but based on Fpz-Cz/Pz-Oz EEGs instead of C4-A1/C3-A2 EEGs, as suggested by [55]. Each 30-second PSG epoch was labeled as one of eight categories {W, N1, N2, N3, N4, REM, MOVEMENT, UNKNOWN}.

The SleepEDF-20 dataset is a smaller version of the SleepEDF-78 dataset consisting of 20 subjects aged 25-34.

4.1.2 Data Preprocessing

The dataset was preprocessed following the practices presented in Phan et al., 2022 [2]. To be precise, sleep stages N3 and N4 were merged into a single stage N3, while MOVEMENT and UNKNOWN epochs were ignored. Fpz-Cz EEG and ROC-LOC EOG timeseries sampled at 100 Hz were used from which time-frequency representation, using the log magnitude spectrum from the Short-Time Fourier Transform (STFT), were also extracted. Additionally, the dataset is split into five folds by subject and the experiments are performed using 5-fold subject-wise cross-validation, in order to evaluate the model in a subject-independent manner that offers better insights into the model's ability to generalize [56].

For the purposes of testing the model under a low-label scenario, we extract 5 additional datasets from the preprocessed training data where we maintain only 1, 5, 10, 50 or 75 percent of the original labels.

4.2 Supervised Model

Given a training set $\{X_n\}_{n=1}^{N}$ of size N where $x_i = \{\left(x_{i_{ts}}, x_{i_{tf}}\right), y_i\}$ represents the i-th sample from the training set. In the above training sample $x_{i_{ts}} \in R^{C \times L}$ and $x_{i_{tf}} \in R^{T \times F \times C}$, where $C = 2$ is the number of channels used (EEG and EOG), $L = 3000$ is the length of the timeseries, $T = 29$ is the number of time frames in the time-frequency image and $F = 128$ is the number of frequency bins extracted, represent the 2-channel 30-second raw EEG and EOG timeseries sampled at 100Hz and the time-frequency image extracted from those timeseries, respectively.

The architecture of MMSleepNet is illustrated in Figure 24. The network extracts embeddings from the raw timeseries signal using a Dual Kernel CNN Encoder architecture inspired from the AttnSleep model [1]. The encoder uses a two-branch CNN architecture, with a different kernel size per branch, to extract features from the 30-second EEG and EOG

signals. To be precise, it extracts high-frequency features by utilizing the small kernel network and low-frequency features by performing convolutions with the wide kernel. This multi-resolution approach aims to improve the quality of the extracted features since different sleep stages are characterized by different frequency ranges [57]. The extracted features from the different frequency bands are unified using a multi-head attention mechanism.

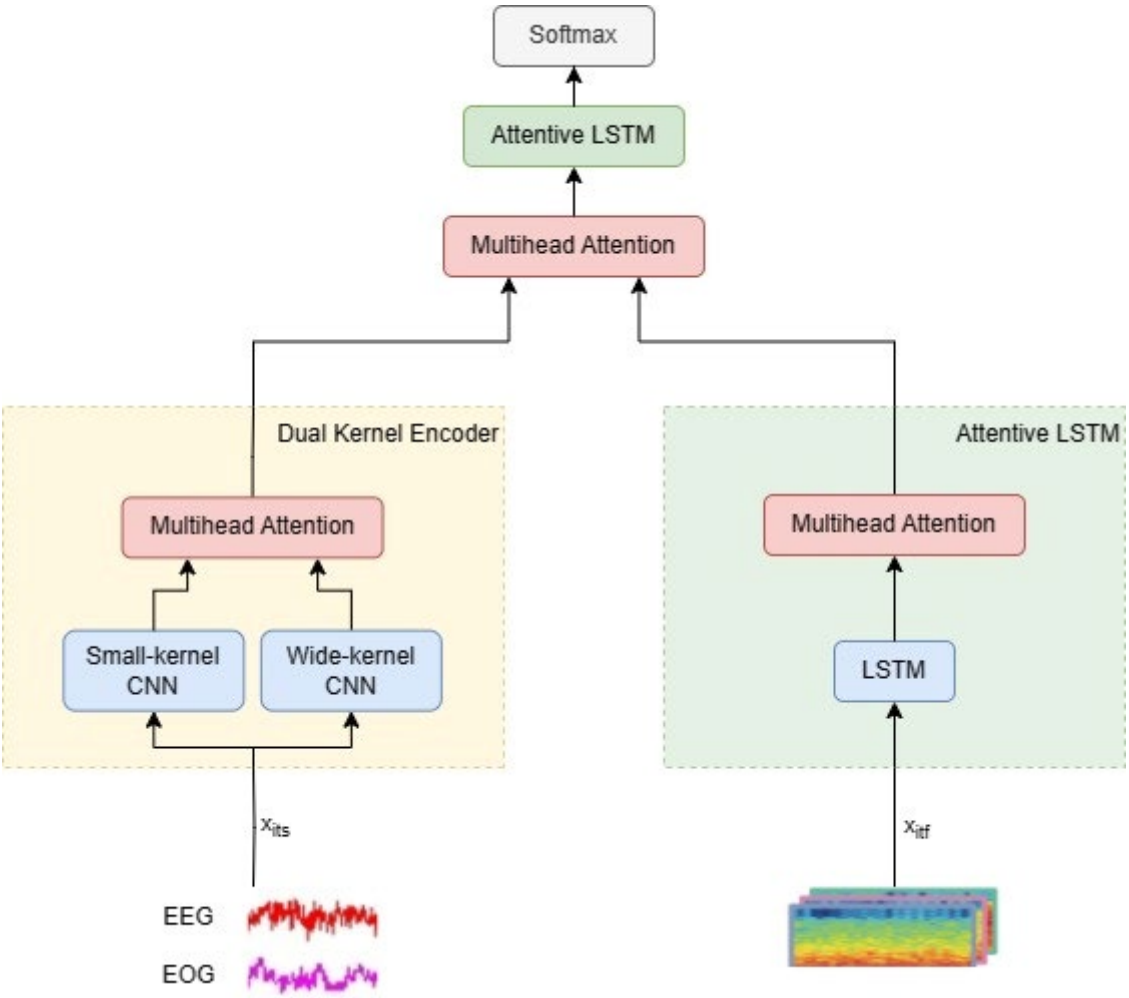


Figure 24: Architecture of MMSleepNet

The time-frequency representations of the signals are handled via an attentive LSTM block which is composed of a bidirectional LSTM encoder with a multi-head attention mechanism being used to improve the generated context vector. This combination addresses the limitations of LSTMs by enhancing the model's ability to capture global dependencies, improving information flow, and enhancing the processing of long sequences.

The two outputs from the raw signal and time-frequency image encoders are then combined into a single representation using multi-head attention and the resulting representation for the 30-second signals is fed to another attentive LSTM block whose output is used to extract the class predicted by the model.

4.3 Self-Supervised Algorithms

To evaluate the performance of the proposed model in low-label scenarios four Self-Supervised Learning (SSL) algorithms which are used to adjust the starting weights of the feature extractors, namely the dual kernel and attentive LSTM encoders, during a pretraining phase using the raw signals in the dataset without utilizing the corresponding labels. Following this, the entire model is fine-tuned using the reduced-label data in an end-to-end manner. Specifically, the weights of the feature extractors are set to those that resulting from the above pretraining with the SSL algorithms and are then frozen for the a few epochs to allow for the randomly initialized weights of the rest of the models to be updated without the potentially large losses leading to large unnecessary updates to the pretrained weights. Following this initial period, the pretrained weights are unfrozen and allowed to be updated with the rest of the model.

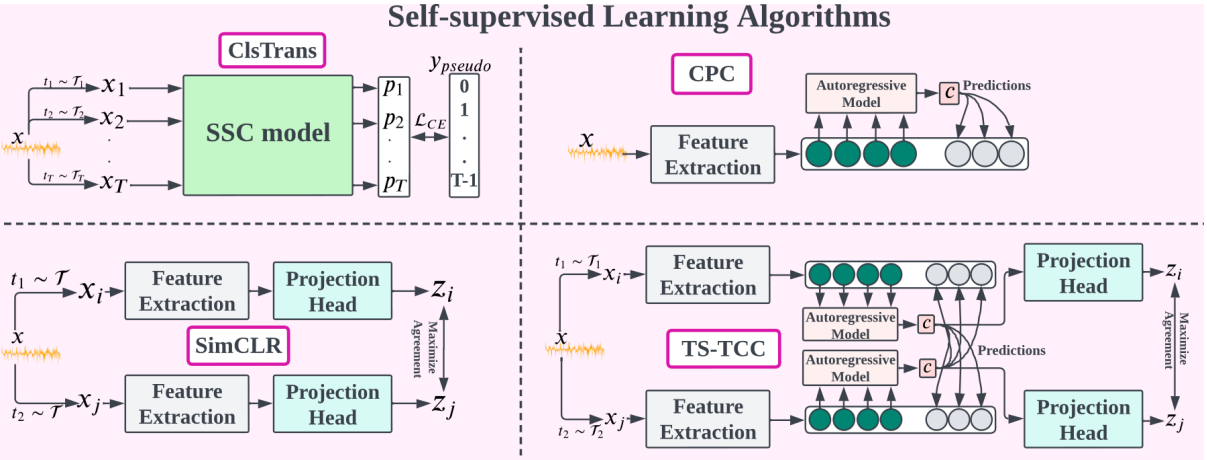


Figure 25: Self-supervised learning algorithms [54]

The four SSL algorithms used are adopted from Eldele et al., 2023 [54]. Namely:

1. **ClsTran:** Classifying Transformations is an auxiliary classification task, in which, first, some transformations are applied to the input signal and then, an automatically generated pseudo label is associated with each transformation. Finally, the model is trained to classify the transformed signals based on these pseudo labels.
2. **SimCLR:** Simple framework for Contrastive Learning of Visual Representation, as it is presented in Chen et al., 2022 [58], is a contrastive SSL algorithm that relies on data augmentations to learn invariant representations. It consists of four major components. The first is data augmentations, which are utilized to generate two correlated views of the same sample. The second is the feature extractor network that transforms the augmented views into latent space. The third is the projection head, which maps the features into a low-dimensional space. The fourth is the NT-Xent loss, which aims to maximize the similarity between an anchor sample with its augmented views while minimizing its similarity with the augmented views of the other samples within the mini-batch.
3. **CPC:** Contrastive Predictive Coding, as it is presented in Oord et al., 2019 [59], is a predictive contrastive SSL approach that learns representations of timeseries signals by predicting the future timesteps in the embedding space. To do so, the feature extractor first generates the latent feature embeddings for the input signals. Next, an autoregressive model receives a part of the embeddings, i.e., the past timesteps,

then generates a context vector and uses it to predict the other part, i.e., the future timesteps. CPC deploys a contrastive loss such that the embedding should be close to positive future embeddings and distant from negative future embeddings. CPC showed improved downstream performance in various time-series and speech recognition-related tasks, without the need for any data augmentation.

4. **TS-TCC:** Time-Series representation learning via Temporal and Contextual Contrasting, as it is presented in Eldele et al., 2021 [60], is yet another contrastive SSL approach for time-series data. TS-TCC relies on strong and weak augmentations to generate two views of an anchor sample. Next, the feature embeddings of these views are generated. Next, like CPC, a part of the embeddings of each view is sent to an autoregressive model to generate a context vector. Then, the context vector generated for one augmented view is used to predict the future timesteps of the other augmented view with a contrastive loss. Therefore, it pushes the embeddings of one augmented view to the positive future embeddings of the other augmented view, and vice versa. In addition, it leverages the NT-Xent loss (Eq. 1) to maximize the agreement between the context vectors of the same sample, while maximizing it within the contexts of other samples.

4.4 Evaluation

To evaluate the performance of the proposed supervised learning model we compare the performance of the model on the SleepEDF-20 dataset with three baseline models. Namely, the baseline models chosen are:

- **AttnSleep** [1]**:** This model starts with a feature extraction module based on multi-resolution convolutional neural network (MRCNN) and adaptive feature recalibration (AFR) that models the inter-dependencies between the features. The second module is the temporal context encoder (TCE) that leverages a multi-head attention mechanism to capture the temporal dependencies among the extracted features.
- **XSleepNet** [2]**:** This model is capable of learning a joint representation from both raw signals and time-frequency images. Since different views may generalize or overfit at different rates, the proposed network is trained such that the learning pace on each view is adapted based on their generalization/overfitting behavior. The Fpz-Cz EEG and ROC-LOC EOG signals were used. Uses a sequence-to-sequence architecture.
- **DeepSleepNet** [3]**:** This model exploits a custom CNN architecture followed by an LSTM with a residual connection for sleep stage classification.

To evaluate the performance of the models on a per-class basis precision (PR), recall (RE) and f1 (F1) metrics are used, while for evaluating the overall performance of the model across all classes the accuracy (Acc), macro-averaged F1-score (MF1) and Cohen's Kappa (κ) [61] metrics are used.

For the full-label scenario, experiments were conducted using the SleepEDF-20 dataset in two common ways found in the literature. Specifically, (1) only in-bed parts of the recordings were used as recommended in [62], (2) 30 minutes of data before and after in-bed parts were included in the experiments following the recommendation in [3], denoted as SleepEDF-20 (±30 min).

For the reduced-label scenario, experiments were conducted using the SleepEDF-20 (±30 min) dataset with only some of the labels being available in the training set used.

Chapter 5. Experiments

5.1 Implementation Details

To extract the time-frequency input as mentioned in 4.1.2 the EEG and EOG timeseries belonging to a 30-second PSG signal segment sampled at 100 Hz where split into two-second windows with 50 percent overlap, weighted by a Hamming window and transformed to the frequency domain using a 256-point Fast Fourier Transform (FFT). The log-amplitude spectrum was then extracted to create the time-frequency image used which was first normalized to have zero mean and unit standard deviation. This produced an image per channel $x \in R^{T \times F}$ with $F = 128$ frequency bins and $T = 29$ time points which were then concatenated to produce a singe $x \in R^{29 \times 256}$ input feature. The preprocessing steps where performed using Matlab following the methodology given in Phan et al., 2022 [2].

The model was developed and tested using PyTorch 2.1.2 and training was executed on a GeForce GTX 1660 Ti GPU. Subject-wise 5-fold cross-validation was used by dividing the subjects of each dataset into 5 groups. A validation set was left out to evaluate our model during training.

During evaluation under the reduced-label scenarios a percentage of the training set samples had their associated labels removed. Additionally, model training is split into three distinct stages. First, the feature encoders of the model, namely the dual kernel encoder and attentive LSTM encoder, are pretrained using the specified SSL algorithm. The entire model is then fine-tuned with the feature extractors starting with the model weights they had received during the previous stage, while the rest of the model is initialized with random weights. During fine-tuning the pretrained feature extractor weights are initially frozen to avoid large loss values caused by the random initialization of the rest of the model resulting in large changes to their values early on in the training process. The pretrained weights are then unfrozen and their learning rate is set to the same as that of the rest of the model. The SSL algorithms are configured following the implementation presented in Eldele et al., 2023 [54].

The duration of training under the fully supervised regime is set to 40 epochs. When training under the reduced-label regime pretraining is done for 40 epochs, fine tuning with frozen pretrained weights for 10 epochs and fine tuning with the whole model unfrozen is done for another 40 epochs.

For all experiments the network weights were optimized using the AdamW optimizer [63] with an initial learning rate of 0.001. Cross-entropy loss was adopted as the training objective and a minibatch size of 128 was used during training.

5.2 Results

5.2.1 Full-Label Scenario

The confusion matrices that demonstrate the performance of MMSleepNet on the SleepEDF-20 and SleepEDF-20 (±30 min) datasets are shown in Figure 26 and Figure 27 respectively. Rows and columns represent the ground truth and predicted results respectively for each class. The numbers in the diagonal thus represent the correctly classified samples, while off-diagonal entries represent wrong predictions.

From the given figures, it can be observed that while most samples are correctly classified the model has the most difficulty in identifying correctly the N1 stage, which has the fewest samples. From comparing Figure 26 with Figure 27 it can be noted that the addition of 30 minutes of wake time before and after sleep results in better overall classification performance for most classes and especially so for the W class. This is to be expected as the additional samples added to the awake stage repair some of the class imbalances present in this dataset and allow the model to be much more accurate in identifying the W stage.

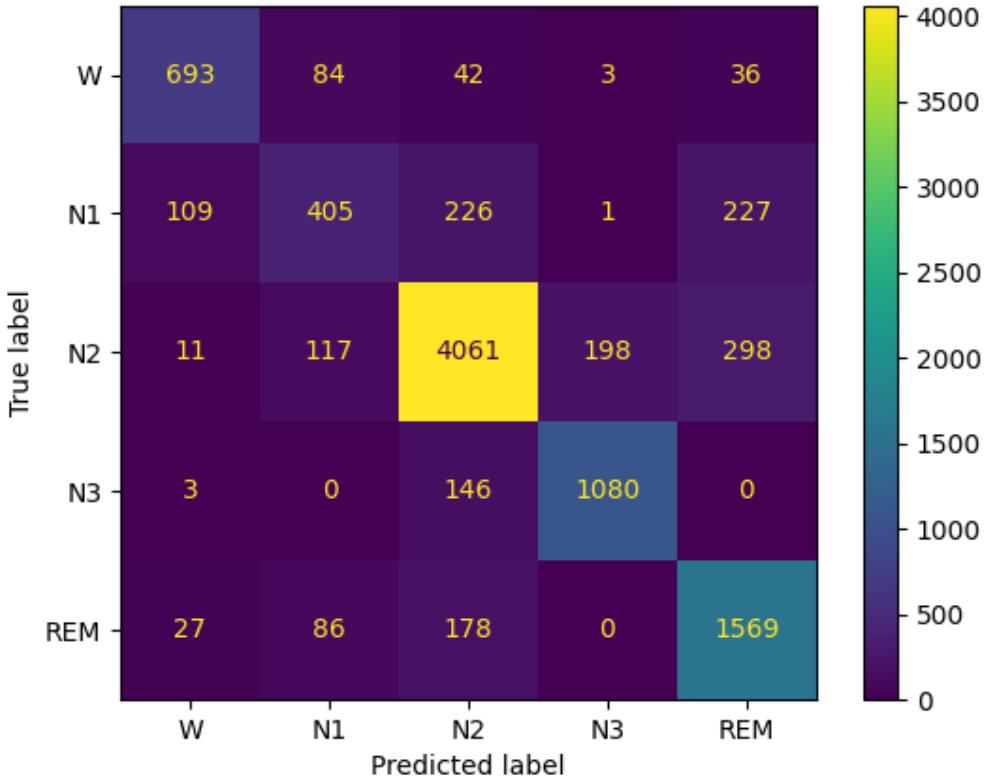


Figure 26: Confusion Matrix on SleepEDF-20 Dataset

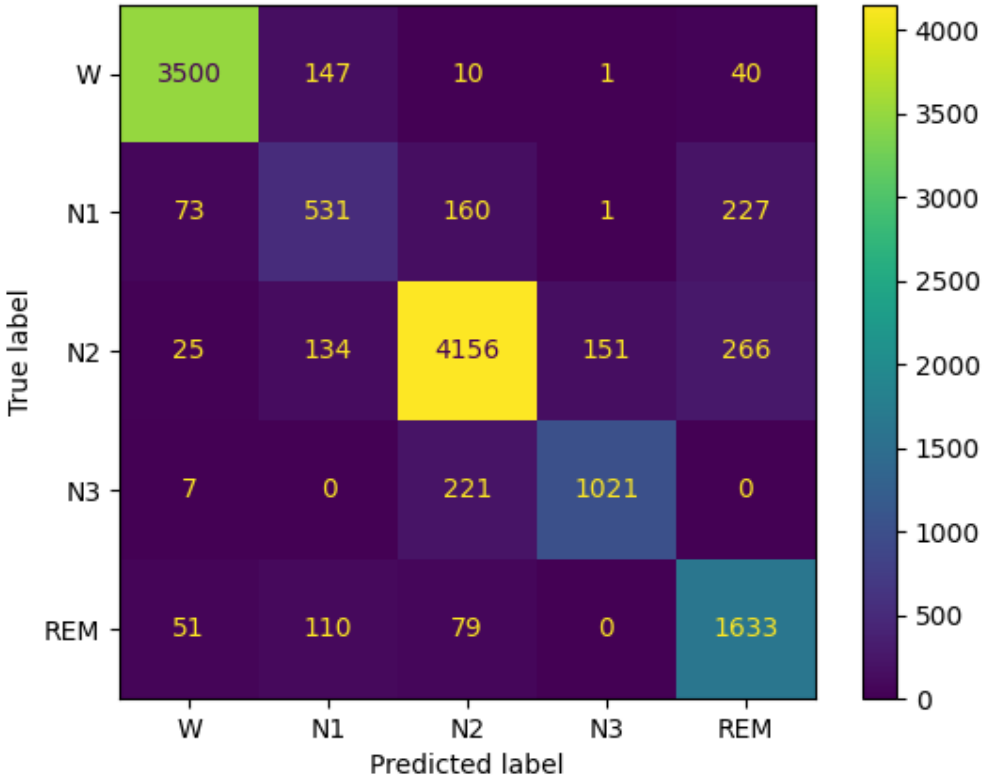


Figure 27: Confusion Matrix on SleepEDF-20 (±30 min) Dataset

The performance of the model across three evaluation metrics, namely precision (PR), recall (RE) and F1-score (F1), on the SleepEDF-20 and SleepEDF-20 (±30 min) datasets are shown in Table 1 and Table 2 respectively. In Table 2 we can also see the comparison between MMSleepNet, AttnSleep [1], DeepSleepNet [3].

From the provided tables, it can be observed that, especially for the W stage, the addition of 30 minutes of extra wake time in SleepEDF-20 (±30 min) results in significantly better results across most metrics. Furthermore, MMSleepNet seems to outperform AttnSleep [1] and DeepSleepNet [3] particularly when it comes to classification of the W and N1 stages.

Table 1: Per-class Performance Metrics on SleepEDF-20 Dataset

	MMSleepNet		
	PR	*RE*	*F1*
W	81.9	83.2	81.9
N1	54.3	39.1	44.4
N2	86.9	87.6	86.6
N3	82.5	86.0	83.1
REM	76.8	83.9	79.2

Table 2: Per-class Performance Metrics on SleepEDF-20 (±30 min) Dataset

	MMSleepNet			AttnSleep [1]			DeepSleepNet [3]		
	PR	RE	F1	PR	RE	F1	PR	RE	F1
W	**94.3**	**92.0**	**93.0**	89.6	89.7	89.7	86.0	83.4	84.7
N1	**57.9**	**51.5**	**53.4**	47.1	39.1	42.8	43.5	50.1	46.6
N2	89.3	88.2	88.6	89.1	**88.6**	**88.8**	**90.5**	81.7	85.9
N3	**85.9**	81.2	82.1	80.7	89.8	**90.2**	77.1	**94.2**	84.8
REM	76.1	**87.3**	81.0	76.1	82.2	79.0	**80.9**	83.9	**82.4**

Table 3 compares MMSleepNet with methods developed in previous studies. The evaluation of overall performance is done by accuracy (Acc), Cohen's kappa (κ), and macro F1-score (MF1), and evaluating the performance of each class with F1-score. The results show that MMSleepNet outperforms AttnSleep [1] and DeepSleepNet [3], while achieving similar results to XSleepNet [2].

Table 3: Performance Comparison with Previous Methods

		Overall metrics			Per-class F1-Score				
Dataset	Model	Acc	κ	MF1	W	N1	N2	N3	N4
SleepEDF-20	**MMSleepNet**	81.6	0.73	75.1	81.9	44.4	86.6	83.1	79.2
	AttnSleep [1]	-	-	-	-	-	-	-	-
	XSleepNet [2]	**83.3**	**0.76**	**77.3**	-	-	-	-	-
	DeepSleepNet [3]	80.8	0.74	74.2	-	-	-	-	-
SleepEDF-20 (±30 min)	**MMSleepNet**	85.7	**0.82**	79.6	**93.0**	**53.4**	88.6	82.1	81.0
	AttnSleep [1]	84.4	0.79	78.1	89.7	42.6	**88.8**	**90.2**	79.0
	XSleepNet [2]	**86.4**	0.81	**80.9**	-	-	-	-	-
	DeepSleepNet [3]	81.9	0.76	76.6	86.7	45.5	85.1	83.3	**82.6**

5.2.2 Reduced-Label Scenario

The confusion matrix that demonstrates the performance of MMSleepNet on the SleepEDF-20 (±30 min) dataset under a reduced-label scenario where only 10% of the labels were left in the training set is shown in Figure 28. From the given figure it can be observed that the supervised model's performance has deteriorated significantly when compared to the corresponding full-label scenario presented in Figure 27.

The confusion matrices that result from the fine-tuned supervised model following pretraining with the various SSL algorithms tested are shown in Figure 29. From the produced matrices it can be observed that SSL pretraining resulted in significant changes in model performance across most classes and algorithms. A particularly interesting result is that all SSL algorithms seem to have produced worst results when it comes to accuracy for the least represented classes in the dataset, namely N1 and N3, when compared to the supervised model without pretraining. However, most of them demonstrate improved accuracy in classifying samples from the most highly represented classes, namely W, REM and N3.

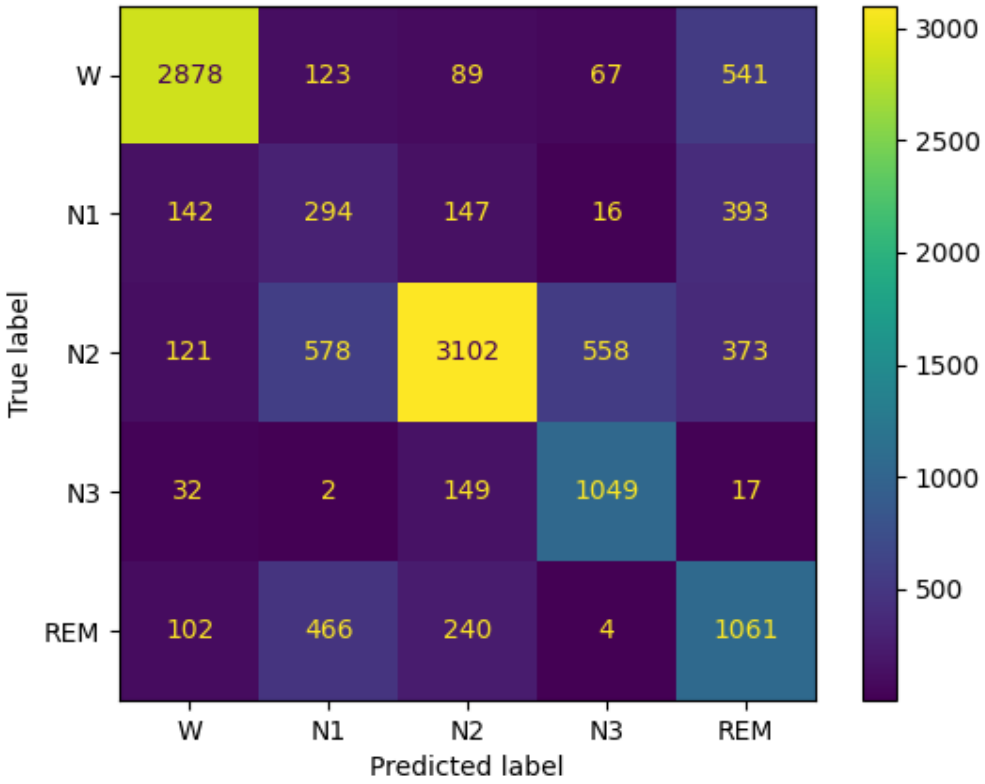


Figure 28: Confusion Matrix of supervised model on SleepEDF-20 (±30 min) Dataset With 10% of the Labels

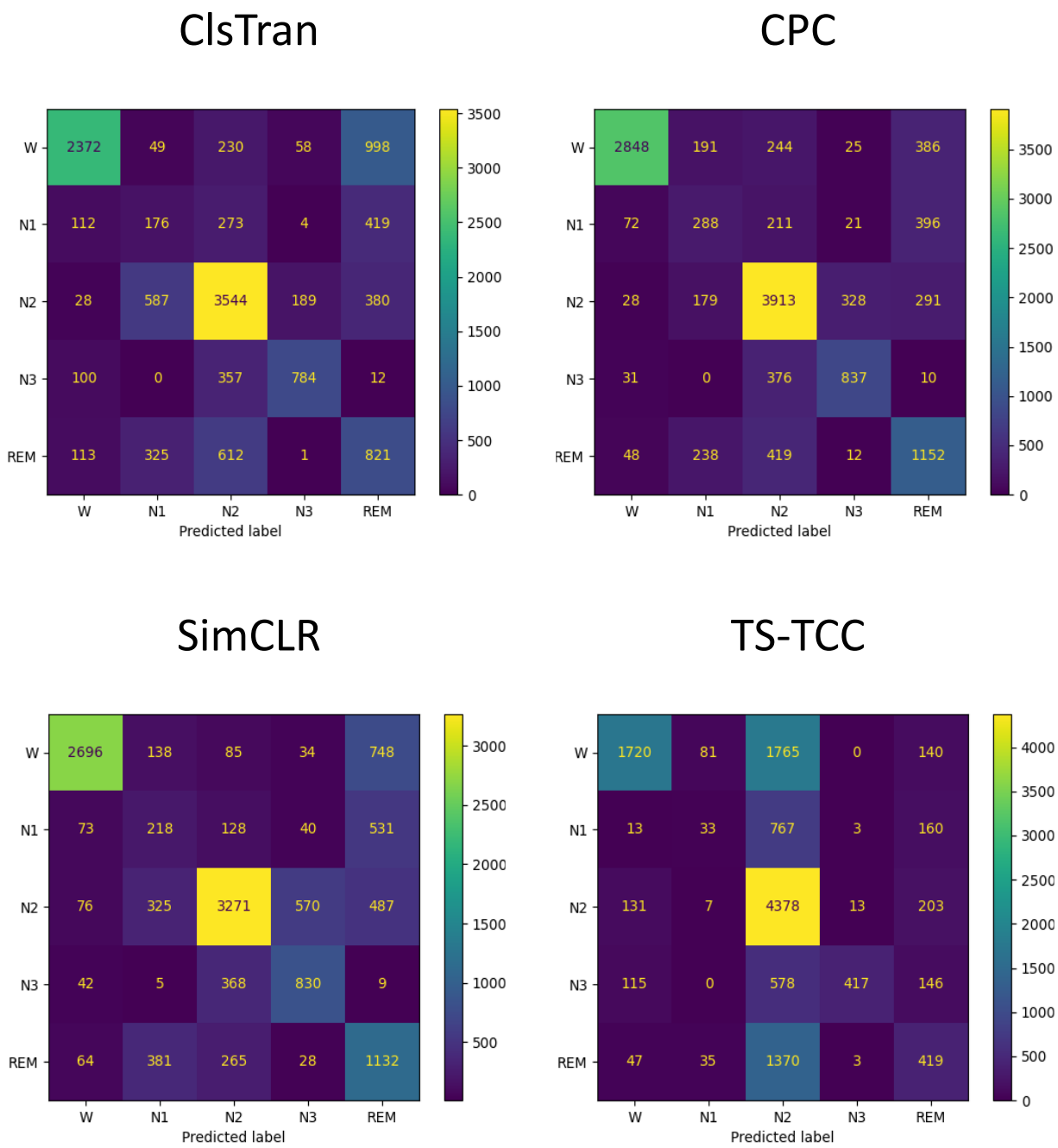


Figure 29: Confusion Matrix after pretraining MMSleepNet with various SSL algorithms on SleepEDF-20 (±30 min) Dataset With 10% of the Labels

In Table 4 performance in both the overall metrics chosen for evaluation and the per-class F1-score is compared between the supervised model without pretraining and the fine-tuned model following pretraining with the corresponding SSL algorithm. In all cases the SleepEDF-20 (±30 min) dataset with 10% of the labels remaining was used.

From the provided table it can be noted that pretraining has significant influence on the end results of the supervised model across most metrics. From the per-class results it can be observed that the influence of the SSL algorithm pretraining affects the classification results of all classes.

The results demonstrate that the choice of SSL algorithm is of monumental importance. When the wrong algorithm for the problem is used, it can lead to significant deterioration of model performance, while when a more appropriate algorithm is applied it can result in important improvements to the predictive capabilities of the resulting model.

On this reduced-label dataset the best results are yielded when pretraining with the CPC SSL algorithm. The possible reason for this result is that this algorithm relies on predicting the future timesteps in the latent space, which allows the feature encoders to learn about temporal features in the EEG and EOG data which are useful for improving classification performance in this particular problem.

Table 4: Performance of fine-tuned MMSleepNet after pretraining with various SSL algorithms on SleepEDF-20 (±30 min) Dataset With 10% of the Labels

	Overall metrics			Per-class F1-Score				
Algorithm	Acc	κ	MF1	W	N1	N2	N3	REM
Supervised	65.7	0.56	**57.4**	82.5	23.9	73.3	**71.3**	49.8
ClsTran	60.6	0.48	49.9	73.8	16.6	72.7	68.5	36.5
SimCLR	63.2	0.53	54.8	81.1	21.2	74.0	60.2	47.4
CPC	**71.4**	**0.62**	**57.4**	**84.7**	**30.6**	**79.0**	67.6	**56.1**
TS-TCC	56.4	0.34	31.6	60.0	5.8	64.4	49.2	28.5

Chapter 6. Conclusions

6.1 Discussion

In this diploma thesis we proposed MMSleepNet for automatic sleep stage classification based on multichannel PSG signals including EEG and EOG in both the time and frequency domain. The model consists of two feature extraction modules, one for extracting features from the timeseries signals and one for the time-frequency images. The former consists of a dual kernel CNN encoder which utilizes two CNNs with different kernel sizes to extract information from the signals at different frequency ranges and then fuses the results to produce a single embedding that encapsulates information across the frequencies examined. The latter is comprised of an attentive LSTM which extracts information directly from the time-frequency images by considering every column of the image as the input vector for the corresponding time step. The resulting embeddings from the feature extractors are then fused and propagated to another attentive LSTM which produces a single prediction for the 30-second input signals. The proposed model demonstrated promising results across various evaluation metrics when compared to previous methods using either single-channel or multi-channel inputs on the SleepEDF-20 dataset.

Additionally, we evaluated the performance of the proposed model under a low-label scenario and performed pretraining of the model with various self-supervised algorithms to examine if this would lead to improvements in the resulting performance when the now pretrained model was fine-tuned with the few labeled samples in the dataset. We demonstrated that many of the self-supervised algorithms used show promise in improving the performance of the model in this label-scarce scenario.

6.2 Future Work

There are several ideas whose potential can be further explored for improving automatic sleep stage classification performance:

- Different methods for feature fusion both across the time and frequency modalities used in the model and within the dual kernel encoder module could be explored.
- Additional PSG channels, such as Pz-Oz EEG, could be added to test if they will further increase classification performance.
- The performance of the proposed model could be tested on additional sleep stage classification datasets.
- Preprocessing methods could be tried that attempt to correct for the inherent class imbalance present in the dataset.
- Additional self-supervised learning algorithms could be evaluated.

References

[1] E. Eldele *et al.*, “An Attention-Based Deep Learning Approach for Sleep Stage Classification With Single-Channel EEG,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 809–818, 2021, doi: 10.1109/TNSRE.2021.3076234.

[2] H. Phan, O. Y. Chén, M. C. Tran, P. Koch, A. Mertins, and M. De Vos, “XSleepNet: Multi-View Sequential Model for Automatic Sleep Staging,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5903–5915, Sep. 2022, doi: 10.1109/TPAMI.2021.3070057.

[3] A. Supratak, H. Dong, C. Wu, and Y. Guo, “DeepSleepNet: a Model for Automatic Sleep Stage Scoring based on Raw Single-Channel EEG,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 11, pp. 1998–2008, Nov. 2017, doi: 10.1109/TNSRE.2017.2721116.

[4] A. N. Goldstein and M. P. Walker, “The Role of Sleep in Emotional Brain Function,” *Annu. Rev. Clin. Psychol.*, vol. 10, pp. 679–708, 2014, doi: 10.1146/annurev-clinpsy-032813-153716.

[5] A. GAUR, A. KALIAPPAN, Y. BALAN, V. SAKTHIVADIVEL, K. MEDALA, and M. UMESH, “Sleep and Alzheimer: The Link,” *Mædica*, vol. 17, no. 1, pp. 177–185, Mar. 2022, doi: 10.26574/maedica.2022.17.1.177.

[6] S.-T. Pan, C.-E. Kuo, J.-H. Zeng, and S.-F. Liang, “A transition-constrained discrete hidden Markov model for automatic sleep staging,” *Biomed. Eng. Online*, vol. 11, p. 52, Aug. 2012, doi: 10.1186/1475-925X-11-52.

[7] A. Kales and A. Rechtschaffen, *A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects*. in NIH publication. Washington, DC: United States Government Printing Office, 1968.

[8] “AASM Scoring Manual - American Academy of Sleep Medicine,” American Academy of Sleep Medicine – Association for Sleep Clinicians and Researchers. Accessed: Dec. 25, 2023. [Online]. Available: https://aasm.org/clinical-resources/scoring-manual/

[9] “Polysomnography,” *Wikipedia*. Dec. 03, 2023. Accessed: Jan. 28, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Polysomnography&oldid=1188055118

[10] J. Malik, Y.-L. Lo, and H.-T. Wu, “Sleep-wake classification via quantifying heart rate variability by convolutional neural network,” *Physiol. Meas.*, vol. 39, no. 8, p. 085004, Aug. 2018, doi: 10.1088/1361-6579/aad5a9.

[11] I. Feinberg and T. C. Floyd, “Systematic trends across the night in human sleep cycles,” *Psychophysiology*, vol. 16, no. 3, pp. 283–291, May 1979, doi: 10.1111/j.1469-8986.1979.tb02991.x.

[12] A. K. Patel, V. Reddy, K. R. Shumway, and J. F. Araujo, “Physiology, Sleep Stages,” in *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2024. Accessed: Jan. 28, 2024. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK526132/

[13] B. Varga, A. Gergely, Á. Galambos, and A. Kis, “Heart Rate and Heart Rate Variability during Sleep in Family Dogs (Canis familiaris). Moderate Effect of Pre-Sleep Emotions,” *Anim. Open Access J. MDPI*, vol. 8, no. 7, p. 107, Jul. 2018, doi: 10.3390/ani8070107.

[14] J. W. Antony, M. Schönauer, B. P. Staresina, and S. A. Cairney, “Sleep Spindles and Memory Reprocessing,” *Trends Neurosci.*, vol. 42, no. 1, pp. 1–3, Jan. 2019, doi: 10.1016/j.tins.2018.09.012.

[15] M. H. Gandhi and P. D. Emmady, “Physiology, K Complex,” in *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2024. Accessed: Jan. 28, 2024. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK557469/

[16] C. J. Hilditch and A. W. McHill, “Sleep inertia: current insights,” *Nat. Sci. Sleep*, vol. 11, pp. 155–165, 2019, doi: 10.2147/NSS.S188911.

[17] H. M. El Shakankiry, “Sleep physiology and sleep disorders in childhood,” *Nat. Sci. Sleep*, vol. 3, pp. 101–114, 2011, doi: 10.2147/NSS.S22839.

[18] C. Della Monica, S. Johnsen, G. Atzori, J. A. Groeger, and D.-J. Dijk, “Rapid Eye Movement Sleep, Sleep Continuity and Slow Wave Sleep as Predictors of Cognition, Mood, and Subjective Sleep Quality in Healthy Men and Women, Aged 20-84 Years,” *Front. Psychiatry*, vol. 9, p. 255, 2018, doi: 10.3389/fpsyt.2018.00255.

[19] J. Peever and P. M. Fuller, “The Biology of REM Sleep,” *Curr. Biol. CB*, vol. 27, no. 22, pp. R1237–R1248, Nov. 2017, doi: 10.1016/j.cub.2017.10.026.

[20] “5 Stages of Sleep: Psychology, Cycle & Sequence.” Accessed: Jan. 28, 2024. [Online]. Available: https://www.simplypsychology.org/sleep-stages.html

[21] T. M. Mitchell, *Machine Learning*, 1st edition. New York: McGraw-Hill Education, 1997.

[22] “WikiMili, The Free Encyclopedia,” WikiMili.com. Accessed: Feb. 04, 2024. [Online]. Available: https://wikimili.com

[23] S. Bhatt, “Reinforcement Learning 101,” Medium. Accessed: Oct. 15, 2023. [Online]. Available: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

[24] “Part 2: Kinds of RL Algorithms — Spinning Up documentation.” Accessed: Oct. 15, 2023. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

[25] “Deep learning,” *Wikipedia*. Oct. 07, 2023. Accessed: Oct. 15, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Deep_learning&oldid=1179017520

[26] “Overfitting vs. Underfitting: What Is the Difference?,” 365 Data Science. Accessed: Oct. 20, 2023. [Online]. Available: https://365datascience.com/tutorials/machine-learning-tutorials/overfitting-underfitting/

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

[28] “Activation Functions in Neural Networks [12 Types & Use Cases].” Accessed: Oct. 20, 2023. [Online]. Available: https://www.v7labs.com/blog/neural-networks-activation-functions, https://www.v7labs.com/blog/neural-networks-activation-functions

[29] “5 Concepts You Should Know About Gradient Descent and Cost Function,” KDnuggets. Accessed: Oct. 20, 2023. [Online]. Available: https://www.kdnuggets.com/5-concepts-you-should-know-about-gradient-descent-and-cost-function.html

[30] “What are Neural Networks? | IBM.” Accessed: Oct. 15, 2023. [Online]. Available: https://www.ibm.com/topics/neural-networks

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[32] “Deep Learning: Classifying Astronomical Images — AstroML Interactive Book.” Accessed: Oct. 16, 2023. [Online]. Available: https://www.astroml.org/astroML-notebooks/chapter9/astroml_chapter9_Deep_Learning_Classifying_Astronomical_Images.html

[33] “CS231n Convolutional Neural Networks for Visual Recognition.” Accessed: Oct. 16, 2023. [Online]. Available: https://cs231n.github.io/convolutional-networks/

[34] “What are Convolutional Neural Networks? | IBM.” Accessed: Oct. 16, 2023. [Online]. Available: https://www.ibm.com/topics/convolutional-neural-networks

[35] M. Mishra, “Convolutional Neural Networks, Explained,” Medium. Accessed: Oct. 16, 2023. [Online]. Available: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[36] J. Nabi, “Recurrent Neural Networks (RNNs),” Medium. Accessed: Oct. 17, 2023. [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

[37] S. Hochreiter and J. Schmidhuber, “Long Short-term Memory,” *Neural Comput.*, vol. 9, pp. 1735–80, Dec. 1997, doi: 10.1162/neco.1997.9.8.1735.

[38] “Understanding LSTM Networks -- colah’s blog.” Accessed: Oct. 17, 2023. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[39] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate.” arXiv, May 19, 2016. doi: 10.48550/arXiv.1409.0473.

[40] N. Adaloglou, “How Attention works in Deep Learning: understanding the attention mechanism in sequence models,” AI Summer. Accessed: Oct. 17, 2023. [Online]. Available: https://theaisummer.com/attention/

[41] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: Oct. 17, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[42] “Which metrics are used to evaluate a multiclass classification model’s performance?” Accessed: Jan. 28, 2024. [Online]. Available:

https://www.pi.exchange/knowledgehub/metrics-to-consider-when-evaluating-a-multiclass-classification-models-performance

[43] C. O'Reilly, N. Gosselin, J. Carrier, and T. Nielsen, "Montreal Archive of Sleep Studies: an open-access resource for instrument benchmarking and exploratory research," *J. Sleep Res.*, vol. 23, no. 6, pp. 628–635, Dec. 2014, doi: 10.1111/jsr.12169.

[44] B. Kemp, Aeilko Zwinderman, B. Tuk, H. Kamphuisen, and J. Oberyé, "The Sleep-EDF Database [Expanded]." physionet.org, 2018. doi: 10.13026/C2X676.

[45] H. Phan, F. Andreotti, N. Cooray, O. Y. Chèn, and M. De Vos, "DNN Filter Bank Improves 1-Max Pooling CNN for Single-Channel EEG Automatic Sleep Stage Classification," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Jul. 2018, pp. 453–456. doi: 10.1109/EMBC.2018.8512286.

[46] H. Phan, F. Andreotti, N. Cooray, O. Y. Chén, and M. De Vos, "SeqSleepNet: End-to-End Hierarchical Recurrent Neural Network for Sequence-to-Sequence Automatic Sleep Staging," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 3, pp. 400–410, Mar. 2019, doi: 10.1109/TNSRE.2019.2896659.

[47] H. Phan, K. Mikkelsen, O. Y. Chén, P. Koch, A. Mertins, and M. De Vos, "SleepTransformer: Automatic Sleep Staging With Interpretability and Uncertainty Quantification," *IEEE Trans. Biomed. Eng.*, vol. 69, no. 8, pp. 2456–2467, Aug. 2022, doi: 10.1109/TBME.2022.3147187.

[48] S. Chambon, M. N. Galtier, P. J. Arnal, G. Wainrib, and A. Gramfort, "A Deep Learning Architecture for Temporal Sleep Stage Classification Using Multivariate and Multimodal Time Series," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 26, no. 4, pp. 758–769, Apr. 2018, doi: 10.1109/TNSRE.2018.2813138.

[49] Z. Jia, X. Cai, G. Zheng, J. Wang, and Y. Lin, "SleepPrintNet: A Multivariate Multimodal Neural Network Based on Physiological Time-Series for Automatic Sleep Staging," *IEEE Trans. Artif. Intell.*, vol. 1, no. 3, pp. 248–257, Dec. 2020, doi: 10.1109/TAI.2021.3060350.

[50] L. Duan *et al.*, "A Novel Sleep Staging Network Based on Data Adaptation and Multimodal Fusion," *Front. Hum. Neurosci.*, vol. 15, 2021, Accessed: Sep. 13, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnhum.2021.727139

[51] Z. Yubo, L. Yingying, Z. Bing, Z. Lin, and L. Lei, "MMASleepNet: A multimodal attention network based on electrophysiological signals for automatic sleep staging," *Front. Neurosci.*, vol. 16, 2022, Accessed: Sep. 12, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2022.973761

[52] H. Banville, O. Chehab, A. Hyvärinen, D.-A. Engemann, and A. Gramfort, "Uncovering the structure of clinical EEG signals with self-supervised learning," *J. Neural Eng.*, vol. 18, no. 4, p. 046020, Aug. 2021, doi: 10.1088/1741-2552/abca18.

[53] Q. Xiao *et al.*, "Self-Supervised Learning for Sleep Stage Classification with Predictive and Discriminative Contrastive Coding," *ICASSP 2021 - 2021 IEEE Int. Conf. Acoust. Speech Signal Process. ICASSP*, pp. 1290–1294, Jun. 2021, doi: 10.1109/ICASSP39728.2021.9414752.

[54] E. Eldele, M. Ragab, Z. Chen, M. Wu, C.-K. Kwoh, and X. Li, “Self-Supervised Learning for Label- Efficient Sleep Stage Classification: A Comprehensive Evaluation,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 31, pp. 1333–1342, 2023, doi: 10.1109/TNSRE.2023.3245285.

[55] B. van Sweden, B. Kemp, H. A. Kamphuisen, and E. A. Van der Velde, “Alternative electrode placement in (automatic) sleep scoring (Fpz-Cz/Pz-Oz versus C4-A1),” *Sleep*, vol. 13, no. 3, pp. 279–283, Jun. 1990, doi: 10.1093/sleep/13.3.279.

[56] I. Tougui, A. Jilbab, and J. El Mhamdi, “Impact of the Choice of Cross-Validation Techniques on the Results of Machine Learning-Based Diagnostic Applications,” *Healthc. Inform. Res.*, vol. 27, no. 3, pp. 189–199, Jul. 2021, doi: 10.4258/hir.2021.27.3.189.

[57] P. Memar and F. Faradji, “A Novel Multi-Class EEG-Based Sleep Stage Classification System,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 26, no. 1, pp. 84–95, Jan. 2018, doi: 10.1109/TNSRE.2017.2776149.

[58] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A Simple Framework for Contrastive Learning of Visual Representations.” arXiv, Jun. 30, 2020. doi: 10.48550/arXiv.2002.05709.

[59] A. van den Oord, Y. Li, and O. Vinyals, “Representation Learning with Contrastive Predictive Coding.” arXiv, Jan. 22, 2019. doi: 10.48550/arXiv.1807.03748.

[60] E. Eldele *et al.*, “Time-Series Representation Learning via Temporal and Contextual Contrasting.” arXiv, Jun. 26, 2021. doi: 10.48550/arXiv.2106.14112.

[61] J. Cohen, “A Coefficient of Agreement for Nominal Scales,” *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, Apr. 1960, doi: 10.1177/001316446002000104.

[62] S. A. Imtiaz and E. Rodriguez-Villegas, “Recommendations for performance assessment of automatic sleep staging algorithms,” *Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Int. Conf.*, vol. 2014, pp. 5044–5047, 2014, doi: 10.1109/EMBC.2014.6944758.

[63] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization.” arXiv, Jan. 04, 2019. doi: 10.48550/arXiv.1711.05101.