NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF BIOMEDICAL ENGINEERING LABORATORY

Discrimination of real and imaginary lower body movement: a Deep Learning approach

DIPLOMA THESIS

of

CHRISTINA MANARA

Supervisor: Prof. George Matsopoulos
Professor

This thesis is submitted in partial fulfilment for M.Sc. Data Science & Machine Learning

Athens, March 2024

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Biomedical Engineering Laboratory

Discrimination of real and imaginary lower body movement: a Deep Learning approach

Diploma Thesis

of

CHRISTINA MANARA

Supervisor: Prof. George Matsopoulos
Professor

This thesis is submitted in partial fulfilment for M.Sc. Data Science & Machine Learning

Approved by the examination committee on 1st March 2024.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Prof. George Matsopoulos	Panagiotis Tsanakas	Georgios Stamou
Professor	Professor	Professor

Athens, March 2024

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Biomedical Engineering Laboratory

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

.....
Christina Manara
1st March 2024

Abstract

This study introduces a cutting-edge method for analyzing topographical maps derived from electroencephalogram (EEG) data to classify leg movements. Leveraging the spatial information encoded in EEG topographic maps, we propose a hybrid model combining Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs). This approach is designed to extract and integrate spatial features from the topographic maps and temporal dynamics of EEG signals, respectively. By applying preprocessig techniques (data augmentation, ensemble method for dataset imbalance etc), enhancing the model's ability to capture the nuanced patterns associated with different leg movements. Different optimizers, such as Adam, RMSprop & SGD, with different parameters, are performed in order to detect the best model's performance. Preliminary results show the model's efficacy in differentiating between specific leg movement tasks, indicating its potential utility in neurorehabilitation and brain-computer interface applications. Our research highlights the significance of advanced signal processing and machine learning techniques in interpreting complex brain signals, suggesting avenues for further exploration in optimizing model architecture and improving real-time prediction capabilities.

Keywords

Topographical brain maps, R3DCNN, Deep Learning, Optimizers, Data augmentation

Abstract

Η παρούσα διπλωματική εργασία επιχειρεί να αναλύσει τοπογραφικούς χάρτες που προέρχονται από δεδομένα ηλεκτροεγκεφαλογραφήματος (EEG) για την ταξινόμηση των κινήσεων των κάτω άκρων. Εκμεταλλευόμενοι τις χωρικές πληροφορίες που κωδικοποιούνται στους τοπογραφικούς χάρτες EEG, προτείνουμε ένα υβριδικό μοντέλο που συνδυάζει τα Δίκτυα Συνέλιξης (CNNs) με τα Επαναληπτικά Νευρωνικά Δίκτυα (RNNs). Αυτή η προσέγγιση σχεδιάστηκε για να εξάγει και να ενσωματώσει χωρικά χαρακτηριστικά από τους χάρτες και τις χρονικές δυναμικές των σημάτων EEG, αντίστοιχα. Εφαρμόζοντας τεχνικές προεπεξεργασίας, βελτιώνεται η ικανότητα του μοντέλου να αναγνωρίζει τα διακριτικά μοτίβα που σχετίζονται με διάφορες κινήσεις των ποδιών. Διαφορετικοί βελτιστοποιητές, όπως ο Adam, ο RMSprop και ο SGD, δοκιμάζονται με διάφορες παραμέτρους για να εντοπιστεί η καλύτερη απόδοση του μοντέλου. Τα προκαταρκτικά αποτελέσματα δείχνουν την αποτελεσματικότητα του μοντέλου στη διάκριση μεταξύ συγκεκριμένων κινητικών εργασιών, υποδεικνύοντας τη δυνατότητά του για χρήση σε εφαρμογές νευροαποκατάστασης και διεπαφές εγκεφάλου-υπολογιστή. Η έρευνά μας τονίζει τη σημασία των προηγμένων τεχνικών επεξεργασίας σημάτων και μηχανικής μάθησης στην ερμηνεία περίπλοκων εγκεφαλικών σημάτων, προτείνοντας δρόμους για περαιτέρω εξερεύνηση στη βελτιστοποίηση της αρχιτεκτονικής του μοντέλου και τη βελτίωση των δυνατοτήτων πραγματικού χρόνου πρόβλεψης.

Λέξεις Κλειδιά

Τοπογραφικοί θερμικοί χάρτες εγκεφάλου, R3DCNN, Deep Learning, Optimizers, Data augmentation

to myself

Acknowledgements

I would like to warmly thank Mr. Georgios Matsopoulos for supervising this thesis and for giving me the opportunity to undertake it in the Biomedical Technology Laboratory. Also, I particularly thank Mr. Ioannis Kakkos for his guidance and the excellent cooperation we had. Finally, I would like to thank my family and my closest friends, especially Hera Katara, who was a companion in this endeavor.

Athens, March 2024

Christina Manara

Table of Contents

List of Figures

List of Tables

Preface

Στην εποχή μας, όπου η καινοτομία αναπτύσσεται με εντυπωσιακούς ρυθμούς, ένα κομμάτι της επιστημονικής προόδου έχει ξεχωρίσει ως ιδιαίτερα εντυπωσιακό: ο συνδυασμός της Βαθιάς Μάθησης και οι βαθύτατες επιδράσεις της στον κόσμο της ιατρικής απεικόνισης στην πραγματική ζωή, ειδικότερα όταν πρόκειται για την εξέταση των λεπτομερειών του ανθρώπινου εγκεφάλου. Η συνεργασία μεταξύ προηγμένης τεχνητής νοημοσύνης και των αινιγματικών πολυπλοκοτήτων του ανθρώπινου νου ανασχηματίζει το τοπίο της ιατρικής διάγνωσης και θεραπείας με τρόπους που προηγουμένως θεωρούνταν ακατόρθωτοι. Είναι ένα συναρπαστικό ταξίδι που εξετάζει τη σύζευξη της μηχανικής νοημοσύνης με την εύθραυστη τέχνη της αποκρυπτογράφησης των μυστηρίων του εγκεφάλου, και οι επιπτώσεις του δεν είναι λιγότερο από επαναστατικές. Αυτή είναι η εκπληκτική ιστορία του πώς η βαθιά μάθηση αφήνει μια ανεξίτηλη σφραγίδα στον κόσμο της εικόνας του εγκεφάλου, υποσχόμενη ελπίδα, ακρίβεια και τη δυνατότητα για πρωτοφανείς προηγμένες εξελίξεις στη νευροεπιστήμη και την κλινική φροντίδα.

Η παρούσα διπλωματική εργασία επικεντρώνεται στην εύρεση των πολύπλοκων λειτουργιών του ανθρώπινου εγκεφάλου με τη χρήση τεχνικών Βαθιάς Μάθησης για τη διάκριση μεταξύ πραγματικών και φανταστικών κινήσεων των κάτω άκρων. Το κύριο αντικείμενο επιτυγχάνεται μέσω της ανάπτυξης και της χρήσης του Αναδρομικού 3D Συνελικτικού Νευρωνικού Δικτύου (R3DCNN). Τα πειραματικά δεδομένα περιλαμβάνουν τη συλλογή εγκεφαλικών τοπογραφικών χαρτών, την κατάλληλη προεπεξεργασία αυτών, και παράλληλα το μοντέλο (R3DCNN) είναι σχεδιασμένο ειδικά για την εξαγωγή χαρακτηριστικών χώρου και χρόνου, καθιστώντας το κατάλληλο για την πολυπλοκότητα των δεδομένων EEG.

Ειδικότερα, η παρούσα έρευνα εξετάζει την απόδοση του μοντέλου (R3DCNN) με διάφορους βελτιστοποιητές και ρυθμούς μάθησης, επικεντρώντας στην αξιολόγηση των ζυγισμένων μετρικών ταξινομητή για να παρέχει μια ολοκληρωμένη αξιολόγηση. Σημαντικό είναι ότι ο βελτιστοποιητής Στοχαστικής Κλίσης (SGD) επιδεικνύει υψηλή απόδοση, υποδηλώνοντας την αποτελεσματικότητά του στον περίπλοκο χώρο βελτιστοποίησης των υψηλών διαστάσεων δεδομένων EEG. Τα πειράματα ρίχνουν επίσης φως στην επίδραση των ρυθμών μάθησης (learning rates), υποδεικνύοντας ότι η μείωση του ρυθμού μάθησης οδηγεί σε βελτιωμένες ζυγισμένες μετρικές, σηματοδοτώντας λεπτομερέστερες προσαρμογές των βαρών του μοντέλου για ενισχυμένη γενίκευση.

Για την εκτέλεση της παρούσας διπλωματικής εργασίας, είναι αναγκαίο να αντιμετωπιστούν ορισμένες σημαντικές προκλήσεις. Μία από αυτές είναι η μοντελοποίηση με τον πλέον βέλτιστο τρόπο, λαμβάνοντας υπόψη τους περιορισμένους υπολογιστικούς πόρους. Παράλληλα, το είδος των δεδομένων, δηλαδή εικόνες που προέρχονται από τα αντίστοιχα ηλεκτροεγκεφαλογράμματα, αποτελεί έναν αρκετά απαιτητικό τύπο δεδομένων. Αυτό απαιτεί

όχι μόνο την κατάλληλη γνώση για την προεπεξεργασία των εικόνων, αλλά και ειδικές γνώσεις που σχετίζονται με την προεπεξεργασία των εικόνων που αφορούν τη χαρτογράφηση του εγκεφάλου. Φυσικά, η ανάπτυξη αντίστοιχων μοντέλων που θα εφαρμοστούν, μέσω των οποίων θα επιτευχθεί το επιθυμητό αποτέλεσμα - που δεν είναι άλλο από την κατηγοριοποίηση μεταξύ πραγματικής και φανταστικής κίνησης - είναι μία εξίσου σημαντική πρόκληση.

Επιπλέον, μία πρόκληση που προκύπτει είναι η ανισορροπία στο σύνολο δεδομένων. Η ανισορροπία στο σύνολο δεδομένων αντιπροσωπεύει μια κατάσταση όπου υπάρχει μια ανισόρροπη κατανομή του αριθμού των παραδειγμάτων από κάθε κατηγορία σε ένα σύνολο δεδομένων, επηρεάζοντας έτσι την εκπαίδευση του μοντέλου. Αυτό συνήθως οδηγεί σε ένα μοντέλο που είναι είτε προκατειλημμένο προς την κλάση με τα περισσότερα παραδείγματα είτε η απόδοσή του επηρεάζεται στην κλάση με τα λιγότερα παραδείγματα, η οποία μπορεί να είναι κρίσιμη σε πολλές εφαρμογές. Τέτοια μοντέλα μπορεί να παρουσιάσουν απειλητικά υψηλή ακρίβεια, αντικατοπτρίζοντας συχνά την κλάση προτίμησης της πλειοψηφίας, παρά την πραγματική τους δυνατότητα πρόβλεψης. Αυτό μπορεί να μειώσει σημαντικά την ευαισθησία του μοντέλου στην κλάση της μειονοτικής πλειονότητας, οδηγώντας σε μεγάλο αριθμό ψευδών αρνητικών για τη λιγότερο αντιπροσωπευόμενη κλάση. Με άλλα λόγια, θα υπήρχαν σοβαρές συνέπειες, κατά την ανίχνευση απάτης ή τη διάγνωση ασθενειών, με αποτέλεσμα να καθίσταται αναγκαία η επαρκής αντιμετώπιση των ανισορροπημένων σύνολων δεδομένων κατά τη διαδικασία ανάπτυξης του μοντέλου.

Η πολυπλοκότητα των χαρτών εγκεφαλικής τοπογραφίας που αναπαριστούν τα δεδομένα EEG χειρίζεται αποτελεσματικά από το μοντέλο (R3DCNN), επιδεικνύοντας τη δυνατότητά του να αποκτήσει χαρακτηριστικά χώρου, φάσματος και χρόνου. Τα ευρήματα υποδεικνύουν ότι το μοντέλο είναι κατάλληλο για εργασίες όπως η ταξινόμηση κινήσεων των κάτω άκρων. Η συζήτηση τονίζει τις δυνητικές εφαρμογές του μοντέλου (R3DCNN) στη νευρολογική έρευνα και τις πρακτικές καταστάσεις, επικεντρώνοντας στην ανθεκτικότητά του και την προσαρμοστικότητά του.

Ένα καίριο βήμα στην επεξεργασία δεδομένων είναι η κατάλληλη επισήμανση του συνόλου δεδομένων, χρησιμοποιώντας έναν συνδυασμό με τρεις αδύναμους μαθητές. Η ιεραρχική προσέγγιση επιτρέπει λεπτομερή ανάλυση, χρήσιμη σε εφαρμογές όπως η ανάλυση βήματος και η παρακολούθηση αποκατάστασης. Για την ενίσχυση της προσαρμοστικότητας του μοντέλου, χρησιμοποιούνται τεχνικές επαύξησης δεδομένων, όπως η μετατροπή εικόνων σε κλίμακα του γκρι και η χρήση της τεχνικής του interpolation (Rbf). Η τελευταία περιλαμβάνει την αλλαγή των διαστάσεων του αρχικού τένσορα, με στόχο την απλοποιήση, χωρίς την απώλεια πληροφορίας, μετά τον καθορισμό των ακριβών σημείων-ηλεκτροδίων στο τριχωτό της κεφαλής.

Στη διάρκεια των πειραμάτων που διεξήχθησαν χρησιμοποιώντας το μοντέλο R3DCNN σε χαρτογραφημένα εγκεφαλικά διαγράμματα, μπορούν να εξαχθούν διάφορα συμπεράσματα σχετικά με την απόδοση του μοντέλου με διάφορους βελτιστοποιητές και ρυθμούς μάθησης, επικεντρώνοντας ιδιαίτερα στις σταθμισμένες μετρικές ταξινομητή.

Το μοντέλο δοκιμάστηκε με διάφορους βελτιστοποιητές - SGD, RMSprop και Adam - κάθε ένας με διαφορετικούς ρυθμούς μάθησης. Σε ό,τι αφορά τις σταθμισμένες μετρικές ταξινομητή, που παρέχουν μια ισορροπημένη άποψη της απόδοσης του μοντέλου σε όλες τις

κατηγορίες, ο βελτιστοποιητής SGD με έναν προγραμματιστή ρυθμού μάθησης φάνηκε να προσφέρει υπεροχή στην απόδοση. Αυτό υποδεικνύει ότι, για τα δεδομένα χαρτογραφημένων εγκεφαλικών διαγραμμάτων, η προσέγγιση του SGD στον χώρο βελτιστοποίησης, λόγω των συνιστωσών της ορμής και των προσαρμογών του ρυθμού μάθησης, είναι αποτελεσματική.

Ο ρυθμός μάθησης είναι μία σημαντική υπερπαράμετρος στην εκπαίδευση των μοντέλων Βαθιάς Μάθησης. Τα πειράματα δείχνουν μια γενική τάση ότι, για το μοντέλο R3DCNN, όσο περισσότερο μειώνεται ο ρυθμός μάθησης, τόσο βελτιώνονται οι σταθμισμένες μετρικές, υποδεικνύοντας μια πιο λεπτομερή προσαρμογή των βαρών του μοντέλου που οδηγεί σε καλύτερη γενίκευση.

Το μοντέλο R3DCNN σχεδιάστηκε για να αντιλαμβάνεται τα πολύπλοκα χαρακτηριστικά χωρικής, φασματικής και χρονικής φύσης που προκύπτουν από τα δεδομένα EEG, όπως αυτά αναπαρίστανται από τα χαρτογραφημένα εγκεφαλικά διαγράμματα. Τα πειράματα επιδεικνύουν την ικανότητα του μοντέλου να αντιμετωπίζει την πολυπλοκότητα των δεδομένων EEG, δείχνοντας μια ελπιδοφόρα κατεύθυνση για την χαρτογράφιση του ανθρώπινου εγκεφάλου.

Συνολικά, αυτή η έρευνα επιβεβαιώνει την αποτελεσματικότητα του μοντέλου (R3DCNN) στη χειρισμό πολύπλοκων δεδομένων EEG, ανοίγοντας το δρόμο για προηγμένες εξελίξεις στη νευρολογική έρευνα και τις πρακτικές εφαρμογές. Η εξαντλητική εξερεύνηση βελτιστοποιητών, ρυθμών μάθησης και τεχνικών επεξεργασίας δεδομένων συνεισφέρει σημαντικές προοπτικές στην απόδοση και γενίκευση του μοντέλου. Η ιεραρχική διαδικασία λήψης αποφάσεων και η μέθοδος ταξινόμησης με ενσωματωμένο σύνολο από τρεις αδύναμους μαθητές ενισχύουν την ευελιξία του μοντέλου, υποδεικνύοντας το δυναμικό του σε διάφορους τομείς πέρα από την ταξινόμηση κινήσεων ποδιών.

Η ένταξη των συνελικτικών νευρωνικών δικτύων (CNNs) για την εξαγωγή χωρικών χαρακτηριστικών με τα αναδρομικά νευρωνικά δίκτυα (RNNs) για την καταγραφή χρονικών εξαρτήσεων έχει δείξει δυναμικό στην αποκωδικοποίηση πολύπλοκων σημάτων EEG. Ωστόσο, η βελτιστοποίηση της αρχιτεκτονικής και η εξερεύνηση πιο προηγμένων μοντέλων θα μπορούσε να οδηγήσει σε σημαντικές βελτιώσεις. Μελλοντικές μελέτες θα μπορούσαν να εξετάσουν την πειραματική χρήση διαφορετικών τύπων συνελικτικών και αναδρομικών στρωμάτων, όπως Transformer-based μοντέλα. Επιπλέον, μια συστηματική προσέγγιση στον ρυθμιστή υπερπαραμέτρων θα μπορούσε να καταστήσει περαιτέρω εκλεπτυσμένη την απόδοση του μοντέλου, χρησιμοποιώντας τεχνικές όπως random search, greed search κ.λπ.

Τέλος, καθώς τα μοντέλα γίνονται πιο πολύπλοκα, η διασφάλιση της ερμηνευσιμότητάς τους γίνεται κρίσιμη, ιδίως σε κλινικά περιβάλλοντα. Επιπλέον, η ένταξη πολυτροπικών δεδομένων, όπως fMRI, με τα δεδομένα EEG θα μπορούσε να προσφέρει μια πιο συνολική εικόνα των εγκεφαλικών δραστηριοτήτων, ενισχύοντας τις προβλεπτικές δυνατότητες του μοντέλου. Η εξερεύνηση αυτών των προσεγγίσεων θα βελτιώσει όχι μόνο την απόδοση και τη χρησιμότητα των μοντέλων βασισμένων σε EEG, αλλά θα συμβάλει επίσης στον ευρύτερο τομέα της νευροτεχνολογίας και των εφαρμογών της.

Chapter 1

Introduction

1.1 Artificial Intelligence

The history of Artificial Intelligence (AI) spans from early theoretical underpinnings in the mid-20th century to its current status as a cornerstone of technological advancement and societal integration. The journey began with foundational work by pioneers who explored the potential for machines to simulate human intelligence. The 1950s saw the term "Artificial Intelligence" being coined at the Dartmouth Conference, marking the official start of AI as a research field.

The initial years were marked by enthusiasm and notable successes, such as the development of early natural language programs. However, the field faced setbacks during the "AI winters" of the late 1970s and late 1980s, when expectations outpaced results, leading to reduced funding and interest.

Renewed focus on machine learning and neural networks in the 1980s brought AI back to prominence. The development of deep learning techniques in the 2010s, particularly in areas such as image and speech recognition, showcased AI's potential to perform complex tasks with accuracy surpassing human benchmarks in some domains, like the game of Go.

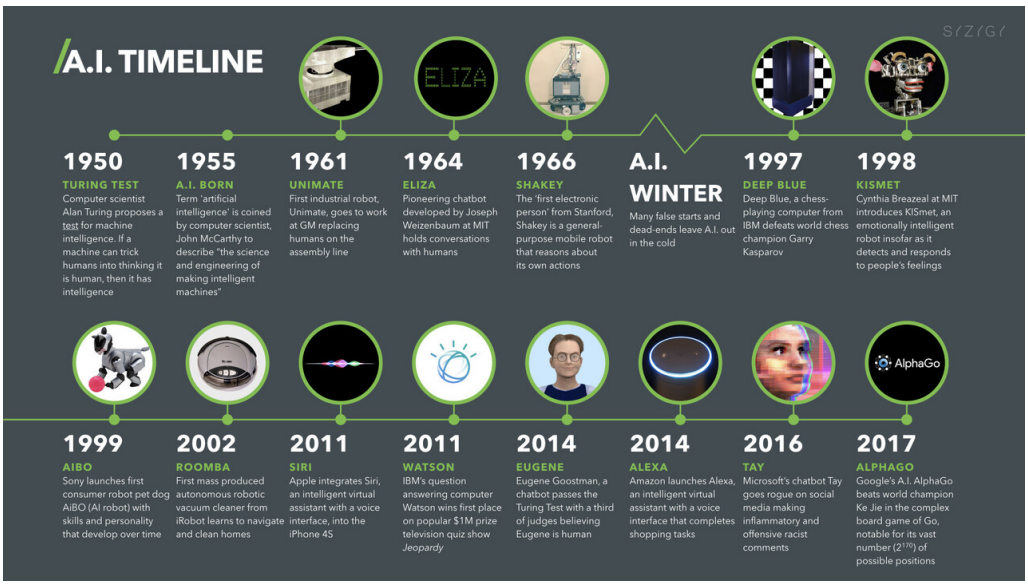

Figure 1.1. *Evolution of AI.*

Today, AI is interwoven into various sectors, driving innovations in automation, entertainment, healthcare, and beyond. Despite its advances, AI's rapid growth also raises ethical concerns regarding privacy, bias, and the impact on employment, sparking debates on how to navigate its future responsibly. This condensed narrative underscores the cyclical nature of AI's development - from theoretical exploration to practical application and societal integration, highlighting the ongoing balance between technological possibilities and ethical considerations[1].

1.2 Machine Learning

Machine Learning (ML) is a field of AI focused on enabling machines to learn from data, improve over time, and make decisions or predictions. Key concepts include:

- **Supervised Learning**: Training models on labeled data to predict outcomes for new, unseen data, applicable in classification and regression tasks.
- **Unsupervised Learning**: Identifying patterns or structures in unlabeled data, used for clustering and dimensionality reduction.
- **Semi-supervised Learning**: Combining labeled and unlabeled data to improve learning efficiency, useful when labels are scarce or expensive to obtain.
- **Reinforcement Learning**: Learning to make decisions by taking actions in an environment to maximize some notion of cumulative reward.
- **Feature Extraction and Selection**: Identifying the most relevant information from the data to use for training models, crucial for model performance and efficiency.
- **Overfitting and Underfitting**: Challenges in ML where models either learn the noise in the training data (overfitting) or fail to capture the underlying data pattern (underfitting), affecting their prediction accuracy.
- **Cross-validation**: A technique for assessing how the outcomes of a statistical analysis will generalize to an independent dataset, helping to mitigate overfitting.
- **Regularization**: Techniques to simplify models to prevent overfitting, ensuring they perform well on new, unseen data.
- **Loss Functions**: Metrics to quantify the difference between the model's predictions and actual data, guiding the training process.
- **Gradient Descent**: An optimization algorithm used for minimizing the loss function in the training process, essential for finding the model parameters that result in the best model performance.

These concepts form the backbone of machine learning, enabling a wide range of applications from predictive modeling and data analysis to autonomous systems and beyond.

1.3 Deep Learning

1.3.1 Artificial Neuron

An artificial neuron is a fundamental building block of artificial neural networks, designed to mimic the properties of biological neurons. The concept of an artificial neuron stems from the desire to replicate the human brain's ability to process and transmit information, thus enabling machines to learn and make decisions.

Structure and Function

An artificial neuron, as it is shown in the image 1.2, typically consists of inputs, weights, a bias, an activation function, and an output. The inputs represent the data received by the neuron, similar to the dendrites in a biological neuron. Each input is associated with a weight, which adjusts the strength or importance of the input. The bias is akin to the neuron's threshold level that needs to be surpassed for activation.

The core operation in an artificial neuron involves the weighted sum of its inputs, plus the bias. This sum is then passed through an activation function, which determines the neuron's output. The activation function's role is to introduce non-linearity into the neuron's output, enabling the neural network to learn and model complex patterns.

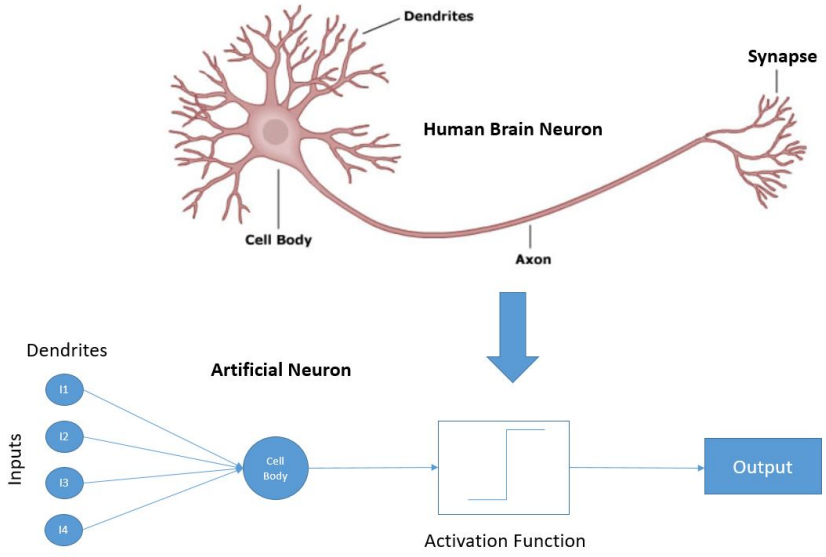

Figure 1.2. *Artificial Neuron.*

1.3.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computing systems inspired by the biological neural networks that constitute animal brains. An ANN 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 from one artificial neuron to another. The receiving neuron processes the signal and then

signals downstream neurons connected to it. ANNs have found applications in a wide range of tasks, from speech recognition and image classification to drug discovery and autonomous vehicle control.

Key Components and Structure

- **Neurons**: The basic computational units of the ANN, designed to mimic the neurons in the brain. Each neuron receives input, processes it, and generates an output.
- **Weights**: These are the parameters within the neural network that transform input data within the network's hidden layers. Weights are adjusted during the training process to minimize the difference between the actual output and the target output.
- **Bias**: A bias value allows you to shift the activation function to the left or right, which can be critical for successful learning.
- **Layers**: ANNs consist of layers: an input layer, one or more hidden layers, and an output layer. The input layer receives the initial data for processing, the hidden layers perform computations with weights and biases, and the output layer produces the final prediction or classification.
- **Activation Function**: This function is applied to the input sum, including weights and bias, to determine the neuron's output. Common activation functions include sigmoid, tanh, ReLU (Rectified Linear Unit), and softmax.

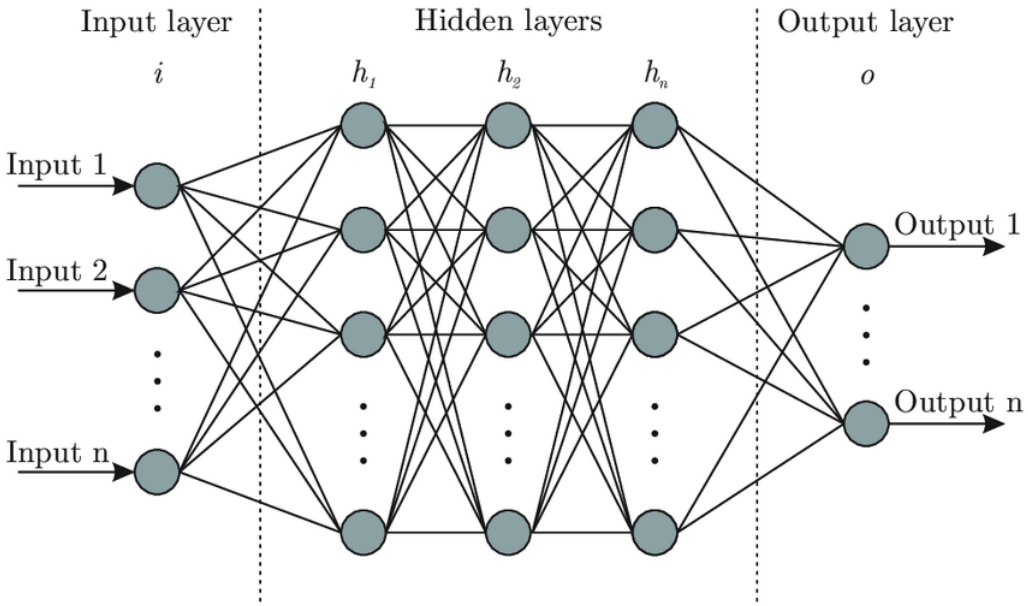

Figure 1.3. *Artificial Neural Network.*

Artificial Neural Networks (ANNs) - 1.3, have revolutionized a wide array of applications across various fields due to their ability to model complex patterns and make intelligent decisions. This survey[2] extensively reviews applications of Artificial Neural Networks

(ANNs) in diverse fields such as healthcare, where ANNs contribute to disease diagnosis and drug discovery; environmental science, aiding in climate modeling and pollution control; and robotics, enhancing autonomous navigation and human-robot interaction. It also touches on the use of ANNs in financial markets for predictive analysis and risk management, and in automotive industries, particularly for the development of self-driving car technologies. These applications underscore the transformative potential of ANNs across various sectors, driving innovation and solving complex challenges. In image and speech recognition, ANNs have significantly improved the accuracy and efficiency of identifying objects within images or transcribing spoken words into text, making technologies like facial recognition and voice-activated assistants increasingly reliable. In the realm of natural language processing (NLP), ANNs are instrumental in translating languages, generating human-like text, and understanding user intents, thereby enhancing communication between humans and machines.

While ANNs have driven much of the progress in AI, they also face challenges such as vulnerability to adversarial attacks, the need for large amounts of training data, and the "black box" nature of deep learning models that makes them hard to interpret. Future research is directed towards making ANNs more efficient, explainable, and robust against attacks, as well as reducing their reliance on large datasets for training.

1.4 Neuroscience

In our rapidly advancing technological era, where innovation unfolds at an astonishing pace, one facet of scientific progress has stood out as particularly awe-inspiring: the marriage of deep learning and its profound effects on the realm of real-life medical imaging, specifically when it comes to examining the intricacies of the human brain. The synergy between cutting-edge artificial intelligence and the enigmatic complexities of the human mind is reshaping the landscape of medical diagnostics and treatment in ways previously thought unattainable. It's a captivating journey that delves into the fusion of machine intelligence and the delicate art of deciphering the brain's mysteries, and its implications are nothing short of revolutionary. This is the remarkable tale of how deep learning is leaving an indelible mark on the world of brain imaging, promising hope, precision, and the potential for unprecedented advancements in neuroscience and clinical care.

Neuroscience delves deeper into understanding how individual neurons operate, how they communicate through neurotransmitters, the importance of various brain regions like the hippocampus in memory, and the prefrontal cortex in decision-making. It also studies the mechanisms behind neuroplasticity, illustrating the brain's adaptability through learning and experience. Fundamental to neuroscience is the exploration of sensory systems, motor control, and the neural basis of consciousness. These concepts are crucial for unraveling the complexities of neural function and dysfunction, offering insights into treating neurological disorders.

1.4.1 Human Brain

The image 1.4 shows a sagittal section of the human brain, highlighting three main regions: the forebrain, midbrain, and hindbrain. The forebrain is the largest part, shown in pink, encompassing the cerebral cortex and underlying structures. Below it, in green, is the midbrain, a small central part of the brainstem that plays a role in vision, hearing, and motor control. The hindbrain, depicted in mauve, includes the cerebellum and brainstem structures that control vital functions such as breathing and heart rate [3].

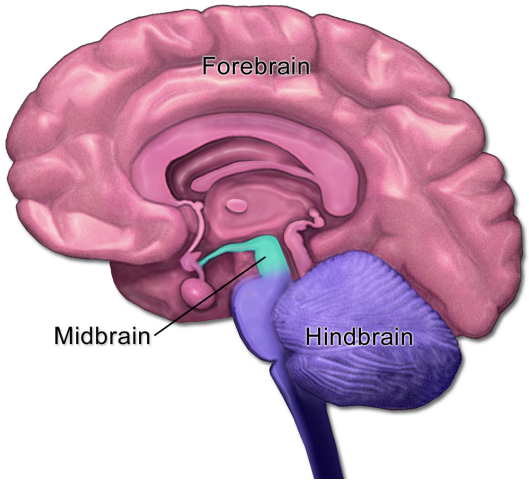

Figure 1.4. *Human Brain.*

1.4.2 EEG

An EEG, or electroencephalogram, is a test that detects electrical activity in the brain using small metal discs attached to the scalp. Brain cells communicate via electrical impulses even when they're at rest, and an EEG can be used to help diagnose brain disorders, monitor the depth of anesthesia, determine if someone is in a coma, or confirm brain death. It's often used in the diagnosis of epilepsy and other neurological conditions. The test is non-invasive and can track brain wave patterns that might signify abnormalities.

The metal discs used in an EEG are called electrodes. They are attached to the scalp with a conductive gel or paste and are responsible for picking up the electrical impulses that occur in the brain. These impulses are then amplified and recorded by the EEG machine, producing a trace for each electrode that reflects the brain's activity. The patterns seen in an EEG can be analyzed by specialists to detect abnormalities. The placement of the electrodes is typically done according to a standardized system known as the International 10–20 system, which ensures that the locations are consistent for each test and between different individuals, Figure 1.5.

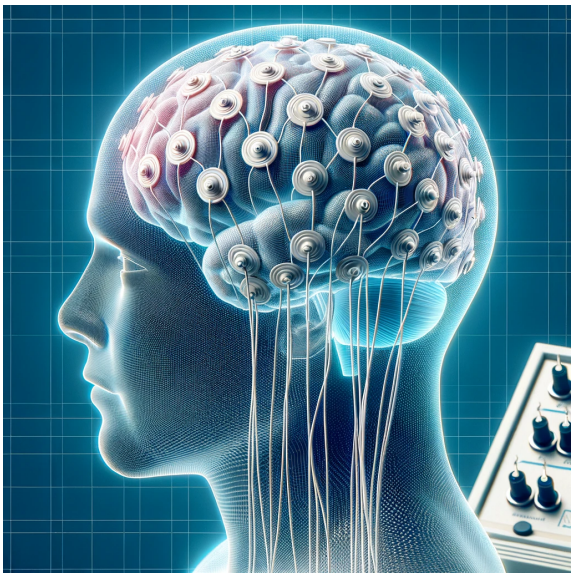

Figure 1.5. *Human brain with electrodes.*

1.4.3 EEG Frequency Bands

EEG waves are categorized into several frequency bands, each associated with different brain states. Delta waves, at 4 Hz or lower, are dominant during deep sleep stages and in infancy. Theta waves, ranging from 4 to 8 Hz, are common in children up to 13 years old. Alpha waves, with frequencies from 8 to 12 Hz, typically appear when an adult is relaxed and are most prominent in the posterior head regions. Lastly, Beta waves, above 12 Hz, are observed during alertness and periods of anxiety or prolonged attention [4], Figure 1.6.

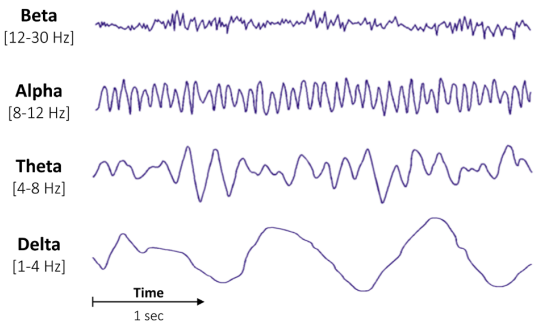

Figure 1.6. *EEG frequency bands.*

1.4.4 EEG Systems

The 10-20 system, figure 1.7, introduced by Herbert Jasper at the 1957 International EEG Congress in Brussels, established a standardized approach for the placement of EEG electrodes. The system's nomenclature, "10" and "20", indicates the percentage increments of the skull's total front-back or right-left distance used to position the electrodes. This distance is measured using four key anatomical landmarks on the scalp: the nasion, the inion, and the two preauricular points. Electrodes are first positioned 10%

away from these landmarks, and subsequent electrodes are spaced at 20% intervals. For instance, electrode Fp1 is located 10% from the nasion, while Fz is positioned 20% from Fp1 along the front-back axis [5].

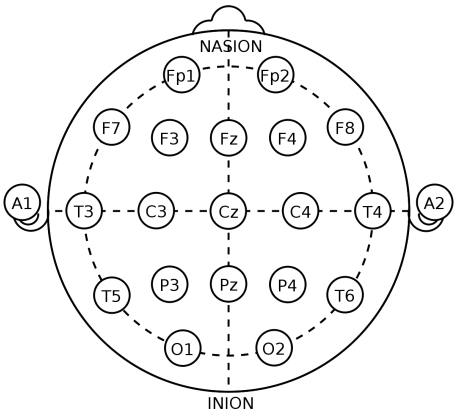

Figure 1.7. *10-20 System.*

Building on this, the 10-10 system, figure 1.8, offers a more detailed electrode placement strategy, essentially doubling the number of electrodes used in the 10-20 system by halving the distance between adjacent sites to 10%. This provides a finer grid for mapping EEG activity, allowing for more precise localization of brain activity. Both systems are crucial for ensuring consistent, reproducible EEG recordings across different studies and clinical assessments [5].

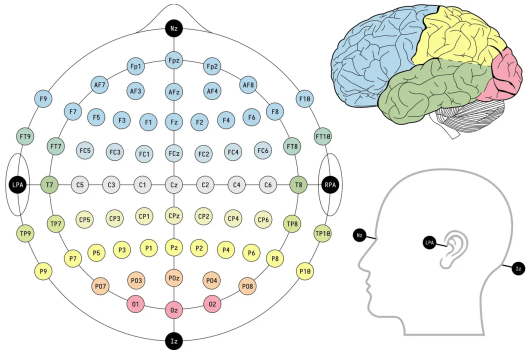

Figure 1.8. *10-10 System.*

1.5 Related Work

To comprehend how the human brain works, this thesis aims to differentiate between real and imaginary lower body movements using Deep Learning. After gathering topographical maps through an experiment, described properly in Chapter 4, R3DCNN is developed and utilized in order to be achieved the main purpose of this diploma thesis. The current research leverages this particular model on analogous data that demands suitable spatial and temporal feature determination. Several research articles have utilized this model in similar ways.

More specifically, in [6], "Learning Spatial-Spectral-Temporal EEG Features With Recurrent 3D Convolutional Neural Networks for Cross-Task Mental Workload Assessment"

aims to tackle the challenge of cross-task mental workload assessment by using deep learning techniques, specifically a concatenated structure of deep recurrent and 3D convolutional neural networks (R3DCNNs). The authors propose a new method to learn EEG features across different tasks without prior knowledge by adding frequency and time dimensions to EEG topographic maps through a Morlet wavelet transformation. The R3DCNN is then trained to learn from these enriched EEG features in spatial, spectral, and temporal dimensions. The paper validates the model using EEG signals collected from 20 subjects performing binary classification of low and high mental workload across spatial n-back and arithmetic tasks. The results show a significant improvement in accuracy over state-of-the-art methods, with the R3DCNN achieving an average accuracy of 88.9%. The research presents a promising direction for using deep learning models to understand and analyze complex brain dynamics across various tasks without the need for hand-crafted features.

The research [7], titled "Human Action Representation Learning Using an Attention-Driven Residual 3DCNN Network," is to address the challenge of recognizing human actions in video streams by proposing a computationally efficient yet robust deep learning model. This paper introduces the Dual-Attentional Residual 3D Convolutional Neural Network (DA-R3DCNN), which incorporates channel and spatial attention mechanisms to improve the identification of human activities. The DA-R3DCNN model is designed to propagate salient features through the network layers, significantly enhancing its performance for the task of human activity recognition. The paper outlines the design of the model, its advantages over other methods, and the results of extensive experiments performed on benchmark datasets. The findings demonstrate that the proposed DA-R3DCNN method achieves substantial improvements in recognition accuracy and efficiency, offering up to an 11% increase in accuracy and a 74$\times$ boost in processing speed compared to existing techniques, making it suitable for real-time applications.

The paper [8] titled "Deep Long Short-term Memory Structures Model Temporal Dependencies Improving Cognitive Workload Estimation" explores the use of deep Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) architectures, to enhance the estimation of cognitive workload from EEG data. This is achieved by leveraging the temporal dependencies present in EEG time-series signals. The study reports significant improvements in day-to-day feature stationarity and classification accuracy when compared to classifiers that do not account for temporal dependence. The research involved training various deep RNN models, including LSTMs, on data collected from six participants over five different days within a month. Each participant-specific classifier was trained on data from the first four days and tested on the fifth day's data, leading to an impressive average classification accuracy of 93.0% using a deep LSTM architecture, which is a 59% reduction in error compared to the best results previously reported for the same dataset. The study also assessed the significance of new features derived from the mean, variance, skewness, and kurtosis of EEG frequency-domain power distributions, finding that mean and variance are statistically significant features for this task.

Part I

Theoretical Part

Chapter 2

Convolutional Neural Networks

2.1 Evolution of CNN Architecture

The history of Convolutional Neural Networks (CNNs) traces back to the 1960s, starting with foundational work on the visual cortex that influenced early neural network models. The concept of CNNs was significantly advanced in the 1980s with Fukushima's Neocognitron, a model for pattern recognition that lacked backpropagation. The introduction of the backpropagation algorithm enabled efficient training of neural networks, setting the stage for LeNet-5 in 1998 by Yann LeCun, which applied CNNs to digit recognition. The breakthrough came in 2012 with AlexNet, which won the ImageNet challenge and ignited the deep learning era in computer vision. Since then, the development of CNNs has seen rapid progress with architectures like GoogLeNet, VGGNet, and ResNet, each pushing the boundaries of accuracy and efficiency. The evolution from theoretical models to complex architectures underscores the transformative impact of CNNs in artificial intelligence and computer vision [9]. Below, in figure 2.1, it is shown the evolution of CNN architecture.

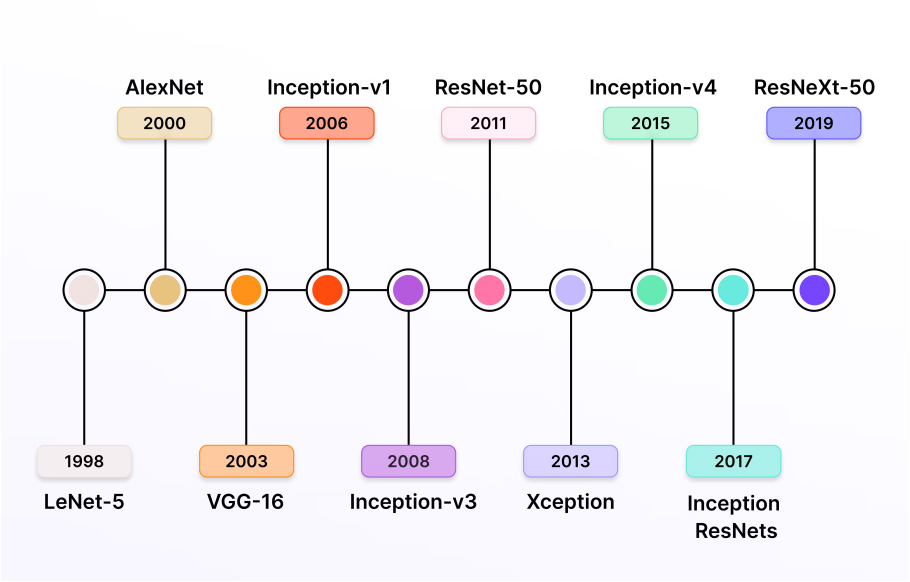

Figure 2.1. *Timeline of CNN evolution.*

Convolutional Neural Networks (CNNs) are a class of deep neural networks highly effective for processing data with a grid-like topology, such as images. CNNs are distinguished

by their unique architecture, designed to automatically and adaptively learn spatial hierarchies of features from input data. This capability makes them particularly suited for computer vision tasks, including image and video recognition, image classification, and object detection, among others.

2.2 Core Components of CNNs

2.2.1 Convolutional Layers

The primary building blocks of a CNN. These layers apply a convolution operation to the input, passing the result to the next layer. A convolutional layer is composed of an array of convolutional filters, commonly referred to as kernels.

A kernel is defined as a matrix of distinct numerical values, where each number is referred to as a kernel weight. Initially, these weights are set to random values when the training of a CNN starts. There are various techniques employed to initialize these weights. Subsequently, through successive training epochs, these weights are fine-tuned, enabling the kernel to effectively identify important features [10].

The convolution emulates the response of an individual neuron to visual stimuli, focusing on small, receptive fields and allowing the network to capture spatial and temporal dependencies in an image.

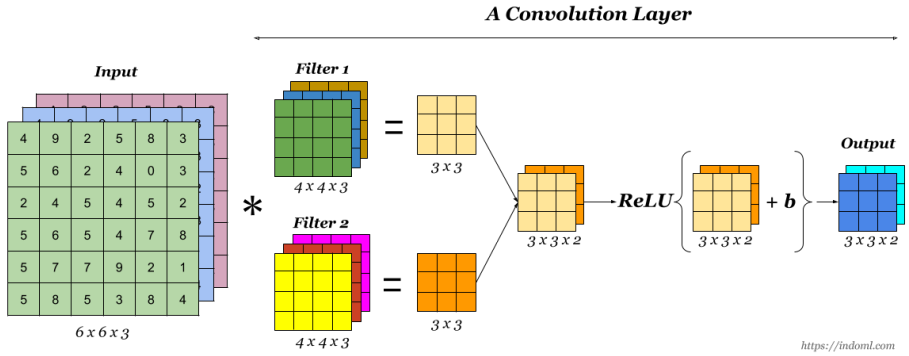

Figure 2.2. *A Convolutional Layer.*

The above image, 2.2, illustrates the process of applying convolutional filters (kernels) to an input matrix, commonly encountered in the convolutional layers of a neural network. The input matrix, which could represent an image or feature map, is shown with various numbers in each cell. Superimposed on the input are several smaller, colored grids representing different convolutional kernels.

During the convolution operation, these kernels slide over the input matrix - a process referred to as convolution - to create new matrices known as feature maps or convolved features. Each position of the kernel over the input matrix involves element-wise multiplication of the kernel weights with the values of the input matrix that are currently under the kernel. The products are then summed up to give a single number in the corresponding cell of the output feature map.

This operation is replicated across the entire input matrix, with the kernel moving over it according to a defined stride, which is the step size with which the kernel moves.

The resulting feature maps highlight different features of the input data, depending on the pattern of weights in the kernel. This is essential in CNNs as it allows the network to detect edges, textures, colors, and other visual elements in image data, or similarly meaningful patterns in other types of data.

2.2.2 Activation Functions

Following convolution, an activation function such as ReLU (Rectified Linear Unit) is applied to introduce non-linearity into the model, enabling it to learn complex patterns. Activation functions in neural networks are vital for mapping inputs to outputs, fundamentally determining whether neurons should be activated based on the given inputs. These functions take the weighted sum of the inputs and biases and generate an output that decides the neuron's activation. In CNNs, non-linear activation functions follow layers with learnable parameters, like fully connected (FC) and convolutional layers, enabling the network to capture complex patterns and learn from them. The ability of these functions to be differentiated is crucial for the backpropagation of errors during network training [10]. Among the most prevalent activation functions are:

- **Sigmoid**: Transforms real-numbered inputs into a range between 0 and 1 with an S-shaped curve.

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

- **Tanh**: Similar to the sigmoid but outputs values from -1 to 1, encompassing a zero-centered range.

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

- **ReLU**: Common in CNNs, it turns all negative inputs into zeros, which simplifies computation but sometimes leads to the "Dying ReLU" problem, where neurons stop activating.

$$\boxed{\mathrm{ReLU}(x) = \max(0, x)}$$

Alternative ReLU variants address its shortcomings:

- **Leaky ReLU**: Allows a small, non-zero gradient when the input is negative, preventing neurons from dying out.
- **Noisy ReLU**: Adds randomness to the ReLU function to aid in robust learning.
- **Parametric Linear Units**: Similar to Leaky ReLU, but the leak factor is learned during training, adding adaptability to the model.

These functions, whose graphs are shown in the figure 2.3, enrich the neural network's learning capacity, enabling a deep model to learn and perform complex tasks effectively.

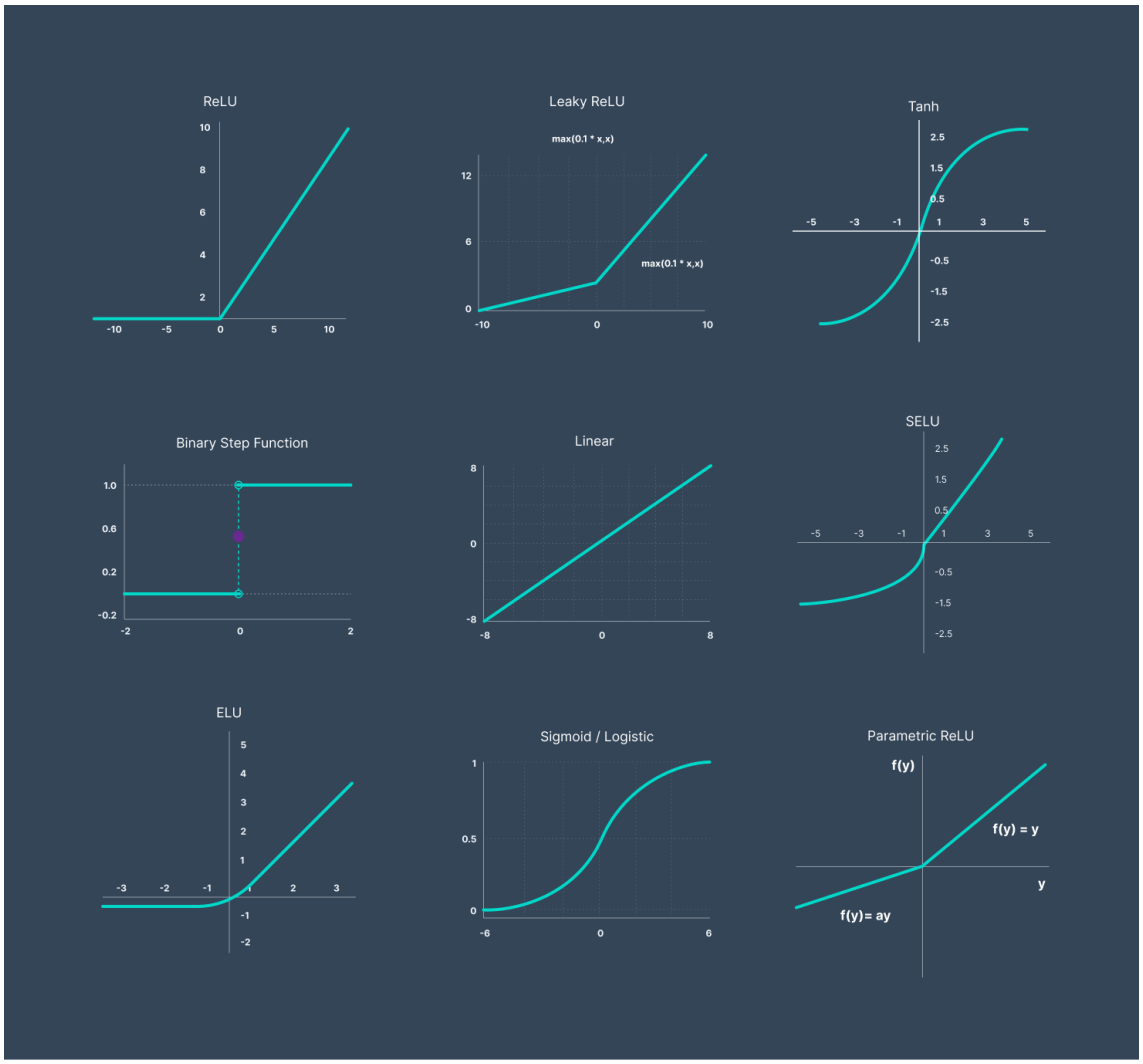

Figure 2.3. *Types of Activations Functions.*

2.2.3 Pooling Layers

These layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Pooling (often max pooling) helps to make the detection of features invariant to scale and orientation changes.

The primary function of the pooling layer in a Convolutional Neural Network (CNN) is to downsample feature maps produced by convolutional layers. This process effectively reduces the size of the feature maps while retaining most of the critical information within them. Similar to convolutional operations, both stride and kernel sizes are predefined before performing pooling. Various pooling techniques exist, including but not limited to average pooling, max pooling, min pooling, global average pooling (GAP), and global max pooling, with max, min, and GAP being the most commonly used. These methods are instrumental in shrinking feature maps to more manageable sizes. However, one notable drawback of pooling is its potential to diminish CNN performance. This is because while pooling can identify the presence of certain features within an input image, it primarily aims to locate these features without preserving all related information, leading to possible

loss of detail crucial for accurate model predictions [10].

This image, 2.4, illustrates three types of pooling operations used in Convolutional Neural Networks: Max Pooling, Min Pooling, and Average Pooling. On the left, a feature map is depicted with various numerical values within each cell, and a highlighted 2x2 region indicates the pool size and stride being applied.

- **Max Pooling**: This operation calculates the maximum value within the specified pool size (2x2 in this case). The output only retains the maximum value from each sub-region, effectively downsampling the feature map and reducing its dimensions while preserving the most significant features, which are the highest values.
- **Min Pooling**: Conversely, min pooling takes the smallest value within each 2x2 sub-region. This method is less common and tends to preserve the low-intensity features.
- **Average Pooling**: This operation computes the average of the values within the pool size, providing an output that represents the general average feature intensity in each sub-region.

In the provided outputs, each 2x2 sub-region of the original feature map is reduced to a single value, corresponding to the max, min, or average of that region, resulting in a downsized output feature map. These pooling methods help to reduce computational load, control overfitting, and provide a form of translation invariance to the feature detection in CNNs.

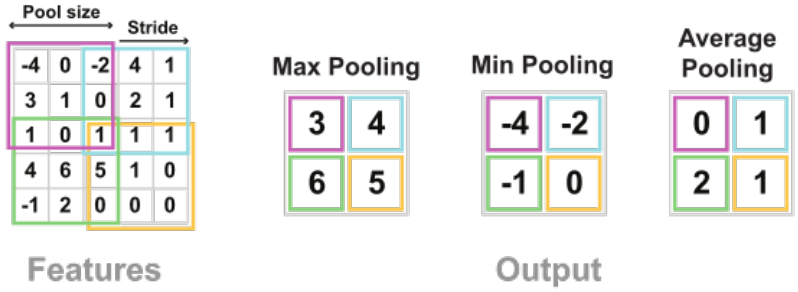

Figure 2.4. *Pooling Types.*

2.2.4 Fully Connected Layers

In Convolutional Neural Networks (CNNs), fully connected (FC) layers play a crucial role in the network's architecture. After the initial layers have detected features using convolution and pooling operations, the role of fully connected layers is to interpret these features and use them for classifying the input into various categories.

In a mathematical context, fully connected layers in a CNN can be represented by a matrix multiplication followed by a bias offset and typically an activation function. For a given fully connected layer, if you have an input vector $\mathbf{x}$, the operation performed by the layer can be represented as follows:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where:

- $\mathbf{x}$ is the flattened input vector from the preceding layer.
- $\mathbf{W}$ represents the weight matrix of the fully connected layer.
- $\mathbf{b}$ is the bias vector.
- f denotes the activation function applied element-wise, such as ReLU, sigmoid, or tanh.
- $\mathbf{y}$ is the output vector of the fully connected layer.

2.2.5 Softmax Layer

The softmax layer is a crucial component at the output of many neural network architectures, particularly in classification tasks. It takes as input the scores (also known as logits) from the previous layer and converts them into probabilities by applying the softmax function. The softmax function is designed to ensure that the output values are in the range of [0, 1] and sum up to 1, making them interpretable as probabilities.

The softmax layer is applied to the logits (raw class scores) from the final layer of a neural network to convert them into probabilities. The softmax function for a score vector $\mathbf{z}$ is defined as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

where:

- e^{z_i} is the exponential of the score for class i.
- The denominator is the sum of exponentials of all raw class scores in the vector $\mathbf{z}$.

The softmax layer ensures that each output value will be in the interval (0,1), and the entire set of values will sum to 1, thus forming a valid probability distribution. This property makes the softmax function particularly useful for multi-class classification problems, where it can provide a clear, probabilistic interpretation of the model's predictions across multiple classes.

In a neural network, applying the softmax function to the final layer's output allows one to interpret the highest probability as the model's prediction for the input's class. This is especially useful in tasks like image classification, natural language processing (NLP), and any other domain where probabilistic outputs are desired.

2.2.6 Loss Functions

The previous section outlined various layers in a CNN's architecture, culminating in the output layer which is responsible for classification. The output layer uses loss functions to measure the discrepancy between the predicted and actual outcomes. This error is then minimized during the network's learning phase. The loss function considers two key parameters: the predicted output (or prediction) and the actual output (or label). Different loss functions are applied based on the problem at hand, including:

- **Cross-Entropy Loss Function**: Often used in classification tasks, it measures the performance of the CNN model by outputting a probability between 0 and 1. It replaces the mean square error in multi-class classification scenarios and works with softmax activations to produce a probability distribution [10].

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e_k^a} \tag{2.1}$$

- **Euclidean Loss Function**: Also known as mean square error, this function is primarily utilized in regression problems to measure the average squared difference between the predicted values and the actual outcome [10].

$$H(p, y) = \frac{1}{2N} \sum_{i=1}^{N} (p_i - y_i)^2 \tag{2.2}$$

- **Hinge Loss Function**: This is typically used in binary classification tasks and is important for Support Vector Machines (SVMs) that aim to maximize the classification margin. It is designed to penalize predictions that are on the wrong side of the margin boundary [10].

$$H(p, y) = \sum_{i=1}^{N} \max(0, m - (2y_i - 1)\, p_i) \tag{2.3}$$

- **Focal Loss**: Focal loss is a modified version of the cross-entropy loss function, designed to address class imbalance in object detection tasks where the vast number of easy negatives can overwhelm the training process. It was introduced by Lin et al. in the context of training highly imbalanced datasets for object detection. The key idea behind focal loss is to apply a modulating term to the cross-entropy loss in order to focus learning on hard negatives. It reduces the relative loss for well-classified examples, putting more focus on hard, misclassified examples. This is particularly useful in scenarios where there is a large class imbalance, as it prevents the vast number of easy negatives from dominating the loss.

  The focal loss function is defined as:

$$FL(p_t) = -a_t(1 - p_t)^{\gamma} \log(p_t) \tag{2.4}$$

where:

- p_t is the model's estimated probability for the class.
- a_t is a balancing factor for the importance of positive/negative examples.
- γ is the focusing parameter that adjusts the rate at which easy examples are down-weighted.

Each of these loss functions plays a role in guiding the CNN to accurate predictions by quantifying and optimizing the prediction error throughout the training process.

2.2.7 A simple CNN architecture

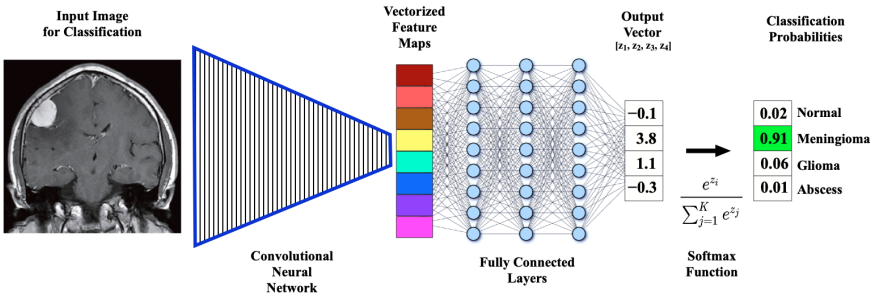

Figure 2.5. *A Convolutional Neural Network.*

The image, 2.5, provides a visualization of a Convolutional Neural Network (CNN) processing an input image for the purpose of classification. The process is broken down into several stages:

- **Input Image for Classification**: The image on the left shows an MRI scan of the brain, which is the input for the CNN.
- **Convolutional Neural Network**: In the center, the network is symbolized by a series of vertical bars, representing the convolutional layers that extract features from the input image.
- **Vectorized Feature Maps**: The extracted features are condensed into color-coded feature maps, which are then flattened into a vector, as shown by the colorful bars transitioning into a column of circles.
- **Fully Connected Layers**: These circles represent the neurons of the fully connected layers, which interpret the features extracted by the convolutional layers. The connections between these neurons are depicted by the mesh of lines.
- **Output Vector [Z1, Z2, Z3, Z4]**: The output of the fully connected layers is a vector of scores (logits), one for each class that the network can classify.
- **Softmax Function**: The logits are then transformed by the softmax function, which is represented by the equation below the output vector. This function converts the logits into classification probabilities.

- **Classification Probabilities**: Finally, the probabilities are shown in a box on the right, with the classification categories listed beside them. In this case, the CNN has classified the input image with a high probability as "Meningioma".
- The highlighted probability (0.91 for Meningioma) indicates the CNN's prediction based on the highest softmax score, signifying that the network is highly confident that the input image is of a meningioma [11].

2.3 Advantages of CNNs

- **Automatic Feature Extraction**: CNNs automatically detect important features without any human supervision required for feature extraction, which is a significant advantage over traditional algorithms.
- **Robustness**: They are less preprocessed compared to other image classification algorithms. This means that they can capture invariant features despite variations in the input.
- **Versatility**: Beyond image processing, CNNs have been successfully applied to a variety of tasks such as natural language processing and time series prediction, demonstrating their adaptability.

2.4 Applications of CNNs

Convolutional Neural Networks (CNNs) are one of the greatest breakthrough technologies for automated visual data categorization and comprehension—the skills that promise colossal values across divergent sectors. Most importantly, it has been used effectively in the digital space to provide descriptive labels for the images. Most importantly, they are used for recommendation algorithms that can suggest products or services based on the analysis of visual information. For example, an online retailer may use it to recommend fashion items that would fit a shopper's sense of aesthetics. Another technology used by visual search, and another industry depending on AI, is image classification, which allows one to perform image searches in yet more visual sectors, for example, fashion.

Semantic segmentation uses object detection to label individual pixels of the image. These applications are vast and range from autonomous cars and security surveillance to even medical diagnostics. One such specialization of image recognition is face recognition, wherein this technology finds wide popularity on social networks through photo tagging and is even being adapted in many software applications for identity verification to enhance security features. Another domain in which CNNs are very powerful is optical character recognition (OCR). The methodology based on machine learning and methods to recognize and convert text from several visible sources into digital text representations. This is, of course, revolutionary with regard to document digitalization and further eases how the data gets processed into a digital text format, including ebooks and online databases. The rise and success of CNNs is the perfect exemplification of how deep learning has turned into the leading factor within the extension of artificial intelligence, providing tools able to understand and interpret visual data in an effective manner.

Chapter 3

Recurrent Neural Networks

3.1 Evolution of RNN Architecture

Recurrent Neural Networks (RNNs) have a history that is the mirror of some major milestones in the research on neural networks with regard to sequential data processing. After some seminal models, such as the Hopfield network conceived in the 1980s, RNNs advanced to the introduction of Elman and Jordan networks, the first capable of modeling temporal architectures. One of the big breakthroughs for LSTM came in 1997 when Hochreiter and Schmidhuber first proposed it as a solution to the vanishing gradient problem and, thus, for the first time, dramatically improved learning of dependencies over many discrete time steps. This decade of the 2010s saw RNNs, in particular LSTM and Gated Recurrent Units (GRUs), become indispensable in many revolutionary achievements in speech recognition and natural language processing, all due to their wider computational powers and training methods. Even with the advent of transformer models and attention mechanisms, RNNs remain at the core of AI tools due to the non-comparable power of modeling and property of temporal sequences, reflecting unceasing growth into more elaborate and potent sequence models [12].

3.2 Core Components of RNNs

Recurrent Neural Networks (RNNs) are a specialized form of artificial neural networks designed to handle sequential or time-series data. They play a crucial role in solving time-related problems across various domains, including language translation, natural language processing (NLP), speech recognition, and generating image descriptions. RNNs are integral to the functionality of widely-used technologies such as voice-activated assistants like Siri, voice search features, and translation services like Google Translate. Unlike feedforward and convolutional neural networks (CNNs) that process inputs in isolation, RNNs leverage their inherent "memory" to use information from previous inputs, allowing the network's outputs to be influenced by the sequence's history. This characteristic enables RNNs to model the dependency between sequence elements effectively. However, standard unidirectional RNNs may not predict future events in a sequence, as they process data in a forward direction without considering subsequent inputs that could inform the prediction [13].

3.2.1 Input Layers in Recurrent Neural Networks

In Recurrent Neural Networks (RNNs), input layers serve as the entry points for sequential data to be processed over time. Unlike feedforward neural networks, RNNs maintain a form of memory by reusing the output from previous steps as part of the input for the current step. This allows them to handle inputs of varying lengths and to process temporal dynamics within the data. The input layer in RNNs must be capable of capturing these temporal dependencies, often through the use of specialized cells like LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit) units that regulate the flow of information to remember patterns over long sequences effectively.

3.2.2 Hidden Layers in Recurrent Neural Networks

The hidden layers in RNNs are crucial for processing sequential data, incorporating a mechanism to remember information across timesteps. The key operations involve:

Hidden State Update

The hidden state at timestep t, denoted as h_t, is updated based on the current input x_t and the previous hidden state h_{t-1}, using the formula:

$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

where f is a non-linear activation function such as *tanh* or *ReLU*, W_{hh} are the weights for the hidden-to-hidden connections, W_{xh} are the weights for the input-to-hidden connections, and b_h is the bias term for the hidden layer.

Output Calculation

The output at timestep t, denoted as y_t, is calculated from the current hidden state using:

$$y_t = g(W_{hy}h_t + b_y)$$

where g is an activation function, often *softmax* for classification tasks, W_{hy} are the weights for the hidden-to-output connections, and b_y is the output bias term [14].

Backpropagation Through Time (BPTT)

Training RNNs involves unfolding them through time and applying backpropagation, a process known as Backpropagation Through Time (BPTT), described by:

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

where L is the loss function, T is the sequence length, and W represents the model weights. This process accounts for the temporal sequence of operations in the gradient

calculation.

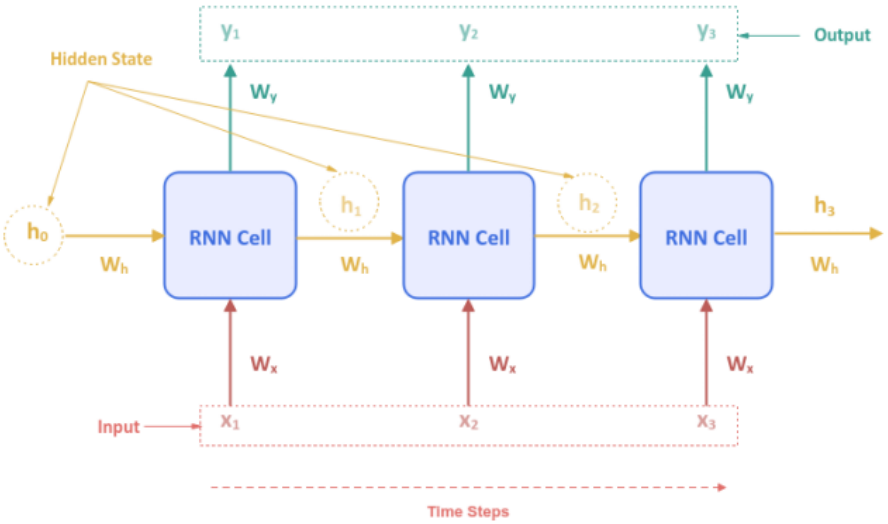

Figure 3.1. *Recurrent Neural Network unfolded over time for a sequence of 3 inputs.*

3.3 Advantages & Limitations of RNNs

Recurrent Neural Networks (RNNs) are recognized for their dynamic structure, which renders them a potent tool for a variety of temporal processing tasks. They are particularly noted for their computational strength and broad applicability across numerous temporal modeling applications. An essential attribute of RNNs is their universal approximation capability, allowing them to model a wide range of nonlinear dynamic systems with a high degree of precision. This is achieved through the network's ability to establish complex relationships between sequences of inputs and outputs. However, RNNs come with certain drawbacks, such as the challenges in practical implementation despite their theoretical prowess. A significant obstacle faced during training RNNs is the vanishing gradient problem, which can impede the learning of long-range dependencies within data sequences. Additionally, the inherent nonlinearity and the complexity involved in adjusting the weights can lead to difficulties in maintaining the stability of the network [15].

3.4 Regularization

Regularization in machine learning and statistics is a technique used to prevent overfitting by imposing penalties on model parameters. Overfitting occurs when a model learns the training data too well, capturing noise along with the underlying patterns, which can negatively affect its performance on new, unseen data. Regularization techniques, such as L1 (Lasso) and L2 (Ridge) regularization, add a complexity penalty to the model's loss function, which constrains the model's coefficients. This can encourage sparser models with fewer parameters or ensure that the model's parameters remain small, making the model simpler and less likely to overfit. Regularization is a cornerstone of effective model training, allowing for models that generalize better to new data.

3.4.1 Data augmentation

Data augmentation is a strategy used to enhance the diversity of training data without collecting new samples. It artificially inflates the dataset by introducing variations of existing data, which helps models generalize better. This is especially useful in addressing imbalances within datasets [16].

3.4.2 Early stopping

Early stopping, a different regularization approach, halts training when performance on validation data stops improving, preventing overfitting [16]. The graph on figure 3.2 illustrates the concept of early stopping in training machine learning models. It plots training and validation loss (error) over the number of training epochs. The blue line represents the training loss, which decreases steadily as training progresses, indicating that the model is learning from the training data. The red line represents the validation loss, which also decreases initially but begins to rise again, signaling overfitting. The point where the validation loss stops decreasing and starts to increase is marked for early stopping. This is where the training should be halted to prevent the model from learning the noise in the training data, thereby generalizing better to new data.

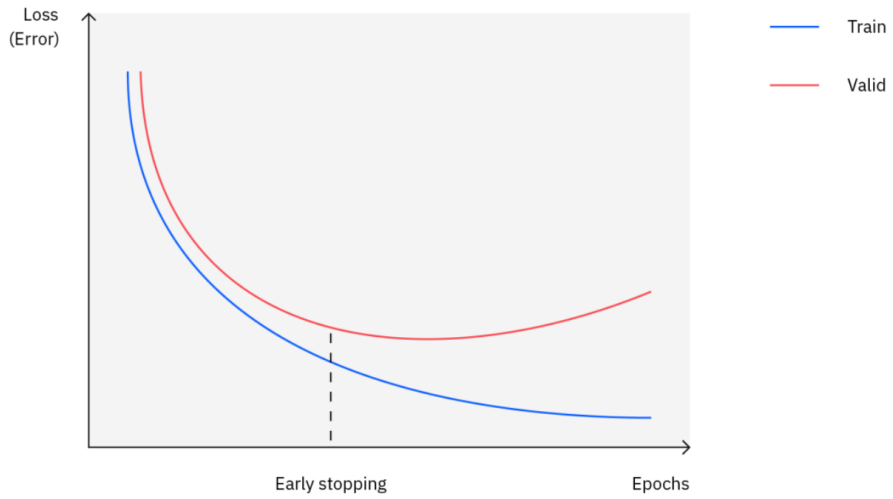

Figure 3.2. *Early stopping.*

3.4.3 Dropout

Neural networks, structured with input, hidden, and output layers, benefit from techniques like dropout and weight decay. Dropout temporarily removes nodes during training to encourage a robust network, simulating an ensemble of networks [16]. On the left of the figure 3.3 is a fully connected network with one input layer, multiple hidden layers, and an output layer. All nodes are active and connected. On the right, dropout has been applied, randomly deactivating nodes (marked with an 'X') and their connections. This

process thins the network, preventing co-adaptation of nodes and promoting generalization. The resultant network simulates training diverse architectures, leading to a model less prone to overfitting and with improved robustness to varied input data.

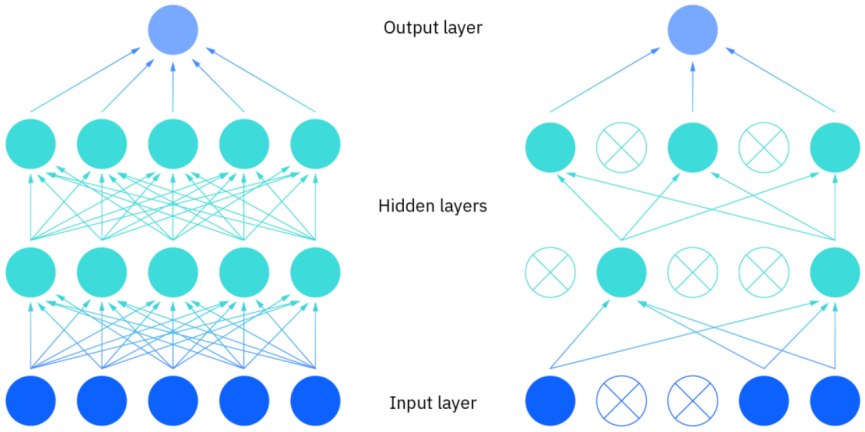

Figure 3.3. *Dropout.*

3.4.4 Weight decay

Weight decay, similar to L1 regularization, encourages sparsity in the network by driving some weights to zero, simplifying the model and mitigating over-complexity. While dropout affects the network's depth, weight decay operates linearly, offering different benefits in combating overfitting [16].

3.5 Optimizers

In machine learning, optimizers are algorithms or methods used to change the attributes of the neural network, such as weights and learning rate, to reduce the losses. Optimizers leverage the gradient (derivative) of the loss function to navigate the complex, high-dimensional space of model parameters towards a region that minimizes the loss. The choice of optimizer can significantly influence the speed and quality of the training process, as well as the ability of the model to converge to a good solution.

3.5.1 Gradient Descent

Gradient descent is a fundamental optimization technique in machine learning, utilized to refine model parameters and reduce the cost function. This iterative method enhances the model's accuracy by adjusting parameters against the gradient of the cost function, which gauges the model's error or loss.

The cost function quantifies the difference between the model's predictions and the actual data. The aim of gradient descent is to identify the parameter values that minimize this error, thereby enhancing the model's predictive accuracy.

In practice, gradient descent evaluates the slope of the cost function at a given point and proceeds in the reverse direction—toward the negative of the gradient. This is because

moving against the gradient leads to a decrease in the cost function's value. The size of the steps taken during each iteration of gradient descent is governed by the learning rate. This hyperparameter is crucial as it affects how quickly the algorithm converges to a minimum, with too large a learning rate potentially causing overshooting and too small leading to slow convergence, figure 3.4.

Gradient descent's versatility allows it to be implemented across a range of machine learning models, from simple linear regression to complex neural networks, making it a versatile tool for optimizing a wide array of predictive algorithms through systematic parameter adjustments [17].

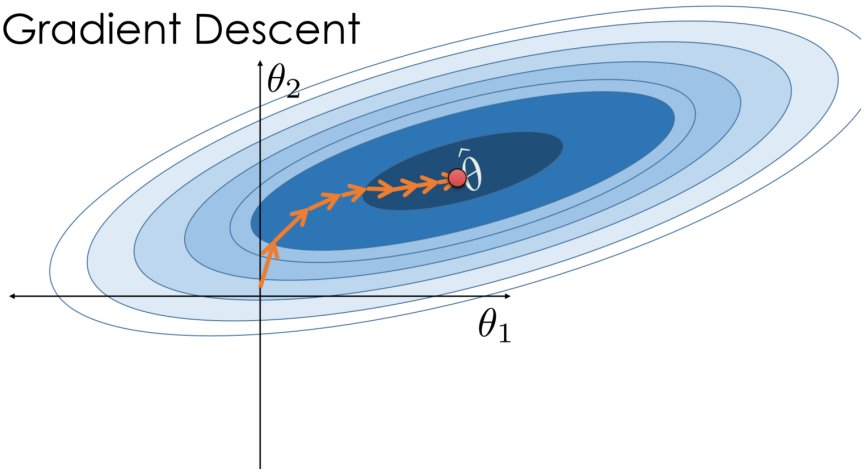

Figure 3.4. *Gradient Descent Algorithm.*

The update rule of gradient descent is given by:

$$\partial_{\text{new}} = \partial_{\text{old}} - a\nabla_{\partial} J(\partial)$$

where:

- ∂ represents the parameters of the model,
- a is the learning rate,
- $\nabla_{\partial} J(\partial)$ is the gradient of the cost function $J(\partial)$ with respect to the parameters ∂.

This rule is applied iteratively to minimize the cost function $J(\partial)$, leading to an improvement in the model's performance.

3.5.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a variant of the gradient descent optimization method tailored for machine learning tasks. It streamlines the optimization process by updating model weights incrementally, using a single randomly selected data point at each iteration, rather than the entire dataset. This approach allows SGD to navigate the hypothesis space, which is useful when dealing with continuously parameterized

hypotheses and differentiable error functions. While traditional gradient descent can be slow and may get stuck in local minima, SGD enhances efficiency by reducing the time required for convergence and is not as prone to overfit to the training data due to its stochastic nature. This makes it particularly effective for large-scale data, which has become increasingly common. SGD has been instrumental in training neural networks, significantly cutting down computational time and catering to complex problems without compromising on model performance [18].

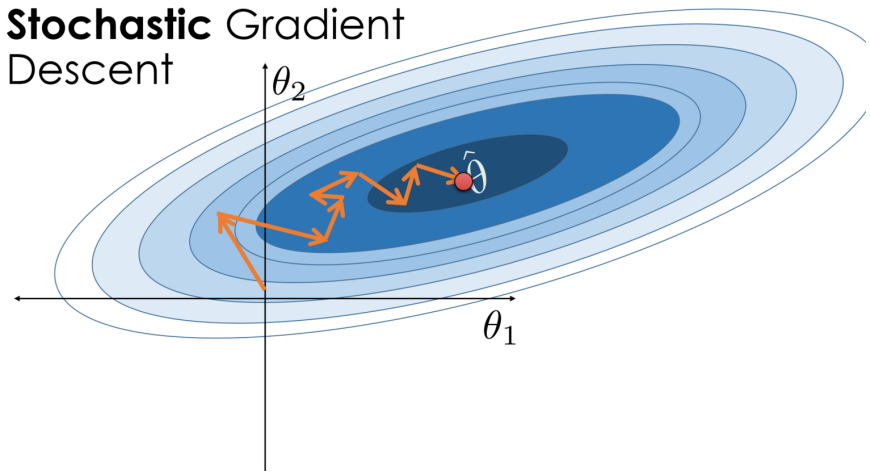

Figure 3.5. *Stochastic Gradient Descent Algorithm.*

3.5.3 Mini-batch Gradient Descent

Mini Batch Gradient Descent is like a cross-over approach between GD and SGD. It deals with first dividing the dataset into small datasets (batches) instead of iterating through the whole dataset or one observation, and for each batch, the gradients are calculated [19].

Given a cost function $J(\partial)$, where ∂ represents the model parameters, the update rule for Mini Batch Gradient Descent at iteration t is given by:

$$\partial_{t+1} = \partial_t - a\nabla_{\partial} J(\partial_t; X^{(i:i+n)}, y^{(i:i+n)})$$

where:

- ∂_t is the value of the parameters at iteration t.
- a is the learning rate.
- $\nabla_{\partial} J(\partial_t; X^{(i:i+n)}, y^{(i:i+n)})$ is the gradient of the cost function with respect to the parameters, evaluated on the mini-batch from i to $i+n$.
- $X^{(i:i+n)}$ and $y^{(i:i+n)}$ represent the input features and targets of the mini-batch, respectively.

This method balances the speed of SGD with the reduced variance in the gradient estimates, leading to more stable and reliable convergence towards the minimum of the cost function.

3.5.4 Momentum

In the context of machine learning and particularly neural network training, momentum is a technique used to accelerate the convergence of the gradient descent optimization algorithm. It helps to navigate the relevant cost landscape more efficiently, preventing oscillations and speeding up convergence by adding a fraction of the previous update vector to the current update. Essentially, it combines the gradient (a measure of how much the loss would change if the model parameters were modified) with the momentum from previous steps. This can prevent the optimization from getting stuck in local minima and provides a more stable and faster convergence towards the global minimum of the loss function.

3.5.5 Adagrad

AdaGrad is a gradient-based optimization algorithm that dynamically adapts the learning rate for each parameter, enhancing efficiency in ML and DL models. Introduced in 2011, it adjusts learning rates based on the accumulation of squared gradients, decreasing rates for frequently updated parameters and increasing for less frequent ones, promoting faster convergence and better handling of data sparsity. However, AdaGrad may suffer from an over-accumulation of gradients, causing the learning rate to diminish too much, potentially halting learning. This limitation is addressed by newer algorithms like Adam, which regulate the accumulation to ensure continuous learning progress [20].

The update rule for the AdaGrad optimization algorithm can be represented mathematically as follows:

$$\partial_{t+1,i} = \partial_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

Where:

- $\partial_{t,i}$ represents the parameter at time step t.
- η is the initial learning rate.
- $G_{t,ii}$ is the sum of the squares of the past gradients with respect to $\partial_{t,i}$ up to time step t.
- ϵ is a small constant added to improve numerical stability.
- $g_{t,i}$ is the gradient at time step t.

3.5.6 Adam

Adam is an optimization algorithm that efficiently uses first-order gradients and requires minimal memory. It calculates adaptive learning rates for each parameter by esti-

mating the moments of gradients, hence the name 'adaptive moment estimation.' Adam blends the strengths of AdaGrad, which excels with sparse gradients, and RMSProp, which is effective for online and non-stationary problems. It offers several benefits: it is unaffected by gradient scaling, imposes stepsize bounds, adapts to non-stationary objectives, handles sparse gradients, and integrates a form of stepsize annealing [21].

The Adam optimization algorithm's update rule is typically represented by the following equations:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

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

Where:

- g_t is the gradient at timestep t.
- m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients, respectively.
- $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected versions of m_t and v_t.
- β_1 and β_2 are the exponential decay rates for these moment estimates.
- θ_t is the parameter vector at timestep t.
- η is the stepsize.
- ϵ is a small scalar used to prevent division by zero.

3.5.7 RMSprop

RMSprop, which stands for Root Mean Square Propagation, is an adaptive learning rate optimization algorithm [22] designed to address some of the issues encountered with the traditional stochastic gradient descent (SGD) method in training artificial neural networks. It was first proposed by Geoff Hinton in his Coursera class on neural networks and is widely used for deep learning applications.

Key Features of RMSprop

- **Adaptive Learning Rates**: Unlike SGD, which maintains a constant learning rate throughout the training process, RMSprop adjusts the learning rate for each parameter dynamically. It makes the learning rate smaller for parameters associated with frequently occurring features and larger for parameters associated with infrequent features.
- **Gradient Squaring**: RMSprop works by keeping a moving average of the squared gradients for each weight. This square gradient emphasizes the influence of gradients with larger magnitudes.
- **Normalization**: The algorithm divides the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. This normalization balances the step sizes, making the optimization process less sensitive to the scale of the gradients.

The running average of the squared gradients is computed by:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2$$

where:

- $E[g^2]_t$ is the expectation of the squared gradients at time step t.
- γ is the decay rate.
- g_t is the gradient at time step t.

The weights are then updated using the equation:

$$\partial_{t+1} = \partial_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

where:

- ∂_{t+1} is the updated weight.
- ∂_t is the current weight.
- η is the learning rate.
- ϵ is a small number to prevent division by zero, typically around $1e - 8$.

Benefits of RMSprop

Stabilizes the Learning Path: By adjusting the learning rate, RMSprop avoids aggressive weight updates that can destabilize the learning process.

- **Improves Convergence**: It tends to converge faster and with less tuning of the learning rate compared to SGD.

- **Effective for Recurrent Neural Networks**: RMSprop has proven effective in training RNNs, which are sensitive to the choice of the learning rate due to the vanishing and exploding gradient issues.

RMSprop has become a go-to optimization technique in various deep learning tasks, particularly useful when dealing with complex models and non-convex optimization landscapes.

3.6 Metrics

In machine learning, metrics [23] are essential for evaluating and comparing the performance of models. They provide quantitative measures to assess how well a model's predictions match the actual data.

3.6.1 Accuracy

Accuracy is a fundamental metric in the field of machine learning and statistics, used to measure the overall correctness of a model's predictions. It is defined as the ratio of correctly predicted observations (both true positives and true negatives) to the total number of observations in the dataset.

Mathematically, it can be expressed as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Accuracy is particularly straightforward to understand and communicate, making it a popular choice for evaluating model performance in classification tasks. However, its simplicity can also be a limitation, especially in cases of imbalanced datasets where one class significantly outnumbers the other. In such scenarios, a model could achieve high accuracy by simply predicting the majority class for all instances, but it would fail to capture the nuances and potentially critical patterns of the minority class.

For example, in a medical diagnosis application where the dataset contains 95% negative (healthy) cases and only 5% positive (disease) cases, a model that predicts 'negative' for all cases would achieve 95% accuracy. Despite the high accuracy, this model would be practically useless since it fails to identify any of the positive cases, which are usually of more significant interest.

Thus, while accuracy is a valuable metric for providing a quick overview of model performance, it is essential to complement it with other metrics like precision, recall, and the F1 score, especially in cases of class imbalance. These additional metrics can provide a more nuanced understanding of a model's strengths and weaknesses, ensuring a comprehensive evaluation of its performance.

3.6.2 F1

The F1 score is a crucial metric in the evaluation of classification models, especially in scenarios where there are imbalanced classes or when the cost of false positives and false negatives varies significantly. It is the harmonic mean of precision and recall, two metrics

that assess the accuracy of a model's positive predictions and its ability to identify all actual positives, respectively.

The formula for the F1 score is:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

where:

- **Precision** is the ratio of true positive predictions to the total positive predictions (including both true positives and false positives).
- **Recall** also known as sensitivity, measures the ratio of true positive predictions to the actual positive cases in the dataset (the sum of true positives and false negatives).

The F1 score ranges from 0 to 1, where a higher score indicates a better balance between precision and recall. A score of 1 signifies perfect precision and recall, whereas a score of 0 indicates that the model fails either in precision or recall.

This metric is particularly valuable because it accounts for both the false positives and false negatives in its calculation, providing a more nuanced view of a model's performance than accuracy alone, especially in datasets where positive cases are rare or in applications where false negatives carry a higher risk than false positives (or vice versa). The F1 score is widely used in binary classification tasks, including document classification, spam detection, and disease diagnosis, where making a correct positive prediction is critical.

3.6.3 Precision

Precision, also known as the positive predictive value, is a key metric in the evaluation of classification models, particularly in the context of binary classification problems. It measures the proportion of true positive predictions in relation to the total number of positive predictions made by the model. In simpler terms, precision answers the question: "Of all the instances the model labeled as positive, how many were actually positive?"

The formula for precision is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

where:

- **True Positives (TP)** are the instances correctly identified as positive by the model.
- **False Positives (FP)** are the instances wrongly identified as positive by the model.

Precision is particularly important in situations where the cost of a false positive is high. For instance, in email spam detection, a false positive (marking a legitimate email as spam) could mean missing an important message, so a high precision model would be preferable to minimize this risk.

However, precision alone does not provide a complete picture of a model's performance. It does not take into account the false negatives (positive instances missed by the model).

Therefore, precision is often used in conjunction with recall (sensitivity), which measures the proportion of actual positives correctly identified by the model. Balancing precision and recall is crucial in many real-world applications, and metrics like the F1 score are used to find a harmonic balance between these two metrics.

3.6.4 Recall

Recall, also recognized as sensitivity or the true positive rate, is a fundamental metric for assessing the effectiveness of classification models, especially when the identification of all actual positives is critical. Recall quantifies the fraction of actual positive instances that the model accurately identifies from the entire set of true positive instances available. It essentially asks, "Of all the instances that are truly positive, how many were successfully detected by the model?"

The mathematical expression for recall is as follows:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

where:

- **True Positives (TP)** refer to the instances that the model correctly predicts as positive.
- **False Negatives (FN)** denote the positive instances that the model incorrectly classifies as negative.

Recall is particularly important in fields such as medical diagnosis or fraud detection, where missing a positive case (such as failing to identify a disease or a fraudulent transaction) could have serious consequences. High recall indicates that the model is capable of capturing most of the positive cases, which is desirable in these sensitive applications.

However, optimizing for recall alone may lead to an increase in false positives, as the model might label more instances as positive to ensure it misses fewer actual positives. Therefore, recall is often used in conjunction with precision to provide a more balanced view of the model's performance. Balancing these two metrics is crucial for developing effective classification models, especially in situations where both identifying all positives and maintaining accuracy in positive predictions are important.

3.6.5 AUROC

The Area Under the Receiver Operating Characteristic Curve (AUROC) serves as a critical metric for assessing the effectiveness of classification models by evaluating their capacity to accurately prioritize examples. In the context of a clinical risk prediction model, AUROC reflects the likelihood that a patient who underwent an event is assigned a higher risk score by the model compared to a patient who didn't experience such an event. Similarly, for a model classifying handwritten digits "1" and "0," AUROC indicates the chance that an image labeled "1" is deemed more likely to be a "1" by the model than an image labeled "0." Essentially, AUROC assesses a model's "discrimination" ability, or

its capability to distinguish between positive and negative examples. An AUROC value of 0.8 signifies strong discrimination power, implying that in 80% of cases, the model will correctly differentiate a patient with an event from one without based on the assigned risk.

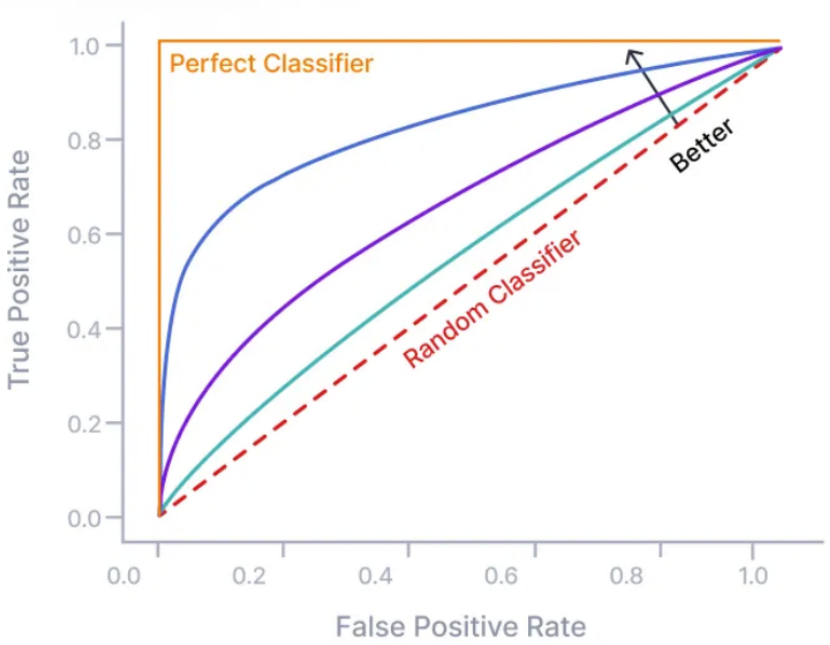

Figure 3.6. *ROC curves.*

The above diagram, 3.6, depicted showcases various ROC curves, with the AUROC (Area Under the Receiver Operating Characteristic Curve) being the space underneath each curve. An AUROC score can range from 0.5, indicative of no predictive ability similar to random guessing, to 1.0, which signifies perfect prediction accuracy.

- An AUROC of 0.5, represented by the red dashed line, equates to the predictive power of flipping a coin, essentially marking a model as ineffective.
- Scores below 0.7 suggest the model's performance is not optimal.
- A score within the 0.70 to 0.80 range is considered to indicate good model performance.
- Scores exceeding 0.8 denote excellent predictive capability.
- A perfect score of 1.0, illustrated by the purple line, denotes an ideal classifier that makes no errors in prediction.

Overall, the AUROC represents the area covered by the ROC curve, which plots the relationship between the true positive rate (TPR) and the false positive rate (FPR) at various threshold settings [24].

3.6.6 Confusion Matrix

Confusion matrices are tools used to evaluate the performance of a classification model by analyzing its predictions on a dataset. They provide insight into how well the model performs, highlighting its accuracy in distinguishing between different classes. A confusion matrix is typically generated from a model's predictions on a validation or test set, not seen by the model during training [25].

In a confusion matrix:

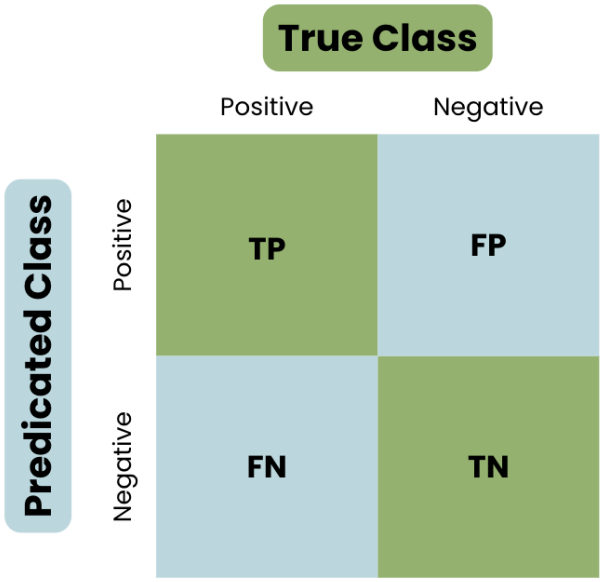

Figure 3.7. *Confusion Matrix.*

The columns labeled "Actually Positive" and "Actually Negative" represent the actual, true labels of the data. These labels could denote whether a digit is genuinely a 1 or 0, if a patient genuinely has a specific disease or not, or whether a chest x-ray truly indicates pneumonia, among other examples.

The rows labeled "Predicted Positive" and "Predicted Negative" capture the model's predictions, indicating whether it perceives the examples as positive or negative.

The elements within the confusion matrix—True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN)—are numerical counts that categorize the predictions:

- True Positives (TP) count the instances correctly identified as positive by the model.
- True Negatives (TN) count the instances correctly identified as negative.
- False Positives (FP) represent the negative instances that were incorrectly labeled as positive by the model.
- False Negatives (FN) tally the positive instances that the model mistakenly labeled as negative.

Understanding these metrics allows for a comprehensive evaluation of a model's predictive capabilities and its potential strengths and weaknesses, facilitating a comparison between different models to select the most suitable one for a specific application.

Part II

Practical Part

Chapter 4

Implementation

4.1 Tools & Libraries

4.1.1 Google Colab

Google Colab is a free, cloud-based platform that facilitates machine learning and data science research by providing a collaborative environment for Python programming. It's ideal for a wide range of applications from machine learning to data analysis, offering zero configuration access to advanced computing resources like GPUs and TPUs. Integrated with Google Drive, it allows for easy storage and sharing of Jupyter notebooks, and comes preloaded with many popular data science libraries, streamlining workflow and education. Colab's browser-based, system-agnostic design ensures accessibility, removing the need for local computational resources and democratizing access to cutting-edge AI tools [26]. A challenge of the present thesis is the modeling using limited resources.

4.1.2 Pytorch Lightning

PyTorch Lightning serves as an efficient wrapper for the PyTorch deep learning framework, offering a simplified workflow for researchers and developers. Its primary objective is to separate the scientific code from the engineering backend, significantly reducing the amount of boilerplate code and bringing clarity to the research process. This framework structures the training code in a way that enhances readability and maintainability, which is particularly beneficial for complex models. One of the standout features of PyTorch Lightning is its emphasis on reproducibility, a vital requirement in scientific research. It ensures that experiments can be easily repeated and verified by others. Moreover, the framework is designed with scalability in mind, enabling models to be effortlessly scaled to run on various hardware configurations, including multiple GPUs and TPUs, as well as across different nodes.

PyTorch Lightning boasts seamless integration with a suite of popular libraries and tools within the machine learning landscape, fostering a robust and flexible environment for model development and tracking. This integration extends to tools for experiment tracking, lifecycle management, and more, which streamlines the development pipeline from research to production. The framework is supported by an active community of contributors, ensuring continuous evolution and the incorporation of cutting-edge fea-

tures. Despite its structured approach, PyTorch Lightning provides researchers with the flexibility to override and tailor specific aspects of the training process to fit their unique requirements.

In summary, PyTorch Lightning is a powerful tool that abstracts away much of the complexity associated with deep learning model training, enabling developers and researchers to focus on the core aspects of their work. Its design principles promote clean code, reproducibility, and ease of scaling, making it an excellent choice for those seeking to accelerate their deep learning projects [27]. PyTorch Lightning is used for the purposes of this diploma thesis.

4.1.3 NumPy

NumPy, standing for Numerical Python, is a foundational Python library for scientific computing, offering robust support for large, multi-dimensional arrays and matrices, alongside a plethora of mathematical functions to efficiently operate on these structures. Central to NumPy is its ndarray object, optimized for high-speed operations and designed for array-oriented computing, making it indispensable for a wide array of mathematical tasks, from linear algebra to statistical analyses. Its capability for broadcasting allows for flexible arithmetic operations on arrays of differing shapes. Widely integrated across scientific computing libraries and applications in domains ranging from image processing to machine learning, NumPy is bolstered by a vast community, ensuring extensive documentation and support. This combination of performance, versatility, and community support cements NumPy's role as a critical tool in data analysis and scientific research [28].

4.1.4 Matplotlib

Matplotlib is an essential library in the Python data science ecosystem, renowned for its ability to generate a wide array of static, animated, and interactive visualizations. With its comprehensive suite of plotting functions and vast customization options, Matplotlib enables users to craft detailed graphs, charts, and figures. This versatility makes it an invaluable tool for data analysts and researchers who need to visualize complex datasets and analytical results. Whether for exploratory data analysis or presenting findings, Matplotlib's robust functionality ensures that users can convey their data in visually compelling and informative ways [29].

4.1.5 Python Package eeg_positions

The eeg_positions Python package enables the computation and visualization of standard EEG electrode placements on a spherical head model, adhering to the 10-20, 10-10, and 10-05 systems. It offers functionality to approximate electrode locations for EEG data analysis, providing a solution to the often undocumented calculation of these positions found in various resources. Utilizing a 3D "RAS" coordinate system, the package considers anatomical landmarks (nasion, left and right preauricular points, vertex, and inion)

to define the electrode coordinates accurately. Users can easily install the package via pip, access pre-computed electrode positions in the repository's data directory, and utilize utility functions for projecting 3D locations into 2D space for plotting. This tool not only facilitates accurate electrode positioning for topographical plotting and sensor location visualization but also offers customization options for specific research needs. Also, the appropriate repository, could be found at this link [30].

4.2 Challenges

In order to carry out the present diploma thesis, it is necessary to overcome some significant challenges. One of these is modeling given limited computational resources in the most optimal way. Concurrently, the type of data, images that have emerged from respective electroencephalograms, represents a quite demanding data type as it requires not only the appropriate knowledge for preprocessing the images in a broader handling approach, but also specific expertise related to the preprocessing of images concerning brain mapping. Naturally, the development of corresponding models to be applied, through which the desired outcome will be achieved - which is none other than the classification between real and imaginary movement, is an equally important challenge.

Moreover, one challenge that arises is the imbalance in the dataset. Dataset imbalance represents a situation where there is an uneven distribution of the number of examples from classes in a dataset, therefore skewing model training. This usually results in a model that is either biased towards the majority class or its performance is compromised on the minority class, which might be a critical class in many applications. Such models might show deceptively high accuracy, rather reflecting the class prevalence of the majority class than showing its true predictive power. This may largely decrease the sensitivity of the model to the minority class, resulting in a large quantity of false negatives for the less represented class. That is, it would often be the case that there would be severe consequences of this kind of model during fraud detection or disease diagnosis—handling adequately of imbalanced datasets during the process of model development.

4.3 Data Collection

The topographic brain map, as it is shown in figure 4.1, is an illustration in which electrical activity or functional imaging data of the brain are projected onto an illustration of the brain or a standard template. Such maps are commonly derived from data that are in turn derived from electroencephalography (EEG), functional magnetic resonance imaging (fMRI), or other neuroimaging techniques. It is useful to illustrate spatial patterns of activity across the brain, such as distribution over the scalp of electrical potentials in EEG, or of areas of increased blood flow in fMRI, which correspond with underlying patterns of neural activity. In that sense, the topographic map helps in identifying which parts of the brain are more active or are involved in carrying out a task, a state of consciousness, or any other conditions related to the processing of certain kinds of stimuli for researchers and clinicians. They have been of great help in science, especially neuroscience and psychology, to assist in some areas such as furthering brain function, diagnosis of neurological disorders, and also for development in brain-computer interfaces.

In our case, the data used in this thesis are topographic maps of the brain with dimensions - (3, 656, 875), which have been derived from electroencephalograms. The experiment conducted involved 33 participants, who were appropriately fitted with an electrode cap to capture their brain activity during the experiment. Specifically, they were asked to either perform or simply imagine the movement of their lower limbs: left foot, right foot, both, or none. Data were recorded for each experiment (real or imaginary movement of the lower limbs), with 20 temporal events for each participant. In relation to each of these events, 50 topographic maps are displayed in which each map represents the illustration of a frequency originating from the participant's electroencephalogram (1-50Hz).

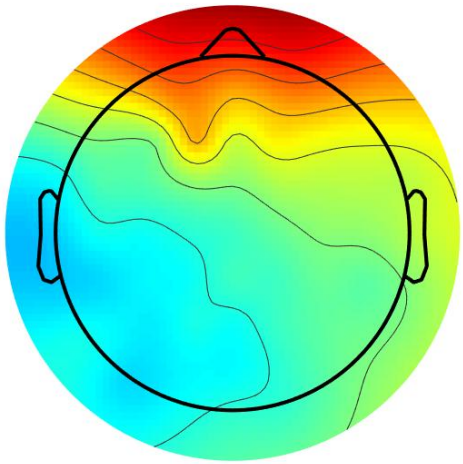

Figure 4.1. *Topographic Map.*

4.4 Preprocessing

The first and important step toward processing the data set: appropriately labeling the data set. Basically, an ensemble method will be used, where three weak learners will be combined, and each of them classifies the dataset into different subsets. In order to make the above process comprehensive, the following diagram is forwarded along with the respective label and the classifier used.

4.4.1 Ensemble Method

The diagram, in figure 4.2, depicts a decision process within a neural network using Recurrent 3D Convolutional Neural Networks (R3DCNN). The process begins with an R3DCNN that determines whether there is any leg movement. If no movement is detected, the decision branches off to indicate 'No movement'. If leg movement is detected, another R3DCNN takes over to further assess the movement. If the movement involves both legs, yet another R3DCNN is employed to differentiate between right and left leg movement. This hierarchical approach allows for a nuanced analysis of leg movements, potentially

useful in applications like gait analysis or physical therapy rehabilitation monitoring. Finally, the use of the ensemble method aims to eliminate the unbalance dataset after the appropriate labeling that is performed.

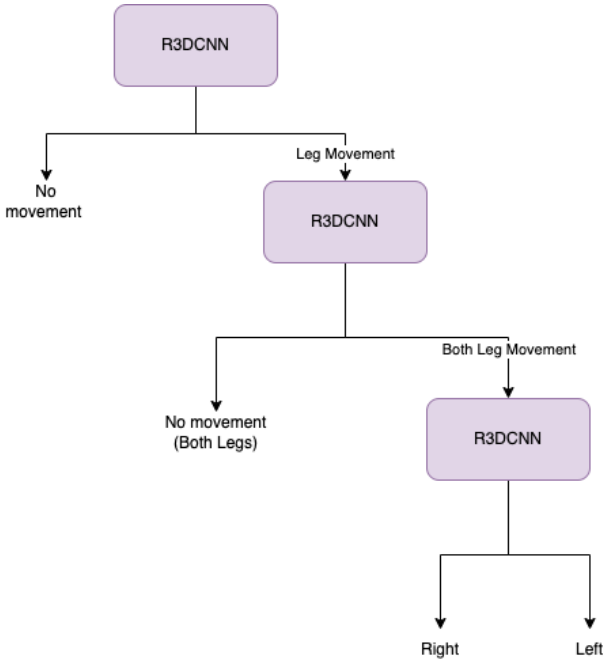

Figure 4.2. *Diagram of Ensemble Classifier.*

To adapt the model to new data and achieve better generalization, the technique of data augmentation is used, where images are transformed into grayscale. Additionally, the technique of interpolation (Rbf) is used, wherein after the precise points of the electrodes on the scalp have been determined (using the eeg_positions library, system 10-20), resizing is performed converting the original tensor from (50, 1, 656, 875) - (sequence, channel, height, width) to (50, 1, 80, 80).

4.5 Training

4.5.1 CNN - GRU

The R3DCNN model is a deep learning framework designed to interpret topographical maps derived from EEG data for the assessment of mental workload. This model integrates recurrent and convolutional neural networks to learn from the spatial, spectral, and temporal dimensions of EEG signals, which are represented in the form of 3D topographic maps.

The topographical maps are transformed into a 3D structure, with the spatial and spectral characteristics derived from EEG channels and frequency bands, respectively. The model is trained on sequences of these 3D EEG cubes, allowing the recurrent neural network part to capture temporal dynamics and the convolutional part to extract spatial-spectral features.

This approach provides a comprehensive understanding of EEG data, tapping into the complex patterns associated with different cognitive states. The R3DCNN model's ability

to accurately classify mental workload levels without the need for hand-crafted features is particularly valuable, potentially offering a robust tool for real-time applications where swift and accurate assessment of mental states is crucial.

Model

Table 4.1. *3D CNN Architecture*

Layer	In Channels	Out Channels	Kernel Size	Stride/Padding
Conv3d	1	32	$3 \times 3 \times 3$	1/1
ReLU	-	-	-	-
MaxPool3d	-	-	$1 \times 2 \times 2$	$1 \times 2 \times 2$
Conv3d	32	64	$3 \times 3 \times 3$	1/1
ReLU	-	-	-	-
MaxPool3d	-	-	$1 \times 2 \times 2$	$1 \times 2 \times 2$
Conv3d	64	128	$3 \times 3 \times 3$	1/1
ReLU	-	-	-	-
MaxPool3d	-	-	$1 \times 2 \times 2$	$1 \times 2 \times 2$

4.6 Evaluation

RMSprop

The four experiments from tables 4.3-4.9 are shown the performance of the R3DCNN model with different learning rates using the RMSprop optimizer.

- **Learning Rate 0.01 (First Table):** The weighted metrics demonstrate moderate performance with an Accuracy of 59.74%, an F1-Score of 0.318, and an AUROC of 0.5032. This indicates that the learning rate might be too high, leading to potentially suboptimal convergence.
- **Learning Rate 0.001 (Second Table):** Lowering the learning rate to 0.001 leads to improved performance, with the Accuracy increasing slightly to 59.02%, the F1-Score jumping to 0.5839, and the AUROC to 0.5168. This suggests that a smaller learning rate begins to improve the model's ability to generalize.
- **Learning Rate 0.0001 (Third Table):** Further decreasing the learning rate to 0.0001 shows mixed results. The Accuracy improves significantly to 63.77%, the best among all four experiments, while the F1-Score and AUROC see slight decreases to 0.5163 and 0.5146, respectively.
- **Learning Rate 0.00001 (Fourth Table):** The lowest learning rate of 0.00001 yields the best overall weighted results, with a marked increase in all metrics. The Accuracy soars to 74.43%, the F1-Score to 0.6888, and the AUROC to 0.7403, indicating that the R3DCNN model achieves its best performance at this learning rate.

In conclusion, the weighted (which has been resulted as the average value of the three classifiers) classifier performance metrics suggest that the R3DCNN model benefits from a

lower learning rate, with 0.00001 being the optimal in this set of experiments. It achieves the highest scores in terms of the ability to correctly identify classes (Accuracy), the balance between precision and recall (F1-Score), and the model's discriminative ability (AUROC). This could indicate that the model as a whole responds better to finer adjustments during training, leading to a more robust generalization.

Parameter	**Value**
Number of Epochs	10
Batch Number	64
Learning Rate (lr)	**0.01**
Alpha	0.99
Epsilon (eps)	1×10^{-8}
Weight Decay	1×10^{-6}
Momentum	0.9
Centered	True

Table 4.2. *RMSprop Optimizer Parameters*

Metric	**Classifier 1**	**Classifier 2**	**Classifier 3**	**Weighted**
F1-Score	0.1871	0.1774	0.5895	**0.318**
Precision	0.7575	0.3401	0.4838	**0.5271**
Recall	0.1086	0.1263	0.7662	**0.3337**
AUROC	0.5131	0.5175	0.4790	**0.5032**
Accuracy	50.52%	63.12%	65.59%	**59.74%**

Table 4.3. *RMSprop Optimizer - lr 0.01*

Parameter	**Value**
Number of Epochs	10
Batch Number	64
Learning Rate (lr)	**0.001**
Alpha	0.99
Epsilon (eps)	1×10^{-8}
Weight Decay	1×10^{-6}
Momentum	0.9
Centered	True

Table 4.4. *RMSprop Optimizer Parameters*

Metric	**Classifier 1**	**Classifier 2**	**Classifier 3**	**Weighted**
F1-Score	0.7531	0.3517	0.6470	**0.5839**
Precision	0.7597	0.3360	0.4871	**0.5276**
Recall	0.7522	0.3762	0.9729	**0.7004**
AUROC	0.5236	0.5200	0.5069	**0.5168**
Accuracy	63.02%	55.10%	58.95%	**59.02%**

Table 4.5. *RMSprop Optimizer - lr 0.001*

Parameter	Value
Number of Epochs	10
Batch Number	64
Learning Rate (lr)	**0.0001**
Alpha	0.99
Epsilon (eps)	1×10^{-8}
Weight Decay	1×10^{-6}
Momentum	0.9
Centered	True

Table 4.6. *RMSprop Optimizer Parameters*

Metric	Classifier 1	Classifier 2	Classifier 3	Weighted
F1-Score	0.7472	0.2497	0.5272	**0.5163**
Precision	0.7591	0.3157	0.4744	**0.5164**
Recall	0.7378	0.2195	0.5970	**0.5181**
AUROC	0.5371	0.5199	0.4868	**0.5146**
Accuracy	62.70%	68.95%	59.68%	**63.77%**

Table 4.7. *RMSprop Optimizer - lr 0.0001*

Parameter	Value
Number of Epochs	10
Batch Number	64
Learning Rate (lr)	**0.00001**
Alpha	0.99
Epsilon (eps)	1×10^{-8}
Weight Decay	1×10^{-6}
Momentum	0.9
Centered	True

Table 4.8. *RMSprop Optimizer Parameters*

Metric	Classifier 1	Classifier 2	Classifier 3	Weighted
F1-Score	0.9503	0.5785	0.5378	**0.6888**
Precision	0.9176	0.5192	0.4823	**0.6397**
Recall	0.9862	0.6635	0.6123	**0.7540**
AUROC	0.9483	0.7321	0.5028	**0.7403**
Accuracy	92.29%	68.85%	62.15%	**74.43%**

Table 4.9. *RMSprop Optimizer - lr 0.00001*

Adam

In the experiment 4.11 using the Adam optimizer, the weighted metrics across different classifiers can be summarized as follows:

- **F1-Score:** The weighted F1-Score is 0.5627, which is a balance between precision and recall. This score suggests a moderate performance in terms of the harmonic mean between these two metrics.
- **Precision:** The weighted precision is 0.5909, indicating that, on average, the model

has a fair probability of not labeling a negative sample as positive.

- **Recall:** The weighted recall is 0.5944, which means the model has a moderate capability of finding all the relevant instances across the classifiers.
- **AUROC:** The Area Under the Receiver Operating Characteristic curve is 0.6245 for the weighted average, showing an acceptable level of separability. This implies that the model has a reasonable ability to distinguish between the positive and negative classes.
- **Accuracy:** The overall weighted accuracy of the model is 59.91%, which is relatively lower compared to the other metrics, indicating that there is room for improvement in correctly identifying all instances.

The weighted metrics indicate that the model performs moderately well with the Adam optimizer at the given learning rate. While the model is relatively good at precision and recall, its accuracy suggests it may still confuse some classes or have difficulty with certain instances. The AUROC indicates that the model's predictive quality is acceptable, but there may be potential to enhance it by tuning other hyperparameters or training for more epochs.

Parameter	Value
Batch Number	64
Number of Epochs	10
Optimizer	Adam
Learning Rate (lr)	1×10^{-5}

Table 4.10. *Adam Optimizer Parameters*

Metric	Classifier 1	Classifier 2	Classifier 3	Weighted
F1-Score	0.6366	0.4922	0.5594	**0.5627**
Precision	0.8688	0.4336	0.4704	**0.5909**
Recall	0.5099	0.5795	0.6939	**0.5944**
AUROC	0.7243	0.6539	0.4954	**0.6245**
Accuracy	57.18%	61.56%	61.01%	**59.91%**

Table 4.11. *Adam Optimizer*

SGD

For the experiment 4.13 utilizing the SGD optimizer with a learning rate of 0.01, momentum of 0.9, and employing a StepLR learning rate scheduler, the weighted metrics across classifiers are:

- **F1-Score:** The weighted F1-Score is 0.8333, indicating a strong balance between precision and recall. This high score suggests that the model is robust in terms of the harmonic mean of precision and recall across the different classifiers.

- **Precision:** The weighted precision is 0.8408, suggesting that when the model predicts a positive class, it is correct 84.08% of the time on average.
- **Recall:** The weighted recall, at 0.8309, shows that the model is able to find 83.09% of all the relevant instances in the dataset, on average.
- **AUROC:** The Area Under the Receiver Operating Characteristic curve is 0.8904, which is quite high and indicates a strong ability of the model to differentiate between the classes.
- **Accuracy:** The weighted accuracy is 84.63%, which is quite high and indicates that the model is correctly identifying a high percentage of all instances.

These weighted metrics from the experiment suggest excellent model performance with the SGD optimizer. The use of the StepLR scheduler, which decreases the learning rate at regular intervals, seems to be effective in conjunction with SGD's momentum term. The model shows a high degree of predictive reliability and discriminative power, as evidenced by the high scores across all metrics. This configuration appears to provide a good balance between learning rate adjustments and momentum-based optimization, resulting in strong classifier performance.

Component	**Configuration**
Optimizer	SGD
Learning Rate (lr)	0.01
Momentum	0.9
Scheduler	StepLR
Step Size	1
Gamma	0.9
Name	lr_scheduler

Table 4.12. *Optimizer and Learning Rate Scheduler Parameters*

Metric	**Classifier 1**	**Classifier 2**	**Classifier 3**	**Weighted**
F1-Score	0.9248	0.7907	0.7846	**0.8333**
Precision	0.9167	0.7783	0.8274	**0.8408**
Recall	0.9344	0.8098	0.7485	**0.8309**
AUROC	0.8960	0.9160	0.8594	**0.8904**
Accuracy	88.75%	85.93%	79.21%	**84.63%**

Table 4.13. *SGD Optimizer*

Graphs

This bar chart, in figure 4.3, represents learning rate vs. accuracy for a model trained with the RMSprop optimization algorithm. It is evident that the model achieves an appreciable increase in accuracy as the learning rate decreases between 0.01 to 0.00001. The accuracy is also relatively constant, moving from 59.74% to 59.02%, as the learning rate

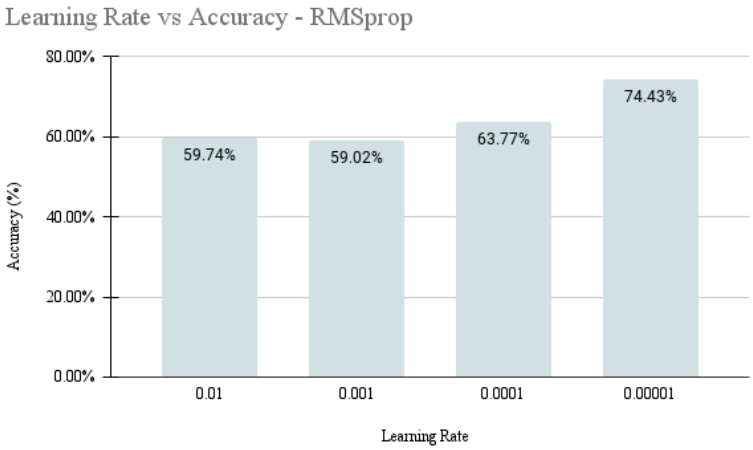

Figure 4.3. *Learning Rate vs Accuracy - RMSprop.*

is reduced from 0.01 to 0.001. However, at lower learning rates, a considerable increase in performance can be observed, with accuracy increasing to 63.77% at 0.0001 and peaking at 74.43% when the learning rate has been further reduced to 0.00001. The graph shows that for this model, use of a lower learning rate will lead to better model accuracy.

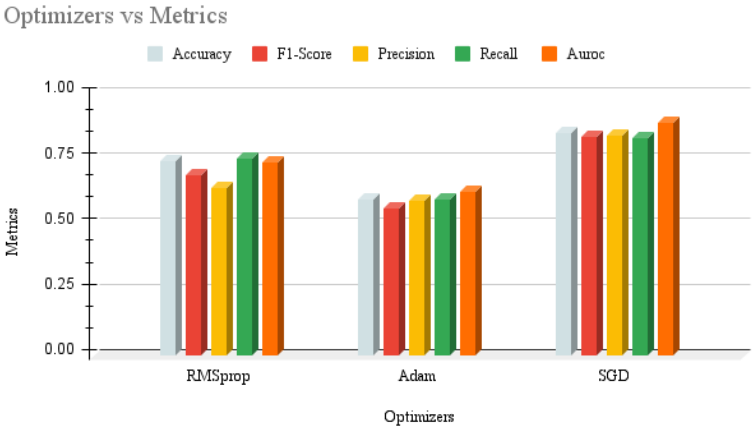

Figure 4.4. *Optimizers vs Metrics*

The bar chart 4.4, "Optimizers v/s Metrics," considers the performance exhibited by the three different optimizers—RMSprop, Adam, and SGD—against the five different metrics: Accuracy, F1-Score, Precision, Recall, and AUROC. Out of the five bars representing the said metric, one bar for each of the groups of optimizer takes the specified value. The scale of values is from 0 to 1, or rather from 0% to 100%. It is clearly seen that all three optimizers depict almost similar performance in terms of the metrics with slight variation. For example, one optimizer would have slightly higher Recall but lower Precision, or higher F1-Score but slightly lesser in Accuracy. The AUROC bars seem high throughout for any of the optimizers, which indicates good class discrimination. Such a plot will, therefore, make it clear to understand the compromise between different metrics by any of the optimizers and help in finding out which optimizer is correct for the given problem

depending on that problem's key metric.

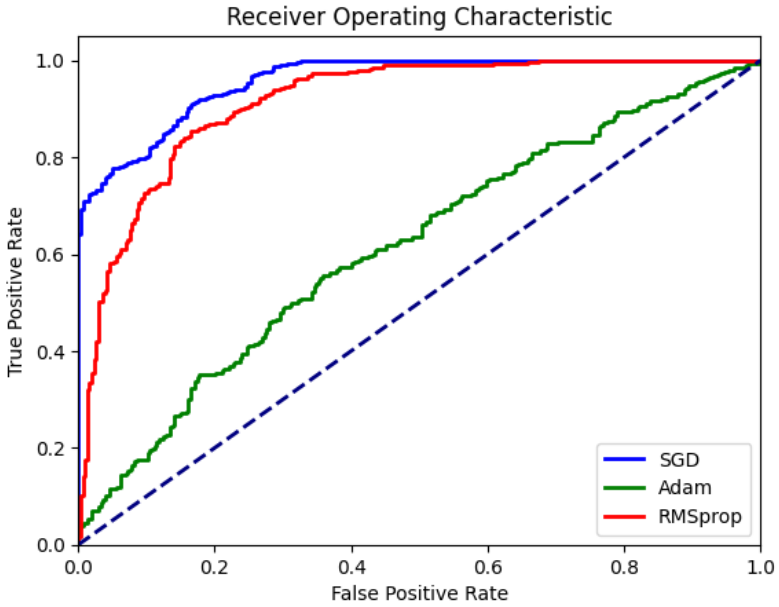

Figure 4.5. *Optimizers vs ROC Curve.*

The Receiver Operating Characteristic (ROC) curve, in figure 4.5, shows the performance of three different optimization algorithms—Stochastic Gradient Descent (SGD), Adam, and RMSprop—used in the current classification of leg movement. The curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.

The curve for SGD, shown in blue, is closest to the top-left corner of the graph, which indicates a higher True Positive Rate (TPR) for any given False Positive Rate (FPR) compared to the other two algorithms. This positioning suggests that SGD has a larger area under the curve (AUC), which is indicative of a better overall performance in classification tasks. Thus, SGD would be considered the best optimizer among the three, based on this ROC curve analysis.

Part III

Epilogue

4.7 Discussion

4.7.1 Final Outcome

Based on the experiments conducted using the R3DCNN model on topographical brain maps, several conclusions can be drawn about the model's performance with different optimizers and learning rates, particularly looking at the weighted classifier metrics.

- **Model Performance Across Optimizers:** The model was tested with various optimizers—SGD, RMSprop, and Adam—each with different learning rates. In terms of weighted classifier metrics, which provide a balanced view of the model's performance across all classes, the SGD optimizer with a learning rate scheduler appeared to offer superior performance. This suggests that for the topographical brain map data, which is complex and high-dimensional, SGD's approach to navigating the optimization landscape, possibly due to its momentum component and learning rate adjustments, is effective.
- **Impact of Learning Rate:** The learning rate is a critical hyperparameter in training deep learning models. The experiments show a general trend that, for the R3DCNN model, as the learning rate decreases, the weighted metrics improve, indicating a finer adjustment to the model weights that leads to better generalization.
- **Data Complexity and Model Suitability:** The R3DCNN model is designed to capture the complex spatial, spectral, and temporal features inherent in EEG data, as represented by the topographical brain maps. The experiments demonstrate the model's capacity to handle the complexity of EEG data, showing a promising direction for leg movement classification task. Given the high dimensionality and variability of EEG signals represented by the topographical brain maps, the use of 3D convolutions and recurrent layers is appropriate for extracting meaningful patterns that are crucial for accurate classification.

The general discussion indicates that the R3DCNN model, paired with the right optimizer and learning rate, can effectively learn from EEG data presented as topographical brain maps. The model's ability to perform well across different tasks (as weighted by classifiers) suggests its robustness and potential for broader applications in neurological research and practical applications such as monitoring other movements of human parts.

4.7.2 Future Work

Building upon the current implementation of the CNN-RNN model for classifying leg movements using EEG data represented as topographical brain maps, there are multiple directions for future research that promise to enhance the model's effectiveness and applicability. The integration of convolutional neural networks (CNNs) for spatial feature extraction with recurrent neural networks (RNNs) for capturing temporal dependencies has shown potential in decoding complex EEG signals. However, optimizing the architecture and exploring more sophisticated models could yield significant improvements.

Future studies might consider experimenting with different types of convolutional and recurrent layers, such as dilated convolutions for capturing spatial features at various scales or Transformer-based models for better temporal analysis. Additionally, a systematic approach to hyperparameter tuning could further refine model performance, making use of techniques like grid search, random search, or even Bayesian optimization to find optimal settings.

Another critical area for future work is enhancing the model's ability to generalize across subjects, a common challenge in EEG signal analysis due to inter-individual variability.

Lastly, as models become more complex, ensuring their interpretability and explainability becomes crucial, especially in clinical settings. Future efforts should focus on making the models more transparent, using techniques such as saliency maps or layer-wise relevance propagation to understand model decisions better. Additionally, integrating multimodal data, such as fMRI or physiological signals, with EEG data could offer a more comprehensive view of neural activities, enhancing the model's predictive capabilities. Pursuing these avenues will not only improve the performance and utility of EEG-based models but also contribute to the broader field of neurotechnology and its applications.

Bibliography

[1] *History of AI*. https://tutorialforbeginner.com/history-and-application-of-ai.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed και Humaira Arshad. *State-of-the-art in artificial neural network applications: A survey*. *Heliyon*, 4:e00938, 2018.

[3] The British Neuroscience Association και The European Dana Alliancefor the Brain. *Neuroscience: Science of the Brain*. The British Neuroscience Association, Liverpool, UK, 2003. An introduction for young students.

[4] Ildar Rakhmatulin. *Progress in neural networks for EEG signal recognition in 2021*. *Available as preprint*, 2021. South Ural State University, Department of Power Plants Networks and Systems, Chelyabinsk, Russia.

[5] TMSi. *The 10-20 System for EEG*. https://info.tmsi.com/blog/the-10-20-system-for-eeg, 2024. Accessed: insert access date here.

[6] Pengbo Zhang, Xue Wang, Weihang Zhang και Junfeng Chen. *Learning Spatial-Spectral-Temporal EEG Features With Recurrent 3D Convolutional Neural Networks for Cross-Task Mental Workload Assessment*. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(1):31–42, 2019.

[7] Taufique Ahmed και Luca Longo. *Examining the Size of the Latent Space of Convolutional Variational Autoencoders Trained With Spectral Topographic Maps of EEG Frequency Bands*. *IEEE Access*, 10:107575–107586, 2022.

[8] Ryan G. Hefron, Brett J. Borghetti, James C. Christensen και Christine M. Schubert Kabban. *Deep Long Short-term Memory Structures Model Temporal Dependencies Improving Cognitive Workload Estimation*. *Pattern Recognition Letters*, 94:96–104, 2017.

[9] Appyhigh Technology Blog. *Convolutional Neural Networks: A Brief History of Their Evolution*. https://medium.com/appyhigh-technology-blog/convolutional-neural-networks-a-brief-history-of-their-evolution-ee3405568597, 2020. Accessed: [insert date here].

[10] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie και Laith Farhan. *Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions*. *Journal of Big Data*, 8(53), 2021.

[11] MRI Questions. *Convolutional Network*. https://mriquestions.com/convolutional-network.html, 2021. Accessed: insert access date here.

[12] Wikipedia contributors. *Recurrent neural network - History*. https://en.wikipedia.org/wiki/Recurrent_neural_network, 2024. Accessed: insert access date here.

[13] IBM. *Recurrent Neural Networks*. https://www.ibm.com/topics/recurrent-neural-networks, 2024. Accessed: insert access date here.

[14] PolarSparc. *Deep Learning - Recurrent Neural Networks*. https://www.polarsparc.com/xhtml/DL-RecurrentNN.html, 2023. Accessed: insert access date here.

[15] First Initial. [if available] Author's Last Name. *The Main Advantages and Disadvantages of the RNN*. https://www.researchgate.net/figure/The-Main-Advantages-and-Disadvantages-of-the-RNN_tbl2_343837591, 2024. Accessed: insert access date here.

[16] IBM. *Regularization*. https://www.ibm.com/topics/regularization, 2024. Accessed: insert access date here.

[17] Analytics Vidhya. *How Does the Gradient Descent Algorithm Work in Machine Learning?* https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/, 2020. Accessed: insert access date here.

[18] Cornell University. *Stochastic Gradient Descent*. https://optimization.cbe.cornell.edu/index.php?title=Stochastic_gradient_descent, 2024. Accessed: insert access date here.

[19] Baeldung. *Gradient, Stochastic, and Mini-Batch Descent Explained*. https://www.baeldung.com/cs/gradient-stochastic-and-mini-batch, 2024. Accessed: insert access date here.

[20] Giskard. *Adaptive Gradient Algorithm (AdaGrad)*. https://www.giskard.ai/glossary/adaptive-gradient-algorithm-adagrad, 2024. Accessed: insert access date here.

[21] Diederik P. Kingma και Jimmy Ba. *Adam: A Method for Stochastic Optimization*. *arXiv preprint arXiv:1412.6980*, 2014.

[22] GeeksforGeeks. *Gradient Descent with RMSprop from Scratch*. https://www.geeksforgeeks.org/gradient-descent-with-rmsprop-from-scratch/, 2024. Accessed: insert access date here.

[23] GeeksforGeeks. *Metrics for Machine Learning Model*. https://www.geeksforgeeks.org/metrics-for-machine-learning-model/, 2024. Accessed: insert access date here.

[24] Glassbox Medicine. *Measuring Performance: AUC (AUROC)*. https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/, 2019. Accessed: insert access date here.

[25] Glassbox Medicine. *Measuring Performance: The Confusion Matrix*. https://glassboxmedicine.com/2019/02/17/measuring-performance-the-confusion-matrix/, 2019. Accessed: insert access date here.

[26] Google. *Google Colaboratory*. https://colab.research.google.com/, 2024. Accessed: insert access date here.

[27] Lightning AI. *Lightning AI*. https://lightning.ai/, 2024. Accessed: insert access date here.

[28] NumPy Developers. *NumPy Documentation*. https://numpy.org/doc/, 2024. Accessed: insert access date here.

[29] Matplotlib Developers. *Matplotlib Documentation*. https://matplotlib.org/, 2024. Accessed: insert access date here.

[30] Stefan Appelhoff. *EEG electrode positions*. https://github.com/sappelhoff/eeg_positions, 2022. Accessed: insert access date here.