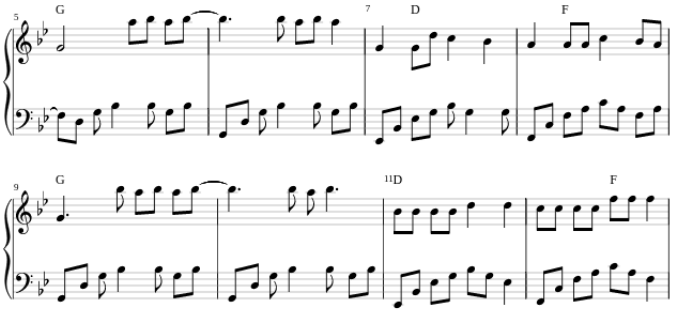
National Technical University of Athens
School of Electrical and Computer Engineering
Data Science & Machine Learning MSc Program

Chord Recognition using Deep Learning Techniques

Diploma Thesis

of

ATHANASIOS AIDINIS



Supervisor: Athanasios Voulodimos
Assistant Professor

Athens, June 2024

National Technical University of Athens
School of Electrical and Computer Engineering
Data Science & Machine Learning MSc Program

Chord Recognition using Deep Learning Techniques

Diploma Thesis

of

ATHANASIOS AIDINIS

Supervisor: Athanasios Voulodimos
Assistant Professor

Approved by the examination committee on July 2024.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Athanasios Voulodimos	Georgios Stamou	Andreas-Georgios Stafilopatis
Assistant Professor	Professor	Professor

Athens, June 2024

National Technical University of Athens
School of Electrical and Computer Engineering
Data Science & Machine Learning MSc Program

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

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

(Signature)

.....
Athanasios Aidinis
July 2024

Abstract

This thesis presents an innovative approach to audio chord recognition, aiming to automatically identify and classify fundamental chord structures within music pieces. Leveraging Convolutional Neural Networks (CNNs) with Bidirectional Long Short-Term Memory (biLSTM) layers, advanced feature engineering, and post-processing techniques rooted in music theory, our research enhances the accuracy and robustness of chord recognition systems. By extracting features from chord representations such as root, bass, and triad qualities, and segmenting the problem into distinct components, our framework creates a solid ground to enhance the accuracy of chord recognition. Additionally, we employ transfer learning techniques to capitalize on pre-trained models, fine-tuning them for our specific chord recognition task, thus improving generalization and robustness. Moreover, our exploration encompasses various Fourier transforms for feature extraction, including Short-Time Fourier Transform (STFT) and Constant Q Transform (CQT), to capture essential information from audio signals and optimize chord recognition performance. Through extensive experimentation and evaluation of different CNN and biLSTM configurations, as well as post-processing techniques, our approach demonstrates significant enhancements in several aspectes of chord recognition. Overall, this research contributes a comprehensive framework that leverages deep learning methodologies, sophisticated feature engineering, and post-processing techniques, showcasing its potential to advance music information retrieval systems.

Keywords

Audio Chord Estimation, Deep Learning, CNN, BiLSTM, R – CNN, Neural Networks, MIREX, CQT, Transfer Learning

Περίληψη

Η παρούσα διπλωματική εργασία παρουσιάζει μια καινοτόμα προσέγγιση για την αναγνώριση συγχορδιών από ήχο, με στόχο την αυτόματη ταυτοποίηση και ταξινόμηση των βασικών δομών συγχορδιών σε μουσικά κομμάτια. Αξιοποιώντας Συνελικτικά Νευρωνικά Δίκτυα (CNNs) με Δισδιάστατες Στρώσεις Μακροπρόθεσμης Βραχυπρόθεσμης Μνήμης (biLSTM), προχωρημένη διαδικασία εξαγωγής χαρακτηριστικών, και τεχνικές μετά-επεξεργασίας βασισμένες στη μουσική θεωρία, η έρευνά μας συμβάλει στην βελτίωση της ακρίβειας και της ανθεκτικότητας των συστημάτων αναγνώρισης συγχορδιών. Με την εξαγωγή χαρακτηριστικών από αναπαραστάσεις συγχορδιών όπως η ρίζα, το μπάσο και οι ποιότητες τριάδας, και την τμηματοποίηση του προβλήματος σε διακριτά συστατικά, το πλαίσιο μας θέτει γερά θεμέλια για την ενίσχυση την ακρίβεια της αναγνώρισης συγχορδιών. Επιπλέον, χρησιμοποιούμε τεχνικές μεταφοράς μάθησης (Transfer Learning) για να εκμεταλλευτούμε προεκπαιδευμένα μοντέλα, τα οποία προσαρμόζουμε για τη συγκεκριμένη μας εργασία μερών των συγχορδιών, βελτιώνοντας έτσι τη γενίκευση και την ανθεκτικότητα. Επιπρόσθετα, η έρευνά μας εξετάζει διάφορους μετασχηματισμούς Fourier για την εξαγωγή χαρακτηριστικών, συμπεριλαμβανομένων των Βραχυπρόθεσμων Μετασχηματισμών Fourier (STFT) και του Συνεχούς Μετασχηματισμού Q (CQT), για να συλλάβει βασικές πληροφορίες από τα ηχητικά σήματα και να βελτιστοποιήσει την απόδοση της αναγνώρισης συγχορδιών. Μέσα από εκτεταμένα πειράματα και αξιολόγηση διαφορετικών διαμορφώσεων CNN και biLSTM, καθώς και τεχνικών μετά-επεξεργασίας, η προσέγγισή μας επιδεικνύει σημαντικά σημεία βελτίωσης στην ακρίβεια αναγνώρισης συγχορδιών. Συνολικά, αυτή η έρευνα συμβάλλει ένα ολοκληρωμένο πλαίσιο που αξιοποιεί μεθοδολογίες βαθιάς μάθησης, προηγμένη μηχανική χαρακτηριστικών και τεχνικές μετά-επεξεργασίας, αναδεικνύοντας το δυναμικό της να προωθήσει τα συστήματα ανάκτησης μουσικών πληροφοριών.

Λέξεις κλειδιά

Εκτίμηση Συγχορδιών από Ήχο, Βαθιά Μάθηση, CNN, BiLSTM, R – CNN, Νευρωνικά Δίκτυα, MIREX, CQT, Μεταφορά Μάθησης

to my parents

Acknowledgements

Firstly, I would like to thank my parents from the bottom of my heart. My academic and personal achievements have been firmly based on their unfailing support, encouragement, and believe in me. I will always be indebted to them for their unending love and dedication, as well as their great sacrifices and advice.

I also want to thank A. Voulodimos and S. Kantarelis deeply. Their knowledge, direction, and help have been invaluable in producing my thesis.

I want to sincerely thank each and every one of you for your unwavering encouragement and support. Your amazing contributions have made this thesis possible; without them, it would not have been conceivable.

Athens, June 2024

Athanasios Aidinis

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Audio chords, as fundamental components of music, are constructed using specific harmonic principles and are perceived as pleasing to the human ear. These chord structures serve as building blocks for musical compositions, providing harmonic stability and emotional depth. The intricate relationships between different chord components, such as root, bass, and triad qualities, contribute to the richness and complexity of musical pieces.

In recent years, deep learning has emerged as a powerful tool for extracting meaningful patterns and relationships from complex, multidimensional data. Its ability to identify nonlinear correlations makes it particularly well-suited for tasks involving audio signal processing and music analysis. This study aims to leverage the capabilities of deep learning to uncover the underlying connections between audio chord structures and their harmonic principles.

The challenge of estimating audio chords has garnered significant attention within the MIREX (Music Information Retrieval Evaluation eXchange) community, attracting participation from numerous researchers annually. MIREX serves as a platform for evaluating and comparing different methods and algorithms for music information retrieval tasks, including chord estimation. It provides specific guidelines regarding datasets, vocabularies, past submissions, and evaluation metrics, all of which are essential for conducting rigorous research in this field.

In this thesis, we employ deep learning techniques and feature extraction methods grounded in music theory to address the challenge of audio chord estimation. We break down the problem into multiple sub-problems based on principles of music comprehension, allowing us to focus on different aspects of chord recognition separately. Additionally, we integrate post-processing techniques inspired by music theory to refine and improve the accuracy of our chord predictions. The results of our approach are compared against other solutions and benchmarks established by MIREX, providing a comprehensive evaluation of our methodology's effectiveness and performance. Through this research, we aim to contribute to the ongoing efforts in advancing the field of music information retrieval and enhancing our understanding of audio chord recognition.

1.1 Structure of the thesis

The following thesis is organized in chapters:

- Chapter 7 focuses on certain concepts that serve as theoretical background.
- Chapter 8 analyses the data and the pre-processing steps followed.
- Chapter 9 focuses on experiments on various model architectures and on the analysis of the results.
- Chapter 10 presents the post-processing techniques that were implimented.
- Chapter 11 contains the evaluation of the proposed model on MIREX metrics along with the comparison with other proposed models and the final conclusions drawn from the experiments, along with suggestions for future work.

Chapter 2

Εισαγωγή

Οι συγχορδίες, ως βασικά στοιχεία της μουσικής, κατασκευάζονται χρησιμοποιώντας συγκεκριμένες αρμονικές αρχές και αντιλαμβάνονται ως ευχάριστα ακούσματα από το ανθρώπινο αυτί. Αυτές οι δομές συγχορδιών λειτουργούν ως κομμάτια για μουσικές συνθέσεις, παρέχοντας αρμονική σταθερότητα και συναισθηματικό βάθος. Οι πολύπλοκες σχέσεις μεταξύ διαφορετικών συνιστωσών ακόρντων, όπως η ρίζα και η τριάδα, συμβάλλουν στην πλούσια και πολύπλοκη φύση των μουσικών κομματιών.

Τα τελευταία χρόνια, η εμφάνιση της βαθιάς μάθησης έχει εξελιχθεί σε ένα ισχυρό εργαλείο για την εξαγωγή σημαντικών προτύπων και σχέσεων από πολύπλοκα, πολυδιάστατα δεδομένα. Η ικανότητά της να αναγνωρίζει μη γραμμικές συσχετίσεις την καθιστά ιδιαίτερα κατάλληλη για εργασίες που σχετίζονται με την επεξεργασία των ακουστικών σημάτων και την ανάλυση της μουσικής. Αυτή η μελέτη στοχεύει στην εκμετάλλευση των δυνατοτήτων της βαθιάς μάθησης για την ανάδειξη των υποκείμενων διασυνδέσεων μεταξύ των δομών ακόρντων και των αρμονικών τους αρχών.

Η πρόκληση της εκτίμησης των συγχορδιών είναι ευρέως αναγνωρισμένη στην κοινότητα του MIREX (Music Information Retrieval Evaluation eXchange), προσελκύοντας τη συμμετοχή πολλών ερευνητών κάθε χρόνο. Το MIREX λειτουργεί ως πλατφόρμα για την αξιολόγηση και σύγκριση διαφορετικών μεθόδων και αλγορίθμων για την ανάκτηση πληροφοριών μουσικής, συμπεριλαμβανομένης της εκτίμησης συγχορδιών. Παρέχει συγκεκριμένες οδηγίες σχετικά με σύνολα δεδομένων, λεξιλογίου, προηγούμενες υποβολές και μετρικές αξιολόγησης, όλες απαραίτητες για τη διεξαγωγή αυστηρών ερευνών σε αυτό το πεδίο.

Σε αυτή τη διπλωματική εργασία, χρησιμοποιούμε τεχνικές βαθιάς μάθησης και μεθόδους εξαγωγής χαρακτηριστικών βασισμένες στη θεωρία της μουσικής για την αντιμετώπιση της πρόκλησης της εκτίμησης συγχορδιών. Χωρίζουμε το πρόβλημα σε πολλαπλά υποπροβλήματα βασισμένα σε αρχές κατανόησης της μουσικής, επιτρέποντάς μας να επικεντρωθούμε σε διαφορετικές πτυχές αναγνώρισης ακόρντων ξεχωριστά. Επιπλέον, ενσωματώνουμε τεχνικές μετα-επεξεργασίας εμπνευσμένες από τη θεωρία της μουσικής για την βελτίωση της ακρίβειας των προβλέψεών μας. Τα αποτελέσματα της προσέγγισής μας συγκρίνονται με άλλες λύσεις που καθιερώθηκαν από το MIREX, παρέχοντας μια σφαιρική αξιολόγηση της αποτελεσματικότητας και της απόδοσης της μεθόδου μας. Μέσω αυτής της έρευνας, στοχεύουμε να συμβάλουμε στις συνεχείς προσπάθειες για την προώθηση του πεδίου της ανάκτησης πληροφοριών μουσικής και τη βελτίωση της κατανόησης της εκτίμησης συγχορδιών.

2.1 Δομή της Εργασίας

Η εργασία αποτελείται από τα εξής κεφάλαια:

- Το Κεφάλαιο 7 επικεντρώνεται σε ορισμένες έννοιες που λειτουργούν ως θεωρητικό υπόβαθρο.
- Το Κεφάλαιο 8 αναλύει τα δεδομένα και τα βήματα προεπεξεργασίας που ακολουθήθηκαν.
- Το Κεφάλαιο 9 επικεντρώνεται στις πειραματικές εργασίες με διάφορες αρχιτεκτονικές μοντέλων και στην ανάλυση των αποτελεσμάτων.
- Το Κεφάλαιο 10 αναφέρεται στις τεχνικές μετεπεξεργασίας που ακολουθήθηκαν.
- Το Κεφάλαιο 11 περιλαμβάνει την αξιολόγηση του προτεινόμενου μοντέλου σε μετρικές του MIREX, μαζί με τη σύγκρισή του με άλλα προτεινόμενα μοντέλα και τα τελικά συμπεράσματα που προκύπτουν από τις εργασίες, μαζί με προτάσεις για μελλοντικές εργασίες.

Chapter 3

Δεδομένα & Προεπεξεργασία

Το κεφάλαιο αυτό αποτελεί περίληψη των σημαντικών στοιχείων του κεφαλαίου 8, στα ελληνικά. Σε αυτό το κεφάλαιο αναλύεται το σύνολο δεδομένων και οι μέθοδοι προεπεξεργασίας που ακολουθήθηκαν.

3.0.1 Σύνολο Δεδομένων Isophonics

Το σύνολο δεδομένων Isophonics είναι μια συλλογή από τραγούδια, σε μορφή ήχου, και μεταδεδομένων σχεδιασμένη για έρευνα στην ανάκτηση μουσικών πληροφοριών (MIR). Αποτελεί πολύτιμο πόρο για τη μελέτη διαφόρων πτυχών της μουσικής, όπως η μελωδία, η αρμονία, ο ρυθμός και η δομή. Για αυτήν τη μελέτη, χρησιμοποιήθηκε το σύνολο δεδομένων Isophonics που παρέχει 180 τραγούδια των Beatles. Οι σημάνσεις συγχορδιών αυτού του συνόλου δεδομένων έχουν ελεγχθεί αρκετές φορές από τον Christopher Harte [12] και την κοινότητα MIR, και μπορούν να χρησιμοποιηθούν με σιγουριά. Οι σημάνσεις συγχορδιών είναι αρχεία .lab. Αυτά είναι αρχεία κειμένου διαχωρισμένα με κενά, με τρεις στήλες που αντιστοιχούν στον χρόνο έναρξης, τον χρόνο λήξης και την ετικέτα συγχορδίας, αντίστοιχα. Ένα παράδειγμα φαίνεται στον πίνακα 8.1. Η μορφή των κομματιών ήταν στερεοφωνικά .mp3 με συχνότητα δειγματοληψίας 44kHz, οπότε ήταν απαραίτητη περαιτέρω επεξεργασία. Για να διασφαλιστεί ότι οι ετικέτες είναι συγχρονισμένες με τον ήχο, χρησιμοποιήθηκε το πρόγραμμα Audacity.

3.1 Προεπεξεργασία Δεδομένων

Για να προετοιμαστεί το ηχητικό δεδομένο για περαιτέρω ανάλυση, χρειάζονται ορισμένα βήματα. Η διαδικασία προεπεξεργασίας περιλαμβάνει τη μετατροπή των αρχείων ήχου από μορφή MP3 σε WAV και από στερεοφωνικό σε μονοφωνικό. Οι μορφές WAV και MP3 διαφέρουν στη συμπίεση και την ποιότητα του ήχου, με το WAV να είναι χωρίς συμπίεση για υψηλή πιστότητα και το MP3 με συμπίεση για μικρότερο μέγεθος. Το στερεοφωνικό χρησιμοποιεί δύο κανάλια για χωρικό βάθος, ενώ το μονοφωνικό χρησιμοποιεί ένα, ιδανικό για μη χωρικές εφαρμογές.

Αρχικά, σχεδιάστηκε ένας αλγόριθμος χρησιμοποιώντας το Python module Pydub που συγκεντρώνει μια λίστα με αρχεία MP3 μέσα σε έναν καθορισμένο κατάλογο αρχείων. Στη συνέχεια, για κάθε αρχείο, χρησιμοποιεί μεθόδους μετατροπής για να το μετατρέψει σε μορφή WAV και σε μονοφωνικό. Αυτή η συστηματική προσέγγιση εξασφαλίζει ότι τα ηχητικά

δεδομένα είναι προετοιμασμένα σε μια τυποποιημένη μορφή κατάλληλη για περαιτέρω ανάλυση. Ένα παράδειγμα ενός μετασχηματισμένου αρχείου μπορεί να οπτικοποιηθεί στην εικόνα 8.2.

Για τη διαδικασία του σήματος, χρησιμοποιήθηκαν δύο μετασχηματισμοί Fourier, ο Short Time Fourier Transform (STFT) και ο Constant-Q Chromagram. Ο STFT δημιούργησε υπερβολικό αριθμό ζωνών συχνοτήτων, αυξάνοντας τον αριθμό των εισόδων για το δίκτυο και δεν απέδωσε τα επιθυμητά αποτελέσματα.

Το χρωμόγραμμα, αποτελούμενο από 12 χαρακτηριστικά, προσφέρει λεπτομερή αναπαράσταση των μουσικών νοτών του ηχητικού σήματος, αλλά δεν παρέχει πληροφορίες για τη διανομή των νοτών στο φάσμα συχνοτήτων. Η Μετατροπή Constant Q (CQT) προτιμάται λόγω της λογαριθμικής κλίμακας συχνοτήτων που χρησιμοποιεί, η οποία ταιριάζει καλύτερα με την αντίληψη της ανθρώπινης ακοής.

Η προετοιμασία των ετικετών περιλαμβάνει τη μετατροπή από το χρονικό πεδίο στο πεδίο συχνοτήτων, διασφαλίζοντας την ευθυγράμμιση των ετικετών με τα αντίστοιχα φάσματα. Στον πίνακα 8.2, παρουσιάζονται οι ετικέτες στον χρονικό τομέα και στον πίνακα 8.3, στον τομέα συχνοτήτων.

Chapter 4

Πειράματα και Αποτελέσματα

Το κεφάλαιο αυτό αποτελεί περίληψη των σημαντικών στοιχείων του κεφαλαίου 9, στα ελληνικά. Το κεφάλαιο αυτό εξετάζει την ανάλυση, εκπαίδευση και σύγκριση διάφορων αρχιτεκτονικών μοντέλων για την αναγνώριση συγχορδιών. Μοντέλα όπως το 1D CNN και το 2D CNN που χρησιμοποιούν BiLSTM θα αναλυθούν.

4.1 Μοντέλο 1: Απλό 1D Συνελικτικό Νευρωνικό Δίκτυο

Η αρχική διερεύνηση ξεκινά με την εφαρμογή ενός απλού 1D Συνελικτικού Νευρωνικού Δικτύου (CNN). Αυτή η προσέγγιση λειτουργεί ως σημείο αναφοράς για την αξιολόγηση των επιδόσεων των επόμενων μοντέλων. Κάθε συγχορδία γίνεται "embed" ξεχωριστά, χωρίς να λαμβάνεται υπόψη η θεωρία της μουσικής και οι σχέσεις των συγχορδιών, γεγονός που μπορεί να περιορίσει τις προβλεπτικές ικανότητες του μοντέλου.

Για αυτή την αρχιτεκτονική, χρησιμοποιήθηκε το Constant-Q Chromagram ως βήμα προεπεξεργασίας, με αποτέλεσμα να προκύψουν 12 χαρακτηριστικά. Η αρχιτεκτονική του μοντέλου περιγράφεται στον πίνακα 9.1 και στο σχήμα 9.1.

Κατά τη διάρκεια της εκπαίδευσης, το σύνολο δεδομένων χωρίστηκε σε δύο υποσύνολα: ένα σύνολο εκπαίδευσης που αποτελείται από το 80% των δεδομένων και ένα σύνολο επικύρωσης που αποτελείται από το 20% των δεδομένων. Η διάσπαση πραγματοποιήθηκε ανά κομμάτι, διατηρώντας την ακεραιότητα της μουσικής δομής κάθε κομματιού.

Το δίκτυο αποτελείται από διάφορα επίπεδα, ξεκινώντας με ένα 1D συνελικτικό επίπεδο με 32 φίλτρα και μέγεθος πυρήνα 3, χρησιμοποιώντας τη συνάρτηση ενεργοποίησης ReLU. Ένα επίπεδο μέγιστης συγκέντρωσης (Max Pooling) εφαρμόζεται για να μειώσει τις διαστάσεις των χαρακτηριστικών. Το επίπεδο flatten αναδιαμορφώνει την έξοδο από το προηγούμενο επίπεδο σε έναν μονοδιάστατο πίνακα, διευκολύνοντας τη συμβατότητα με τα πλήρως συνδεδεμένα επίπεδα. Ένα πλήρως συνδεδεμένο επίπεδο με 128 νευρώνες και ReLU χρησιμοποιείται για την εξαγωγή χαρακτηριστικών, ενώ το επίπεδο εξόδου αποτελείται από 1552 μονάδες με συνάρτηση ενεργοποίησης softmax.

Για την εκπαίδευση του μοντέλου, χρησιμοποιήθηκε μέγεθος παρτίδας (batch size) 32 με ρυθμό εκμάθησης 0.0001 χρησιμοποιώντας τον Adam optimizer και την ενεργοποίηση ReLU. Η συνάρτηση απώλειας που χρησιμοποιήθηκε είναι η Sparse Categorical Cross-entropy. Τα αποτελέσματα της διαδικασίας εκπαίδευσης και αξιολόγησης φαίνονται στο διάγραμμα 9.2. Η ακρίβεια κυμαίνεται γύρω στο 42% τόσο για το σύνολο εκπαίδευσης όσο

και για το σύνολο αξιολόγησης.

Για λόγους σύγκρισης, το μοντέλο εκπαιδεύτηκε και δοκιμάστηκε επίσης στην ταυτοποίηση μόνο της ρίζας της συγχορδίας. Η απλότητα αυτής της αρχιτεκτονικής επιτρέπει μια σαφή αξιολόγηση της απόδοσης του μοντέλου στην απλή εργασία της ταυτοποίησης της ρίζας, λειτουργώντας ως βασικό σημείο αναφοράς για τη σύγκριση πιο σύνθετων μοντέλων. Η διαφορά στην αρχιτεκτονική είναι ότι το επίπεδο εξόδου τώρα αποτελείται μόνο από 13 νευρώνες, κατα αντιστοιχία με τις 13 ρίζες (συμπεριλαμβανομένης και της μη ύπαρξης ρίζας). Τα αποτελέσματα της διαδικασίας εκπαίδευσης και αξιολόγησης φαίνονται στο σχήμα 9.3. Επιτεύχθηκε ακρίβεια σχεδόν 60%.

Στον πίνακα σύγχυσης που φαίνεται στο Σχήμα 9.4, παρατηρούμε την κατανομή των προβλεπόμενων ριζών έναντι των πραγματικών ριζών. Παρά την απλότητα της προσέγγισης, το μοντέλο είναι ικανό να ταυτοποιεί με ακρίβεια τη σωστή ρίζα σε λογικό βαθμό.

4.2 Μοντέλο 2: 1D Συνελικτικό Νευρωνικό Δίκτυο

Συνεχίζοντας την εξερεύνηση των αρχιτεκτονικών των μοντέλων, το επόμενο μοντέλο που εξετάζεται είναι ένα πιο περίπλοκο 1D Συνελικτικό Νευρωνικό Δίκτυο (CNN). Αυτή η αρχιτεκτονική στοχεύει να εκμεταλλευτεί τα ιεραρχικά χαρακτηριστικά που μαθαίνονται από διαδοχικά συνελικτικά επίπεδα για να καταγράψει πιο σύνθετα πρότυπα στα δεδομένα.

Μετά από πειραματισμό με το Constant-Q Chromagram και το Constant Q Transform (CQT), διαπιστώθηκε ότι η μέθοδος CQT υπερέβη τη μέθοδο Constant-Q Chromagram. Η μέθοδος CQT, χρησιμοποιώντας 192 χαρακτηριστικά σε σύγκριση με τα 12 χαρακτηριστικά που χρησιμοποιήθηκαν προηγουμένως, οδήγησε σε καλύτερες επιδόσεις, όπως αναλύθηκε στο Κεφάλαιο 8. Έτσι, σε αυτή την ενότητα θα χρησιμοποιηθεί το Constant Q Transform με ρυθμό δειγματοληψίας 44100, 192 bins (24 bins ανά οκτάβα) και μήκος hop 4096.

Η αρχιτεκτονική του μοντέλου αποτελείται από διάφορα συνελικτικά επίπεδα ακολουθούμενα από επίπεδα μέγιστης συγκέντρωσης (Max Pooling) για μείωση των διαστάσεων των χαρακτηριστικών, επίπεδο flatten για αναδιάταξη των δεδομένων και ένα πλήρως συνδεδεμένο επίπεδο για εξαγωγή χαρακτηριστικών, πριν φτάσει στο επίπεδο εξόδου που χρησιμοποιεί τη softmax για πολυκατηγοριακή ταξινόμηση.

Για την εκπαίδευση του μοντέλου, χρησιμοποιήθηκαν οι ίδιοι υπερπαράμετροι με το προηγούμενο μοντέλο: μέγεθος παρτίδας (batch size) 32, ρυθμός εκμάθησης 0.0001 και η συνάρτηση απώλειας Sparse Categorical Cross-entropy. Τα αποτελέσματα της διαδικασίας εκπαίδευσης και αξιολόγησης παρουσιάζονται στο σχήμα 9.7, δείχνοντας μια σημαντική βελτίωση στην ακρίβεια συγκριτικά με το προηγούμενο μοντέλο. Η ακρίβεια του εκπαιδευτικού συνόλου έφτασε το 72% και του συνόλου επικύρωσης το 68%.

Για την ανάλυση της απόδοσης του μοντέλου στη ταυτοποίηση της ρίζας, το επίπεδο εξόδου τροποποιήθηκε ώστε να περιλαμβάνει μόνο 13 νευρώνες, αντιπροσωπεύοντας τις δυνατές ρίζες. Το μοντέλο κατάφερε να επιτύχει ακρίβεια 85% στη ταυτοποίηση της ρίζας.

Η προσέγγιση του 1D Συνελικτικού Νευρωνικού Δικτύου δείχνει ότι η χρήση πιο σύνθετων μοντέλων και περισσότερων χαρακτηριστικών από την προεπεξεργασία των δεδομένων μπορεί να βελτιώσει σημαντικά την ακρίβεια της πρόβλεψης.

Το μοντέλο στη συνέχεια εκπαιδεύτηκε και αξιολογήθηκε στην εύρεση όλων των μερών των συγχορδιών, όπως περιγράφεται αναλυτικά στο Κεφάλαιο 9.

4.3 Μοντέλο 3: Δισδιάστατο Συνελικτικό Νευρωνικό Δίκτυο με Επαναλαμβανόμενο Δίκτυο LSTM

Όπως περιγράφεται στο 7, τα συνελικτικά δίκτυα εξειδικεύονται στην αντιμετώπιση δεδομένων που έχουν χωρικές σχέσεις. Τα φασματογράμματα, στο προηγούμενο μοντέλο, ερμηνεύθηκαν ως χρονοσειρές διανυσμάτων με 192 χαρακτηριστικά. Σε αυτό το μοντέλο, τα φασματογράμματα θα αντιμετωπίζονται ως εικόνες, εκμεταλλευόμενα τα συνελικτικά επίπεδα πριν από το αναδρομικό επίπεδο για τη διευκόλυνση της εξαγωγής χαρακτηριστικών.

Για να το επιτύχουμε αυτό, όπως αναφέρεται στο 8, θα χρησιμοποιηθεί η τεχνική του 'chunking' ως ένα βήμα προεπεξεργασίας για να προετοιμαστούν τα δεδομένα για τη χρήση σε δισδιάστατα μοντέλα βαθιάς μάθησης. Αυτή η διαδικασία περιλαμβάνει τη διαίρεση των εισαγόμενων δεδομένων σε μικρότερα τμήματα, κάθε ένα αποτελούμενο από 100 συνεχόμενα χρονικά βήματα (frames), που αντιστοιχούν περίπου σε 7 δευτερόλεπτα ήχου. Αυτός ο αριθμός θεωρήθηκε ως υπερπαράμετρος κατά την εκπαίδευση των μοντέλων και καθορίστηκε μετά από λεπτούς ρυθμίσεις. Η λογική πίσω από αυτήν την προσέγγιση προέρχεται από τη χρήση ενός BiLSTM layer στην αρχιτεκτονική μας.

Η χρήση μικρών κομματιών δεδομένων είναι κρίσιμη για τη βελτίωση της αποτελεσματικότητας και της αποδοτικότητας του LSTM. Διασπώντας τη χρονοσειρά εισόδου σε διαχειρίσιμα τμήματα, το αναδρομικό επίπεδο LSTM μπορεί να κατανοήσει καλύτερα τις χρονικές εξαρτήσεις εντός των δεδομένων. Αυτή η λεπτομέρεια επιτρέπει στο μοντέλο να μάθει αποτελεσματικά μοτίβα και σχέσεις σε μικρότερα χρονικά διαστήματα, ενθαρρύνοντας πιο ακριβείς προβλέψεις και βελτιωμένη απόδοση.

Με την αντιμετώπιση κάθε τμήματος ως μια μοναδική εικόνα, οι συνελικτικοί στρώσεις μπορούν να εξάγουν σημαντικά χαρακτηριστικά εντός αυτών των μικρότερων χρονικών περιοχών του κομματιού. Αφού εξαχθούν τα χαρακτηριστικά από αυτά τα τμήματα χρησιμοποιώντας τις συνελικτικές στρώσεις, το επαναλαμβανόμενο δίκτυο LSTM θα πάρει αυτά τα επεξεργασμένα χαρακτηριστικά και θα αναλύσει τις ακολουθιακές εξαρτήσεις ανάμεσα στα τμήματα, παρέχοντας μια κατανομή της μουσικής σύνθεσης με την πάροδο του χρόνου.

Οι συνελικτικές στρώσεις σε αυτό το μοντέλο έχουν σχεδιαστεί για να αντιστοιχίσουν τα δεδομένα του φασματογράμματος σε ένα διάνυσμα εξόδου. Αυτό το διάνυσμα χρησιμοποιείται στη συνέχεια ως είσοδος για το επαναλαμβανόμενο επίπεδο LSTM. Αυτή η αρχιτεκτονική επιτρέπει στο LSTM να κατανοήσει αποτελεσματικά τις πληροφορίες περί πλαισίου με την πάροδο του χρόνου.

Μετά τα εκτεταμένα πειράματα που αναφέρονται στο 9.2, καθορίστηκε ότι η μέθοδος Constant-Q Transform (CQT) επετεύχθη καλύτερη απόδοση σε σχέση με την προσέγγιση Constant-Q Chromagram. Με τη χρήση 192 χαρακτηριστικών αντί των 12 που χρησιμοποιήθηκαν στο Constant-Q Chromagram, η μέθοδος CQT επέτυχε καλύτερη απόδοση. Συνεπώς, σε αυτό το κεφάλαιο, θα χρησιμοποιήσουμε το Constant-Q Transform με ρυθμό δειγματοληψίας 44100 Hz, 192 bins και μήκος hop 4096, παραμέτρους που ρυθμίστηκαν

για βέλτιστη απόδοση. Το μέγεθος του κομματιού ρυθμίστηκε και τέθηκε στα 100, που σημαίνει ότι κάθε κομμάτι καλύπτει περίπου 7 δευτερόλεπτα ήχου.

Η αρχιτεκτονική του Μοντέλου 3 αποτελείται από αρκετές συνελικτικές στρώσεις ακολουθούμενες από στρώσεις μέγιστης συμπίεσης, οι οποίες μειώνουν τις χωρικές τους διαστάσεις. Για την πρόληψη του overfit, περιλαμβάνονται στρώσεις dropout, οι οποίες απορρίπτουν τυχαία ένα τμήμα των μονάδων εισόδου κατά την εκπαίδευση. Η έξοδος από τις συνελικτικές στρώσεις γίνεται flatten και περνά από το επαναλαμβανόμενο επίπεδο BiLSTM, το οποίο επεξεργάζεται περαιτέρω τα εξαγόμενα χαρακτηριστικά πριν περάσει τελικά σε ένα πλήρως συνδεδεμένο πυκνό επίπεδο για την κατηγοριοποίηση.

4.3.1 Διαίρεση Δεδομένων Εκπαίδευσης, Αξιολόγησης και Δοκιμής

Το σύνολο δεδομένων, προσαυξημένο και προεπεξεργασμένο όπως περιγράφεται στο Κεφάλαιο 8, διαιρέθηκε σε τρία διακριτά σύνολα για τη διευκόλυνση των δοκιμών και της αξιολόγησης του μοντέλου. Για τη διατήρηση της συνοχής των δεδομένων εντός κάθε κομματιού, σχεδιάστηκε ένας αλγόριθμος για τη διαίρεση του συνόλου δεδομένων με βάση το κομμάτι. Ειδικότερα, τα κομμάτια από τα άλμπουμ CD1, CD2, Help, και Please Please Me κρατήθηκαν αποκλειστικά για τελική δοκιμή. Επιπλέον, το 15% των υπολοίπων δεδομένων ανατέθηκε για αξιολόγηση, ενώ το υπόλοιπο 85% ανατέθηκε για σκοπούς εκπαίδευσης. Αυτή η προσέγγιση διασφάλισε μια συνεκτική αξιολόγηση διατηρώντας την ακεραιότητα της δομής του συνόλου δεδομένων.

4.3.2 Κατηγοριοποίησης της Ρίζας των συγχορδιών με το Μοντέλο 3

Αυτή η εργασία περιλαμβάνει την κατηγοριοποίηση της ρίζας κάθε ακόρντου σε μία από τις 13 δυνατές ρίζες (συμπεριλαμβάνεται και η μη ύπαρξη νότας) στη μουσική κλίμακα. Για την επίτευξη αυτού, δοκιμάστηκαν αρκετές αρχιτεκτονικές, αλλά αυτή που παράγει τα καλύτερα αποτελέσματα φαίνεται στον παρακάτω πίνακα 9.8.

Η απόδοση του μοντέλου στην κατηγοριοποίηση των ριζών φαίνεται στο γράφημα που φαίνεται στο Σχήμα 9.18. Το μοντέλο εκπαιδεύτηκε για 40 εποχές μετά από πειραματισμούς με άλλες επιλογές. Τόσο η ακρίβεια στα δεδομένα εκπαίδευσης όσο και στα αξιολόγησης σταθεροποιείται σε περίπου 93% και 86% αντίστοιχα, ένα πολύ καλύτερο αποτέλεσμα σε σύγκριση με το Μοντέλο 2.

Το μοντέλο στη συνέχεια εκπαιδεύτηκε και αξιολογήθηκε στην εύρεση όλων των μερών των συγχορδιών, όπως περιγράφεται αναλυτικά στο Κεφάλαιο 9.

4.4 Σύνοψη των Αποτελεσμάτων

Σε αυτό το κεφάλαιο εξερευνήσαμε τρεις διαφορετικές προσεγγίσεις αρχιτεκτονικής για το πρόβλημα της αναγνώρισης των ακόρντων. Το αρχικό μοντέλο χρησιμοποίησε ένα απλό 1D Συνελικτικό Νευρωνικό Δίκτυο (CNN) ως βάση για αξιολόγηση, επικεντρώνοντας σε μεμονωμένα embeddings ακόρντων χωρίς να λαμβάνει υπόψη τη θεωρία της μουσικής και τις σχέσεις των ακόρντων. Ωστόσο, αυτή η προσέγγιση μπορεί να έχει περιορισμένες δυνατότητες πρόβλεψης λόγω της αγνόησης της μουσικής θεωρίας και των σχέσεων των ακόρντων.

Το επόμενο μοντέλο χρησιμοποίησε μια πιο πολύπλοκη αρχιτεκτονική 1D CNN για να αναλύσει περισσότερα περίπλοκα μοτίβα εντός των δεδομένων ακόρντων. Μέσω πειραματισμού, καθορίστηκε ότι η χρήση της μεθόδου Constant Q Transform (CQT) με 192 χαρακτηριστικά υπερτερεί της προσέγγισης Constant-Q Chromagram. Αυτό το μοντέλο περιλάμβανε συνελικτικές στρώσεις, στρώσεις Max Pooling και στρώσεις dropout για την πρόληψη του overfitting. Επιπλέον, εφαρμόστηκαν τεχνικές μεταφοράς μάθησης για να αντιμετωπίσουν αποτελεσματικά εργασίες όπως η αναγνώριση της ρίζας, η βασική νότα, η τριάδα και οι επεκτάσεις, ενισχύοντας την απόδοση του μοντέλου σε διάφορα στοιχεία της αναγνώρισης ακόρντων.

Στο τελικό μοντέλο, χρησιμοποιήθηκε ένα 2D Συνελικτικό Νευρωνικό Δίκτυο με επαναλαμβανόμενα επίπεδα Bidirectional Long Short-Term Memory (BiLSTM). Αυτό το μοντέλο είχε ως είσοδο τα φασματογράμματα ως εικόνες, διευκολύνοντας την εξαγωγή σημαντικών χαρακτηριστικών σε μικρότερα χρονικά διαστήματα χρησιμοποιώντας συνελικτικές στρώσεις. Η διαδικασία του "chunking", όπως αναλύεται στο 8, χρησιμοποιήθηκε ως ένα βήμα προεπεξεργασίας για να χωρίσει τα δεδομένα εισόδου σε διαχειρίσιμα τμήματα, βελτιώνοντας την ικανότητα του LSTM να αιχμαλωτίζει χρονικές εξαρτήσεις. Όπως και στο προηγούμενο μοντέλο, η μέθοδος Constant Q Transform προτιμήθηκε έναντι της προσέγγισης Constant-Q Chromagram, και εφαρμόστηκαν τεχνικές μεταφοράς μάθησης για να βελτιώσουν την απόδοση σε διάφορες εργασίες αναγνώρισης ακόρντων.

Όλα τα προαναφερθέντα μοντέλα εκπαιδεύτηκαν, δοκιμάστηκαν και αξιολογήθηκαν χρησιμοποιώντας μια μονάδα επεξεργασίας γραφικών (GPU), ειδικότερα μια κάρτα γραφικών NVIDIA GeForce GTX 1650 SUPER με 4 GB μνήμης. Λόγω της περιορισμένης μνήμης, εφαρμόστηκαν προσαρμοσμένοι generators πακέτων αντί της χρήσης των προκωδικοποιημένων υλοποιήσεων που είναι διαθέσιμες στο πακέτο TensorFlow. Είναι ενδιαφέρον να σημειωθούν οι χρόνοι εκπαίδευσης για κάθε μοντέλο. Οι χρόνοι εκπαίδευσης, μετρημένοι σε δευτερόλεπτα, φαίνονται στο Σχήμα 9.34. Ο χρόνος εκπαίδευσης του Μοντέλου 2 για τις εργασίες κατηγοριοποίησης Ρίζας και Βάσης είναι συγκρίσιμος με αυτόν του Μοντέλου 3. Ωστόσο, για τις υπόλοιπες τρεις εργασίες, το Μοντέλο 3 απαιτεί σημαντικά περισσότερο χρόνο λόγω της αυξημένης πολυπλοκότητας των επιπλέον στρωμάτων του.

Chapter 5

Μετά-Επεξεργασία Αποτελεσμάτων

Το κεφάλαιο αυτό αποτελεί περίληψη των σημαντικών στοιχείων του κεφαλαίου 10, στα ελληνικά. Στο κεφάλαιο αυτό, προχωράμε σε μια εκτενή ανάλυση διαφόρων τεχνικών μετεπεξεργασίας που στοχεύουν στην τελειοποίηση των προβλέψεων συγχορδιών που προκύπτουν από τα μοντέλα μας. Αυτές οι τεχνικές περιγράφονται λεπτομερώς για πλήρη κατανόηση και εφαρμογή.

Αρχικά, ξεκινάμε με τον συνδιασμό των προβλέψεων των επιμέρους στοιχείων των συγχορδιών, ώστε να πάρουμε τα τελικά δεδομένα πρόβλεψης, κάθε ένα από τα οποία αντιστοιχεί σε ένα συγκεκριμένο συγχορδιακό στοιχείο, όπως η ρίζα, το μπάσο, η τριάδα και οι επεκτάσεις. Αυτά τα μεμονωμένα στοιχεία συγχωνεύονται στη συνέχεια σε μια πρόβλεψη, τη συγχορδία, που αποτελεί το τελικό αποτέλεσμα.

Στη συνέχεια, εξετάζουμε τις τεχνικές μετεπεξεργασίας που σχεδιάστηκαν ειδικά για τη βελτίωση των προβλέψεων των συγχορδιών. Μία από αυτές περιλαμβάνει το φιλτράρισμα κάθε συγχορδίας βάσει προκαθορισμένων κανόνων που θα δούμε λεπτομερώς.

Επιπλέον, παρουσιάζουμε έναν μηχανισμό ομαλοποίησης σχεδιασμένο για να ενισχύσει τη συνοχή και τη συνέπεια των προβλέψεων συγχορδιών. Αυτή η διαδικασία αντικαθιστά τιμές εντός ενός καθορισμένου παραθύρου με την πιο συνηθισμένη τιμή στην περιοχή, εξαλείφοντας απότομες διακυμάνσεις και εξασφαλίζοντας ομαλότερες μεταβάσεις μεταξύ των συγχορδιών.

Επίσης, πειραματιζόμαστε με την εφαρμογή τεχνικών ομαλοποίησης τόσο στις συγχορδίες που συναρμόζουν όσο και στα μεμονωμένα στοιχεία των συγχορδιών. Παρατηρήθηκε ότι η ομαλοποίηση κάθε συγκεκριμένου στοιχείου της συγχορδίας έδωσε καλύτερα αποτελέσματα σε σύγκριση με την ομαλοποίηση των συνολικών συγχορδιών. Αυτή η προσέγγιση αποδεικνύεται αποτελεσματική, καθώς λαμβάνει υπόψη τα διακριτά χαρακτηριστικά και τη δυναμική του κάθε συγχορδιακού στοιχείου στο στάδιο της μετεπεξεργασίας.

Τέλος, εξετάζουμε τις μεθόδους φιλτραρίσματος που εφαρμόζονται στα δεδομένα συγχορδιών μετά την ομαλοποίηση. Η κύρια εστίαση εδώ είναι στους κανόνες που χρησιμοποιούνται για τον εντοπισμό και το φιλτράρισμα συγχορδιών βάσει συγκεκριμένων κριτηρίων. Η συνδυασμένη εφαρμογή των τεχνικών μετεπεξεργασίας αποδεικνύεται ως ισχυρή προσέγγιση που ενισχύει σημαντικά την απόδοση και την ακρίβεια του συστήματος αναγνώρισης συγχορδιών μας.

Chapter 6

Αποτελέσματα και Προτάσεις για Μελλοντική Έρευνα

6.1 Αποτελέσματα

Το κεφάλαιο αυτό αποτελεί περίληψη των σημαντικών στοιχείων του κεφαλαίου 11, στα ελληνικά. Σε αυτό το κεφάλαιο, αξιολογούμε το σύστημα αναγνώρισης συγχορδιών μας χρησιμοποιώντας τις μετρικές του MIREX (Music Information Retrieval Evaluation eXchange). Το MIREX παρέχει ένα προτυποποιημένο πλαίσιο για την αξιολόγηση των συστημάτων ανάκτησης πληροφοριών μουσικής, καθιστώντας το ένα ιδανικό μέτρο σύγκρισης των αποτελεσμάτων μας με αυτά από άλλες μελέτες. Ξεκινώντας από την εξήγηση των μετρικών αξιολόγησης του MIREX (λεπτομερείς εξηγήσεις στο Κεφάλαιο 7) και τη σημασία τους στην αναγνώριση συγχορδιών, διασφαλίζουμε ότι η αξιολόγηση μας είναι τόσο πλήρης όσο και συγκρίσιμη με άλλα κορυφαία συστήματα.

Στο πλαίσιο της αξιολόγησης του MIREX, είναι ενδιαφέρον να συγκρίνουμε τη λύση μας με άλλα άρθρα. Οι Gasser και Strasser [13], στην υποβολή τους, ακολούθησαν μια παρόμοια προσέγγιση χωρίζοντας τα ακκόρντα σε συστατικά και στη συνέχεια χρησιμοποίησαν μια αρχιτεκτονική CNN εφαρμόζοντας Μεταφορά Μάθησης για κάθε συστατικό. Η κύρια διαφορά είναι η χρήση επιπέδων Bi LSTM στην περίπτωσή μας και οι τεχνικές μετεπεξεργασίας.

Οι Park, Choi κ.ά. [14] χρησιμοποιούν έναν μηχανισμό αυτο-προσοχής (self-attention) για την αναγνώριση ακκόρντων προκειμένου να επικεντρωθούν σε συγκεκριμένες περιοχές των ακκόρντων. Η εκπαίδευση του προτεινόμενου Μοντέλου Bi-directional Transformer για την αναγνώριση ακκόρντων (BTC) αποτελείται από μια μόνο φάση, ενώ εμφανίζει ανταγωνιστική απόδοση. Προέκυψε ότι το μοντέλο μπόρεσε να διαχωρίσει τμήματα των ακκόρντων χρησιμοποιώντας το προσαρμοστικό πεδίο αλληλεπίδρασης του μηχανισμού προσοχής. Επιπλέον, παρατηρήθηκε ότι το μοντέλο μπόρεσε αποτελεσματικά να αντιληφθεί μακροπρόθεσμες εξαρτήσεις, χρησιμοποιώντας τις ουσιώδεις πληροφορίες ανεξαρτήτως απόστασης.

Τέλος, οι Jiang, Ke Chen κ.ά. [15] προτείνουν ένα νέο μοντέλο για πρακτικές εργασίες μεταγραφής ακκόρντων. Η κύρια έννοια του νέου μοντέλου είναι να αναπαραστήσει οποιοδήποτε ετικέτα ακκόρντου με ένα σύνολο υπομονάδων (δηλ. ρίζα, τριάδα, βάση) σύμφωνα με τις κοινές μουσικές δομές τους. Ένας multitask classifier εκπαιδεύεται στη συνέχεια να αναγνωρίζει όλες τις υπομονάδες δεδομένης της χαρακτηριστικής περιγραφής

του ήχου, και στη συνέχεια οι ετικέτες των μεμονωμένων υπομονάδων επανασυναρμολογούνται για να σχηματίσουν την τελική ετικέτα ακκόρντου. Για την κατασκευή του πολυεργαστηριακού ταξινομητή χρησιμοποιείται ένα Αναδρομικό Συνελικτικό Νευρωνικό Δίκτυο (RCNN).

Εξετάζοντας τις μετρικές στον πίνακα 11.2, βλέπουμε ότι το μοντέλο μας *Model2+S+F* παρουσιάζει εξαιρετική απόδοση γενικά, με την ακρίβεια αναγνώρισης της έβδομης νότας να είναι ιδιαίτερα εντυπωσιακή. Αυτή η σημαντική επίτευξη υπογραμμίζει την ανθεκτικότητα του μοντέλου μας, το οποίο εξειδικεύεται στην αναγνώριση των λεπτομερειών που περιλαμβάνονται σε αυτές τις περίπλοκες μουσικές δομές ακκόρντων. Όχι μόνο το μοντέλο μας επιδεικνύει εξαιρετική επίδοση σε αυτό το συγκεκριμένο πρόβλημα, αλλά συνεχίζει να εμφανίζει ανταγωνιστική απόδοση και σε άλλες κατηγορίες αξιολόγησης MIREX. Αυτή η διαπίστωση υποστηρίζεται από αναλύσεις σύγκρισης με άλλα μοντέλα, όπως τα KBK2, BTC+CRF και JLCX1, τα οποία είναι όλα εξαιρετικά μοντέλα.

6.2 Προτάσεις για Μελλοντική 'Έρευνα

Αυτή η διπλωματική εργασία έχει δημιουργήσει ένα στέρεο έδαφος για συνεχή καινοτομία και έρευνα στον τομέα της αναγνώρισης συγχορδιών. Πολλές επιλογές προτείνονται για μελλοντική έρευνα, κάθε μία από τις οποίες υπόσχεται να διευρύνει τις δυνατότητες και την ακρίβεια των αλγορίθμων αναγνώρισης συγχορδιών.

6.2.1 Επέκταση των Πηγών Δεδομένων

Η περιορισμένη πρόσβαση σε αρχεία ήχου λόγω περιορισμών πνευματικών δικαιωμάτων εμπόδισε την απόκτηση ενός μεγαλύτερου συνόλου δεδομένων για εκπαίδευση και δοκιμή. Με την προσθήκη δεδομένων από ένα ευρύτερο φάσμα μουσικών στυλ, το αποτέλεσμα θα μπορούσε να προσφέρει μεγαλύτερη προσαρμοστικότητα και ακρίβεια στην αναγνώριση συγχορδιών, ειδικά στο πλαίσιο της σύγχρονης δυτικής pop μουσικής.

6.2.2 Ενσωμάτωση Αρχών Θεωρίας της Μουσικής

Η ενσωμάτωση αρχών της θεωρίας της μουσικής στους αλγορίθμους αναγνώρισης συγχορδιών θα μπορούσε να οδηγήσει σε πιο μουσικά ενημερωμένες προβλέψεις. Με την αξιοποίηση γνώσεων για προγρεσσιονικές σειρές συγχορδιών, αρμονικές λειτουργίες και μελωδικές τάσεις, ο μοντέλο θα μπορούσε να παράγει πιο συνακόλουθες και αισθητικά ικανοποιητικές αποτελέσματα.

6.2.3 Ενσωμάτωση Beat Tracking

Η καταγραφή ρυθμού, αν και ένα ξεχωριστό πρόβλημα ταξινόμησης στο MIREX, κρύβει τεράστιο δυναμικό ως πρόσθετη είσοδος για τα μοντέλα αναγνώρισης συγχορδιών. Με την συμπερίληψη δεδομένων καταγραφής ρυθμού ως δευτερεύουσα είσοδο, ο αλγόριθμος θα μπορούσε να διακρίνει καλύτερα τον χρόνο των αλλαγών συγχορδιών, οδηγώντας σε πιο ακριβή και συμπερασματικά αποτελέσματα. Αυτή η ενσωμάτωση θα οδηγήσει στην ανάπτυξη πιο δυναμικών και ρυθμικά ευαίσθητων συστημάτων αναγνώρισης συγχορδιών.

6.2.4 Αναγνώριση της Κλίμακας (key detection) ως βήμα Προ-Επεξεργασίας

Η ενσωμάτωση αλγορίθμων αναγνώρισης της κλίμακας ως ένα βήμα προ-επεξεργασίας θα μπορούσε να προσφέρει πολλά οφέλη στη βελτίωση της ακρίβειας της αναγνώρισης συγχορδιών. Απορρίπτοντας συγχορδίες που δεν περιλαμβάνονται στην κλίμακα, το σύστημα μπορεί να επικεντρωθεί στην ανάλυση και πρόβλεψη συγχορδιών που είναι αρμονικά συνεκτικές και μουσικά σημαντικές. Αυτή η προσέγγιση θα μπορούσε να διευκολύνει τη διαδικασία αναγνώρισης συγχορδιών και να βελτιώσει τη συνολική ποιότητα των προβλέψεων.

Chapter 7

Theoretical Background

The purpose of this section is to analyze certain theoretical concepts that will be mentioned across this work and play a major role in understanding the models or the problems they try to solve. It is also important to analyze basic music theory concepts that are relevant to the problem.

7.1 Basic Music Theory

In this chapter, we explore the fundamental concepts of music theory that underpin chord recognition, providing essential insights into the harmonic structures and relationships present in musical compositions.

7.1.1 Intervals

A popular harmonic representation in the audio and symbolic realms is the chord-label. Particularly in North America, there is a long history of Roman numeral-focused teaching that gives rise to the usage of chord labels in symbolic music (e.g., [16], [17]).

Approaches rooted in music theory and cognitive science, such as the methodologies proposed by Krumhansl [18] and Lerdahl [19], have influenced the development of computational distance metrics for chords (e.g., [20]-[21]. However, within the field of music information retrieval, much emphasis has been placed on chord recognition due to its straightforward translation into a classification task. Existing chord estimation metrics, as seen in current MIREX standards (refer to Table 1), primarily focus on predicting chord labels, with varying levels of simplification implemented to accommodate differences in chord vocabulary size [22].

It's essential now to analyze intervals to better grasp these concepts. Intervals are the building blocks of scales, chords (or harmonies), and melodies. Intervals are a measurement between two pitches, either vertically or horizontally. When measuring vertically, we refer to harmonic intervals because the two notes sound simultaneously. When measuring horizontally, we refer to melodic intervals because the notes occur one after the other [1].

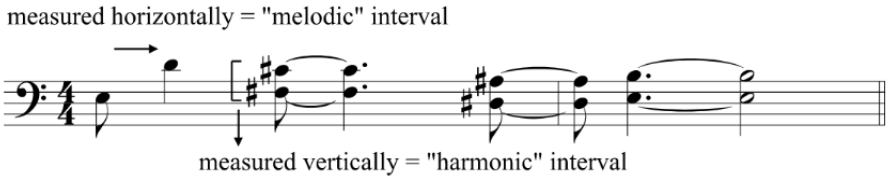


Figure 7.1. *Difference between harmonic and melodic intervals [1].*

The following intervals can be found by measuring a major scale from the tonic up to each scale degree:

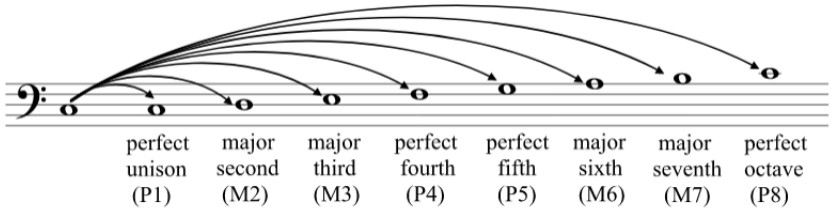


Figure 7.2. *A visualization of either "perfect" or "major" intervals [1]."*

To summarize an interval is a certain distance between 2 notes. They are divided in two categories: Melodic and Harmonic. In melodic intervals the notes are played one after the other, while in harmonic they are played together. The smallest musical interval frequently used in Western tonal music is the semitone, sometimes known as a half step or a half tone, which is also thought to be the most dissonant when sounded harmonically. Semitone counting alone cannot determine the names given here 7.1.

Table 7.1. *Intervals and their Properties*

Number of Semitones	Minor, Major, or Perfect Intervals	Augmented or Diminished Intervals
0	Perfect unison	Diminished second
1	Minor second	Augmented unison
2	Major second	Diminished third
3	Minor third	Augmented second
4	Major third	Diminished fourth
5	Perfect fourth	Augmented third
2*6	-	Diminished fifth
	-	Augmented fourth
7	Perfect fifth	Diminished sixth
8	Minor sixth	Augmented fifth
9	Major sixth	Diminished seventh
10	Minor seventh	Augmented sixth
11	Major seventh	Diminished octave
12	Perfect octave	Augmented seventh

7.1.2 Introduction to Chords

Chords form the backbone of music, providing harmony and depth to melodies. In essence, a chord is a combination of three or more notes played simultaneously. These notes are typically chosen from a specific scale and arranged in intervals, creating unique sounds and emotions. Chords serve as the building blocks of songs across various genres.

Splitting the chord structure into major components provides a systematic approach to understanding its composition and function. A chord can be dissected into several elements, each contributing to its overall sound and character. The primary components include the root, bass, triad, and extensions.

- The **Root** of a chord serves as its foundation, determining its fundamental pitch and tonal center. It is the note from which the chord derives its name and provides a sense of stability and resolution.
- The **Bass** note is the lowest sounding pitch in the chord. While it often corresponds to the root note, especially in basic chord structures, it can sometimes differ, providing additional depth and color to the chord.
- The **Triad** forms the core structure of many chords, consisting of three notes stacked in intervals of a third. These notes typically include the root, a third (either major or minor), and a fifth, creating the basic harmonic framework of the chord.

The simplest example of a chord is a Triad. A class of chords known as the triad is made up of three-note chords, or root, third, and fifth, expressed by the formula 1-3-5 [11]. They are made up of two successive thirds in this example 7.2.

Table 7.2. *Kinds of Triads (♭ represents "flat" or "lowered", ♯ represents "sharp" or "raised") [11].*

	Root	3rd	5th
Major	1	3	5
Minor	1	♭3	5
Augmented	1	3	♯5
Diminished	1	♭3	♭5

7.1.3 Chord Representation

Chord representation is essential in music theory as it allows us to understand the harmonic content of a piece of music. Chords can be represented in various ways, including chord symbols, chord diagrams, and chord charts.

One common method of chord representation is through chord symbols, which use letters and symbols to denote the root, quality, and extensions of a chord. For example, the chord symbol "Cmaj7" represents a C major seventh chord, consisting of the notes C, E, G, and B. Similarly, "Dm7" represents a D minor seventh chord, comprising the notes D, F, A, and C.

Every chord symbol in the table below has its matching intervals and the chord's whole name next to it 7.3.

Table 7.3. *Basic Chords and Their Representations*

Chord Symbol	Intervals	Full Name
C	1, 3, 5	C major
Cmaj7	1, 3, 5, 7	C major seventh
Dm7	1, ♭3, 5, ♭7	D minor seventh
E7	1, 3, 5, ♭7	E dominant seventh
Fm	1, ♭3, 5	F minor
Gsus4	1, 4, 5	G suspended fourth
Am9	1, 3, 5, 9	A minor ninth
Bm7♭5	1, ♭3, ♭5, ♭7	B half-diminished seventh

7.2 Machine Learning & Deep Learning

7.2.1 Machine Learning

Machine learning (ML) is a branch of artificial intelligence (AI) that enables computers to learn from data and make predictions or decisions without being explicitly programmed [23]. Its goal is to develop algorithms and models that can automatically detect patterns and insights in data and make predictions or decisions based on these patterns. There are three main types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is the most common type, involving training a model on a labeled dataset where the output variable or target is known. The model is then tested on new, unseen data to evaluate its performance. Examples of supervised learning include image classification, speech recognition, and natural language processing [24].

On the other hand, unsupervised learning involves training a model on an unlabeled dataset where the output variable or target is unknown. The model is then used to identify patterns and information in the data, such as clustering or dimensionality reduction. Examples of unsupervised learning include anomaly detection, market segmentation, and feature extraction.

Reinforcement learning is a type of machine learning that focuses on training models to make decisions based on feedback or rewards. It is often used in situations where the outcome of a decision is uncertain or the optimal decision may change over time. Examples of reinforcement learning include robotics, games, and control systems.

Gathering and cleansing data is usually the first step in the machine learning process. This is an important stage because the model's performance can be greatly impacted by the quality of the data. The data is separated into a training set and a test set once it has been cleaned and prepared. The test set is used to assess the model's performance, whereas the training set is used to train the model.

Choosing an appropriate model and method for the given task is the next step. Numerous models and techniques are at one's disposal, including support vector machines, random forests, decision trees, and neural networks. The kind of problem, the intricacy of the data, and the resources at hand all influence the model and algorithm selection. The model is chosen, and it is subsequently trained using the training set. Finding the model's optimal parameters to minimize error on the training set is the aim of training

and it is called fine tuning. Subsequently, the model's performance is evaluated on the test set to make sure it doesn't overfit the training set. Overfitting occurs when a machine learning model captures noise in the training data rather than the underlying pattern, resulting in poor generalization to new data [25].

The model can be used to forecast new, unseen data after it has been trained and evaluated. This is commonly known as deployment or inference. Applications for the model include natural language processing, speech recognition, and image classification.

Machine learning has several uses and has been implemented in a number of industries, including manufacturing, healthcare, finance, and transportation. For instance, machine learning has been applied to the healthcare industry to create models that assist in medical diagnosis, identify potential health hazards, and forecast patient outcomes. Machine learning has been used in finance to forecast stock values, find possible investment opportunities, and detect fraudulent transactions. Supply chain optimization, traffic prediction, and driverless cars are just a few of the transportation systems that have benefited from the application of machine learning. Machine learning has been applied in manufacturing to enhance quality control, forecast equipment breakdowns, and optimize production processes.

7.2.2 Deep Learning

Deep learning (DL) is a sub-field of machine learning that utilizes deep neural networks to model and solve complex problems. Originating in artificial intelligence, the idea of deep learning has become increasingly popular in recent years because it can deliver state-of-the-art performance on a variety of tasks, including speech recognition, image classification, and natural language processing [26].

Deep neural networks, also known as deep learning models, consist of multiple layers of interconnected artificial neurons, where each layer receives input from the previous layer and produces output for the next layer. The architecture of deep neural networks allows them to automatically learn and extract features from raw input data and perform highly complex nonlinear transformations. This contrasts with traditional machine learning models, which typically require manual feature engineering [27].

Several techniques can be used, depending on the application, to train deep learning models. The most popular technique for training deep learning models is supervised learning, in which the model is trained using a labeled dataset with the aim of minimizing the difference between the expected and actual outputs. Additional training techniques include reinforcement learning, in which the model learns to make decisions in response to rewards or feedback, and unsupervised learning, in which the model is trained on an unlabeled dataset.

Deep learning has also been used to generate cutting-edge results in a variety of applications, including audio recognition, image classification, and natural language processing. For instance, deep learning models have been applied to picture classification to automatically identify objects and scenes in photos and to reach performance that is on par with or even better than human ability. Deep learning models have been applied to

voice recognition in order to obtain performance that is on par with or even greater than that of conventional methods, including automatic speech transcription. Deep learning models have been applied to natural language processing to generate and understand language automatically, with results that are on par with or better than those of more conventional approaches.

Moreover, new methods and architectures are being created quickly in the rapidly evolving discipline of deep learning. More sophisticated and potent layouts, like transformer, recurrent, and convolutional neural networks, have been the subject of research in recent years. In order to enhance the performance of deep learning models on novel tasks and domains, new methods for training them—such as domain adaptation and transfer learning—have also been created.

In Conclusion, deep learning is a potent instrument with the ability to transform numerous industries and tackle a broad variety of issues. Deep learning does, however, have several drawbacks, much like machine learning, including the requirement for vast amounts of high-quality data, the complexity of the models, and the computational resources needed for training and refining deep learning models. Despite these difficulties, deep learning is anticipated to maintain its prominent position in the artificial intelligence space and to propel the creation of novel and fascinating applications.

7.3 Convolution Neural Networks (CNN)

Artificial Neural Networks (ANNs) are computational systems inspired by the biological nervous systems, such as the human brain. They consist of interconnected computational nodes, or neurons, which collectively learn from input data to optimize their final output [2].

The basic structure of an ANN can be illustrated as depicted in 7.3. Input data, usually in the form of a multidimensional vector, is fed into the input layer and propagated to the hidden layers. These hidden layers make decisions based on the previous layer's output, adjusting their internal parameters to improve the final output through a process known as learning. When multiple hidden layers are stacked upon each other, it is referred to as deep learning.

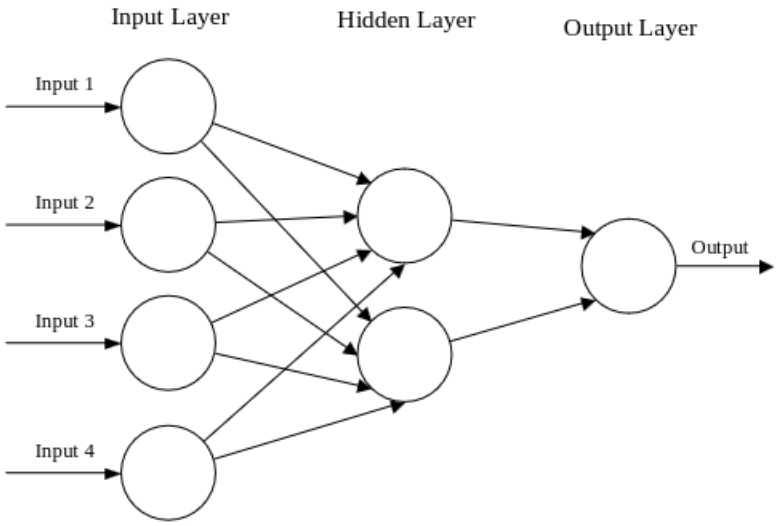


Figure 7.3. *A simple three layered feedforward neural network (FNN), comprised of a input layer, a hidden layer and an output layer. This structure is the basis of a number of common ANN architectures, included but not limited to Feed- forward Neural Networks (FNN), Restricted Boltzmann Machines (RBMs) and Recurrent Neural Networks (RNNs) [2].*

Similar to conventional ANNs, convolutional neural networks (CNNs) are made up of neurons that learn to optimize themselves. ANNs are based on the fact that each neuron will continue to receive an input and carry out an operation (such a scalar product followed by a non-linear function). The entire network will continue to express a single perceptive score function (the weight) from the raw picture vectors that are input to the class score that is generated at the end. The final layer will include loss functions linked to the classes; all of the standard advice and techniques created for conventional ANNs still holds true [2].

CNN has four layers: convolution layer, pooling layer, fully connected layer, and nonlinearity layer [3]. Illustrations of those four layers are presented in 7.4. Further explanations regarding the description of each layer will be shown in this chapter.

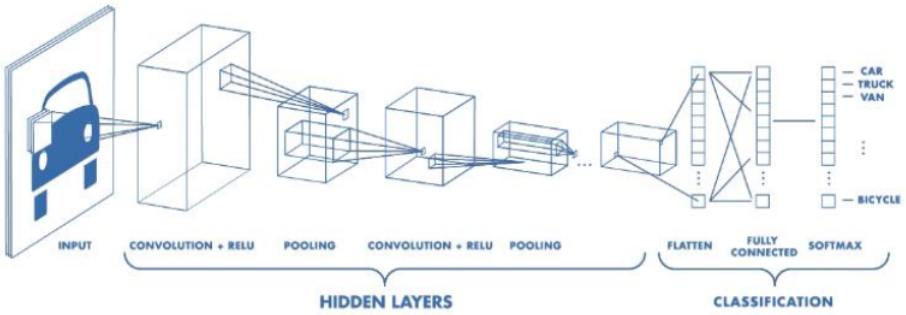


Figure 7.4. *CNN Architecture [3].*

7.3.1 Convolutional Layer

In order to determine the convolution of the input images and extract the essential information, the convolutional layer uses a kernel filter. Compared to the input image, the filter kernel's dimensions are the same, but its constant parameter value is less [28]. Filters are applied to the input data using a process known as convolution in Convolutional Neural Networks (CNNs), where the filter moves over the input data and the dot product between the filter and the input data is computed at each position. The result of the convolution is a activation map, which is a representation of the input data that is more abstract and compact than the original input data. The convolutional layer's general equation can be written as in 7.1. A straightforward representation of the CNN calculation process that yields the activation map can be shown in 7.5. A convolutional layer is characterized by its kernel size, stride length, and padding [29]. The kernel size refers to the dimensions of the filter or the moving kernel [30]. Stride length denotes the number of steps taken by kernels before computing product points and generating output pixels [31]. Padding represents the dimensions of the zero-padding applied around the input feature map [32].

$$\text{Activation map} = \text{Input} * \text{Filter} = \sum_{y=0}^{cols} \left(\sum_{x=0}^{rows} \text{Input}(x-p, y-q)\text{Filter}(x, y) \right) \tag{7.1}$$

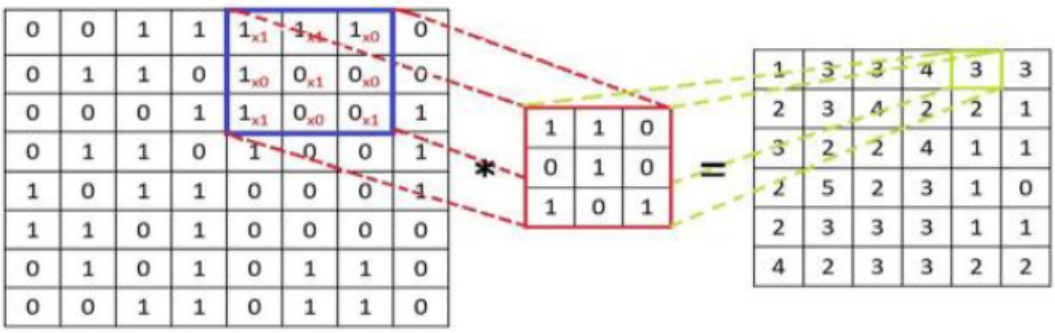


Figure 7.5. *Convolutional Layer [3].*

7.3.2 Pooling Layer

Pooling layers partition the input data into smaller sections known as pooling windows or receptive fields. Within each window, an aggregation operation is performed, like selecting the maximum or averaging the values. This process decreases the dimensions of the feature maps, creating a condensed portrayal of the input data. In 7.7 an example of max pooling is presented.

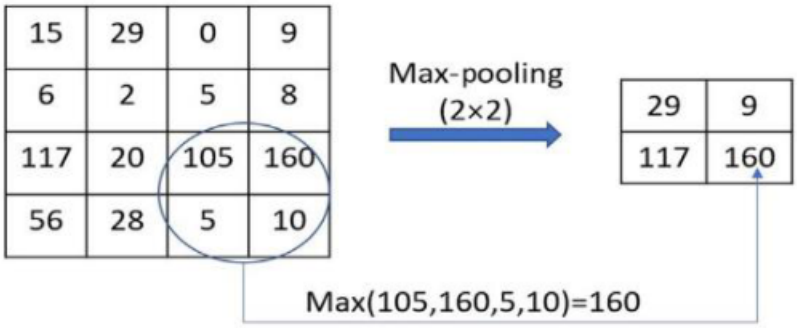


Figure 7.6. *Max Pooling [3].*

7.3.3 Fully Connected Layer

Typically, the final convolution or pooling layer's output feature maps are flattened, or converted into a one-dimensional array or vector of numbers [33]. After that, this array is connected to one or more fully connected layers, also called dense layers, in which every input and every output are coupled by a trainable weight. nce the features extracted by the convolution layers and downsampled by the pooling layers are created, they are mapped by a subset of fully connected layers to the final outputs of the network, such as the probabilities for each class in classification tasks. The final fully connected layer typically has an equal number of output nodes to classes. A nonlinear function like ReLU comes after each completely connected layer.

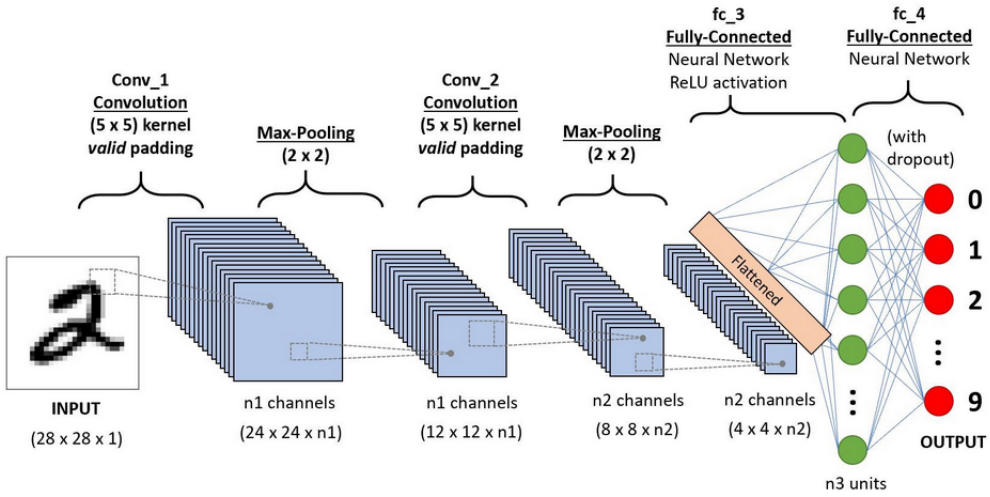


Figure 7.7. *CNN Architecture [4].*

7.3.4 Nonlinearity Layer (Activation Function)

Activation functions (AFs) are pivotal components within neural networks, as they facilitate the learning of abstract features through nonlinear transformations [34]. Several key properties characterize AFs:

- they introduce nonlinear curvature to the optimization landscape, enhancing the

network's training convergence.

- they should maintain a reasonable level of computational complexity within the model.

- they must not obstruct the gradient flow during training.

- they should preserve the distribution of data to facilitate more effective network training.

Two commonly used examples of activation functions are ReLU (Rectified Linear Unit) 7.2 and Sigmoid Function 7.3.

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{7.3}$$

7.4 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) constitute a specific type of neural network architecture primarily designed for recognizing patterns within sequential data. This data can encompass various forms such as handwriting, genetic sequences, text, or numerical time series, commonly encountered in industrial contexts like stock market data or sensor readings [35]. Furthermore, RNNs can also be adapted to process images by breaking them down into patches and treating them as sequential data . In broader applications, RNNs are utilized in tasks such as Language Modeling, Text Generation, Speech Recognition, Image Captioning, and Video Tagging [5].

What sets Recurrent Neural Networks apart from Feedforward Neural Networks, also known as Multi-Layer Perceptrons (MLPs), is the way information flows through the network. While Feedforward Networks transmit information linearly, RNNs incorporate cycles, allowing information to circulate back within the network. This characteristic enables RNNs to consider not only the current input but also past inputs, thus extending the capabilities of Feedforward Networks. This distinction is illustrated at a high level in Figure 1, where multiple hidden layers are represented by a single "Hidden Layer" block (H), which can be expanded to accommodate multiple layers as needed.

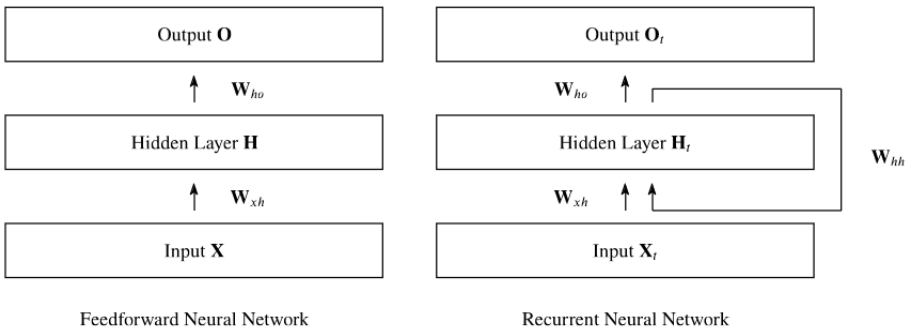


Figure 7.8. *Visualisation of differences between Feedfoward NNs und Recurrent NNs [5].*

At a high level, an RNN processes a sequence of inputs $x = x_1, x_2, \ldots, x_T$ and produces a sequence of outputs $y = y_1, y_2, \ldots, y_T$. The fundamental concept of an RNN is to utilize a hidden state vector h_t to capture information from previous inputs in the sequence and use this hidden state to compute the next output in the sequence. The hidden state is updated at each time step based on the current input and the previous hidden state: $h_t = f(x_t, h_{t-1})$, where f is a non-linear function mapping the input and the previous hidden state to a new hidden state. The output at each time step is then computed as a function of the current hidden state: $y_t = g(h_t)$, where g is a non-linear function mapping the hidden state to the output.

We can represent the process of transferring information from the previous iteration to the hidden layer using the mathematical notation introduced in [36]. To do so, we denote the hidden state and the input at time step t respectively as $H_t \in \mathbb{R}^{n \times h}$ and $X_t \in \mathbb{R}^{n \times d}$, where n is the number of samples, d is the number of inputs for each sample, and h is the number of hidden units. Additionally, we utilize a weight matrix $W_{xh} \in \mathbb{R}^{d \times h}$, a hidden-state-to-hidden-state matrix $W_{hh} \in \mathbb{R}^{h \times h}$, and a bias parameter $b_h \in \mathbb{R}^{1 \times h}$. Finally, all this information is passed through an activation function ϕ, typically a logistic sigmoid or tanh function, to prepare the gradients for use in backpropagation [5]. Combining all these notations results in 7.4 for the hidden variable and 7.5 for the output variable:

$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h) \tag{7.4}$$

Since H_t recursively includes H_{t-1} and this process occurs for every time step, the RNN incorporates traces of all hidden states that preceded H_{t-1} as well as H_{t-1} itself. This process can be represented by the following equation:

$$O_t = \phi_o(H_t W_{ho} + b_o) \tag{7.5}$$

One of the primary advantages of RNNs is their ability to handle input sequences of arbitrary lengths, as the hidden state captures information from all previous inputs in the sequence. However, a significant challenge in training RNNs is the problem of vanishing or exploding gradients, which occurs when the gradients used to update the network parameters become either very small or very large, making it difficult to learn long-term dependencies.

To address this issue, various variations of RNNs have been proposed, including the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures, which employ specialized gating mechanisms to control the flow of information within the network.

7.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks provide a solution to the vanishing error problem in neural networks. They can effectively learn to bridge time lags of more than 1,000 discrete time steps. This is achieved through the use of constant error carousels (CECs), which maintain a consistent flow of error within specialized cells [6]. Access to these cells is regulated by multiplicative gate units, which dynamically control when to grant access. The LSTM layout can be seen in 7.10.

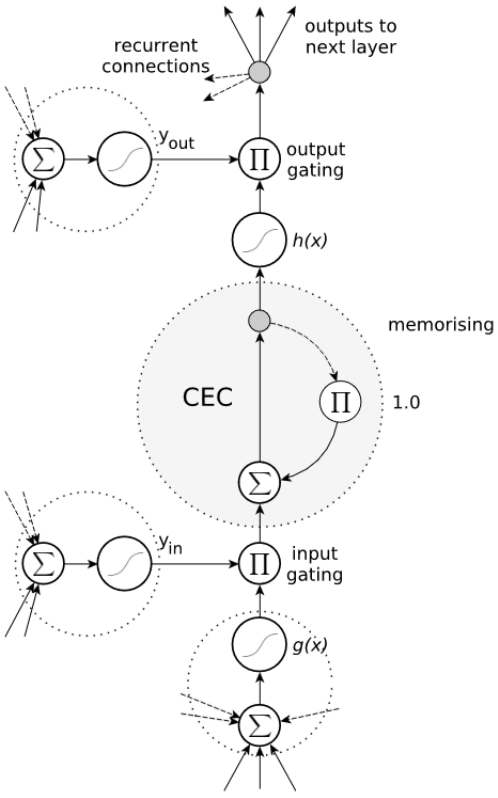


Figure 7.9. *A standard LSTM memory block consists of (at least) one cell with a recurrent self-connection (CEC) and a weight of '1'. The state of the cell is denoted as s_c. Read and write access is regulated by the input gate, y_{in}, and the output gate, y_{out}. The internal cell state is calculated by multiplying the result of the squashed input, g, by the result of the input gate, y_{in}, and then adding the state of the last time step, $s_c(t-1)$. Finally, the cell output is calculated by multiplying the cell state, s_c, by the activation of the output gate, y_{out} [6].*

7.5.1 Constant Error Carousel

In a simplified scenario where we have only one unit u with a self-connection, the local error backflow of u at a time-step τ is determined by 7.6:

$$\vartheta_u(\tau) = f'(u(z_u(\tau)))W[u,u]\vartheta_u(\tau+1) \tag{7.6}$$

To ensure a constant error flow through u, it is required that $f'(u(z_u(\tau)))W[u,u] = 1.0$. By integration, it follows that $f_u(z_u(\tau)) = z_u(\tau)W[u,u]$. This implies that f_u must be linear, and u's activation must remain constant over time, i.e., $y_u(\tau+1) = f_u(z_u(\tau+1)) = f_u(y_u(\tau)W[u,u]) = y_u(\tau)$. This is achieved by using the identity function $f_u = id$ and setting $W[u,u] = 1.0$. This preservation of error is termed the constant error carousel (CEC), a pivotal aspect of LSTM enabling short-term memory storage for extended durations [6].

7.5.2 Memory Blocks

While the CEC ensures constant backflow of error in the absence of new inputs, within a neural network, the CEC is interconnected with other units. This necessitates considering additional weighted inputs and outputs. LSTM addresses this by extending the CEC with input and output gates connected to the network input layer and other memory cells, forming a more complex unit known as a memory block.

The input gates, simple sigmoid threshold units with an activation range of $[0,1]$, regulate signals from the network to the memory cell by scaling them accordingly. They can also learn to shield stored contents from irrelevant signals. The activation of a CEC by the input gate is termed the cell state. Output gates manage access to memory cell contents, protecting other cells from disturbances originating from the current unit. The multiplicative gate units essentially control access to the constant error flow through the CEC by either permitting or denying it.

Memory blocks also feature a forget gate, which weighs the information inside the cells. Whenever previous information becomes irrelevant for some cells, the forget gate can reset the state of these cells, enabling continuous prediction and preventing biases.

Similar to other algorithms, LSTM requires a fixed network topology. The number of memory blocks remains constant throughout training, limiting the overall memory capacity of the network. Increasing the network size uniformly is unlikely to overcome this limitation. Instead, modularization is proposed as a strategy for effective learning, although the specifics of this process remain unclear [6].

7.6 CRNN

The Convolutional Recurrent Neural Network (CRNN) integrates convolutional layers for feature extraction and recurrent layers, particularly Long Short-Term Memory (LSTM) units or gated recurrent unit (GRU), for capturing temporal dependencies.

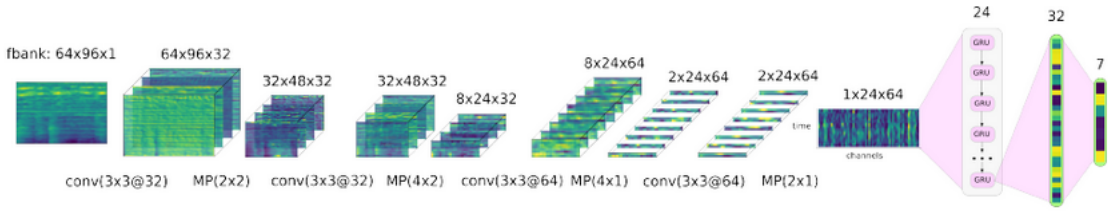


Figure 7.10. *The CRNN architecture employs a sequence of operations to process input features, which consist of matrices representing consecutive frames of log-Mel filter banks (64 filter banks by 96 time frames). Initially, convolutional and max-pooling operations are applied sequentially to extract informative features from the input. Subsequently, these features are passed through gated recurrent units (GRUs) to capture temporal dependencies. The network produces sigmoid scores as outputs, indicating the presence of various acoustic events in the audio signal [7].*

In CRNN, LSTM/ GRUs units play a critical role in modeling the temporal context of acoustic events. These units maintain long-term memory and effectively handle the vanishing gradient problem, thus allowing the model to learn long-range dependencies. Mathematically, the LSTM units in CRNN process input sequences iteratively, retaining information over time through the following equations:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{7.7}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{7.8}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{7.9}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{7.10}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{7.11}$$

$$h_t = o_t \odot \tanh(C_t) \tag{7.12}$$

Here, f_t, i_t, and o_t are the forget gate, input gate, and output gate vectors at time t respectively. C_t is the cell state, $\tilde{C}_t$ is the candidate cell state, and h_t is the hidden state at time t. W and b represent weight matrices and bias vectors, and σ denotes the sigmoid function.

CRNN offers effective feature learning capabilities, robustness to input variability, and the ability to capture complex temporal dynamics in sequential data, making it well-suited for tasks such as audio chord recognition and speech recognition.

7.7 Transfer Learning

Transfer learning has emerged as a potent strategy within the realm of deep learning, offering a means to significantly enhance model performance. This technique involves the utilization of pre-trained models, which have been trained on extensive datasets, to extract generalized feature representations. These representations encapsulate high-level abstractions and patterns, applicable across a broad spectrum of tasks. Integrating transfer learning into deep learning workflows brings forth several compelling advantages, rendering it an indispensable tool in modern applications of machine learning.

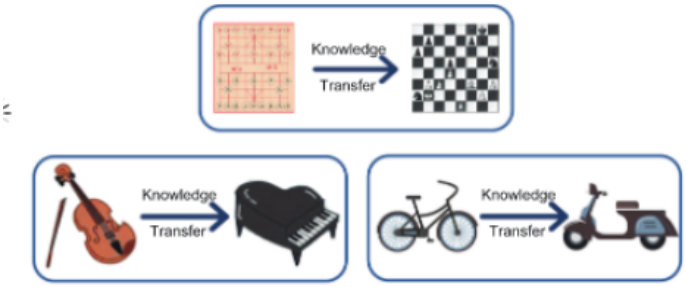


Figure 7.11. *Intuitive examples about transfer learning [8].*

A notable benefit of transfer learning lies in its capacity to tackle challenges associated with limited data availability and the arduous process of data labeling. Acquiring labeled data for a specific task often proves to be a laborious and time-consuming endeavor. Transfer learning alleviates this burden by leveraging knowledge distilled from a source task and transferring it to a target task. Through this mechanism, the pre-trained model captures foundational features and patterns from the source domain, enabling it to generalize effectively to the target domain, even when labeled data is scarce. This capability not only streamlines the data collection and annotation process but also conserves valuable resources and time.

Moreover, the adoption of transfer learning expedites the model training process and enhances convergence rates. Pre-trained models have already assimilated representations of essential features and patterns from vast datasets. By initializing the model with these learned representations, transfer learning facilitates a highly effective starting point for training. This not only accelerates the training phase but also promotes quicker convergence, resulting in improved overall efficiency.

This approach proves particularly advantageous in scenarios where the pre-trained model and the target task exhibit shared underlying patterns or structures. Such alignment enables the pre-trained model to capture pertinent information, which can then be transferred to the target task, leading to enhanced performance. The model's adeptness in extracting significant and discriminative features from the source domain facilitates effective knowledge transfer, especially during fine-tuning or feature extraction stages. Consequently, the model can adapt and specialize efficiently to the target task, leveraging prior knowledge while accommodating task-specific intricacies.

By harnessing the potency of transfer learning, deep learning models can achieve

heightened levels of performance, robustness, and efficiency, thereby solidifying its status as an indispensable technique across a diverse array of applications.

7.8 Metrics

7.8.1 Accuracy

Accuracy is a common metric used to evaluate the performance of a classification model. It measures the ratio of correctly predicted instances to the total instances in the dataset [37]. In **binary classification** it is defined as:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.13}$$

In **multiclass classification** it is defined as:

$$Acc = \frac{correct\ classifications}{all\ classifications} \tag{7.14}$$

where:

TP - True Positives

TN - True Negatives

FP - False Positives

FN - False Negatives

It represents the proportion of correctly classified instances over the total instances. A higher accuracy indicates better performance of the classification model.

7.8.2 Precision & Recall

Precision is a measure of the closeness of agreement between independent test results obtained under stipulated conditions. It is often used in the context of measurement methods and results to assess the repeatability and reproducibility of a standard measurement method [37]. In domains such as pattern recognition, information retrieval, object detection, and classification within machine learning, precision and recall stand as crucial performance metrics applicable to data extracted from a collection, corpus, or sample space.

Precision, often termed as positive predictive value, denotes the proportion of relevant instances present among the retrieved instances 7.15.

$$Precision = \frac{Relevant\ retrieved\ isntances}{All\ retrieved\ instances} \tag{7.15}$$

Recall, also referred to as sensitivity, represents the fraction of relevant instances that were successfully retrieved 7.16.

$$Precision = \frac{Relevant\ retrieved\ isntances}{All\ relevant\ instances} \tag{7.16}$$

In classification tasks, the terms true positives, true negatives, false positives, and false negatives, as defined in Type I and Type II errors, are used to evaluate the performance of the classifier being tested against trusted external judgments. The terms "positive" and "negative" relate to the classifier's predictions (also known as expectations), while "true" and "false" indicate whether those predictions align with the external judgments (also known as observations).

So using those terms, precision and recall can be expressed as following (7.17, 7.18):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7.17}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7.18}$$

7.8.3 F1 score

In the realm of statistical analysis concerning binary classification and information retrieval systems, the F-score or F-measure is a measure of predictive performance. Derived from the precision and recall of a test, as we mentioned earlier, precision denotes the ratio of true positive results to the total number of samples predicted as positive, encompassing both correct and incorrectly identified instances. Similarly, recall signifies the ratio of true positive results to the total number of samples that ought to have been identified as positive. Precision, synonymous with positive predictive value, and recall, also known as sensitivity in diagnostic binary classification, are fundamental components of this assessment.

The F1 score, serving as the harmonic mean of precision and recall, encapsulates both aspects in a single metric, thus providing a balanced representation 7.19.

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{7.19}$$

The F-score ranges between 0 and 1, where a value of 1.0 signifies perfect precision and recall, while a value of 0 indicates the absence of either precision or recall.

7.8.4 Confusion Matrix

A confusion matrix presents a summary of predictions in a matrix format, detailing the number of correct and incorrect predictions for each class. It aids in discerning which classes the model confuses with others. The figure 7.12 illustrates a sample confusion matrix for a binary classification problem.

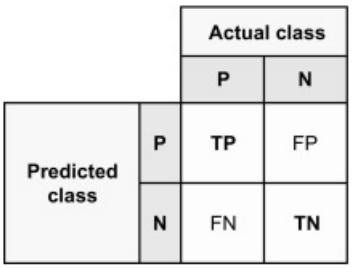


Figure 7.12. *Confusion matrix..*

TP denotes True Positive predictions. In a binary classification scenario, such as identifying fraudulent transactions as "1", TP signifies the count of correctly classified "1"s, representing the number of fraudulent transactions accurately identified. TN, or True Negative predictions, indicate the count of correctly classified "0"s, representing non-fraudulent transactions accurately identified. FP (False Positive) denotes the count of non-fraudulent transactions misclassified as fraudulent, while FN (False Negative) represents the count of fraudulent transactions misclassified as non-fraudulent.

7.8.5 MIREX Chord Estimation Metric

To assess the accuracy of an automatic transcription, it is compared to a ground truth generated by human annotators. In the context of the Music Information Retrieval Evaluation eXchange (MIREX), chord symbol recall (CSR) is commonly used to gauge the alignment between predicted chords and the ground truth [38]:

$$CSR = \frac{\text{total duration of segments where annotation equals estimation}}{\text{total duration of annotated segments}} \tag{7.20}$$

Historically, MIREX [38] has employed an approximate CSR approach, sampling both ground truth and automatic annotations every 10 ms and calculating the ratio of correctly annotated samples to the total number of samples. However, following the methodology proposed by Christopher Harte (2010, §.1.2), the ground truth and estimated annotations can be viewed as continuous audio segmentations, allowing for a more precise CSR calculation. This method considers the cumulative length of correctly overlapping segments, providing enhanced accuracy and computational efficiency by reducing the number of segment comparisons. Additionally, to account for variations in music length, the CSR can be weighted by the duration of each song when computing an average across a corpus, resulting in the weighted chord symbol recall (WCSR).

Chord Vocabularies

MIREX [38] introduces a set of single chord evaluation measures for MIREX, building upon previous iterations and incorporating evaluation metrics from the literature to offer a comprehensive assessment of transcription quality. Inspired by Pauwels and Peeters (2013), they proposes using CSR with five distinct chord vocabulary mappings.

Each vocabulary mapping categorizes full chord descriptions from either estimated or ground-truth transcriptions into predefined classes:

1. Chord root note only.
2. Major and minor chords: {N, maj, min}.
3. Seventh chords: {N, maj, min, maj7, min7, 7}.
4. Major and minor chords with inversions: {N, maj, min, maj/3, min/b3, maj/5, min/5}.
5. Seventh chords with inversions: {N, maj, min, maj7, min7, 7, maj/3, min/b3, maj7/3, min7/b3, 7/3, maj/5, min/5, maj7/5, min7/5, 7/5, maj7/7, min7/b7, 7/b7}.

The mapping process involves examining the root note, bass note, and relative interval structure of chord labels. A mapping is established if both root notes and bass notes match, and the output label structure is the largest possible subset given the vocabulary. For instance, G:7(#9) is mapped to G:maj because the interval set of G:maj ({1,3,5}) is a subset of G:7(#9) ({1,3,5,b7,#9}). Conversely, if a chord cannot be represented by a certain class, e.g., mapping D:aug or F:sus4(9) to {maj, min}, it is excluded from evaluation if present in the ground truth, or considered a mismatch if present in the estimated annotation.

MIREX's vocabulary recommendations are informed by chord quality frequencies in the McGill Billboard corpus 7.4, a representative sample of American popular music spanning several decades. Notably, major and minor chords collectively account for the majority of chord occurrences, with other qualities such as augmented and diminished chords being rare. This distribution guides our selection of chord vocabulary to maximize coverage while maintaining relevance to real-world musical contexts.

Table 7.4. *Most frequent chord qualities in the McGill Billboard corpus.*

Quality	Freq. (%)	Cum. Freq (%)
maj	52	52
min	13	65
7	10	75
min7	8	83
maj7	3	86
5	2	88
1	2	90
maj(9)	1	91
maj6	1	92
sus4	1	93
sus7	1	94
sus9	1	94
7(#9)	1	95
min9	1	96

Chord Segmentation

In addition to CSR, chord transcription evaluation encompasses metrics focusing on the segmentation of automatic transcriptions. We propose incorporating the directional Hamming distance, a metric utilized in the chord transcription literature. The directional Hamming distance measures the segmentation quality by finding the maximally overlapping segment for each annotated segment in both transcriptions and summing the differences. This yields a measure of over- or under-segmentation, which can be combined to derive an overall quality metric:

$$Q = 1 - \frac{\text{maximum of directional Hamming distances in either direction}}{\text{total duration of song}} \tag{7.21}$$

7.9 Audio Signal Processing

Audio signal processing has traditionally relied on digital signal processing (DSP) techniques to extract features from audio signals, such as phonemes for understanding human speech. This approach demanded substantial domain-specific expertise to fine-tune systems for optimal performance. However, with the rise of Deep Learning, there's been a notable shift. Deep Learning has emerged as a powerful tool in handling audio data, eliminating the need for traditional DSP techniques. Instead, deep learning models can directly process audio data, bypassing the manual feature extraction step. Interestingly, deep learning models commonly operate on image data rather than raw audio. This transformation is achieved by generating Spectrograms from audio signals, which convert audio data into visual representations. Spectrograms provide a way to visualize the frequency content of audio signals over time, enabling deep learning models to effectively process audio information using standard Convolutional Neural Network (CNN) architectures. Thus, while it might sound like science fiction, converting sound into images through Spectrograms has become a commonplace and integral part of modern audio signal processing workflows.

7.9.1 Time & Frequency Domains

In the time domain, signals are represented as amplitude variations over time. This representation allows for the observation of signal dynamics and temporal patterns. Time domain analysis involves techniques such as waveform visualization, signal detection, and time-based analysis. By examining signals in the time domain, researchers can gain insights into how signals evolve over time and identify temporal characteristics.

Conversely, the frequency domain representation portrays signals in terms of their frequency components. By analyzing signals in the frequency domain, researchers can identify the spectral composition of the signal, including individual frequencies, harmonics, and overtones. Frequency domain analysis is particularly useful for tasks such as spectral analysis, filtering, and frequency-based feature extraction.

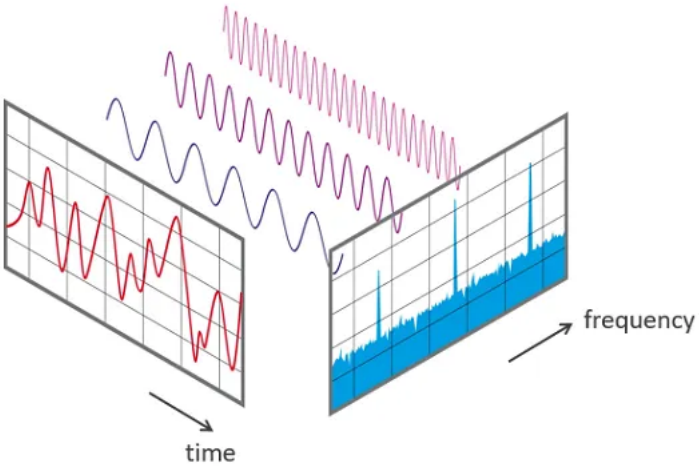


Figure 7.13. *Time Domain vs Frequency Domain [9].*

7.9.2 Spectogram

A spectrogram provides a visual representation of the frequency spectrum of a signal over time. Often referred to as sonographs, voiceprints, or voicegrams when applied to audio signals, spectrograms showcase how the signal's frequencies evolve. When represented in a 3D plot, they may be termed as waterfall displays [39].

In various fields such as music, linguistics, sonar, radar, speech processing, seismology, and ornithology, spectrograms find extensive utility. They enable phonetic identification of spoken words and analysis of animal vocalizations.

Generated through methods like optical spectrometers, bank of band-pass filters, Fourier transform, or wavelet transform (also known as scaleograms or scalograms), spectrograms offer diverse insights into signals. They are typically visualized as heat maps, with intensity depicted through variations in color or brightness.

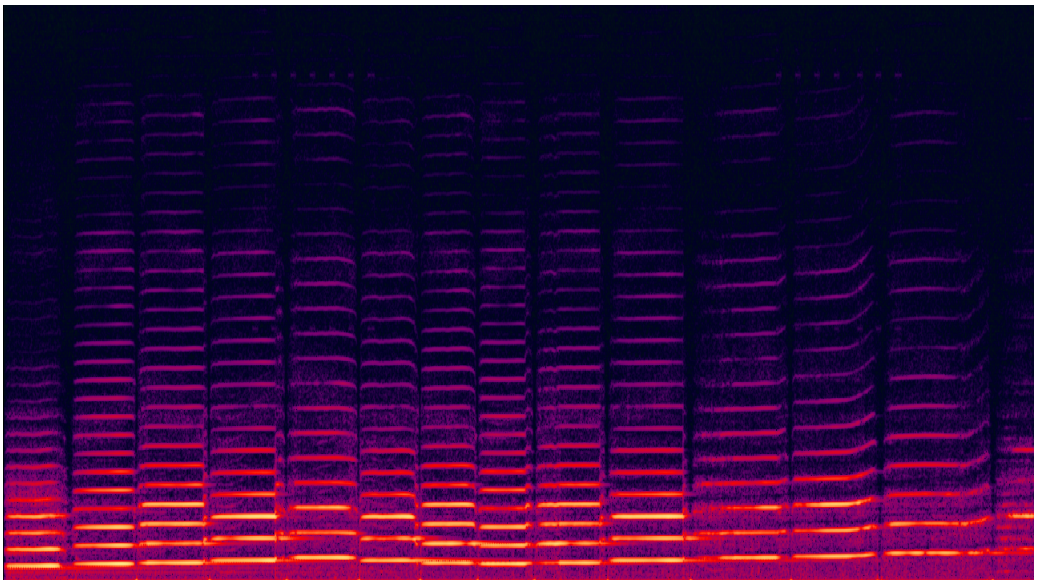


Figure 7.14. *The spectrogram of this violin recording showcases the harmonics occurring at whole-number multiples of the fundamental frequency. [10].*

7.9.3 Short-Time Fourier Transform (STFT)

Short-Time Fourier Transform (STFT) is a method used for analyzing the frequency content of non-stationary signals over time. It operates by dividing a time-domain signal into frames, either separate or overlapping, through multiplication with a window function, followed by application of the Fast Fourier Transform (FFT) on each frame. This dynamic approach allows for tracking frequency changes as the window moves through the signal [10]. Consequently, STFT finds extensive application in domains needing continuous frequency monitoring, such as radar systems and voice-signal processing. In the realm of deep learning, STFT is employed due to its ability to provide valuable time-frequency representations of signals, which serve as essential inputs for neural networks engaged in tasks like speech recognition, audio classification, and environmental sound analysis.

The STFT can be expressed as 7.22:

$$X_{\text{STFT}}[k, n] = \frac{1}{N} \sum_{m=0}^{N-1} x[m + nH] \cdot w[m] \cdot W_k^m, \tag{7.22}$$

where:

- $x[m]$ is the input signal,
- $w[m]$ is the window function,
- N is the length of the window,
- n is the time frame index,
- k is the frequency index,

- H represents the hop length,
- W_k is the complex exponential term.

The overlap length between adjacent frames is $N - H$, and the overlap ratio between consecutive frames is $\frac{N-H}{N}$. At a specific time nH, the signal $x[m]$ is multiplied by the window function $w[m]$. Therefore, 7.22 represents the FFT operation of $x[m + nH]w[m]$.

The STFT processor measures the frequency over time by moving the window function $w[m]$ along the signal $x[m]$ according to the hop length and performing the FFT operation on samples inside the window.

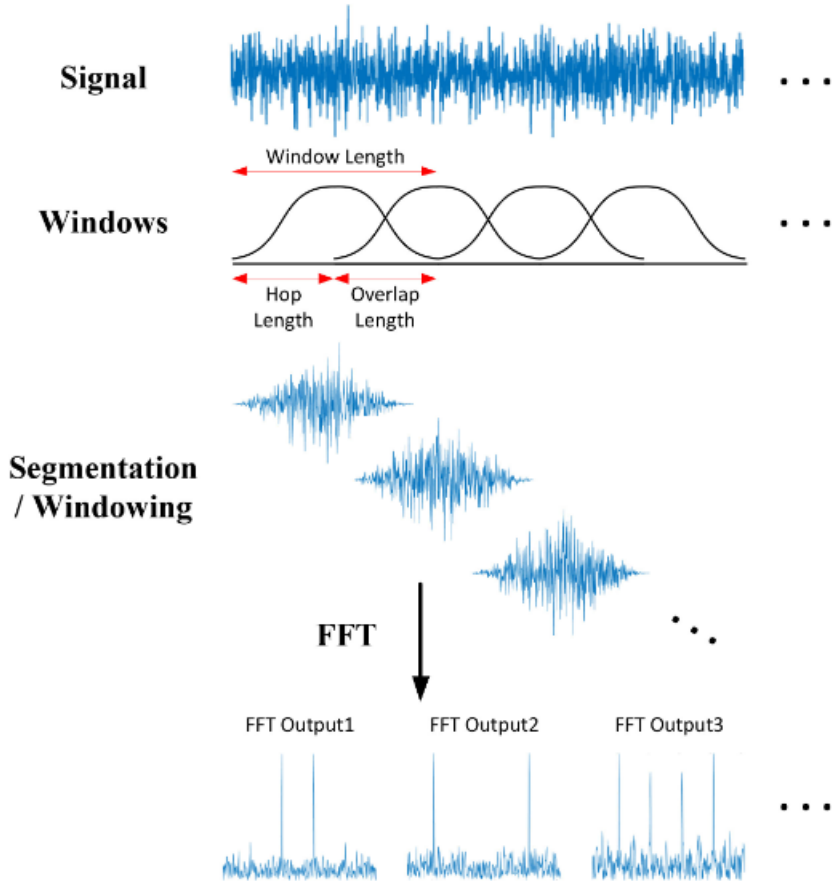


Figure 7.15. *Short-time Fourier transform (STFT) overview [10].*

7.9.4 Constant-Q Transform (CQT)

The Constant-Q Transform (CQT) and its closely related Variable-Q Transform (VQT) are pivotal tools in mathematics and signal processing, particularly tailored for the frequency domain analysis of data series, with a particular focus on musical representation. Comparable to the Fourier transform and intricately linked to the complex Morlet wavelet transform, the CQT/VQT unveils the frequency components of a signal, especially useful in discerning musical notes.

The foundation of the CQT lies in a series of logarithmically spaced filters, indexed by k, each with a bandwidth δf_k, which is a multiple of the bandwidth of the previous filter:

$$\delta f_k = 2^{1/n} \cdot \delta f_{k-1} = \left(2^{1/n}\right)^k \cdot \delta f_{\min}, \tag{7.23}$$

where δf_k represents the bandwidth of the k-th filter, $\delta f_{\min}$ is the bandwidth of the lowest filter, and n signifies the number of filters per octave.

In the calculation of the CQT, the short-time Fourier transform of a data series $x[n]$ for a frame shifted to sample m is computed using:

$$X[k, m] = \sum_{n=0}^{N-1} W[n - m]x[n]e^{-j2\pi kn/N}, \tag{7.24}$$

where $X[k, m]$ denotes the short-time Fourier transform result, W is the windowing function, $x[n]$ represents the input data, j is the imaginary unit, k signifies the frequency bin, and N denotes the number of samples.

To delve into the specifics of the transform, several key parameters are defined:

- Filter width, δf_k.
- Quality factor, Q, calculated as:

$$Q = \frac{f_k}{\delta f_k}. \tag{7.25}$$

  This factor signifies the integer number of cycles processed at a center frequency f_k, thus delineating the time complexity of the transform.

- Window length for the k-th bin:

$$N[k] = \frac{f_s}{\delta f_k} = \frac{f_s}{f_k} Q, \tag{7.26}$$

  where f_s represents the sampling frequency, f_k stands for the center frequency of the k-th bin, and Q is the quality factor. This equation signifies the number of samples processed per cycle at frequency f_k, with Q representing the number of integer cycles processed at this central frequency.

After appropriate adjustments, the transformed output $X[k]$ can be expressed as:

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k, n]x[n]e^{-j2\pi Qn/N[k]}, \tag{7.27}$$

where $X[k]$ denotes the transformed output, W represents the windowing function, $x[n]$ signifies the input data, j is the imaginary unit, k denotes the frequency bin, and $N[k]$ represents the window length for the k-th bin.

Chapter 8

Data & Preprocessing

In this chapter the Dataset and the preprocessing methodologies followed are analysed.

8.1 Isophonics Dataset

The Isophonics dataset is a collection of audio recordings, annotations, and metadata designed for music information retrieval (MIR) research. It's a valuable resource for studying various aspects of music, including melody, harmony, rhythm, and structure. For this study, The Isophonics dataset was used providing 180 songs by The Beatles. This dataset's chord annotations have been checked several times by Christopher Harte [12] and the MIR community, and can be used with confidence. The Chord labels are .lab files. Those are whitespace delimited text files with three columns, corresponding to, onset time, offset time and chord label, respectively. An example can be seen in the table 8.1 below.

Start Time	End Time	Label
0.000000	2.612267	N
2.612267	11.459070	E
11.459070	12.921927	A
12.921927	17.443474	E
17.443474	20.410362	B
20.410362	21.908049	E
21.908049	23.370907	E:7/3
23.370907	24.856984	A
24.856984	26.343061	A:min/b3
26.343061	27.840748	E

Table 8.1. *First 10 rows of the labels for "I saw her standing there", start and end time are counted in seconds.*

The format of the tracks found was stereo .mp3 at 44kHz sample rate so further processing was necessary. To ensure labels are in sync with the audio, Audacity was utilised.

8.2 Pre-processing

In order to prepare the audio data for further analysis, some pre-processing steps need to be completed. A high level view of the pre-processing procedure can be seen in the figure 8.1 below.

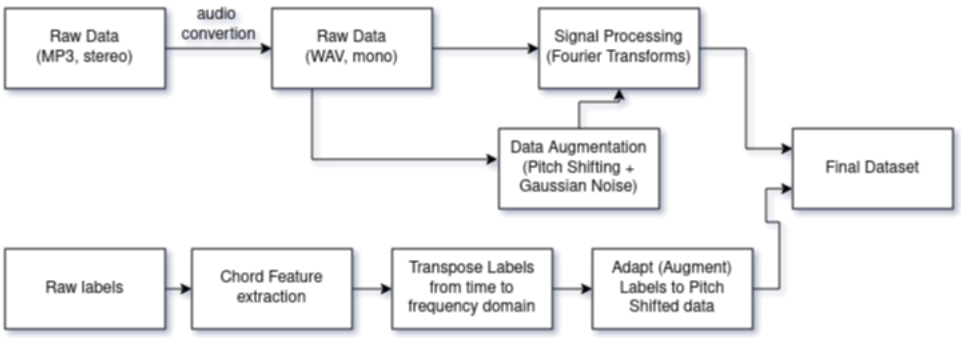


Figure 8.1. *Data Pre-processing flow chart.*

8.2.1 Audio Conversion

The audio conversion procedure revolves around converting audio files from MP3 to WAV format and from stereo to mono. WAV and MP3 formats differ in compression and audio quality, with WAV uncompressed for high fidelity and MP3 lossy for smaller size. Stereo utilizes two channels for spatial depth, while mono uses one, ideal for non-spatial applications. WAV files, being uncompressed, retain all original audio data, preserving high fidelity crucial for tasks like speech recognition and music generation. dditionally, mono audio simplifies the input data by reducing the dimensionality, making it more computationally efficient for processing and training deep neural networks.

Initially, a script was designed using Pydub Python module that gathers a list of MP3 files within a specified directory. Then, for each file, it employs conversion methods to transform it into WAV format and to mono. This systematic approach ensures that the audio data is prepared in a standardized format suitable for further analysis. An example of a transformed file can be visualized as in Figure 8.2

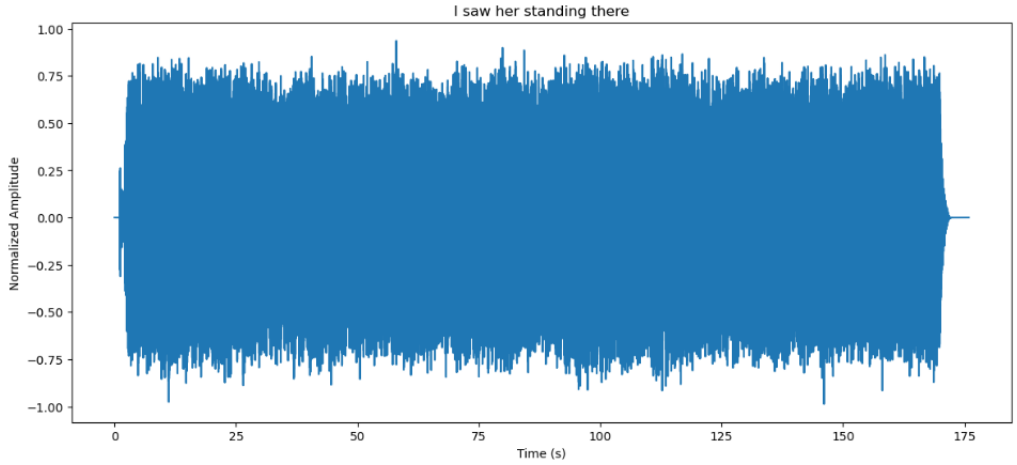


Figure 8.2. *Waveform representation of track "I was her standing there".*

8.2.2 Signal Processing - Fourier Transforms

Convolutional neural networks (CNNs) heavily rely on their convolutional layers, which are fundamental components responsible for applying filters to input data or feature maps, thus convoluting the output from preceding layers. These layers are tasked with learning the filter weights, and in intricate CNN architectures characterized by numerous layers and filters, the computational burden can escalate significantly. Integrating Fourier transform into CNNs alters the calculation approach of these layers to element-wise products in the frequency domain, effectively conserving computational resources. While the network's objective remains unchanged, leveraging Fourier transform aids in optimizing computations, often achieved through the utilization of fast Fourier transform techniques. In essence, the operation of convolutional layers can be analogized to Fourier transform, wherein convolution in the temporal domain corresponds to multiplication in the frequency domain. This analogy offers insight into convolutional operations, akin to polynomial multiplication, facilitating a deeper understanding of CNN functionality.

In this part two Fourier transforms were tested, Short time Fourier Transform and Logarithmic frequency spectrogram (constant-Q) and the Constant-Q chromagram.

Short time Fourier Transform

The standard linear Short-Time Fourier Transform (STFT) generated an excessive number of frequency bands, resulting in an abundance of input features for the network. Consequently, this led to a substantial increase in the overall parameter count of the network and did not yield the desired results. In the example below 8.3 the visualization of the Short time Fourier Transform of "I saw her standing" there is presented. The parameter n_fft specifies the number of samples in each FFT window, affecting the frequency resolution of the spectrogram. A larger n_fft results in finer frequency bins but may require more computational resources. The hop_length parameter determines the number of samples between successive frames, influencing the time resolution of the spectrogram. A larger hop_length reduces temporal resolution but may improve compu-

tational efficiency.

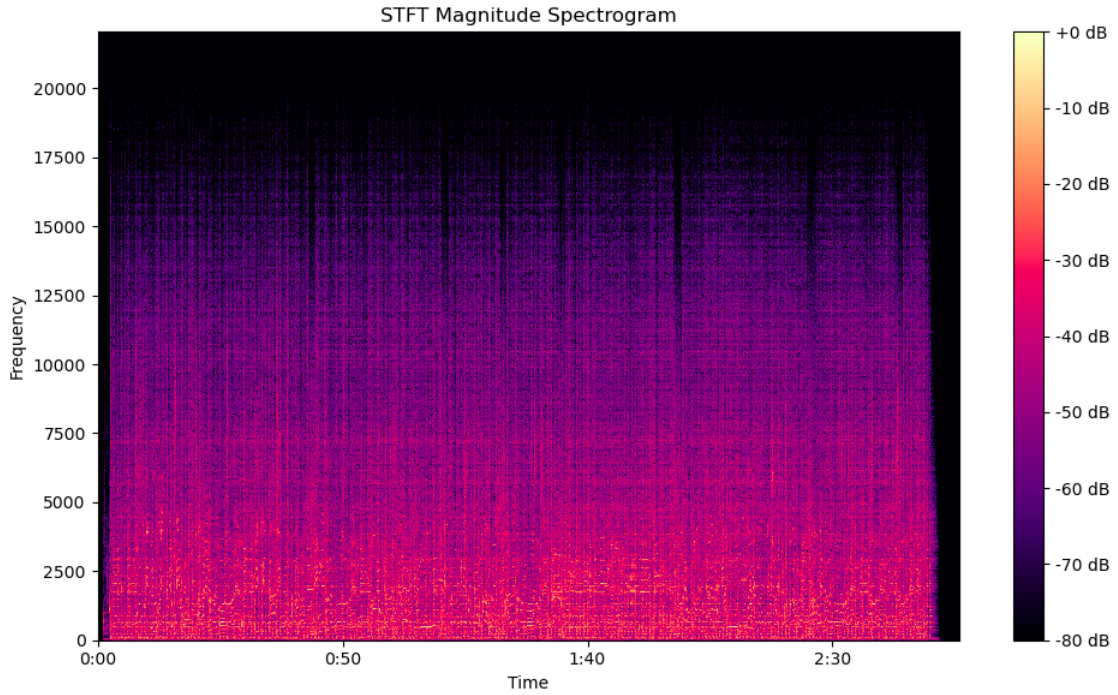


Figure 8.3. *Short Time Fourier Transform of the track "I saw her standing there". The STFT parameters used are n_fft=8192 and hop_length=4410.*

Constant-Q Chromagram

The Chromagram, consisting of 12 input features, offers a detailed representation of the musical notes present in the audio signal. However, its focus primarily revolves around identifying the pitch classes or musical notes, rather than providing insights into how these notes are distributed across the frequency spectrum. For instance, while the chromagram may effectively distinguish between different pitches, such as G# and D, it fails to capture their specific frequency locations within the audio spectrum. This limitation makes it challenging to understand the spatial distribution of notes, hindering the ability to analyze complex musical structures accurately. Moreover, detecting inversions, where the notes of a chord are rearranged, becomes more difficult due to the lack of frequency-specific information provided by the chromagram. In summary, while the chromagram offers valuable information about musical notes, its inability to represent the frequency distribution of t hese notes poses limitations when analyzing intricate musical compositions.

In the example below 8.4 the visualization of the Short time Fourier Transform of "I saw her standing" there is presented. The hop_length parameter in audio processing, such as in the context of calculating the Pitch Class Profile (PCP) Chromagram, refers to the number of samples between successive frames of the analysis. In other words, it determines the spacing between the starting points of each frame as the analysis window slides along the audio signal. A smaller hop_length results in more frames being analyzed per unit time, providing higher temporal resolution but potentially leading to increased

computational complexity. Conversely, a larger hop_length reduces the number of frames analyzed per unit time, resulting in lower temporal resolution but potentially improving computational efficiency.

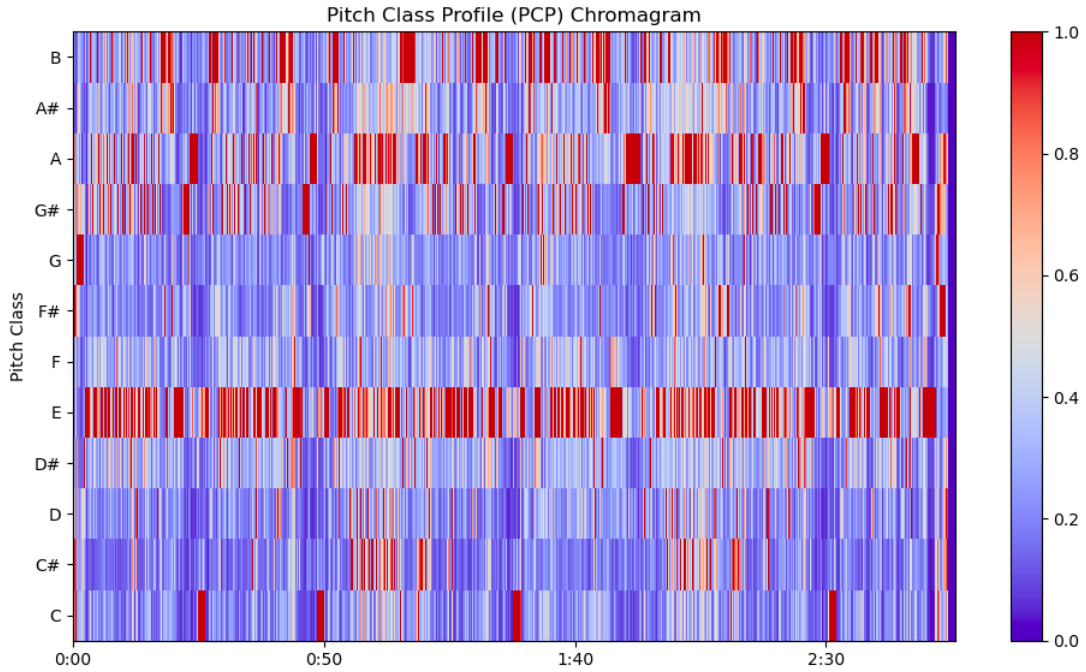


Figure 8.4. *Constant-Q Chromagram of the track "I saw her standing there". The parameters used are hop_length=4410.*

Constant Q Transform (CQT)

The Constant Q Transform (CQT) is preferred over other methods due to its unique properties that make it well-suited for audio signal analysis. Unlike the traditional Short-Time Fourier Transform (STFT), which employs a linearly spaced frequency scale, the CQT utilizes a logarithmically spaced frequency scale. This logarithmic scaling better matches the human auditory system's perception of pitch, resulting in improved representation of musical content, especially for signals with complex harmonic structures.

Additionally, the CQT offers a fixed frequency resolution across all octaves, ensuring consistent representation of pitch classes regardless of the signal's frequency content. This property is particularly advantageous for tasks such as pitch estimation and music transcription, where accurate representation of pitch is essential.

Below is a visualization of the Constant-Q transform 8.5 using a sample rate of 44100, 192 bins (24 bins per octave), and a hop length of 4096, I ensured thorough coverage of the audio signal's frequency spectrum, facilitating effective feature extraction. The nbins parameter determines the number of frequency bins utilized in the spectrogram, impacting the resolution of frequency representation. Similarly, bins per octave controls the frequency resolution within each octave, influencing the trade-off between precision and computational efficiency. Additionally, the hop length parameter dictates the temporal resolution of the spectrogram by specifying the spacing between successive frames of analysis.

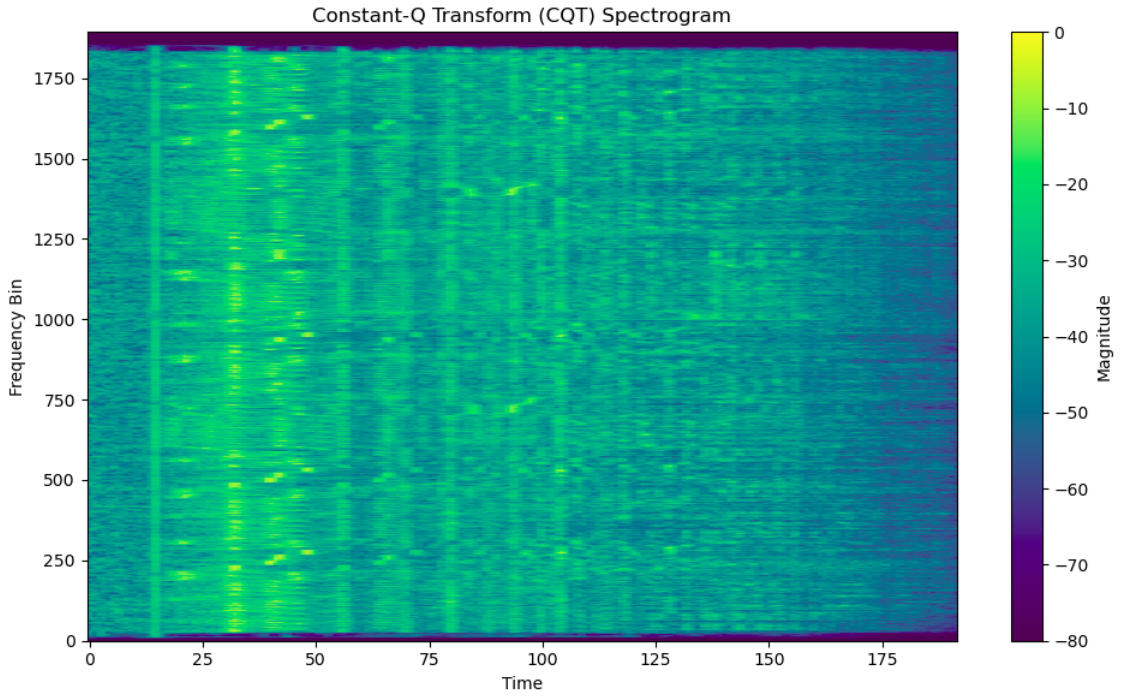


Figure 8.5. *Constant-Q Fourier Transform of the track "I saw her standing there". The parameters used are a sample rate of 44100, 192 bins (24 bins per octave), and a hop length of 4096.*

8.2.3 Label pre-processing

Time to Frequency domain

Before proceeding, it was essential to transform the labels from the time domain to the frequency domain. This step was crucial for aligning the labels with the corresponding frequency bins in the spectrogram representation. By converting the labels, we ensured that they accurately reflected the spectral characteristics of the audio signal, facilitating further analysis and interpretation in the frequency domain. This transformation allowed for a seamless integration of label information with the Fourier transformed data.

The transformation from time to frequency domain can be seen in the tables 8.2 and 8.3 below. This result can be achieved by the following process, first, the duration of the audio track is determined using the end time of the last label. Next, the duration of each time step (or frame) is computed based on the total duration of the track and the desired number of steps. Then, a series of timestamps are generated evenly spaced over the duration of the track. For each timestamp, the corresponding label is assigned based on the time-based labels. If a timestamp falls within the duration of a time label, it is assigned the corresponding label. Otherwise, it advances to the next time label and assigns its associated label to the next timestamp.

Start	End	Chord
0	0.000000	N
1	2.612267	E
2	11.459070	A
3	12.921927	E
4	17.443474	B
⋮		

Table 8.2. *Labels on the time domain.*

Time signature of Frame	Chord
0.0	N
0.0116	N
0.0232	N
0.0348	N
⋮	
2.7503	E
2.7619	E
2.7735	E
⋮	

Table 8.3. *Labels on Frequency domain with time signature of each frame.*

Feature Extraction - Chord analysis

Feature extraction (Chord analysis) is a necessary step to achieve better results when modelling. A script was implemented for extracting essential features from chord annotations in music data, such as root notes, bass notes, chord triads, and chord extensions(4th, 5th note). This feature extraction process is crucial for several reasons. First, it provides a structured representation of chord annotations, making them easier to interpret and analyze. By extracting root and bass notes, it identifies the foundational elements of each chord, providing insights into harmonic structure and progression. Additionally, categorizing chords into triads and extensions allows for a better understanding of their tonal characteristics and complexity.

The chord is split into 5 features. The root and bass note are two of them. The third feature is the triad note, in this project as **Triad** we will be referring to the third note (major, minor, diminished, augmented, sus2, sus4). Fourth and Fifth feature are extension 1 and extension 4, meaning 4th and 5th note accordingly.

At this point it is essential to mention that to simplify the embedding process later, all Sharps were translated to their Flats, for example C# was translated to Db which represents the same exact frequency.

Each Chord was analysed to its components (shown on table 8.4 below.

Component	Values	Embedding
Root	N, C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb, B	0 - 12
Bass	N, C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb, B	0 - 12
Triad	N, Major, Minor, Diminished, Augmented, Sus 2, Sus4	0 - 6
Extension 1	N, dim7, hdim7, maj7, maj6, min7, 7	0 - 6
Extension 2	N, 9	0 - 1

Table 8.4. *Chord representation into components and Embedding values.*

An Example of this Chord representation can be seen in the table below for a number of Chords 8.5.

Chord	Root	Bass	Triad	Extension 1	Extension 2
N	N	N	N	N	N
D	D	D	maj	N	N
A	A	A	maj	N	N
E	E	E	maj	N	N
G	G	G	maj	N	N
D/b7	D	C	maj	N	N
G/3	G	B	maj	N	N
G:min(9)/b3	G	Bb	min	min7	9
D/5	D	A	maj	N	N
B	B	B	maj	N	N
F#:min7	Gb	Gb	min	min7	N
G#:(1)	Ab	Ab	maj	N	N
Bb	Bb	Bb	maj	N	N
D:min	D	D	min	N	N
⋮	⋮	⋮	⋮	⋮	⋮

Table 8.5. *Chord Representation Example.*

8.2.4 Data Analysis

In this sections a simple data analysis will take place regarding the Chords and their occurrence in the dataset.

Firstly, it would be helpful to see the overall Chord distribution in the dataset. A visualization of this can be seen in the figure below 8.6.

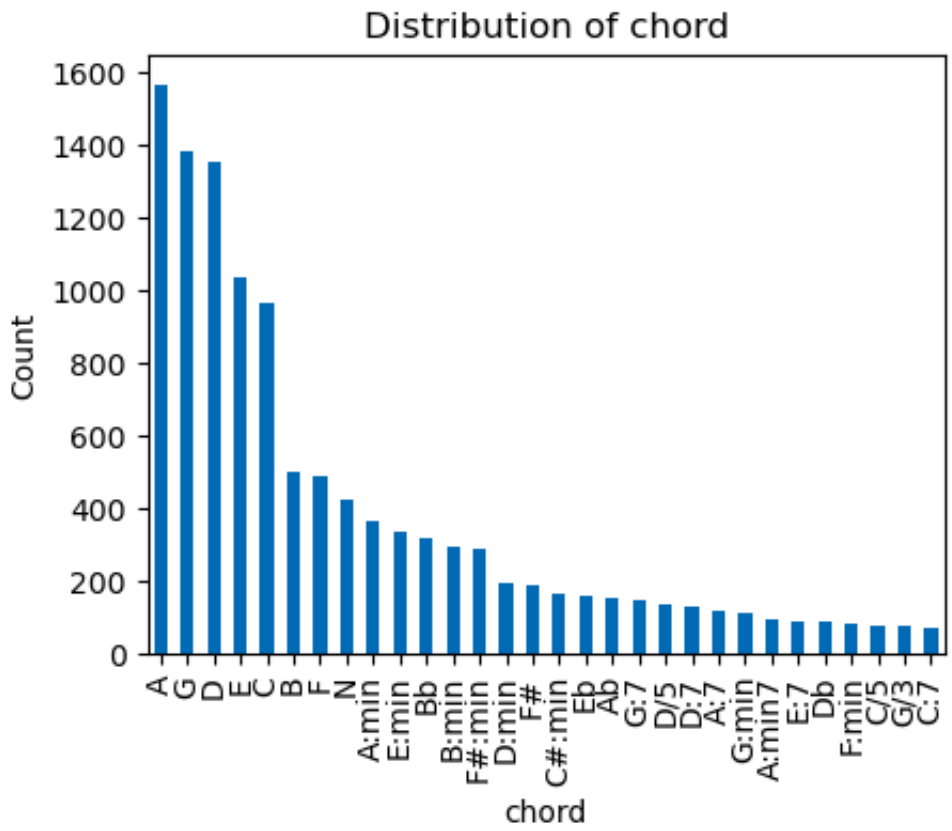


Figure 8.6. *Chord distribution in the Dataset.*

The frequent appearance of basic chords like A, G, D, and E major suggests a preference for straightforward harmony in the music dataset. These chords are common in various genres, likely chosen for their simplicity and familiarity, especially on instruments like the guitar. Their prevalence hints at recurring structural patterns or tonal schemes within the music pieces, providing insights into the harmonic language and compositional techniques used by composers or performers.

Now that we've finished analyzing the chords and understand their structure, we're ready to dive deeper into examining each component of the chords. The figure 8.7 gives information about root note distribution in the dataset.

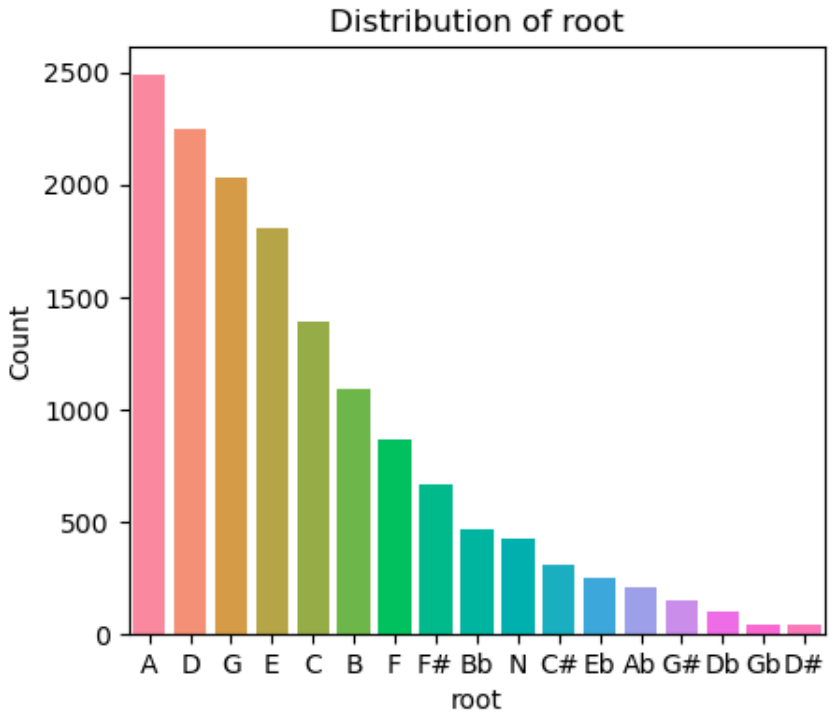


Figure 8.7. *Root note distribution in the Dataset.*

The figure 8.8 presents the distribution of the bass note in the dataset.

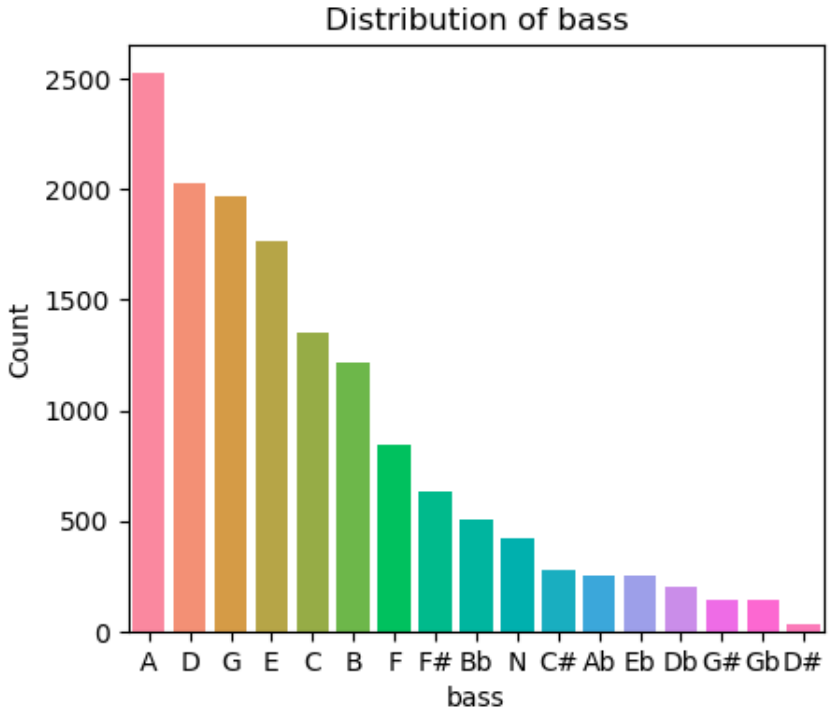


Figure 8.8. *Bass note distribution in the Dataset.*

The distribution patterns of root and bass notes in the dataset reveal a noticeable imbalance, which could potentially impact the performance of our deep learning models. To address this issue and ensure reliable chord recognition, we will implement data

augmentation techniques, specifically pitch shifting. By applying pitch shifting to existing chord samples, we aim to augment the dataset, thereby creating a more balanced representation across all chord types.

It is worth analysing the rest of the chord structure's components 8.9. Here, we observe a similar imbalance in the dataset, highlighting the infrequent usage of certain triads such as sus4, aug, and sus2 across most music genres. Additionally, extension 1 and 2 notes exhibit low occurrence rates. This kind of distribution was expected from the dataset due to the music style it represents.

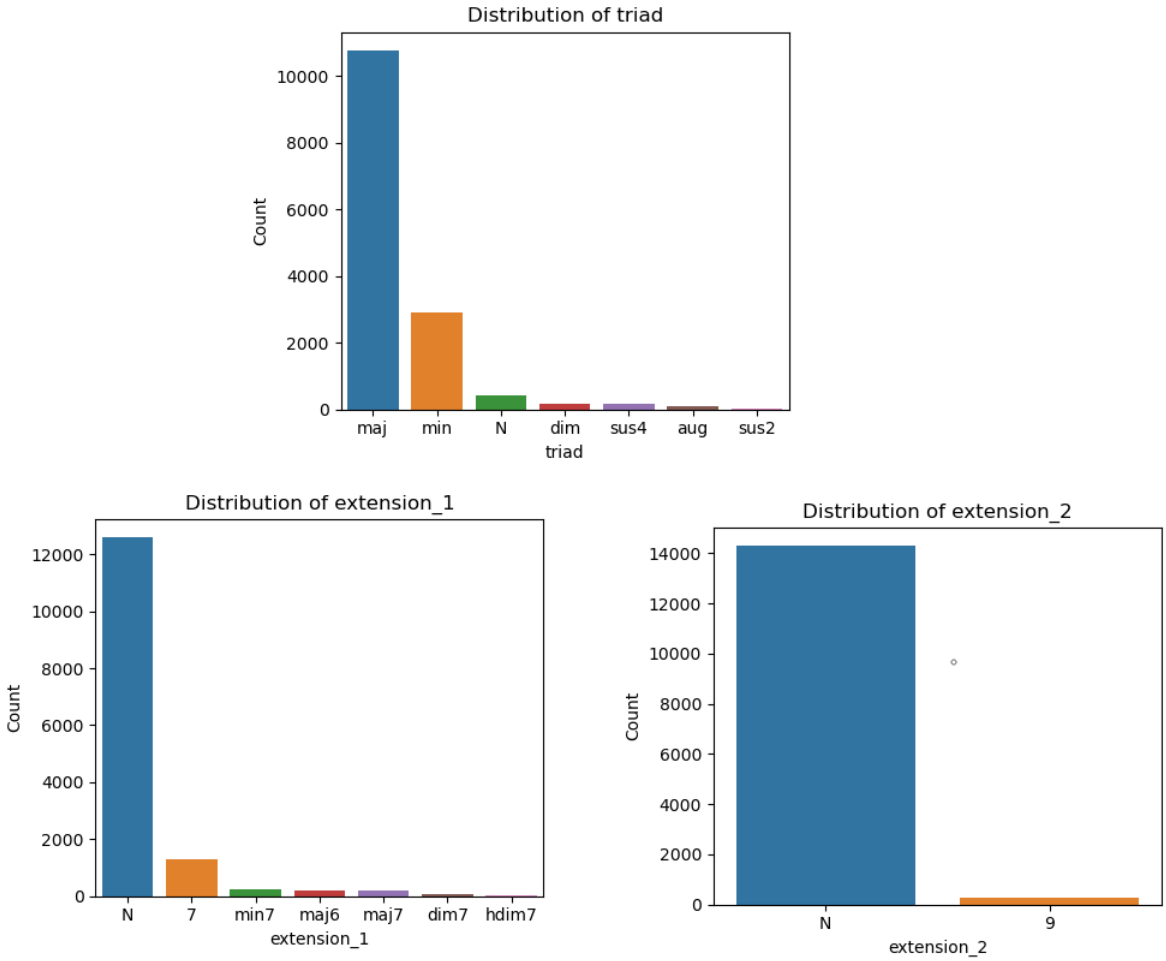


Figure 8.9. *Triad, Extension 1 and Extension 2 notes distribution in the Dataset.*

8.2.5 Data Augmentation with Pitch Shifting

In the realm of audio signal processing, data augmentation plays a pivotal role in enhancing the robustness and generalization capability of machine learning models. Augmentation techniques aim to diversify the training dataset by introducing variations in the input data without altering its inherent semantic meaning. Among various augmentation methods, pitch shifting stands out as a fundamental approach to manipulate audio signals while preserving their structural characteristics. A simple and high level example of data augmentation can be seen in the figure 8.10 below.

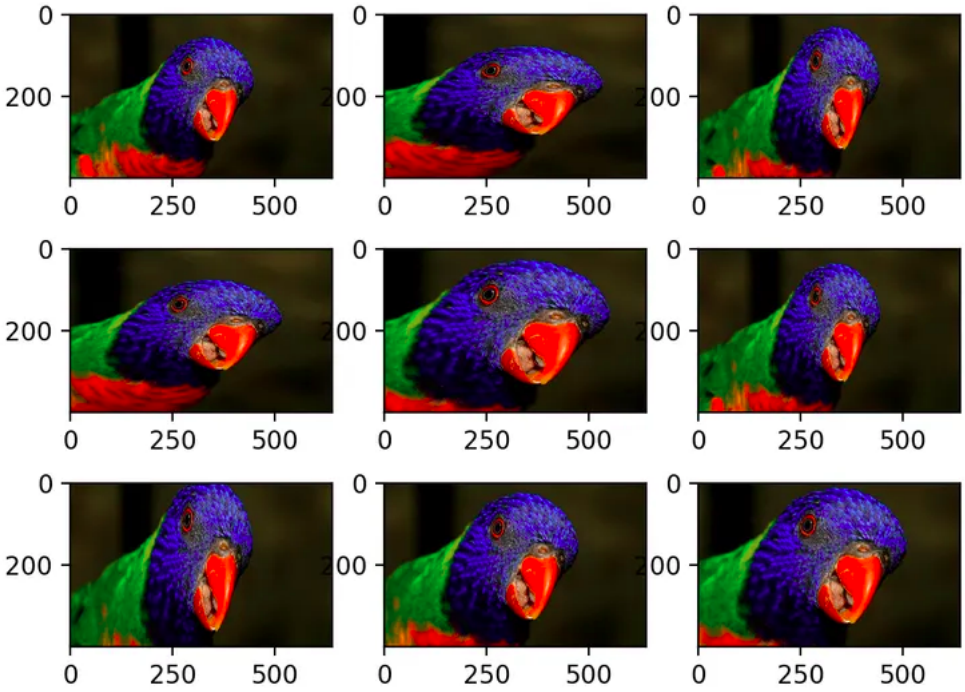


Figure 8.10. *Simple example of Data Augmentation.*

Pitch shifting involves altering the frequency content of an audio signal, thereby modifying its perceived pitch without affecting its temporal duration. This technique is commonly used in music production and audio processing to achieve desired musical effects, such as transposing melodies or harmonizing vocal lines. In the context of data augmentation, pitch shifting offers a versatile tool for generating augmented samples with varying pitch levels while retaining the original harmonic structure of the audio.

In this project, the process of pitch shifting was implemented to manipulate the frequency content of audio signals. Specifically, pitch shifting was applied to alter the perceived pitch of the audio by shifting its spectral components either upwards or downwards. The extent of pitch shifting was controlled within a range of up to 5 semitones, allowing for variations in the pitch without compromising the overall harmonic structure of the audio.

Pitch shifting within the range of up to 5 semitones enabled us to explore a wide spectrum of pitch variations while maintaining the integrity of the original audio samples. By shifting the pitch in both upward and downward directions, we could generate augmented samples with varying pitch levels, thereby enriching the diversity of the training dataset. This augmentation strategy is particularly beneficial for training machine learning models to recognize and classify audio signals across a broader range of pitch variations, enhancing their robustness and adaptability in real-world scenarios.

In addition to pitch shifting, an essential aspect of the data augmentation strategy involved the incorporation of Gaussian noise into the pitch-shifted audio data. Gaus-

sian noise, characterized by its random distribution following a Gaussian or normal distribution, introduces stochastic variability into the audio signals, mimicking real-world environmental conditions and enhancing the model's robustness to noise.

By including Gaussian noise in the pitch-shifted data, we aimed to simulate the inherent variability and unpredictability present in real-world audio recordings. This augmentation technique helps the machine learning models learn to distinguish between signal and noise, improving their ability to generalize to noisy environments and unseen data by reducing over-fitting.

8.2.6 Data Chunking

To prepare the data for utilization in two-dimensional Deep Learning models, we employed a technique known as data chunking. This process involved partitioning the input data into smaller segments, each comprising 100 consecutive time steps, equivalent to approximately 7 seconds of audio. This number was treated as a hyper parameter when training the models and was resulted after fine tuning. The rationale behind this approach stems from the utilization of a bidirectional Long Short-Term Memory (LSTM) layer in our model architecture.

The use of small data chunks is imperative for enhancing the LSTM's learning efficacy and efficiency. By breaking down the input time series into manageable segments, the bidirectional LSTM layer can better capture and understand temporal dependencies within the data. This granularity allows the model to effectively learn patterns and relationships across shorter time intervals, facilitating more accurate predictions and improved performance.

In essence, data chunking plays a vital role in optimizing the training process of two-dimensional Deep Learning models, particularly those employing bidirectional LSTM layers. By providing the model with smaller, temporally coherent segments of data, we enable it to learn more effectively and extract meaningful features from the input audio signals, ultimately enhancing its performance in various audio processing tasks.

Chapter 9

Experiments and Results

This chapter delves into the analysis, training, and comparison of various model architectures for chord recognition tasks. Models such as 1D CNN and 2D CNN utilizing LSTM and fully connected layers will be examined.

9.1 Model 1: Simple 1D Convolutional Neural Network

The initial exploration begins with the implementation of a simple 1D Convolutional Neural Network (CNN). This approach serves as a baseline for evaluating model performance metrics. Each chord is embedded individually without incorporating any chord structural representations mentioned in the previous chapter, thereby disregarding music theory and chord relations, which may limit the model's predictive capabilities.

For this architecture, a Constant-Q Chromagram was used as a preprocessing step, resulting in 12 features as discussed in the previous chapter 8. The architecture of the model is outlined in the figure 9.1 and table 9.1 below.

Table 9.1. *Basic 1D Convolutional Neural Network (CNN) model summary.*

Layer (type)	**Output Shape**	**Parameters #**
conv1d (Conv1D)	(None, 10, 32)	128
max_pooling1d (MaxPooling1D)	(None, 5, 32)	0
flatten (Flatten)	(None, 160)	0
dense (Dense)	(None, 128)	20608
dense_1 (Dense)	(None, 1552)	200208
Total params:		**220944 (863.06 KB)**
Trainable params:		**220944 (863.06 KB)**
Non-trainable params:		**0 (0.00 Byte)**

For the training process, the dataset underwent a split into two subsets: a training set comprising 80% of the data and a validation set comprising 20% of the data. Notably, the split was performed track-wise, ensuring that individual tracks remained intact rather

than being divided into smaller segments. This approach maintains the integrity of the musical structure within each track and provides a more accurate representation of real-world scenarios during model training and evaluation.

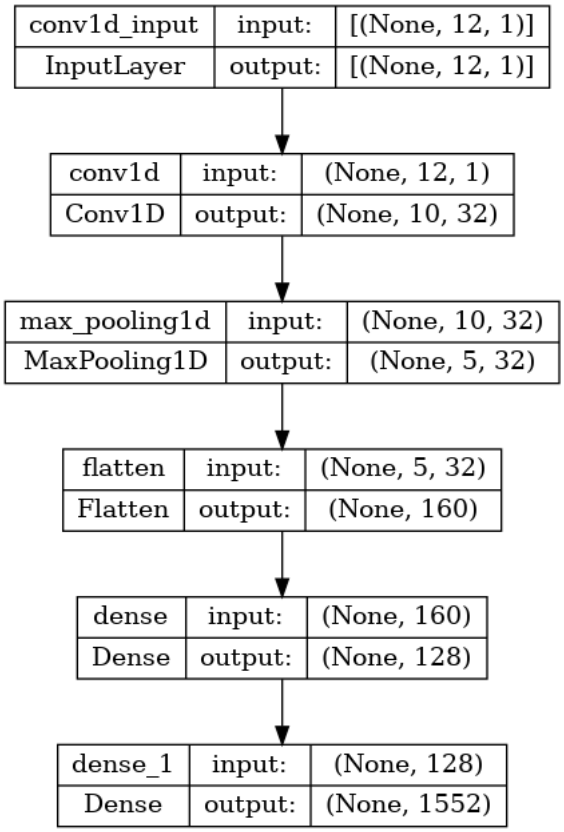


Figure 9.1. *Architecture of basic 1D Convolutional Neural Network (CNN)*

The network comprises several layers, beginning with a 1D Convolutional layer with 32 filters and a kernel size of 3, employing the ReLU activation function. Subsequently, a max-pooling layer with a pool size of 2 is applied to downsample the feature maps. The flattened layer reshapes the output from the previous layer into a one-dimensional vector, facilitating compatibility with fully connected layers. Following this, a densely connected layer with 128 neurons and ReLU activation is utilized for feature extraction. Finally, the output layer consists of 1552 (all unique chords in the dataset) units with a softmax activation function.

For training the model, the batch size used is 32 with the learning rate set to 0.0001 using the Adam optimizer and the Relu as activation function. The loss function used is the Sparse Categorical Cross-entropy. The results of the training and evaluation process can be seen in the graph 9.2 below. The Accuracy falls around 42 % for both training and evaluation set.

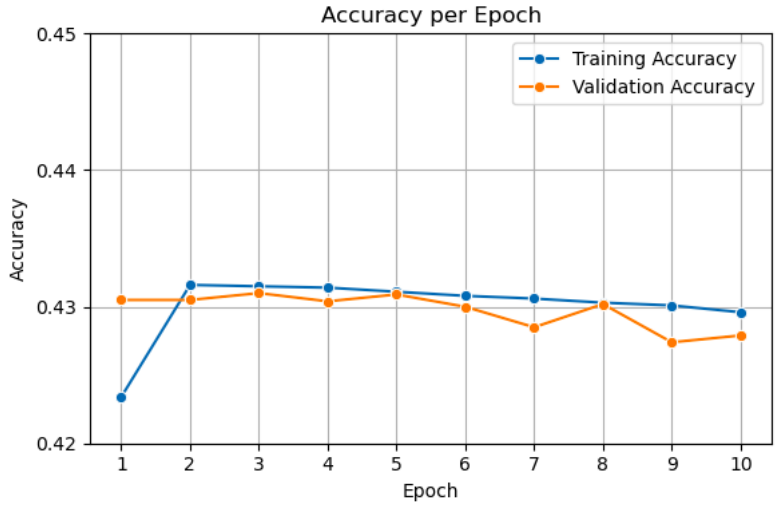


Figure 9.2. *Training results of basic 1D Convolutional Neural Network (CNN)*

For comparison purposes this model was also trained and tested on identifying the root note only. The simplicity of this architecture allows for a clear evaluation of the model's performance on the basic task of root note identification, serving as a foundational benchmark against which more complex models can be compared. By analyzing the results from this simpler model, we can better understand the potential improvements and challenges when using more advanced architectures. The only difference in the architecture is the output layer that now consist of only 13 neurons, corresponding to the 13 possible root notes. The results of the training and evaluation process can be seen in the figure 9.3 below. An accuracy of almost 60 % was achieved. The accuracy of this model is higher compared to the previous one because the task of identifying the root note is significantly simpler.

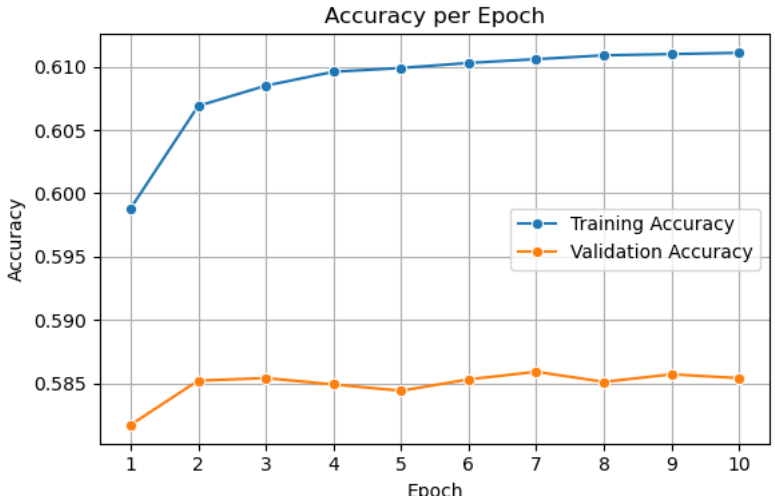


Figure 9.3. *Training results of basic 1D Convolutional Neural Network (CNN) on the task of identifying the root note.*

In the confusion matrix shown in Figure 9.4, we can observe the distribution of predicted root notes versus the actual root notes. Despite being the simplest approach, this

model is capable of accurately identifying the correct root note to a reasonable extent.

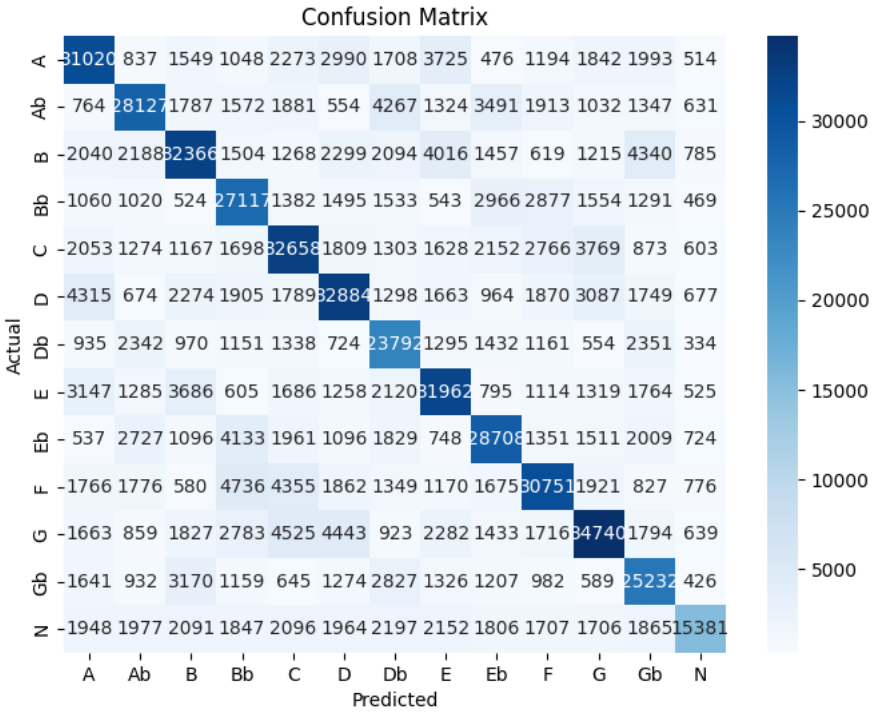


Figure 9.4. *Confusion matrix of the basic 1D Convolutional Neural Network (CNN) on the task of identifying the root note.*

9.2 Model 2: 1D Convolutional Neural Network

Continuing with the exploration of model architectures, the next model to be examined is a deeper 1D Convolutional Neural Network (CNN). This architecture aims to leverage the hierarchical features learned by successive convolutional layers to capture more complex patterns in the data.

Following experimentation with both Constant-Q Chromagram and Constant Q Transform (CQT), it was observed that the CQT approach outperformed the Constant-Q Chromagram method. The CQT method, utilizing 192 features compared to the 12 features used previously, resulted in superior performance metrics, as detailed in Chapter 8. So in the section the Constant Q Transform will be utilised using a sample rate of 44100, 192 bins (24 bins per octave), and a hop length of 4096 (those parameters were fine tuned).

The model architecture consists of several convolutional layers followed by max-pooling layers to downsample the feature maps and reduce the spatial dimensions. Dropout layers are also incorporated to prevent overfitting by randomly dropping a fraction of the input units during training. After the convolutional layers, the flattened output is passed through fully connected dense layers, which further process the extracted features before the final classification.

The problem has been subdivided into several distinct tasks, each aimed at identifying

different aspects of the musical chord:

- Identifying the root note.
- Identifying the bass note.
- Identifying the triad.
- Identifying the first extension (4th note).
- Identifying the second extension (5th note).

To address these tasks efficiently, transfer learning techniques were employed. Initially, experimentation was conducted to determine the optimal "core" architecture, which would serve as the foundation for all subsequent tasks. Following this, a multi-step approach was adopted that can be seen in the figure below 9.5. More information about Transfer Learning, can be found in the previous chapter 7.

Figure 9.5. *Flow chart of the transfer learning approach used in the Model 2.*

The detailed "core" architecture of the model, after "freezing" the layers, is summarized in Table 9.2.

Table 9.2. *Model architecture summary for the "core" convolutional neural network (CNN), after "freezing" the layers.*

Layer (type)	**Output Shape**	**Param #**
conv1d (Conv1D)	(None, 192, 16)	64
conv1d_1 (Conv1D)	(None, 192, 16)	784
max_pooling1d (MaxPooling1D)	(None, 96, 16)	0
dropout (Dropout)	(None, 96, 16)	0
conv1d_2 (Conv1D)	(None, 96, 32)	1568
conv1d_3 (Conv1D)	(None, 96, 32)	3104
max_pooling1d_1 (MaxPooling1D)	(None, 48, 32)	0
dropout_1 (Dropout)	(None, 48, 32)	0
conv1d_4 (Conv1D)	(None, 48, 64)	6208
conv1d_5 (Conv1D)	(None, 48, 64)	12352
max_pooling1d_2 (MaxPooling1D)	(None, 24, 64)	0
dropout_2 (Dropout)	(None, 24, 64)	0
conv1d_6 (Conv1D)	(None, 24, 128)	24704
conv1d_7 (Conv1D)	(None, 24, 128)	49280
max_pooling1d_3 (MaxPooling1D)	(None, 12, 128)	0
dropout_3 (Dropout)	(None, 12, 128)	0
flatten (Flatten)	(None, 1536)	0
Total params:		**98064 (383.06 KB)**
Trainable params:		**0 (0.00 Byte)**
Non-trainable params:		**98064 (383.06 KB)**

9.2.1 Training, Evaluation and Test data split

The dataset, augmented and preprocessed as described in Chapter 3, was split into three distinct sets to facilitate the testing and evaluation of the model. To maintain the coherence of the data within each track, an algorithm was designed to split the dataset on a track-by-track basis. Additionally, 20% of the data was allocated for evaluation, while the remaining 80% was designated for training purposes. This approach ensured a comprehensive evaluation while preserving the integrity of the dataset structure.

9.2.2 Model 2: Root classification task

The first task in the chord recognition pipeline focuses on identifying the root note of each chord. This task involves classifying the root note of each chord into one of the 13 possible root notes (included "None") in the musical scale. To accomplish this, the "core" architecture 9.2 was used and the model was further fine tuned on identifying the root note. Two fully connected layers were added to the model 9.3 and the model was once again trained and evaluated.

Table 9.3. *Dense layers added to the "core" architecture for the task of identifying the root note using the Model 2.*

Layer (type)	**Output Shape**	**Param #**
dense (Dense)	(None, 128)	196736
dense_1 (Dense)	(None, 13)	1677
Total params:		**296477 (1.13 MB)**
Trainable params:		**296477 (1.13 MB)**
Non-trainable params:		**0 (0.00 Byte)**

The figure below (9.6) provides a comprehensive overview of the finalized architecture of Model 2, specifically designed for the task of root note identification.

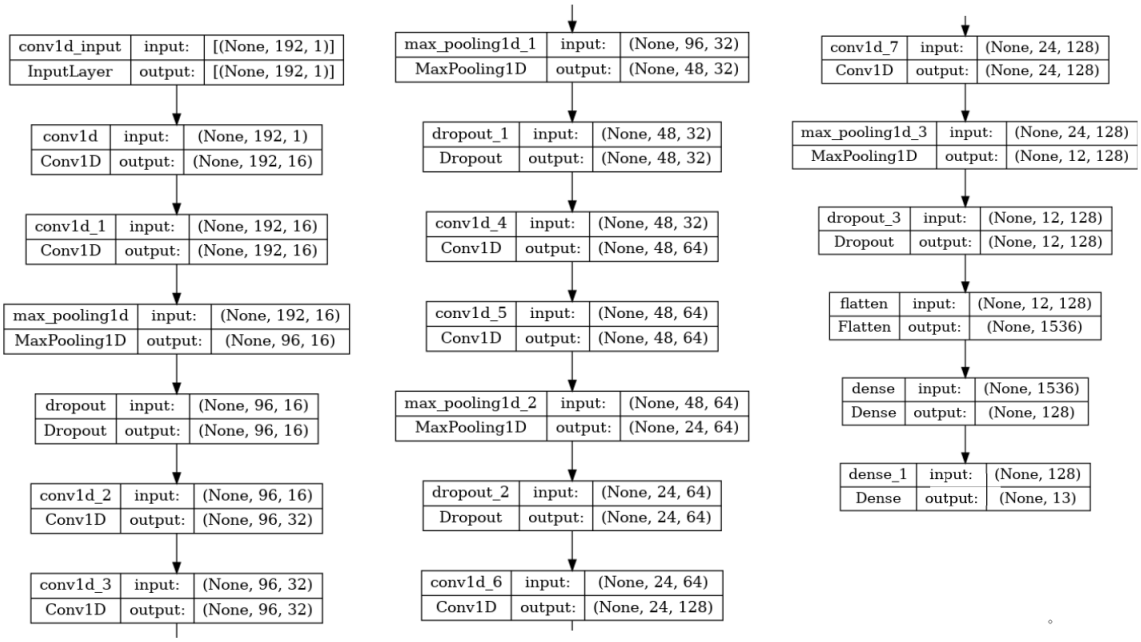


Figure 9.6. *Architecture of the Model 2 in the task of identifying the root note.*

For training the model, the batch size used is 32 with the learning rate set to 0.0001 using the Adam optimizer and the ReLU as activation function. The loss function used is the Sparse Categorical Cross-entropy. The performance of the model in identifying the root notes is depicted in the graph shown below (Figure 9.7). At Epoch 10, both the training and evaluation accuracy stabilize at approximately 67%. However, beyond this epoch, while the training accuracy marginally improves, the validation accuracy begins to decline, suggesting overfitting. Therefore, the model's state at Epoch 10 was chosen for further analysis.

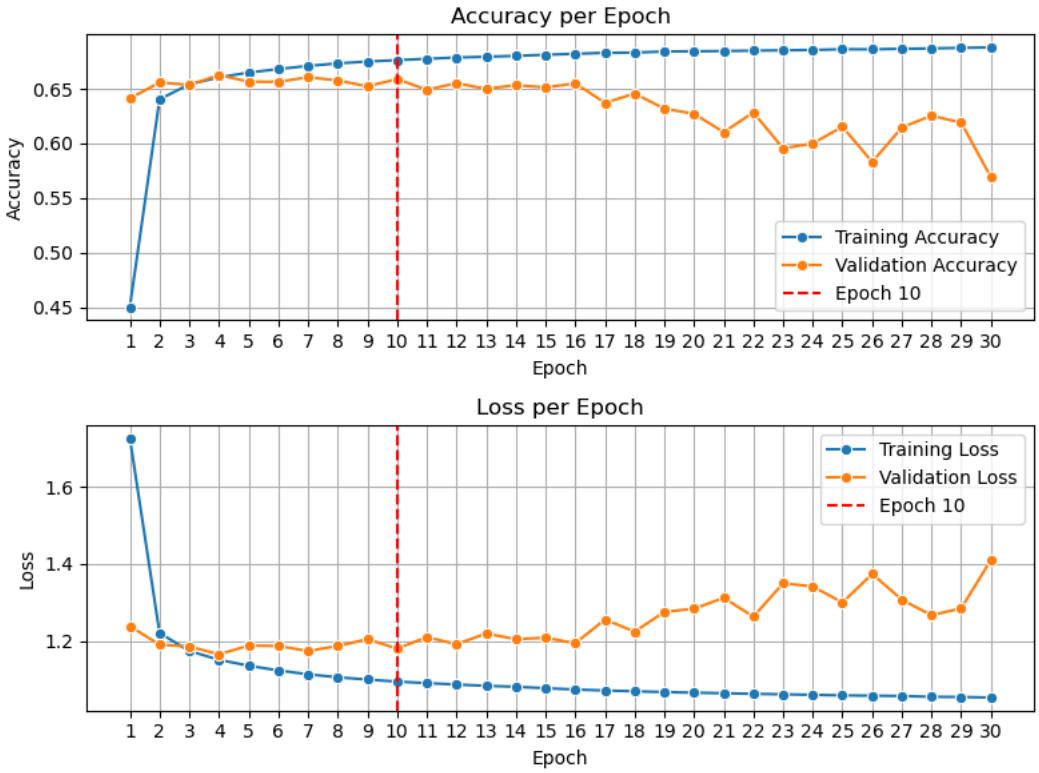


Figure 9.7. *Training results of Model 2 in the task of identifying the root note. Batch size=32, Learning Rate=0.0001*

The confusion matrix, as illustrated in Figure 9.8, offers a detailed insight into the model's performance by showcasing the distribution of predicted root notes against the actual ones. Notably, compared to Model 1, we observed a noteworthy 7% enhancement in accuracy on the evaluation set.

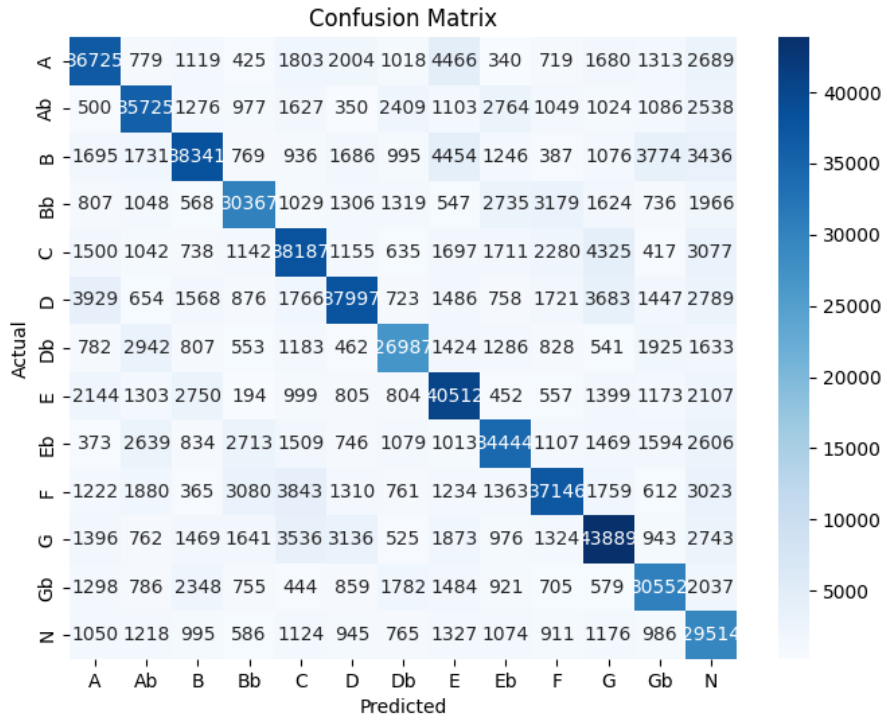


Figure 9.8. *Confusion matrix of the Model 2 on the task of identifying the root note.*

9.2.3 Model 2: Bass note classification task

The second task in the chord recognition pipeline is dedicated to identifying the bass note of each chord. Similar to the root identification task, this task involves classifying the bass note of each chord into one of the 13 possible notes (included "None") within the musical scale. To tackle this task, we leverage the "core" architecture outlined in Table 9.2, which served as the foundation for our Model 2. This architecture is further refined and adapted to specialize in identifying bass notes.

In Table 9.4, we outline the additional dense layers integrated into the core architecture to tailor it for bass note identification.

Table 9.4. *Dense layers added to the "core" architecture for the task of identifying the bass note using Model 2.*

Layer (type)	**Output Shape**	**Param #**
dense (Dense)	(None, 128)	196736
dense_1 (Dense)	(None, 13)	1677
Total params:		**296477 (1.13 MB)**
Trainable params:		**198413 (775.05 KB)**
Non-trainable params:		**98064 (383.06 KB)**

During training, the model is fine-tuned using a batch size of 32, a learning rate of 0.0001, the Adam optimizer, and the ReLU activation function. The model was trained for 10 epochs for the reasons discussed in the previous section. The loss function em-

ployed is the Sparse Categorical Cross-entropy. The performance metrics of Model 2 in identifying bass notes are visualized in Figure 9.9. The results are similar to the root note classification task due to the similarity of those two tasks.

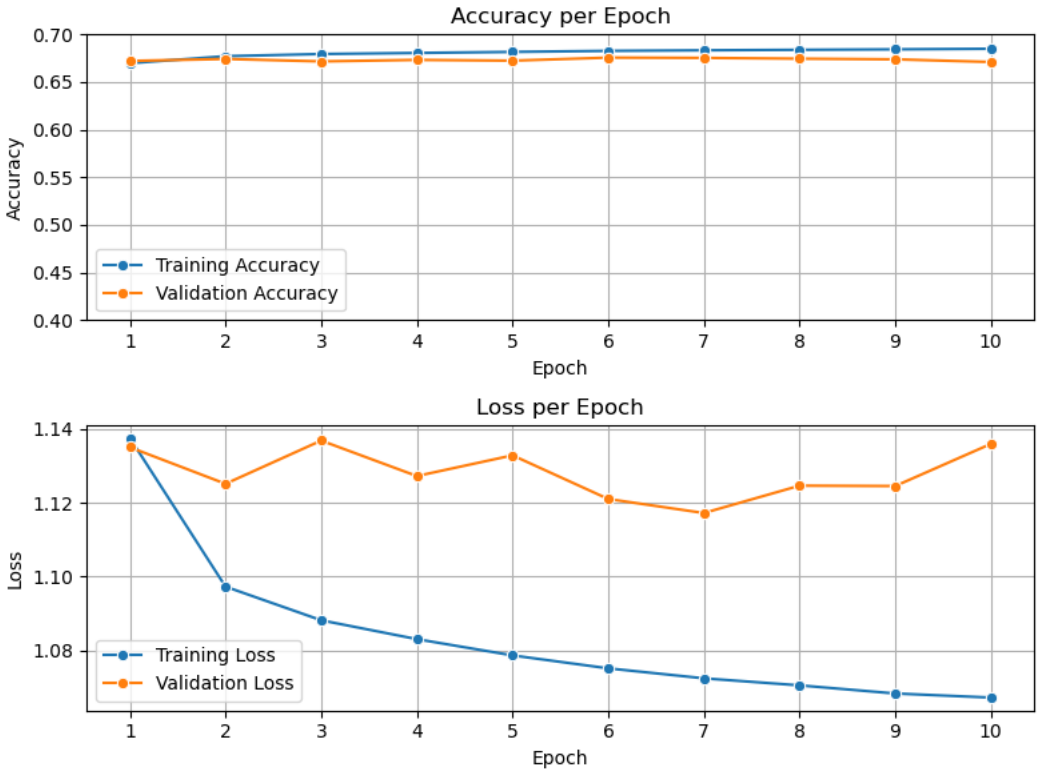


Figure 9.9. *Training results of Model 2 for bass note identification. Batch size=32, Learning Rate=0.0001*

The confusion matrix depicted in Figure 9.10 provides a detailed breakdown of the model's predictions against the ground truth for bass note identification. This visualization offers valuable insights into the model's performance and its ability to accurately classify bass notes.

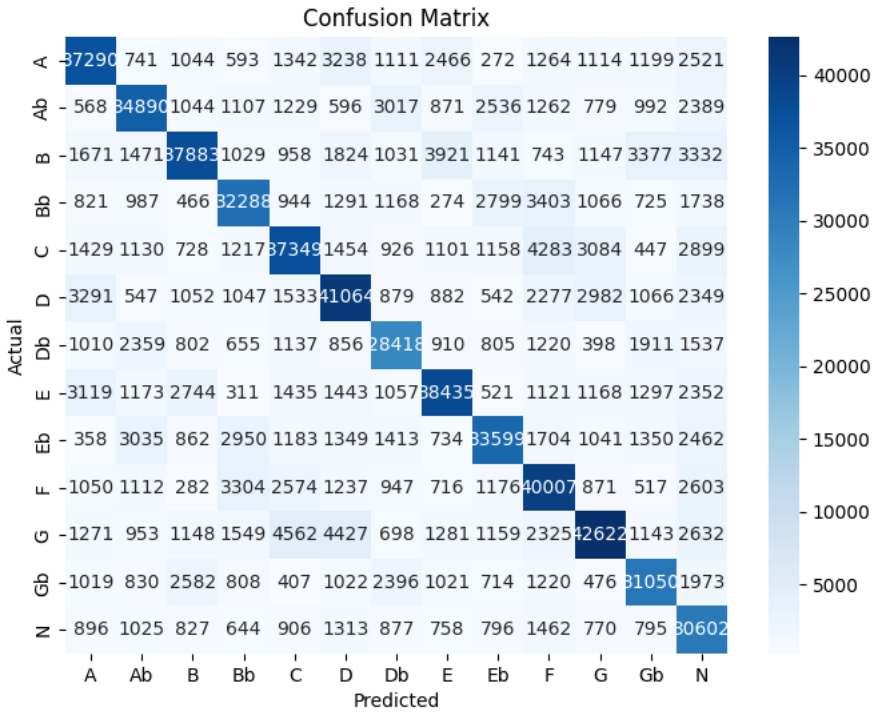


Figure 9.10. *Confusion matrix of Model 2 for bass note identification.*

9.2.4 Model 2: Triad classification Task

The third task in the chord recognition pipeline focuses on identifying the triad of each chord. This task entails classifying the triad note of each chord into one of the 7 possible triads (including "None") in the musical scale.

To accomplish this task, we leverage the "core" architecture from Model 2. We extend this architecture by adding two additional fully connected layers tailored for triad identification. We, then, fine tune the model on the task of classifying the triad note. The extra Dense layers added can be seen in the table 9.5 below.

Table 9.5. *Dense layers added to the "core" architecture for the task of identifying the triad using Model 2.*

Layer (type)	**Output Shape**	**Param #**
dense (Dense)	(None, 128)	196736
dense_1 (Dense)	(None, 7)	903
Total params:		**295703 (1.13 MB)**
Trainable params:		**197639 (772.03 KB)**
Non-trainable params:		**98064 (383.06 KB)**

Similar to previous tasks, the model is fine-tuned using a batch size of 32, a learning rate of 0.0001, the Adam optimizer, and the ReLU activation function. The model was trained for 10 epochs for the reasons discussed in the previous section. The loss function employed is the Sparse Categorical Cross-entropy. The performance metrics of Model 2

in identifying bass notes are visualized in Figure 9.11. The model reached 80% training and 76% evaluation accuracy.

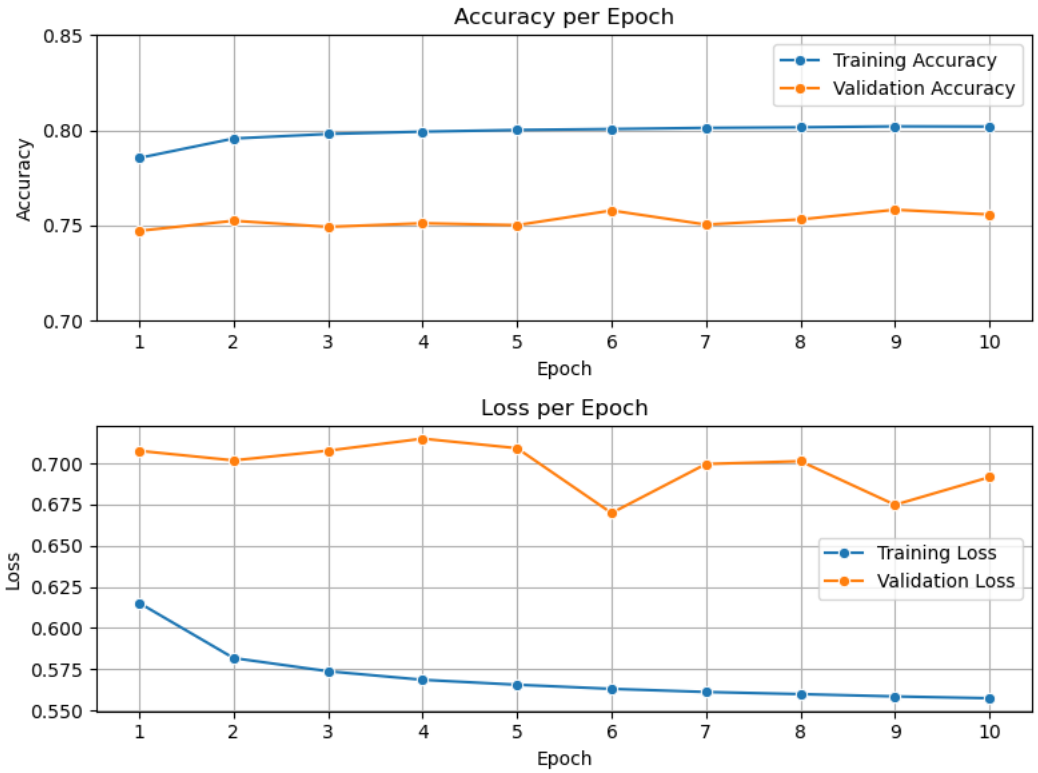


Figure 9.11. *Training results of Model 2 for triad note identification. Batch size=32, Learning Rate=0.0001*

The confusion matrix depicted in Figure 9.12 provides a detailed breakdown of the model's predictions against the ground truth for triad note identification. This visualization offers valuable insights into the model's performance and its ability to accurately classify triad notes. It is evident that the model classifies most of the chords as major chords. This bias towards major chords could be due to the imbalanced representation of major chords in the dataset. Consequently, the model's performance in accurately identifying other types of chords, such as minor or diminished chords, might be compromised. Therefore, while the model demonstrates proficiency in identifying major chords, its effectiveness in classifying other chord types is limited. This highlights the need for further investigation and potential model refinement to achieve more balanced and accurate triad note identification.

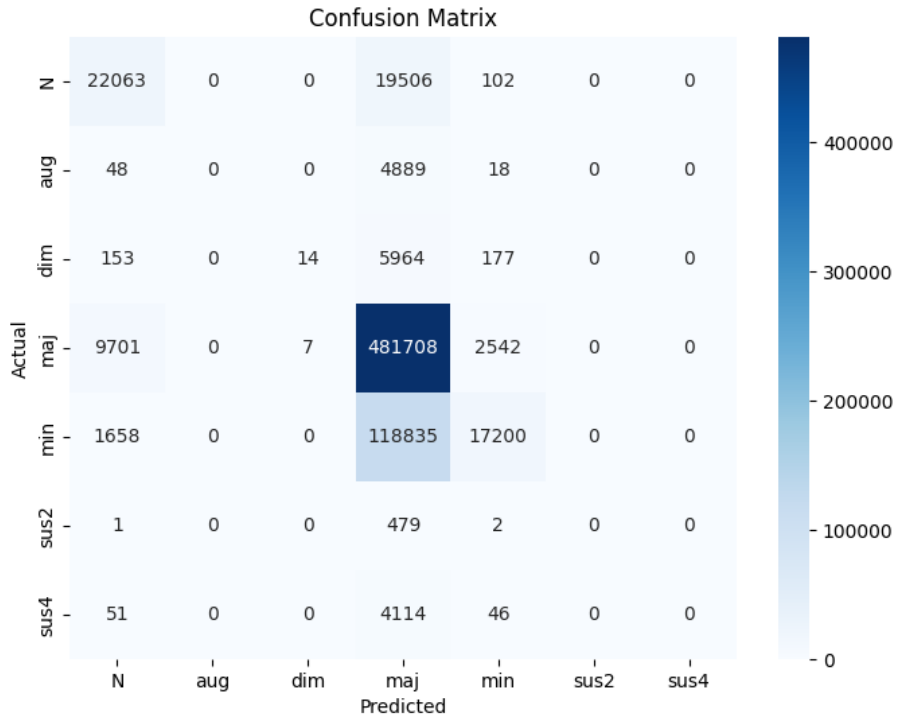


Figure 9.12. *Confusion matrix of Model 2 for triad note identification.*

9.2.5 Model 2: Extension 1 classification Task

The fourth task in the chord recognition pipeline focuses on identifying the extension 1 of each chord. Extension 1 refers to the first additional note beyond the triad in a chord. This task involves classifying the extension 1 note of each chord into one of the 6 possible extension 1 notes (including "None") in the musical scale.

To tackle this task, we build upon the "core" architecture utilized in Model 2. We extend this architecture by adding two additional fully connected layers specifically designed for extension 1 identification. The additional Dense layers integrated into the model are outlined in Table 9.6 below.

Table 9.6. *Dense layers added to the "core" architecture for the task of identifying extension 1 using Model 2.*

Layer (type)	**Output Shape**	**Param #**
dense (Dense)	(None, 128)	196736
dense_1 (Dense)	(None, 7)	774
Total params:		**295703 (1.13 MB)**
Trainable params:		**197639 (772.03 KB)**
Non-trainable params:		**98064 (383.06 KB)**

Similar to the previous tasks, the model is fine-tuned using a batch size of 32, a learning rate of 0.0001, the Adam optimizer, and the ReLU activation function. The training process spans 10 epochs to balance training time and model convergence. The

Sparse Categorical Cross-entropy is employed as the loss function.

The performance metrics of Model 2 in identifying extension 1 notes are visually represented in Figure 9.13. The model achieves a training accuracy of 85% and an evaluation accuracy of 74%.

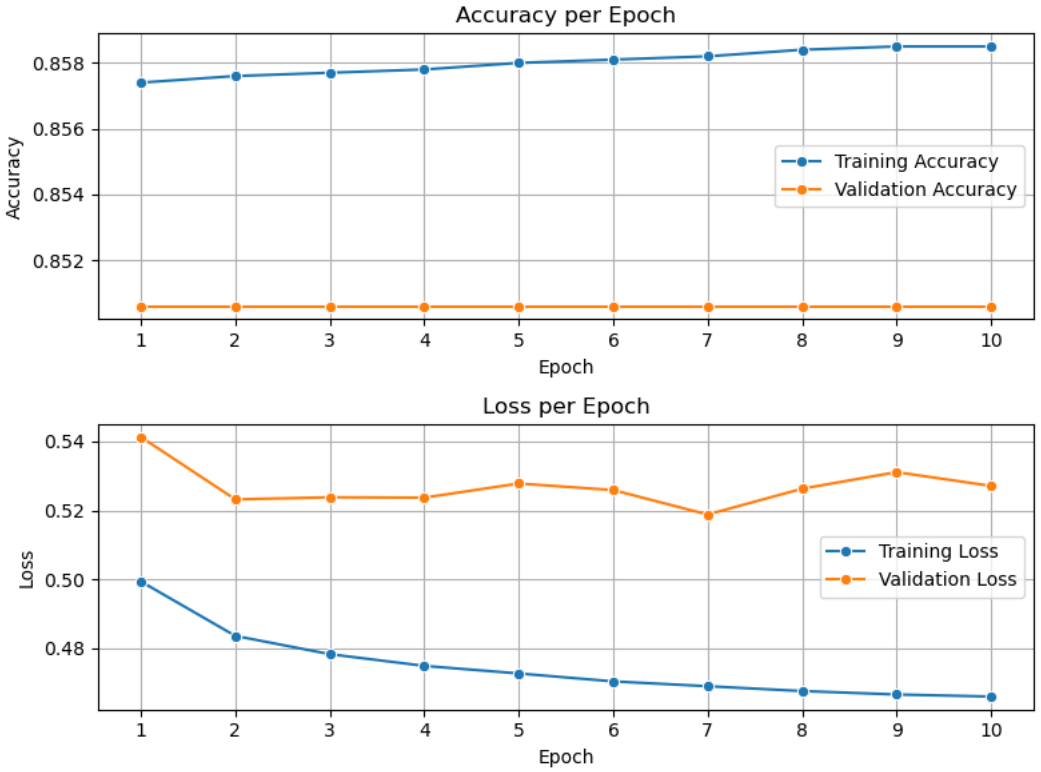


Figure 9.13. *Training results of Model 2 for extension 1 note identification. Batch size=32, Learning Rate=0.0001*

The confusion matrix depicted in Figure 9.14 provides a detailed breakdown of the model's predictions compared to the ground truth for extension 1 note identification. This visualization offers valuable insights into the model's performance and its ability to accurately classify extension 1 notes. Despite the relatively high accuracy, it is evident that the model fails completely on identifying the 4th note (extension 1). The model has predicted "None" in the majority of its predictions.

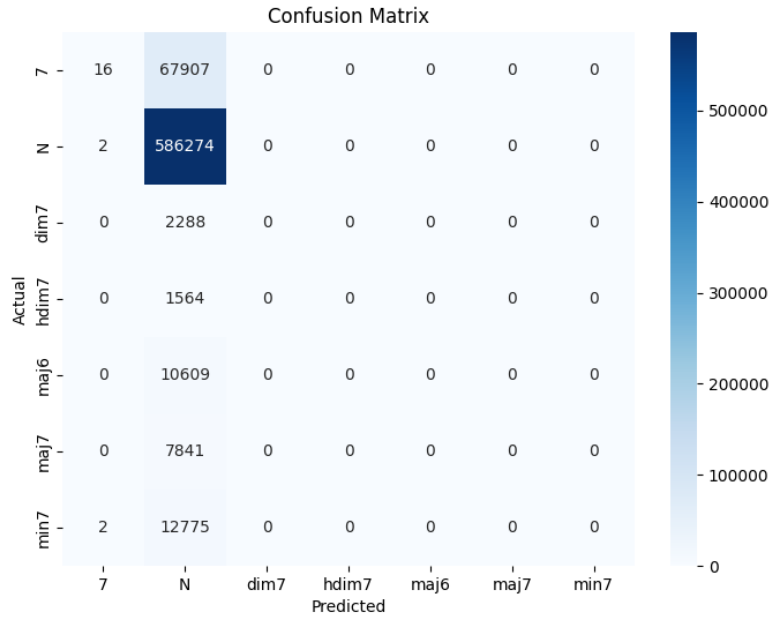


Figure 9.14. *Confusion matrix of Model 2 for extension 1 note identification.*

9.2.6 Model 2: Extension 2 classification Task

The fifth task in the chord recognition pipeline focuses on identifying the extension 2 of each chord. Extension 2 refers to the second additional note beyond the triad in a chord. This task basically involves classifying the chord as major 9 chord or not.

To address this task, we expand upon the "core" architecture utilized in Model 2. We enhance this architecture by integrating two additional fully connected layers specifically tailored for extension 2 identification. The additional Dense layers incorporated into the model are detailed in Table 9.7 below.

Table 9.7. *Dense layers added to the "core" architecture for the task of identifying extension 2 using Model 2.*

Layer (type)	**Output Shape**	**Param #**
dense (Dense)	(None, 128)	196736
dense_1 (Dense)	(None, 2)	258
Total params:		**295381 (1.13 MB)**
Trainable params:		**197317 (771.72 KB)**
Non-trainable params:		**98064 (383.06 KB)**

Following the methodology employed in previous tasks, the model is fine-tuned using a batch size of 32, a learning rate of 0.0001, the Adam optimizer, and the ReLU activation function. The training process extends over 10 epochs to balance training time and model convergence, with the Sparse Categorical Cross-entropy serving as the loss function.

The performance metrics of Model 2 in identifying extension 2 notes are visually depicted in Figure 9.15. The model achieves a training accuracy of 98% and an evaluation accuracy of 98%.

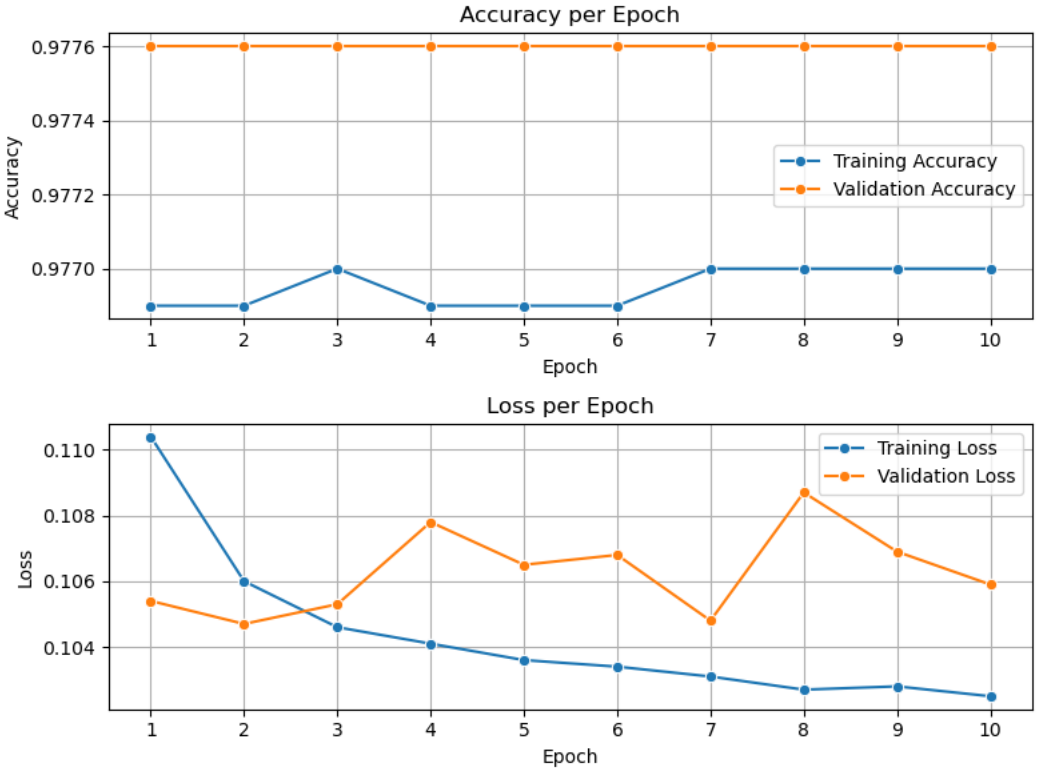


Figure 9.15. *Training results of Model 2 for extension 2 note identification. Batch size=32, Learning Rate=0.0001*

The confusion matrix illustrated in Figure 9.16 presents a comprehensive overview of the model's predictions compared to the ground truth for extension 2 note identification. This visualization provides valuable insights into the model's performance and its ability to accurately classify extension 2 note. Despite the very high accuracy, it is evident that the model fails to classify the 5th note correctly. The model has only predicted as "None" all the samples. The high imbalance of the dataset for the extension 2 has lead to the misleading high Accuracy scores.

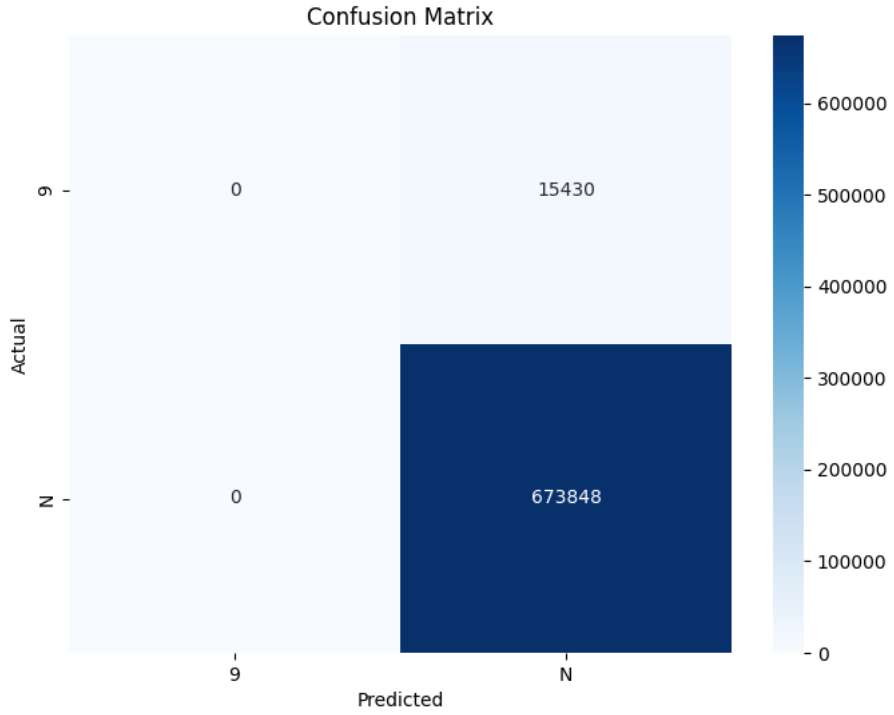


Figure 9.16. *Confusion matrix of Model 2 for extension 2 note identification.*

9.3 Model 3: 2D Convolutional Neural Network with BiLSTM

As described in 7, convolutional networks excel in handling data with a grid-like topology that possesses spatial relationships. Spectrograms, in the previous model, were interpreted as time series of vectors with 192 features. In this model, spectrograms will be treated as images, leveraging convolutional layers before the recurrent layer to facilitate feature extraction.

To achieve that, as mentioned in 8, 'chunking' will be used as a preprocessing step to prepare the data for utilization in two-dimensional deep learning models. This process involves partitioning the input data into smaller segments, each comprising 100 consecutive time steps, equivalent to approximately 7 seconds of audio. This number was treated as a hyperparameter when training the models and was determined after fine-tuning. The rationale behind this approach stems from the utilization of a bidirectional Long Short-Term Memory (LSTM) layer in our model architecture.

The use of small data chunks is imperative for enhancing the LSTM's learning efficacy and efficiency. By breaking down the input time series into manageable segments, the bidirectional LSTM layer can better capture and understand temporal dependencies within the data. This granularity allows the model to effectively learn patterns and relationships across shorter time intervals, facilitating more accurate predictions and improved performance.

By treating each chunk as an individual image, the convolutional layers can extract meaningful features within these smaller regions of time within the track. This chunking

process not only simplifies the computational complexity but also ensures that the input data is standardized, which is crucial for consistent training performance. Once the features are extracted from these chunks using the convolutional layers, the BiLSTM layer will take these processed features and analyze the sequential dependencies across chunks, providing a comprehensive understanding of the musical piece over time.

The convolutional layers in this model are designed to map the spectrogram data into an output vector. This vector is subsequently used as the input for the bidirectional Long Short-Term Memory (BiLSTM) layer. This architecture enables the BiLSTM to effectively capture and interpret contextual information over time.

Following the extensive experimentation mentioned in 9.2, it was determined that the Constant-Q Transform (CQT) method outperformed the Constant-Q Chromagram approach. By utilizing 192 features as opposed to the 12 features employed in the Constant-Q Chromagram, the CQT method achieved superior performance metrics. Therefore, in this chapter, we will utilize the Constant-Q Transform with a sample rate of 44100 Hz, 192 bins (24 bins per octave), and a hop length of 4096, parameters that were fine-tuned for optimal performance. The chunk size was tuned and set to 100 meaning each chunk covers about 7 seconds of audio.

The architecture of Model 3 consists of several convolutional layers followed by max-pooling layers, which downsample the feature maps and reduce their spatial dimensions. To prevent overfitting, dropout layers are incorporated, which randomly drop a fraction of the input units during training. The output from the convolutional layers is then flattened and passed through BiLSTM layer, which further process the extracted features before the final passing to a fully connected dense layer for classification.

Like the previous section, the problem has been subdivided into several distinct tasks, each aimed at identifying different aspects of the musical chord:

- Identifying the root note.
- Identifying the bass note.
- Identifying the triad.
- Identifying the first extension (4th note).
- Identifying the second extension (5th note).

To address these tasks efficiently, transfer learning techniques were employed. Initially, we determined the optimal "core" architecture, for each task this time, that would serve as the foundation. Following this, a multi-step approach was adopted, as illustrated in Figure 9.17. This approach facilitated the sequential transfer of learned features from one task to another, thereby enhancing the model's performance across all tasks. More details on transfer learning can be found in Chapter 7.

Figure 9.17. *Flow chart of the transfer learning approach used in the Model 3.*

9.3.1 Training, Evaluation and Test data split

The dataset, augmented and pre-processed as described in Chapter 3, was split into three distinct sets to facilitate the testing and evaluation of the model. To maintain the coherence of the data within each track, an algorithm was designed to split the dataset on a track-by-track basis. Notably, tracks from the albums *CD1*, *CD2*, *Help*, and *Please Please Me* were reserved exclusively for final testing. Additionally, 15% of the remaining data was allocated for evaluation, while the remaining 85% was designated for training purposes. This approach ensured a comprehensive evaluation while preserving the integrity of the dataset structure.

9.3.2 Model 3: Root classification task

As mentioned in 9.2.2, the first task in the chord recognition pipeline focuses on identifying the root note of each chord. This task involves classifying the root note of each chord into one of the 13 possible root notes (included "None") in the musical scale. To accomplish this, several architectures were tried but the one yielding the best results can be seen in the table below **??**

Table 9.8. *Model Architecture for the task of identifying the root note using the Model 3.*

Layer (type)	**Output Shape**
conv2d (Conv2D)	(None, 100, 192, 32)
conv2d_1 (Conv2D)	(None, 100, 192, 32)
conv2d_2 (Conv2D)	(None, 100, 192, 32)
conv2d_3 (Conv2D)	(None, 100, 192, 32)
batch_normalization (BatchNormalization)	(None, 100, 192, 32)
max_pooling2d (MaxPooling2D)	(None, 100, 96, 32)
dropout (Dropout)	(None, 100, 96, 32)
conv2d_4 (Conv2D)	(None, 100, 96, 64)
conv2d_5 (Conv2D)	(None, 100, 96, 64)
batch_normalization_1 (BatchNormalization)	(None, 100, 96, 64)
max_pooling2d_1 (MaxPooling2D)	(None, 100, 48, 64)
dropout_1 (Dropout)	(None, 100, 48, 64)
conv2d_6 (Conv2D)	(None, 100, 48, 128)
conv2d_7 (Conv2D)	(None, 100, 48, 64)
batch_normalization_2 (BatchNormalization)	(None, 100, 48, 64)
max_pooling2d_2 (MaxPooling2D)	(None, 100, 12, 64)
dropout_2 (Dropout)	(None, 100, 12, 64)
time_distributed (TimeDistributed)	(None, 100, 768)
bidirectional (Bidirectional)	(None, 100, 256)
time_distributed_1 (TimeDistributed)	(None, 100, 13)

Training the Model on Root Task

For training the model, it was compiled using the Adam optimizer with the following hyperparameters, learning rate of 0.0001, β_1 set to 0.9, β_2 set to 0.99, and epsilon (ϵ) set to 1×10^{-8}. The loss function employed for training was categorical crossentropy, and the model's performance was evaluated using accuracy as the metric and the batch size used is 16. The parameter β_1 represents the exponential decay rate for the first moment estimates, which essentially determines how quickly the optimizer forgets past gradients and focuses on the current gradients. Typically, β_1 is set close to 1.0. Similarly, β_2 represents the exponential decay rate for the second-moment estimates, indicating how quickly the optimizer adjusts its learning rate based on the variance of the gradients. Like β_1, β_2 is also commonly set close to 1.0. Finally, ϵ is a small constant added to the denominator to prevent division by zero, ensuring numerical stability during the computation of adaptive learning rates. The loss function used is the Categorical Cross-entropy.

The performance of the model in classifying the root notes is shown in the graph shown below (Figure 9.18). The model was trained for 40 Epochs after experimenting with other options. Both the training and the evaluation accuracy stabilize at approximately 93% and 86% accordingly, a much better result in comparison to Model 2.

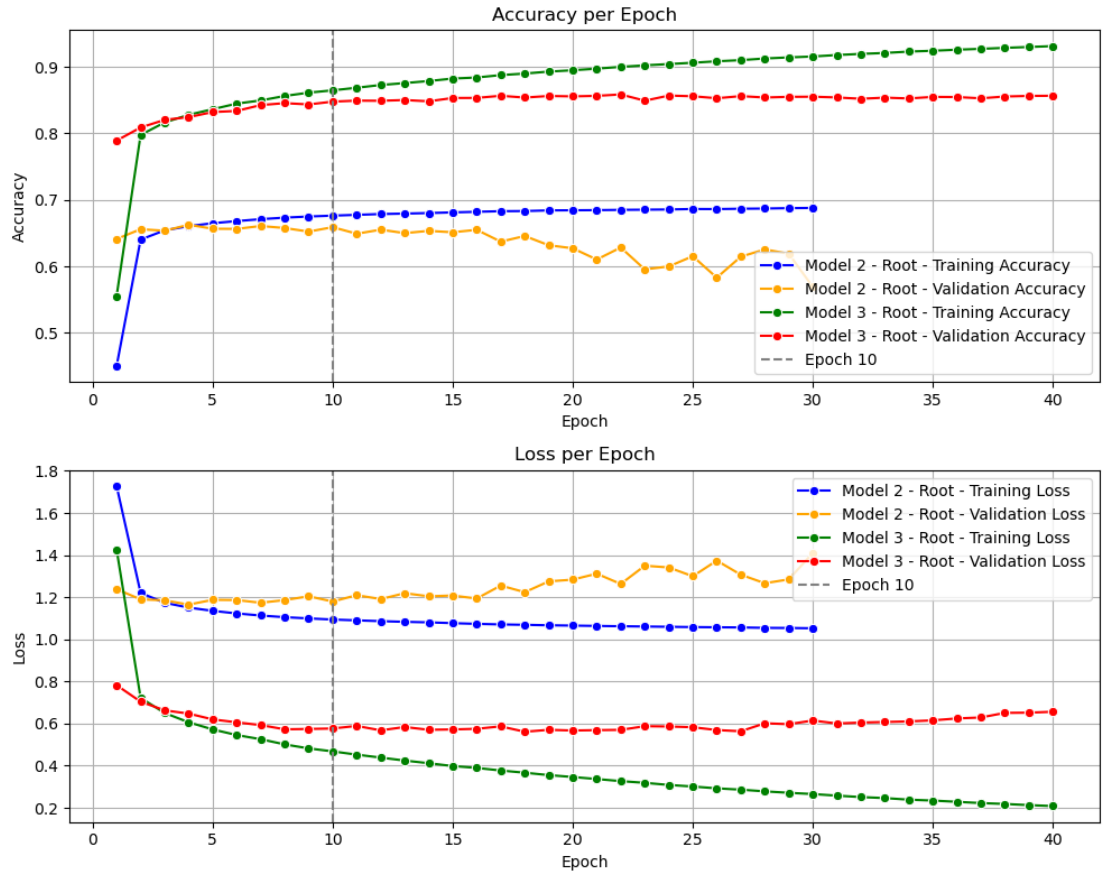


Figure 9.18. *Training results of Model 3 in the task of identifying the root note. Batch size=16, Learning Rate=0.0001, chunk size=100*

The confusion matrix, as illustrated in Figure 9.19, offers a detailed insight into the model's performance in the evalutation set by showcasing the distribution of predicted root notes against the actual ones. Notably, compared to Model 2, we observed an excessive 19% enhancement in accuracy on the evaluation set.

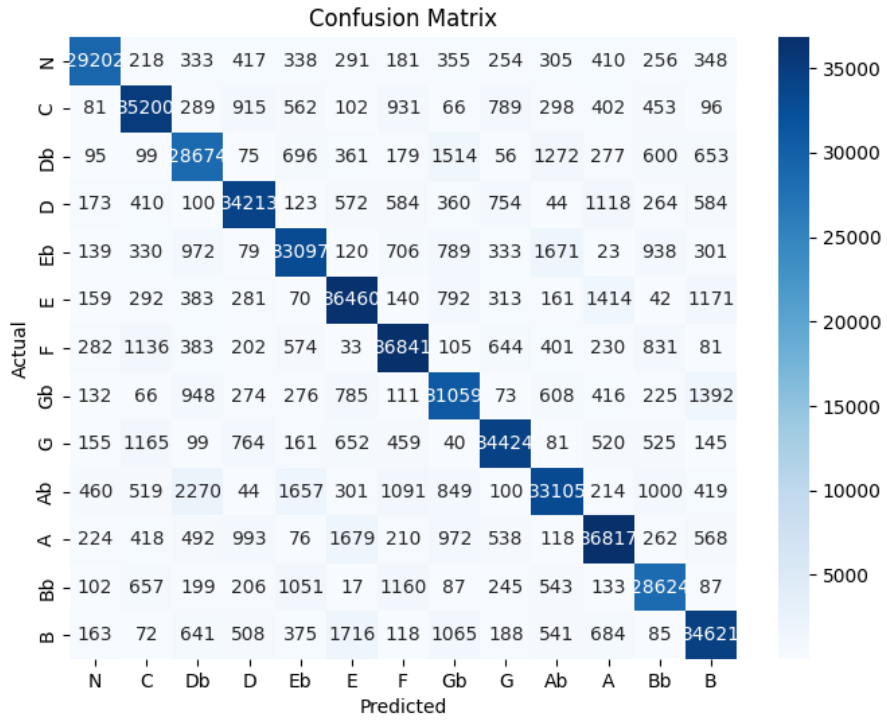


Figure 9.19. *Confusion matrix of the Model 3 on the task of identifying the root note on the evaluation set.*

A classification report table 9.9 provides a detailed classification report for a multiclass classification task, evaluating the performance of a model across various classes (N, C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb, B) based on precision, recall, F1-score, and support metrics.

The table reveals that the model achieves high precision, recall, and F1-score across most classes, indicating its effectiveness in correctly identifying musical chord classes. Overall, the model demonstrates strong performance, with an accuracy of 0.86, indicating that it correctly predicts the majority of classes with high confidence. These metrics provide valuable insights into the model's performance and can guide further optimization efforts.

	Precision	Recall	F1-score	Support
N	0.93	0.89	0.91	32908
C	0.87	0.88	0.87	40184
Db	0.80	0.83	0.82	34551
D	0.88	0.87	0.87	39299
Eb	0.85	0.84	0.84	39498
E	0.85	0.87	0.86	41678
F	0.86	0.88	0.87	41743
Gb	0.82	0.85	0.83	36365
G	0.89	0.88	0.88	39190
Ab	0.85	0.79	0.82	42029
A	0.86	0.85	0.86	43367
Bb	0.84	0.86	0.85	33111
B	0.86	0.85	0.85	40777
Accuracy			0.86	504700
Macro	0.86	0.86	0.86	504700
Weighted avg	0.86	0.86	0.86	504700

Table 9.9. *Classification Report for Model 3 on the task of identifying the root note on the evaluation set.*

The Model's performance on the test set mentioned in 9.3.1 can be seen on the Figure 9.20 below.

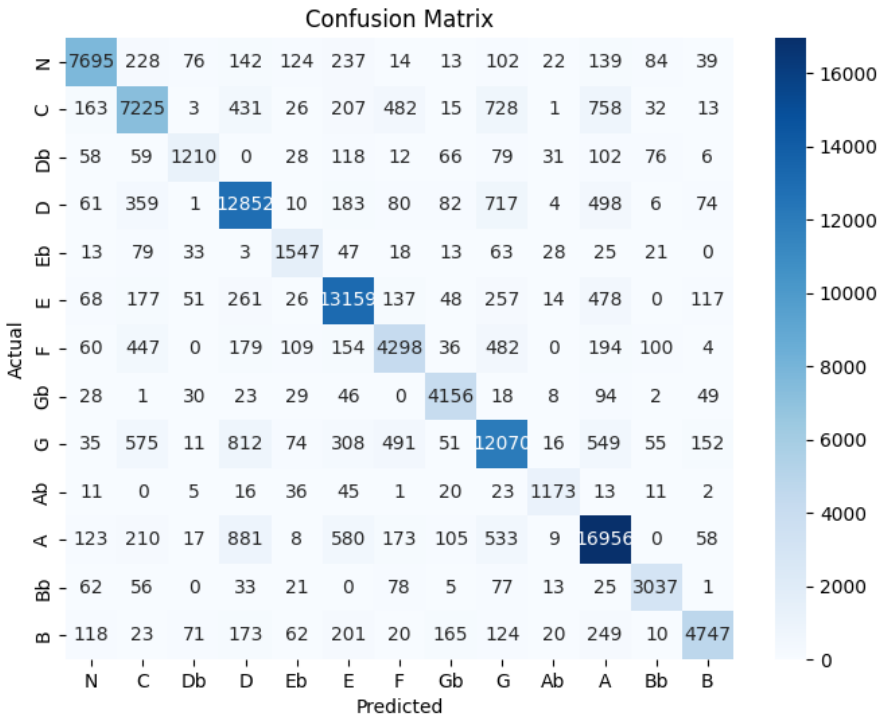


Figure 9.20. *Confusion matrix of the Model 3 on the task of identifying the root note on the test set.*

The table 9.10 below illustrates the classification report for Model 3, tasked with

identifying the root note on the evaluation set. It showcases precision, recall, F1-score, and support metrics for each class, along with overall accuracy, macro-averaged metrics, and weighted-averaged metrics. Overall, the model has about 83% Accuracy indicating a relatively high performance on the test set, noting that the test set only consists of 108600 samples in comparison to 504700 in the evaluaton set.

	Precision	Recall	F1-score	Support
N	0.91	0.86	0.88	8915
C	0.77	0.72	0.74	10084
Db	0.80	0.66	0.72	1845
D	0.81	0.86	0.84	14927
Eb	0.74	0.82	0.78	1890
E	0.86	0.89	0.87	14793
F	0.74	0.71	0.72	6063
Gb	0.87	0.93	0.90	4484
G	0.79	0.79	0.79	15199
Ab	0.88	0.87	0.87	1356
A	0.84	0.86	0.85	19653
Bb	0.88	0.89	0.89	3408
B	0.90	0.79	0.84	5983
Accuracy			0.83	108600
Macro	0.83	0.82	0.82	108600
Weighted avg	0.83	0.83	0.83	108600

Table 9.10. *Classification Report for Model 3 on the task of identifying the root note on the test set.*

As previously noted and clearly illustrated in Tables 9.9 and 9.10, the dataset is imbalanced, with certain root notes occurring less frequently than others. This imbalance can lead to a biased model that performs well on more frequent classes while underperforming on less frequent ones. To address this issue, we experimented with setting weights for each note based on its frequency of occurrence. By assigning higher weights to less frequent notes and lower weights to more frequent ones, we aimed to balance the influence of each class during the training process, thereby mitigating any biases introduced by the uneven distribution of the data.

The weighted approach intended to promote a more equitable learning process, ensuring that the model gives appropriate attention to the underrepresented classes. However, as shown in Figure 9.21 , the introduction of weights did not yield the expected improvement in model performance. Contrary to our objectives, it resulted in a slight decrease of approximately 1% - 2% in accuracy for both the training and evaluation sets. This outcome suggests that while the weighted approach addresses class imbalance, it does not contribute in the better outcome of the task. That might be cause we have already addressed major imbalances by data augmentation, detailed in 8.

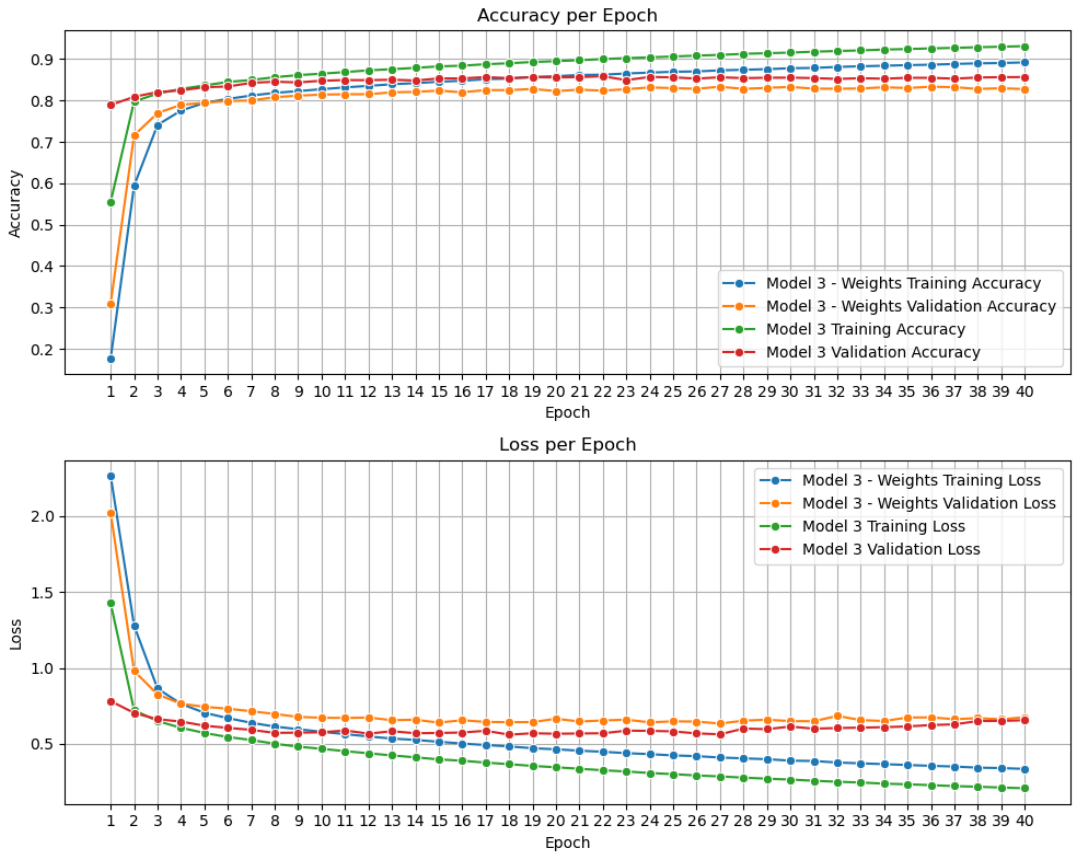


Figure 9.21. *Training results of Model 3 in the task of identifying the root note in comparison with same model but also using weights. Batch size=16, Learning Rate=0.0001, chunk size=100*

With this architecture in place, we will now proceed to the second task: classifying the bass note.

9.3.3 Model 3: Bass Classification Task

With the task of the root note classification completed, we now shift our focus to the bass note classification task. This task bears a strong resemblance to the root classification task in terms of its fundamental objective: identifying one note from the set of all possible notes, including the "None" category.

The architecture and approach for the bass classification task will largely build upon the architecture used in the previous root classification task, employing transfer learning techniques as detailed in 8. The decision on which parts of the model to reuse for transfer learning was derived from extensive experimentation, involving various combinations to identify the optimal configuration. The architecture that yielded the best results involved retaining and freezing the first 11 layers, which primarily consist of convolutional layers. To this foundation, we added 2 additional convolutional layers, a bidirectional LSTM (BiLSTM) layer, and a final fully connected dense layer. The detailed architecture is depicted in the following Tables 9.11 and 9.12 below.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 100, 192, 32)	320
conv2d_1 (Conv2D)	(None, 100, 192, 32)	9248
conv2d_2 (Conv2D)	(None, 100, 192, 32)	9248
conv2d_3 (Conv2D)	(None, 100, 192, 32)	9248
batch_normalization (BatchNormalization)	(None, 100, 192, 32)	128
max_pooling2d (MaxPooling2D)	(None, 100, 96, 32)	0
dropout (Dropout)	(None, 100, 96, 32)	0
conv2d_4 (Conv2D)	(None, 100, 96, 64)	18496
conv2d_5 (Conv2D)	(None, 100, 96, 64)	36928
batch_normalization_1 (BatchNormalization)	(None, 100, 96, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 100, 48, 64)	0
Total params		83872
Trainable params		0
Non-trainable params		83872

Table 9.11. *Detailed Architecture used as 'core' for Transfer Learning for the Bass Classification Task. Those layers are frozen meaning the paramteters will not be trained again.*

By freezing those layers, meaning those layers will not be trained again when fine-tuning for the bass classification task, we aim to retain the learned features from the root classification task, which are likely to be beneficial for the bass classification as well due to the similarity of those two tasks. This approach helps in reducing the computational cost and time required for training, as well as prevents overfitting by avoiding the need to learn new representations for the shared features. By incorporating transfer learning in this manner, we leverage the knowledge gained from the previous task and adapt it to the new task, potentially improving the overall performance of the model. The final Architecture of the model is shown in the Table 9.12 below.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 100, 192, 32)	320
conv2d_1 (Conv2D)	(None, 100, 192, 32)	9248
conv2d_2 (Conv2D)	(None, 100, 192, 32)	9248
conv2d_3 (Conv2D)	(None, 100, 192, 32)	9248
batch_normalization (BatchNormalization)	(None, 100, 192, 32)	128
max_pooling2d (MaxPooling2D)	(None, 100, 96, 32)	0
dropout (Dropout)	(None, 100, 96, 32)	0
conv2d_4 (Conv2D)	(None, 100, 96, 64)	18496
conv2d_5 (Conv2D)	(None, 100, 96, 64)	36928
batch_normalization_1 (BatchNormalization)	(None, 100, 96, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 100, 48, 64)	0
bass_conv (Conv2D)	(None, 100, 48, 64)	36928
bass_conv2 (Conv2D)	(None, 100, 48, 128)	73856
BatchNorm_bass (BatchNormalization)	(None, 100, 48, 128)	512
MaxPooling_bass (MaxPooling2D)	(None, 100, 16, 128)	0
Dropout_bass (Dropout)	(None, 100, 16, 128)	0
Flatten_bass (TimeDistributed)	(None, 100, 2048)	0
LSTM_layer (Bidirectional)	(None, 100, 256)	2229248
out (TimeDistributed)	(None, 100, 13)	3341
Total params		2427757
Trainable params		2343629
Non-trainable params		84128

Table 9.12. *Detailed Architecture for the Bass Classification Task including all layers (frozen and added).*

Training the Model on Bass Task

For training the model, as mentioned before, it was compiled using the Adam optimizer with the following hyperparameters, learning rate of 0.0001, β_1 set to 0.9, β_2 set to 0.99, and epsilon (ϵ) set to 1×10^{-8}. The loss function employed for training was categorical crossentropy, and the model's performance was evaluated using accuracy as the metric and the batch size used is 16. The performance of the model in classifying the root notes is shown in the graph shown below (Figure 9.22). The model was trained for 30 Epochs and the model at Epoch 20 found to be the best one due to the validation loss increase and the decrease of training loss after that Epoch indicating overfitting. Both the training and the evaluation accuracy stabilize at approximately 95% and 86% accordingly, a much better result in comparison to Model 2.

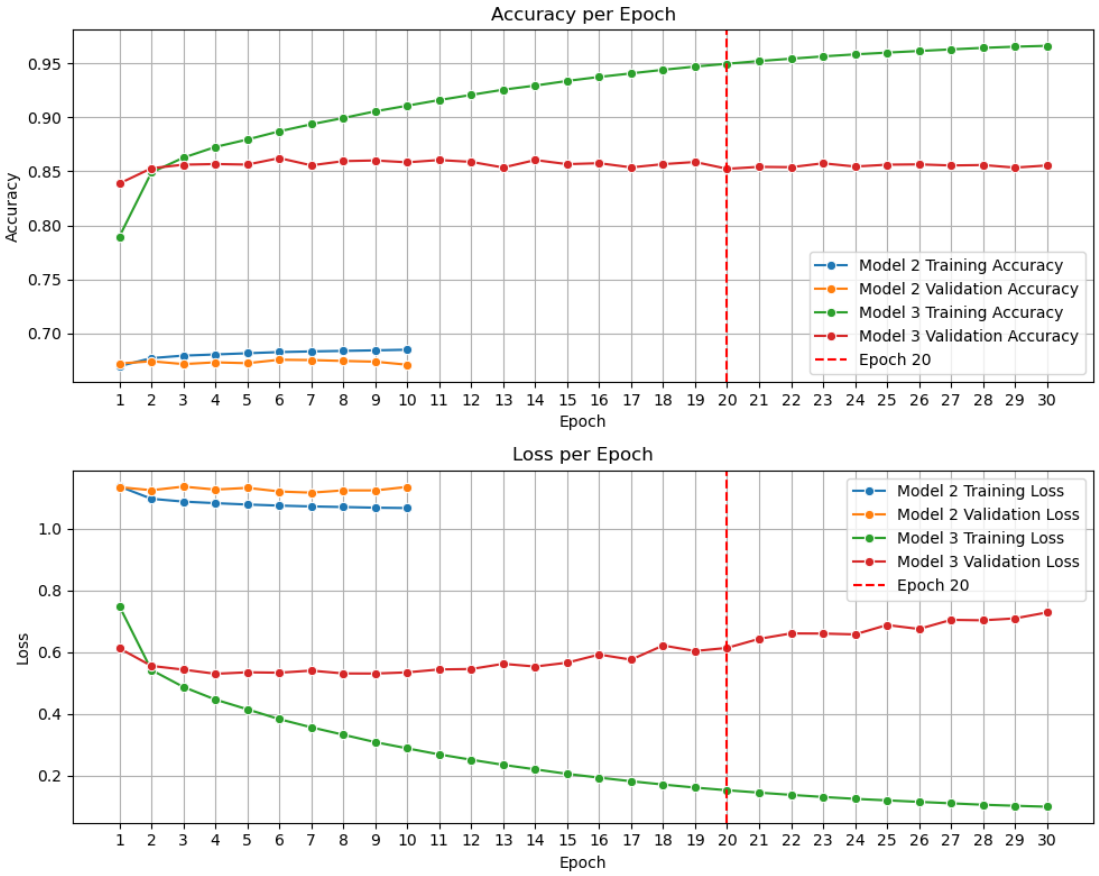


Figure 9.22. *Training results of Model 3 in the task of identifying the bass note. Batch size=16, Learning Rate=0.0001, chunk size=100*

The confusion matrix, depicted in Figure 9.23, provides a detailed overview of the model's performance on the evaluation set, showcasing the distribution of predicted bass notes compared to the actual ones. Impressively, in contrast to Model 2, we observed a substantial 19% improvement in accuracy on the evaluation set.

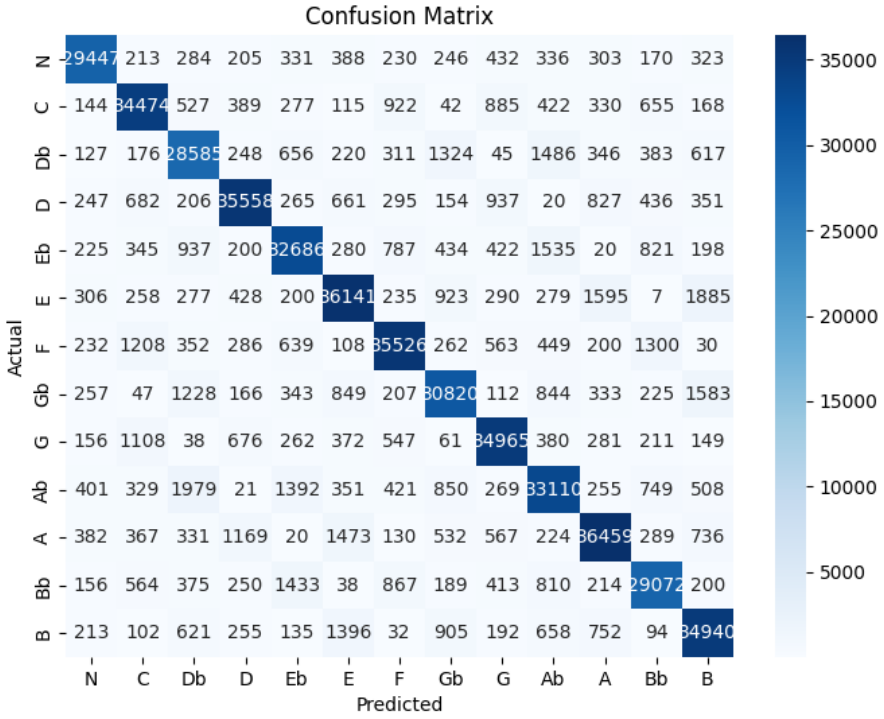


Figure 9.23. *Confusion matrix of Model 3 for identifying the bass note on the evaluation set.*

A classification report table (Table 9.13) presents a comprehensive breakdown of the model's performance in a multiclass classification task, evaluating its accuracy across various bass note classes (e.g., A, Bb, C, Db, D, etc.) based on precision, recall, F1-score, and support metrics.

The table reveals that the model achieves high precision, recall, and F1-score across most bass note classes, indicating its effectiveness in correctly identifying them. Overall, the model demonstrates strong performance, with an accuracy of 0.86, indicating that it correctly predicts the majority of bass note classes with high confidence. These metrics offer valuable insights into the model's effectiveness and can inform further optimization efforts.

	Precision	Recall	F1-score	Support
N	0.91	0.89	0.90	32908
C	0.86	0.88	0.87	39350
Db	0.80	0.83	0.81	34524
D	0.89	0.87	0.88	40639
Eb	0.85	0.84	0.84	38890
E	0.85	0.84	0.85	42824
F	0.88	0.86	0.87	41155
Gb	0.84	0.83	0.84	37014
G	0.87	0.89	0.88	39206
Ab	0.82	0.81	0.82	40635
A	0.87	0.85	0.86	42679
Bb	0.84	0.84	0.84	34581
B	0.84	0.87	0.85	40295
Accuracy			0.86	504700
Macro	0.86	0.86	0.86	504700
Weighted avg	0.86	0.86	0.86	504700

Table 9.13. *Classification Report for Model 3 on the task of identifying the bass note on the evaluation set.*

The model's performance on the test set, mentioned in Section 9.3.1, is illustrated in Figure 9.36 below.

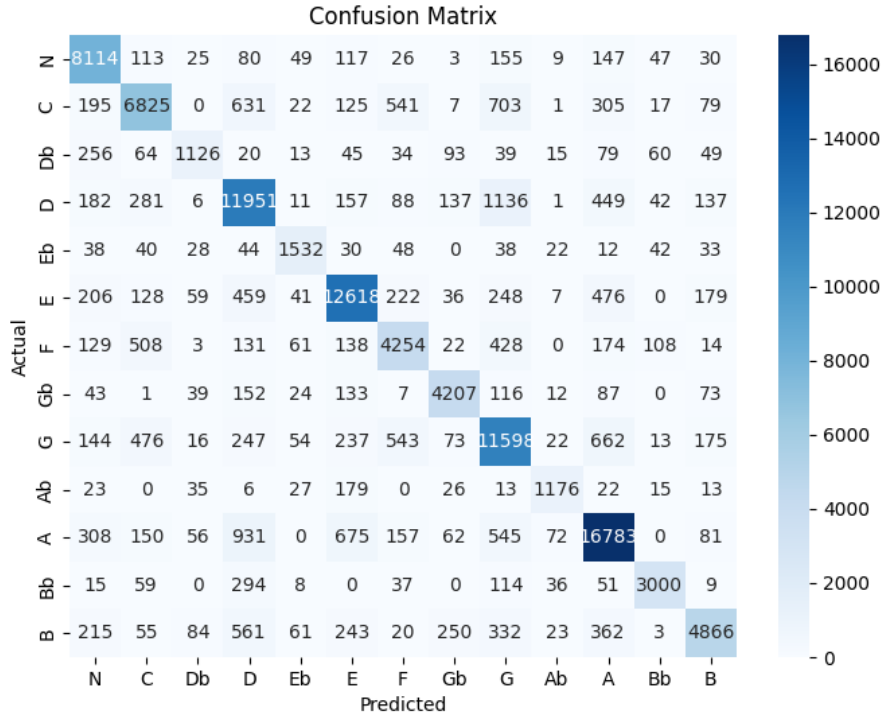


Figure 9.24. *Confusion matrix of Model 3 for identifying bass notes on the test set.*

Table 9.14 presents the classification report for Model 3, tasked with identifying bass notes on the evaluation set. It showcases precision, recall, F1-score, and support met-

rics for each bass note class, along with overall accuracy, macro-averaged metrics, and weighted-averaged metrics. Overall, the model achieves an accuracy of approximately 81%, indicating a relatively high performance on the test set, considering its size of 108,600 samples compared to the evaluation set's 504,700 samples.

	Precision	Recall	F1-score	Support
N	0.82	0.91	0.86	8915
C	0.78	0.72	0.75	9451
Db	0.76	0.59	0.67	1893
D	0.77	0.82	0.79	14578
Eb	0.81	0.80	0.80	1907
E	0.86	0.86	0.86	14679
F	0.71	0.71	0.71	5970
Gb	0.86	0.86	0.86	4894
G	0.75	0.81	0.78	14260
Ab	0.84	0.77	0.80	1535
A	0.86	0.85	0.85	19820
Bb	0.90	0.83	0.86	3623
B	0.85	0.69	0.76	7075
Accuracy			0.81	108600
Macro	0.81	0.79	0.80	108600
Weighted avg	0.81	0.81	0.81	108600

Table 9.14. *Classification Report for Model 3 on the task of identifying the bass note on the test set.*

Now we will move forward to the task of identifying the triad note.

9.3.4 Model 3: Triad Classification Task

With the successful completion of the bass note classification task, our attention now turns to the triad classification task. Similar to the previous classification tasks, the primary objective here remains the identification of a single triad from the comprehensive set of possible triads, including the "None" category. Those, as mentioned previously, are N, maj, min, dim, aug, sus2 and sus4.

For the triad classification task, we build upon the architecture and methodology established in the previous classification tasks, utilizing transfer learning techniques outlined in prior work. Extensive experimentation guided our decision-making process, where we explored various configurations to determine the most effective approach. Ultimately, the optimal architecture involved retaining and freezing the initial 11 layers, as in the previous task, predominantly comprising convolutional layers. To this framework, we introduced two additional convolutional layers, dropout and flattend layers, a bidirectional LSTM (BiLSTM) layer, and a final fully connected dense layer.

The detailed architecture can be seen in the Tables 9.15 and 9.16 below.

Layer (type)	Output Shape	Param #
conv2d	(None, 100, 192, 32)	320
conv2d_1	(None, 100, 192, 32)	9248
conv2d_2	(None, 100, 192, 32)	9248
conv2d_3	(None, 100, 192, 32)	9248
batch_normalization	(None, 100, 192, 32)	128
max_pooling2d	(None, 100, 96, 32)	0
dropout	(None, 100, 96, 32)	0
conv2d_4	(None, 100, 96, 64)	18496
conv2d_5	(None, 100, 96, 64)	36928
batch_normalization_1	(None, 100, 96, 64)	256
max_pooling2d_1	(None, 100, 48, 64)	0
Total params		**83872**
Trainable params		**0 (0.00 Byte)**
Non-trainable params		**83872 (327.62 KB)**

Table 9.15. *Detailed Architecture used as 'core' for Transfer Learning for the Triad Classification Task. Those layers are frozen meaning the parameters will not be trained again.*

Layer (type)	Output Shape	Param #
conv2d	(None, 100, 192, 32)	320
conv2d_1	(None, 100, 192, 32)	9248
conv2d_2	(None, 100, 192, 32)	9248
conv2d_3	(None, 100, 192, 32)	9248
batch_normalization	(None, 100, 192, 32)	128
max_pooling2d	(None, 100, 96, 32)	0
dropout	(None, 100, 96, 32)	0
conv2d_4	(None, 100, 96, 64)	18496
conv2d_5	(None, 100, 96, 64)	36928
batch_normalization_1	(None, 100, 96, 64)	256
max_pooling2d_1	(None, 100, 48, 64)	0
triad_conv	(None, 100, 48, 64)	36928
triad_conv2	(None, 100, 48, 128)	73856
BatchNorm_triad	(None, 100, 48, 128)	512
MaxPooling_triad	(None, 100, 16, 128)	0
Dropout_triad	(None, 100, 16, 128)	0
Flatten_Triad	(None, 100, 2048)	0
LSTM_layer (Bidirectional)	(None, 100, 256)	2229248
out (TimeDistributed)	(None, 100, 7)	1799
Total params		**2426215**
Trainable params		**2342087 (8.93 MB)**
Non-trainable params		**84128 (328.62 KB)**

Table 9.16. *Detailed Architecture for the Triad Classification Task including all layers (frozen and added).*

As mentioned before, by preserving the parameters of these layers, they are effectively excluded from further training during the adaptation process for the triad classification task. This strategy aims to preserve the learned patterns from the root classification task, which are likely to be advantageous for the triad classification task due to the

intrinsic similarities between the two tasks. This methodology not only streamlines the computational burden and training time but also guards against overfitting by obviating the necessity to relearn features already captured in the shared layers. Employing transfer learning in this manner enables us to capitalize on the insights gleaned from prior tasks and tailor them to the specific demands of the new task, potentially enhancing the overall efficacy of the model. The finalized model architecture is detailed in Table 9.16.

Training the Model on Triad Task

For training the model, we utilized the Adam optimizer with a learning rate of 0.0001, β_1 set to 0.9, β_2 set to 0.99, and epsilon (ϵ) set to 1×10^{-8}. The loss function employed for training was categorical crossentropy, and we evaluated the model's performance using accuracy as the metric with a batch size of 16.

The performance of the model in classifying the triad notes is shown in Figure 9.25. The model underwent training for 30 epochs, where it demonstrated significant improvement in accuracy over time. Both the training and validation accuracy stabilized at approximately 99% and 96%, respectively, showcasing notable enhancement compared to previous models.

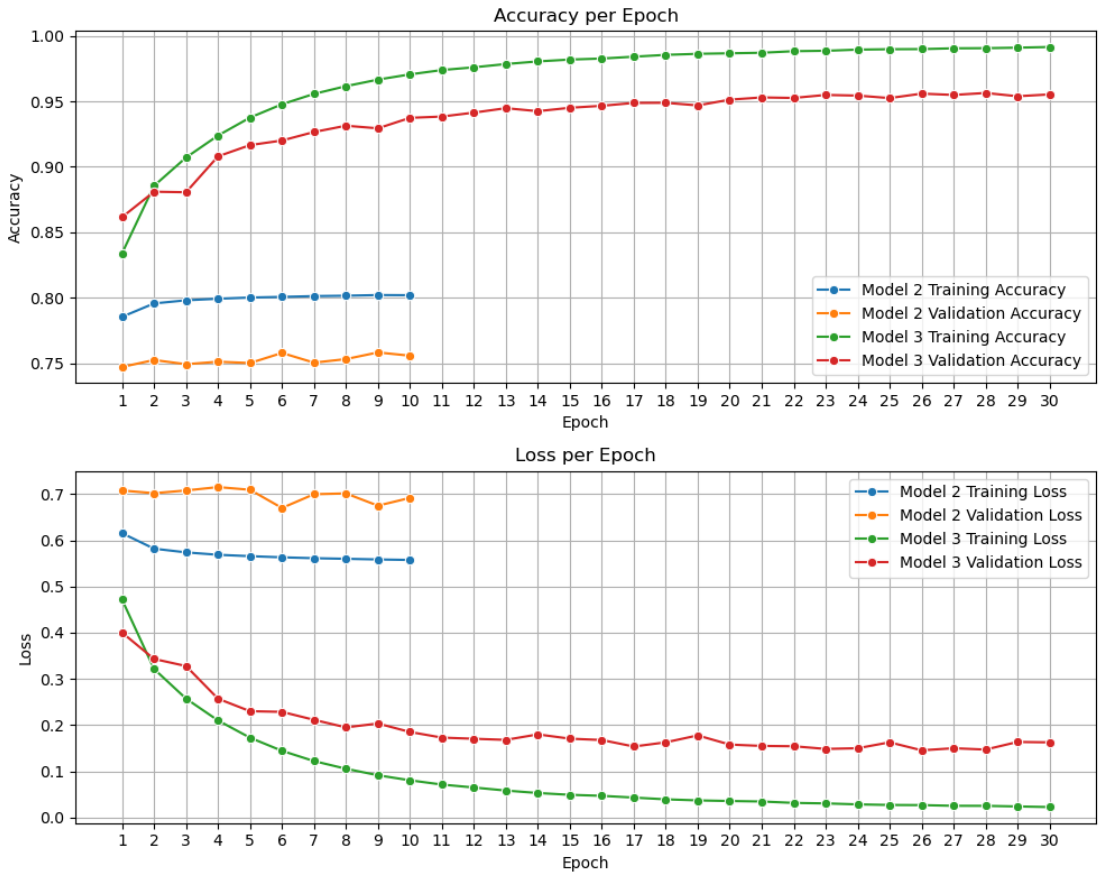


Figure 9.25. *Training results of Model 3 in the task of identifying the triad note. Batch size=16, Learning Rate=0.0001, chunk size=100*

The confusion matrix in Figure 9.26 provides a detailed overview of the model's per-

formance on the evaluation set, illustrating the distribution of predicted triad notes compared to the actual ones. Notably, we observed a substantial improvement in accuracy on the evaluation set, achieving a significant advancement over previous models.

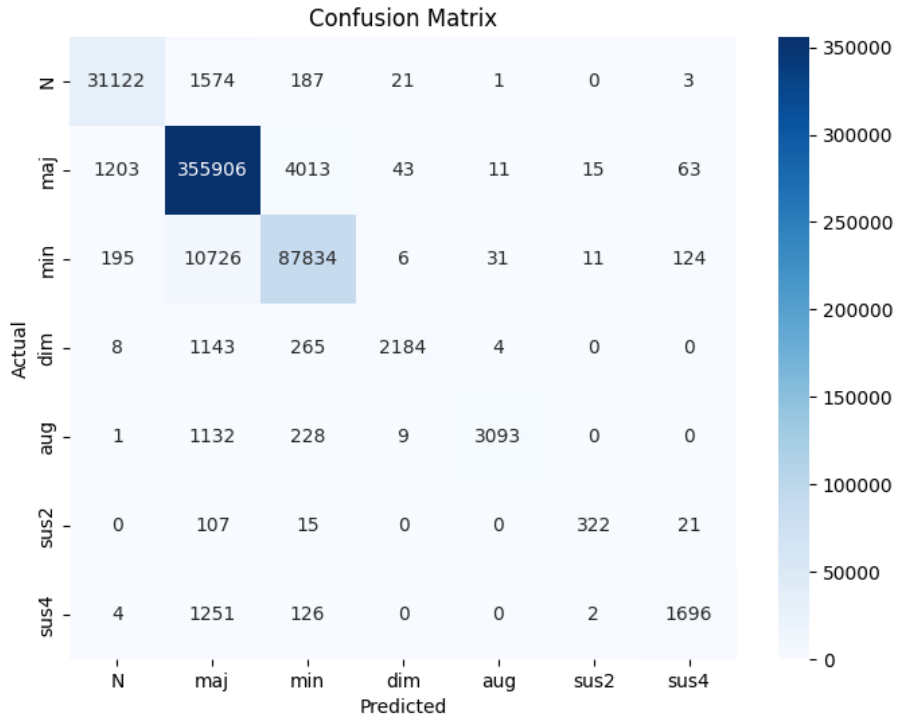


Figure 9.26. *Confusion matrix of Model 3 for identifying the triad note on the evaluation set.*

A classification report table (Table 9.17) can be seen in the Table 9.17

The comprehensive analysis presented in the table showcases the model's remarkable performance in identifying triad notes, as evidenced by high precision, recall, and F1-score metrics across the majority of triad classes. Notably, the model exhibits an impressive accuracy of 96%, signifying its capability to accurately predict the triad classes with a high level of confidence. This substantial improvement from Model 2, which achieved 76% accuracy on this task, highlights the efficacy of the enhancements implemented in Model 3. Particularly noteworthy is the model's ability to overcome the challenge of accurately predicting minority classes, demonstrating its robustness and reliability across the entire spectrum of triad notes.

	Precision	Recall	F1-score	Support
N	0.96	0.95	0.95	32908
maj	0.96	0.99	0.97	361254
min	0.95	0.89	0.92	98927
dim	0.97	0.61	0.74	3604
aug	0.99	0.69	0.81	4463
sus2	0.92	0.69	0.79	465
sus4	0.89	0.55	0.68	3079
Accuracy			0.96	504700
Macro	0.95	0.77	0.84	504700
Weighted avg	0.96	0.96	0.95	504700

Table 9.17. *Classification Report for Model 3 on the task of identifying the triad note on the evaluation set.*

The model's performance on the test set, mentioned in Section 9.3.1, is illustrated in Figure 9.27. Table 9.18 presents the classification report for Model 3, tasked with identifying triad notes on the evaluation set, providing precision, recall, F1-score, and support metrics for each triad note class, along with overall accuracy, macro-averaged metrics, and weighted-averaged metrics. Overall, the model achieves an accuracy of approximately 97%, indicating a relatively high performance on the test set, considering its size of 108,600 samples compared to the evaluation set's 504,700 samples.

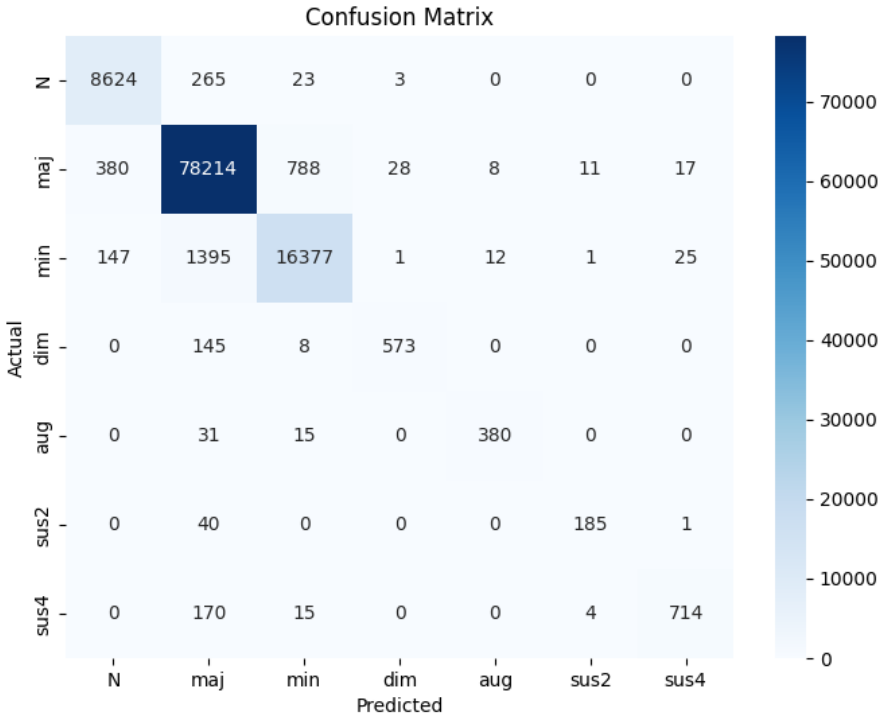


Figure 9.27. *Confusion matrix of Model 3 for identifying triad notes on the test set.*

	Precision	Recall	F1-score	Support
N	0.94	0.97	0.95	8915
maj	0.97	0.98	0.98	79446
min	0.95	0.91	0.93	17958
dim	0.95	0.79	0.86	726
aug	0.95	0.89	0.92	426
sus2	0.92	0.82	0.87	226
sus4	0.94	0.79	0.86	903
Accuracy			0.97	108600
Macro	0.95	0.88	0.91	108600
Weighted avg	0.97	0.97	0.97	108600

Table 9.18. *Classification Report for Model 3 on the task of identifying the triad note on the test set.*

9.3.5 Model 3: Extension 1 Classification Task

Having successfully tackled the previous tasks, our focus now shifts to the extension 1 classification task. In this task, we aim to classify extension 1 notes into a predefined set of categories, which include N, dim7, hdim7, maj6, maj7, and min7. As in the previous tasks, we also have the "None" as "N" category indicating the absence of an extension 1 note.

For the extension 1 classification task, we build upon the established architecture and methodology, leveraging transfer learning techniques honed in prior tasks. Through extensive experimentation, we iteratively refined our approach, exploring various configurations to identify the most effective strategy. Ultimately, we arrived at an optimal architecture that builds upon the foundation of the previous tasks while incorporating additional layers tailored to the demands of the extension 1 classification task. The detailed architecture, delineated in Tables 9.19 and 9.20, outlines the structure of our model.

The 'core' architecture remains consistent for this task as well, providing a stable foundation for model training. Similarly, the additional layers introduced in previous tasks are retained here, as they have proven to be effective for this task as well. When we say these layers are the same, we are referring to their architecture, indicating that they were once again utilized and trained specifically for the extension 1 classification task.

Layer (type)	Output Shape	Param #
conv2d	(None, 100, 192, 32)	320
conv2d_1	(None, 100, 192, 32)	9248
conv2d_2	(None, 100, 192, 32)	9248
conv2d_3	(None, 100, 192, 32)	9248
batch_normalization	(None, 100, 192, 32)	128
max_pooling2d	(None, 100, 96, 32)	0
dropout	(None, 100, 96, 32)	0
conv2d_4	(None, 100, 96, 64)	18496
conv2d_5	(None, 100, 96, 64)	36928
batch_normalization_1	(None, 100, 96, 64)	256
max_pooling2d_1	(None, 100, 48, 64)	0
Total params		**83872**
Trainable params		**0 (0.00 Byte)**
Non-trainable params		**83872 (327.62 KB)**

Table 9.19. *Detailed Architecture used as 'core' for Transfer Learning for the Extension 1 Classification Task. These layers are frozen, meaning the parameters will not be trained again.*

Layer (type)	Output Shape	Param #
conv2d	(None, 100, 192, 32)	320
conv2d_1	(None, 100, 192, 32)	9248
conv2d_2	(None, 100, 192, 32)	9248
conv2d_3	(None, 100, 192, 32)	9248
batch_normalization	(None, 100, 192, 32)	128
max_pooling2d	(None, 100, 96, 32)	0
dropout	(None, 100, 96, 32)	0
conv2d_4	(None, 100, 96, 64)	18496
conv2d_5	(None, 100, 96, 64)	36928
batch_normalization_1	(None, 100, 96, 64)	256
max_pooling2d_1	(None, 100, 48, 64)	0
ext1_conv	(None, 100, 48, 64)	36928
ext1_conv2	(None, 100, 48, 128)	73856
BatchNorm_ext1	(None, 100, 48, 128)	512
MaxPooling_ext1	(None, 100, 16, 128)	0
Dropout_ext1	(None, 100, 16, 128)	0
Flatten_Ext1	(None, 100, 2048)	0
LSTM_layer (Bidirectional)	(None, 100, 256)	2229248
out (TimeDistributed)	(None, 100, 7)	1542
Total params		**2369884**
Trainable params		**2281756 (8.70 MB)**
Non-trainable params		**88128 (343.12 KB)**

Table 9.20. *Detailed Architecture for the Extension 1 Classification Task including all layers (frozen and added).*

As with the previous tasks, by freezing the parameters of certain layers, we ensure they are excluded from further training during the adaptation process for the extension 1 classification task. This strategy aims to preserve the learned patterns from the previous

tasks, which can prove advantageous due to the inherent similarities between the classification objectives. By retaining these learned features, we streamline the training process, reduce computational overhead, and mitigate the risk of overfitting. Leveraging transfer learning in this manner allows us to capitalize on insights gleaned from prior tasks and tailor them to the specific demands of the extension 1 classification task, potentially enhancing the model's overall performance.

Training the Model on Extension 1 Task

For training the model, we utilized the Adam optimizer with a learning rate of 0.0001, β_1 set to 0.9, β_2 set to 0.99, and epsilon (ϵ) set to 1×10^{-8}. The loss function employed for training was categorical crossentropy, and we evaluated the model's performance using accuracy as the metric with a batch size of 16.

The performance of the model in classifying the triad notes is shown in Figure 9.28. The model underwent training for 30 epochs, where it demonstrated significant improvement in accuracy over time. Both the training and validation accuracy stabilized at approximately 99% and 96%, respectively, in comparison to 85% which was the previous model's accuracy on this task.

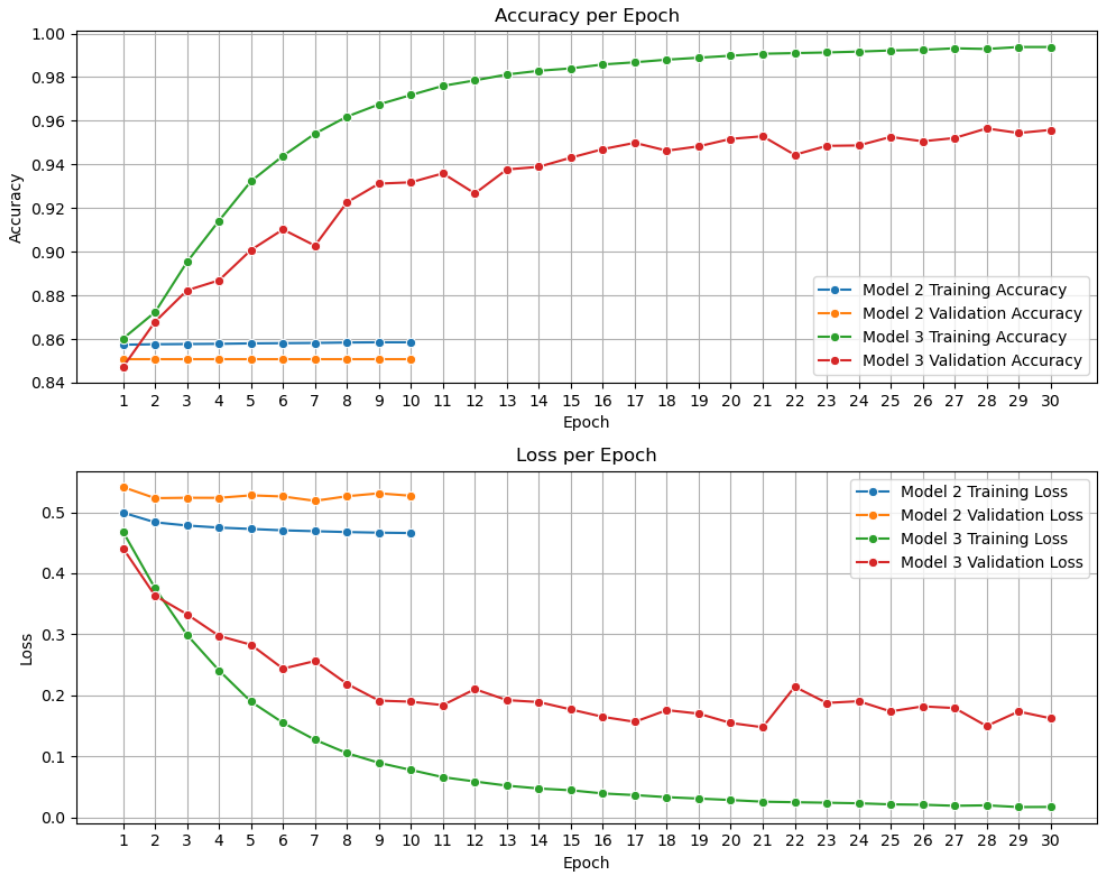


Figure 9.28. *Training results of Model 3 in the task of identifying the Extension 1. Batch size=16, Learning Rate=0.0001, chunk size=100*

The confusion matrix depicted in Figure 9.29 offers an insightful overview of how

well the model performs on the evaluation set, delineating the distribution of predicted Extension 1 notes against the actual ones. A significant advancement of this model over its predecessor (Model 2) lies in its proficiency in predicting minority classes. In contrast, when scrutinizing the confusion matrix (Figure 9.14) associated with Model 2, it becomes apparent that the majority of samples were misclassified as the "None" class. This stark observation underscored the previous model's incapacity to effectively handle this task. In contrast, the updated model showcased here exhibits the capability to accurately predict all classes, marking a notable improvement in its predictive performance.

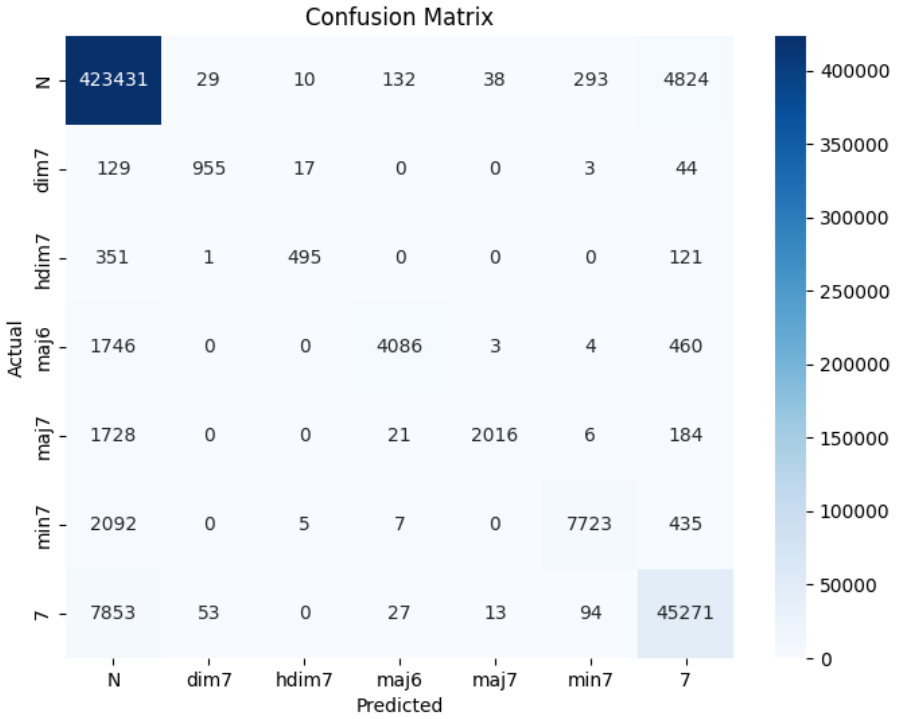


Figure 9.29. *Confusion matrix of Model 3 for identifying the Extension 1 on the evaluation set.*

A classification report table (Table 9.25) can be seen in the Table 9.25

The comprehensive analysis presented in the table showcases the model's remarkable performance in identifying triad notes, as evidenced by high precision, recall, and F1-score metrics across the majority of triad classes. Notably, the model exhibits an impressive accuracy of 96%, signifying its capability to accurately predict the triad classes with a high level of confidence. As mentioned previously, the major improvement is that this model exhibits the capability to accurately predict all classes

	Precision	Recall	F1-score	Support
N	0.97	0.99	0.98	428757
dim7	0.92	0.83	0.87	1148
hdim7	0.94	0.51	0.66	968
maj6	0.96	0.65	0.77	6299
maj7	0.97	0.51	0.67	3955
min7	0.95	0.75	0.84	10262
7	0.88	0.85	0.87	53311
Accuracy			0.96	504700
Macro	0.94	0.73	0.81	504700
Weighted avg	0.96	0.96	0.96	504700

Table 9.21. *Classification Report for Model 3 on the task of identifying the Extension 1 on the evaluation set.*

The model's performance on the test set, mentioned in Section 9.3.1, is illustrated in Figure 9.30. Table 9.22 presents the classification report for Model 3, tasked with identifying triad notes on the evaluation set, providing precision, recall, F1-score, and support metrics for each triad note class, along with overall accuracy, macro-averaged metrics, and weighted-averaged metrics. Overall, the model achieves an accuracy of approximately 98%, indicating a relatively high performance on the test set, considering its size of 108,600 samples compared to the evaluation set's 504,700 samples.

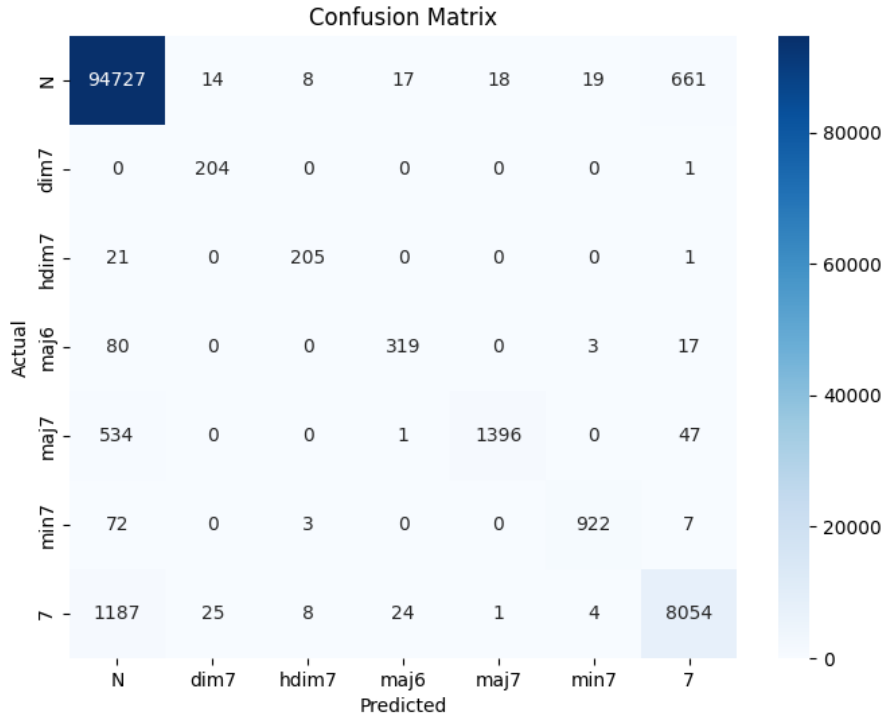


Figure 9.30. *Confusion matrix of Model 3 for identifying Extension 1 on the test set.*

	Precision	Recall	F1-score	Support
N	0.99	0.99	0.99	95464
dim7	0.84	1.00	0.91	205
hdim7	0.92	0.90	0.91	227
maj6	0.88	0.76	0.82	419
maj7	0.99	0.71	0.82	1978
min7	0.97	0.92	0.94	1004
7	0.92	0.87	0.89	9303
Accuracy			0.97	108600
Macro	0.93	0.88	0.90	108600
Weighted avg	0.97	0.97	0.97	108600

Table 9.22. *Classification Report for Model 3 on the task of identifying the Extension 1 on the test set.*

9.3.6 Model 3: Extension 2 Classification Task

The final task is the classification of the Extension 2. In this task, we aim to classify extension 2 notes into a predefined set of categories, which include N, and 9 so it is a binary classification problem. As in the previous tasks, we also have the "None" as "N" category indicating the absence of an extension 2 note.

For the Extension 2, the architecture used in Extension 1 was fined to work best. The detailed architecture, delineated in Tables 9.23 and 9.24, outlines the structure of our model.

Layer (type)	**Output Shape**	**Param #**
conv2d	(None, 100, 192, 32)	320
conv2d_1	(None, 100, 192, 32)	9248
conv2d_2	(None, 100, 192, 32)	9248
conv2d_3	(None, 100, 192, 32)	9248
batch_normalization	(None, 100, 192, 32)	128
max_pooling2d	(None, 100, 96, 32)	0
dropout	(None, 100, 96, 32)	0
conv2d_4	(None, 100, 96, 64)	18496
conv2d_5	(None, 100, 96, 64)	36928
batch_normalization_1	(None, 100, 96, 64)	256
max_pooling2d_1	(None, 100, 48, 64)	0
Total params		**83872**
Trainable params		**0 (0.00 Byte)**
Non-trainable params		**83872 (327.62 KB)**

Table 9.23. *Detailed Architecture used as 'core' for Transfer Learning for the Extension 2 Classification Task. These layers are frozen, meaning the parameters will not be trained again.*

Layer (type)	Output Shape	Param #
conv2d	(None, 100, 192, 32)	320
conv2d_1	(None, 100, 192, 32)	9248
conv2d_2	(None, 100, 192, 32)	9248
conv2d_3	(None, 100, 192, 32)	9248
batch_normalization	(None, 100, 192, 32)	128
max_pooling2d	(None, 100, 96, 32)	0
dropout	(None, 100, 96, 32)	0
conv2d_4	(None, 100, 96, 64)	18496
conv2d_5	(None, 100, 96, 64)	36928
batch_normalization_1	(None, 100, 96, 64)	256
max_pooling2d_1	(None, 100, 48, 64)	0
ext2_conv	(None, 100, 48, 64)	36928
ext2_conv2	(None, 100, 48, 128)	73856
BatchNorm_ext2	(None, 100, 48, 128)	512
MaxPooling_ext2	(None, 100, 16, 128)	0
Dropout_ext2	(None, 100, 16, 128)	0
Flatten_Ext2	(None, 100, 2048)	0
LSTM_layer (Bidirectional)	(None, 100, 256)	2229248
out (TimeDistributed)	(None, 100, 2)	1542
Total params		**2424930 (9.25 MB)**
Trainable params		**2340802 (8.93 MB)**
Non-trainable params		**84128 (343.12 KB)**

Table 9.24. *Detailed Architecture for the Extension 2 Classification Task including all layers (frozen and added).*

Training the Model on Extension 2 Task

For training the model, we utilized the Adam optimizer with a learning rate of 0.0001, β_1 set to 0.9, β_2 set to 0.99, and epsilon (ϵ) set to 1×10^{-8}. The loss function employed for training was categorical crossentropy, and we evaluated the model's performance using accuracy as the metric with a batch size of 16.

The performance of the model in classifying the triad notes is shown in Figure 9.31. The model underwent training for 30 epochs, where it demonstrated significant improvement in accuracy over time. Both the training and validation accuracy stabilized at approximately 99% and 99%, respectively.

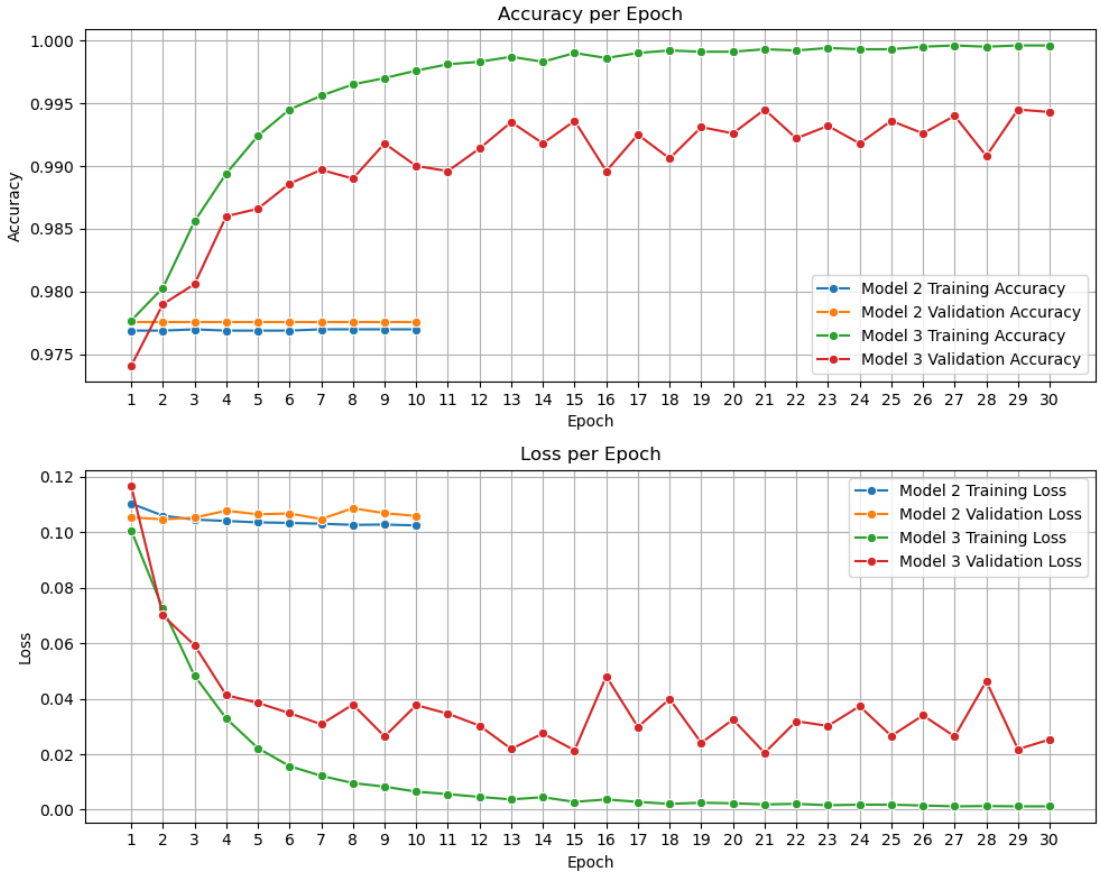


Figure 9.31. *Training results of Model 3 in the task of identifying the Extension 2. Batch size=16, Learning Rate=0.0001, chunk size=100*

The confusion matrix depicted in Figure 9.32 offers an insightful overview of how well the model performs on the evaluation set, delineating the distribution of predicted Extension 1 notes against the actual ones. A significant advancement of this model over its predecessor (Model 2) lies in its proficiency in predicting minority class. In contrast, when scrutinizing the confusion matrix (Figure 9.16) associated with Model 2, it becomes apparent that the majority of samples were misclassified as the "None" class. The previous model did not have the ability to predict the "9" class. In contrast, the new model showcases major improvements in this aspect of the problem.

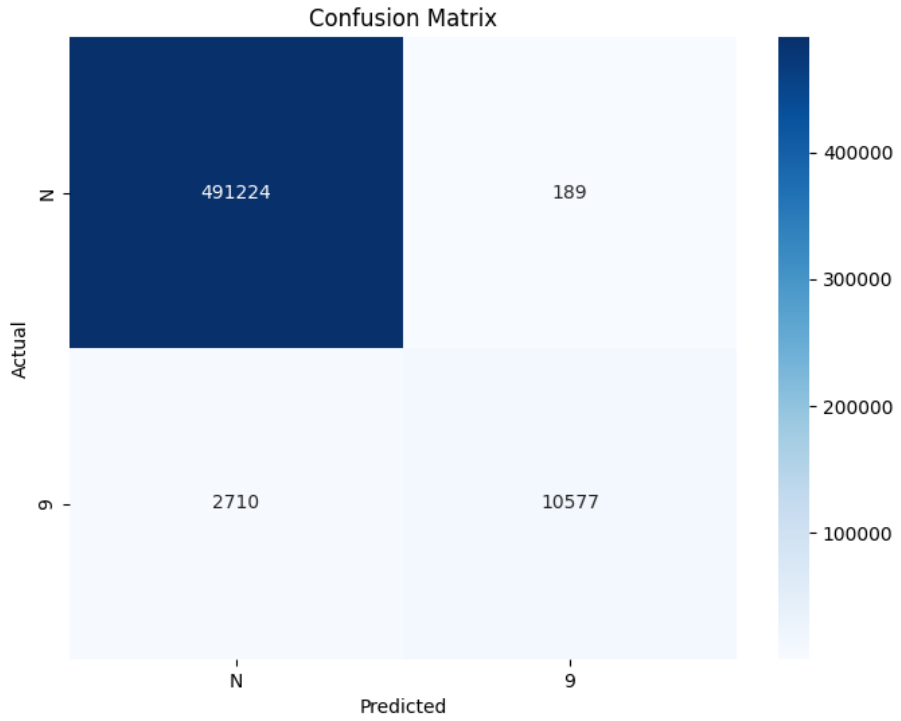


Figure 9.32. *Confusion matrix of Model 3 for identifying the Extension 2 on the evaluation set.*

A classification report table (Table **??**) can be seen in the Table **??**

The comprehensive analysis presented in the table showcases the model's remarkable performance in identifying triad notes, as evidenced by high precision, recall, and F1-score metrics across the majority of triad classes. Notably, the model exhibits an impressive accuracy of 99%, signifying its capability to accurately predict the triad classes with a high level of confidence. As mentioned previously, the major improvement is that this model exhibits the capability to accurately predict all classes. The most important metric here is the macro avg which shows a significant 94% score.

	Precision	Recall	F1-score	Support
N	0.99	1.00	1.00	491413
9	0.98	0.80	0.88	13287
Accuracy			0.99	504700
Macro	0.99	0.90	0.94	504700
Weighted avg	0.99	0.99	0.99	504700

Table 9.25. *Classification Report for Model 3 on the task of identifying the Extension 1 on the evaluation set.*

The model's performance on the test set, mentioned in Section 9.3.1, is illustrated in Figure 9.30. Table 9.22 presents the classification report for Model 3, tasked with identifying triad notes on the evaluation set, providing precision, recall, F1-score, and support metrics for each triad note class, along with overall accuracy, macro-averaged

metrics, and weighted-averaged metrics. Overall, the model achieves an accuracy of approximately 100% and a 95% macro average score on the test set.

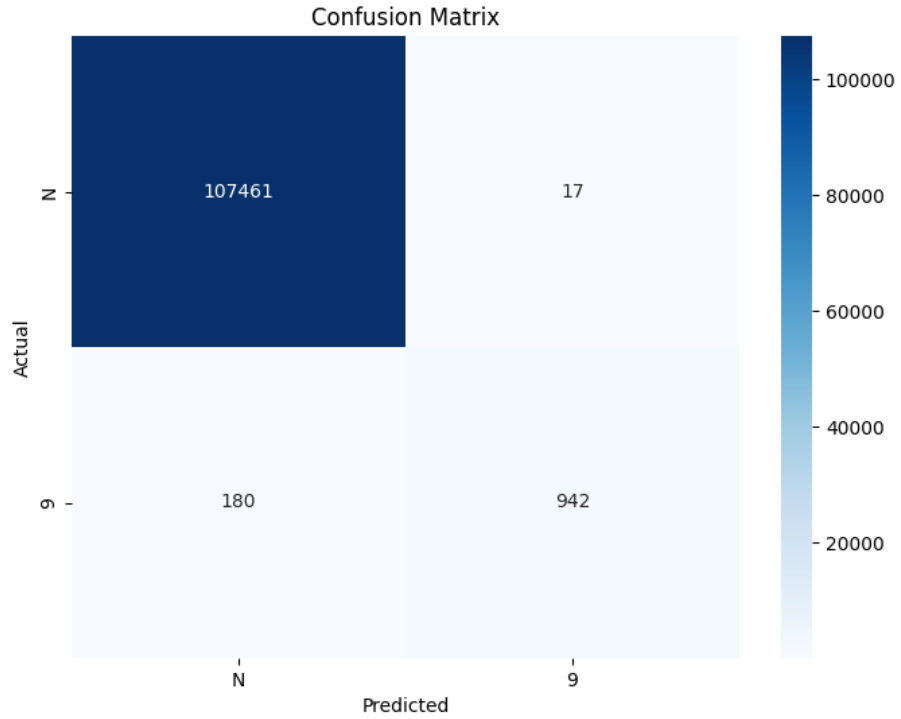


Figure 9.33. *Confusion matrix of Model 3 for identifying Extension 2 on the test set.*

	Precision	Recall	F1-score	Support
N	1.00	1.00	1.00	107478
9	0.98	0.84	0.91	1122
Accuracy			1.00	108600
Macro	0.99	0.92	0.95	108600
Weighted avg	1.00	1.00	1.00	108600

Table 9.26. *Classification Report for Model 3 on the task of identifying the Extension 2 on the test set.*

9.4 Summary of the Results

In this chapter we explored three different architecture approaches for the problem of chord recognition. The initial model employed a simple 1D Convolutional Neural Network (CNN) as a baseline for evaluation, focusing on individual chord embeddings without considering chord structural representations. However, this approach may have limited predictive capabilities due to its disregard for music theory and chord relations.

The subsequent model utilized a more complex 1D CNN architecture to capture more complex patterns within the chord data. Through experimentation, it was determined that employing the Constant Q Transform (CQT) method with 192 features outperformed the

Constant-Q Chromagram approach. This model incorporated convolutional layers, max-pooling layers, and dropout layers to prevent overfitting. Additionally, transfer learning techniques were applied to efficiently address tasks such as identifying the root note, bass note, triad, and extensions, enhancing the model's performance across various chord recognition tasks.

In the final model, a 2D Convolutional Neural Network with Bidirectional Long Short-Term Memory (BiLSTM) layers was utilized. This model treated spectrograms as images, facilitating the extraction of meaningful features across smaller time intervals using convolutional layers. 'Chunking', as detailed in 8, was employed as a preprocessing step to partition the input data into manageable segments, improving the LSTM's ability to capture temporal dependencies. Similar to the previous model, the Constant Q Transform method was favored over the Constant-Q Chromagram approach, and transfer learning techniques were applied to enhance performance across different chord recognition tasks.

All the aforementioned models were trained, tested, and evaluated using a GPU, specifically an NVIDIA GeForce GTX 1650 SUPER with 4 GB of memory. Due to the limited memory, custom batch generators were implemented instead of using the pre-coded implementations available in the TensorFlow package. It is interesting to note the training times for each model. The training times, measured in seconds, are shown in Figure 9.34 below. Model 2's training time for the Root and Bass classification tasks is comparable to that of Model 3. However, for the remaining three tasks, Model 3 requires significantly more time due to the increased complexity of its additional layers.

Figure 9.34. *Training time in seconds for each model and task using one NVIDIA GeForce GTX 1650 SUPER.*

As discussed in the previous sections, each model showed a significant increase in accuracy compared to its predecessor. The accuracy for each model and task is illustrated in Figure 9.35.

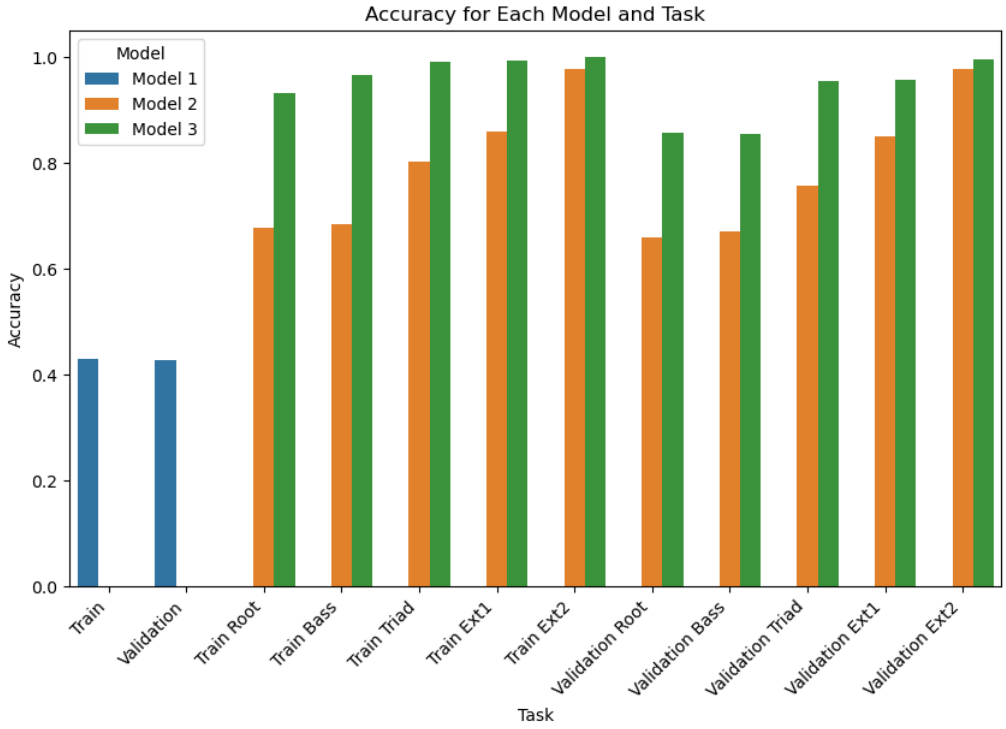


Figure 9.35. *Accuracy scores for each model and each task (training & validation).*

Another interesting comparison is between Model 2 and Model 3 in their ability to classify the Extension 2 task. While this has been analyzed previously, a direct comparison makes the superior performance of Model 3 clear. This is shown in Figure 9.36, which compares the confusion matrices of the two models.

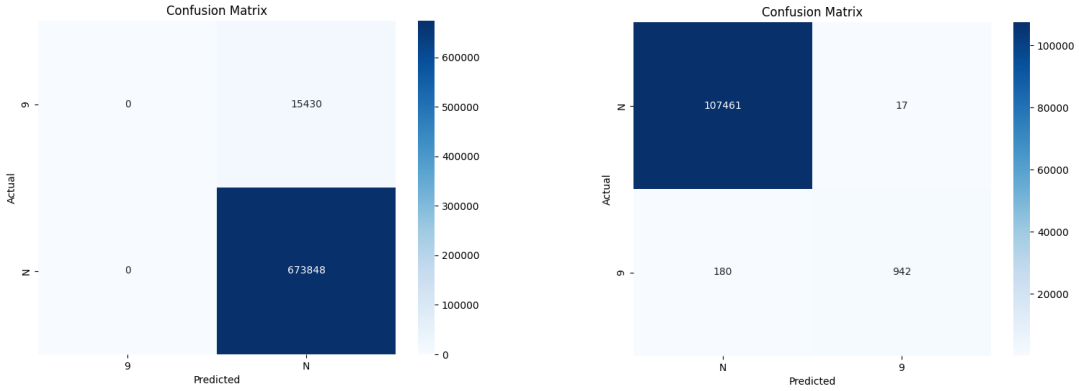


Figure 9.36. *Comparison between Model 2 and Model 3 on the task of classifying Extension 2.*

It is also important to analyze other metrics, such as the F1 score, as they provide additional insights into model performance. The bar plot in Figure 9.37 illustrates these scores for Model 3, which outperformed the other models.

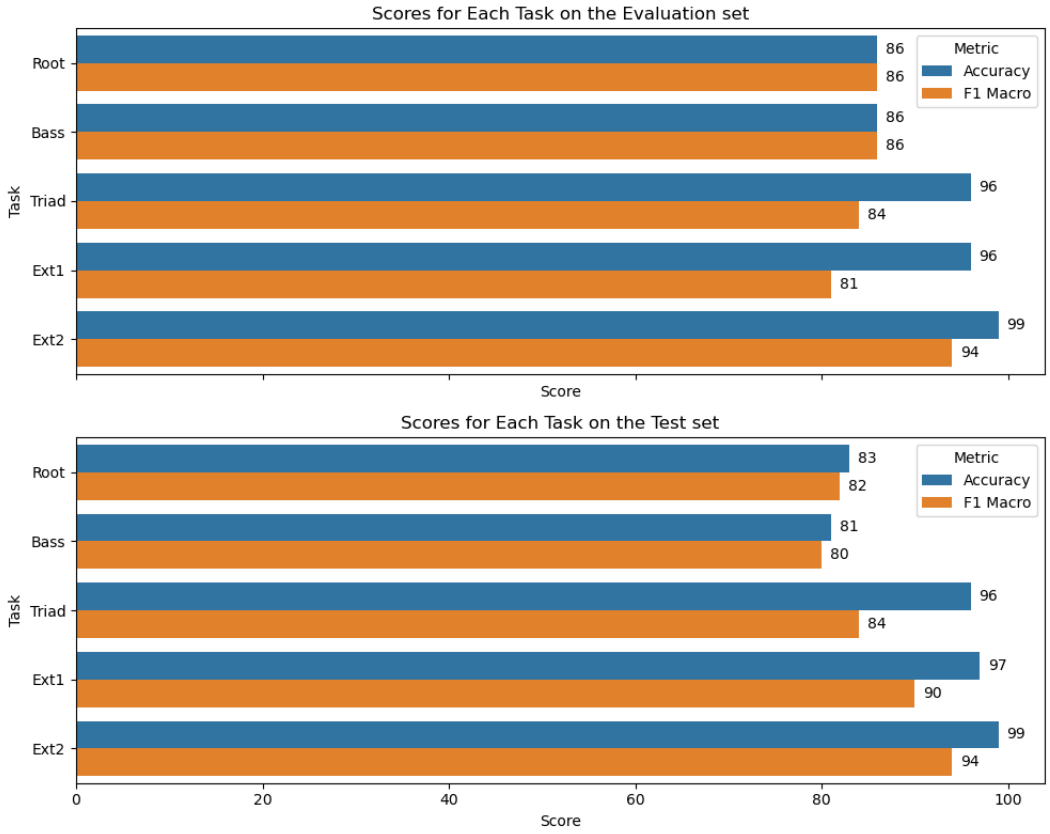


Figure 9.37. *Bar plot showing accuracy and macro-averaged F1-score for all five tasks for Model 3.*

As we continue, we'll explore post-processing techniques designed to enhance model accuracy. These methods play a crucial role in refining model performance, ensuring optimal outcomes.

Chapter 10

Post Processing

In this chapter, we elaborate on several post-processing techniques aimed at refining the chord predictions obtained from our models. These techniques, are detailed here for comprehensive understanding.

Firstly, we undertake the task of assembling data, each pertaining to a specific chord component such as root, bass, triad, and extensions. These individual components are consolidated into a prediction, the chord.

Subsequently, we delve into the post-processing techniques designed specifically for chord refinement. One such function involves filtering each chord based on predetermined rules that we will see in detail.

Additionally, we introduce a smoothing mechanism designed to enhance the coherence and consistency of the chord predictions. This function replaces values within a specified window with the most common value in the vicinity, thereby mitigating erratic fluctuations and ensuring smoother transitions between chords.

The comprehensive pipeline, encompassing all the steps mentioned in previous chapters, can be visually depicted through the accompanying flowchart (10.1).

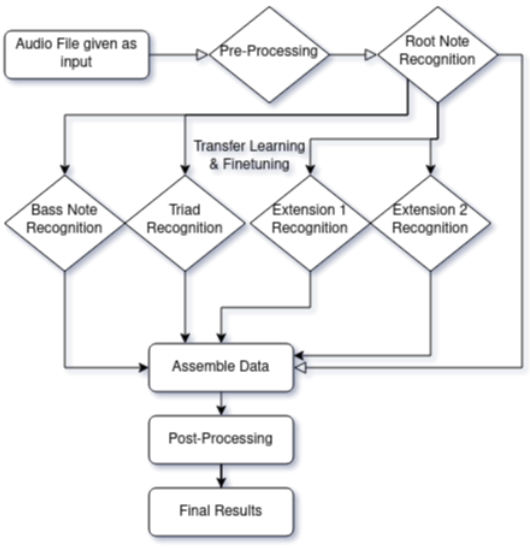


Figure 10.1. *Comprehensive Pipeline of all the steps of the Chord Recognition task including Post Processing.*

10.1 Chord Assembly

In this section, we dive into a crucial phase of our analysis, where we bring together the different elements of musical chords into a single dataset. This step is vital as it lays the groundwork for the subsequent stages of our processing pipeline. Here, we meticulously integrate various components such as root notes, bass lines, triads, and extensions extracted from different sources, including audio recordings and digital representations.

At the heart of this process lies the careful organization and integration of data from diverse sources. Through meticulous handling and processing, our aim is to create a comprehensive dataset that captures the intricate details of musical chords found within the input audio files. This section explores the methodologies and techniques we employ to harmonize these different chord components, setting the stage for a detailed analysis and interpretation of musical harmonies.

The methodology involves combining predictions made for each specific task. Initially, the algorithm identifies the files representing various chord components like root, bass, triad, and extensions. It then systematically retrieves data from each file, ensuring that all pertinent columns are included. This consolidation process harmonizes the disparate data streams, facilitating a holistic view of the harmonic structures within the musical compositions, the chords. The methodology can be visualized as in the following Figure 10.2.

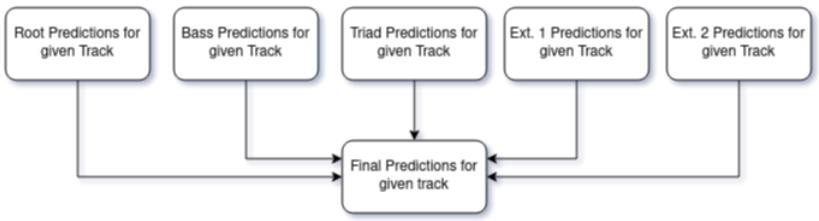


Figure 10.2. *Algorithm that assembles the predictions of each task into the final prediction, the chord.*

10.2 Smoothing

The post-processing technique we here define as 'smoothing' aims to enhance the coherence and consistency of chord predictions obtained from the analysis. By employing a sliding window approach, this method iteratively replaces individual chord values with the most prevalent value within a specified window size. This process effectively reduces abrupt fluctuations in the chord progression, resulting in a more harmonious and continuous musical interpretation.

The smoothing algorithm operates on a the chord data we got as predictions. It iterates over each chord value, considering a window of neighboring values to determine the most common chord within that window. If the predominant chord occurs with sufficient frequency within the window, it replaces the original chord value. Otherwise, the original chord value remains unchanged, ensuring preservation of significant chord transitions.

Utilizing this smoothing technique enhances the overall quality of chord predictions, thereby refining the accuracy and interpretability of musical analyses. Smoothing plays an important role in our analysis due to the nature of the data we work with. In this project, as mentioned in the previous chapters, each frame spans approximately around 0.07 seconds, making it improbable for rapid chord changes to occur and then revert to the same chord. Such rapid transitions are likely attributed to noise or irregularities in the data. By implementing smoothing techniques, we effectively eliminate these outliers, resulting in more precise and reliable chord predictions. This process enhances the accuracy of our analysis by ensuring that the predicted chord progressions align more closely with the inherent structure and flow of the music, ultimately providing a clearer and more coherent interpretation of the musical composition.

We experimented with applying smoothing techniques to both assembled chords and individual chord components. Interestingly, smoothing each component of the chord independently resulted in superior outcomes compared to applying smoothing to the assembled chords as a whole. This approach to smoothing yielded more effective results, highlighting the importance of considering the distinct characteristics and dynamics of each chord component in the post-processing stage. An example can be seen in the following Figure 10.3.

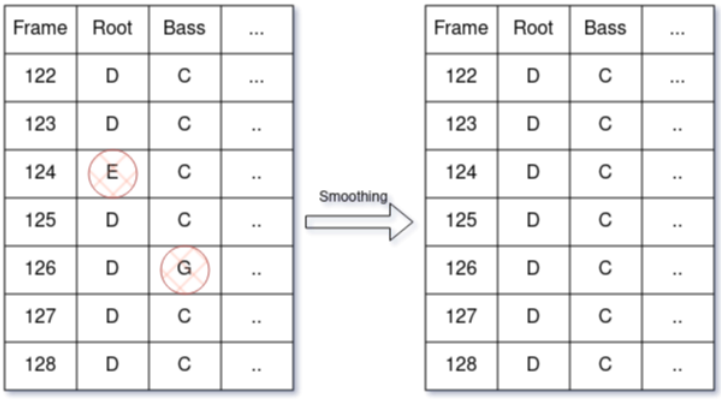


Figure 10.3. *Example of Smoothing Algorithm usage.*

The window size was fine-tuned and established at 5 frames, equivalent to approximately 0.35 seconds of musical data. It is interesting to evaluate this methodology based on metrics such as the accuracy on the test set, comprising the albums *CD1*, *CD2*, *Help*, and *Please Please Me* as detailed in 9.3.1. The Figure 10.4 below offers a visual representation of the findings. Notably, a slight increase in accuracy is observed, indicating the efficiency of the adjusted window size.

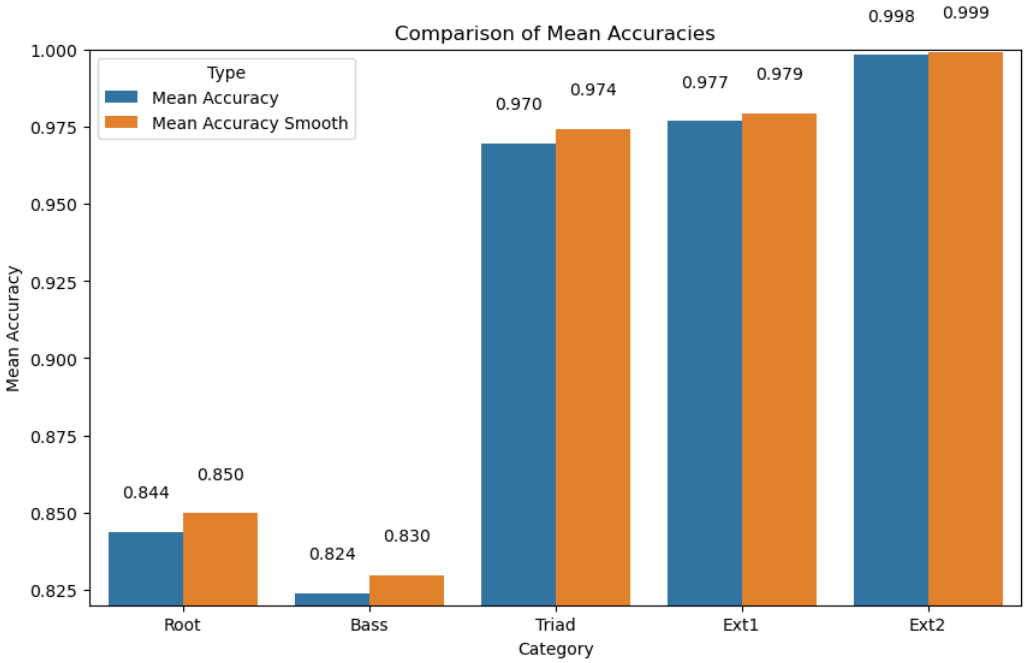


Figure 10.4. *Accuracy comparison between data with smoothing and without smoothing applied on the Test set.*

10.3 Filtering Algorithm

In this section, we will discuss the post-processing steps applied to the chord data after smoothing. The primary focus will be on the methods used to identify and filter chords based on certain rules.

The chord filtering process is implemented and applies a set of rules to each row (or frame) of the data containing the chords. The rules are as follows:

1. If the root is None, then the whole chord is set to None.
2. If the bass is None, then the bass is set to the root.
3. If the triad is None, then the triad is set to the closest value.

The Algorithm iterates over each row of the data. For each row, it checks the root, bass, and triad values and applies the rules accordingly. The rules are implemented in a sequential and rigid order. In the third step, if the triad is None, then a process to get the nearest triad is initiated.

The process of identifying the closest triad to a given index is a crucial step in our post-processing pipeline. This process is implemented and takes as input the data containing the chords, an index (of the frame), and a direction to search for the closest triad. The direction can be either "left" or "right" meaning before and after the given frame. If the index is within bounds and the triad at the given index is not None, it returns the triad and the index. If the triad at the given index is None, it recursively calls itself to find the closest non-zero triad in the specified direction. The whole process can be visualized as in the following Figure 10.5

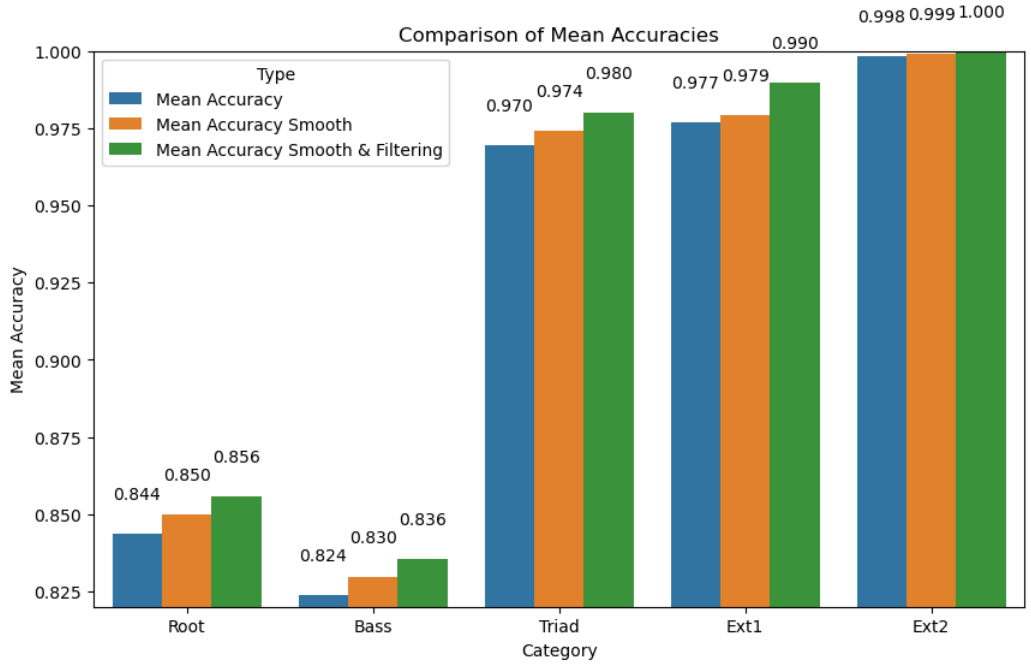


Figure 10.5. *Flowchart of filtering Algorithm on the Test set.*

We can observe an increase of over 1% in accuracy on most tasks when comparing the raw predictions to the filtered results. This improvement demonstrates the effectiveness of our filtering algorithm in enhancing the chord data. By systematically applying the rules and accurately identifying the nearest triad, the algorithm improves the overall quality of the chord predictions. This leads to more precise and reliable chord identifications, which are essential for applications such as music analysis and automated transcription. The combination of smoothing and filtering thus proves to be a robust approach, significantly boosting the performance and accuracy of our chord recognition system.

Chapter 11

MIREX Results and Comparisons

In this chapter, we evaluate our chord recognition system using the MIREX (Music Information Retrieval Evaluation eXchange) metrics. MIREX provides a standardized framework for assessing music information retrieval systems, making it an ideal benchmark for comparing our results with those from other studies.

We'll start by explaining the MIREX evaluation metrics briefly (detailed explanation is on 7) and their importance in chord recognition. Using these standardized metrics ensures our evaluation is both thorough and comparable to other leading systems.

Finally, we'll compare our results with those reported in other papers. This comparison will put our performance in context, showing how our system measures up against existing solutions. By examining these comparisons, we aim to highlight the effectiveness of our approach and its contribution to the field of music information retrieval.

This thorough evaluation and comparison will help validate the effectiveness of our chord recognition system and provide insights into its performance relative to other research in the field.

11.1 Metrics

Following Pauwels and Peeters (2013) [38], we will be using the CSR with five different chord vocabulary mappings. Detailed analysis of CSR can be found in 7.

In each of these calculations, the full chord descriptions from either the estimated or the ground-truth transcriptions, which might include complex chord annotations, are mapped to the following classes:

- Chord root note only;
- Major and minor: {N, maj, min};
- Seventh chords: {N, maj, min, maj7, min7, 7};
- Major and minor with inversions: {N, maj, min, maj/3, min/b3, maj/5, min/5}; or
- Seventh chords with inversions: {N, maj, min, maj7, min7, 7, maj/3, min/b3, maj7/3, min7/b3, 7/3, maj/5, min/5, maj7/5, min7/5, 7/5, maj7/7, min7/b7, 7/b7}.

Except for no-chords, calculating the vocabulary mapping involves examining the root note, the bass note, and the relative interval structure of the chord labels. A mapping exists if both the root notes and bass notes match, and the structure of the output label is the largest possible subset of the input label given the vocabulary. For instance, in the major and minor case, G:7(#9) is mapped to G:maj because the interval set of G:maj, {1, 3, 5}, is a subset of the interval set of G:7(#9), {1, 3, 5, b7, #9}. In the seventh-chord case, G:7(#9) is mapped to G:7 instead because the interval set of G:7 {1, 3, 5, b7} is also a subset of G:7(#9) but is larger than G:maj. If a chord cannot be represented by a certain class, e.g., mapping a D:aug or F:sus4(9) to {maj, min}, the chord is excluded from the evaluation if it occurs in the ground-truth, and it is considered a mismatch if it occurs in an estimated annotation. MIREX Accuracy requires that at least three notes of the chord are correct.

11.2 Components Accuracy

Firstly, it will be interesting to examine how our chord split into components translates into recognizing the parts of those components. The following table displays the accuracy for each number of parts. For instance, "number of parts" means that for 3 parts, the root, bass, and triad must all be correct; for 4 parts, the root, bass, triad, and extension 1 must all be correct etc. These scores are provided in the Table 11.1 below

Number of Parts	1	2	3	4	5
Accuracy (%)	85.6	81.2	79.4	76.9	76.8

Table 11.1. *Accuracy for each number of parts*

11.3 Mirex evaluation and comparison

Moving on to the MIREX evaluation metrics, it will be interesting to compare our solution with other papers. Gasser and Strasser [13], in their submission, followed a similar approach of splitting the chord into components and then using a CNN architecture utilizing Transfer Learning for each component. The key difference is the use of Bi LSTM layers in our case and the post-processing techniques.

Park, Choi et al. [14] utilize a self-attention mechanism for chord recognition to focus on certain regions of chords. Training of the proposed Bi-directional Transformer for chord recognition (BTC) consists of a single phase while showing competitive performance. Through an attention map analysis, they have visualized how attention was performed. It turns out that the model was able to divide segments of chords by utilizing the adaptive receptive field of the attention mechanism. Furthermore, it was observed that the model was able to effectively capture long-term dependencies, making use of essential information regardless of distance.

Finally, Jiang, Ke Chen et al. [15] propose a new model for practical chord transcription tasks. The core concept of the new model is to represent any chord label by a set of subparts (i.e., root, triad, bass) according to their common musical structures. A

multitask classifier is then trained to recognize all the subparts given the audio feature, and then labels of individual subparts are reassembled to form the final chord label. A Recurrent Convolutional Neural Network (RCNN) is used to build the multitask classifier.

In the table below, we can observe the outcomes of each approach, including our own. Some results are missing, likely due to variations in analysis. It's worth noting that our method, described in Section 9.3.1, reserved specific albums, namely *CD1*, *CD2*, *Help*, and *Please Please Me*, exclusively for final testing. These albums were chosen due to their comprehensive representation of chords from The Beatles' entire discography.

In contrast, the *KBK2* model underwent testing on a broad range of albums, including those from The Beatles, Queen, Zweieck 2, Robbie Williams 3, RWC Popular 4, and the public portion of the McGill Billboard dataset, utilizing 8-fold cross-validation. The *JLCX1* model, however, was evaluated on a subset of 1217 songs from Isophonics, Billboard, RWC Pop, and MARL collections, although the specific composition of this subset remains unclear. Lastly, the *BTC-CRF* model was subjected to testing on various songs by The Beatles, Carole King, Queen, Zweieck, Robbie Williams, and a subset of songs from UsPop2002, employing 5-fold cross-validation.

Table 11.2. *MIREX Metrics comparison. Model2+S+F is our proposed Model 2 including smoothing and filtering as the post-processing steps. KBK2, BTC+CRF and JLCX1 are the proposed models on the corresponding papers cited.*

Metrics	Models			
	Model2+S+F	KBK2 [13]	BTC+CRF [14]	JLCX1 [15]
Root	85.6	**86.30**	83.9	83.8
MajMin	85.4	**86.02**	83.1	83.1
Sevenths	**79.9**	61.12	70.7	70.1
MajMinInv	80.1	**83.12**	-	80.1
SeventhsInv	**77.7**	58.75	-	70.0
MIREX	**81.6**	-	81.4	-

Examining the metrics in the table, we can see that our *Model2+S+F* performs exceptionally well overall, with its recognition accuracy of seventh chords being especially impressive. This important accomplishment highlights the resilience of our model, which skillfully captures the fine details included in these intricate chord structures. Not only does our method do exceptionally well in this particular measure, but it also continues to perform competitively in other chord categories. This claim is supported by comparison analyses with other models, such as KBK2, BTC+CRF, and JLCX1, which are all excellent models but not very good in seventh chord detection. These results highlight the effectiveness and versatility of our model. A visual representation in the form of a bar plot can be seen in the Figure 11.1 below.

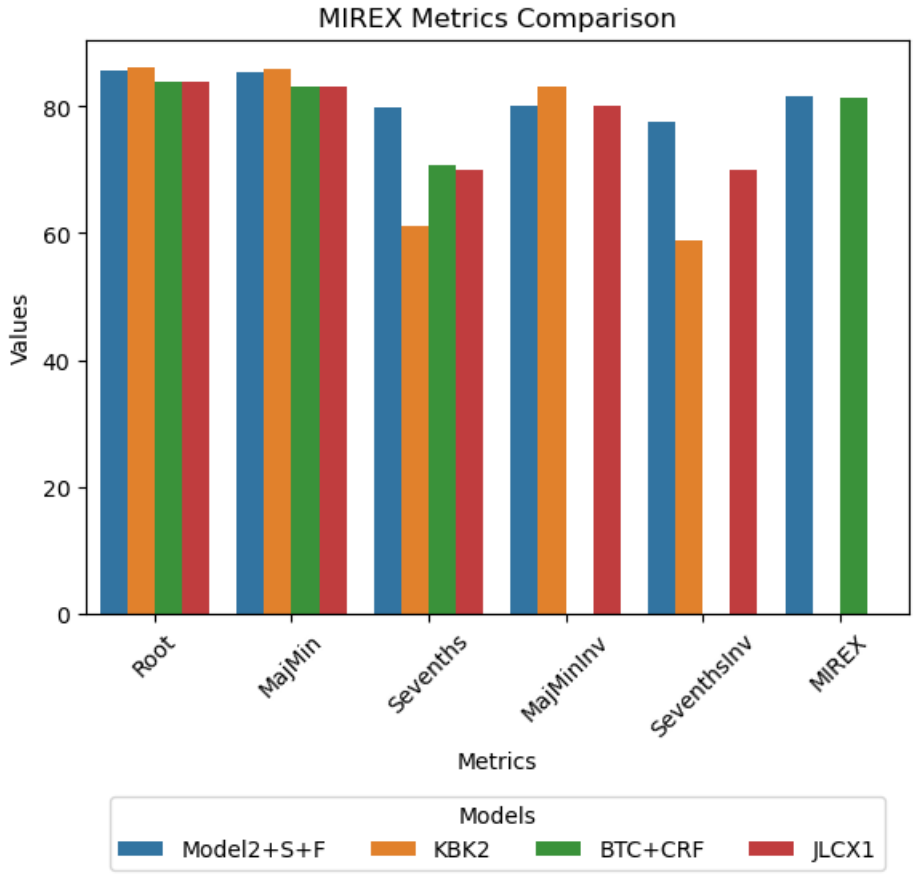


Figure 11.1. *MIREX Metrics comparison. Model2+S+F is our proposed Model 2 including smoothing and filtering as the post-processing steps. KBK2, BTC+CRF and JLCX1 are the proposed models on the corresponding papers cited.*

11.4 Conclusion

Throughout this thesis, we went on a study of alternative models and architectures to solve the complicated job of chord recognition. We started with a simple strategy of individually embedding each chord and applying a convolutional neural network (CNN) for modeling. However, as anticipated, this approach generated unsatisfactory results, underscoring the need for a more advanced strategy. Subsequently, we established an approach where the chord was divided into its constituent components, with a specific model originally crafted for the Root component. Leveraging transfer learning, we then fine-tuned this model for each additional component, resulting in better performance.

Building upon this foundation, we introduced a preprocessing step we called "chunking," which involved segmenting the data to generate "images" representing periods of frames. By incorporating 2D CNN layers to extract features and integrating a bidirectional long short-term memory (BiLSTM) layer before the dense fully connected layer, we witnessed a significant enhancement in results. This approach emerged as our proposed model, delivering competitive accuracy in chord recognition. Subsequent to model training, we employed post-processing techniques to further enhance the results.

In this chapter, we evaluated our chord recognition system using the MIREX (Music

Information Retrieval Evaluation eXchange) metrics, providing a standardized framework for assessing music information retrieval systems and enabling a thorough comparison of our results with those from other studies. We described the MIREX evaluation metrics briefly (with detailed explanations in 7) and emphasized their importance in chord recognition. Through these standardized metrics, we ensured our evaluation was comprehensive and comparable to other leading systems. Finally, we compared our results with those reported in other papers, placing our performance in context and showcasing how our system measures up against existing solutions. This thorough evaluation and comparison validated the effectiveness of our chord recognition system, providing valuable insights into its performance relative to other research in the field.

Upon analyzing the performance of our model in contrast to others, it becomes obvious that our technique demonstrates substantial strengths in chord recognition. While appreciating the virtues of rival models, our approach distinguishes itself via its constant and complete proficiency. Unlike some algorithms which excel in specific chord components, our model exhibits adaptability across numerous areas of chord identification, particularly in the correct detection of seventh chords. This broad competency shows the usefulness of our methodology, which incorporates unique preprocessing techniques, complex neural network topologies, and enhanced post-processing methodologies. Moreover, our model demonstrates resilience in adjusting to varied musical styles and circumstances, indicative of its potential for practical applications. In summary, our chord identification method represents a competitive methodology in the industry, giving a compelling solution that achieves high performance.

11.5 Future Work

This thesis has created a solid platform for ongoing innovation and exploration in the field of chord recognition. Several options suggest themselves for future research, each promising to expand the capabilities and accuracy of chord identification algorithms.

11.5.1 Expansion of Data Sources

The limitation imposed by copyright restrictions on audio tracks prevented the procurement of a more large dataset for training and testing. By adding data from a broader spectrum of musical styles, the resulting model could offer greater adaptability and precision in chord detection, particularly in the context of modern Western pop music. This larger dataset would also permit a more extensive comparison between similar models, potentially producing more conclusive conclusions.

11.5.2 Incorporation of Music Theory Principles

Integrating principles of music theory into chord recognition algorithms could lead to more musically informed predictions. By leveraging knowledge of chord progressions, harmonic functions, and melodic tendencies, the model could produce more contextually relevant and aesthetically pleasing results.

11.5.3 Integration of Beat Tracking

Beat tracking, albeit a separate MIREX classification job, holds tremendous potential as a supplemental input for chord identification models. By including beat tracking data as a secondary input, the algorithm could better discern the timing of chord changes, leading to more exact and contextually relevant predictions. This integration would lead to the development of more dynamic and rhythmically sensitive chord identification systems.

11.5.4 Key Recognition as Pre-Processing

The incorporation of key recognition algorithms as a pre-processing step could offer several benefits in refining chord recognition accuracy. By filtering out chords that do not conform to the established key of a musical piece, the system can focus its resources on analyzing and predicting chords that are harmonically coherent and musically relevant. This approach could streamline the chord recognition process and enhance the overall quality of predictions.

In conclusion, the future of chord recognition holds immense potential for advancement and refinement. By exploring these avenues of research and development, we can pave the way for more sophisticated and effective chord recognition systems that better serve the needs of musicians, researchers, and music enthusiasts alike.

Bibliography

[1] Puget Sound. *Intervals Introduction*, n.d.

[2] Keiron O'Shea και Ryan Nash. *An Introduction to Convolutional Neural Networks*. *arXiv preprint arXiv:1511.08458*, 2015.

[3] Purwono Purwono, Alfian Ma'arif, Wahyu Rahmaniar, Haris Imam, Haris Imam Karim Fathurrahman, Aufaclav Frisky και Qazi Mazhar Ul Haq. *Understanding of Convolutional Neural Network (CNN): A Review*. 2:739–748, 2023.

[4] Analytics Vidhya. *What is the Convolutional Neural Network (CNN) Architecture?*, 2020.

[5] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*, 2019.

[6] Ralf C. Staudemeyer και Eric Rothstein Morris. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*, 2019.

[7] Ivan Kukanov, Ville Hautamäki και Kong Aik Lee. *Maximal Figure-of-Merit Embedding for Multi-Label Audio Classification*. 2018.

[8] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong και Qing He. *A Comprehensive Survey on Transfer Learning*. *CoRR*, abs/1911.02685, 2019.

[9] OpenStax College. *FFT-Time-Frequency-View*. https://commons.wikimedia.org/wiki/File:FFT-Time-Frequency-View.png, 2013. Accessed: April 24, 2024.

[10] Hohyub Jeon, Yongchul Jung, Seongjoo Lee και Yunho Jung. *Area-Efficient Short-Time Fourier Transform Processor for Time-Frequency Analysis of Non-Stationary Signals*. *Applied Sciences*, 10:7208, 2020.

[11] Wikibooks contributors. *Music Theory/Chords*.

[12] Christopher Harte. *Towards automatic extraction of harmony information from music signals*. 2010.

[13] Stefan Gasser και Franz Strasser. *MIREX 2018: Multi Objective Chord Estimation*. Machine Learning and Audio: A Challenge, Johannes Kepler University, Linz, Austria, 2018. k1455519@students.jku.at.

[14] Jonggwon Park, Kyoyun Choi, Sungwook Jeon, Dokyun Kim και Jonghun Park. *A Bi-Directional Transformer for Musical Chord Recognition*. Department of Industrial Engineering & Center for Superintelligence, Seoul National University, Seoul, Republic of Korea, 2020. {jayg996, andyhome1907, wookee3, kdk01, jonghun}@snu.ac.kr.

[15] Junyan Jiang, Ke Chen, Wei Li και Guangyu Xia. *MIREX 2018 Submission: A Structural Chord Representation for Automatic Large-Vocabulary Chord Transcription*. Fudan University and New York University Shanghai, 2018. {jiangjy14, kchen15, weilifudan}@fudan.edu.cn, gxia@nyu.edu.

[16] H. Riemann. *Harmony Simplified: Or the Theory of the Tonal Functions of Chords*. Augener & Company, 1895.

[17] W. Piston. *Harmony*. WW Norton, New York, 1948.

[18] C. L. Krumhansl. *Cognitive Foundations of Musical Pitch*. Oxford University Press, 1990.

[19] F. Lerdahlet al. *Tonal Pitch Space*. Oxford University Press, USA, 2001.

[20] J. P. Bello και J. Pickens. *A Robust Mid-level Representation for Harmonic Content in Music Signals*. *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, τόμος 5, σελίδες 304–311. Citeseer, 2005.

[21] T. Crawford, J. Pickens και G. Wiggins. *Dimensionality Reduction in Harmonic Modeling for Music Information Retrieval*. *International Symposium on Computer Music Modeling and Retrieval*, σελίδες 233–248. Springer, 2005.

[22] J. Pauwels και G. Peeters. *Evaluating Automatically Estimated Chord Sequences*. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, σελίδες 749–753, 2013.

[23] Osvaldo Simeone. *A Very Brief Introduction to Machine Learning With Applications to Communication Systems*. *CoRR*, abs/1808.02342, 2018.

[24] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2ndη έκδοση, 2010.

[25] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1η έκδοση, 2007.

[26] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[27] Ilya Sutskever, Oriol Vinyals και Quoc V Le. *Sequence to Sequence Learning with Neural Networks*. *Advances in neural information processing systems*, 27:3104–3112, 2014.

[28] S. A. Singh, T. G. Meitei και S. Majumder. *Short PCG classification based on deep learning*. σελίδες 141–164, 2020.

[29] S. A. Suha και T. F. Sanam. *A deep convolutional neural network-based approach for detecting burn severity from skin burn images*. *Machine Learning with Applications*, 9(April):100371, 2022.

[30] C. Ding, Y. Li, Y. Xia, L. Zhang και Y. Zhang. *Automatic kernel size determination for deep neural networks based hyperspectral image classification*. *Remote Sensing*, 10(3), 2018.

[31] R. Riad, O. Teboul, D. Grangier και N. Zeghidour. *Learning strides in convolutional neural networks*. *International Conference on Learning Representations*, σελίδες 1–17, 2022.

[32] A. Nguyen, S. Choi, W. Kim, S. Ahn, J. Kim και S. Lee. *Distribution Padding in Convolutional Neural Networks*. *2019 IEEE International Conference on Image Processing (ICIP)*, σελίδες 4275–4279, 2019.

[33] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do και Kaori Togashi. *Convolutional neural networks: an overview and application in radiology*. *Insights into Imaging*, 9(4):611–629, 2018.

[34] Shiv Ram Dubey, Satish Kumar Singh και Bidyut Baran Chaudhuri. *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*, 2022.

[35] Junyoung Chung και et al. *Gated Feedback Recurrent Neural Networks*. *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, σελίδες 2067–2075, Lille, France, 2015. JMLR.org.

[36] Aston Zhang και et al. *Dive into Deep Learning*. 2019.

[37] *Accuracy (trueness and precision) of measurement methods and results - Part 1: General principles and definitions*, 1994.

[38] Music Information Retrieval Evaluation eXchange (MIREX). *Audio Chord Estimation*. https://www.music-ir.org/mirex/wiki/2019:Audio_Chord_Estimation, 2019.

[39] Mark French και Rod Handy. *Spectrograms: Turning Signals into Pictures*. *Journal of Engineering Technology*, 24:32–35, 2007.

List of Abbreviations

MIREX	Music Information Retrieval Evaluation eXchange
CNN	Convolutional neural network
BiLSTM	Bidirectional Long Short-Term Memory Network
STFT	Short-time Fourier transform
CQT	Constant-Q transform
WAV	Waveform Audio File Format
BTC	Bi-directional Transformer
CPU	Central Processing Unit