UNIVERSITY OF PATRAS

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF THE SUPERVISOR: COMPUTER LOGIC
LABORATORY OF THE SUPERVISOR: LABORATORY OF INFORMATION SYSTEMS AND ARTIFICIAL INTELLIGENCE

SPEECH EMOTION RECOGNITION USING DEEP LEARNING

D I P L O M A T H E S I S

GEORGIOS SKOULIDIS

SUPERVISOR: SPYROS SIOUTAS

PATRAS – AUGUST 2023

University of Patras, Department of Electrical and Computer Engineering.

Georgios Skoulidis

CERTIFICATION

It is certified that the Diploma Thesis titled

SPEECH EMOTION RECOGNITION USING DEEP LEARNING

of the Department of Electrical and Computer Engineering student

GEORGIOS SKOULIDIS

Registration Number: 1059506

was presented publicly at the Department of Electrical and Computer Engineering at

14/09/2023

and was examined by the following examining committee:

Spyros Sioutas, Professor, Computer Engineering and Informatics Department, University of Patras (supervisor)

Kyriakos Sgarbas, Associate Professor, Electrical and Computer Engineering, University of Patras (committee member)

Michael Logothetis, Professor, Electrical and Computer Engineering, University of Patras (committee member)

The Supervisor	The Director of the Division
Spyros Sioutas Professor	Michael Logothetis Professor

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΒΛΕΠΟΝΤΟΣ: ΛΟΓΙΚΟΥ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΕΠΙΒΛΕΠΟΝΤΟΣ: ΕΡΓΑΣΤΗΡΙΟ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

ΑΝΑΓΝΩΡΙΣΗ ΣΥΝΑΙΣΘΗΜΑΤΩΝ ΟΜΙΛΙΑΣ ΜΕ ΧΡΗΣΗ ΒΑΘΕΙΑΣ ΜΑΘΗΣΗΣ

Δ Ι Π Λ Ω Μ Α Τ Ι Κ Η Ε Ρ Γ Α Σ Ι Α

ΓΕΩΡΓΙΟΣ ΣΚΟΥΛΙΔΗΣ

ΕΠΙΒΛΕΠΩΝ: ΣΠΥΡΟΣ ΣΙΟΥΤΑΣ

ΠΑΤΡΑ -ΑΥΓΟΥΣΤΟΣ 2023

Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών.

Γεώργιος Σκουλίδης

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η Διπλωματική Εργασία με τίτλο

ΑΝΑΓΝΩΡΙΣΗ ΣΥΝΑΙΣΘΗΜΑΤΩΝ ΟΜΙΛΙΑΣ ΜΕ ΧΡΗΣΗ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ

του/της φοιτητή/τριας του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΓΕΩΡΓΙΟΣ ΣΚΟΥΛΙΔΗΣ ΤΟΥ ΦΩΤΙΟΥ

Αριθμός Μητρώου: 1059506

Παρουσιάστηκε δημόσια στο Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών στις

14/09/2023

και εξετάστηκε από την ακόλουθη εξεταστική επιτροπή:

Σπύρος Σιούτας, Καθηγητής, Τμήμα Μηχανικών Η/Υ & Πληροφορικής, (επιβλέπων)

Κυριάκος Σγάρμπας, Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών (μέλος επιτροπής)

Μιχαήλ Λογοθέτης, Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών (μέλος επιτροπής)

Ο/Η Επιβλέπων/ουσα

Σπύρος Σιούτας
Καθηγητής

Ο/Η Διευθυντής/τρια του Τομέα

Μιχαήλ Λογοθέτης
Καθηγητής

PREFACE

I am deeply grateful to my supervisor, Professor Spyros Sioutas from the Computer Engineering and Informatics Department, for granting me the chance to pursue my own idea and for his mentorship.

I also extend my heartfelt thanks to Giorgos Drakopoulos for his support of my work. Moreover, I am grateful to Panagiotis Hadjidoukas and Manos Georgoudakis from the Computer Engineering and Informatics Department for generously providing me with access to their server with the high-powered GPU, which significantly aided my research.

Beyond my academic circle, I owe an immense debt of gratitude to my unwavering supporting pillars - my parents, Fotis and Konstantina, who have been my source of strength throughout the years. Their encouragement and belief in me have been instrumental in my accomplishments.

I would also like to express my thanks to all my friends, who have stood by me during this journey. Furthermore, I am grateful for the love and support of my sister Stefania and my grandmothers, Voula and Eirini.

Lastly, I would like to take a moment to honor the memory of my beloved late grandfather, Ioannis. His endless curiosity and artistic nature were the traits that I inherited from him and proudly carry with me today.

ABSTRACT

SPEECH EMOTION RECOGNITION USING DEEP LEARNING

STUDENT NAME, SURNAME:
GEORGIOS SKOULIDIS

SUPERVISOR NAME, SURNAME:
SPYROS SIOUTAS

This thesis aims to build a robust system for recognizing speech emotions through the utilization of supervised labeled data and advanced deep-learning techniques with image classification. The core objective of this study is to construct a model that can precisely distinguish between emotional classes for English speakers.

The investigation extends to the analysis of the influence of three specific spectral features. These features are treated as images, each displayed in four different output sizes resulting from a quadratic transformation achieved through bilinear image interpolation. We also emphasize the evaluation of three custom abstract Convolutional Neural Network (CNN) architectures. These architectures are characterized by their composition of three convolutional layers and three fully-connected layers, among other components. We use parameter tuning to identify the optimal internal parameters and concretize the CNN structures, but to also adjust the batch size and learning rate values to enhance performance. Furthermore, to improve generalization, a custom early-stopping algorithm is integrated with the 5-fold cross-validation method. Specific pre-processing steps are employed, along with some audio-based techniques for cross-validation data.

An additional objective is to study the impact of the optimized pre-trained English Speech Emotion Recognition model when applied to speech samples from Greek speakers. A limited dataset of Greek speech is employed to train, validate, and test the model's performance, while we assess the knowledge of the model's pre-trained layers.

ΕΚΤΕΤΑΜΈΝΗ ΕΛΛΗΝΙΚΉ ΠΕΡΊΛΗΨΗ

Α Εισαγωγή

Στην διπλωματική αυτή εργασία, θα πραγματευτούμε το ζήτημα της Αναγνώρισης Συναισθημάτων Ομιλίας (Speech Emotion Recognition) με χρήση Βαθιάς Μάθησης (Deep Learning). Στόχος μας είναι να διεισδύσουμε ακόμα περισσότερο στο πεδίο, κάνοντας χρήση Συνελικτικών Νευρωνικών Δικτύων (Convolutional Neural Networks) για την ταξινόμηση του συναισθηματικού περιεχομένου δισδιάστατων φασματικών χαρακτηριστικών. Πιο συγκεκριμένα, θα εμβαθύνουμε το σκοπό της έρευνας στην εύρεση των βέλτιστων παραμέτρων για την μοντελοποίηση του καλύτερου δυνατού ταξινομητή, σε ένα σετ κυματομορφών στην Αγγλική γλώσσα, ενώ θα αποτυπώσουμε την επίδραση των καλύτερων συνδυασμών παραμέτρων για μελλοντική αναφορά. Έπειτα, αφού μοντελοποιήσουμε τον ταξινομητή, θα εξερευνήσουμε την περίπτωση της Μεταφοράς Μάθησης (Transfer Learning) στην Ελληνική γλώσσα. Θα δούμε αν είναι δυνατόν να εκμεταλλευτούμε τη γνώση του ήδη προ-εκπαιδευμένου μοντέλου για την ταξινόμηση συγκεκριμένων συναισθημάτων Ελληνόφωνων ομιλητών.

Β Μηχανική Μάθηση

Η μηχανική μάθηση είναι ένα υποπεδίο της τεχνητής νοημοσύνης που περιλαμβάνει την εκπαίδευση των πρακτόρων για την εκτέλεση διαφόρων εργασιών, οι οποίες είναι συχνά πολύπλοκες και υπολογιστικά εντατικές.

Στην πράξη, οι πράκτορες εκμάθησης αντιπροσωπεύονται ως μοντέλα που χρησιμοποιούν αλγόριθμους για να μάθουν από τα δεδομένα εκπαίδευσης και να προσαρμόσουν τις δομές τους για να κάνουν καλύτερες προβλέψεις. Μια πρόβλεψη γίνεται προσδιορίζοντας μία έξοδο y για ένα σετ εισόδων x. Η εποπτευόμενη μάθηση είναι τεχνική μηχανικής μάθησης μέσω της οποίας ένα μοντέλο μαθαίνει να κάνει προβλέψεις μελετώντας μια ομάδα εισόδων x που αντιστοιχούν σε ορισμένες εξόδους y. Η μάθηση χωρίς επίβλεψη από την άλλη βασίζεται μόνο σε εισόδους x των οποίων οι έξοδοι δεν διατίθενται, μελετώντας τα μοτίβα και τα χαρακτηριστικά που εμφανίζουν οι είσοδοι. Τέλος η ενισχυτική μάθηση ανταμείβει το μοντέλο για κάθε επιθυμητή ενέργεια και ”τιμωρεί” για κάθε ανεπιθύμητη.

Η Ταξινόμηση (Classification) είναι μια υποκατηγορία επιβλεπόμενης μάθησης για διακριτές εξόδους που εκπροσωπούν μία “τάξη”. Για να μετρήσουμε την απόδοση ενός ταξινομητή χρησιμοποιούμε μεγέθη όπως η Ακρίβεια (Accuracy) και το Macro F1-Score. Γενικά τα μεγέθη μέτρησης απόδοσης παίρνουν τιμές από το 0 έως το 1 και συχνά χρησιμοποιούνται τα ποσοστά τους σε %. Το πρόβλημα της Ανίχνευσης Συναισθημάτων μπορεί να εκφραστεί και ως πρόβλημα ταξινόμησης.

Τα Νευρωνικά Δίκτυα (Neural Networks) αποτελούν μεθόδους ιδιαίτερα χρήσιμες για προβλήματα που πολύπλοκων δεδομένων ή δεδομένων με μεγάλη διαστατικότητα. Φυσικά, ένα νευρωνικό δίκτυο μπορεί να κάνει προβλέψεις κάθε είδους μάθησης. Τα ΝΔ αποτελούνται από κόμβους ή νευρώνες. Μια σειρά κόμβων που συμβάλλουν στην εκτέλεση μιας συγκεκριμένης λειτουργίας, αντιπροσωπεύουν ένα στρώμα, ενώ κόμβοι διαφορετικών στρωμάτων μπορούν να συνδέονται μεταξύ τους, από την είσοδου έως και την έξοδο. Τα στρώματα ενδιάμεσα της εισόδου και εξόδου λέγονται κρυφά. Ένα δίκτυο με δύο ή περισσότερα κρυφά στρώματα, ονομάζεται Βαθύ Νευρωνικό Δίκτυο (Deep Neural Network).

Η γενική εξίσωση (1) συνδέει τον k-οστό κόμβο ή νευρώνα του j-οστού στρώματος $n_{j,k}$ ενός Νευρωνικού Δικτύου και ονομάζεται προς τα εμπρός διάδοση (forward propagation).

$$n_{j,k} = \sum_{i=0}^{N} w_{i,j,k} \cdot n_{j-1,i} \tag{1}$$

Στην πράξη, για να μοντελοποιήσουμε έναν ταξινομητή, αυτό που κάνουμε είναι να εκπαιδεύουμε ένα μοντέλο με ζεύγη εισόδων και εξόδων, ώστε να μπορεί μετέπειτα να προβλέψει την έξοδο ή κλάση όταν δίνεται μια είσοδος. Η πρώτη διαδικασία λέγεται εκπαίδευση (training) ενώ η διαδικασία ελέγχου της ικανότητας του μοντέλου λέγεται δοκιμή ή επικύρωση. Ο όρος επικύρωση (validation) θα χρησιμοποιηθεί πιο ειδικά για μερικό έλεγχο της ικανότητας ενός μοντέλου με χρήση των δεδομένων εκπαίδευσης, ενώ η δοκιμή (testing) θα γίνει μία φορά στο τέλος αφού έχει βρεθεί το βέλτιστο μοντέλο. Παρόλα αυτά, για να εκπαιδευτεί ένα μοντέλο θα πρέπει να εκτελεστεί μία διαδικασία ελαχιστοποίησης μίας συνάρτησης κόστους ή απωλειών, με χρήση ενός αλγορίθμου βελτιστοποίησης, που για δομές νευρωνικών δικτύων συχνά αποκαλούμε βελτιστοποιητή (optimizer). Οι αλγόριθμοι αυτοί βασίζονται συχνά στον αλγόριθμο ελαχιστοποίησης Gradient Descent ο οποίος δίνεται από τη μαθηματική σχέση (2), όπου η είναι ο ρυθμός μάθησης, J η συνάρτηση απωλειών και w οι παραμέτροι των βαρών (weights) και των πολώσεων (biases).

$$w = w - \eta \cdot \nabla_w J(w) \tag{2}$$

Οι βελτιστοποιητές είναι αλγόριθμοι που συμβάλλουν στην ελαχιστοποίηση της συνάρτησης απωλειών, η τιμή της οποίας εξαρτάται από τα βάρη των νευρώνων ενός δικτύου. Τα βάρη αυτά θα πρέπει να ανανεωόνται ανά διαστήματα, με στόχο τη βελτίωση της ικανότητας του μοντέλου, όπως φαίνεται και στη σχέση (2). Η συνολική διαδικασία με την οποία ανανεώνουμε τα βάρη από τους κόμβους εξόδου μέχρι τους κόμβους εισόδου, με στόχο την ελαχιστοποίηση της συνάρτησης απωλειών, ονομάζεται προς τα πίσω διάδοση (back propagation).

Ακόμη, για να ενισχύσουμε την ακρίβεια πρόβλεψης ενός μοντέλου ΝΔ και να κάνουμε τη λειτουργία του πιο μη-γραμμική, εισάγουμε συχνά στις εξόδους των νευρώνων συναρτήσεις, τις οποίες αποκαλούμε συναρτήσεις ενεργοποίησης (activation functions). Μερικές από αυτές είναι η ReLu (Rectified Linear Unit) και η Leaky ReLu που αποτελεί παραλλαγή της πρώτης, με τη διαφορά ότι έχει έναν συντελεστή αρνητικής κλίσης.

Όπως γίνεται κατανοητό για την εκπαίδευση και την δοκιμή ενός μοντέλου χρειαζόμαστε ένα σετ δεδομένων. Κάθε φορά που χρησιμοποιούμε όλα τα δεδομένα εκπαίδευσης για την βελτιστοποίηση των παραμέτρων του μοντέλου, έχουμε διανύσει μία εποχή (epoch). Εάν εκπαιδεύσουμε το μοντέλο

για πολύ λίγες εποχές τότε συχνά παρουσιάζεται το φαινόμενο της υποπροσαρμογής (underfitting), ενώ για εκπαίδευση πολλών εποχών μπορεί να παρουσιαστεί το αντίθετο φαινόμενο, αυτό της υπερπροσαρμογής (overfitting), που σημαίνει ότι το μοντέλο έχει γίνει ιδιαίτερα πολύπλοκο και παρότι είναι αποδοτικό για προβλέψεις στο σετ εκπαίδευσης, θα υπολειτουργεί σε ένα νέο σετ δοκιμής, νέων δεδομένων. Εντούτοις, οι εποχές δεν είναι η μόνη παράμετρος που μπορεί να επηρεάσει την σωστή προσαρμογή ενός μοντέλου, αλλά κάθε παράμετρος μπορεί είτε να βελτιώσει την απόδοση, είτε να την μειώσει.

Για να βρούμε το καλύτερο μοντέλο, συχνά χρησιμοποιούμε τη διαδικασία της διασταυρωμένης επικύρωσης (cross validation). Ειδικότερα θα δουλέψουμε με τη μέθοδο της διασταυρωμένης επικύρωσης Κ-πτυχών (K-Fold cross validation), κατά της οποίας θα χωρίζουμε το σετ δεδομένων εκπαίδευσης σε Κ ίσα μέρη και θα χρησιμοποιούμε τα Κ-1 από αυτά για εκπαίδευση και το μέρος που περισσεύει για επικύρωση. Εάν λοιπόν επαναλάβουμε τη διαδικασία αυτή για Κ πτυχές, δηλαδή για σετ επικύρωσης ένα διαφορετικό μέρος δεδομένων κάθε φορά, και ταυτόχρονα υπολογίζουμε την τιμή ενός επιλεγμένου μεγέθους απόδοσης για την κάθε πτυχή, τότε στο τέλος θα μπορούμε να βγάλουμε έναν μέσο όρο της απόδοσης του εκάστοτε μοντέλου, ώστε να το συγκρίνουμε στη συνέχεια με άλλα μοντέλα και ταυτόχρονα να αποφύγουμε πιθανή υπερπροσαρμογή.

Η διαδικασία αυτή μπορεί να εκτελεστεί όσες φορές επιθυμούμε, αναλόγως τον αριθμό των τιμών που θέλουμε να δώσουμε στις διάφορες παραμέτρους και να εξερευνήσουμε. Οι συνδυασμοί αυτοί μπορεί να είναι άπειροι και έτσι χρειάζεται να δημιουργήσουμε ένα πλέγμα πιθανών διανυσματικών τιμών οι οποίες μπορούν να υιοθετηθούν από το διάνυσμα των παραμέτρων που θέλουμε να ρυθμίσουμε. Φυσικά, με το πλέγμα αυτό επιχειρούμε να θέσουμε κάποια όρια και συχνά να εξοικονομήσουμε χρόνο. Κατόπιν, μπορούμε να κάνουμε χρήση διάφορων τεχνικών αναζήτησης, μερικές από τις οποίες είναι η αναζήτηση πλέγματος (grid search) και η τυχαία αναζήτηση (random search), κατά την οποία καθορίζουμε εκ των προτέρων τον αριθμό των διανυσματικών τιμών που θέλουμε να αναζητήσουμε και έτσι μπορούμε να δουλεύουμε και σε μεγαλύτερα πλέγματα.

Γενικά, με την μέθοδο της διασταυρωμένης επικύρωσης μπορούμε να δοκιμάσουμε νέους συνδυασμούς παραμέτρων στο μοντέλο μας και να εντοπίσουμε ποιοι δουλεύουν καλύτερα. Αξίζει όμως να σημειωθεί ότι για την παράμετρο των εποχών εκπαίδευσης, συχνά χρησιμοποιούμε τη μέθοδο της πρόωρης διακοπής (early-stopping), όπου με την ενεργοποίηση μίας σειράς προκαθορισμένων συνθηκών, η εκπαίδευση σταματά και έτσι εξοικονομείται χρόνος και αποφεύγεται η υπερπροσαρμογή. Μπορούμε να συνδυάσουμε αυτή την τεχνική με την τεχνική της διασταυρωμένης επικύρωσης Κ-πτυχών για καλύτερα αποτελέσματα.

Β.1 Μεταφορά Μάθησης

Συχνά κατηγοριοποιούμε τον τύπο Μεταφοράς Μάθησης (ΜΜ) με άξονα την γνώση που θέλουμε να μεταφέρουμε. Οι κύριες υποκατηγορίες είναι:

1. Επαγωγική (Inductive) ΜΜ:
   Απαιτούνται δεδομένα με ετικέτες (δηλαδή δοσμένες εξόδους) για τη δημιουργία ενός μοντέλου, ικανού να εκτελέσει μία αρχική εργασία. Έπειτα, η γνώση που έχει αποκτήσει το μοντέλο χρησιμοποιείται σε συνδυασμό με νέα δεδομένα με ετικέτες, για την εκτέλεση μίας νέας διαφορετικής εργασίας.
2. Μεταδοτική (Transductive) ΜΜ:

Απαιτούνται δεδομένα με ετικέτες για τη δημιουργία ενός μοντέλου, ικανού για την εκτέλεση μίας εργασίας σε έναν συγκεκριμένο τομέα, όμως δεν παρέχονται επιπλέον δεδομένα για την ίδια εργασία στον νέο τομέα. Συνεπώς, το μοντέλο πρέπει να εκμεταλλευτεί τη γνώση που έχει ήδη λάβει από τα αρχικά δεδομένα.

3. Μη-Επιβλεπόμενη (Unsupervised) ΜΜ Δεν παρέχονται οι ετικέτες των δεδομένων στους δύο τομείς, ενώ οι εργασίες είναι διαφορετικές αλλά σχετικές μεταξύ τους.

Στην διπλωματική αυτή θα εφαρμόσουμε Επαγωγική ΜΜ, αφού ο τομέας είναι ένας και απαράλλακτος (Αναγνώριση Συναισθημάτων Ομιλίας) και οι εργασίες είναι διαφορετικές (Αγγλική και Ελληνική Ταξινόμηση), καθώς τα δεδομένα των δύο εργασιών παρέχονται με ετικέτες.

Επιπρόσθετα, θα δοκιμάσουμε να παγώσουμε διάφορους συνδυασμούς στρωμάτων του προ-εκπαιδευμένου μοντέλου Συνελικτικού Νευρωνικού Δικτύου (ΣΝΔ), δηλαδή να διατηρήσουμε τις τιμές των κόμβων τους ως έχουν, για να παρατηρήσουμε την επίδραση της ΜΜ στην τελική ταξινόμηση. Με άλλα λόγια, θα ερευνήσουμε τη χρησιμότητα της γνώσης που έχουν λάβει τα διάφορα στρώματα του μοντέλου.

C Συνελικτικά Νευρωνικά Δίκτυα

Τα δίκτυα που θα μελετήσουμε στην περίπτωσή μας, είναι τα δισδιάστατα Συνελικτικά Νευρωνικά Δίκτυα.

Η δισδιάστατη διακριτή συνέλιξη εφαρμόζεται σε όλα τα κρυφά συνελικτικά στρώματα. Για δύο δισδιάστατα σήματα και συγκεκριμένα μία είσοδο x και έναν συνελικτικό πυρήνα k, ο τύπος της συνέλιξης εκφράζεται ως:

$$y[n_1, n_2] = \sum_i \sum_j x[n_1 + i, n_2 + j] \cdot k[i, j] \tag{3}$$

Εάν οι πίνακες εισόδου με μέγεθος I, οι πυρήνες με μέγεθος K, οι πίνακες προσθήκης μηδενικών (zero-padding) $P \geq 0$ και οι πίνακες βήματος (stride) $S \geq 1$, είναι όλοι τετραγωνικοί, τότε το μέγεθος της τετραγωνικής εξόδου ενός συνελικτικού στρώματος δίνεται από τον παρακάτω μαθηματικό τύπο.

$$O_c = \left\lfloor \frac{I - K + 2P}{S} \right\rfloor + 1 \tag{4}$$

Συχνά, εφαρμόζουμε στην έξοδο ενός συνελικτικού στρώματος, μία συνάρτηση ενεργοποίησης.

Κατόπιν, υπάρχει η τάση να μειώνουμε κι άλλο το μέγεθος των δεδομένων. Το σκοπό αυτό εξυπηρετούν τα στρώματα συγκέντρωσης (pooling layers).

$$O_p = \left\lfloor \frac{I - K}{S} \right\rfloor + 1 \tag{5}$$

Εφόσον διατηρούμε πάντα τη μεγαλύτερη τιμή ενός υποπίνακα, λέμε ότι έχουμε μέγιστη συγκέντρωση (max pooling), ενώ για εξαγωγή της μέσης τιμής των στοιχείων κάθε υποπίνακα και τη διατήρησή τους, λέμε ότι έχουμε μέση συγκέντρωση (average pooling).

Το πρώτο τμήμα της αρχιτεκτονικής ενός δισδιάστατου ΣΝΔ, αποτελείται συνήθως από συνελικτικά στρώματα και στρώματα συγκέντρωσης. Για να περάσουμε στο δεύτερο τμήμα, εκτελούμε μία διαδικασία εξομάλυνσης (flattening), κατά την οποία μετατρέπουμε έναν τρισδιάστατο πίνακα δεδομένων αποτελούμενο από τις διαστάσεις των καναλιών, του ύψους και του πλάτους, σε ένα μονοδιάστατο διάνυσμα, ως εξής:

$$(\text{αριθμός καναλιών}, \text{ύψος}, \text{πλάτος}) \rightarrow (\text{αριθμός καναλιών} \times \text{ύψος} \times \text{πλάτος})$$

Στη συνέχεια, το διάνυσμα μπορεί να τροφοδοτηθεί σε ένα ή και περισσότερα πλήρως-συνδεδεμένα στρώματα, οι νευρώνες των οποίων εισάγουν ένα γραμμικό μετασχηματισμό. Έπειτα από κάθε τέτοιο στρώμα, μπορούμε να εφαρμόσουμε μία συνάρτηση ενεργοποίησης για να αυξήσουμε την ικανότητα της ταξινόμησης.

Ένα άλλο είδος στρωμάτων που μπορούμε να εισάγουμε σε ένα ΣΝΔ, είναι τα στρώματα κανονικοποίησης (regularization layers), τα οποία στοχεύουν στην βελτίωση της γενίκευσης και αφαιρούν την περιττή πολυπλοκότητα. Τέτοια στρώματα είναι τα Dropout και τα Batch-Normalization.

D Αναγνώριση Συναισθημάτων Ομιλίας

Η μοντελοποίηση της αναγνώριση συναισθημάτων είναι δυνατόν να γίνει με δύο τρόπους. Ο πρώτος είναι με τη μέθοδο ταξινόμησης διακριτών τάξεων. Το μοντέλο του Έκμαν παρουσιάζει έξι κύριες διακριτές τάξεις συναισθημάτων, οι οποίες εκφράζουν θυμό (anger), αηδία (disgust), φόβο (fear), χαρά (happiness), θλίψη (sadness) και έκπληξη (surprise), καθώς και μία έβδομη ουδέτερη (neutral) συναισθηματική κατάσταση. Πολλά σετ δεδομένων με ετικέτες που παρέχονται στο διαδίκτυο ακολουθούν το μοντέλο αυτό. Ο δεύτερος τρόπος εμπίπτει στο πεδίο της παλινδρόμησης (regression) και αφορά την εύρεση μίας συνεχούς, διανυσματικής αριθμητικής τιμής, και τον προσδιορισμό του συναισθηματικού περιεχομένου με βάση το ποια τιμή είναι κοντινότερα σε αυτή.

Εμείς θα κάνουμε χρήση του μοντέλου του Έκμαν για τον ταξινομητή μας στην Αγγλική γλώσσα και θα εκμεταλλευτούμε κάποια από τα σετ δεδομένων με ετικέτες που παρέχονται για ταξινόμηση και επιβλεπόμενη μάθηση.

Επιπλέον, θα μελετήσουμε το συναισθηματικό περιεχόμενο τριών διαφορετικών δισδιάστατων, φασματικών χαρακτηριστικών (spectral features) που μπορούν να εξαχθούν από μία κυματομορφή ομιλίας. Οι αλγόριθμοι εξαγωγής των χαρακτηριστικών παρέχονται από τη βιβλιοθήκη της Librosa, και τα χαρακτηριστικά είναι τα Mel Spectrograms, MFCC και STFT Chromagrams.

Θα περάσουμε τετραγωνικές μήτρες διαφόρων μεγεθών, οι οποίες αναπαριστούν τα παραπάνω φασματικά χαρακτηριστικά, ως εισόδους στα ΣΝΔ και θα μελετήσουμε την τελική τους απόδοση. Έτσι το πρόβλημα Αναγνώρισης Συναισθημάτων Ομιλίας εμπίπτει στην κατηγορία της ταξινόμησης εικόνων με Συνελικτικά Νευρωνικά Δίκτυα.

Ακόμη, θα εφαρμόσουμε τεχνικές προ-επεξεργασίας στην αρχική κυματομορφή με στόχο να αυξήσουμε την απόδοση της ταξινόμησης, καθώς και στα δισδιάστατα φασματικά χαρακτηριστικά για να καταλήξουμε σε τετραγωνικές εικόνες με τεχνικές παρεμβολής εικόνας (image interpolation techniques). Η τεχνική για αλλαγή του μεγέθους στο ύψος (συχνότητα) και στο πλάτος (χρόνος) των εικόνων, θα είναι η Διγραμμική Παρεμβολή (Bilinear Interoplation) μέσω της βιβλιοθήκης PIL.

Ε Πειραματική Διάταξη & Περιβάλλον Εργασίας

Ε.1 Λογισμικό και Περιβάλλον

Για την περάτωση της διπλωματικής αξιοποιήσαμε την έκδοση 3.10 της Python, καθώς και διάφορες βιβλιοθήκες που είναι συμβατές με αυτή.

Βιβλιοθήκη	**Λειτουργία**
`PyTorch`	Προσφέρει λειτουργίες για εκπαίδευση και βελτιστοποίηση μοντέλων μηχανικής εκμάθησης.
`Matplotlib`	Χρησιμοποιείται για την ανάλυση και την αναπαράσταση δεδομένων.
`Numpy`	Παρέχει θεμελιώδεις λειτουργίες για αριθμητικούς υπολογισμούς.
`OS`	Μονάδα που χρησιμοποιείται για την αλληλεπίδραση με το λειτουργικό σύστημα.
`Pandas`	Επιτρέπει τον χειρισμό και την ανάλυση δεδομένων.
`Scipy`	Παρέχει αλγόριθμους για πολλαπλές αριθμητικές πράξεις και προβλήματα.
`Scikit-Learn`	Χρησιμοποιείται για προεπεξεργασία δεδομένων, μοντελοποίηση και αξιολόγηση.
`Librosa`	Παρέχει λειτουργίες για επεξεργασία ήχου.
`PIL`	Βιβλιοθήκη για επεξεργασία εικόνας.
`Math`	Δημοφιλής βιβλιοθήκη με ποικιλία μαθηματικών συναρτήσεων.
`Random`	Παρέχει τη δυνατότητα δημιουργίας τυχαίων αριθμών και τυχαίας δειγματοληψίας.
`Time`	Προσφέρει λειτουργίες για τη μέτρηση του χρόνου και τη μετατροπή μεταξύ αναπαραστάσεων χρόνου.

Η μονάδα επεξεργασίας γραφικών (GPU) που χρησιμοποιήθηκε για εκπαίδευση και δοκιμή των μοντέλων μας ήταν η NVIDIA TESLA V100-PCIE-GPU 32 GB.

Ε.2 Βάσεις Δεδομένων Συναισθημάτων Ομιλίας Πειραματικής Ανάλυσης

Υπάρχουν ποίκιλες βάσεις δεδομένων που διατίθενται ηλεκτρονικά και περιέχουν αρχεία ομιλίων με ετικέτες συναισθημάτων. Οι ομιλίες που συλλέξαμε για την εκπαίδευση, επικύρωση και δοκιμή των μοντέλων μας έχουν παραχθεί από ηθοποιούς. Οι βάσεις δεδομένων από τις οποίες συλλέξαμε αρχεία φαίνονται παρακάτω (συμβολίζουμε με sr τον ρυθμό δειγματοληψίας):

- RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) :
  1440 αρχεία audio, Αγγλικά, 24 ηθοποιοί, 8 κλάσεις συναισθημάτων, $sr = 48$ kHz.
- TESS (Toronto Emotional Speech Set) :
  2800 αρχεία audio, Αγγλικά, 2 γυναίκες ηθοποιοί, 7 κλάσεις συναισθημάτων, $sr = 24.414$ kHz.

- SAVEE (Surrey Audio-Visual Expressed Emotion) :
  480 αρχεία audio, Αγγλικά, 4 άνδρες ηθοποιοί, 7 κλάσεις συναισθημάτων, $sr = 44.1\,\text{kHz}$.
- CREMA-D (Crowd-Sourced Emotional Multimodal Actors Dataset) :
  7442 αρχεία audio, Αγγλικά, 91 ηθοποιοί, 6 κλάσεις συναισθημάτων, $sr = 16\,\text{kHz}$.
- AESDD (Acted Emotional Speech Dynamic Database) :
  605 αρχεία audio, Ελληνικά, 5 ηθοποιοί, 5 κλάσεις συναισθημάτων, $sr = 44.1\,\text{kHz}$.

Επιπρόσθετα, ο παρακάτω πίνακας απεικονίζει επακριβώς τα δεδομένα που έχουμε συλλέξει ανά κλάση και ανά βάαη δεδομένων για τον ταξινομητή συναισθημάτων από ομιλία στα αγγλικά:

Συναίσθημα	**RAVDESS***	**TESS***	**SAVEE**	**CREMA-D***	**Σύνολο**
Χαρά	192	400	60	273	925
Θυμός	192	399	60	273	924
Αηδία	192	400	60	273	925
Φόβος	192	399	60	273	924
Θλίψη	192	400	60	273	925
Ουδέτερη	96	400	120	509	1125
Έκπληξη	192	400	60	0	652
Σύνολο	1248	2798	480	1874	**6400**

Τα σετ εκπαίδευσης και δοκιμής χωρίστηκαν σε ένα ποσοστό 80%-20% αντίστοιχα, ενώ για την διασταυρωμένη επικύρωση χρησιμοποιήθηκαν 5 πτυχές, δηλαδή για κάθε πτυχή, ένα νέο 20% των δεδομένων εκπαίδευσης λειτουργούσε ως σετ αξιολόγησης του μοντέλου.

Για τον ταξινομητή με μεταφορά μάθησης στα ελληνικά, χρησιμοποιήσαμε 604 αρχεία από το AESDD. Συγκεκριμένα είχαμε ένα σετ με 121 ετικέτες θυμού, 122 αηδίας, 120 φόβου, 119 χαράς και 122 θλίψης. Ένα αρχείο με ετικέτα θλίψης ήταν καταστραμμένο και έτσι το αποκλείσαμε.

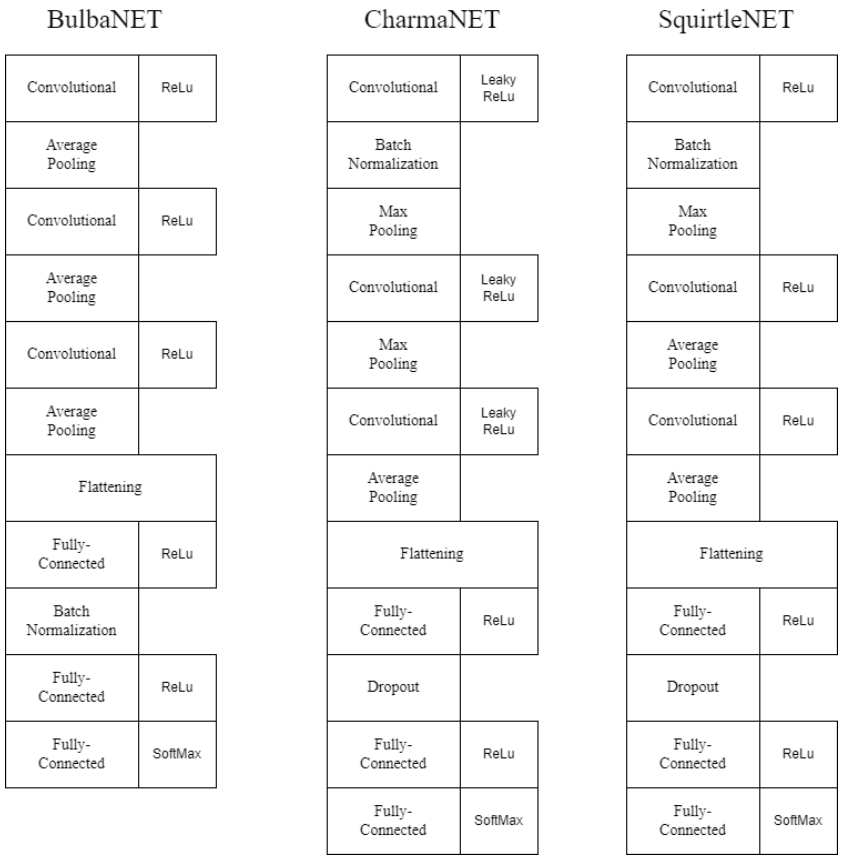
Ε.3 Προ-Επεξεργασία

Αφού αποκτήσαμε τα δεδομένα εκπαίδευσης και δοκιμής, εφαρμόζουμε μια σειρά αλεπάλληλων μεθόδων προ-επεξεργασίας, προτού τροφοδοτήσουμε την τελική επεξεργασμένη έξοδο στο ΣΝΔ.

1. Προ-Έμφαση:
   Ενίσχυση των υψηλών συχνοτήτων ενός ηχητικού σήματος.
2. Απαλοιφή μηδενικών:
   Απαλοιφή των μηδενικών στην αρχή και στο τέλος της ψηφιακής κυματομορφής.
3. Εξαγωγή Φασματικών Χαρακτηριστικών:
   Mel Spectrograms, MFCC, STFT Chromagrams.
4. Μετατροπή του πλάτους σε dB.
5. Τετραγωνοποίηση εικόνας μέσω Διγραμμικής Παρεμβολής (Bilinear Interoplation).

Ε.4 Αρχιτεκτονικές ΣΝΔ Πειραματικής Ανάλυσης

Οι αρχιτεκτονικές των τριών ΣΝΔ που χρησιμοποιήσαμε για την έρευνά μας, συμπεριλαμβάνουν τρία συνελικτικά στρώματα, το καθένα ακολουθούμενο από στρώμα μέγιστης ή μέσης συγκέντρωσης, καθώς και τρία πλήρως-συνδεδεμένα στρώματα που έπονται της εξομάλυνσης. Οι αρχιτεκτινικές και τα στρώματά των ΣΝΔ φαίνονται παρακάτω επακριβώς:



Οι αρχιτεκτονικές των τριών ΣΝΔ που χρησιμοποιήσαμε για την έρευνά μας, συμπεριλαμβάνουν τρία συνελικτικά στρώματα, το καθένα ακολουθούμενο από στρώμα μέγιστης ή μέσης συγκέντρωσης, καθώς και τρία πλήρως-συνδεδεμένα στρώματα που έπονται της εξομάλυνσης. Οι αρχιτεκτινικές και τα στρώματά των ΣΝΔ φαίνονται παρακάτω επακριβώς:

Ε.5 Ειδική Χειραγώγηση Δεδομένων Επικύρωσης

Για να αποφύγουμε την υπερπροσαρμογή και να ενισχύσουμε τη γενίκευση του μοντέλου μας, εφαρμόσαμε κάποιες τεχνικές χειραγώγησης σε ένα ποσοστό των δεδομένων που χρησιμοποιήθηκαν για επικύρωση. Συγκεκριμένα, δημιουργήσαμε ένα αντίγραφο του TESS και πριν εκτελεστεί οποιαδήποτε μέθοδος προ-επεξεργασίας, εφαρμόσαμε μία εκ των τεχνικών: χρονική διάταση (time-stretching), μετατόπιση τόνου (pitch-shifting), προσθήκη θορύβου Gauss, ή έναν συδυασμό αυτών, στις κυματομορφές του, με στόχο να επικυρώσουμε μετέπειτα τα μοντέλα μας βάσει μοτίβων ομιλίας με μικρότερο βαθμό συσχέτισης από τα σήματα ομιλίας στο υπάρχον σετ δεδομένων.

Να σημειωθεί ότι το αρχικό TESS, καθώς και το επεξεργασμένο αντίγραφό του βρίσκονταν σε πλήρη αντιστοιχία μεταξύ των αρχείων τους, και έτσι αφού συγχωνεύσαμε το καθένα εξ αυτών με τα υπόλοιπα τρία σετ και με τα αντίγραφά τους, μπορέσαμε να εφαρμόσουμε την ίδια τυχαία συνάρτηση χωρίσματος για το συνολικό σετ δεδομένων, τόσο για το σετ εκπαίδευσης-δοκιμής, όσο και για την μέθοδο της διασταυρωμένης επικύρωσης στα αρχικά σετ εκπαίδευσης για τις πτυχές εκπαίδευσης-επικύρωσης. Άρα, είχαμε ένα μικρό ποσοστό δεδομένων επικύρωσης με τροποποιήσεις στον τόνο (ένα τόνο ή ημιτόνιο, πάνω ή κάτω), στον χρόνο και στον θόρυβο. Η διαδικασία φαίνεται αναλυτικά παρακάτω:

1. Δημιούργησε αντίγραφο του TESS.
2. Εφάρμοσε τις ειδικές τεχνικές χειραγώγησης στα σήματα ομιλίας του αντιγράφου.
3. Εκτέλεσε όλα τα βήματα προ-επεξεργασίας για κάθε σετ δεδομένων.
4. Συγχώνευσε τις εικόνες φασματικών χαρακτηριστικών που παρασκευάστηκαν από το καθαρό TESS με εκείνες των υπόλοιπων σετ δεδομένων (RAVDESS, SAVEE, CREMA-D). Επανέλαβε για αυτές του επεξεργασμένου TESS με ένα αντίγραφο των εικόνων των υπόλοιπων σετ δεδομένων. Για λόγους απλότητας θα αναφερόμαστε στο πρώτο συγχωνευμένο σύνολο δεδομένων ως ”καθαρό” και στο δεύτερο ως ”χειραγωγημένο”.
5. Κάλεσε δύο συνεχόμενες τυχαίες συναρτήσεις διαχωρισμού για τα σετ εκπαίδευσης και δοκιμής, μία για το διαχωρισμό του καθαρού σετ και μία για εκείνον του χειραγωγημένου (το σετ δοκιμής του οποίου δεν χρειαζόμαστε). Με την βιβλιοθήκη random, πριν καλέσουμε τις συναρτήσεις, θέτουμε έναν ορισμένο σπόρο γεννήτριας ψευδοτυχαίων αριθμών (random seed) ώστε τα δύο σύνολα δεδομένων να χωριστούν με πλήρη αντιστοιχία και να μην ρισκάρουμε υπερπροσαρμογή.
6. Για τον διαχωρισμό του συνόλου εκπαίδευσης στη διασταυρωμένη επικύρωση, επανέλαβε το παραπάνω βήμα και για κάθε υποσύνολο επικύρωσης που χρησιμοποιείται για την αξιολόγηση ενός εκ των K μοντέλων, κάνε χρήση του χειραγωγημένου σετ δεδομένων.
7. Αργότερα στην δοκιμή και αξιολόγηση του τελικού μοντέλου, θα χρησιμοποιήσουμε το καθαρό σετ δοκιμής που έχουμε κρατήσει και δεν έχουμε ακόμα χρησιμοποιήσει.

F Πειραματική Ανάλυση

F.1 Εξερευνώντας τις Αρχιτεκτονικές των ΣΝΔ και τα Χωρικά Χαρακτηριστικά Εισόδου για Αναγνώριση Συναισθημάτων Αγγλικής Ομιλίας

Για την εύρεση του βέλτιστου μοντέλου Αναγνώρισης Συναισθημάτων Ομιλίας στην Αγγλική γλώσσα, χρησιμοποιήσαμε το σύνολο των δεδομένων που προαναφέραμε στην προηγούμενη ενότητα. Οι τάξεις συναισθημάτων είναι οι 7 που συνθέτουν το μοντέλο του Έκμαν.

Χωρίσαμε τις παραμέτρους τις οποίες θα ρυθμίσουμε για να βρούμε τον βέλτιστο ταξινομητή, σε δύο κατηγορίες. Η πρώτη αποτελείται από το είδος του φασματικού χαρακτηριστικού (Mel Spectrograms, MFCC, STFT Chromagrams), από το τελικό διαστασιακό μέγεθος των τετραγωνικών εικόνων που τροφοδοτούνται στο δίκτυο (30, 50, 60, 80) και τον τύπο αρχιτεκτονικής του ΣΝΔ

(BulbaNET, CharmaNET, SquirtleNET). Οι δευτερεύουσες παράμετροι αποτελούν παραμέτρους της εσωτερικής δομής των ΣΝΔ, καθώς και υπερπαραμέτρους.

Κύριες Παράμετροι:

- Φασματικά Χαρακτηριστικά.
- Διαστάσεις Εικόνων.
- Αρχιτεκτονική ΣΝΔ.

Εσωτερικοί Παράμετροι ΣΝΔ:

- Κανάλια Εξόδου κάθε συνελικτικού στρώματος
- Χαρακτηριστικά Εξόδου των δύο πρώτων πλήρως-συνδεδεμένων στρωμάτων.
- Μέγεθος πυρήνα συνελικτικών στρωμάτων.
- Μέγεθος πυρήνα στρωμάτων συγκέντρωσης.
- Βήμα (Stride) στρωμάτων συγκέντρωσης.
- Πιθανότητα στοιχείου να μηδενιστεί στο dropout στρώμα.
- Ορμή (Momentum) στο batch normalization στρώμα.
- Συντελεστής αρνητικής κλίσης σε Leaky ReLU.

Υπερπαράμετροι:

- Batch size.
- Ρυθμός Μάθησης.

Σταθεροί Παράμετροι:

- Βήμα (Stride) συνελικτικών στρωμάτων: 1.
- Βήμα (Stride) στα στρώματα συγκέντρωσης των SquirtleNET: 2.
- Προσθήκη μηδενικών (Padding) σε συνελικτικά στρώματα: 1.

Για να ρυθμίσουμε τις παραπάνω παραμέτρους εκπαιδεύσαμε και επικυρώσαμε διάφορα μοντέλα για πολλούς διαφορετικούς συνδυασμούς παραμέτρων. Ταυτόχρονα, εφαρμόσαμε τρεις κατηγορίες συνθηκών Πρόωρης Διακοπής ώστε να εξοικονομήσουμε χρόνο και να αποφύγουμε πιθανά φαινόμενα υπερπροσαρμογής.

Η **γενική μεθοδολογία** η οποία ακολουθήθηκε αποτελείται από τα παρακάτω βήματα:

1. Καθορίσαμε τον τύπο του φασματικού χαρακτηριστικού και το μέγεθος της εικόνας.
2. Εκτελέσαμε όλα τα βήματα Προεπεξεργασίας.
3. Διαχωρίσαμε τα δεδομένα σε σετ εκπαίδευσης και δοκιμών (80%-20%). Χρησιμοποιήθηκε μόνο το σετ εκπαίδευσης για συντονισμό των παραμέτρων.
4. Καθορίσαμε το πλέγμα των δευτερευόντων παραμέτρων και τη μέθοδο αναζήτησης.

5. Καθορίσαμε τις συνθήκες του αλγορίθμου Early-Stopping (όταν ήταν απαραίτητο)
6. Για κάθε διάνυσμα δευτερεύοντων παραμέτρων, χρησιμοποιήθηκε διασταυρωμένη επικύρωση 5-πτυχών και υπολογίστηκαν οι μέσες μετρήσεις αξιολόγησης για όλες τις πτυχές. Χρησιμοποιήθηκαν οι ακόλουθες μετρήσεις για σύγκριση:
    - Απώλεια Εκπαίδευσης (Train loss)
    - Απώλεια Δοκιμής (Test loss)
    - Ακρίβεια
    - Macro F1 score
7. Αναλύσαμε τις καμπύλες μάθησης για να κατανοήσουμε καλύτερα το πρόβλημα και να εντοπίσουμε πιθανή υπερπροσαρμογή (μόνο εάν είναι απαραίτητο).
8. Επιλέξαμε τους συνδυασμούς παραμέτρων με την υψηλότερη απόδοση με βάση τις μέσες μετρήσεις που υπολογίσαμε στη διαδικασία διασταυρωμένης επικύρωσης.

Συγκεκριμένα, ακολουθήσαμε την παραπάνω γενική μεθοδολογία σε τέσσερα διαφορετικά Στάδια Πρόκρισης (ΣΠ). Αυτά φαίνονται στην συνέχεια:

1. Πραγματοποιήθηκε μια αρχική τυχαία αναζήτηση (random search) δευτερευουσών παραμέτρων για κάθε καθορισμένο συνδυασμό κύριων παραμέτρων. Για τις δευτερεύουσες παραμέτρους, τα ίδια διανύσματα χρησιμοποιούνται για τις πρώτες δέκα επαναλήψεις κάθε συνδυασμού κύριων παραμέτρων. Ταυτόχρονα, δοκιμάστηκαν δύο διαφορετικές κλίσεις 0.005 και 0.050 της LeakyReLu, στις αρχιτεκτονικές CharmaNET, για 50x50 STFT Chromagrams και Mel Spectrograms. **(Πρώτο ΣΠ)**
2. Εκτελέστηκε ένας δεύτερος γύρος τυχαίας αναζήτησης χρησιμοποιώντας μόνο τις πιο υποσχόμενες κύριες παραμέτρους. Επιλέχθηκαν οι συνδυασμοί εκείνοι που αποδίδουν την υψηλότερη απόδοση και συγχωνεύτηκαν με τους προηγούμενους καλύτερους συνδυασμούς. **(Δεύτερο ΣΠ)**
3. Επιλέχθηκαν τα καλύτερα τρία διανύσματα παραμέτρων από την προηγούμενη λίστα και πραγματοποιήθηκε ένας επιπρόσθετος γύρος τυχαίας αναζήτησης με μικρές τροποποιήσεις στις υπάρχουσες βέλτιστες δευτερεύουσες παραμέτρους. Για άλλη μια φορά, προστέθηκαν στη λίστα οι συνδυασμοί με την υψηλότερη απόδοση. **(Τρίτο ΣΠ)**
4. Επιλέχθηκαν τα καλύτερα μοντέλα από τη λίστα και εκπαιδεύτηκαν για μεγαλύτερο αριθμό εποχών χωρίς να εφαρμοστεί καμία συνθήκη Πρόωρης Διακοπής. Στη συνέχεια, προσδιορίστηκε το βέλτιστο διάνυσμα παραμέτρου. **(Τέταρτο ΣΠ)**

Για την εκπαίδευση των μοντέλων, χρησιμοποιήσαμε τη συνάρτηση απώλειας `CrossEntropyLoss()` από την βιβλιοθήκη `torch.nn` και κατά τη διάρκεια της εκπαίδευσης χρησιμοποιήσαμε τον βελτιστοποιητή `Adam` χωρίς μείωση του ρυθμού εκμάθησης.

Παρατηρήθηκε ότι όσον αφορά τις κύριες παραμέτρους, οι σταθερές MFCC, το μέγεθος εικόνων 30, καθώς και τα BulbaNET αποτυγχάνουν να παρέχουν υψηλές αποδόσεις ταξινόμησης. Από την άλλη, τόσο τα Mel Spectrograms όσο και οι αρχιτεκτονικές SquirtleNET παρείχαν τα καλύτερα αποτελέσματα.

Έπειτα από εκτενή πειραματική μελέτη, καταλήξαμε στον παρακάτω βέλτιστο συνδυασμό κύριων και δευτερεύοντων παραμέτρων:

Table 1.1: Βέλτιστες Παράμετροι για Αναγνώριση Συναισθημάτων Αγγλικής Ομιλίας.

Αρχιτεκτονική ΣΝΔ	Squirtle NET
Φασματικό Χαρακτηριστικό	Mel Spectrogram
NxN	60x60
Κανάλια Εξόδου Συνελικτικών Στρωμάτων	188, 156, 340
Μεγέθη Πυρήνα Συνελικτικών Στρωμάτων	3, 3, 3
Μεγέθη Πυρήνα Στρωμάτων Συγκέντρωσης	2, 3, 3
Strides Στρωμάτων Συγκέντρωσης	2, 2, 2
Batch-Normalization Ορμή	0.4
Dropout Πιθανότητα	0.40
Χαρακτηριστικά Εξόδου των δύο πρώτων πλήρως-συνδεδεμένων στρωμάτων	3634, 2816, 7
Batch Size	32
Ρυθμός Μάθησης	6e-05

Η τελική Ακρίβεια στη δοκιμή της ταξινόμησης ισούται με **81.88%** και το τελικό Macro F1-Score ισούτασι με **81.83%**. Οι τιμές αυτές επιτεύχθηκαν για εκπαίδευση 400 εποχών.

F.2 Αξιολόγηση της αποτελεσματικότητας της Μεταφοράς Μάθησης στην Αναγνώριση Συναισθημάτων Ελληνικής Ομιλίας

Στο δεύτερο μέρος της έρευνας, πραγματευόμαστε την ερώτηση ”Μπορούμε να εκμεταλλευτούμε την εκπαίδευση του μοντέλου μας για να αναγνωρίσουμε συναισθήματα σε ομιλίες στην Ελληνική γλώσσα;”. Έτσι χρησιμοποιήσαμε το προ-εκπαιδευμένο μοντέλο μας στην Αγγλική, και εκπαιδεύσαμε με Ελληνικές ομιλίες με ετικέτες 5 κλάσεων συναισθημάτων. Παγώνοντας διάφορα στρώματα του μοντέλου, επιδιώξαμε να μάθουμε το πόσο ικανή είναι η γνώση του κάθε προ-εκπαιδευμένου στρώματος για την τελική ταξινόμηση στα νέα δεδομένα.

Χρησιμοποιήσαμε διασταυρωμένη επικύρωση 5 πτυχών για την εκπαίδευση, τη σύγκριση και την εξαγωγή μετρητικών μεγεθών, διάφορων μοντέλων, με διαφορετικούς ρυθμούς μαθήσεως και batch sizes, καθώς και διαφορετικούς συνδυασμούς παγωμένων στρωμάτων. Με λίγα λόγια, ρυθμίσαμε κάτα αυτό τον τρόπο τις παραμέτρους, ενώ ταυτόχρονα μαθαίναμε περισσότερα για το μοντέλο και για τις προοπτικές της μεταφοράς μάθησης πάνω στο συγκεκριμένο πρόβλημα. Λόγω μίας σχετικής ισορροπίας στον αριθμό δεδομένων κάθε κλάσης του νέου συνόλου δεδομένων, καθώς και λόγω του ότι, όπως είδαμε από τα αποτελέσματα ταξινόμησης στην Αγγλική γλώσσα, οι τιμές του Macro F1-Score και της Ακρίβεια είναι συνήθως πολύ κοντά μεταξύ τους, αποφασίσαμε να συμπεριλάβουμε ως κύρια μετρητική μόνο την Ακρίβεια %.

Εκτός των παραμέτρων που προαναφέρθηκαν, όλες οι υπόλοιπες παράμετροι διατήρησαν τις τιμές που είτε υιοθέτησαν μετά την ρύθμιση για τον ταξινομητή της Αγγλικής γλώσσας, είτε είχαν ήδη από την αρχή (πχ. ορίσματα συναρτήσεων προ-επεξεργασίας ή τύπος βελτιστοποιητή).

Σε πρώτη φάση, επιδιώχθηκε η περαιτέρω κατανόηση του νέου μας προβλήματος. Εκπαιδεύσαμε το προ-εκπαιδευμένο SquirtleNet με 60x60 Mel Spectrogramns, για 150 εποχές και για διάφορες

τιμές ρυθμού μάθησης και batch size. Ο πρώτος συνδυασμός παγωμένων στρωμάτων που δοκιμάσαμε ήταν για τα 4 πρώτα στρώματα (Conv1, Batchnorm, Conv2, Conv3), ενώ ο δεύτερος δεν αποτελούσε παγωμένο στρώμα, και συνεπώς όλα τα στρώματα του Νευρωνικού Δικτύου ήταν διαθέσιμα για εκπαίδευση.

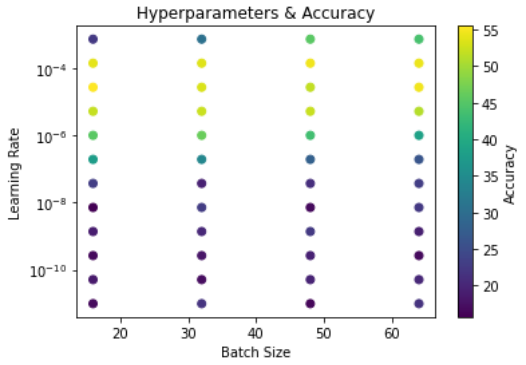


Figure 1.1: Ακρίβεια για διάφορες υπερπαράμετρους, για μοντέλο με παγωμένα τα 4 πρώτα στρώματα.

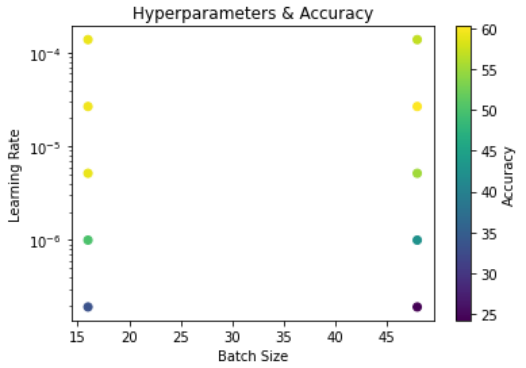


Figure 1.2: Ακρίβεια για διάφορες υπερπαράμετρους, για μοντέλο με κάθε στρώμα διαθέσιμο για εκπαίδευση.

Περιορίσαμε συνεπώς τους ρυθμούς μάθησης που δίνουν τα καλύτερα αποτελέσματα και δοκιμάσαμε τώρα εκπαίδευση 150 εποχών για λιγότερους συνδυασμούς υπερπαραμέτρων και περισσότερα υποσύνολα παγωμένων ΣΝΔ στρωμάτων.

Για απλούστευση και ευκολία, ονομάσαμε αυτά τα υποσύνολα με ένα πεζό γράμμα της Αγγλικής όπως φαίνεται στον πίνακα της επόμενης σελίδας.

Πιο συγκεκριμένα, εκπαιδεύσαμε τα μοντέλα (a), (b), (d), (e), (f), για 150 εποχές. Καταγράψαμε τις μέση τιμή Ακρίβειας για το κάθε μοντέλο, λαμβάνοντας υπόψιν τις Ακρίβεις σε κάθε δυάδα υπερπαραμέτρων.

	Convolutional 1	Batch-Normalization	Convolutional 2	Convolutional 3	Fully-Connected 1	Dropout
(a)	FROZEN	FROZEN	FROZEN	FROZEN	FROZEN	FROZEN
(b)	FROZEN	FROZEN	FROZEN	FROZEN	FROZEN	
(c)	FROZEN	FROZEN	FROZEN	FROZEN		
(d)	FROZEN	FROZEN	FROZEN			
(e)	FROZEN	FROZEN				
(f)	FROZEN					
(g)						

Οι ακρίβειες ήταν 38.94% για το μοντέλο (a), 48.88% για το μοντέλο (b), 52.64% για το μοντέλο (d), 54.06% για το μοντέλο (e), και 53.87% για το μοντέλο (f).

Ενώ διαπιστώνουμε ότι όσα παραπάνω επίπεδα παγώνουμε, τόσο περισσότερο δυσκολεύεται στην αναγνώριση ο ταξινομητής, έχουμε μία μικρή αύξηση στη μέση ακρίβεια όταν κρατάμε παγωμένο το στρώμα Batch-Normalization, από όταν είναι διαθέσιμο για επανεκπαίδευση. Ίσως αυτό είναι μία μικρή ένδειξη πως μπορούμε να εκμεταλλευτούμε κάποια ενσωματωμένη γνώση του μοντέλου, ωστόσο η άυξηση της μέσης ακρίβειας είναι μικρή, και εκτός αυτού, στη ρύθμιση παραμέτρων ενδιαφερόμαστε για την μέγιστη απόδοση και όχι για τη μέση.

Έτσι πραγματοποίησαμε ένα ακόμα βήμα για να μελετήσουμε κυρίως αν υπάρχει κάποιο ποσό αρχικής γνώσης στο προ-εκπαιδευμένο μοντέλο μας. Εκπαιδεύσαμε μόνο για ένα μικρό χρονικό διάστημα και συγκεκριμένα για 5 εποχές και καταγράψαμε τα αποτελέσματα των μοντέλων (c), (e), και (g) για διάφορες υπερπαραμέτρους.

Υπολογιστικά, αυτή η πειραματική φάση ήταν πολύ απλή και ελαφρύα, και δεν χρειαστήκαμε πολύ χρόνο για να λάβουμε αποτέλεσματα.

Οι μέσες ακρίβειες που λάβαμε ήταν 36.95% για το (c), 36.59% για το (e), και 29.96% για το (g).

Σε αντίθεση με την προηγούμενη φάση όπου η διάρκεια της εκπαίδευσης ήταν μεγαλύτερη, εδώ για μικρό αριθμό εποχών, η αύξηση του αριθμού των παγωμένων στρωμάτων οδηγεί σε βελτιωμένη απόδοση του προεκπαιδευμένου μοντέλου.

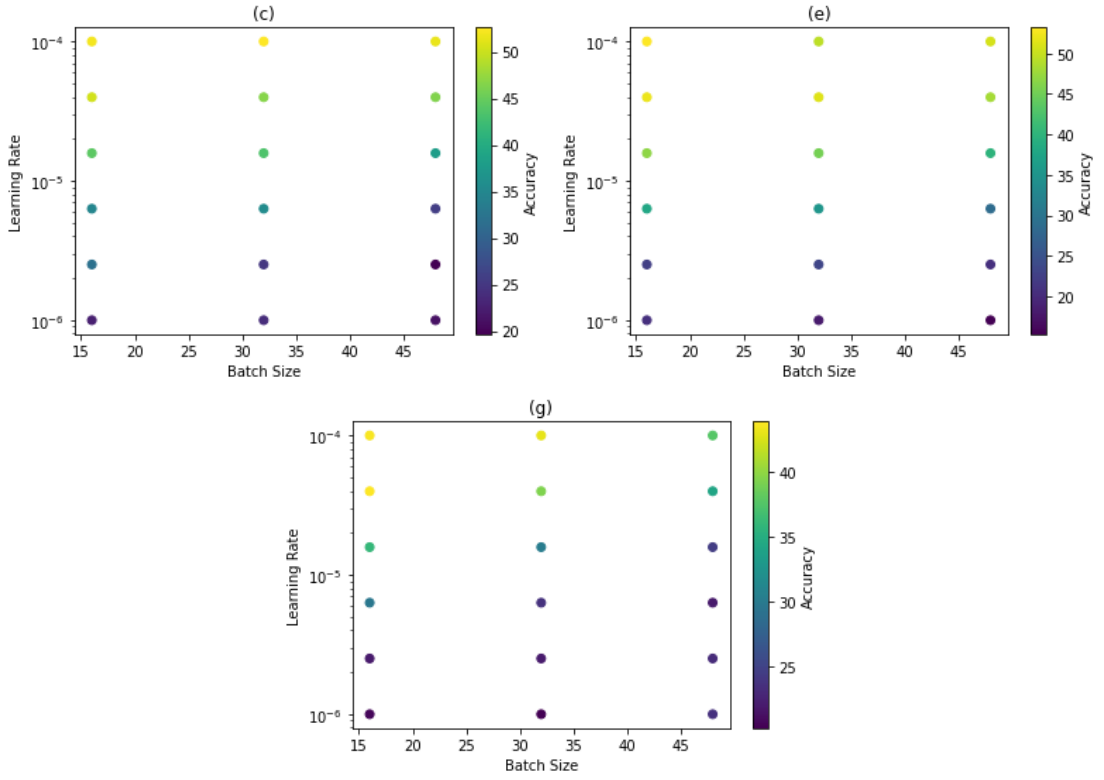


Figure 1.3: Ακρίβεια για διάφορες υπερπαράμετρους, για μοντέλα διαφορετικών υποσυνόλων παγωμένων στρωμάτων, για 5 εποχές εκπαίδευσης.

Έπειτα, εκπαιδεύσαμε ένα μη-εκπαιδευμένο SquirtleNET μοντέλο για 150 εποχές, με batch size ίσο με 32 και ρυθμοής μάθησης 1e-6, 1e-5, και 1e-4. Οι ακρίβειες που λάβαμε ήταν 57.97%, 55.90%, και 59.84%, αντίστοιχα. Συγκρίναμε τις τιμές αυτές με προηγούμενες τιμές Ακρίβειας προ-εκπαιδευμένων μοντέλων, και συμπεράναμε ότι είναι ελαφρώς υψηλότερες, οδηγώντας μας στο γενικότερο συμπέρασμα ότι η Μεταφορά Μάθησης για το βέλτιστο SquirtleNET μοντέλο, από επιβλεπόμενη εκπαίδευση με Αγγλικές ομιλίες που έχουν μετατραπεί σε 60x60 Mel Spectrograms, σε εκπαίδευση με Ελληνικές ομιλίες που μετατράπηκαν σε 60x60 Mel Spectrograms, μάλλον δεν είναι εφαρμόσιμη.

Τέλος, εκμεταλλευτήκαμε τα δεδομένα δοκιμής που αφήσαμε στην άκρη, και αφού αναζητήσαμε από όλες τις τιμές απόδοσης των μοντέλων που εκπαιδεύτηκαν μέχρι τώρα, επιλέξαμε το μοντέλο με την υψηλότερη απόδοση. Το μοντέλο αυτό ήταν προ-εκπαιδευμένο, εκπαιδεύτηκε με Ελληνικά δεδομένα για 150 εποχές, δίχως παγωμένα στρώματα, και για ρυθμό μάθησης ίσο με 2.68e-5 και batch size ίσο με 48.

Για 615 εποχές εκπαίδευσης με χρήση ολόκληρου του συνόλου εκπαίδευσης, λάβαμε το βέλτιστο μοντέλο με Ακρίβεια ίση με **66.12%** και Macro F1-score ίσο με **65.80%**.

F.3 Συμπεράσματα

Τα Mel Spectrograms έχουν δείξει την υψηλότερη απόδοση μεταξύ των φασματικών χαρακτηριστικών στην Αναγνώριση Συναισθημάτων Αγγλικής Ομιλίας, ενώ τα STFT Chromagrams έχουν σημειώσει επίσης σχετικά υψηλές αποδόσεις. Αντιθέτως, οι MFCC δεν έχουν συμβάλει σημαντικά στην κατασκευή ικανών ταξινομητών.

Όσον αφορά το μέγεθος της εικόνας, τα πειράματα με εικόνες 50x50, 60x60 και 80x80 έχουν δώσει πολλά υποσχόμενα αποτελέσματα. Ωστόσο, μοντέλα που εκπαιδεύτηκαν με 30x30 εικόνες απέτυχαν να πετύχουν υψηλές αποδόσεις.

Η βέλτιστη ΣΝΔ αρχιτεκτονική είναι η SquirtleNET. Ακολουθεί η αιθαίρετη αρχιτεκτονική CharmaNET, η οποία έδωσε επίσης ικανοποιητικά αποτελέσματα. Δυστυχώς, η BulbaNET δεν ανταποκρίθηκε στις προσδοκίες μας. Θα πρέπει να σημειωθεί ότι θα μπορούσε κανείς να επεξεργαστεί τη δομή αυτών των αρχιτεκτονικών και ενδεχομένως να προσθέσει περισσότερα στρώματα ή να αλλάξει τις συναρτήσεις ενεργοποίησης.

Η Μεταφορά Μάθησης στην Ελληνική γλώσσα, χρησιμοποιώντας το βέλτιστο προ-εκπαιδευμένο μοντέλο σε Αγγλικά δεδομένα, δεν απέδωσε τα αναμενόμενα αποτελέσματα και αποδείχθηκε μη-επαρκής στην πράξη. Κατά τη διάρκεια των πειραμάτων, δοκιμάσαμε διάφορα υποσύνολα παγωμένων στρωμάτων για την εκπαίδευση, χρησιμοποιώντας 60x60 Mel Spectrograms και το πλήρως καθορισμένο μοντέλο SquirtleNET. Παράλληλα, δοκιμάσαμε και την εκπαίδευση ενός μοντέλου που δεν είχε προηγουμένως εκπαιδευτεί με Αγγλικά δεδομένα. Παρατηρήσαμε ότι για λίγες εποχές εκπαίδευσης, μπορέσαμε να πετύχουμε ικανοποιητικές αποδόσεις και στις δύο περιπτώσεις.

Contents

List of Figures

List of Tables

1. Introduction

In a time where artificial intelligence applications such as super-intelligent chatbots, advanced home automation systems, autonomous vehicles, and art generation networks have become an integral part of our daily lives, the importance of researching Speech Emotion Processing is greater than ever before. While other subdomains of artificial intelligence and machine learning have made significant progress, Speech Emotion Processing remains a crucial and practical field to explore.

Understanding human emotions holds great significance as emotions play a fundamental role in our lives, interactions, and decisions. Speech is one of the main ways through which emotions are expressed and comprehended. Therefore, Speech Emotion Processing, a popular domain involving the field of machine learning, has been extensively studied for several years. It can be divided into two categories: Speech Emotion Recognition (SER) and Speech Emotion Synthesis (SES).

Speech Emotion Recognition focuses on comprehending the emotional content conveyed through speech, while Speech Emotion Synthesis involves generating realistic emotion-directed human speech. Both categories are equally significant and have a wide range of applications.

This thesis dissertation will primarily focus on studying Speech Emotion Recognition (SER). In healthcare, SER can be utilized to diagnose emotional disorders, assist with issues like anger, depression, and anxiety, and aid in improving overall mental well-being. Additionally, in customer service, SER can be used to satisfy the customer by incorporating emotional elements into robotic speeches, establishing a more personal connection with clients.

However, the research will mainly explore SER in English speech, with a secondary focus on Greek speech. The effects of different parameter combinations will be studied and various structures of Convolutional Neural Networks (CNNs) will be investigated. The performance of specific acoustic spectral features extracted from the initial speech data used to train the models will be also observed. These features will be treated as images and fed into the CNN, incorporating the domain of image recognition. Furthermore, we will delve into the field of transfer learning. After training the optimal English Speech Emotion Recognition CNN model, the capability of adapting this model to Greek speech will be explored.

The objective of this research is to uncover valuable insights into the domain of English Speech Emotion Recognition, analyze the impact of different parameter combinations, and investigate the adaptability of an English SER classifier to Greek speech data.

2. Machine Learning

Machine learning is a subfield of Artificial Intelligence, where agents are trained to perform a plethora of tasks, often complex and computationally heavy. Such tasks involve recognition, diagnosis, and prediction [18]. We say that those agents learn. As defined by [16], a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. This process is similar to how humans learn through trial and error. After all, machine learning reflects many aspects of biological learning.

In practice, learning agents can be represented as models that use certain algorithms to learn from training data and adjust their structures to make better predictions. A model's structure includes its architecture and other parameters that can be used for optimization. The vast majority of the applications are associated with three types of machine learning; supervised learning, unsupervised learning, and reinforcement learning.

Supervised Learning is a machine learning approach where an agent learns the relationship between inputs and outputs by observing pairs of input-output examples [24] . The objective here is to learn a function that accurately predicts the output for new unseen input data. This function is called a hypothesis, while we say that our data are labeled, meaning that with every available input vector, we also have access to its output. The two main subcategories of supervised learning are regression and classification. In regression, the agent aims to determine the relationship h between a set of independent variables x and a real continuous numerical dependent variable y, also known as a target, while in classification, the agent's goal, given a set of inputs x, is to estimate a function h for predicting the value of a dependent discrete variable y, commonly known as a label.

On the other hand, unsupervised learning is the approach where an agent learns patterns by only observing the inputs and their features. [24] Unlike supervised learning, there is no access to labeled output data, meaning that the algorithm has to identify hidden patterns in the input data, without explicit output information. The most common case of unsupervised learning is called clustering, where an agent is trying to identify data of similar nature and then group them to create clusters.

Lastly, reinforcement learning is the approach where a learning agent is rewarded after desired actions and penalized after undesired ones. The optimal result is achieved when the agent discovers the optimal sequence of actions, performed on its interacting environment, by maximizing the reward and minimizing the penalty.

2.1 A Quick Dive into Classification: Metrics and Applications

Let's suppose that we have a training set of M (input, output) pairs:

$(x(1), y(1)), (x(2), y(2)), ..., (x(M), y(M))$

There is an unknown function f that satisfies the equation $y = f(x)$ for every (x, y) pair of values. For a given input x, the corresponding output y, derived from $y = f(x)$, is said to be a ground truth [24]. However, we do not possess efficient knowledge over f.

Our goal here is to construct an approximation function h, also called a hypothesis, which will satisfy a new equation $\hat{y} = h(x)$, with $\hat{y}$ being our prediction. In practice, we will only have access to a limited amount of training data, and thus, it will be challenging to synthesize a capable predictor. We would like to determine a proper estimation function h so that the $\hat{y}$ values are almost identical to the ground truth values y, by applying good machine learning techniques and algorithms.

If y belongs to a closed set of discrete values, we call our optimization task a classification task. An instance y does not necessarily need to be a mathematical number but can be represented by any type of value, such as a word or a boolean. Therefore, in classification, we use the instance y to represent the class of the input x that we want to predict.

There are fundamentally two separate classification approaches; binary classification, which predicts between two opposite classes, and multi-class classification, which predicts between a set of more than two different classes.

One of the top-priority requirements for every machine learning model, including a suitable classifier, is the need for good metrics to evaluate its performance. Although a metrics selection might vary based on specific needs and preferences, certain metrics have emerged as the most commonly used and effective ones for evaluating classification models.

• **Accuracy:** One of the most popular and widely used classification metrics. It is defined as:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}, \quad \in [0, 1]$$

It should also be noted that accuracy might be misleading when dealing with very imbalanced datasets, where a certain class might appear much more frequently than others.

Also, based on whether a class's prediction is correct or incorrect, we define the following four key metrics:

• **True Positives or TP:** A true positive is a correctly predicted instance as positive.

• **True Negatives or TN:** A true negative is a correctly predicted instance as negative.

• **False Positives or FP:** A false positive is an incorrectly predicted negative class instance as positive.

• **False Negatives or FN:** A false negative is an incorrectly predicted positive class instance as negative.

While it can be clear that in binary classifiers, positives and negatives have opposite meanings, in multiclass classifiers we define a given class as a positive, and all the other classes as negatives. Hence, the evaluation metrics defined above may adopt different values for different classes.

The TP, TN, FP, and FN values for each class can be represented in a widely used metric called a confusion matrix.

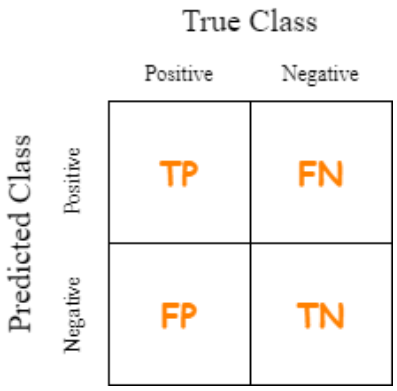


Figure 2.1: Confusion Matrix for Binary Classification.

• **Precision:** [10] The precision of a class can be defined as the following ratio:

$$Precision = \frac{TP}{TP+FP}, \quad \in [0, 1]$$

• **Recall:** [10] The recall of a class can be defined as the following ratio:

$$Recall = \frac{TP}{TP+FN}, \quad \in [0, 1]$$

• **F1-Score:** [10] The F1-Score is a commonly used metric that balances both precision and recall metrics and is particularly efficient in evaluating imbalanced datasets. It is defined as the harmonic mean of precision and recall.

$$F1 - Score = \frac{2}{\left(\frac{1}{\text{precision}} + \frac{1}{\text{recall}}\right)} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \quad \in [0, 1]$$

In case of multiclass classifiers, an F1-Score value corresponds to each class.

When dealing with multiple classes, it is often common to compute the average metrics for each class. Therefore, we will briefly discuss some of the performance metrics used in multiclass prediction tasks.

• **Average Multiclass Accuracy:** Similar to accuracy, the average multiclass accuracy is the arithmetic mean of each per-class accuracy score.

$$\text{Average Multiclass Accuracy} = \frac{\sum \text{Accuracy}}{\text{Total classes}}, \quad \in [0, 1]$$

In the context of multiclass classification, it can simply be referred to as accuracy.

• **Macro Average Precision:** The macro average precision [10] is the arithmetic mean of each per class precision.

$$\text{Macro Average Precision} = \frac{\sum \text{Precision}}{\text{Total classes}}, \quad \in [0, 1]$$

• **Macro Average Recall:** The macro average recall [10] is the arithmetic mean of each per class recall.

$$\text{Macro Average Recall} = \frac{\sum \text{Recall}}{\text{Total classes}}, \quad \in [0, 1]$$

• **Macro F1-Score:** An important metric for evaluating performance is the Macro F1-Score [10], which is similar to the F1-Score, but instead of computing the harmonic mean of precision and recall for each class, it computes the macro-average precisions and macro-average recalls. The Macro F1-Score is then calculated using the same formula:

$$\text{Macro F1-Score} = \frac{2 \cdot (\text{Macro Avg. Precision} \cdot \text{Macro Avg. Recall})}{\text{Macro Avg. Precision} + \text{Macro Avg. Recall}}, \quad \in [0, 1]$$

There are multiple other evaluation metrics. One could even use a custom formula, assuming it is tested and makes sense. In some cases, it may even be necessary to develop a custom performance metric that takes into account specific requirements or constraints of the existing task.

To assess a machine learning model, we obviously need to choose an appropriate metric. However, before this can be done, a learning algorithm to estimate the hypothesis function is required. Such algorithms are linear regression, where y is assumed to be a linear function of x, and logistic regression which is the equivalent simplistic method but for binary classification problems and returns the probability of an event occurring, based on the input x. Another example are decision trees, which can be used for both regression and classification tasks. By examining whether the data satisfy (or not) certain ”if” statement nodes, the output can finally be decided at the leaf node. Other widely known methods are support vector machines, which can also be a good choice for regression or classification tasks, or K means clustering, which is used to solve the clustering problem. There is no one-size-fits-all method, as different types of tasks may require different approaches.

Classification applications encompass a wide range of domains, including Image Classification, Medical Diagnosis, and Speech Emotion Recognition, which is the primary focus of this thesis dissertation. The objective of the thesis is to develop a classifier that processes speech audio data, extracts relevant features, and endeavors to identify the most probable emotional class within a specified range of emotions. Thus, a chosen metric will be utilized to depict the classifier's performance and assess its ability to correctly recognizing the emotional state of the speaker.

2.2 Neural Networks

A modern and common approach, particularly useful for problems where input data might have complicated structures and high dimensionality, is the use of **neural networks**.These structures were inspired by our understanding of how a human brain processes information [1]. Neural networks can be initially represented as black boxes, which consist of a set of nodes, each containing a specific mathematical value. The goal is to find the optimal values for these nodes such that, when an input is passed through the neural network, the desired output is produced. This can be achieved by connecting a set of weights from a single node to another set of nodes. These different sets of nodes represent a layer, and with multiple sets of weights we are able to connect two consecutive processing layers. A network with two or more layers is called a deep neural network and falls under the scope of deep learning, a subfield of machine learning for processing input information in a hierarchical way. Each subsequent level of processing extracts more abstract features and thus gives the model the ability to identify complex data patterns. This approach can be easily used in supervised, unsupervised and reinforcement learning tasks.

Let's compute a specific neuron's outcome, similarly to how a neural network would do. For example, let's take $n_{1,1}$ as shown in 2.2. The neuron's quantity can be computed from the following equation:

$$n_{1,1} = w_{1,1,1} \cdot x_1 + w_{2,1,1} \cdot x_2 + w_{0,1,1} \cdot x_0 \tag{2.1}$$

where $w_{i,j,k}$ is a weight that connects the k-th unit of the j-th layer with the i-th unit of its previous layer. The additional unit x_0 is called bias, and one such unit is often included on every layer except the output.

In addition, the general equation for the k-th neuron of the j-th layer $n_{j,k}$ is:

$$n_{j,k} = \sum_{i=0}^{N} w_{i,j,k} \cdot n_{j-1,i} \tag{2.2}$$

This process of computing the numerical quantity of each neuron in a neural network is known as forward propagation. This is because we can sequentially move forward through the network, computing the quantities of every neuron of each layer and passing them as inputs to the next layer until we reach the output.

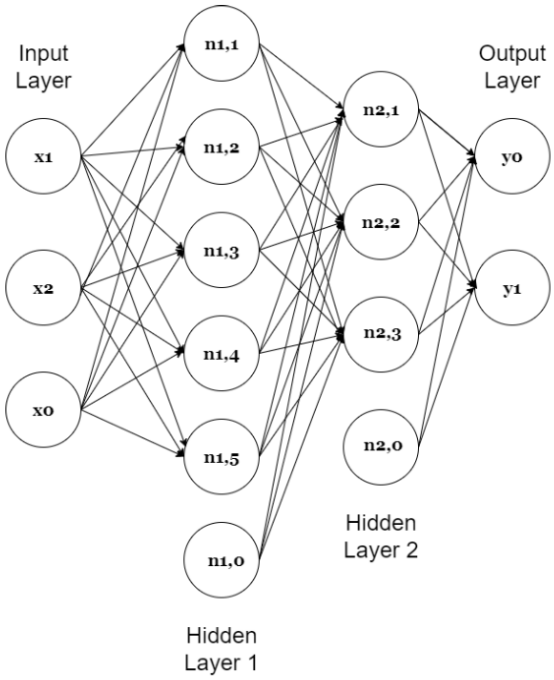


Figure 2.2: Simple representation of a Neural Network.

2.3 Activation Functions

One of the main strengths of deep neural networks is their ability to handle non-linearities, which allows them to represent complex relationships between inputs and outputs. In contrast, a linear function like the one used in 2.2 limits the expressive power of the model and is unable to capture complex relationships between a layer's input and output. Remember that each level can be responsible for identifying a different crucial pattern that has a major role in the final output. We introduce non-linearities to the neuron's outputs with the use of an activation function. This greatly increases the model's expressive power and can lead to more accurate predictions.

By using an activation function g, 2.2 can now be transformed as follows:

$$n_{j,k} = g\left(\sum_{i=0}^{N} w_{i,j,k} \cdot n_{j-1,i}\right) \tag{2.3}$$

The selection of activation function g is a critical factor in designing a neural network but greatly depends on the problem's nature. There are a variety of popular activation functions to choose from.

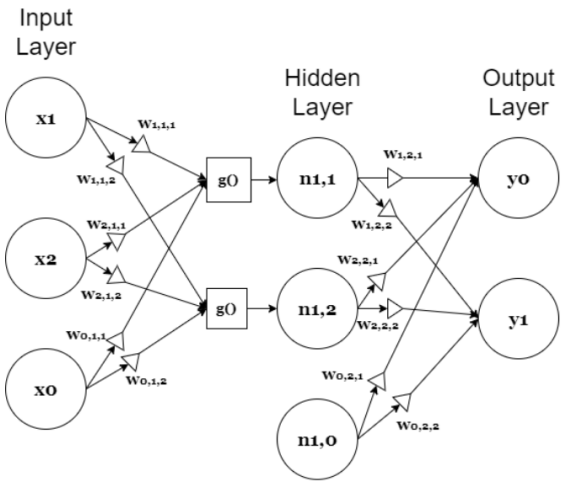


Figure 2.3: Representation of a Neural Network with weights, biases, and activation functions.

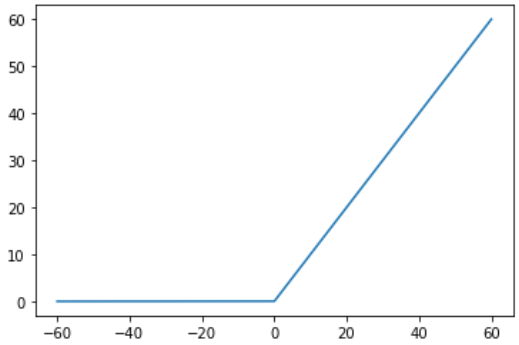
Some of them are the Sigmoid, which is theoretically used for binary classification tasks but is not used a lot in practice, softmax which can be used for multiclass classification, and tanh which prevailed the Sigmoid for its better performance.

For example, the sigmoid function is often used for binary classification tasks but is not as commonly used in practice due to its tendency to cause vanishing gradients. The softmax function is a popular choice for multi-class classification tasks, while the tanh function is often used in place of sigmoid due to its better performance.

In this thesis dissertation, we will make use of the Rectified Linear Unit or simply ReLu, arguably the most popular and commonly employed activation function, as well as a variant of it called Leaky ReLu.

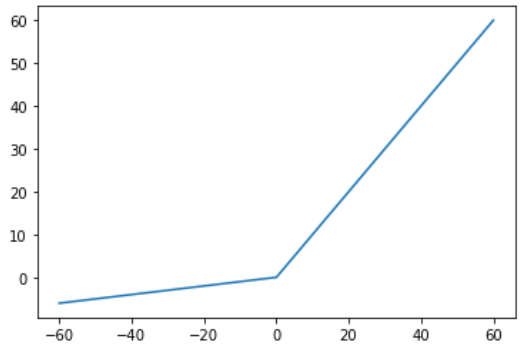
- **ReLu:**

$$g(x) = max(0, x)$$



• **Leaky ReLu:**

$$g(x) = max(x, ax)$$



We should note that ReLu and Leaky ReLu are identical for a=0.

2.4 Optimizers

The primary objective of a classifier is to make as many accurate predictions as possible. In neural network theory, this can be translated into minimizing the loss function. This function measures the difference between the predicted output and the ground truth. Such functions are the Mean Squared Error and the Categorical Cross Entropy.

As we have already discussed, a neural network's output is a function of the network's input and the weight vectors. The weights have to be repeatedly updated through a process called back-propagation which is based on the chain-rule of calculus and contributes to minimizing the loss function. It is computationally cheap and allows the information from the loss function to be propagated backward through the network. Combined with the forward-propagation method, a neural network can be trained, or in other words adjust its weights so that it makes better predictions.

More specifically, backpropagation is necessary to compute the gradient of the loss function, but an optimization algorithm called an optimizer, needs to be implemented to do the minimization and update the parameters. The infamous Gradient descent [22] is such an algorithm, given by the equation:

$$w = w - \eta \cdot \nabla_w J(w) \tag{2.4}$$

where η is the learning rate, ranging from 0 to 1, and J is the loss function. Additionally, w in this context can refer not only to the weights but also to the bias parameters.

Gradient descent has three main variations used in neural network training [22] Each one focuses on a different trade-off between the parameter's update efficiency and the computational time.

Batch Gradient Descent

The loss function's gradient has to be computed for the entire dataset for the weights to be updated. This can lead to great results but slow processing.

Stochastic Gradient Descent

A single random data sample is used for computing the loss function's gradient and performing a parameter update. This approach is faster and more efficient in practice, although it may lead to less accurate parameter estimates.

Mini-Batch Gradient Descent

By computing the gradient and updating the parameters based on a subset of data, also referred to as a batch, we can attain a balance between the accuracy of the updates and the time it takes to perform them. The batch's size can be any number m that belongs to the open interval between 1 and M, where M is the total number of data samples in the dataset.

The optimization steps are performed through a process known as training. In neural network theory, time is measured in epochs. An epoch is completed whenever the whole training set passes through the algorithm and is considered a hyperparameter.

Table 2.1: Updates per Epoch for Gradient Descent Variations.

Batch Gradient Descent	Stochastic Gradient Descent	Mini-Batch Gradient Descent
1	M	M/m

Another significant hyperparameter that also contributes to the accuracy-time trade-off is the learning rate. A large learning rate value may speed up the process but increase the risk of overshooting the optimal solution or failing to converge altogether, while a small one may need a longer time to converge, with an additional risk of getting trapped in a suboptimal local minimum and failing to achieve the best possible performance.

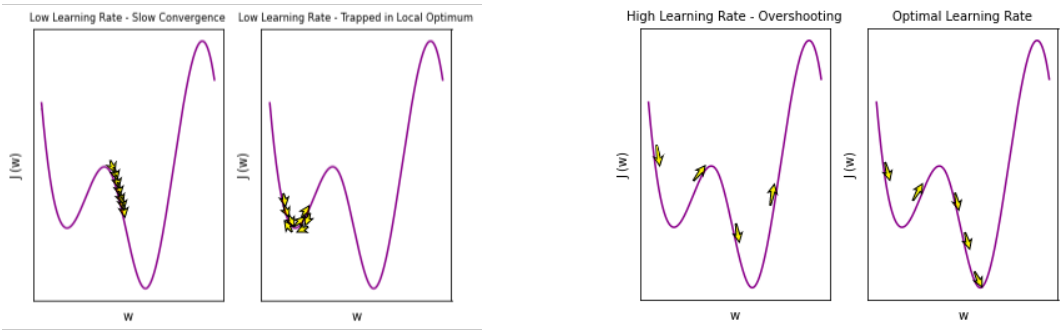


Figure 2.4: Effect of learning rates on optimization: Low Vs High Vs Optimal Learning Rate.

A clever method to help the algorithm maintain a consistent direction to convergence is the momentum algorithm or simply momentum [9]. Momentum is designed to accelerate learning by calculating a decaying moving average from past gradients.

Another solution to overcome the slow convergence and overshooting problems is adaptive learning optimizers. Adaptive learning optimizers are algorithms that can adjust the learning rate during training based on previous gradients. The optimizer can take smaller steps when the gradients are small, as it is closer to the optimal solution, and larger steps when the gradients are large to speed up convergence. In [12] an adaptive learning algorithm is proposed, named Adam. Adam has been shown to be highly effective in minimizing stochastic loss functions and adapting the learning rate in deep learning models. It will be used later for training convolutional neural network classifiers for speech emotion recognition.

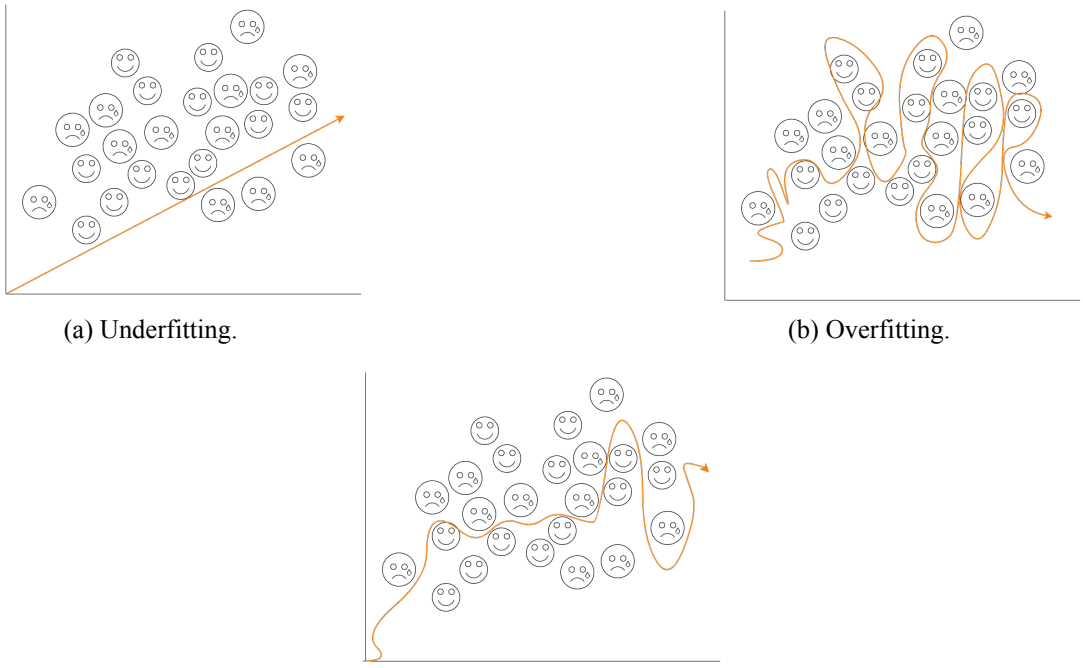
2.5 Fitting

To train a model, we must choose the loss function we wish to minimize, the optimizer which will perform the parameter renewal, and the hyperparameter values like learning rate, batch size, and total epochs. Those are all elements that will affect how much the model has learned. The model's ability to predict will be evaluated with new unseen data in a phase called testing. A model must be able to generalize from its experience. If not, then the testing error will be high and the performance will be low for new data.

If the training is not sufficient and the model does not perform well on training data, it will most likely not perform on testing data, too. In this case, we have to deal with a phenomenon called underfitting. This might mean that the model's hypothesis function is too simple and not complex enough to capture the training data. Possible solutions to underfitting might include training with more data, using more features of the training data, using other machine learning techniques and algorithms, including a different loss function or optimizer, or often making use of another hyperparameter combination.

On the other hand, a complex model might perform well on the training data but poorly on

new data. This phenomenon is called overfitting and could be partially solved by applying different training algorithms or changing the hyperparameter values. Regularization techniques, dropout, or early stopping can help prevent overfitting as well. Another solution might include decreasing the network's layers or neurons per layer and changing its architecture.



(a) Underfitting.

(b) Overfitting.

(c) Good fitting.

Figure 2.5: Examples of different hypothesis functions for binary classification on a 2-D plane.

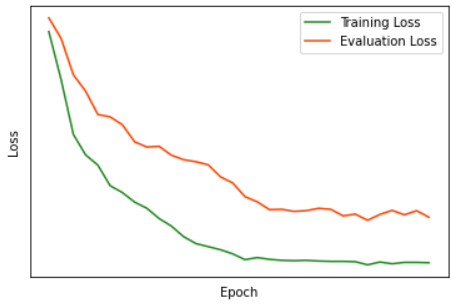
Underfitted or overfitted models can be visually detected with the use of learning curves. One of the most common diagnostic curves is the plot of train and evaluation losses with respect to the number of training epochs. Eventually, the training loss will converge to lower values as the training progresses. However, the ultimate goal is to achieve a low evaluation loss. If the gap between the train and evaluation losses is large, it may indicate overfitting, whereas a high train loss often indicates underfitting. The same can be seen with other types of diagnostics. For multiclass classification, we can plot an accuracy or macro F1-score curve concerning the number of epochs to observe where it begins to converge.

Any machine learning model, including neural networks, must not be trained for an unsuitable duration. Insufficient or excessive training can both lead to undesired results. Training for too long might cause the evaluation loss to increase. In this case, we need to stop the training, as the model starts overfitting. There are two obvious paths we can follow.

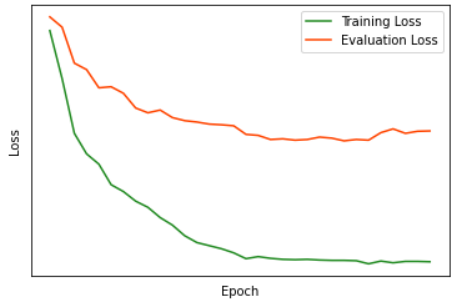
First, we can stop training manually. An easy way to do so is by testing the model on a new set of data, once per training epoch and displaying the train and evaluation losses.

By observing them we can then decide when it is time to stop. This process is called early-stopping.

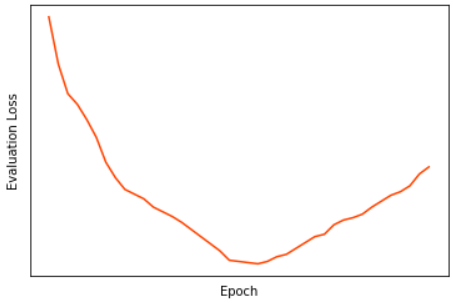
The second option is to detect overfitting automatically by employing an early-stopping algorithm. The algorithm may consist of various statements related to loss or performance metrics and may vary from application to application. It is necessary to observe the peculiarities of our problem or the model's behavior to determine the structure of our early-stopping method. Choosing a weak algorithm will certainly lead to underfitting or overfitting.



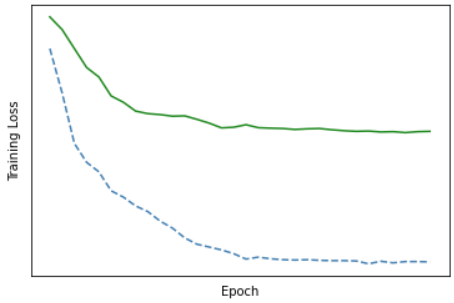
(a) Both the training and evaluation loss converge to a low value, indicating a good performance.



(b) The gap between the training and evaluation loss is large and thus this model is overfitted. In this case, we might need to try new hyperparameter values or experiment with the model's structure to improve the evaluation performance and achieve better generalization.



(c) Another case of overfitting. The evaluation loss starts increasing after several epochs of training. Therefore, we can use early-stopping to store our optimal weights during the critical epoch.



(d) The training loss is high enough, indicating underfitting. To address this, we might have to try changing the model's structure or its hyperparameters.

Figure 2.6: Learning curve examples for identifying overfitting, underfitting, and good fit.

2.6 K-Fold Cross Validation

Overfitting and underfitting might force us to experiment with various model parameters such as the number of layers, activation functions, or neurons per layer, or different hyperparameters such as learning rate, momentum, or batch size. In addition, we might need to adjust our early-stopping criteria and even feed completely new input features to the network. Each opposed modification leading to a training process manufactures a new algorithm that aims to estimate a better hypothesis function and improve generalization.

To compare learning algorithms it is essential to use a separate set of new unseen data for testing the already trained models. This new dataset is called a validation set, as it is used to validate the models and determine the optimal learning algorithm. It is crucial to ensure that the validation dataset has not been used for training and will also not be used in the final evaluation, or else we risk overfitting. The model with the highest performance metric value or the lowest error or loss value on the validation dataset wins and can be re-trained using both the train and validation sets before being finally evaluated on the unseen testing data.

The validation technique which will be used in this dissertation is the K-fold cross-validation. If the initial training data are already randomly parted, we then split them into K-equal subsets and use one of the subsets as the validation set. The rest K-1 subsets are used for training the model. We repeat this process K times, by using a different validation set and we measure the average accuracy or macro F1-score after each training process.

2.7 Parameter Tuning

As we have already mentioned, a variety of parameters might need to be tuned to improve performance.

Internal Model Parameters

- Number of neurons
- Number of layers
- Layer types
- Layer parameters
- Activation functions

Inputs

- Feature Selection
- Pre-Processing techniques
- Pre-Processing parameters

Hyperparameters

- Learning rate
- Batch size
- Number of epochs
- Momentum

It is impossible to test every parameter and its effect on learning. However, most of them can be partially explored with certain searching algorithms, some of which will be explained below. The objective is to efficiently explore a wide range of parameters in a fast and cost-effective manner.

2.7.1 Grid Search

If a grid represents a set of potential (hyper)parameter combinations (as shown in Figure 2.7), then we can initiate searching by training the model for each one of them. Once we distinguish the best set of performing parameters, we can zoom in and start searching again on a smaller grid around the best combination, with a much shorter step. This is a reasonable approach only when tuning a few parameters while the grid is small. Nonetheless, with the number of parameters comes an exponential growth in computational load [9].

Table 2.2: Grid Search Algorithm for Parameter Tuning

Step	Description
1	Define the parameter space by specifying the range or specific values for each parameter.
2	Create a grid of all possible combinations. Each combination represents a unique set of parameters.
3	For each combination in the specified grid, train and evaluate the model in the training dataset, using the K-Fold cross-validation or any other validation technique for evaluation.
4	Compare the performance of models trained with different hyperparameter combinations based on the chosen evaluation metric and select the combination that yields the highest performance.

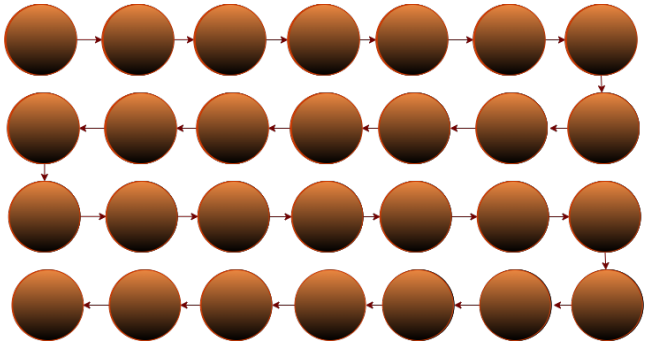


Figure 2.7: Grid search consecutive iterations in a 2D plane. The nodes represent the parameter combinations and the arrows define the movement inside the grid.

2.7.2 Random Search

To reduce the computational cost, we often use a technique called random search. After having initialized our grid area, we define the number of hyperparameter combinations that we wish to explore within the grid. Those sets of parameters are picked randomly. In contrast with the grid search method, we can cover much larger grid areas. Once we determine the best set, we can repeat the process on a smaller neighboring grid for further optimization.

Although grid search and random search methods are often used for hyperparameter tuning, we can use them for tuning other parameters, such as the number of neurons in every layer of a neural network.

Table 2.3: Random Search Algorithm for Parameter Tuning

Step	Description
1	Define the parameter space by specifying the range or specific values for each parameter.
2	Set the number of iterations for a random search.
3	For each iteration: a. Randomly sample a set of parameters from the defined space. b. Train and evaluate the model in the training dataset, using the K-Fold cross-validation or any other validation technique for evaluation.
4	Compare the performance of models trained with different hyperparameter combinations based on the chosen evaluation metric and select the combination that yields the highest performance.

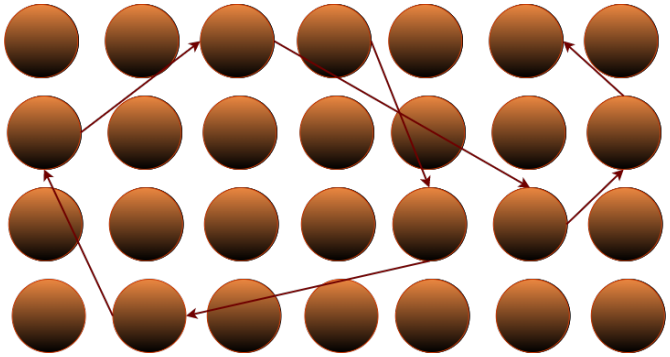


Figure 2.8: Random search consecutive iterations in a 2D plane. The nodes represent the parameter combinations and the arrows define the movement inside the grid.

2.8 Transfer Learning

Transfer learning allows knowledge learned from previous tasks to be applied to new tasks, even when the domains, tasks, and distributions differ. It focuses on extracting knowledge from source tasks and applying it to a target task. [2]

When referring to transfer learning and the knowledge we want to transfer, as reported in [2] it is common to subcategorize the term into three subcategories: inductive transfer learning, transductive transfer learning, and unsupervised transfer learning. Inductive transfer learning involves dealing with different source and target tasks, requiring labeled data from the target domain to build the model. Transductive transfer learning, on the other hand, involves the same tasks but different domains, where labeled data is only available in the source domain. Unsupervised transfer learning focuses on different but related tasks, without any labeled data available in either domain. In this discussion, we will refer to inductive transfer learning as transfer learning, as we will explore how knowledge can be transferred from one task to another when possessing labeled data for both tasks.

Moreover, a model that has been trained on a specific task (1) tends to perform well on that task. However, when we encounter a new task (2) for which we have limited training data, transfer learning can be employed. Transfer learning leverages the knowledge gained from the pre-trained model and can contribute to modifying or re-training certain layers to enhance performance on the new task. This approach is particularly effective when tasks (1) and (2) share similarities. For instance, an individual who aims to excel in a running race can capitalize on their years of experience playing soccer, reducing the amount of training required for running.

To adapt a model to a new task, it may be necessary to experiment with modifying different parts of its internal structure. Some of the pre-trained layers may contain less relevant information and will thus need re-training, possibly involving adjustments to the number of neurons. Typically, the target of the data differs in a new task, particularly in classification scenarios where new classes are introduced. As a result, the output layer requires updates and re-training.

Many times, when adapting a model to a new task, it is important to determine the optimal configuration for the frozen layers in the network. K-fold cross-validation can be employed to compare the average performance results for K folds and identify which layers contain valuable information and which ones may need modification and re-training. It is worth noting that there are no limitations when fine-tuning the model. Multiple hyperparameter values can be experimented with, additional layers can be added or unnecessary ones can be removed, and various adjustments can be made to achieve the desired performance on the new task.

In this thesis dissertation, one of our objectives will be to analyze the specific layers of a pre-trained convolutional network, along with other hyperparameters, to determine the parts of the model that contribute to achieving high performance on a new task.

3. Convolutional Neural Networks

Convolutional neural networks (CNN) are a state-of-the-art technology for capturing patterns of multi-dimensional grid-structure topology data, in computer vision. Typically, their first hidden layers use the operation of convolution to extract features such as shapes, objects and edges. While lower level layers often detect more general patterns, deeper layers can capture more specific ones. Their preference amongst researchers and engineers in the field mostly relies on their computational effectiveness. and combined with the modern hardware's high capabilities, CNN can be run very quickly and efficiently. This approach has been most successful on two-dimensional images. [9]

We will gain a brief insight over the convolutional neural networks by studying the type of layers usually encountered in their structure and by having a look at the basic math behind them.

3.1 Convolutional Layers

The mathematical operation of discrete convolution is applied throughout hidden layers called convolutional layers. The 2D discrete convolution of two 2D signals x and k can be written as:

$$y[n_1, n_2] = \sum_i \sum_j x[n_1 + i, n_2 + j] \cdot k[i, j] \tag{3.1}$$

In practice, x is the input in the layer, usually referred to as the input feature map and k is a small matrix of weights called a convolutional kernel or simply a kernel, which is regularly smaller than the input. A kernel's primary objective is to reduce the size of the feature map and eventually decrease the number of layer-to-layer connections. Hence, the output feature map y must have a smaller shape than x.

The size of the convolution's output array along axis i, denoted as O_i, depends on the following entities [4]:

- I_i: Input size along axis i
- K_i: Kernel size along axis i

- P_i: Zero-Padding along axis i
- S_i: Stride along axis i

The true formula [4] is as follows:

$$O_i = \left\lfloor \frac{I_i - K_i + 2P_i}{S_i} \right\rfloor + 1 \tag{3.2}$$

Stride $S_i \geq 1$ is the step size of a kernel moving across the input feature map during convolution. It provides the option to further decrease the computational requirements by basically downsampling the input.

Zero-Padding $P_i \geq 0$ defines the additional layers of zero values surrounding the input feature map, providing the option to balance the decrease in size by the convolution operation and also dealing with the problem of loss of information in the corners.

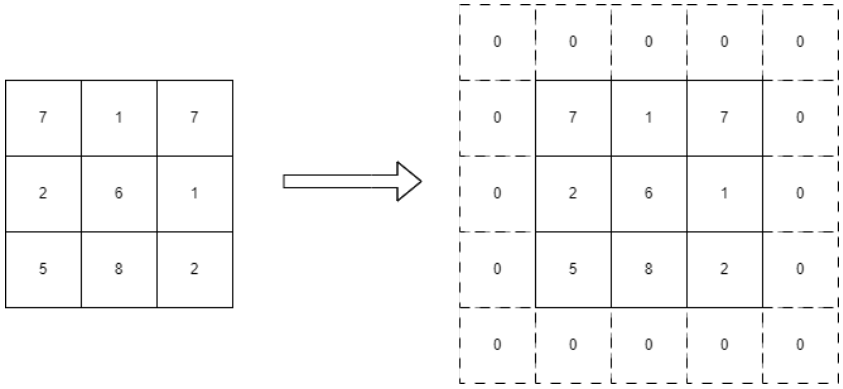


Figure 3.1: CNN - Zero-padding for Pi=1.

We will make two necessary assumptions about our 2D convolution:

- Inputs are square: $N_1 = N_2 = N$
- Kernels are square: $K_1 = K_2 = K$
- Stride is the same along both axes: $S_1 = S_2 = S$
- Zero-padding is the same along both axes: $P_1 = P_2 = P$

Based on the above, equation 3.2 can be written as:

$$O = \left\lfloor \frac{I - K + 2P}{S} \right\rfloor + 1 \tag{3.3}$$

After a convolutional layer, we can use an activation function to introduce non-linearity to our outputs. Kernel, padding, and stride sizes can all be tuned and may have a major impact on the results of the network. The number of layers or the number of output channels is also critical for better performance.

3.2 Pooling Layers

It is common practice to further reduce the size of the data after a convolutional layer. Moreover, we can add an extra layer that aims to maintain all the necessary information while reducing the feature map's size. This filtering layer is called pooling, and it is the operation of a new kernel that transfers a block's total information to a single pixel.

The pooling formula [4] to get the output's size can be written as:

$$O = \left\lfloor \frac{I - K}{S} \right\rfloor + 1 \tag{3.4}$$

In Figure 3.2, we can see an example of a 2×2 filter along with a stride of 2 being applied to the input map. The output is calculated in two ways: average pooling and max pooling.

Similar to before, kernel and stride sizes in pooling can be tuned, along with the type of pooling (e.g., average pooling or max pooling).

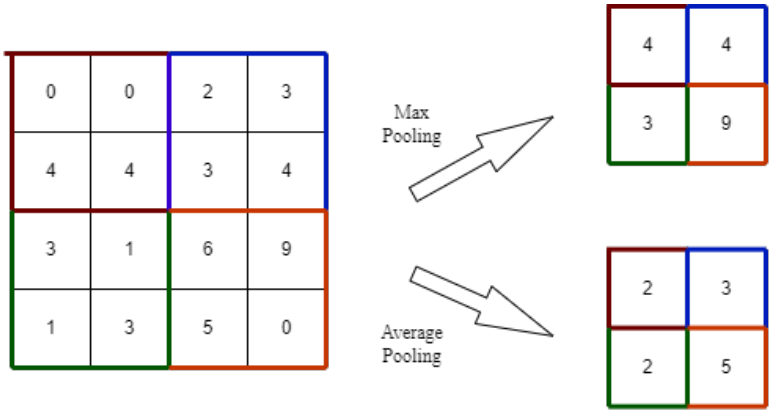


Figure 3.2: CNN - Max and Average pooling for K=2 and S=2.

3.3 Fully Connected Layers

Practically, we might deal with 4D inputs instead of 2D. The reason is that we might have more than one input channel (e.g., RGB images), while also having a number of input data adding up to the batch size. Before we pass a 4D input to a fully-connected layer, we need to apply a simple process called flattening, which reshapes the 4D matrix to a 2D matrix

by concatenating all the feature maps for each data sample in the batch. The shapes before and after flattening are:

$$(\text{batch size}, \text{num\_channels}, \text{height}, \text{width}) \rightarrow (\text{batch size}, \text{num\_channels} \times \text{height} \times \text{width})$$

The flattened output can then be passed as an input to a fully-connected layer (also called a dense layer), a type of hidden layer where each neuron applies a linear transformation to the input. If followed by an activation function, this type of layer is crucial for the final classification. In addition, the number of neurons might be necessary for tuning and ensuring great performance.

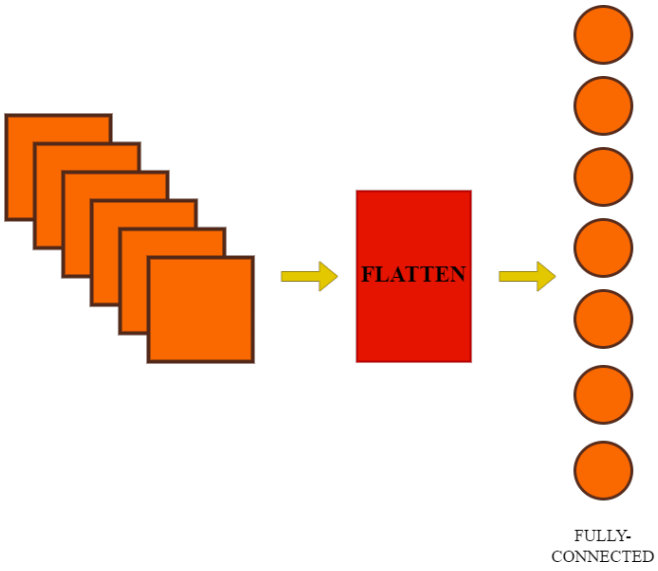


Figure 3.3: CNN structure for flattening the multiple-channel input and passing it to the first fully-connected layer.

3.4 Regularization Layers

By incorporating regularization layers into CNNs, we can improve their generalization ability and train them in a manner that ensures enhanced performance on unseen data. Widely used such layers are the Batch-Normalization and Dropout.

3.4.1 Batch-Normalization

During the training of a deep neural network, the gradients are used to update the parameters of each layer. However, when multiple layers are updated simultaneously, unexpected results can occur. This is because the functions that compose the network are changed simultaneously using updates that were computed under the assumption that the other layers remain constant [9].

Batch normalization is a method which addresses the challenge of coordinating updates across many layers in a deep neural network and can be applied to any input or hidden layer in the network. Let H be a minibatch of activations of the layer to be normalized, arranged as a design matrix, where each example's activations appear as a row in the matrix. To normalize H, we transform it into H' by subtracting the mean μ and dividing by the standard deviation σ, as follows [9]:

$$H' = \frac{H - \mu}{\sigma}$$

The mean μ is calculated by taking the average of each unit's activations across a minibatch, while the standard deviation σ is computed using the average squared difference from the mean.

In practice, a batch normalization layer can be inserted between any two layers within a convolutional neural network. Its placement can vary depending on the specific experimental setup, either before or after the activation function. Additionally, batch normalization is commonly used as a regularization technique. By introducing noise to the network activations, enhances generalization and helps prevent overfitting.

3.4.2 Dropout

Another common regularization technique is dropout. By randomly dropping out units during training, it reduces co-adaptation among neurons and prevents the network from overfitting to the training data.

Dropout is a process where a set of non-output units is temporarily deactivated from the network by being multiplied by zero. The remaining units remain active, creating a binary mask of ones and zeros. The probability of including a unit can be chosen to be anywhere between 0 and 1, with 0.5 being a commonly used value. During training, the dropout is applied to encourage the network to learn more generalizable features by encouranging it to rely on a subset of units rather than specific neurons for making predictions.

However, dropout is not applied when testing the network. The testing process is performed without dropout in order to evaluate the network's performance on unseen data accurately.

4. Speech Emotion Recognition

Speech processing is widely used with machine learning, whether it involves speech recognition or synthesis. There are several applications worth studying. One such application is speech emotion recognition, which creates models to identify the emotional content of a speaker's voice. This can be derived from either acoustic or linguistic information and can often be combined with facial expressions.

Many theoretical models have been suggested for emotion classification. One notable model is Ekman's six major emotional states: anger, disgust, fear, happiness, sadness, and surprise, along with a seventh neutral state [6]. Models like Ekman's have traditionally been a common way to model emotional content [25]. In this model, emotions are represented discretely. However, there are other models, such as Russell's circumplex model, which organizes emotional content into two dimensions: valence and arousal [23], that aim to represent emotions as continuous values, leaning more towards regression than classification.

While the field has significantly improved in recent years, it still faces major challenges [25].

Most of the algorithms for spectral feature extraction and audio manipulation discussed in this chapter are provided by the Librosa library [15]. Librosa is a widely-used Python package specifically designed for music and audio analysis.

Additionally, in the context of resizing techniques mentioned in this chapter, we utilized the Pillow (PIL) library [20]. This powerful Python library offers extensive image processing capabilities, allowing us to efficiently manipulate and resize audio-related visual representations.

4.1 Speech Emotion Databases

There are a variety of databases available for speech emotion classification. Most of them contain speech produced by professional or semi-professional actors because capturing real-world emotions in speech might raise legal and moral issues. Most of these databases emphasize on the acoustic properties of speech emotions and thus making linguistic context profitless or unavailable. The same sentence is often simulated with multiple emotional contexts to allow researchers to study the effect of acoustic features.

The main challenges to existing SER data are:

1. Datasets can present an unnatural and imbalanced distribution regarding multiple factors such as age (most data includes adult voices), gender, or emotional state (in real life, some emotions, such as neutrality, occur more frequently than others).
2. Noisy environments and low sampling rates may downgrade the classification accuracy.
3. Poor human recognition performance due to the fact that acted emotions might not be simulated in a clear and natural way. In supervised learning, a machine's optimal ability approaches the human's ability.
4. The lack of linguistic information might result in unnatural classification.
5. The lack of diversity in language and ethnicity often leads to biased models.
6. The number of examples may be very limited, which could lead to a model that is not representative of the real world.
7. The limited emotion categories do not reflect the real-world complexity of emotions.

4.2 Spectral Acoustic Features

The extracted acoustic features must be robust in varying factors like noise, language, or the speaker's state (e.g., cold, stressed, tired, intoxicated, etc.). They can be related to a variety of elements such as the duration of the signal's components, the fundamental frequency, the voice quality, the intensity, the rhythm, and the frequency.

Frequency-related acoustic features or spectral acoustic features play a significant role in speech emotion recognition [7]. They are extracted from the spectral information of the speech signal and often reveal useful information about the speaker's emotional state.

In the fields of music and speech emotion recognition, it is common to use the mel-scale [26] to represent the frequency dimension. We can do this by applying the following formula to our frequency values:

$$\text{mel scaled frequency} = 2595 \log_{10}(1 + \frac{f}{700})$$

If the spectral content of a sound signal's feature is represented as a function of time, then it is a 1D acoustic feature. The fundamental difference between 1D and 2D acoustic features is that 2D features represent the spectral content as a function of both time and frequency.

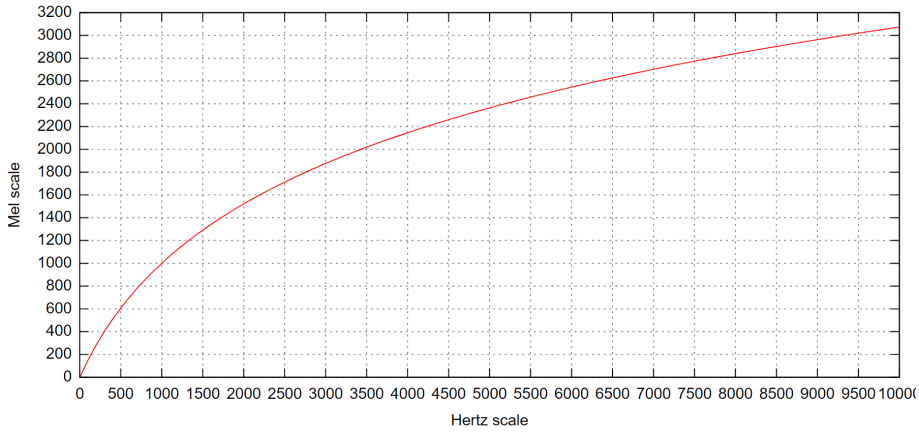


Figure 4.1: Mel-Scale.
[27]

4.2.1 Mel-Frequency Cepstrum Coefficients (MFCC)

MFCCs are popular for their high performance and their resemblance to the human voice system. Based on [11] their extraction involves the following stages:

1. Compute the FFT of the speech waveform.
2. Map onto Mel scale using a constructed Mel filter bank.
3. Convert the log mel-spectrum back to time.

The resulting amplitudes are the MFCCs. Usually, only the first few coefficients are needed as they provide sufficient information, but it depends on the specific application. In addition, the extraction algorithm for MFCCs may vary across different applications and implementations.

Thus, we can extract MFCCs along the time axis and organize them into a 2D image. This image can then be fed into a Convolutional Neural Network (CNN) to identify the emotional content embedded in the image. This approach transforms the overall Speech Emotion Recognition (SER) task into an image recognition task.

4.2.2 Mel-Spectrogram

Another popular feature in speech processing is the mel spectrogram. The simplified process of computing based on [15] involves:

1. Compute magnitude spectrogram.
2. Map onto Mel scale using a constructed Mel filter bank.

A mel spectrogram provides a visual representation of the spectral content of a speech signal over time. It can be used as input for various speech analysis and emotion recognition tasks.

Similarly to MFCCs, mel spectrograms are another type of 2D image feature that can be used in image recognition tasks to identify the emotional class.

4.2.3 Chromagram

Chroma features or chromagrams are valuable representations of frequency content in speech signals or music pieces. There is a variety of methods available for converting an audio recording into a chromagram. For example, as reported in [17], the steps for computing a chromagram are the following:

1. Compute the spectrogram.
2. Use the log-frequency axis.
3. Apply cycle projection (a specific mathematical operation applied).

Later in the Experimental Analysis chapter, we will use the Chroma STFT [8] feature provided by Librosa [15]. This involves computing a spectrogram of the audio signal using the Short-Time Fourier Transform (STFT), which produces a representation of the signal in the time-frequency domain and features a slightly different process of extraction.

Chromagrams are the last 2D features that will be utilized in the construction of a SER classifier in the Experimental Analysis.

4.3 Audio Manipulation Techniques

There is a broad scope of features that can be modified in an audio signal. While speech signals are often complex and contain a vast amount of information, we can utilize intelligent algorithms to extract, add or remove specific traits from the audio signal.

Some of the following techniques can be employed to manipulate a digital sound waveform and generate a different one, while others aim to emphasize or remove information from the speech signal to facilitate the identification of valuable information patterns by the classifier. In machine learning, we refer to the former as audio augmentation techniques, which assist the model in becoming more robust to variations encountered in real-world environments. Essentially, we generate new data that can be used for both training and testing purposes. Regarding the latter type of technique, they fall under the scope of speech pre-processing, which is essential before feeding a signal to a classifier to enhance the classification performance.

4.3.1 Pre-Emphasis

A pre-processing technique commonly employed to enhance the high-frequency components of an audio signal. Its purpose is to balance the natural attenuation of high frequencies that occurs during the recording process in digital audio systems. By boosting the high frequencies, the technique aims to restore a more balanced and natural sound representation in the final audio output.

4.3.2 Zero-Trimming

A technique used to remove parts of a signal that contain unnecessary information. It involves eliminating elements from the signal that are either equal to zero or fall below a defined threshold. By removing this extraneous data, zero trimming reduces the computational requirements needed for processing the signal.

4.3.3 Pitch-Shifting

Pitch-shifting is an audio manipulation technique that changes the pitch of an audio (or its frequency content) without affecting its initial duration. By predetermining the desired frequency content of a step and the total steps we want the pitch to increase or decrease, we can achieve the desired result. In music applications, a step is often equal to a semitone.

4.3.4 Time-Stretching

Time-stretching is an audio manipulation technique that alters the speed of an audio (or its total duration) without affecting its pitch. By predetermining a desired speed factor, we can modify the original speed of the audio by either compressing its duration (if the speed factor is greater than 1) or expanding it (if the speed factor is between 0 and 1). Essentially, increasing the speed leads to a decrease in duration, and vice versa.

4.3.5 Gaussian Noise Addition

Gaussian noise addition is an audio manipulation technique that involves the addition of Gaussian noise to the original signal. Gaussian noise is a sequence of random numbers drawn from a Gaussian distribution. To generate and add such noise, it is often necessary to specify the mean and standard deviation of the noise, as well as the length of the noise signal. When adding Gaussian noise to the entire audio signal, we set the length of the noise equal to the length of the signal.

To determine the Signal-to-Noise Ratio (SNR) for the desired audio signal, we can calculate it using the formula:

$$\text{SNR} = \frac{\text{Signal Power}}{\text{Noise Power}}$$

where SNR is the ratio of the signal power to the noise power. We can also express the SNR in decibels (dB) using the formula:

$$\text{SNR\_dB} = \text{Signal Power\_dB} - \text{Noise Power\_dB}$$

Knowing the SNR and the signal power (which is basically the square of the signal), we can compute the noise power. Subsequently, we can easily calculate the standard deviation by setting it equal to the square root of 10 raised to the power of (Noise Power_dB / 10).

Finally, we can generate the Gaussian noise sequence and add it to the original signal to obtain the final manipulated audio signal.

4.4 Image Resizing Techniques

When we extract a 2D spectral feature from a speech signal, it typically results in a new image, which may not have a square shape. The objective of this dissertation is to input these spectral images into a convolutional neural network. Depending on the various parameters in spectral feature extraction methods, such as the length of the FFT window or the number of samples between successive frames in a library like librosa, the resulting image may be large and necessitate downsampling to reduce the computational requirements during propagation. Consequently, it may be necessary to adjust the length of the time dimension of the image to match the frequency dimension. This is because each speech signal has a different duration, which means their time axis sizes are likely to differ. The aim is to obtain a set of equally sized square images that can be easily classified by the network, while still preserving significant information. Techniques like image resizing or interpolation can be employed to achieve this.

4.4.1 Nearest Neighbor Interpolation

Nearest neighbor interpolation is one of the simplest image resizing techniques, where the value of the new pixel is determined based on the exact value of the nearest pixel in the original image. For downsampling, this means that the new pixel takes the value of the nearest pixel from a corresponding group of pixels located in the original image, depending on the resampling ratio. However, this algorithm has certain drawbacks. Despite its low computational requirements and ease of implementation, it can lead to poor results and a decrease in image quality.

4.4.2 Bilinear Interpolation

Bilinear interpolation is a commonly used resizing technique that provides much smoother results than nearest-neighbor interpolation. This technique is still relatively easy to implement, as the algorithm utilizes four surrounding pixels to calculate the new pixel's value, which is a weighted average of those pixel values. The specific formula [13] for bilinear interpolation is given as:

$$f(x, y) = a + bx + cy + dxy$$

where:

$$a = f_{11} - bx_1 - cy_1 - dx_1y_1$$
$$b = \frac{f_{11} - f_{21}}{x_1 - x_2} - dy_1$$
$$c = \frac{f_{11} - f_{12}}{y_1 - y_2} - dx_1$$

$$d = \frac{f_{11} - f_{12} - f_{21} + f_{22}}{(x_1 - x_2)(y_1 - y_2)}$$

In this formula, f_{11}, f_{12}, f_{21}, and f_{22} represent the pixel values of the four surrounding pixels used in the interpolation process. f_{11} corresponds to the pixel value at the top-left position, f_{12} corresponds to the top-right, f_{21} corresponds to the bottom-left, and f_{22} corresponds to the bottom-right. These pixel values are used to calculate the new pixel value based on its position within the grid of surrounding pixels. This interpolation method helps create a smoother transition between neighboring pixels during the image resizing process.

5. Experimental Setup and Data Collection

The primary objective of this thesis is to develop an effective speech emotion recognition system and investigate the impact of various parameters on its performance. Further details will be discussed later. Before defining the scope of our study, several configurations and the working environment were established. Additionally, labeled data was collected, cleaned, and relevant libraries were imported. Subsequently, appropriate pre-processing techniques were applied to the data, enabling us to define the specific focus of our research.

5.1 Environment and Setup

First things first, we need to specify the software we are going to use. It has been decided that the model optimization will be done in Python, a high-level language that is easy to use and has a vast ecosystem of modules, making it ideal for handling deep learning tasks. The specific framework we used for deep learning was **PyTorch**. To install the necessary libraries, we utilized the Anaconda distribution. The code was executed in Python 3.10 [21], and the primary modules involved are listed in the table below.

Module	Functionality
`PyTorch`	Offers functionalities for training and optimizing machine learning models.
`Matplotlib`	Used for analyzing and visualizing data.
`Numpy`	Provides fundamental functionalities for numerical computing.
`OS`	Module used for interacting with the operating system.
`Pandas`	Allows data manipulation and analysis.
`Scipy`	Provides algorithms for multiple numerical operations and problems.
`Scikit-Learn`	Used for data pre-processing, modeling, and evaluation.
`Librosa`	Provides functions for audio processing.
`PIL`	Library for image processing.
`Math`	Popular library with a variety of mathematical functions.
`Random`	Grants the ability for random number generation and random sampling.
`Time`	Offers functionalities for measuring time and converting between time representations.

The training of a deep neural network is typically performed on powerful computers equipped with graph processing units (GPUs) to accelerate the computational time. To achieve satisfactory results within reasonable timelines, we utilized an **NVIDIA TESLA V100-PCIE-32GB GPU** for training and testing our models.

5.2 The Dataset for English SER

Our speech-emotion recognizers were trained and evaluated using data samples collected from four distinct English SER datasets. The classifiers were specifically trained to distinguish between the seven emotion classes in Ekman's model; $happy$, $angry$, $disgust$, $fearful$, sad, $neutral$ and $surprised$.

The first speech dataset we collected samples from, was the RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) dataset [14]. It comprises 1440 audio clips, featuring eight different emotions ($happy$, $angry$, $disgusted$, $fearful$, sad, $neutral$, $surprised$ and $calm$), spoken by a total of 24 actors, evenly divided between males and females. A total of 1248 speech samples were utilized, meaning we excluded the $calm$ label instances and focused on the first seven emotion classes.

The second dataset we employed was TESS (Toronto Emotional Speech Set) [5], a popular SER dataset consisting of 2800 audio clips, half of them generated by a 26-year-old actress and the other half generated by a 64-year-old actress. Each actress recorded 200 phrases per acted emotion. The targeted emotions in TESS are seven ($happy$, $angry$, $disgusted$, $fearful$, sad, $neutral$, $pleasantly\text{-}surprised$). However, 2 audio clips were found to be corrupted, resulting in the usage of 2798 samples. For our classification, $pleasant\text{-}surprise$ speeches were assumed to belong to the $surprise$ class.

We also obtained samples from the SAVEE (Surrey Audio-Visual Expressed Emotion)

dataset [19], the smallest among the four datasets. It consists of only 480 data samples, all spoken by 4 male actors, each vocalizing seven different emotional states ($happy$, $angry$, $disgusted$, $fearful$, sad, $neutral$, $surprised$) and was chosen to reduce the gender bias caused by female speakers in the TESS dataset.

Finally, we assembled a subset of the CREMA-D (Crowd-Sourced Emotional Multimodal Actors Dataset) [3]. The complete CREMA-D consists of 7442 audio clips generated by 91 actors, 48 male and 43 female between the ages of 20 and 74, coming from various races. Each emotion class, except for $neutral$, is produced in four different intensities; low, medium, high, and unspecified. To reduce the computational load, only the audios generated with emotions of the first three intensities (all except unspecified) were collected, along with 509 random neutral samples, all adding up to a total of 1874 instances. These samples represent six different classes ($happy$, $angry$, $disgusted$, $fearful$, sad, $neutral$).

Table 5.1: Emotion Distribution across the Total Dataset.

Emotion	**RAVDESS***	**TESS***	**SAVEE**	**CREMA-D***	**Total**
Happy	192	400	60	273	925
Angry	192	399	60	273	924
Disgusted	192	400	60	273	925
Fearful	192	399	60	273	924
Sad	192	400	60	273	925
Neutral	96	400	120	509	1125
Surprised	192	400	60	0	652
Total	1248	2798	480	1874	**6400**

It is evident that the neutral class comprises more samples than any other class in our dataset. In everyday life, neutral speech is more common than speech expressing specific emotions such as happiness or anger. As a result, neutral samples occupy approximately 17.58% of our final dataset, whereas the remaining classes account for up to 14.45% of the total audio samples. While the classifier demonstrates strong capability in recognizing neutral class speech, it is expected to have lower accuracy when classifying surprised speech due to the scarcity of training examples for that particular class. This class's imbalance can impact the model's performance and may lead to relatively lower accuracy in identifying surprise emotions compared to other emotions.

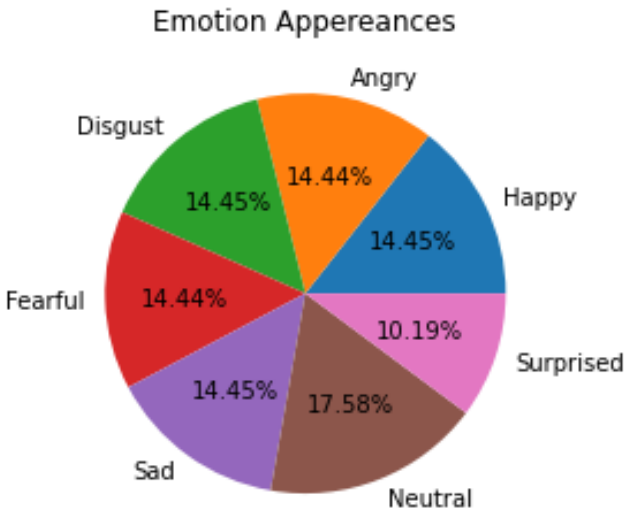


Figure 5.1: Appereances of every class in the Total Dataset.

It is important to note that a dataset can sometimes exhibit an unnatural distribution regarding the speaker's gender. Sadly, our SER classifier is likely to exhibit some bias towards the emotional context of female speakers, as approximately 67.2% of the total data samples are estimated to be from female speakers. Due to the random selection process of instances from the CREMA-D dataset, we can only provide an estimation of the percentage.

Furthermore, 1874 samples out of 7442 total samples were randomly selected, and 48 actors out of 91 in the dataset are male. Our estimation is that we have $\frac{48}{91} \times 1874 = 988$ male speaker utterances and $\frac{43}{91} \times 1874 = 886$ female speaker utterances.

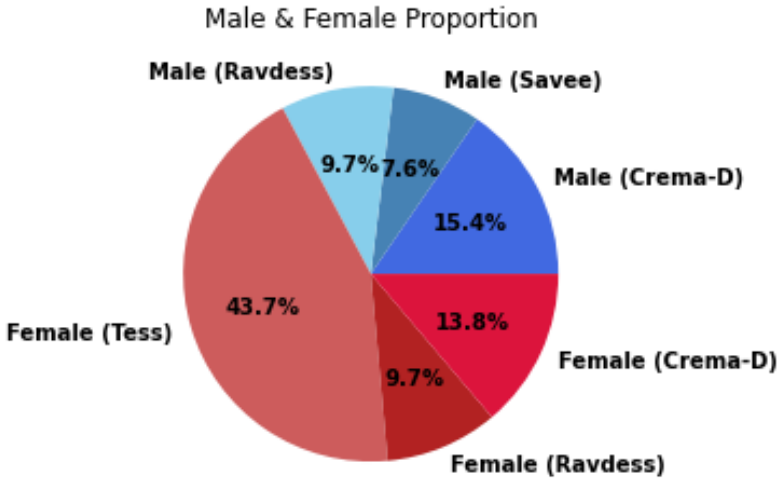


Figure 5.2: Gender proportion in the Total Dataset.

5.3 Data Pre-Processing

After we have gathered our SER data, the speech waveforms were appropriately manipulated before being fed to our CNN models. The pre-processing steps are listed below:

1. Pre-Emphasis
2. Trim Zeros (both in front and at the end of the signals)
3. Spectral Feature Extraction
4. Convert amplitude to dB scale
5. NxN Resize Transform

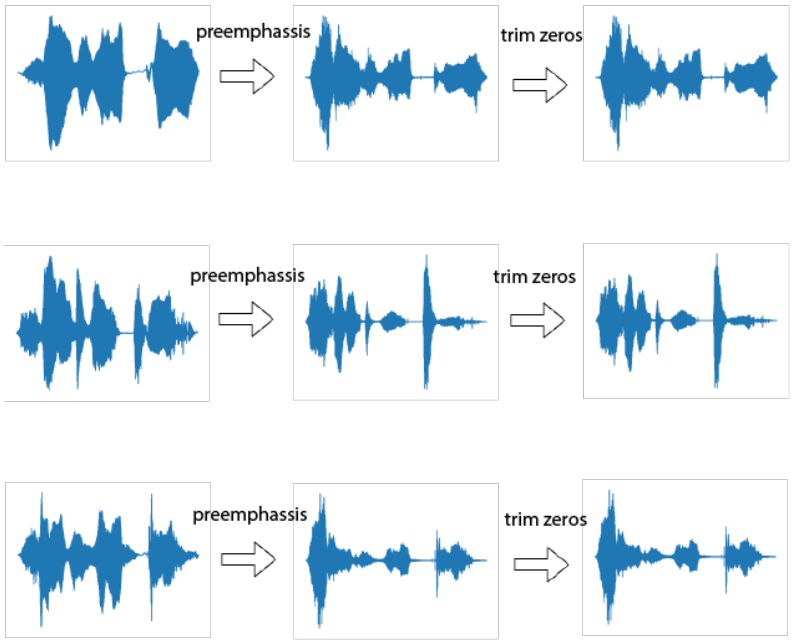


Figure 5.3: Pre-Processing Steps: Pre-Emphasis and Zero-Trimming.

The spectral features which have been extracted for classification were STFT chromagrams, mel spectrograms, and MFCCs. The necessary functions in Python were provided by Librosa [15]. When calling the corresponding functions for each feature extraction, it is important to set some important parameters. These parameters remained fixed throughout the entire process of this dissertation.

1. sr: The sampling rate of the waveform varies for each initial dataset:
   - RAVDESS: $sr = 48,000$ Hz
   - TESS: $sr = 24,414$ Hz
   - SAVEE: $sr = 44,100$ Hz

- CREMA-D: $sr = 16,000$ Hz

2. n_fft: This parameter determines the length of the FFT window and balances computational load and frequency resolution. Higher values provide better resolution but also increase computational complexity. We used $n\_fft = 2048$.

3. hop_length: This parameter represents the number of samples between successive frames in the STFT analysis. Smaller values increase time resolution but also increase computational complexity. We used $hop\_length = 512$.

4. N: This parameter represents the number of bins or coefficients used in different feature extraction methods. For example, when computing the MFCC, N represents the n_mfcc parameter, which is the number of MFCC coefficients computed. In other words, this parameter defines the length of the frequency axis in our image. The values we used for N were 30, 50, 60, and 80.

As for the resize transform, we used the bilinear image interpolation technique to convert our rectangular images into NxN square images. Moreover, the time-axis of the extracted features was rescaled to match the length of the frequency-axis, so the predetermined parameter N value we mentioned above, was also employed into this preprocessing step. The PIL module was employed for this preprocessing step.

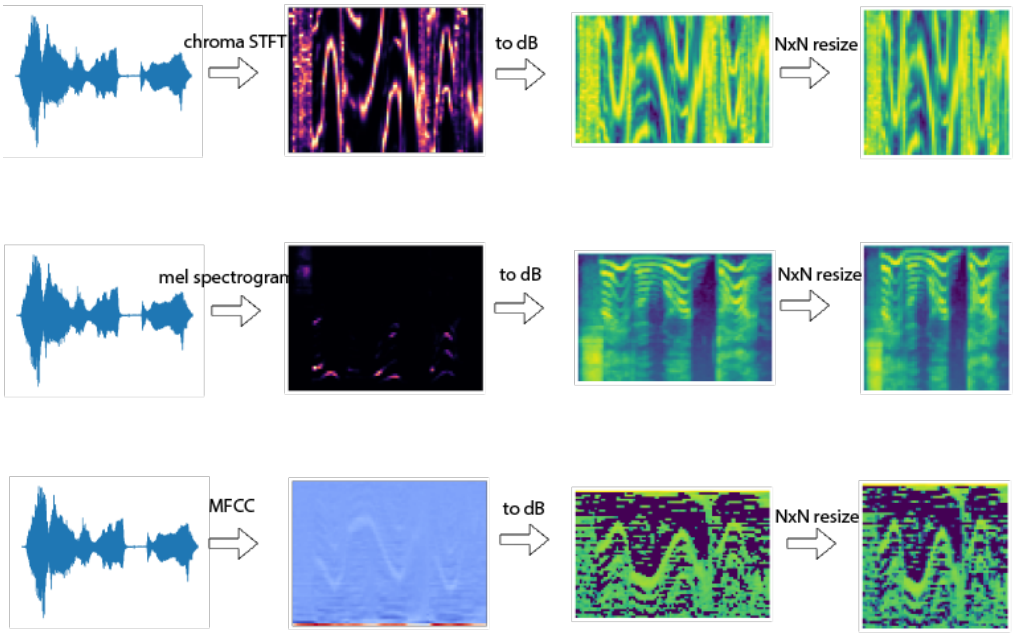


Figure 5.4: Pre-Processing Steps: Feature Extraction, Convert to dB and NxN Resize Transform.

Following the collection and preprocessing of the dataset, we proceeded to perform a random split. In this split, 80% of the data was allocated for training and cross-validation

purposes, while the remaining 20% was set aside as unseen instances for testing the final classifier and measuring its performance metrics. To validate our training models, we employed the 5-fold cross-validation technique.

To enhance the classifier's generalization ability, we applied additional audio processing to a subset of our validation data (a proportion of the data used in the K-fold cross-validation method's validation set). In particular, we selected the TESS dataset, known for its cleanliness and ease of classification for speech emotion recognition, and modified some of its original audio signals by either applying time-stretching, pitch-shifting, Gaussian noise addition, or a combination of those techniques before conducting any preprocessing steps. Three primary conditions needed to be fulfilled with this approach: first, maintaining a low computational complexity; second, ensuring that the TESS data would be clean when used for training and further processed only when used for validating; and third, ensuring that the folds remained separate and that no data was used for validation multiple times to prevent overfitting. To achieve the best approach, we started by duplicating our TESS data and modifying the duplicated dataset. Subsequently, we collected two distinct datasets: one consisting of clean data and the other incorporating the RAVDESS, SAVEE, CREMA-D, and the modified TESS. After preprocessing both datasets, we employed the same random splitting sequence, utilizing the same random seed to divide them into training and test sets. When creating the 5 folds, we used the first dataset for training and the second dataset for validation.

To implement these audio processing methods, certain parameters needed to be defined. For time-stretching, the stretch rate parameter was determined. For pitch-shifting, the shifting was specified in a semitone scale. For Gaussian noise addition, the signal-to-noise ratio (SNR) in decibels (dB) was defined, while the length of the noise signal was set to be the same as the length of each speech segment, and the mean of the Gaussian noise was set to 0. Each sample was assigned random parameter values selected from specific grids. The grids of values used are listed below:

- Stretch rate values: $0.75, 0.80, 0.85, 0.90, 1.05, 1.10, 1.20, 1.25$
- Semitone values: $-2, -1, +1, +2$
- SNR(dB) values: $7, 8, 11, 12, 14, 15, 17, 18, 20$

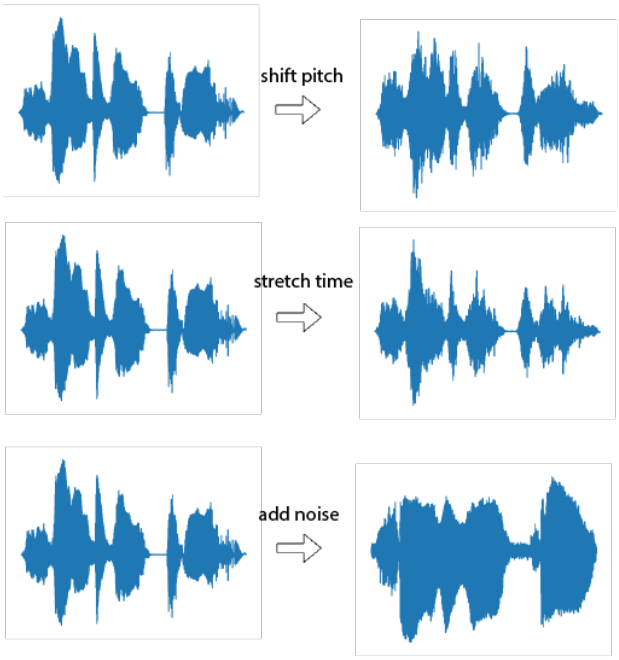


Figure 5.5: Data Manipulation Techniques applied in the TESS dataset.

Nevertheless, since the modified signals that were included in the test set after the random split were never used for validation, it is evident that this approach resulted in some computational waste.

1. Create a copy of TESS.
2. Apply the audio manipulation techniques to the replica's speech signals.
3. Perform all pre-processing steps for each dataset.
4. Merge the spectral feature images extracted from the original TESS with those of the remaining datasets (RAVDESS, SAVEE, CREMA-D). Repeat for those extracted from the manipulated TESS with a copy of the images of the remaining datasets. For simplicity we will refer to the first merged data set as "clean" and the second as "manipulated".
5. Call two consecutive random split functions for the training and test sets, one to split the clean dataset and one to split the manipulated one (we will not need the manipulated test set). Before calling the functions, we set a certain random seed (function of the random library) so that the two datasets are separated by an exact match and we don't risk overfitting.
6. To split the training set in cross-validation, repeat the above step and for each validation subset used to evaluate one of the K models, use the manipulated validation dataset, instead.

7. Later in testing and evaluating the final model, we'll use the clean test set that we've kept and haven't utilized yet.

5.4 CNN Architectures

We will now present the CNN architectures used for training and optimization. This study presents three custom, abstract convolutional neural network architectures for 2D acoustic feature speech emotion classification tasks, which are inspired by the popular Pokémon characters: Bulbasaur, Charmander, and Squirtle. We refer to these networks as BulbaNET, CharmaNET, and SquirtleNET, respectively. Each network has three convolutional layers, three fully-connected layers and a variety of different pooling layers, activation functions and regularization layers that can be seen in 5.6. Several of their internal architecture parameters were also used for tuning and optimization.

By training a CNN with labeled 2D acoustic features, we should choose the one which returns the best classification performance. BulbaNET employs a simpler architecture with a set of average pooling layers and ReLu as the only activation function. CharmaNET utilizes the Leaky ReLu activation function after each convolutional layer, along with Batch Normalization and Dropout layers for regularization. SquirtleNET is similar to CharmaNET with the exception that it uses ReLu as the main activation function after each convolutional layer and applies average pooling instead of max pooling after the second convolutional layer.

Figure 5.6: CNN Architectures: BulbaNET, CharmaNET and SquirtleNET.

6. Experimental Analysis

6.1 Phase 1: Exploring CNN Architectures and Input Spatial Features for English SER

To find the classifier with the highest performance in the collected dataset, we will tune a vast number of parameters. We are categorizing these parameters into primary and secondary. The primary parameters include the 2D spectral feature exported, the axial size of the corresponding image after resizing and the CNN architecture used. On the other hand, the secondary parameters are more specific and can adopt a wide range of possible values. These parameters require a searching/tuning method to be explored. Below are listed both the primary and secondary parameters used for tuning, as well as those which were set to a fixed default value for training and validation. It is worth noting that the Negative slope in Leaky ReLU was initially set to a default value, but this value was changed to certain values for certain experiments when necessary.

The **primary parameters** used for optimization included the three following:

- 2D acoustic spectral feature.
- Image size after the interpolation method.
- CNN architecture.

The **secondary parameters** used for optimization included all of the following:

CNN structural parameters:

- Output channels of each convolutional layer.
- Output features of the first two dense layers.
- Kernel size of each convolutional layer.
- Kernel size of each pooling layer.
- Stride in each pooling layer.
- Probability for an element to be zeroed in the dropout layer.

- Momentum in the batch normalization layer (controls the running mean and running variance).

Hyperparameters:

- Batch size.
- Learning rate.

Default Parameters:

Additionally, certain parameter values were fixed as default:

- Stride in convolutional layers: 1.
- Stride in SquirtleNET's pooling layers: 2.
- Padding in convolutional layers: 1.
- Negative slope in CharmaNET's Leaky ReLU: 0.010 (Partially Tuned).
- Pre-Processing parameters.

The primary parameters will be explored using a small grid of values. Specifically, we will consider the following spectral features: **Mel Spectrograms, MFCC,** or **STFT Chromagrams**, extracted using the corresponding Librosa function. Regarding image dimensionality, we will consider sizes of **30x30, 50x50, 60x60** and **80x80**. The goal is to determine which image size works best for classification. Additionally, the CNN architecture will be chosen from among **BulbaNET, CharmaNET,** or **SquirtleNET**.

To tune all the parameters, we followed a series of steps that constitute the **general methodology**. Let's take a look at these steps before moving on.

1. Define the spectral feature and the image size.
2. Execute all of the Pre-Processing steps.
3. Split the data into training and testing sets (80%-20%). Use only the training set for tuning.
4. Specify the grid of secondary parameters and the searching method.
5. Define the Early-Stopping algorithm conditions (if any).
6. For every secondary parameter vector, use 5-fold cross-validation and compute the average evaluation metrics for all folds. We computed the following metrics for comparison:
    - Train loss
    - Test loss
    - Accuracy

- Macro F1 score

7. Analyze learning curves to better understand the problem and identify potential overfitting (only if necessary).
8. Select the parameter combinations with the highest performance based on the average metrics computed in the cross-validation process.

For training the models, we utilized the `CrossEntropyLoss()` loss function from the `torch.nn` module, and during training, we employed the `Adam optimizer without any learning rate decay`. We also have constructed an early-stopping algorithm to minimize computational requirements, which will be thoroughly discussed in this chapter. It should be noted than the algorithm becomes more strict as we progress and get more satisfying results.

The general methodology described above was implemented in four different parameter qualification stages, during which certain steps were occasionally modified. The objective of these four stages was to explore various sets of parameters, including different CNN architectures and spectral features, and retain those that yielded the best performance. We examined the various parameters by employing the random search algorithm across different grids, as outlined below:

1. Conduct an initial secondary-parameter random search for every specified combination of primary parameters. For the secondary parameters, the same vectors are used for the first ten iterations of every primary parameter combination. The total random search iterations completed on this stage are displayed in Table 6.1. An additional random search was performed for two new Leaky ReLu Slopes in CharmaNET, for 50x50 Mel Spectrograms and STFT Chromagrams. **(First Parameter Qualification Stage)**
2. Perform a second round of random search using only the most promising primary parameters. Select the combinations that yield the highest performance and merge them with the previously best combinations. **(Second Parameter Qualification Stage)**
3. Select the best three parameter vectors from the previous list and conduct another round of random search with slight modifications to the existing optimal secondary parameters. Once again, add the combinations that demonstrate the highest performance into the list. **(Third Parameter Qualification Stage)**
4. Select the best models from the list and train them for a greater number of epochs without applying any Early-Stopping condition. Then, identify the optimal parameter vector. **(Fourth Parameter Qualification Stage)**

The Early-Stopping Algorithm

Instead of training each model for a specific number of epochs, we chose to define a set of early stopping conditions based on the models' behaviors. For this, we performed an extra

step. We trained several models, each containing a set of different random parameters, for a high number of epochs and carefully observed their convergence patterns and performance trends. We monitored the train loss - test loss curve as well as the accuracy and macro F1 score values per training epoch. A

After observing the overall behavior, we were able to gain a brief insight on what kind of early stopping conditions we should apply to training. We set the maximum number of training epochs equal to 300, and two sets of conditions were defined, along with a third overall condition.

Class A early stopping conditions were designed to immediately stop the cross-validation process if a model's classification ability was considered poor. Moreover, if the Macro F1 score fell below a predefined threshold in specific epochs, the training and validation processes were abandoned. In such cases, a new set of random parameters was selected for the successive iterations. This class of conditions allowed us to identify and discard poorly performing models fast, ensuring efficiency in the optimization process.

Class B early stopping conditions were responsible for detecting overfitting and preventing further training of a well-performing model, also reducing the computational cost. For each predefined step of epochs, we examined a set of consecutive test losses and Macro F1 scores. If the values consistently increased for losses or decreased for F1 scores, it indicated the emergence of overfitting. In such cases, we stopped training to avoid the model becoming inordinately specialized to the training data. The maximum number of epochs for the next cross-validation folds, was set based on the point at which convergence began to emerge. Most importantly, this saved us a significant amount of computational time and resources.

In addition to the class A and class B conditions, we also incorporated an overall early stopping condition, referred to as the C condition. This condition aimed to terminate the validation process if either the Accuracy or Macro F1 score values, evaluated after an entire training fold, fell below a predefined threshold. The purpose of this condition was to address scenarios where models might have survived the class A conditions, indicating a certain level of classification ability, but still exhibit very poor overall performance.

The initially defined early stopping (E.S.) conditions were defined below:

Class A:

- A1: Stop if epoch equals 20 and macro F1 score (f1) is less than 17%.
- A2: Stop if epoch equals 50 and macro F1 score (f1) is less than 20%.
- A3: Stop if epoch equals 75 and macro F1 score (f1) is less than 24%.
- A4: Stop if epoch equals 100 or epoch equals 125 and macro F1 score (f1) is less than 27%.
- A5: Stop if epoch equals 150 and macro F1 score (f1) is less than 36%.

- A6: Stop if epoch equals 200 and macro F1 score (f1) is less than 39%.
- A7: Stop if epoch equals 250 and macro F1 score (f1) is less than 42%.

Class B:

- B1: Stop if epoch is greater than 50, divisible by 4, and conditions k1, k2, k3, and k4 are all true, where:
  - k1: Test loss at epoch is greater than test loss at epoch-4.
  - k2: Test loss at epoch-4 is greater than test loss at epoch-8.
  - k3: Test loss at epoch-8 is greater than test loss at epoch-12.
  - k4: Test loss at epoch-12 is greater than test loss at epoch-16.
- B2: Stop if epoch is greater than 50, divisible by 4, and values v1, v2, v3, and v4 are all less than 1.0%, where:
  - v1: Difference between macro F1 score at epoch and macro F1 score at epoch-4.
  - v2: Difference between macro F1 score at epoch-4 and macro F1 score at epoch-8.
  - v3: Difference between macro F1 score at epoch-8 and macro F1 score at epoch-12.
  - v4: Difference between macro F1 score at epoch-12 and macro F1 score at epoch-16.
- B3: Stop if epoch is greater than 110, divisible by 40, and the difference between macro F1 score at epoch and macro F1 score at epoch-40 is less than 1.2%.

Class C:

- C1: Stop if fold's macro F1 score is less than 45% and fold's accuracy is less than 42%.
- C2: Stop if fold's macro F1 score is less than 50% and fold's accuracy is less than 50%.
- C3: Stop if fold's macro F1 score is less than 60% and fold's accuracy is less than 60%.

The graph in Figure [Class A and C conditions curve] shows an estimate of the curve that largely defines the initial macro F1 score total threshold, based on the conditions above. For the trained model to be considered a good fit, the cross-validation Macro F1-score values per epoch should be located above this curve.

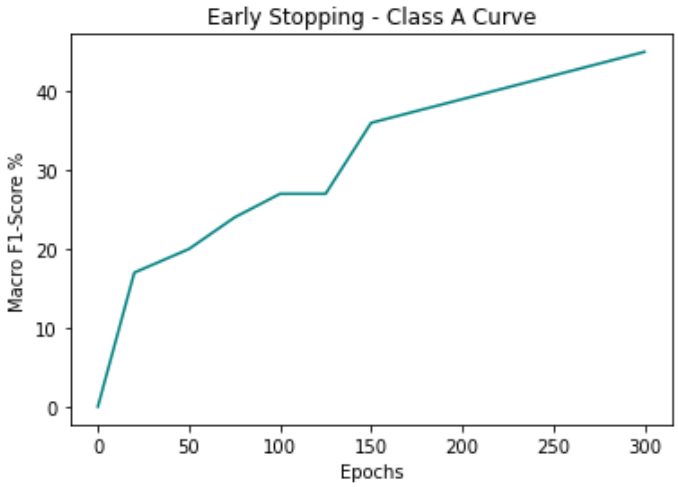


Figure 6.1: Approximation of Macro F1 Score per epoch: Initial Class A and C conditions.

It is important to highlight that throughout the process, only one of the C conditions was selected at a time. Initially, C1 was chosen as we gradually increased the thresholds while obtaining favorable results. During our optimal parameter searching process, we made changes to some of the conditions' thresholds as necessary. After comparing the performance of 30x30 and 50x50 chromagrams, 30x30 and 50x50 mel spectrograms, and 30x30, 50x50, and 60x60 MFCC, we decided to modify the Accuracy and Macro F1 score thresholds for the C1 condition. Initially, we set the thresholds at 50% since we already had classifiers surpassing the 50% Accuracy and Macro F1 score mark (C2). However, as we obtained even better models, we revised the thresholds to 60% (C3). As a result, we ensured that we focused our training efforts only on elite models.

In addition, we observed that in the first fold, all models that were not stopped by a Class A condition were eventually stopped by a Class B condition, meaning that no model reached the maximum of 300 epochs during training. The vast majority of models that completed training for all 5 folds and thereafter were solely stopped by class B conditions (making them suitable for comparison) required between 52 to less than 80 epochs of training. We can assume that this is also the average range of epochs until learning convergence starts to emerge. None of the models required more than 120 epochs of training in the entire 5-fold cross validation process.

1st Parameter Qualification Stage

Below are the corresponding values used for the initial random search tuning of the First Qualification Stage:

Convolutional Output Channels: 100, 116, 132, 148, 164, 180, 196, 212, 228, 244, 260, 276, 292, 308, 324, 340, 356, 372, 388, 404, 420

Convolutional Kernel Sizes: 2, 3

Pooling Kernel Sizes: 2, 3

Pooling Strides: 2, 3

Batch Normalization Momentum: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

Dropout Probability: 0.25, 0.30, 0.35, 0.40, 0.45, 0.50

Fully Connected Output Channels: 1024, 1152, 1280, 1408, 1536, 1664, 1792, 1920, 2048, 2176, 2304, 2432, 2560, 2688, 2816, 2944, 3072, 3200, 3328, 3456, 3584, 3712, 3840, 3968, 4096, 4224

Batch Size: 16, 32, 48, 64

Learning Rate: 2e-8, 6e-8, 1e-7, 5e-7, 1e-6, 3e-6, 6e-6, 9e-6, 2e-5, 4e-5, 6e-5, 8e-5, 1e-4, 3e-4, 7e-4

The first notable observation is the difficulty in making accurate predictions with MFCC images. One of the contributing factors to this challenge is likely the significant changes that occur in MFCCs when techniques such as pitch shifting or time stretching are applied to the data. Considering that 43.7% of the total dataset belongs to the TESS data, we understand that it was challenging for the classifier to achieve high accuracy and macro F1 scores, as the cross-validation data included numerous heavily altered MFCC images. These alterations likely had an unfavorable effect on the overall performance.

Images of a 30x30 size were not favorable either. The classification performance was lower compared to larger-sized images, which demonstrated slightly better performance. Chromagrams and mel spectrograms of sizes 50x50, 60x60, and 80x80 produced promising results that merit further investigation and study.

When it comes to the network architectures used in the study, both CharmaNET and SquirtleNET exhibited exceptional performance in comparison to BulbaNET. While BulbaNET did produce satisfactory results, it was found to be less suitable for the specific task of speech emotion recognition. The superior performance displayed by CharmaNET and SquirtleNET suggests that these architectures are better suited for accurately classifying and recognizing emotions in speech. Additional investigation and analysis of these architectures would provide valuable insights into their respective strengths and contributions in the field.

In addition, an **extra step** was implemented here as we performed a random parameter search in which we trained the CharmaNET models using two different negative slope values in the Leaky ReLU activation functions, equal to 0.005 and 0.050, for 50x50 chromagram and mel spectrogram inputs. Here, the negative slope is denoted as a.

Table 6.1: First Qualification Stage: Random Iterations per parameter vector.

CNN	Feature	Width x Height	Random Iterations
BulbaNET	Chromagram	30x30	25
BulbaNET	Chromagram	50x50	25
BulbaNET	Chromagram	60x60	10
BulbaNET	Chromagram	80x80	10
BulbaNET	Mel Spectrogram	30x30	10
BulbaNET	Mel Spectrogram	50x50	10
BulbaNET	Mel Spectrogram	60x60	10
BulbaNET	Mel Spectrogram	80x80	10
BulbaNET	MFCC	30x30	10
BulbaNET	MFCC	50x50	10
BulbaNET	MFCC	60x60	10
BulbaNET	MFCC	80x80	10
CharmaNET	Chromagram	30x30	10
CharmaNET	Chromagram	50x50	10
CharmaNET	Chromagram	60x60	10
CharmaNET	Chromagram	80x80	10
CharmaNET	Mel Spectrogram	30x30	10
CharmaNET	Mel Spectrogram	50x50	10
CharmaNET	Mel Spectrogram	60x60	10
CharmaNET	Mel Spectrogram	80x80	10
CharmaNET	MFCC	30x30	10
CharmaNET	MFCC	50x50	12
CharmaNET	MFCC	60x60	10
CharmaNET	MFCC	80x80	10
SquirtleNET	Chromagram	30x30	10
SquirtleNET	Chromagram	50x50	10
SquirtleNET	Chromagram	60x60	10
SquirtleNET	Chromagram	80x80	10
SquirtleNET	Mel Spectrogram	30x30	10
SquirtleNET	Mel Spectrogram	50x50	10
SquirtleNET	Mel Spectrogram	60x60	10
SquirtleNET	Mel Spectrogram	80x80	10
SquirtleNET	MFCC	30x30	10
SquirtleNET	MFCC	50x50	10
SquirtleNET	MFCC	60x60	10
SquirtleNET	MFCC	80x80	10

Table 6.2: Additional Random Search Iterations for new Leaky ReLu Slopes in CharmaNET.

CNN	Feature	Width x Height	Random Iterations
CharmaNET ($a = 0.005$)	Chromagram	50x50	10
CharmaNET ($a = 0.005$)	Mel Spectrogram	50x50	10
CharmaNET ($a = 0.050$)	Chromagram	50x50	10
CharmaNET ($a = 0.050$)	Mel Spectrogram	50x50	10

The parameter vectors which achieved Macro F1-Score and Accuracy more than 60% are displayed in Table 6.3. There are included all the primary and secondary parameter values that lead to the best SER performance, along with the number of epochs required before the early-stopping algorithm was activated and the corresponding execution time.

- **CNN**: Convolutional Neural Network Architecture
- **Feat**: Spectral Feature
- **NxN**: Width x Height of image
- **COC**: Convolutional Output Channels
- **CKS**: Convolutional Kernel Sizes
- **PKS**: Pooling Kernel Sizes
- **PS**: Pooling Stride
- **BNM**: Batch Normalization Momentum in the Batch-Normalization Layer
- **DP**: Dropout Probability in the Dropout Layer
- **FCOC**: Fully-Connected Output Channels
- **BS**: Batch Size
- **LR**: Learning Rate
- **ET**: Execution Time
- **EP**: Number of Epochs Needed due to Early-Stopping

Table 6.3: Highest-Performance Parameter Vectors (Columns) for First Qualification Stage.

CNN	Squirtle NET	Squirtle NET	Charma NET (a= 0.005)	Squirtle NET	Squirtle NET	Squirtle NET	Squirtle NET	Charma NET	Charma NET	Squirtle NET	Squirtle NET
Feat	Chroma	Mel Spec	Chroma	Chroma	Chroma	Mel Spec	Mel Spec	Chroma	Mel Spec	Mel Spec	Mel Spec
NxN	50x50	50x50	50x50	80x80	80x80	80x80	60x60	60x60	80x80	60x60	60x60
COC	276, 100, 244	100, 164, 244	324, 180, 420	276, 148, 420	244, 132, 116	276, 148, 420	388, 116, 116	260, 212, 356	388, 116, 116	404, 356, 372	212, 148, 356
CKS	3, 3, 2	3, 3, 3	3, 2, 3	2, 2, 3	2, 3, 2	2, 2, 3	3, 2, 2	2, 3, 3	3, 2, 2	3, 2, 3	3, 3, 3
PKS	2, 2, 2	2, 3, 3	3, 3, 3	3, 3, 3	3, 3, 2	3, 3, 3	2, 3, 2	2, 2, 3	2, 3, 2	2, 3, 2	2, 3, 3
PS	2, 2, 2	2, 2, 2	2, 3, 2	2, 2, 2	2, 2, 2	2, 2, 2	2, 2, 2	2, 2, 2	2, 2, 2	2, 2, 2	2, 2, 2
BNM	0.6	0.3	0.3	0.5	0.9	0.5	0.5	0.9	0.1	0.4	0.3
DP	0.45	0.25	0.40	0.45	0.50	0.45	0.25	0.25	0.30	0.25	0.45
FCOC	1536, 3072, 7	1792, 2432, 7	1536, 1024, 7	4224, 3584, 7	3456, 3200, 7	4224, 3584, 7	1664, 2688, 7	2048, 2304, 7	1664, 2688, 7	1792, 1280, 7	3840, 2944, 7
BS	32	48	64	48	32	48	16	16	16	48	32
LR	8e-05	8e-05	3e-04	4e-05	8e-05	4e-05	2e-05	6e-05	2e-05	8e-05	6e-05
ET (min)	12.01	9.02	10.35	47.15	26.04	52.08	28.36	26.94	22.17	38.45	31.14
EP	52	64	56	56	52	76	84	52	52	64	60
Macro F1-Score %	60.09	60.11	60.22	61.13	61.36	61.48	61.71	61.82	62.43	63.91	66.42
Accuracy %	60.33	60.25	60.86	61.82	61.70	62.81	62.68	62.46	63.07	64.39	66.89

Now that we have the list with the optimal parameter vectors, we eliminated certain trained models to identify the top performers. We focused on lighter structures and discarded the

two models with the longest computational times, which were 52.08 and 47.15 minutes, respectively. However, after careful consideration, it was decided that only the three models that exhibited the best Macro F1 scores will be chosen.

2nd Parameter Qualification Stage

Next, additional random search iterations were conducted for parameters that displayed promising results. We performed 10 new random iterations for each CNN architecture using 60x60 mel spectrograms, as well as 10 iterations for each CNN architecture using 60x60 chromagrams. However, we modified the early stopping conditions to be more strict in order to save more time and prioritize achieving superior classification performance. This approach allowed us to explore further variations and evaluate the performance of even more parameter combinations within a parameter grid that was more likely to concede positive results.

The early stopping algorithm critical quantities were now reset as follows:

- A1: epoch==20 and macro F1<42%
- A2: epoch==50 and macro F1<45%
- A3: epoch==75 and macro F1<48%
- A4: epoch==100 or epoch==125 and macro F1<51%
- A5: epoch==150 and macro F1<54%
- A6: epoch==200 and macro F1<56%
- A7: epoch==250 and macro F1<58%
- C3: f1<60% and acc<60%

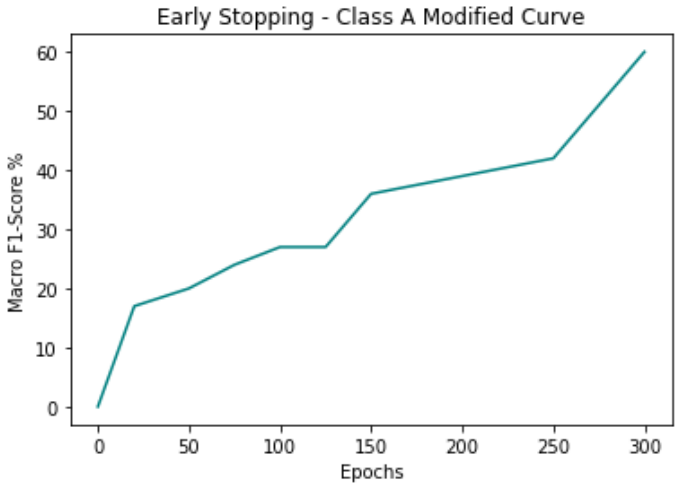


Figure 6.2: Approximation of Macro F1 Score per epoch: Modified Class A and C conditions.

Nonetheless, despite our efforts, we were unable to identify new parameter vectors that outperformed the top three options which are already included on our list.

3rd Parameter Qualification Stage

In this third stage of the qualification process, an additional random search within grid areas centered around the parameter combinations of the top three models was included. The objective was to explore and identify potential improvements by zooming in on more optimal regions within the parameter space.

However, to ensure that the final model allocates minimal memory storage and to reduce the dimensionality of the parameter plane, certain constraints were established. Moreover, the following constraints were set:

- Avoid significant increases in Convolutional Layer Output Channels or Fully-Connected Layer Output Channels. This restriction aimed to minimize memory allocation in the final model.
- Maintain consistent values for Convolutional Kernel Sizes, Pooling Kernel Sizes, Pool Strides, Batch Sizes, and Learning Rates. By keeping these parameters unchanged, we focused on optimizing a specific subplane of parameters rather than exploring the entire parameter space.

To complete the explanation of this step, we shall report that 10 random iterations in the neighboring grids of each of the three optimal parameter vectors were performed. The new parameter grid is based on the following:

1. Convolutional Output Channels: Select a random value with uniform distribution from an array with the following values to add to the existing number of channels: -32, -24, -16, -8, 0, 0, +8, +16, +24, and +32.
2. Convolutional Kernel Sizes: No change.
3. Pooling Kernel Sizes: No change.
4. Pooling Strides: No change.
5. Batch Normalization Momentum: If momentum equals 0.1, randomly choose either to keep it as 0.1 or to double it to 0.2; otherwise, select a random value with uniform distribution from an array with the following values to add to the existing momentum: -0.1, 0, 0, 0, 0.1.
6. Dropout Probability: Select a random value with uniform distribution from an array with the following values to add to the existing probability value in the dropout layer: -0.10, -0.05, -0.05, 0, 0, 0, 0, +0.05, +0.05, +0.10.
7. Fully-Connected Output Channels: Select a random value with uniform distribution from an array with the following values to add to the existing number of output channels in the dense layers: -230, -206, -160, -128, -86, -62, -34, -18, 0, 0, +18, +34, +62, +86.

8. Batch Size: No change.
9. Learning Rate: No change.
10. LeakyReLU Slope: Select randomly one of the following values with uniform distribution in the random selection, to use as the LeakyReLU's negative slope: 0.005, 0.008, 0.010, 0.013, 0.018, 0.028.

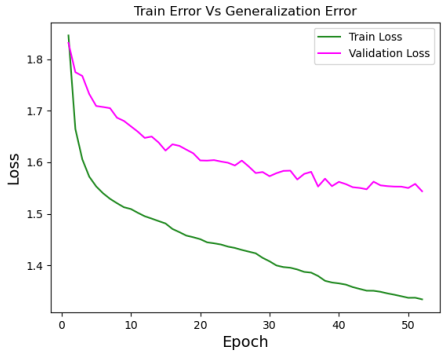
After conducting this parameter searching process, we collected new combinations and more trained models that achieved high accuracy performances. The table 6.4 below provides these newfound combinations along with their performance metrics.

It is evident in Table 6.4 that all of the new qualifying models are variations of the (111) model and were discovered within the grid centered around the (111) parameter vector. Conducting a more extensive search within this specific region could be a worthwhile endeavor. Searching this area in-depth can involve exploring a wider range of parameter combinations or employing more sophisticated optimization techniques. By delving deeper into this parameter region, there is a higher likelihood of discovering more models with improved Accuracy and Macro F1 scores.

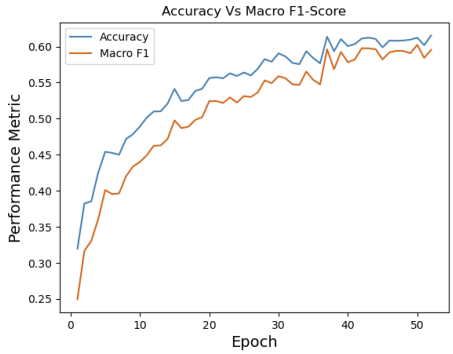
Table 6.4: Highest-Performance Parameter Vectors after Third Parameter Qualification Stage.

ID	000	001	010	011	100	101	110	111
CNN	Charma Net	Squirtle NET	Squirtle NET	Squirtle NET	Squirtle NET	Squirtle NET	Squirtle NET	Squirtle NET
Feat	Mel Spec	Mel Spec	Mel Spec	Mel Spec	Mel Spec	Mel Spec	Mel Spec	Mel Spec
NxN	80x80	60x60	60x60	60x60	60x60	60x60	60x60	60x60
COC	388, 116 116	220 140 356	404 356 372	212 148 356	236 124 356	204 124 356	188 156 340	212 148 356
CKS	3 2 2	3 3 3	3 2 3	3 3 3	3 3 3	3 3 3	3 3 3	3 3 3
PKS	2 3 2	2 3 3	2 3 2	2 3 3	2 3 3	2 3 3	2 3 3	2 3 3
PS	2 2 2	2 2 2	2 2 2	2 2 2	2 2 2	2 2 2	2 2 2	2 2 2
BNM	0.1	0.3	0.4	0.2	0.3	0.4	0.4	0.3
DP	0.30	0.45	0.25	0.45	0.50	0.45	0.40	0.45
FCOC	1664 2688 7	3610 2910 7	1792 1280 7	3840 2858 7	3610 2816 7	3926 2858 7	3634 2816 7	3840 2944 7
BS	16	32	48	32	32	32	32	32
LR	2e-05	6e-05	8e-05	6e-05	6e-05	6e-05	6e-05	6e-05
ET(min)	22.17	13.67	38.45	29.88	21.40	18.07	17.78	31.14
EP	52	52	64	96	76	64	64	60
Macro F1-Score %	62.43	63.20	63.91	64.03	64.60	65.27	65.82	66.42
Accuracy %	63.07	63.61	64.39	64.80	64.92	65.74	65.29	66.89

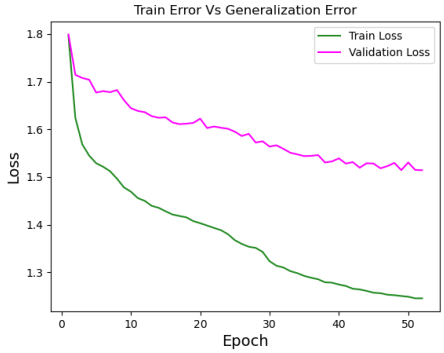
An ID tag was assigned to each parameter vector and displayed in the first row in order for us to easily be able to refer to them. We finalized our searching process, not just by comparing the Macro F1 scores and Accuracy values, but by also analyzing the learning curves through two types of graphs: the test loss - train loss curves per epoch and the Macro F1 score - Accuracy per epoch. In other words we have conducted a curve diagnosis in order to qualify the best results. All of the graphs are being displayed in figures 6.3 and 6.4.



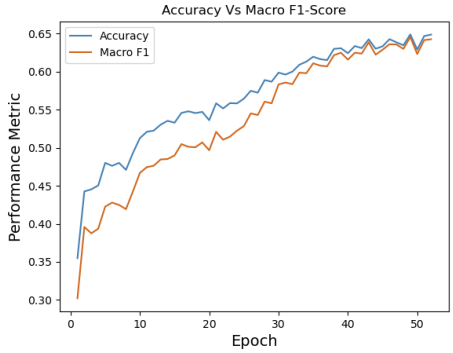
ID: 000
Train/Test Losses



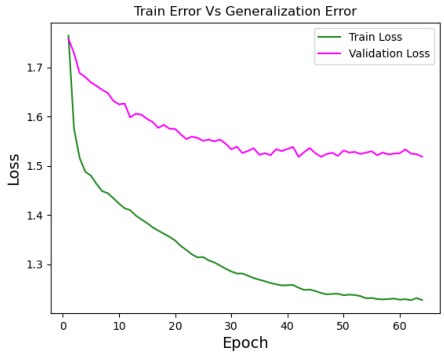
ID: 000
Accuracy/Macro F1



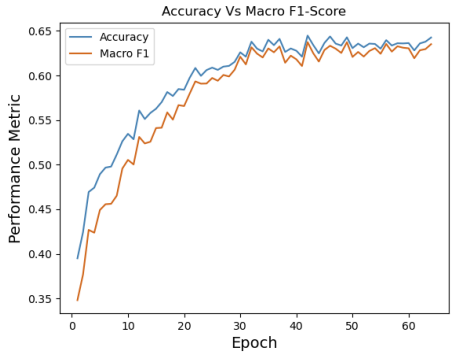
ID: 001
Train/Test Losses



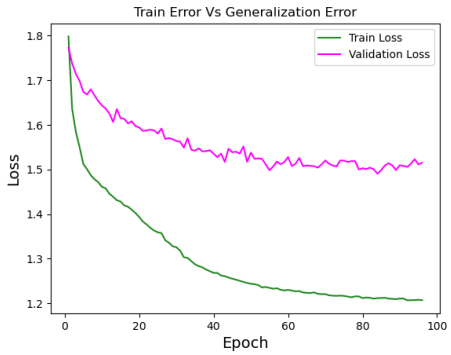
ID: 001
Accuracy/Macro F1



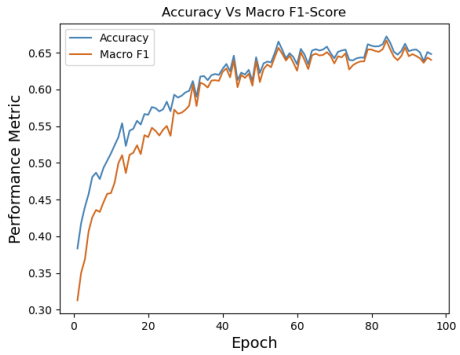
ID: 010
Train/Test Losses



ID: 010
Accuracy/Macro F1

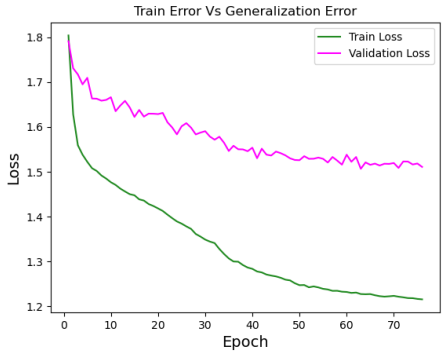


ID: 011
Train/Test Losses

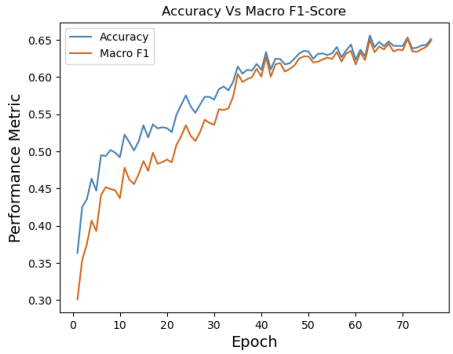


ID: 011
Accuracy/Macro F1

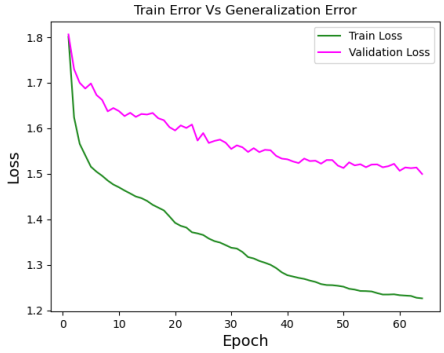
Figure 6.3: Learning Curves of the Qualifying models: ID:000 to ID:011



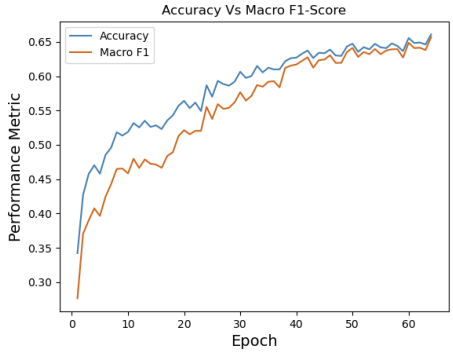
ID: 100
Train/Test Losses



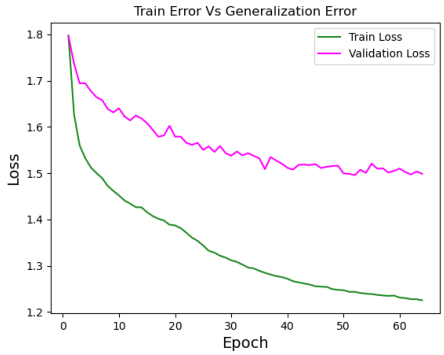
ID: 100
Accuracy/Macro F1



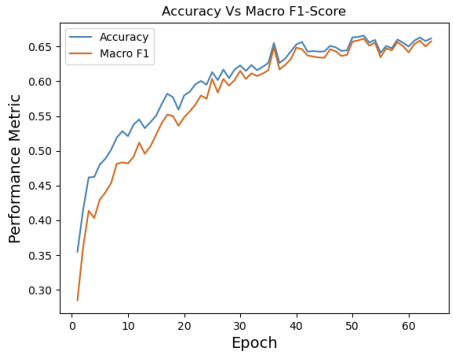
ID: 101
Train/Test Losses



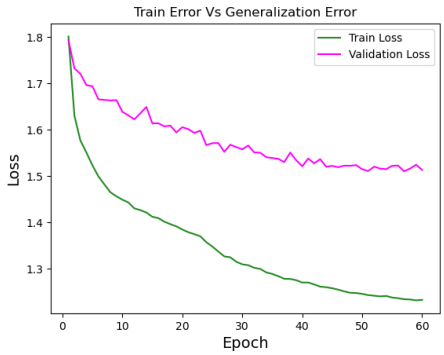
ID: 101
Accuracy/Macro F1



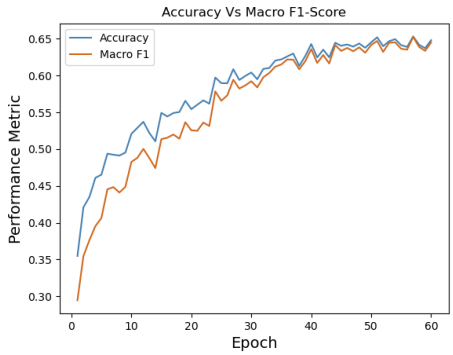
ID: 110
Train/Test Losses



ID: 110
Accuracy/Macro F1



ID: 111
Train/Test Losses



ID: 111
Accuracy/Macro F1

Figure 6.4: Learning Curves of the Qualifying models: ID:100 to ID:111

4th Parameter Qualification Stage

While not much can be inferred from the loss curves, it appears that (001) has not converged to the optimal value yet, indicating that the model still has potential to learn. This lack of convergence is more likely due to a misleading early stopping activated condition. Considering that this model would require additional training to reach its true performance level, we have chosen not to exclude it. In addition, models (101), (110), and (111) have also demonstrated Macro F1 scores and Accuracies above 65%, all in convergence. As a result, after eliminating the rest of the parameter vectors, the updated list of optimal models are (001), (101), (110) and (111).

An important point to note is that convergence in learning does not necessarily indicate true overfitting. In our case, convergence simply signifies a reduction in the rate at which the model is learning and the area where the model learns less and harder. Overfitting occurs when the performance of the model starts to decline significantly. It is worth mentioning that the early stopping algorithm has performed admirably thus far. However, since the number of epochs required for convergence appears to be similar for all qualifying models, we can extend the training by a few more epochs to measure the progress and learning capacity of these models.

In addition, considering that (001) has not yet converged, it presents a great opportunity to conduct this comparative test. We can train all the qualifying models for 150 epochs without implementing an early stopping condition and then compare their accuracy and macro F1 scores to determine which model performs best. Training for 150 epochs is reasonable since, assuming a linear relationship between execution time and the total number of epochs, the estimated training and validation time for each model would typically range between 40 minutes to 80 minutes. Spending more than this time range for this deep-learning model validation would indeed require a significant investment in time. Therefore, it is important to consider the trade-off between the potential gains in performance and the additional time required for validation.

Hence, 6.5 shows the peformance results for training the qualified models for 150 epochs.

Table 6.5: Best Models trained for 150 Epochs.

ID (Model)	Epochs needed	Macro F1-Score (%)	Accuracy (%)
001	150	63.14	64.02
101	150	65.16	65.53
110	150	65.35	66.05
111	150	64.21	65.00

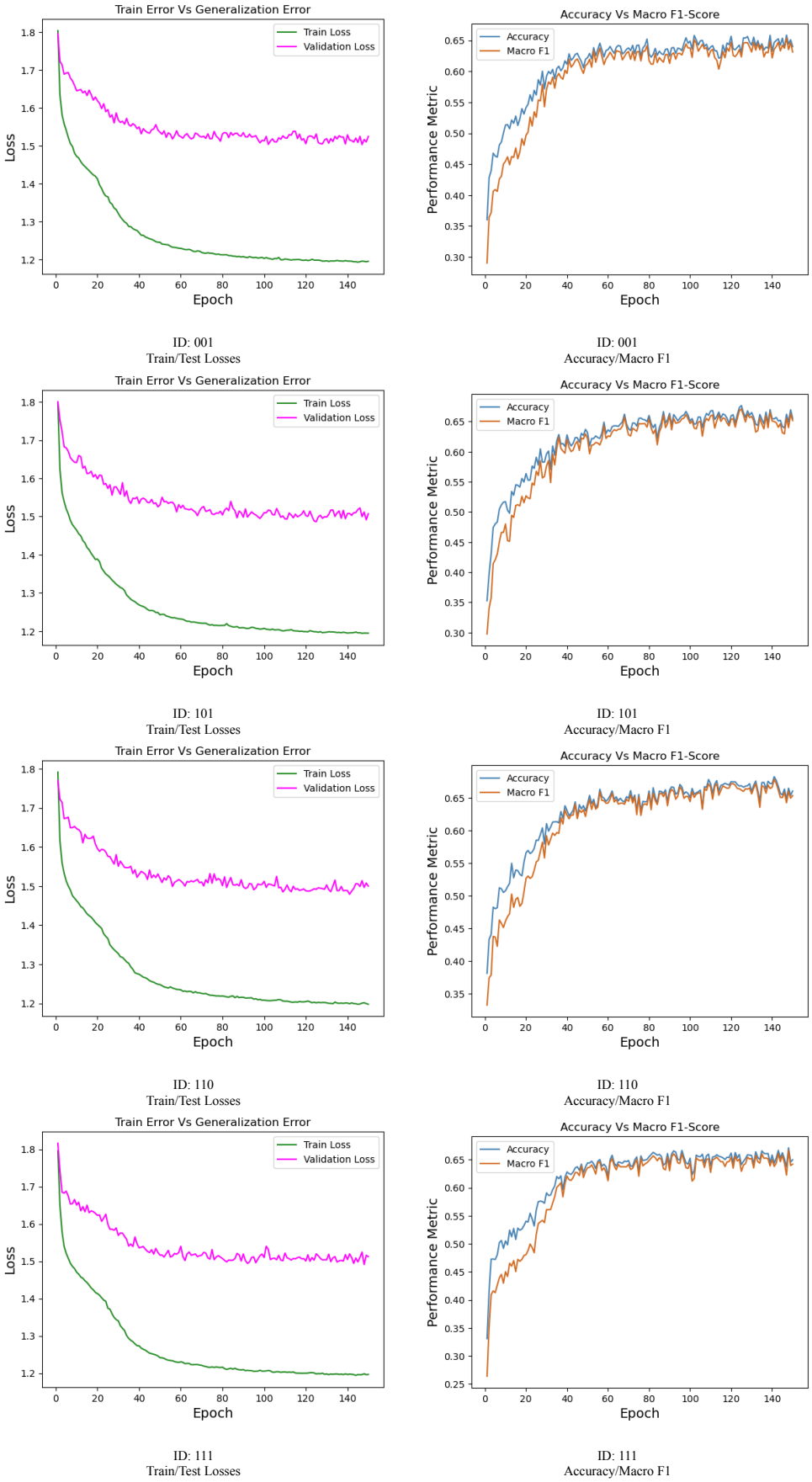
It is interesting to note that after conducting a 5-fold cross-validation process to obtain average performance metrics for the four models, the performances of (101), (110), and (111) appear to be poorer compared to the initial evaluation. This decline in performance is not necessarily attributed to overfitting caused by additional training epochs but may depend on two other factors:

- K-fold cross-validation results often deviate from the true mean, particularly when the value of K is relatively low (e.g., 5 or 10). The variability introduced by the cross-validation process can lead to differences in performance compared to the initial evaluation.
- There is a degree of variability or ”riffling” around the convergence mean.

Although overfitting might not be the case here, it is obvious that the early stopping algorithm implemented initially successfully detected convergence, as the models' classification performance did not significantly improve beyond the point of convergence.

Lastly, we could state that all of the above combinations would easily be a good fit for our speech emotion recognition model. Foremost, if the training and validation process were repeated, the results would very possibly be significantly different than before, especially considering that every batch in the training set is randomly re-shuffled each time.

The learning graphs for the training of 150 epochs without early stopping conditions can be seen in 6.5.



ID: 001
Train/Test Losses

ID: 001
Accuracy/Macro F1

ID: 101
Train/Test Losses

ID: 101
Accuracy/Macro F1

ID: 110
Train/Test Losses

ID: 110
Accuracy/Macro F1

ID: 111
Train/Test Losses

ID: 111
Accuracy/Macro F1

Figure 6.5: Learning Curves of the Qualifying models for 150 epochs.

Not much can be concluded from the learning curves in 6.5, as all classifiers have properly converged. Therefore, it is reasonable to compare them based on the final performance score achieved at epoch 150.

It appears that the parameter vector (110) not only achieves the maximum macro F1-score and accuracy for 150 epochs, but also offers faster execution time compared to other parameter vectors. This combination of superior performance and reduced execution time makes (110) an engaging choice for our SER model.

Table 6.6: Optimal Parameters for English SER Model.

CNN Architecture	Squirtle NET
Spectral Feature	Mel Spectrogram
NxN	60x60
Convolutional Output Channels	188, 156, 340
Convolutional Kernel Sizes	3, 3, 3
Pooling Kernel Sizes	2, 3, 3
Pooling Strides	2, 2, 2
Batch-Normalization Momentum	0.4
Dropout Probability	0.40
Fully-Connected Output Channels	3634, 2816, 7
Batch Size	32
Learning Rate	6e-05

Training and Testing on the Entire Dataset

In the final step of our English SER experimental phase, the SquirtleNET model, configured with the optimal model parameters determined earlier, was trained using 60x60 mel spectrograms from the entire training dataset. The model was trained for a large number of 4,200 epochs and evaluated on a separate testing dataset that had not been previously used.

Since the training data is a lot more than before and the testing data contain new unseen images, we might not have been able to predict the model's behavior based on its previous one, by using the already existing early stopping criteria. As a result, the conventional early stopping algorithm that had been used previously was not considered applicable in this scenario.

The objective here was rather to capture the model's behavior at the exact moment when the generalization loss reaches its minimum value. While this approach is similar to certain early stopping techniques, the intention here was not to prematurely stop the training process but test the model's performance every 10 epochs instead and only store the optimal model (the one which displays the minimum generalization loss). If the current testing loss was lower than the minimum loss recorded thus far, the model was stored or overwritten, and the current loss value was updated as the new minimum.

One of the reasons behind the decision to continue the training process without stopping it was to showcase the learning curve and potentially identify any signs of overfitting. By allowing the training to proceed for an entire duration of 4,200 epochs, it became possible to visualize the model's learning progress over time and display it in this thesis dissertation for future reference.

The total execution time for training and testing the SquirtleNET model for 4,200 epochs was approximately 253.04 minutes, which is equivalent to about 4 hours and 13 minutes.

A summary of the epochs when the model was updated during the training process, along with some significant performance metrics is provided in 6.7.

The model reached its optimal performance after being trained for 400 epochs. During this period, it achieved a minimum generalization loss of 1.348, accompanied by an Accuracy of **81.88%** and a Macro F1-score of **81.83%**. These metrics indicate the model's ability to generalize and accurately classify speech emotions.

Table 6.7: Every Update in Final English SER model's Training.

Epoch	Test Loss	Accuracy	F1-Score
0	1.61215	0.5453	0.4956
10	1.52352	0.6359	0.5847
20	1.41498	0.7461	0.7459
30	1.38019	0.7828	0.7816
40	1.36727	0.7945	0.7946
50	1.35884	0.8008	0.797
90	1.35772	0.8055	0.8073
150	1.35546	0.8078	0.8089
350	1.35534	0.8094	0.8077
400	1.348	0.8188	0.8183

As we can see in the confusion matrix plotted in Figure 6.7, it is apparent that predicting the neutral class was comparatively easier than the other classes. This observation can be attributed to the fact that the dataset contained a higher number of instances belonging to the neutral class compared to the instances of every other class.

Overall, the classification results appear promising and satisfactory. The confusion matrix, along with the other performance metrics, demonstrate that the classifier is capable of accurately distinguishing between the seven emotion classes. This indicates that the model has learned to capture and differentiate the unique features associated with each emotion.

However, it is important to acknowledge that the SER problem poses several challenges. The performance of the trained classifier may be influenced by various biases, limitations, and assumptions inherent to the dataset and the model itself. These biases could arise from factors such as data collection methods, class representation imbalances, and potential cultural or contextual differences.

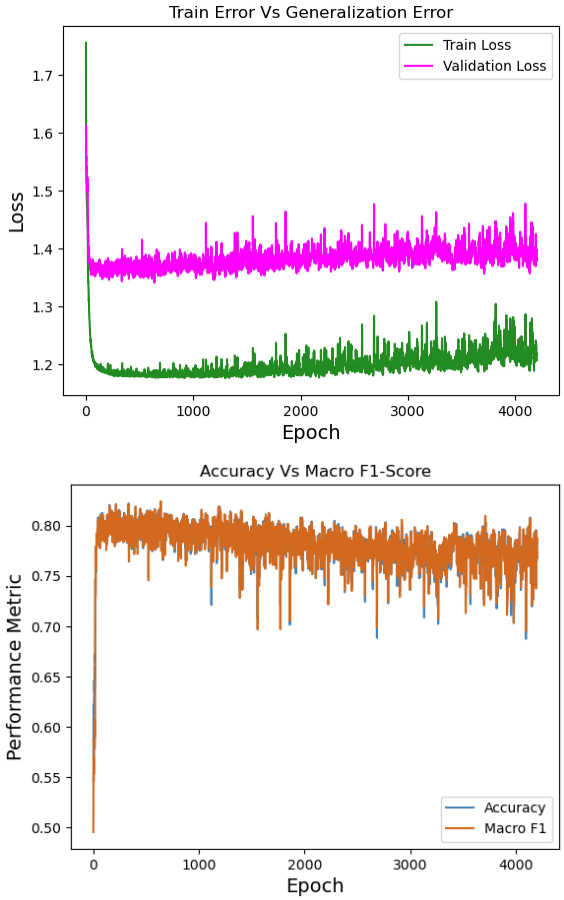


Figure 6.6: Losses and Performance Metrics.

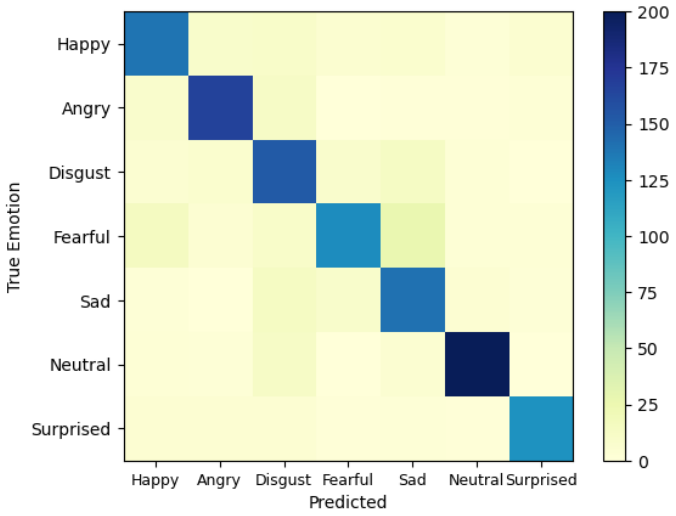


Figure 6.7: Confusion Matrix.

6.2 Phase 2: Evaluating the Efficacy of Transfer Learning in Greek SER

One possible question that could be raised is whether our English SER classifier is suitable for recognizing emotions in various types of speech data. It is interesting exploring speech emotion recognition for speech samples generated by speakers of different languages. And if the already trained classifier underperforms on different language data, it impels us to consider how much we can utilize its existing training and knowledge.

In this section of the thesis dissertation, we conducted a brief evaluation of the efficacy of our pre-trained model in recognizing emotions from speech data in the Greek language, specifically spoken by Greek actors. We analyzed the applicability of the pre-trained model's knowledge and determined its optimal performance in this context. We treated this SER classification as a different task from the previous one where we predicted emotions in English phrases.

We utilized transfer learning techniques to gain insights into the similarities and differences between the English and Greek SER tasks. Our approach involved freezing a subset of layers from the pre-trained model and training the remaining layers on the new Greek dataset. The goal was to achieve high performance despite the limited number of samples available. By employing this transfer learning methodology, we aimed to identify which pre-trained layers possess significant knowledge that can contribute to our new SER task in the Greek language.

The Dataset for Greek SER

The dataset we used was the Acted Emotional Speech Dynamic Database [28], a publically available SER dataset that contains samples of acted emotional speech in the Greek language. The speech samples were generated with five different emotions: anger, disgust, fear, happiness, and sadness, while the accuracy of human listeners was estimated to be at around 74% [29]. The dataset contains a total of 605 speech samples, however by removing a single corrupted sad audio file, we utilized 604; 121 anger files, 122 disgust files, 120 fear files, 119 happiness files, and 122 sadness files. 3 out of 5 speakers were female, making this dataset appropriate, as our English SER classifier already has some bias towards female speakers.

The sampling rate used in this case was 44.1 kHz. The pre-processing steps remained unchanged. Each audio signal was pre-emphasized, the zero parts at the beginning and end were trimmed, mel spectrograms were extracted, converted to dB scale, and then transformed into 60x60 square images using bilinear interpolation. Similarly to the previous experimental phase, the train/test set split was 80% to 20%. Unlike before, no manipulation technique was applied to any subset of the cross-validation set this time. In addition, no early stopping algorithm was applied during training, as the new dataset is small and the execution is typically really fast.

The new dataset, unfortunately, contains a smaller number of emotion classes compared

to the ones our pre-trained model had been originally trained to recognize. Specifically, only 5 out of the previous 7 emotions were present in this dataset. However, this did not pose a significant issue for the practical part of our transfer learning approach, as we just modified the output channels of the last fully connected layer to align with the available emotion classes in the Greek dataset, by replacing this layer with a new fully-connected layer of 5 output channels.

Like we mentioned before, during the experimentation, we assessed the impact of the transfer learning process on the pre-trained model by systematically freezing different sets of layers. Specifically, we did that while training each model for various combinations of learning rates and batch sizes. As a result, we were able to tune these hyperparameters while seeking the optimal subset of frozen layers.

Also, given that the dataset is completely balanced, we decided to employ Accuracy as the model's performance metric.

1st Experimental Stage

The first step of this experimental phase included a brief understanding over our new task. It should be noted, that except the subset of freezing layers, the learning rate and the batch size, all of the other parameters remained the same and were not used for tuning. The number of layers, as well as the number of neurons per layer and the analytical structure of our pre-trained CNN remained untouched. Similarly to the previous experiments, we used the CrossEntropyLoss as the criterion and the Adam without weight decay as our optimizer.

Hence, a pre-trained SquirtleNET model, with its first layers frozen, was trained for several hyperparameter combinations and later compared with a pre-trained SquirtleNET model which was now re-trained with a variety of learning rates and batch sizes without any layer frozen. All models were trained for a total of 150 epochs. The performances of each model were validated using the 5-fold cross validation technique. It is highly important to mention once again that the third and last fully-connected layer of the pre-trained model was replaced with a new, untrained fully-connected layer featuring 5 output channels.

The Table 6.8 presents different rows, each corresponding to a unique combination of learning rate and batch size used for training and two columns, each indicating the accuracy percentages achieved by each of the two models.

Table 6.8: Accuracy% for various (Learning Rate, Batch Size) vectors: Training with Frozen Layers in comparison to Non-Frozen, for 150 training epochs.

	4 frozen layers including: *Conv1, Batchnorm, Conv2, Conv3*	Non Frozen layer
LR = 1e-11 Batch size = 16	**16.78**	-
LR = 1e-11 Batch size = 32	**21.75**	-

LR = 1e-11 Batch size = 48	**16.36**	-
LR = 1e-11 Batch size = 64	**21.55**	-
LR = 5.18e-11 Batch size = 16	**18.43**	-
LR = 5.18e-11 Batch size = 32	**16.98**	-
LR = 5.18e-11 Batch size =48	**19.68**	-
LR = 5.18e-11 Batch size = 64	**20.08**	-
LR = 2.68e-10 Batch size = 16	**16.74**	-
LR = 2.68e-10 Batch size = 32	**18.02**	-
LR = 2.68e-10 Batch size = 48	**19.45**	-
LR = 2.68e-10 Batch size = 64	**16.77**	-
LR =1.39e-9 Batch size = 16	**18.83**	-
LR =1.39e-9 Batch size = 32	**20.11**	-
LR =1.39e-9 Batch size = 48	**22.55**	-
LR =1.39e-9 Batch size = 64	**19.27**	-
LR =7.2e-9 Batch size = 16	**15.72**	-
LR =7.2e-9 Batch size = 32	**22.77**	-
LR =7.2e-9 Batch size = 48	**16.76**	-
LR =7.2e-9 Batch size = 64	**22.36**	-
LR =3.73e-8 Batch size = 16	**22.98**	-
LR =3.73e-8 Batch size = 32	**19.45**	-
LR =3.73e-8 Batch size = 48	**21.1**	-
LR =3.73e-8 Batch size = 64	**22.99**	-
LR =1.93e-7 Batch size = 16	**37.68**	**33.33**

LR =1.93e-7 Batch size = 32	**34.16**	-
LR =1.93e-7 Batch size = 48	**27.95**	**24.23**
LR =1.93e-7 Batch size = 64	**26.32**	-
LR =1e-6 Batch size = 16	**45.35**	**50.11**
LR =1e-6 Batch size = 32	**46.39**	-
LR =1e-6 Batch size = 48	**43.47**	**42.45**
LR =1e-6 Batch size = 64	**38.72**	-
LR =5.18e-6 Batch size = 16	**51.98**	**59.0**
LR =5.18e-6 Batch size = 32	**52.4**	-
LR =5.18e-6 Batch size = 48	**52.19**	**55.08**
LR =5.18e-6 Batch size = 64	**51.15**	-
LR =2.68e-5 Batch size = 16	**55.51**	**59.42**
LR =2.68e-5 Batch size = 32	**53.23**	-
LR =2.68e-5 Batch size = 48	**51.78**	**60.26**
LR =2.68e-5 Batch size = 64	**54.68**	-
LR =1.39e-4 Batch size = 16	**53.85**	**59.22**
LR =1.39e-4 Batch size = 32	**53.64**	-
LR =1.39e-4 Batch size = 48	**54.67**	**56.94**
LR =1.39e-4 Batch size = 64	**54.27**	-
LR =7.2e-4 Batch size = 16	**22.38**	-
LR =7.2e-4 Batch size = 32	**30.45**	-
LR =7.2e-4 Batch size = 48	**45.33**	-
LR =7.2e-4 Batch size = 64	**43.94**	-

Based on our observations on this experimental phase, we have drawn the following conclusions:

- Learning rates outside the approximate range of 1e-6 to 1e-4 do not provide promising results. This finding allows us to narrow down our research area and focus on exploring learning rates within this range. By doing so, we can save computational time during the grid search process.
- Training the entire pre-trained model rather than just the first four layers, demonstrates better results. This suggests that utilizing the first pre-trained layers alone may not be sufficient for achieving optimal performance in the Greek SER task.

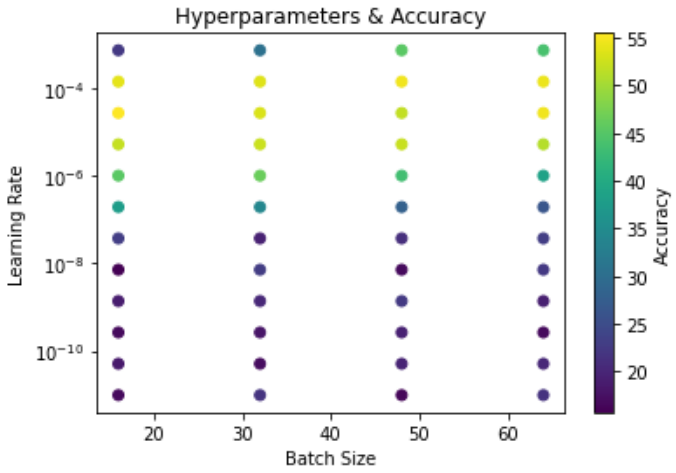


Figure 6.8: Accuracy with Hyperparameters exported from Table 6.8, for freezing the first 4 layers of the pre-trained model.

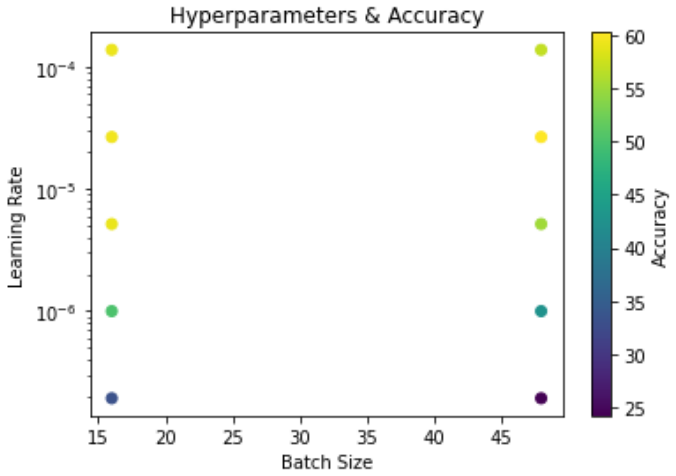


Figure 6.9: Accuracy with Hyperparameters exported from Table 6.8, for no frozen layers.

2nd Experimental Stage

To gain a deeper understanding of which pre-trained layers have a significant impact on the SER task, we conducted a more comprehensive grid search. In this second step of the experimental process, we restricted our learning rate range to six values between 1e-6 and 1e-4 and explored four different batch sizes: 16, 32, 48 and 64. The aim was to examine additional model variations and identify the optimal combination of frozen layers.

To simplify referencing the models, we assigned some letters as identity tags to each variation:

(a): Pre-Trained SquirtleNET with the following frozen layers: Convolutional 1, Batch-Normalization, Convolutional 2, Convolutional 3, Fully-Connected 1, Dropout

(b): Pre-Trained SquirtleNET with the following frozen layers: Convolutional 1, Batch-Normalization, Convolutional 2, Convolutional 3, Fully-Connected 1

(c): Pre-Trained SquirtleNET with the following frozen layers: Convolutional 1, Batch-Normalization, Convolutional 2, Convolutional 3

(d): Pre-Trained SquirtleNET with the following frozen layers: Convolutional 1, Batch-Normalization, Convolutional 2

(e): Pre-Trained SquirtleNET with the following frozen layers: Convolutional 1, Batch-Normalization

(f): Pre-Trained SquirtleNET with the following frozen layers: Convolutional 1

(g): Pre-Trained SquirtleNET with no frozen layers

Table 6.9: Accuracy% for various (Learning Rate, Batch Size) vectors per number of Frozen Layers, for 150 training epochs.

	(a)	(b)	(d)	(e)	(f)
LR = 1e-06 Batch size = 16	**37.91**	**47.43**	**50.12**	**51.15**	**51.77**
LR = 1e-06 Batch size = 32	**34.37**	**42.45**	**48.05**	**49.48**	**50.74**
LR = 1e-06 Batch size = 48	**31.28**	**39.77**	**43.69**	**43.9**	**42.02**
LR = 1e-06 Batch size = 64	**30.84**	**39.98**	**42.65**	**42.04**	**41.83**
LR = 2.51e-06 Batch size = 16	**40.38**	**48.03**	**53.85**	**54.67**	**55.93**
LR = 2.51e-06 Batch size = 32	**37.91**	**48.06**	**51.99**	**55.3**	**55.3**
LR = 2.51e-06 Batch size = 48	**39.15**	**46.81**	**50.54**	**52.81**	**53.44**
LR = 2.51e-06 Batch size = 64	**40.4**	**41.22**	**47.21**	**50.73**	**51.15**

LR = 6.31e-06 Batch size = 16	**38.73**	**52.18**	**54.25**	**57.78**	**56.33**
LR = 6.31e-06 Batch size = 32	**41.01**	**51.78**	**52.19**	**56.13**	**57.79**
LR = 6.31e-06 Batch size = 48	**40.18**	**50.74**	**55.29**	**54.25**	**55.91**
LR = 6.31e-06 Batch size = 64	**38.72**	**47.22**	**54.46**	**53.64**	**54.68**
LR = 1.58e-05 Batch size = 16	**42.45**	**51.99**	**55.7**	**58.2**	**57.57**
LR = 1.58e-05 Batch size = 32	**38.73**	**52.4**	**56.12**	**57.16**	**56.75**
LR = 1.58e-05 Batch size = 48	**39.97**	**51.98**	**55.5**	**56.95**	**58.39**
LR = 1.58e-05 Batch size = 64	**39.56**	**50.31**	**56.33**	**55.71**	**53.64**
LR =3.98e-05 Batch size = 16	**41.0**	**52.81**	**55.29**	**56.74**	**54.27**
LR =3.98e-05 Batch size = 32	**42.25**	**52.39**	**56.75**	**55.72**	**54.68**
LR =3.98e-05 Batch size = 48	**39.15**	**49.71**	**55.09**	**55.93**	**56.96**
LR =3.98e-05 Batch size = 64	**40.18**	**51.56**	**52.82**	**56.33**	**56.96**
LR = 1e-04 Batch size = 16	**41.42**	**52.59**	**52.17**	**58.2**	**54.87**
LR = 1e-04 Batch size = 32	**38.72**	**50.1**	**55.08**	**53.23**	**56.34**
LR = 1e-04 Batch size = 48	**41.42**	**50.32**	**53.86**	**54.45**	**54.47**
LR = 1e-04 Batch size = 64	**38.73**	**51.35**	**54.45**	**56.94**	**51.13**

In general, we observe an increasing pattern in accuracy scores as we decrease the number of frozen layers in the pre-trained model. However, this observation does not hold across all cases. By calculating the average accuracy for each model across all hyperparameter combinations, we obtained the following results: 38.94% for model (a), 48.88% for model (b), 52.64% for model (d), 54.06% for model (e), and 53.87% for model (f).

These average accuracy values possibly indicate that there is a modest performance improvement when we utilize and thus freeze the pre-trained batch normalization layer, as opposed to retraining it. This suggests that there might be some transferable knowledge from the original English SER task to the new Greek SER task. Yet, the average accuracy likely plays a role of minor importance to the usefulness of the classifier's pre-learned patterns, and we might need more proof to support such an argument. After all, we are

searching for the highest performance and the optimal hyperparameter vectors when tuning, and we still have not received significant results.

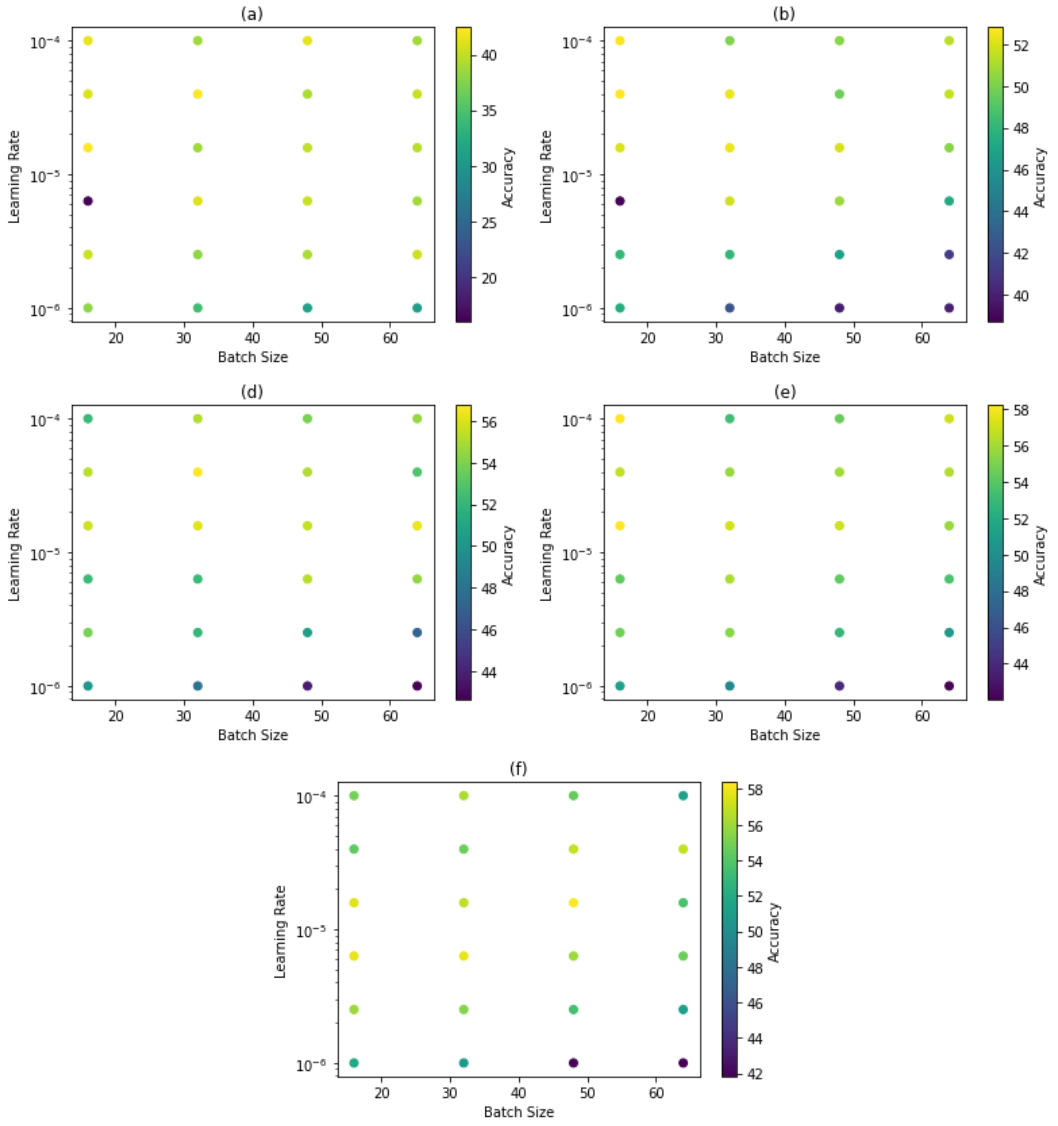


Figure 6.10: Accuracy with Hyperparameters exported from Table 6.9, for the different subsets of frozen layers.

3rd Experimental Stage

A slightly different approach that involved a quick analysis of the transferability of pre-trained layers is adopted. In terms of training duration, this step was quickly completed, due to the lower computational load involved. We specifically focused on models (c), (e), and (g) and lightly trained them for only 5 epochs.

We aimed to investigate whether any of the pre-trained layers possessed initial transferable

knowledge that could contribute to the classification ability of the network and to examine if any sudden changes in the content of the network, such as freezing or unfreezing specific layers, would have an impact on the classification ability. By closely inspecting the performance after a few epochs of training, we might identify any significant changes or degradation in accuracy that might have arisen from altering the network's structure.

Table 6.10: Accuracy% for various (Learning Rate, Batch Size) vectors per number of Frozen Layers, for 5 training epochs.

	(c)	(e)	(g)
LR = 1e-06 Batch size = 16	**22.59**	**20.51**	**20.69**
LR = 1e-06 Batch size = 32	**23.81**	**18.01**	**20.29**
LR = 1e-06 Batch size = 48	**21.1**	**15.3**	**23.6**
LR = 2.51e-06 Batch size = 16	**31.89**	**22.37**	**21.94**
LR = 2.51e-06 Batch size = 32	**24.85**	**23.2**	**21.95**
LR = 2.51e-06 Batch size = 48	**19.66**	**20.49**	**23.17**
LR = 6.31e-06 Batch size = 16	**34.79**	**37.9**	**29.61**
LR = 6.31e-06 Batch size = 32	**35.6**	**35.2**	**23.61**
LR = 6.31e-06 Batch size = 48	**25.67**	**28.57**	**21.93**
LR = 1.58e-05 Batch size = 16	**44.33**	**46.8**	**36.02**
LR = 1.58e-05 Batch size = 32	**43.46**	**45.35**	**30.23**
LR = 1.58e-05 Batch size = 48	**37.91**	**39.77**	**24.45**
LR =3.98e-05 Batch size = 16	**50.32**	**52.18**	**43.88**
LR =3.98e-05 Batch size = 32	**46.59**	**51.36**	**39.33**
LR =3.98e-05 Batch size = 48	**46.38**	**48.05**	**34.18**
LR = 1e-04 Batch size = 16	**51.98**	**53.22**	**43.67**
LR = 1e-04 Batch size = 32	**52.6**	**49.51**	**43.05**
LR = 1e-04 Batch size = 48	**51.57**	**50.75**	**37.7**

By computing the average accuracy percentages for models (c), (e), and (g) across every pair of learning rate and batch size values in 6.10, we obtained the following scores: 36.95% for (c), 36.59% for (e), and 29.96% for (g). These results further support the observation that the pre-trained SquirtleNET model possesses some degree of knowledge applicable to the Greek SER task, since the more the frozen layers, the higher the accuracy.

It is evident that in this scenario, for a limited number of epochs, unlike the previous phase where the training duration was longer, increasing the number of frozen layers leads to improved performance of the pre-trained model.

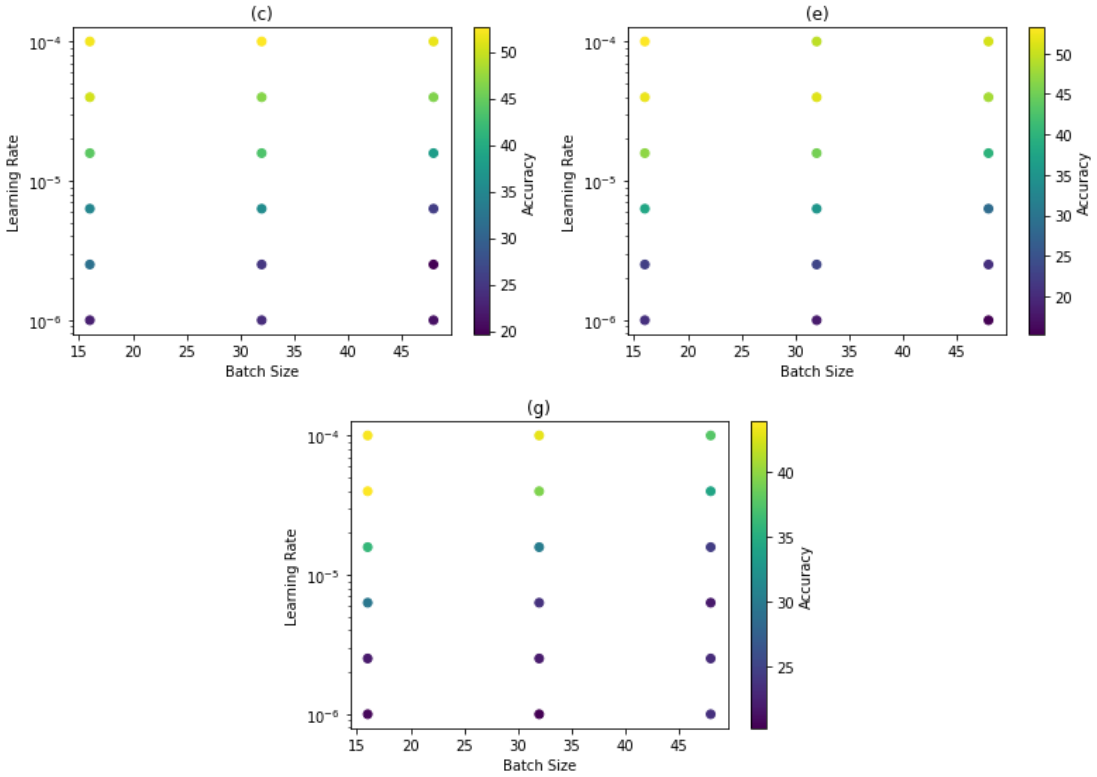


Figure 6.11: Accuracy with Hyperparameters exported from Table 6.10, for the different subsets of frozen layers, when only training for 5 epochs.

4th Experimental Stage

To validate the applicability of transfer learning in this context, **we conducted a final small grid search using a completely untrained SquirtleNET model**. The model was trained for 150 epochs and evaluated using the 5-fold cross-validation method, in three instances, all with a batch size of 32 and learning rates of 1e-6, 1e-5, and 1e-4, respectively. The accuracy scores obtained from the 5-fold cross-validation process were 57.97%, 55.90%, and 59.84%, respectively. If compared with the ones in table 6.9, these results indicate that training an untrained model yields better overall performance compared to training a pre-trained model. These findings suggest that while the pre-trained SquirtleNET model

provides some applicable knowledge, it may not be sufficient for optimal performance in Greek SER.

Training and Testing on the Entire Dataset

Finally, we took advantage of the remaining testing data to test the pre-trained model with the highest accuracy so far and then evaluated it. We have investigated tables 6.8, 6.9, 6.10 and selected the highest-performance case. The highest accuracy present in our experiments is displayed in table 6.8 for (LR, Batch Size) = (2.68e-5, 48) and for a model with no frozen layers present (every layer is re-trained). We proceeded to train the pre-trained English SER model, on the entire Greek training set for a total of 2,000 epochs, and test it after every epoch, storing the results in order to plot the learning curves. We monitored the performance throughout the training process and updated the best model every 5 epochs. If the testing loss was smaller than the previous minimum testing loss, we set the current loss as the new minimum and stored/overwrote the model accordingly, following the same procedure as before. The specific updates are provided in table 6.11.

Table 6.11: Every Update in Final Greek SER model's Training.

Epoch	Test Loss	Accuracy	F1-Score
0	1.60715	0.1736	0.0592
5	1.48266	0.3967	0.3477
10	1.42297	0.4545	0.4313
15	1.42106	0.4711	0.4680
20	1.41261	0.4711	0.4549
25	1.37832	0.5207	0.5182
30	1.37434	0.5372	0.5263
40	1.35561	0.5372	0.5416
55	1.34463	0.5537	0.5570
65	1.32099	0.5785	0.5717
75	1.31576	0.5950	0.5970
90	1.31144	0.5785	0.5797
95	1.29611	0.5950	0.5932
110	1.29428	0.6116	0.6063
125	1.29117	0.5950	0.5918
165	1.28960	0.5950	0.5943
170	1.28484	0.5950	0.5988
350	1.27992	0.6033	0.6036
355	1.27898	0.6198	0.6203
360	1.27496	0.6281	0.6230
375	1.27183	0.6198	0.6200
385	1.27140	0.6198	0.6211
415	1.25427	0.6364	0.6356
605	1.25305	0.6446	0.6465
615	1.23522	0.6612	0.6580

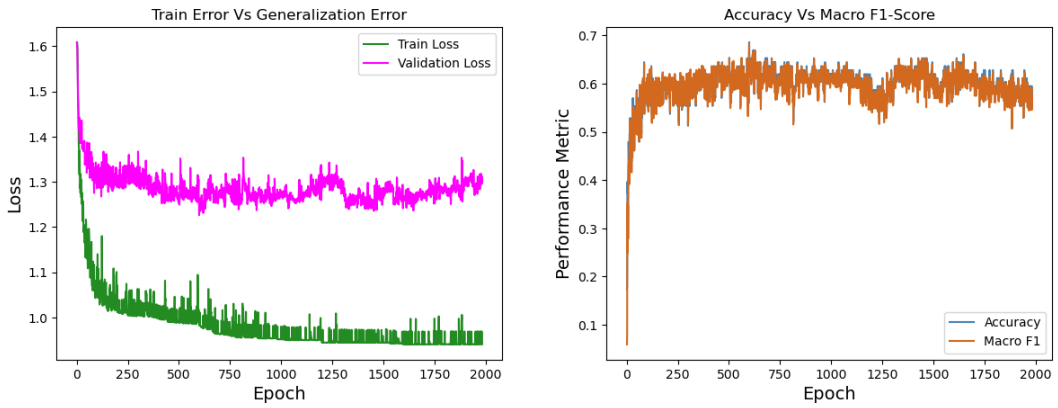


Figure 6.12: Final Greek SER model's Learning Curves, after training a pre-trained SquirtleNet on new 60x60 mel spectrograms derived from the AESD Greek audio data.

After training the pre-trained model with no frozen layers for 615 epochs on Greek speech data, we achieved a promising level of Accuracy and Macro F1-score in recognizing emotions. The model achieved a total Accuracy of **66.12%** and a Macro F1-score of **65.80%** when distinguishing between the five different emotion classes in the Greek speech data of the Acted Emotional Speech Dynamic Database.

While the classifier's performance is not bad, considering it has only been trained on a very small new dataset, we were hoping for higher performance results by leveraging the existing knowledge of a pre-trained model on a SER task. However, there are several crucial factors to consider:

1. The pre-trained model was already trained on 7 instead of 5 classes which were required for the Greek SER task, with a significant focus on identifying neutral class patterns, which are not present in the Greek AESDD dataset. As a result, some valuable knowledge may have been wasted.
2. English speakers may express emotions differently from Greek speakers. Due to the vast linguistic differences between the two languages, certain emotions might be conveyed using a range of pitches in one language but not often utilized in the other. This aspect requires further research and a comprehensive study.
3. When applying transfer learning, additional parameter tuning may be necessary. Secondary parameters, such as output channels and kernel sizes, could be adjusted to improve classification results on the new task.
4. The element of acted emotions by actors in the dataset may have distorted the true emotional context of the audio speeches, causing a loss of similarity between the two tasks.
5. The timing and duration of the audio samples in the dataset might affect performance. Emotions can be expressed differently depending on the length of the speech segment and the specific moments within the speech. Nevertheless, duration also affects the

information stored inside a Mel Spectrogram image, which is further altered after the bilinear image interpolation.

6. The Mel-Spectrogram feature may not effectively capture the shared information between the two language sets of speeches. The two tasks could exhibit similarities that are not adequately represented by this particular spectral feature.

The Mel-Spectrogram Dissimilarities

To investigate the limitations of the pre-trained English SER model in utilizing its existing knowledge on the Greek SER, we will dedicate this small subsection to highlight the dissimilarities observed between the extracted mel spectrogram features from the English dataset and the Greek dataset. Attached figures will be provided to illustrate these differences. It is important to note that a 2D mel spectrogram was chosen to capture emotional patterns within speech samples. Consequently, the noticeable differences between the mel spectrograms of the two datasets strongly suggest significant variations in emotional patterns between English and Greek languages.

In Figure 6.13, we present nine randomly selected mel-spectrogram images extracted from Greek audio speech samples, each labeled with its corresponding emotional class. Additionally, in Figure 6.14, we showcase mel-spectrogram images extracted from English speech data. In this case, we also indicate the specific dataset from which each image was derived.

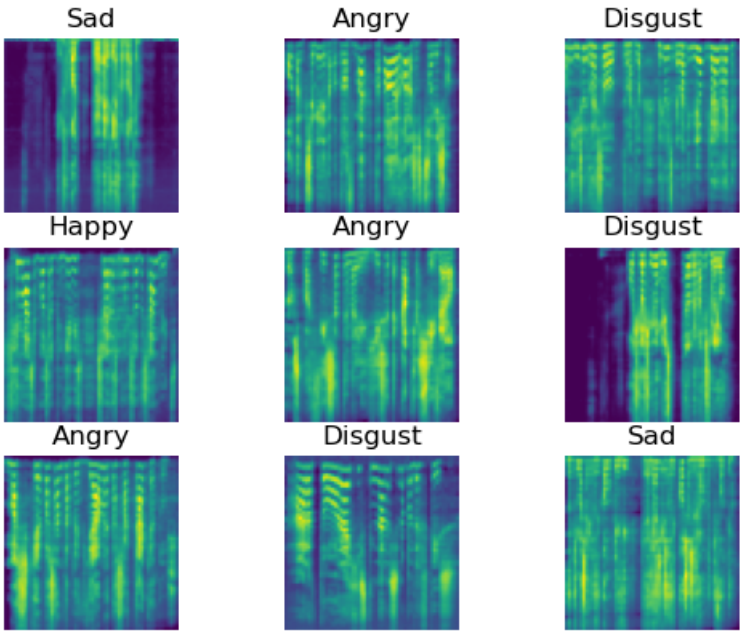


Figure 6.13: A random set of 60x60 Mel Spctrograms derived from the AESD Greek audio data, with their labels/classes.

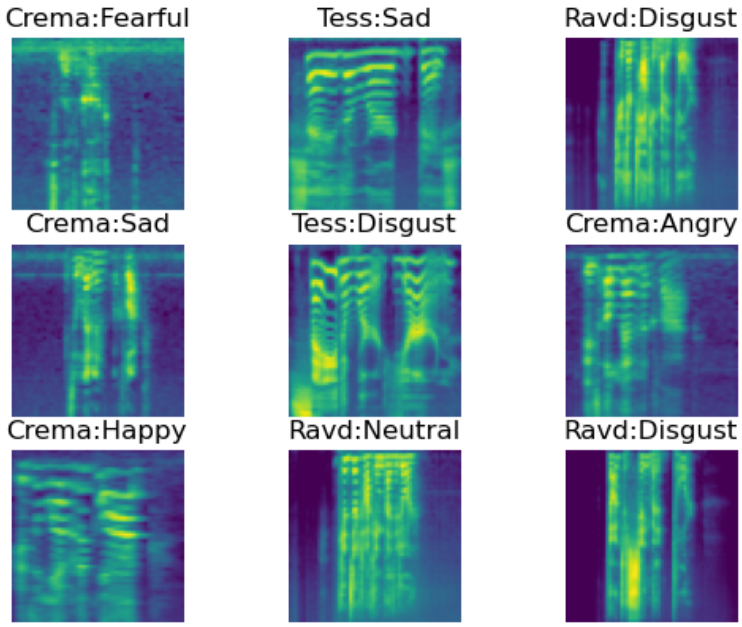


Figure 6.14: A random set of 60x60 Mel Spctrograms derived from the English audio data, with their labels/classes and the specific dataset they were derived from.

There are a few observations which can be made in the two figures. In Figure 6.13, the images appear to have a more uniform distribution of intensity across the time axis. On the other hand, Figure 6.14 shows images that appear more 'edgy' with larger regions of lower intensities. However, the interpretation of the plots and the identification of the key differences between the two sets of images will be left to the reader, as the objective of this thesis is not focused on image analysis. Nonetheless, it is significant to acknowledge that possible dissimilarities between the Mel Spectrograms in the two languages could have posed some major challenges for the transfer learning approach.

6.3 Conclusions & Future Work

Mel Spectrograms have shown the highest performance among the English-SER 2D spectral features, while STFT Chromagrams have also demonstrated relatively high scores. On the other hand, the MFCC features have not contributed significantly to building a capable classifier.

Regarding image size, experiments with 50x50, 60x60, and 80x80 images have yielded promising results. The 30x30 images appeared to lack essential information, resulting in reduced classification performance.

The optimal CNN architecture is SquirtleNET, delivering superior results, closely followed by CharmaNET, which also provided satisfying outcomes. Unfortunately, BulbaNET did not meet our expectations. It should be noted that these architectures could be furtherly

altered by adding more layers. Consequently, the research could be expanded on wider CNN architectures.

In general, the Speech Emotion Recognition task was shifted into an Image Classification task. We suggest that those interested in replicating our approach should utilize Mel Spectrogram features, specifically of size 60x60. Larger image sizes do not necessarily lead to higher accuracy.

For the application of Transfer Learning to the Greek language, it was discovered that for longer number of training epochs, enough for the model's performance metrics to converge, more freedom is required and consequently fewer frozen layers achieve higher performance. Conversely, when the training duration is too short and the model is far from convergence, better performance is obtained with more frozen layers, possibly suggesting that some degree of knowledge is embedded within the structure of these pre-trained layers.

However, training a fully defined SquirtleNET architecture for 150 epochs revealed that leveraging a pre-trained model is not necessary to achieve high performance. Therefore, the task of English SER and the larger amount of English data did not significantly aid in the new task of Greek SER.

As for future work, the defined SquirtleNET model can be trained with 60x60 Mel Spectrograms extracted from speech, and be utilized in several datasets that follow the Ekman's model. Experiments could also be made for more than 7 emotional classes, while a regression approach could also be adopted for continuous emotion vector-values.

As we've already mentioned, the research could be extended to wider CNN structures. Some other suggestions for a future research include; experiments with various pre-processing techniques, executing parameter tuning with more parameters included (eg. activation functions, algorithmic arguments), or applying transfer learning to more languages and testing if we can leverage knowledge.

Keep in mind that supervised learning relies on human labeling, which can be prone to errors, so this approach may encounter limitations. The recognition of emotions in speech is a complex process, and even humans may not always execute it accurately. Alternative methods such as unsupervised learning could be implemented and combined with our existing model, to provide better and safer results.

Bibliography

[1] Advances in data analysis with computational intelligence methods.

[2] Houwei Cao, David G. Cooper, Michael K. Keutmann, Ruben C. Gur, Ani Nenkova, and Ragini Verma. Crema-d: Crowd-sourced emotional multimodal actors dataset. *IEEE Transactions on Affective Computing*, 5(4):377–390, 2014.

[3] Houwei Cao, David G. Cooper, Michael K. Keutmann, Ruben C. Gur, Ani Nenkova, and Ragini Verma. Crema-d: Crowd-sourced emotional multimodal actors dataset. *IEEE Transactions on Affective Computing*, 5(4):377–390, 2014.

[4] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning.

[5] Kate Dupuis and M. Kathleen Pichora-Fuller. Tess Dataset, 2010.

[6] Paul Ekman. Facial expressions of emotion: New findings, new questions. 3(1):34–38.

[7] Moataz El Ayadi, Mohamed S. Kamel, and Fakhri Karray. Survey on speech emotion recognition: Features, classification schemes, and databases. 44(3):572–587.

[8] Daniel P.W. Ellis. Chroma feature analysis and synthesis, April 21 2007.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. The MIT Press.

[10] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview.

[11] Md Rashidul Hasan, Mustafa Jamil, MGRMS Rahman, et al. Speaker identification using mel frequency cepstral coefficients. *variations*, 1(4):565–568, 2004.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

[13] Earl J. Kirkland. *Advanced Computing in Electron Microscopy*. Springer US.

[14] Steven R. Livingstone and Frank A. Russo. The ryerson audio-visual database of emotional speech and song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in north american english. 13(5):e0196391.

[15] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. Librosa.

[16] Tom M. Mitchell. *Machine Learning*. McGraw-Hill series in computer science. McGraw-Hill.

[17] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*, volume 5. Springer, 2015.

[18] Nils J Nilson. Introduction to machine learning. an early draft of a proposed textbook. Stanford, Ca., http://robotics.stanford.edu/~nilsson/MLBOOK.pdf, 2005.

[19] Sanaul Haq Peter J. Jackson. The Surrey Audio-Visual Expressed Emotion (SAVEE) Database. URL: http://personal.ee.surrey.ac.uk/Personal/P.Jackson/SAVEE/.

[20] Pillow Contributors. Pillow: Python imaging library (fork).

[21] Python 3.10. https://www.python.org/downloads/release/python-310/, 2021. Python Software Foundation.

[22] Sebastian Ruder. An overview of gradient descent optimization algorithms.

[23] James A. Russell. A circumplex model of affect. 39(6):1161–1178.

[24] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Pearson, third edition, global edition edition.

[25] Björn W. Schuller. Speech emotion recognition: two decades in a nutshell, benchmarks, and ongoing trends. 61(5):90–99.

[26] S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. 8(3):185–190.

[27] Krishna Vedala. Krishna Vedala, CC BY-SA 3.0. Available at http://creativecommons.org/licenses/by-sa/3.0/, 2018. via Wikimedia Commons.

[28] N. Vryzas, R. Kotsakis, A. Liatsou, C. A. Dimoulas, and G. Kalliris. Speech emotion recognition for performance interaction. *Journal of the Audio Engineering Society*, 66(6):457–467, 2018.

[29] N. Vryzas, M. Matsiola, R. Kotsakis, C. Dimoulas, and G. Kalliris. Subjective Evaluation of a Speech Emotion Recognition Interaction Framework. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, page 34. ACM, September 2018.