



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΕΙΚΟΝΑΣ ΚΑΙ ΣΗΜΑΤΩΝ

Generative Music: Seq2Seq Models for polyphonic enrichment

DIPLOMA THESIS

by

Pavlaki Paraskevi Evgenia

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2024



Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Πληροφορικής
Εργαστήριο Ψηφιακής Επεξεργασίας Εικόνας και Σημάτων

Generative Music: Seq2Seq Models for polyphonic enrichment

DIPLOMA THESIS

by

Pavlaki Paraskevi Evgenia

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29^η Μαρτίου, 2024.

.....
Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

.....
Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π.

.....
Αθανάσιος Βουλόδημος
Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2024

.....
ΠΑΥΛΑΚΗ ΠΑΡΑΣΚΕΥΗ ΕΥΓΕΝΙΑ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © – All rights reserved Pavlaki Paraskevi Evgenia , 2024.

Με επιφύλαξη παντός δικαιώματος.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η δημιουργία μουσικής στοχεύει την παραγωγή μουσικών κομματιών που συνδυάζουν ένα ευχάριστο και αρμονικό αποτέλεσμα με ένα σύνθετο και πλούσιο περιεχόμενο, δημιουργώντας στον ακροατή συναισθήματα. Η παραγωγή πολυφωνικής μουσικής μπορεί να μοντελοποιηθεί ως πολυδιάστατη παραγωγής γλώσσας, επομένως, η συμβολή των σύγχρονων μοντέλων, τεχνικών και προσεγγίσεων στην Επεξεργασία Φυσικής Γλώσσας μπορεί να είναι περισσότερο από ωφέλιμες αν εφαρμοστούν σε μουσικά δεδομένα. Στην παρούσα εργασία, προτείνουμε μια προσέγγιση παραγωγής μουσικής με Seq2Seq μοντέλα, σχεδιασμένη να εμπλουτίζει συνθέσεις, ενσωματώνοντας είτε μια άνω είτε μια κάτω φωνή σε δοσμένη μελωδία. Για να μετατρέψουμε το περιεχόμενο ενός MIDI αρχείου σε ακολουθία, χρησιμοποιούμε σύγχρονα εργαλεία κωδικοποίησης που παρουσιάζονται στο [9]. Αυτό μας επιτρέπει να αξιοποιήσουμε πλεονάζουσες τεχνικές που χρησιμοποιούνται συνήθως στην Επεξεργασία Φυσικής Γλώσσας, όπως η κωδικοποίηση ζεύγους, η επαύξηση δεδομένων και άλλες τεχνικές. Παρά το γεγονός ότι η αξιολόγηση της μουσικής αποτελεί αντικείμενο διαφωνίας μεταξύ των μελών της ερευνητικής κοινότητας, ο τομέας παραμένει ενεργός για ακαδημαϊκή διερεύνηση. Στη δική μας διαδικασία αξιολόγησης, εισάγουμε μια νέα προσέγγιση που ενσωματώνει την ποσοτική αξιολόγηση με ποιοτικά κριτήρια, με στόχο να γεφυρώσουμε το χάσμα μεταξύ αντικειμενικών μετρήσεων και ανθρώπινης αντίληψης. Καταλήγουμε σε ποσοτικές αξιολογήσεις που αντικατοπτρίζουν σε μεγάλο βαθμό εκείνες που προκύπτουν από την αντίστοιχη ανθρώπινη αξιολόγηση.

Λέξεις-κλειδιά — Παραγωγή μουσικής, Παραγωγικά Μοντέλα, Μετασχηματιστές, Μουσική, Συμβολική Μουσική, Μηχανική Μάθηση.

Abstract

Music generation aspires to produce musical pieces that have a pleasant harmonic result along with a complex and stimulating context, engaging listeners in emotional, intellectual and aesthetic dimensions. Polyphonic Music Generation can be modeled as multi-dimensional language generation task, thus, the contribution of state-of-the-art models, techniques and approaches in Natural Language Processing can be more than beneficial when applied in musical context. In this thesis, we propose a Seq2Seq generation approach, designed to enhance compositions by integrating additional voices, incorporating either an upper or lower voice to the given melody.

To transform symbolic music contained on a MIDI file into sequential format, suitable for transformer models, we employ modern encoding tools. This enables us to integrate advantages such as byte pair encoding, data augmentation, and other techniques commonly used in Natural Language Processing.

Despite music evaluation being a subject of disagreement among members of the research community, the field remains active for academic exploration. In our evaluation process, we introduce a novel approach that integrates quantitative assessment with qualitative criteria, aiming to bridge the gap between objective metrics and human perception.

Keywords — Music Generation, Conditional Generation, Harmonization, Seq2Seq, Transformers, Music, Machine Learning,

Ευχαριστίες

Αυτό το έργο δεν θα ήταν δυνατό χωρίς την υποστήριξη πολλών ανθρώπων. Ευχαριστώ πολύ τον επιβλέποντα μου, κ. Στάμου Γεώργιο, για την πολύτιμη καθοδήγηση του στην εκπόνηση αυτής της εργασίας. Ευχαριστώ επίσης την Έντυ Ντερβάχο για τη στενή συνεργασία και την υποστήριξη καθ' όλη τη διάρκεια εξερεύνησης των καινούριων αυτών αντικειμένων, καθώς και τους Σπύρο Κανταρέλη, Βασίλη Λυμπεράτο, Γεώργιο Φιλανδριανό που μοιράστηκαν τη δουλειά και τις ιδέες τους μαζί μου.

Η διπλωματική αυτή θα καθιστούνταν αδύνατη χωρίς τη συμβολή και την παρουσία όλων των μελών του εργαστηρίου AILS, εξασφαλίζοντάς μου ένα ευχάριτο, ζεστό και δημιουργικό περιβάλλον.

Το ταξίδι αυτό δεν θα ήταν το ίδιο χωρίς τη συνεισφορά και την ιδιαίτερη στήριξη των φίλων και συνεργατών μου, της Ιωάννας, της Μαρίνας και της Νεφέλης, οι οποίες στάθηκαν δίπλα μου σε όλη την διάρκεια αυτής της ακαδημαϊκής περιπέτειας.

Ιδιαίτερα θα ήθελα να ευχαριστήσω τους γονείς μου, Βαλεντίνα και Παύλο, που μου προσέφεραν τη δυνατότητα να κάνω τα όνειρά μου πραγματικότητα, αλλά και τον αδερφό μου το Γιώργο που συνοδεύει με την ιδιαίτερη μουσική παρουσία την πλούσια "ορχήστρα" της οικογένειάς μου.

Παυλάκη Παρασκευή Ευγενία,
Μάρτιος 2024

Contents

Contents	xi
List of Figures	xv
1 Εκτεταμένη Περίληψη στα Ελληνικά	1
1.1 Θεωρητικό υπόβαθρο	2
1.1.1 Μετασχηματιστές	2
1.1.2 Ο μετασχηματιστής T5	2
1.1.3 Μουσική	4
1.1.4 Miditok	7
1.1.5 Μέθοδοι αξιολόγησης	8
1.2 Πειράματικό Μέρος	9
1.2.1 Σύνολο Δεδομένων	9
1.2.2 Προεπεξεργασία Δεδομένων	10
1.2.3 Πειράματα	12
1.3 Συμπεράσματα	20
1.4 Συζήτηση	20
2 Introduction	21
3 Machine Learning	23
3.1 Learning Categories	24
3.2 Training a Machine Learning Model	24
3.2.1 Architecture of a Neural Network	24
3.2.2 The role of the architecture of a model	25
3.2.3 Loss Functions, Backpropagation and Activation Functions	26
3.2.4 Overfitting and hyperparameter fine-tuning	29
3.3 Natural Language Processing	30
3.3.1 Brief Course of Evolution	31
3.3.2 Recurrent Neural Networks	31
3.3.3 LSTMs and GRUs	31
3.4 Transformers	32
3.4.1 Attention Mechanism	32
3.4.2 Transformers Architecture	33
3.4.3 Pretraining	35
3.5 T5 Model	36
4 Music Theory and Data Representation	37
4.1 Theoretical Background of Music	38
4.1.1 Math, Sound and Perception	38
4.1.2 Harmonic resonance in music	38
4.2 Overview of Historical Periods	39
4.3 Music notation	42

4.4	Symbolic Music Representation	43
4.4.1	MIDI files	43
4.4.2	MusicXML files	45
4.4.3	Pianoroll	45
5	Tokenization	47
5.1	MidiTok	48
5.1.1	The Tokenization Process	48
5.2	MidiTok Tokenizers	49
5.2.1	REMI	49
5.2.2	MIDI-Like Tokenizer	49
5.2.3	TSD	50
5.2.4	Structured tokenizer	50
5.2.5	Embedding-Based Tokenization: Octuple, CPword, and MuMIDI	50
5.2.6	Byte Pair Encoding (BPE)	51
6	Evaluation Methods	53
6.1	Qualitative Evaluation	54
6.2	Quantitative Evaluation	54
6.2.1	Loss Functions	54
6.2.2	Tokenization Errors	55
6.2.3	Evaluation Metrics	55
6.2.4	Proposed Evaluation Metric	55
7	Experiments	57
7.1	Dataset - JSB Chorales	58
7.2	Data Preprocessing	58
7.2.1	MIDI file formatting	59
7.2.2	Tokenization	59
7.2.3	Prefix tokens	60
7.2.4	Data Augmentation	60
7.2.5	Byte Pair Encoding	60
7.3	Results	61
7.3.1	Parameter Selection	61
7.3.2	Test Set Generation	62
7.3.3	Applying proposed Evaluation Metric	63
7.3.4	Further Experiments: REMI, Structured & Tokenizers	68
8	Conclusion	69
8.1	Discussion	69
8.2	Future Work	70
9	Bibliography	71

List of Figures

1.1.1 Ο μετασχηματιστής - η αρχιτεκτονική του μοντέλου που προτάθηκε από [27]	3
1.1.2 Μετασχηματιστής T5: ένας ευέλικτος μετασχηματιστής κατάλληλος για πολυεργασιακή μάθηση	4
1.1.3 Αλλοιώσεις νοτών	5
1.1.4 Βασικές νότες σε ένα πεντάγραμμο	5
1.1.5 Συμβολικές αναπαραστάσεις μουσικής: μια αναπαράσταση παρτιτούρας, μια αναπαράσταση MIDI (σε απλουστευμένη, πινακοποιημένη μορφή) και μια αναπαράσταση ρολού πιάνου.	6
1.1.6 Φύλλο παρτιτούρας για απεικόνιση του συμβολοποιητή, όπως παρουσιάζεται στο	7
1.1.7 Απεικόνιση του συμβολοποιητή REMI	8
1.1.8 Απεικόνιση του συμβολοποιητή REMI	8
1.2.1 Κατανομή των χορικών του Bach με βάση το πλήθος φωνών.	10
1.2.2 Οπτική αναπαράσταση των τροποποιήσεων που υπέστησαν τα αρχεία MIDI κατά τη διάρκεια του μέρους της προεπεξεργασίας των δεδομένων που περιγράφεται στην ενότητα 7.2.1.	11
1.2.3 Συναρτήσεις απωλειών των πρώτων πειραμάτων 1.1. Αρχικά εκτελούμε πειράματα με παραμέτρους του μοντέλου σύμφωνα με τη βιβλιογραφία [8]. Επιλέξαμε ένα αρκετά μεγάλο εύρος τιμών, προκειμένου να διερευνήσουμε την κατεύθυνση περαιτέρω πειραματισμού.	12
1.2.4 Καμπύλη απωλειών των μοντέλων που εκπαιδεύτηκαν με TSD συμβολοποιητή και μέγεθος λεξιλογίου = 1500. Οι υπόλοιπες παράμετροι περιγράφονται στο 1.2	13
1.2.5 Καμπύλη απωλειών των μοντέλων που εκπαιδεύτηκαν με TSD συμβολοποιητή και μέγεθος λεξιλογίου = 2500. Οι υπόλοιπες παράμετροι περιγράφονται στο 1.2	14
1.2.6 Σύγκριση των κατανομών απόστασης μεταξύ των μετρικών που μετρήθηκαν στις παραγόμενες εκροές και της βασικής αλήθειας σε ολόκληρο το μουσικό κομμάτι (αριστερά) και ειδικά στη παραγόμενη φωνή (δεξιά).	15
1.2.7 Ανάλυση 16 μοντέλων, με χρήση του TSD tokenizer και των παραμέτρων που περιγράφονται στο 1.2: Κατανομή των μετρικών αποστάσεων μεταξύ των παραγόμενων εξόδων και της βασικής αλήθειας, που υπολογίστηκαν στο σύνολο των μουσικών κομματιών του συνόλου δοκιμών, με ρύθμιση της θερμοκρασίας στο 0,2	15
1.2.8 Analysis of 16 models using TSD tokenizer and parameters described on 1.2: Distribution of metric distances between generated outputs and ground truth, calculated specifically on the generated voice in the test set, with temperature settings ranging from 0.05 to 0.35.	16
1.2.9 Ανάλυση 16 μοντέλων με τη χρήση TSD συμβολοποιητή και παραμέτρων που περιγράφονται στο 1.2: Κατανομή των μετρικών αποστάσεων μεταξύ των παραγόμενων εξόδων και του δοσμένου που υπολογίστηκαν στο σύνολο των μουσικών κομματιών του συνόλου δοκιμών, με ρύθμιση της θερμοκρασίας στο 0,2.	17
1.2.10 Ανάλυση 16 μοντέλων με τη χρήση TSD συμβολοποιητή και παραμέτρων που περιγράφονται στο 1.2: Κατανομή των μετρικών αποστάσεων μεταξύ των παραγόμενων εξόδων και του δοσμένου, που υπολογίστηκαν ειδικά για την παραγόμενη φωνή στο σύνολο δοκιμών, με ρυθμίσεις θερμοκρασίας που κυμαίνονται από 0,05 έως 0,35	18
1.2.11 Μικρά μοντέλα που εκπαιδεύονται σε δεδομένα που έχουν επισημειωθεί με τους συμβολοποιητές REMI, Structured & TSD: Σύγκριση των κατανομών των αποστάσεων μεταξύ των μετρικών που μετρώνται στις παραγόμενες εξόδους και της βασικής αλήθειας σε ολόκληρο το μουσικό κομμάτι (αριστερά) και συγκεκριμένα στην παραγόμενη φωνή (δεξιά)	19
3.2.1	25

3.2.2 Examples of activation functions.	28
3.2.3 ReLU activation function and variants.	28
3.2.4 Comparison of a neural network before and after dropout is applied	30
3.3.1 Standard RNN and Unfolded Version [6]	31
3.3.2 RNN mathematical equations as given in [6]	32
3.4.1 The Transformer - a model architecture proposed by [27]	34
3.5.1 T5 Transformer: a flexible transformer suitable for multitask learning	36
4.1.1 Harmonic partials on strings	38
4.1.2 Harmonic Intervals of note C2	39
4.3.1 Basic notes on a staff. The pitch is defined by the vertical position between the staff lines and spaces.	42
4.3.2 Accidentals of a note	43
4.3.3 Note duration values and equivalent rests	43
4.4.1 Symbolic music representations: a sheet music representation, a MIDI representation (in a simplified, tabular form), and a piano-roll representation. [24]	44
4.4.2 Pianoroll Music Representation [28]	45
5.2.1 Lead Sheet for tokenizer illustration, as presented in	49
5.2.2 Illustration of REMI tokenizer	49
5.2.3 Illustration of REMI tokenizer	49
5.2.4 Illustration of TSD tokenizer	50
5.2.5 Illustration of Structured tokenizer	50
5.2.6 Illustration of Octuple, an embedding-based tokenizer	51
7.2.1 Distribution of JSB Chorales Dataset	58
7.2.2 Visual representation of the modifications that the MIDI files underwent during the data preprocessing part described in 7.2.1.	59
7.3.1 Loss functions of first experiments 7.1. We initially run experiments with model parameters according to the bibliography [8]. We chose a rather wide range of values, in order to explore the direction of further experimenting.	61
7.3.2 Loss curve of models trained with TSD tokenizer and vocabulary size = 1500. The rest of the parameters are described in 7.2	62
7.3.3 Loss curve of models trained with TSD tokenizer and vocabulary size = 2500. The rest of the parameters are described in 7.2	62
7.3.4 Comparison of distance distributions between metrics measured on generated outputs and ground truth across the entire musical piece (left) and specifically on the generated voice (right).	63
7.3.5 Analysis of 16 models, using TSD tokenizer and parameters described on 7.2: Distribution of metric distances between generated outputs and ground truth, calculated across the entirety of musical pieces in the test set, with a temperature setting at 0.2	64
7.3.6 Analysis of 16 models using TSD tokenizer and parameters described on 7.2: Distribution of metric distances between generated outputs and ground truth, calculated specifically on the generated voice in the test set, with temperature settings ranging from 0.05 to 0.35.	65
7.3.7 Analysis of 16 models using TSD tokenizer and parameters described on 7.2: Distribution of metric distances between generated outputs and ground truth, calculated across the entirety of musical pieces in the test set, with a temperature setting at 0.2.	66
7.3.8 Analysis of 16 models using TSD tokenizer and parameters described on 7.2: Distribution of metric distances between generated outputs and ground truth, calculated specifically on the generated voice in the test set, with temperature settings ranging from 0.05 to 0.35.	67
7.3.9 Small models trained on data tokenized with REMI, Structured & TSD tokenizers: Comparison of distance distributions between metrics measured on generated outputs and ground truth across the entire musical piece (left) and specifically on the generated voice (right).	68

Chapter 1

Εκτεταμένη Περίληψη στα Ελληνικά

1.1 Θεωρητικό υπόβαθρο

Για αιώνες, η κατασκευή μιας μηχανής ικανής να αναπαράγει την ανθρώπινη λογική και αντίληψη αποτελούσε φιλοδοξία της ανθρωπότητας. Η τεχνητή νοημοσύνη έχει οραματιστεί σε όλη τη διάρκεια της ιστορίας από πολλούς πολιτισμούς. Στη ραγδαία τεχνολογική εξέλιξη του 20ού αιώνα, οι επιστήμονες προσπαθούν να κατανοήσουν τον μηχανισμό των ανθρώπινων συστημάτων αναγνώρισης, να τον ερμηνεύσουν με μαθηματικές αρχές και να τον μεταφράσουν σε λογικές ακολουθίες. [20]. Έτσι γεννήθηκε ο τομέας της μηχανικής μάθησης.

Η μηχανική μάθηση περιστρέφεται γύρω από την έννοια της δημιουργίας υπολογιστικών συστημάτων που μπορούν να μάθουν και να κατανοήσουν τον κόσμο. Είναι ένας τομέας μελέτης στο πλαίσιο της τεχνητής νοημοσύνης που χρησιμοποιεί πιθανότητες, στατιστική, διαφορικές εξισώσεις προκειμένου να δημιουργήσει μοντέλα που είναι ικανά να ολοκληρώσουν συγκεκριμένες εργασίες. Αυτές οι εργασίες περιλαμβάνουν τη λήψη αποφάσεων, την πρόβλεψη, την επισήμανση, τη δημιουργία, τη βελτιστοποίηση και άλλα. Με απλούστερους όρους, ένα μοντέλο μηχανικής μάθησης "εκπαιδεύεται" σε δεδομένα, με στόχο τον εντοπισμό μοτίβων και σχέσεων εντός των δεδομένων εισόδου και την εξαγωγή των μαθηματικών καμπυλών που τα καθορίζουν - τόσο σε γενική όσο και σε λεπτομερή μορφή. Ο απώτερος στόχος είναι να βελτιωθεί η απόδοση του μοντέλου με την πάροδο του χρόνου, προσαρμόζοντας τις παραμέτρους του με κάθε εποχή εκπαίδευσης, παρόμοια με τον άνθρωπο που μαθαίνει από την εμπειρία.

Με την πάροδο του χρόνου, ο τομέας εξελίχθηκε από το να στοχεύει αρχικά στην επίλυση μαθηματικά απαιτητικών διαδικασιών, στο να φιλοδοξεί τώρα να αποκτήσει την ανθρώπινη αντίληψη και να μιμηθεί τις ανθρώπινες συλλογιστικές διαδικασίες. Η μηχανική μάθηση έχει εξελιχθεί σε ένα ευέλικτο εργαλείο που μπορεί να εφαρμοστεί σε πολυάριθμους επιστημονικούς τομείς, υπηρεσίες και στην καθημερινή ζωή του ατόμου. Οι αρχές του 21ου αιώνα ήταν μια εποχή επανάστασης, καθώς η έρευνα ενσωματώνεται σε μεγάλο αριθμό εφαρμογών, όπως η ταξινόμηση εικόνων, τα συστήματα συστάσεων, οι αλγόριθμοι βελτιστοποίησης, η επεξεργασία φυσικής γλώσσας αλλά και η ανάλυση και σύνθεση μουσικής.

1.1.1 Μετασχηματιστές

Οι μετασχηματιστές, ή transformers, αποτελούν μια σύγχρονη προσέγγιση στη μηχανική μάθηση, χαρακτηριζόμενη από τη χρήση προηγμένων μοντέλων νευρωνικών δικτύων. Κεντρικό στοιχείο αυτών των μοντέλων είναι οι μηχανισμοί αυτοπροσοχής και προσοχής, οι οποίοι επιτρέπουν στα μοντέλα να αντιληφθούν και να αξιοποιήσουν την πληροφορία από διάφορες πηγές μεγάλου μήκους σε είσοδο και έξοδο.

Η ευελιξία των μετασχηματιστών έγκειται στην ικανότητά τους να προσαρμόζονται σε διάφορες αρχιτεκτονικές παραλλαγές, η κάθε μία προσαρμοσμένη σε συγκεκριμένα καθήκοντα και απαιτήσεις.

Τα μοντέλα **κωδικοποιητή-αποκωδικοποιητή**, με παράδειγμα τον T5, περιλαμβάνουν την πλήρη αρχιτεκτονική του μετασχηματιστή, επεξεργάζονται ακολουθίες εισόδου για να παράγουν ακολουθίες εξόδου, γεγονός που τους καθιστά ιδανικούς για εργασίες κειμένου-σε-κειμένο, όπως η μηχανική μετάφραση και η παραγωγή ακολουθιακών δεδομένων.

Μοντέλα **μόνο με αποκωδικοποιητή**, που αντιπροσωπεύονται από το GPT2, επικεντρώνονται αποκλειστικά στην αποκωδικοποίηση, προβλέποντας το επόμενο σύμβολο σε μια ακολουθία χωρίς πλαίσιο από έναν κωδικοποιητή. Αυτή η αρχιτεκτονική χρησιμοποιείται συνήθως σε εργασίες παραγωγής κειμένου και περίληψης.

Μοντέλα **μόνο με κωδικοποιητή**, όπως το BERT, αξιοποιούν τον μηχανισμό αυτοπροσοχής των μετασχηματιστών για να εξάγουν πληροφορίες από τις ακολουθίες εισόδου. Διακρίνονται σε εργασίες γλωσσικής κατανόησης, όπως η επισήμανση ακολουθιών και η ανάλυση συναισθήματος.

Αυτό τους επιτρέπει να αντιμετωπίσουν αποτελεσματικά πολύπλοκα προβλήματα σε διάφορους τομείς, όπως η επεξεργασία χειριστών δεδομένων, η γλωσσική αναγνώριση και η ανάλυση δεδομένων που εξελίσσονται με το χρόνο. Η ευελιξία και η αποδοτικότητά τους τους καθιστούν κρίσιμους στην ανάπτυξη προηγμένων εφαρμογών μηχανικής μάθησης σε ποικίλους τομείς.

1.1.2 Ο μετασχηματιστής T5

Το μοντέλο T5 είναι ένα μοντέλο κειμένου-σε-κειμένο, που αναπτύχθηκε από την Google AI και βασίζεται στην αρχιτεκτονική Κωδικοποιητή-Αποκωδικοποιητή. Ο κωδικοποιητής χρησιμοποιεί μηχανισμούς αυτοπροσοχής για

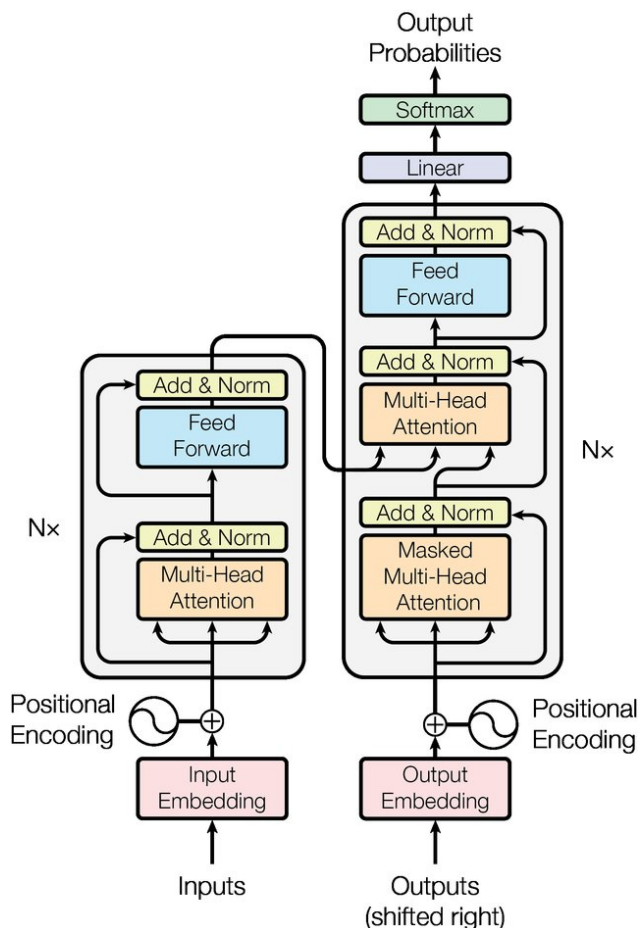


Figure 1.1.1: Ο μετασχηματιστής - η αρχιτεκτονική του μοντέλου που προτάθηκε από [27]

τη σύλληψη πληροφοριών εντός της ακολουθίας εισόδου, γεγονός που επιτρέπει στο μοντέλο να κατανοεί και να κωδικοποιεί αποτελεσματικά τις αποχρώσεις του κειμένου εισόδου. Αντίθετα, ο Αποκωδικοποιητής χρησιμοποιεί την αυτοπροσοχή, ενώ παράλληλα παρακολουθεί την έξοδο του Κωδικοποιητή, επιτρέποντάς του να παράγει συνεκτικές και συναφείς με τα συμφραζόμενα ακολουθίες εξόδου.

Η αρχιτεκτονική T5 υπερέρχει στην καταγραφή μακροπρόθεσμων αλληλεξαρτήσεων, καθιστώντας την κατάλληλη για εργασίες που απαιτούν κατανόηση εκτεταμένου πλαισίου.

Λειτουργώντας σε ακολουθίες εισόδου και εξόδου σταθερού μήκους, το μοντέλο T5 αναπαριστά τόσο την είσοδο όσο και την έξοδο ως ακολουθίες κειμένου, διευκολύνοντας την απρόσκοπτη επεξεργασία από κείμενο σε κείμενο. Ο πολυταξικός χαρακτήρας του επιτρέπει την ταυτόχρονη εκπαίδευση σε διαφορετικές εργασίες, προωθώντας την ανταλλαγή γνώσεων σε διάφορους τομείς και διευκολύνοντας τη χρήση μιας αντικειμενικής συνάρτησης απώλειας.

Όλα τα παραπάνω καθιστούν τον T5 ένα ευέλικτο εργαλείο για διερεύνηση πολλαπλών εργασιών, κατάλληλο για ποικίλες εργασίες και προσφέροντας ευφορο έδαφος για εξερεύνηση.

Προεκπαίδευση του T5

Παρόμοια με πολλούς άλλους μετασχηματιστές, ο T5 μπορεί να προ-εκπαιδευτεί με αυτο-επιβλεπόμενο ή με επιβλεπόμενο τρόπο και αργότερα να "καλιμπραριστεί" λεπτομερώς για συγκεκριμένες εργασίες.

Μία προσέγγιση προ-εκπαίδευσης είναι αυτή να γίνει με αυτο-επιβλεπόμενο τρόπο σε ένα εκτεταμένο σώμα δεδομένων κειμένου, προκειμένου να μάθει από ένα τεράστιο φάσμα γλωσσικών προτύπων, δομών, και αλληλεξαρτήσεων.

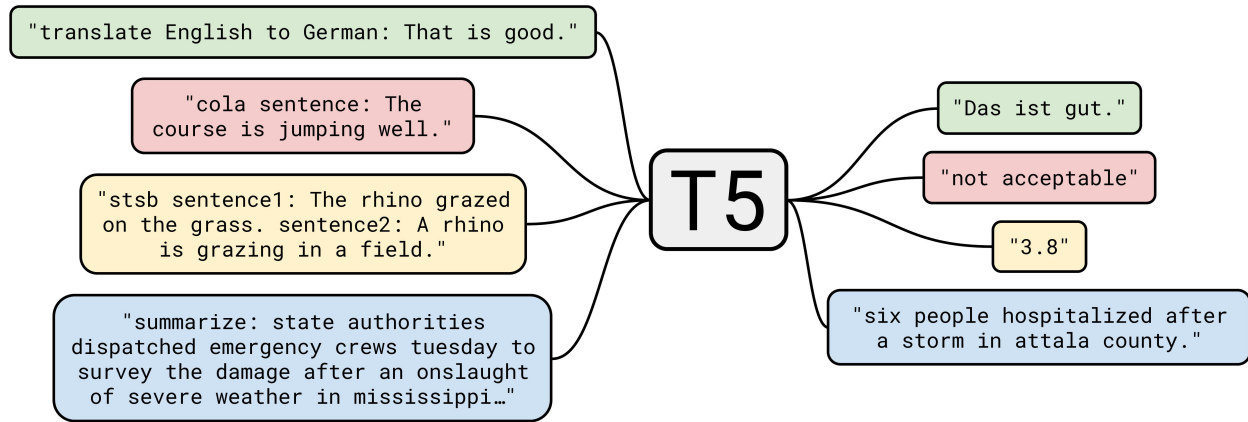


Figure 1.1.2: Μετασχηματιστής T5: ένας ευέλικτος μετασχηματιστής κατάλληλος για πολυεργασιακή μάθηση

Μια πιο επιβλεπόμενη προσέγγιση για την προεκπαίδευση, είναι η προεκπαίδευση από κείμενο-σε-κείμενο: τα δεδομένα διαμορφώνονται σε ένα πλαίσιο ερώτησης-απάντησης, όπου κάθε περίπτωση εκπαίδευσης περιέχει τόσο ακολουθίες κειμένου εισόδου όσο και εξόδου. Αυτή η προσέγγιση διευκολύνει την ικανότητα του μοντέλου να κατανοεί και να παράγει κείμενο σε διάφορες εργασίες.

Το Masked Language Modeling (MLM) χρησιμεύει ως πρόσθετη τεχνική προεκπαίδευσης με στόχο την αύξηση της κατανόησης του γλωσσικού πλαισίου από το μοντέλο. Κατά τη διάρκεια αυτής της διαδικασίας, ένα τυχαίο υποσύνολο από σημεία εντός της ακολουθίας εισόδου αποκρύπτεται, προτρέποντας το μοντέλο να προβλέψει αυτά τα αποκρυπτόμενα σημεία, αντλώντας από τις πληροφορίες πλαισίου που είναι ενσωματωμένες στο περιβάλλον κείμενο. Σε αυτό τον τρόπο προεκπαίδευσης, το μοντέλο οξύνει την ικανότητά του να συμπεραίνει τις πληροφορίες που λείπουν και να αυξάνει τη συνολική του ικανότητα κατανόησης περιεχομένου.

1.1.3 Μουσική

Μουσική Αντίληψη

Ο καθε ένας μας αντιλαμβάνεται τη μουσική με διαφορετικό τρόπο, έχει διαφορετικές προτιμήσεις, ωστόσο όλοι μας μπορούμε να διακρίνουμε έναν διάφωνο από έναν σύμφωνο ήχο, όπως για παράδειγμα ένα διάστημα 5ης Καθαρό από ένα διάστημα 4ης Αυξημένο. Η ικανότητα μας να διαφοροποιούμε τους δύο αυτούς ήχους και να δείχνουμε προτίμηση στον πρώτο, είναι αυτό που οδήγησε τον άνθρωπο να δημιουργήσει μουσική.

Η γλωσσική με τη μουσική ικανότητα του ανθρώπου αναπτύχθηκαν παράλληλα, και οι δύο μοιράζονται παρόμοιες δομές και χαρακτηριστικά:

- έχουν και οι δύο διαδοχική φύση που εξαρτάται από το χρόνο
- αναπαριστώνται από ακολουθίες συμβόλων και
- διέπονται από κανόνες, η μὲν γραμματικούς, συντακτικούς και εννοιολογικούς, η δὲ από αρμονικούς και μελωδικούς

Η επικάλυψη μεταξύ αυτών των πεδίων ενθαρρύνει τη διεπιστημονική έρευνα, επιτρέποντας στους ερευνητές να αξιοποιήσουν τα υπάρχοντα μοντέλα και τις μεθοδολογίες στην επεξεργασία φυσικής γλώσσας για να προσεγγίζουν την παραγωγή μουσικής με έναν αλγοριθμικό τρόπο.

Δομή της μουσικής

Η ανάπτυξη της μουσικής σημειογραφίας προήλθε από την ανάγκη του ανθρώπου να επικοινωνεί και να αναπαριστά την μουσική, με αποτέλεσμα να δημιουργηθούν διάφορα συστήματα αναπαράστασης μουσικής σε διάφορους πολιτισμούς και κουλτούρες. Αρχικά, τα συστήματα αυτά εξελίχθηκαν με την επιδίωξη της χριστιανικής εκκλησίας για συνοχή στις ψαλμωδίες, η οποία εξελίχθηκε και έθεσε τα θεμέλια για την καθιέρωση της δυτικής μουσικής σημειογραφίας.

Στον πυρήνα της μουσικής σημειογραφίας βρίσκεται το **πεντάγραμμο**, το οποίο αποτελείται από πέντε γραμμές και τέσσερα κενά όπου τοποθετούνται τα μουσικά σύμβολα, καθένα από τα οποία αντιστοιχεί σε ένα συγκεκριμένο τόνο που καθορίζεται από ένα **μουσικό κλειδί**. Το κλειδί αυτό βρίσκεται πάντα στην αρχή του πενταγράμμου, καθορίζει το εύρος του τόνου για το μουσικό κομμάτι που ακολουθεί.

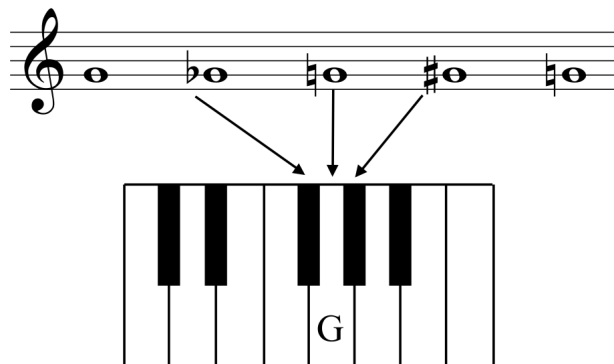


Figure 1.1.3: Αλλοιώσεις νοτών

Μετά το κλειδί, ο οπλισμός υπογράφει την τονικότητα του κομματιού, η οποία αντιπροσωπεύεται από τα σύμβολα διέσεων, υφέσεων και αναιρέσαιων, και ο ρυθμός του κομματιού που ορίζει τις χρονικές αξίες που θα περιλαμβάνονται εντός ενός μέτρου.

Οι νότες είναι τα οβάλ σύμβολα που, ανάλογα τη θέση τους στο πεντάγραμμο και τη μορφή του μαστουνού τους, έχουν διαφορετικό τονικό ύψος και διάρκεια ¹.

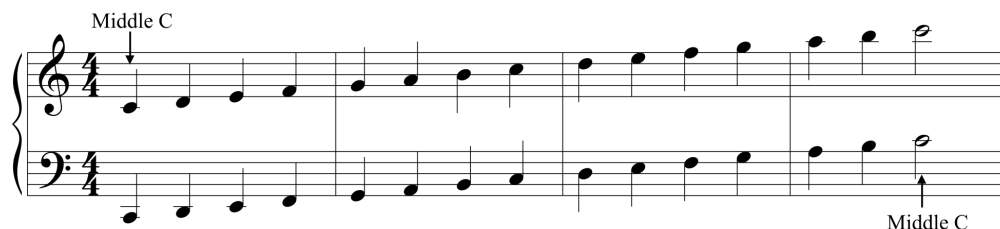


Figure 1.1.4: Βασικές νότες σε ένα πεντάγραμμο

Ένα μουσικό κομμάτι διαβάζεται και γράφεται από αριστερά προς τα δεξιά. Ανάλογα με το όργανο στο οποίο αναφέρεται το μουσικό κομμάτι, η μουσική σημειογραφία μπορεί να περιλαμβάνει αρμονικά διαστήματα (συγχορδίες) ή μελωδικά διαστήματα (μελωδία). Για παράδειγμα, ένα μονοφωνικό όργανο, όπως το φλάουτο, περιορίζεται στο να παίζει μία νότα κάθε φορά, οπότε η μουσική σημειογραφία αποτελείται μόνο από μια μελωδία γραμμένη σε πεντάγραμμο. Από την άλλη πλευρά, το πιάνο χρησιμοποιεί ένα ευρύτερο φάσμα τόνων κάθε φορά μέσα σε ένα μουσικό κομμάτι. Συνήθως οι παρτιτούρες του πιάνου χαρακτηρίζονται από τη δομή διπλής παρτιτούρας, την ευελιξία για πολυφωνική γραφή και τη συμπερίληψη υπογραφών κλειδιών, αλλαγών κλειδιού, σημάνσεων πεντάλ. Αρκετές μουσικές συνθέσεις περιλαμβάνουν επίσης κατευθύνσεις εκτέλεσης, όπως δυναμική, tempo και σημάνσεις έκφρασης, για να διευκολύνουν την ακριβή και εκφραστική εκτέλεση στο πιάνο.

Η σύνθεση μουσικής έχει ως στόχο την οργάνωση των αρμονικών και μελωδικών στοιχείων ώστε να παραχθεί ένα αποτέλεσμα που να έχει:

- αρμονική συνάφεια, ευχάριστο ηχητικό αποτέλεσμα σε χρονική στιγμή t
- πλούσιο και ενδιαφέρον μελωδικό περιεχόμενο, σε διάστημα χρόνου δt

Αυτό καθιστά την παραγωγή μουσικής, και ιδιαίτερα πολυφωνικής μουσικής, ως μια πολύπλοκη και πολυδιάστατη εργασία παραγωγής λόγου.

¹<https://musictheory.pugetsound.edu/mt21c/OctaveRegisters.html>

Αρχεία μουσικής και ψηφιακές μουσικές αναπαραστάσεις

Η μουσική μπορεί να αναπαρασταθεί στο πλαίσιο υπολογιστικών μηχανών με τη χρήση διαφόρων μορφών αποθήκευσης που χρησιμοποιούν διαφορετικές μορφές δεδομένων.

Στα πλαίσια αυτής της διπλωματικής θα εστιάσουμε στη συμβολική αναπαράσταση αρχείων μουσικής με αρχεία MIDI.

Το MIDI (Musical Instrumental Interface) [29] είναι ένα τεχνικό πρότυπο που περιγράφει ένα πρωτόκολλο επικοινωνίας, μια ψηφιακή διεπαφή και ηλεκτρικές συνδέσεις που συνδέουν μια μεγάλη ποικιλία ηλεκτρονικών μουσικών οργάνων, υπολογιστών και συναφών συσκευών ήχου για την αναπαραγωγή, την επεξεργασία και την εγγραφή μουσικής.

Σε μια απλουστευτική άποψη, ένα αρχείο MIDI μπορεί να θεωρηθεί ως *παρτιτούρα με πρόσθετες προαιρετικές σημειώσεις*.

Τα αρχεία MIDI αποτελούνται κυρίως από ψηφιακές εντολές που αντιπροσωπεύουν μουσικά γεγονότα, όπως μηνύματα για την ενεργοποίηση και το κλείσιμο της νότας, την ταχύτητα, το ύψος και τη διάρκεια, όπως φαίνεται στο 1.1.5. Χρησιμεύει ως μορφή για τη συγγραφή μουσικής, επιτρέποντας την προσομοίωση συνθέσεων και διευκολύνοντας την ποικιλία και την επικοινωνία μεταξύ διαφορετικών οργάνων, επιτρέποντάς τους να ελέγχουν το ένα το άλλο [24].

Figure 1.13 from
[Müller, FMP, Springer 2015]



Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0

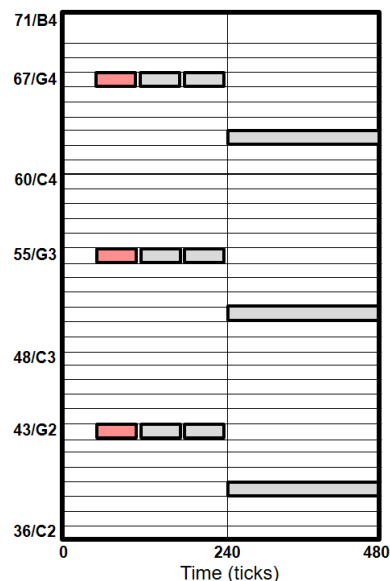


Figure 1.1.5: Συμβολικές αναπαραστάσεις μουσικής: μια αναπαράσταση παρτιτούρας, μια αναπαράσταση MIDI (σε απλουστευμένη, πινακοποιημένη μορφή) και μια αναπαράσταση ρολού πιάνου.

Οι μεταβλητές που εξοπλίζουν τα μουσικά γεγονότα περιγράφουν τα χαρακτηριστικά της νότας που παίζεται σε μια συγκεκριμένη χρονική στιγμή.

Ο αριθμός τονικού ύψους (pitch) είναι ένας ακέραιος αριθμός που κυμαίνεται μεταξύ 0 και 127 και κωδικοποιεί το ύψος μιας νότας. Αντικατοπτρίζοντας την ευθυγράμμιση του ακουστικού πιάνου, όπου υπάρχουν 88 πλήκτρα που αντιστοιχούν σε μουσικά ύψη που κυμαίνονται από τις νότες A0 έως C8, ο αριθμός MIDI κωδικοποιεί τα μουσικά ύψη C0 έως G9 με αυξανόμενη σειρά. Για παράδειγμα, το μεσαίο C αντιστοιχεί στον αριθμό τονικού ύψους 60.

Η ταχύτητα του πλήκτρου συμβολίζεται και πάλι με έναν ακέραιο αριθμό που κυμαίνεται από 0 έως 127, ο οποίος ελέγχει την ένταση της νότας που παίζεται. Σε συμβάντα note-on, ρυθμίζει την ένταση, ενώ σε συμβάντα note-off, διαχειρίζεται την εξασθένιση κατά τη φάση της απελευθέρωσης της νότας. Ωστόσο, η ακριβής επίδρασή της ποικίλλει ανάλογα με το όργανο ή το συνθεσάιζερ που χρησιμοποιείται.

Το κανάλι MIDI (program) λειτουργεί ως μέσο οργάνωσης και δρομολόγησης δεδομένων MIDI για τον έλεγχο

διαφόρων οργάνων ή ήχων εντός μιας εγκατάστασης MIDI. Συνήθως ένα κανάλι συνδέεται με ένα συγκεκριμένο όργανο, αν και δεν είναι υποχρεωτικό. Το κανάλι είναι επίσης ένας ακεραίος αριθμός που κυμαίνεται από 0-15.

Πέραν της ευελιξία τους, ένας λόγος για τον οποίο χρησιμοποιούμε αρχεία MIDI, είναι το μεγάλο εύρος δεδομένων που διατίθεται σε αυτή τη μορφή, γεγονός που μας δίνει τη δυνατότητα να επεκτείνουμε την έρευνα μας σε πολλές κατευθύνσεις.

1.1.4 Miditok

Η εμφάνιση των Γλωσσικών Μοντέλων έφερε επανάσταση στην αυτόματη παραγωγή μουσικής, προκαλώντας τη μετάβαση στη μουσική αναπαράσταση από πολύπλοκα συστήματα αναπαραστάσεων σε μια πιο γλωσσική μορφή. Αυτή η μετατόπιση οδήγησε στην πρόταση της **συμβολοποίησης για τη συμβολική μουσική**, ένα κρίσιμο βήμα προεπεξεργασίας για την εισαγωγή δεδομένων MIDI σε Νευρωνικά Δίκτυα. Δημοφιλοποιημένη στα τέλη της δεκαετίας του 2010 και στις αρχές της δεκαετίας του 2020 με την άνοδο της αρχιτεκτονικής του μετασχηματιστή, η ιδέα απέκτησε απήχηση.

Στα τέλη του 2021, οι Fradet et al [9]. παρουσίασαν το **Miditok**, ένα πακέτο Python σχεδιασμένο για να μετασχηματίζει τη μουσική γλώσσα από αρχεία MIDI σε ακολουθίες συμβόλων. Λειτουργώντας ως εργαλειοθήκη ανοικτού κώδικα, το Miditok παρέχει στους ερευνητές μια φιλική προς το χρήστη προσέγγιση για τη μετατροπή αρχείων MIDI σε token, ενθαρρύνοντας έτσι τη μεγαλύτερη συμμετοχή στον τομέα της δημιουργίας μουσικής. Η πλατφόρμα προσφέρει διάφορους συμβολοποιητές και βασικές λειτουργίες προεπεξεργασίας, όπως αύξηση δεδομένων, συμβολοποίηση συνόλου δεδομένων και Byte Pair Encoding.

Η κωδικοποίηση των αρχείων MIDI είναι απαραίτητη για την προετοιμασία των δεδομένων για την εκπαίδευση των μοντέλων με την αντιμετώπιση της πρόκλησης του μεγάλου εύρους τιμών που υπάρχουν στις μουσικές πληροφορίες. Τα αρχεία MIDI περιέχουν διάφορα χαρακτηριστικά, όπως το τονικό ύψος και η ταχύτητα, τα οποία αναπαρίστανται ως συμβάντα νότας που συμβαίνουν σε συγκεκριμένα χρονικά σημεία. Με την κωδικοποίηση αυτών των πληροφοριών, τα συνεχή και ποικίλα χαρακτηριστικά υποβαθμίζονται και χβαντίζονται σε διακριτές τιμές, μειώνοντας την πολυπλοκότητα και βελτιστοποιώντας την απόδοση του μοντέλου. Ενώ η υποδειγματοληψία των τιμών της ταχύτητας βοηθά στην αποτελεσματική σύλληψη των αποχρώσεων της δυναμικής, η διατήρηση του *πλήρους εύρους του τόνου* είναι ζωτικής σημασίας για την επαρκή αναπαράσταση της μουσικής. Συνολικά, η συμβολοποίηση επιτρέπει στα μοντέλα να μαθαίνουν μοτίβα αποτελεσματικά, να γενικεύουν πληροφορίες και να κάνουν προβλέψεις πιο αποτελεσματικά.

Το Miditok γεφυρώνει το χάσμα μεταξύ των εργασιών Παραγωγής μουσικής και της Επεξεργασίας Κειμένου, επιτρέποντας στους ερευνητές να αξιοποιήσουν τα Γλωσσικά Μοντέλα για να ανταποκριθούν στις απαιτήσεις των εργασιών συνυφασμένες με την μουσική. Το Miditok αποτελεί συνδετικό κόμβο μεταξύ δεδομένων μουσικής και γλωσσικών μοντέλων και σε αυτή τη διπλωματική. Τις απαραίτητες πληροφορίες και οδηγίες για τη χρήση όλων των απαραίτητων εργαλείων αντλήσαμε από τη συνεχώς συντηρούμενη και ενημερωμένη πλατφόρμα του Miditok, η οποία ανατροφοδοτείται και βελτιώνεται συνεχώς [9, 7].

Για τις ανάγκες αυτής της περίληψης, θα αναφέρουμε μερικές βασικές πληροφορίες για τους δύο πιο χρησιμους κωδικοποιητές που χρησιμοποιήσαμε.

Κωδικοποιητής REMI



Figure 1.1.6: Φύλλο παρτιτούρας για απεικόνιση του συμβολοποιητή, όπως παρουσιάζεται στο ²

Ο συμβολοποιητής REMI, εισήχθη με τον Pop Music Transformer στο [14]. Ο REMI αναπαριστά τις μουσικές νότες διαδοχικά, μετατρέποντας τα δεδομένα MIDI σε tokens Pitch, Velocity και Duration για τις νότες και σε tokens Bar και Position για το χρόνο.

Η εκτεταμένη έκδοση REMI+ μπορεί να χειριστεί πολλαπλά όργανα προσθέτοντας μάρκες Program πριν από τις μάρκες Pitch. Οι πληροφορίες ρυθμού αναπαρίστανται ως ένα μόνο Tempo token. Κατά την αποκωδικοποίηση

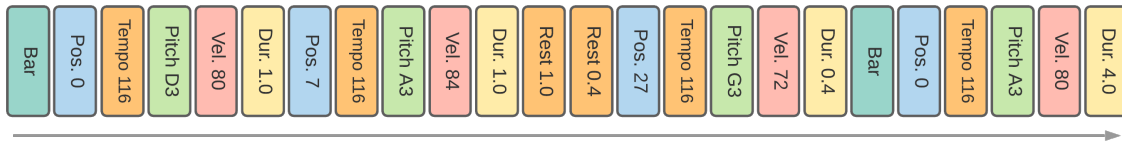


Figure 1.1.7: Απεικόνιση του συμβολοποιητή REMI

πολλαπλών ακολουθιών συμβόλων, αποκωδικοποιούνται μόνο οι ρυθμοί και οι χρονικές υπογραφές της πρώτης ακολουθίας για ολόκληρο το MIDI.

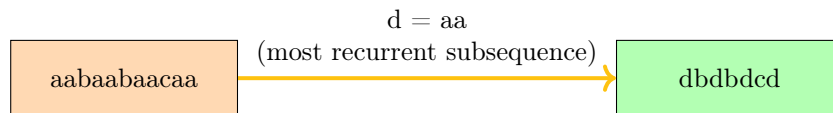
Κωδικοποιητής TSD



Figure 1.1.8: Απεικόνιση του συμβολοποιητή REMI

Αντιμετωπίζοντας τις προκλήσεις των Note-Off συμβόλων και της δυσκολίας της χρονικής αναπαράστασης, ο συμβολοποιητής TSD, ή Time Shift Duration tokenizer, αναπαριστά τα δεδομένα MIDI χρησιμοποιώντας ρητά σύμβολα διάρκειας (duration tokens) για τις διάρκειες των νοτών. Όπως και οι δύο προηγούμενοι, μπορεί να συμπεριλάβει σύμβολα Program πριν από κάθε μάρκα Pitch για να καθορίσει πληροφορίες οργάνου, εάν έχει την κατάλληλη ρύθμιση.

Κωδικοποίηση σε ζεύγη



Η κωδικοποίησης ζεύγους χρησιμοποιούνται συνήθως στην επεξεργασία φυσικής γλώσσας. ρόκειται για μια τεχνική συμπίεσης που αντικαθιστά επαναλαμβανόμενες ακολουθίες χαρακτήρων σε ένα σώμα κειμένου με νεοδημιουργημένα σύμβολα, επιτρέποντας την κωδικοποίηση σπάνιων λέξεων και την κατάτμηση άγνωστων ή σύνθετων λέξεων σε υπο-λεξιλογικές μονάδες.

Στη συμβολική μουσική, η κωδικοποίηση ζεύγους ενισχύει τα λεξιλόγια συμβολισμού με την ομαδοποίηση των βασικών συμβολισμών σε νέα σύμβολα, με αποτέλεσμα πιο συμπαγείς αναπαραστάσεις [8]. Η κωδικοποίηση και συμπίεση των αρχείων αυτών διευκολύνει τη γρήγορη εκπαίδευση, ενισχύοντας την απόδοση των μοντέλων σε συμβολικές μουσικές εργασίες διευκολύνοντας την εκμάθηση αναπαραστάσεων.

Είναι σημαντικό να εφαρμόζεται με συνέπεια σε όλα τα σύνολα δεδομένων που αλληλεπιδρούν με ένα μοντέλο, συμπεριλαμβανομένων των συνόλων εκπαίδευσης, επικύρωσης και δοκιμής, για να διασφαλιστεί η βέλτιστη απόδοση.

1.1.5 Μέθοδοι αξιολόγησης

Η διαδικασία αξιολόγησης στην παραγωγή μουσικής αποτελεί μια σύνθετη πρόκληση, δεδομένης της υποκειμενικής φύσης της μουσικής αντίληψης και των πολύπλευρων πτυχών της μουσικής ποιότητας. Παρά την ύπαρξη αντικειμενικών κανόνων για την αντίστιξη, τη σύνθεση και τη μουσική εναρμόνιση, ο καθορισμός ενός αντικειμενικού επιπέδου για τη σύγκριση και την αξιολόγηση παραμένει θέμα διαφωνίας στην ερευνητική κοινότητα, αλλά και ένα ενεργά εξεταζόμενο ακαδημαϊκό πεδίο. Η αξιολόγηση εργασιών παραγωγής μουσικής περιλαμβάνει συνήθως ποιοτικές αξιολογήσεις ακρόασης και ποσοτικές αναλύσεις.

Ποιοτικές Αξιολογήσεις

Ενώ είναι δύσκολο να καθοριστούν καθολικά κριτήρια για την αξιολόγηση της ποιότητας της μουσικής, η ανθρώπινη αξιολόγηση προσφέρει γνώσεις πέρα από απλές ποσοτικές μετρήσεις. Στη διαδικασία αξιολόγησης συμμετέχουν συνήθως εκπαιδευμένοι μουσικοί, συνθέτες, μουσικολόγοι και άτομα με διαφορετικά επίπεδα εμπειρογνωμοσύνης.

Οι μετρικές για την ανθρώπινη αξιολόγηση σε εργασίες δημιουργικής μουσικής συχνά περιλαμβάνουν την μουσική ορθότητα, τη συνεκτική ποικιλομορφία, το ενδιαφέρον του μουσικού περιεχομένου και τη συνολική υποκειμενική προτίμηση. Για την αποδοτικότερη και περισσότερο κατευθηνμένη αξιολόγηση, γίνεται διάσπαση της διαδικασίας σε συγκεκριμένες ερωτήσεις που αφορούν διαφορετικών διαστάσεων της ποιότητας της μουσικής, περιλαμβανομένου πτυχών όπως η συνοχή, ο πλούτος και η διαρρύθμιση.

Ποσοτικές Αξιολογήσεις

Παρά την πολυπλοκότητα της μετατροπής των υποκειμενικών μουσικών ιδιοτήτων σε αντικειμενικές μετρήσεις, οι ποσοτικές αξιολογήσεις προσφέρουν πολύτιμες πληροφορίες σχετικά με τη συνολική ποιότητα της παραγόμενης μουσικής.

Αυτές οι αξιολογήσεις εμπεριέχουν διάφορες μετρικές, συμπεριλαμβανομένων των συναρτήσεων απωλειών, των σφαλμάτων συμβολισμού και άλλων μετρικών, όπως η παραμονή στην ίδια κλίμακα και η αρμονική συνέπεια. Συγκρίνοντας τις εξόδους του μοντέλου με τις τιμές του προτύπου, μπορούμε να μετρήσουμε ποσοτικά τις αποκλίσεις και να αξιολογήσουμε την ικανότητα του μοντέλου να μιμείται το ύφος και την έκφραση των πρωτότυπων μουσικών συνθέσεων, καθώς και να παράγει ηχητικά ευχάριστο μουσικό κείμενο.

Προτεινόμενος Τρόπος Αξιολόγησης

Για τις ανάγκες αυτής της διπλωματικής εργασίας, αποφασίσαμε να χρησιμοποιήσουμε μια προσαρμοσμένη μετρική αξιολόγησης, προκειμένου να αναπτύξουμε μια μετρική που να αποτυπώνει τόσο την αποτελεσματικότητα της εκπαίδευσης του μοντέλου όσο και την τελική του απόδοση. Στόχος μας είναι η εξαγωγή ποσοτικών μέτρων που θα μας επιτρέψουν να αξιολογήσουμε την ποιότητα της εκπαίδευσης και της προεπεξεργασίας, καθώς και τα τελικά αποτελέσματα του μοντέλου.

Χρησιμοποιώντας την βιβλιοθήκη Muspy, εξάγουμε από τα παραγόμενα αποτελέσματα και τα αρχικά δεδομένα τις παρακάτω μετρικές:

- Εντροπία νότας: Συχνότητα εμφάνισης μιας ομάδας νοτών στο μουσικό κείμενο
- Εντροπία συχνότητας: Συχνότητα εμφάνισης μιας νότας στο μουσικό κείμενο
- Συνέπεια κλίμακας: Το μεγαλύτερο ποσοστό νοτών σε όλες τις μείζονες και ελάσσονες κλίμακας.

Συνδυάζουμε τις παραπάνω μετρικές ως προς την ευκλείδια απόστασή τους στο παρακάτω τύπο:

$$\text{distance} = \sqrt{(\text{or_PCE} - \text{pr_PCE})^2 + (\text{or_PE} - \text{pr_PE})^2 + (\text{or_SC} - \text{pr_SC})^2}$$

1.2 Πειράματικό Μέρος

1.2.1 Σύνολο Δεδομένων

Στην παρούσα διπλωματική, και για την εκπαίδευση του μοντέλου παραγωγής μουσικής, θα χρησιμοποιήσουμε το σύνολο δεδομένων χορικών συνθέσεων του Johann Sebastian Bach, κοινώς γνωστό ως "JSB Chorales".

Αυτό το σύνολο δεδομένων αποτελείται από αρμονικά πλούσιες χορωδίες του Μπαχ, γεγονός που το καθιστά ιδανικό θεμέλιο για τη μελέτη συστημάτων παραγωγικής μουσικής λόγω των περίπλοκων μελωδιών και της καλά καθορισμένης μουσικής δομής του.

Τα χορικά του Μπαχ αποτελούν παράδειγμα της μαεστρίας του στην αρμονία και την αντίστιξη, χρησιμεύοντας ως ανεκτίμητα διδακτικά εργαλεία στην εκπαίδευση της θεωρίας της μουσικής και προσφέροντας ένα γόνιμο έδαφος για τη μελέτη της μουσικής δομής και των τεχνικών σύνθεσης.

Με μια συλλογή από 372 τετράφωνες χορωδίες, καθεμία από τις οποίες περιέχει μελωδικές γραμμές για φωνές σοπράνο, άλτο, tenόρο και μπάσο, αυτό το σύνολο δεδομένων παρέχει άφθονο υλικό για μουσικά πειράματα.

Παρακάτω περιγράφονται τα βήματα προεπεξεργασίας, συμπεριλαμβανομένου του καθαρισμού και της κανονικοποίησης των δεδομένων, για να διασφαλιστεί η συνοχή και η ακρίβεια των μουσικών δεδομένων, όπως περιγράφεται λεπτομερώς στην επόμενη ενότητα.

1.2.2 Προεπεξεργασία Δεδομένων

Ξεκινήσαμε επεξεργάζοντας το σύνολο δεδομένων των χορικών JSB και απομονώνοντας τις 372 τετράφωνες χορωδίες, από τις 514 χορωδίες του συνόλου των δεδομένων. Στη συνέχεια, χωρίσαμε τα 372 αρχεία MIDI σε ξεχωριστά σύνολα εκπαίδευσης, επικύρωσης και δοκιμής, διασφαλίζοντας ότι δεν θα υπάρξει διαφροή πληροφοριών κατά τη διαδικασία εκπαίδευσης του μοντέλου.

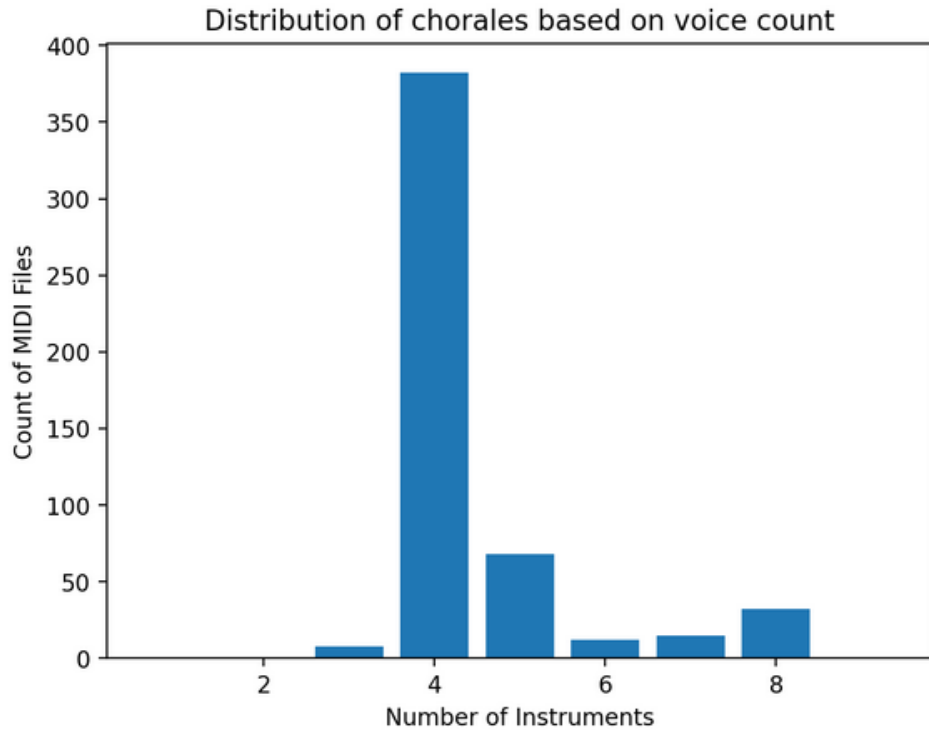


Figure 1.2.1: Κατανομή των χορικών του Bach με βάση το πλήθος φωνών.

Στα επόμενα βήματα προεπεξεργασίας, θα δημιουργήσουμε "αρχεία-παιδιά" από τα αρχικά αρχεία, καθένα από τα οποία θα περιέχει έναν συγκεκριμένο τύπο εργασίας. Πρέπει να διασφαλίσουμε ότι κανένα "αρχείο-παιδί" δεν διαχωρίζεται από το αντίστοιχο "γονέα" του.

Στην προσέγγισή μας για την προετοιμασία αρχείων MIDI για την εκπαίδευση ενός μοντέλου κειμένου-σε-κείμενο όπως ο μετασχηματιστής T5, ξεκινήσαμε με πρωτότυπα αρχεία MIDI που περιείχαν τις τέσσερις διαφορετικές μουσικές φωνές.

Στόχος μας ήταν να συγχωνεύσουμε γειτονικά κομμάτια με βάση συγκεκριμένες απαιτήσεις της εργασίας, με αποτέλεσμα να προκύψουν αρχεία MIDI με μόνο δύο κομμάτια που να ευνοούν την αποτελεσματική κωδικοποίηση σε ακολουθίες εισόδου/εξόδου.

Για τον εξορθολογισμό της διαδικασίας συμβολοποίησης και της υλοποίησης εργασιών όπως "Προσθήκη ανώτερης αρμονικής ακολουθίας" και "Προσθήκη κατώτερης αρμονικής ακολουθίας", δομήσαμε τη μορφή εισόδου-εξόδου απευθείας μέσα στα αρχεία MIDI.

Αυτό περιελάμβανε τη δημιουργία 12 τροποποιήσεων για κάθε μουσικό αρχείο, ισομερώς κατανομημένων μεταξύ των δύο εργασιών.

Figure 1.2.2 consists of two musical score images. Image (a) shows a four-voice Bach chorale with a tempo of J=92. It features four staves labeled 'Piano, Instrument 1' through 'Piano, Instrument 4'. The music is in G major and 4/4 time. Image (b) shows a modified file with a tempo of J=96. It features two staves: 'Piano, Instrument 0' and 'Piano, Instrument 1'. The music is in G major and 4/4 time, with the soprano voice isolated.

(a) Original overview of a four-voice Bach chorale.

(b) Modified file, with isolated soprano voice.

Figure 1.2.2: Οπτική αναπαράσταση των τροποποιήσεων που υπέστησαν τα αρχεία MIDI κατά τη διάρκεια του μέρους της προεπεξεργασίας των δεδομένων που περιγράφεται στην ενότητα 7.2.1.

Συμβολοποίηση

Μετά τις προηγούμενες τροποποιήσεις, προχωράμε στη διαδικασία συμβολοποίησης των τροποποιημένων αρχείων MIDI.

Χρησιμοποιώντας τους συμβολοποιητές και τις ρουτίνες της βιβλιοθήκης Midityok, που περιγράφονται παραπάνω, μετατρέπουμε αυτές τις μουσικές συνθέσεις σε δομημένες ακολουθίες συμβόλων.

Για τις ανάγκες των αρχικών πειραμάτων της ιδέας μας, κάνουμε χρήση του συμβολοποιητή TSD.

Οι παράμετροι του συμβολοποιητή ορίζουν διάφορες πτυχές της διαδικασίας συμβολοποίησης για αρχεία MIDI. Περιλαμβάνουν προδιαγραφές όπως το εύρος του τόνου που λαμβάνεται υπόψη, την ανάλυση του ρυθμού, τον αριθμό των δειγμάτων ταχύτητας και τη συμπερίληψη ειδικών συμβόλων για το γέμισμα, την αρχή και το τέλος της ακολουθίας.

Το Midityok παρέχει μια ποικιλία παραμέτρων διαμόρφωσης του συμβολοποιητή, οι οποίες ελέγχουν την ενσωμάτωση μουσικών στοιχείων, όπως συγχορδίες, παύσεις, τέμπο, χρονικές υπογραφές, αλλαγές μουσικού οργάνου (program) και κάμψεις τόνων. Ορισμένες από αυτές τις παραμέτρους, όπως το εύρος του τόνου και ταχύτητας, αναφέρονται στην υποδειγματοληψία των πληροφοριών που περιέχονται στα γεγονότα νοτών, επιτρέποντας την καταγραφή τους από λιγότερα tokens.

Ο tokenizer μπορεί να προσαρμοστεί για να χειρίζεται αρχεία MIDI πολλαπλών φωνών και προσφέρει επιλογές για την οργάνωση των ροών token, είτε ως ενιαία ροή είτε ως ξεχωριστές ροές για κάθε φωνή. Κάνουμε χρήση αυτού του πλεονεκτήματος για να εξάγουμε τη μορφή εισόδου/εξόδου των ακολουθιών με tokenized που περιγράφηκαν προηγουμένως.

Προθέματα

Για να επιτύχουμε την εξειδίκευση της εργασίας κατά τη διάρκεια της εκπαίδευσης, εισαγάγαμε στον συμβολοποιητή δύο έξτρα σύμβολα τα οποία θα λειτουργήσουν σαν *προθέματα*: το πρόθεμα "Add Upper" και το πρόθεμα "Add Lower". Αυτά, ως σύμβολα, υποδηλώνουν την επαύξηση των ανώτερων και των κατώτερων αρμονικών φωνών, αντίστοιχα. Αυτή η στρατηγική προσθήκη, που χρησιμοποιείται ως *δείκτης*, ο οποίος καθοδηγεί τον μετασχηματιστή T5, διευκολύνοντας την κατανόηση και την εκτέλεσή του στις συγκεκριμένες εργασίες παραγωγής μουσικής που περιγράφονται στο πειραματικό μας πλαίσιο.

Επαύξηση Δεδομένων

Ο συμβολοποιητής συνεχίζει με ένα βήμα επαύξησης δεδομένων για περαιτέρω εμπλουτισμό του συνόλου δεδομένων. Αυτή η επαύξηση περιλαμβάνει τη χειραγώγηση των συμβόλων τόνου, ταχύτητας και διάρκειας για την εισαγωγή παραλλαγών στις μουσικές ακολουθίες. Συγκεκριμένα, η αύξηση του τόνου εφαρμόζεται με μετατόπιση του τόνου κατά 2 οκτάβες προς τα πάνω και προς τα κάτω. Η αύξηση της ταχύτητας και της διάρκειας εφαρμόζεται με μετατόπιση 1 διαστήματος η καθεμία. Με τη συστηματική προσαρμογή αυτών των παραμέτρων, ο συμβολοποιητής ενισχύει την ποικιλομορφία του συνόλου δεδομένων, επιτρέποντας στο μοντέλο να μαθαίνει και να γενικεύει καλύτερα σε διαφορετικά μουσικά πλαίσια.

Χρήση κωδικοποίησης ζεύγους

Επιπλέον, η χρήση κωδικοποίησης ζεύγους byte (BPE) ενισχύει περαιτέρω την αποδοτικότητα της μεθόδου κωδικοποίησης, εξασφαλίζοντας βέλτιστες επιδόσεις σε όλη τη φάση της εκπαίδευσης και της δημιουργίας [8].

Η κωδικοποίηση ζεύγους byte πρέπει να μαθητεύει στο σύνολο των δεδομένων (σύνολα εκπαίδευσης, επικύρωσης και δοκιμής, αντίστοιχα). Αυτό διασφαλίζει τη συνέπεια και τη συνοχή της κωδικοποίησης σε όλα τα σύνολα δεδομένων, διευκολύνοντας την ομοιόμορφη αναπαράσταση των μουσικών ακολουθιών για το μοντέλο. Δημιουργούμε ένα νέο, επαυξημένο λεξιλόγιο, το οποίο αποτελείται από τις πιο συχνές υποακολουθίες.

Αργότερα, τα σύνολα δεδομένων εκπαίδευσης, επικύρωσης και δοκιμής κωδικοποιούνται με το νέο λεξιλόγιο BPE και, έτσι, συμπίεζονται σε μικρότερο μέγεθος που αναπαριστά αποτελεσματικά τις πληροφορίες που απαιτούνται για την αποτελεσματική εκπαίδευση του μοντέλου.

1.2.3 Πειράματα

Επιλογή Παραμέτρων

Στα αρχικά μας πειράματα διερευνήσαμε πως επηρεάζουν το μοντέλο οι παράμετροι του μεγέθους του λεξιλογίου με το οποίο γίνεται η κωδικοποίηση ζεύγους και ποσοστό εγκατάλειψης του μοντέλου. Όσον αφορά το μέγεθος του μοντέλου, επιλέξαμε τις παραμέτρους που χρησιμοποιούν στις δημοσιεύσεις της βιβλιογραφίας [8]

Table 1.1: Παράμετροι για τα αρχικά πειράματα

Tokenizer	Vocab Size	Num Layers	Num Attention Heads	Dropout Rate
TSD	2500	8	16	0.2
	5000			0.4

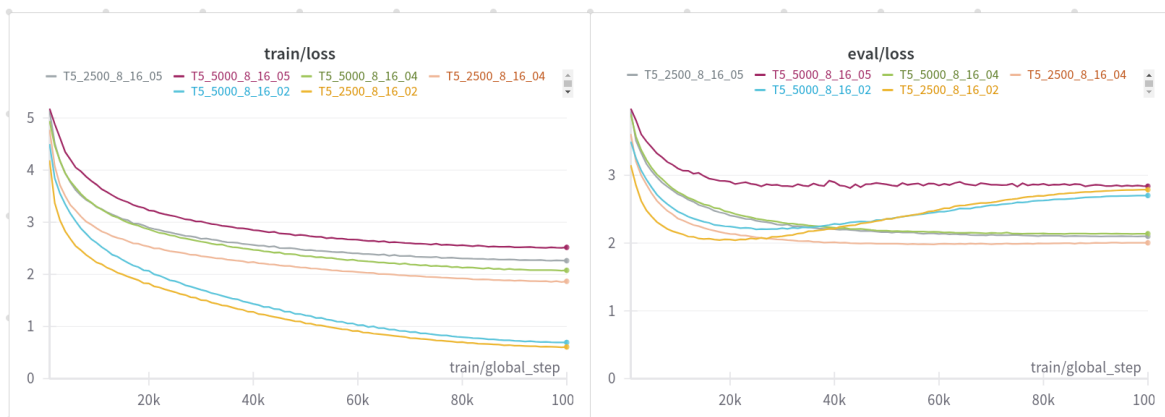


Figure 1.2.3: Συναρτήσεις απωλειών των πρώτων πειραμάτων 1.1. Αρχικά εκτελούμε πειράματα με παραμέτρους του μοντέλου σύμφωνα με τη βιβλιογραφία [8]. Επιλέξαμε ένα αρκετά μεγάλο εύρος τιμών, προκειμένου να διερευνήσουμε την κατεύθυνση περαιτέρω πειραματισμού.

Η ανάλυση των αποτελεσμάτων, που απεικονίζονται στο 1.2.3, αποκάλυψε αξιοσημείωτες τάσεις. Τα μοντέλα που εκπαιδεύτηκαν με ποσοστό εγκατάλειψης 0,2 παρουσίασαν ταχεία υπερπροσαρμογή στα δεδομένα μας, υποδεικνύοντας ανεπαρκή κανονικοποίηση. Αντίθετα, τα μοντέλα με ποσοστό διακοπής 0,5 παρουσίασαν σημάδια υποπροσαρμογής, υποδηλώνοντας υπερβολική κανονικοποίηση. Κατά συνέπεια, καθορίσαμε ότι το βέλτιστο ποσοστό εγκατάλειψης βρίσκεται γύρω στο 0,4, επιτυγχάνοντας μια ισορροπία μεταξύ υπερπροσαρμογής και υποπροσαρμογής.

Αναφορικά με το μέγεθος του λεξιλογίου, οι παρατηρήσεις έδειξαν ότι τα μοντέλα με μικρότερο μέγεθος λεξιλογίου (2500) πέτυχαν χαμηλότερες τιμές απωλειών σε σύγκριση με εκείνα με μεγαλύτερο μέγεθος (5000). Το φαινόμενο αυτό είναι δικαιολογημένο, καθώς ένα μεγαλύτερο λεξιλόγιο σε ένα σχετικά μικρό σύνολο δεδομένων μπορεί να οδηγήσει στην απόκρυψη πολύτιμων πληροφοριών πίσω από κωδικοποιήσεις ζευγών. Κατά συνέπεια, το μοντέλο μπορεί να δυσκολεύεται να συλλάβει αποτελεσματικά τα υποκείμενα μοτίβα των δεδομένων.

Με τη θεμελίωση της πειραματικής μας διαδικασίας, περιορίζουμε τώρα το πεδίο των τιμών των παραμέτρων για να εμβαθύνουμε περισσότερο στην έρευνά μας. Σε αυτή τη φάση, στοχεύουμε να αξιολογήσουμε τον τρόπο με τον οποίο το μοντέλο συλλαμβάνει πληροφορίες υπό διαφορετικές συνθήκες, εστιάζοντας σε συγκεκριμένες διαμορφώσεις παραμέτρων.

Για τους παρακάτω πειραματισμούς, ερευνούμε την απόδοση των μοντέλων για τις παραμέτρους που αναφέρονται στο 1.2.

Table 1.2: Περεταίρω πειραματισμοί με στόχο την εύρεση βέλτιστων παραμέτρων μοντέλου

Tokenizer	Vocabulary Size	Model Layers	Model Attention Heads	Dropout Rate
TSD	1500	8	12	0.3
	2500	10	16	0.4

Μεταβάλλοντας αυτές τις παραμέτρους, επιδιώκουμε να αποκτήσουμε γνώσεις σχετικά με το πώς διαφορετικές αρχιτεκτονικές διαμορφώσεις επηρεάζουν την απόδοση του μοντέλου στην καταγραφή και κατανόηση των υποκείμενων προτύπων των δεδομένων.

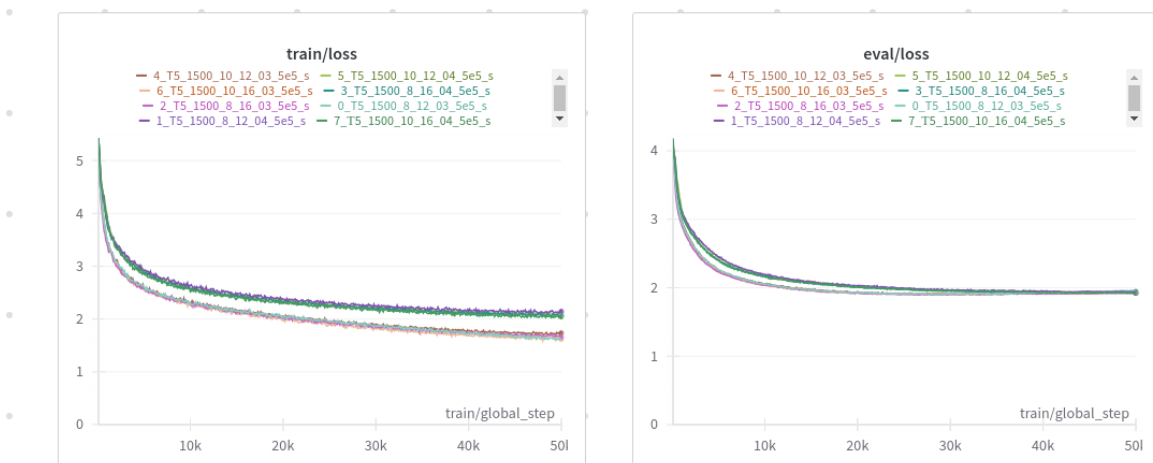


Figure 1.2.4: Καμπύλη απωλειών των μοντέλων που εκπαιδεύτηκαν με TSD συμβολοποιητή και μέγεθος λεξιλογίου = 1500. Οι υπόλοιπες παράμετροι περιγράφονται στο 1.2

Μετά την εκτέλεση των πειραμάτων με τις προαναφερθείσες παραμέτρους, παρατηρούμε ότι, όσον αφορά τις απώλειες, τα μοντέλα φαίνεται να καταλήγουν σε παρόμοια αποτελέσματα. Τα μοντέλα με χαμηλότερο ποσοστό εγκατάλειψης φαίνεται να πιάνουν ευκολότερα τις πληροφορίες και τα μοτίβα στην επιθυμητή έξοδο, εμφανίζοντας πιο απότομες καμπύλες.

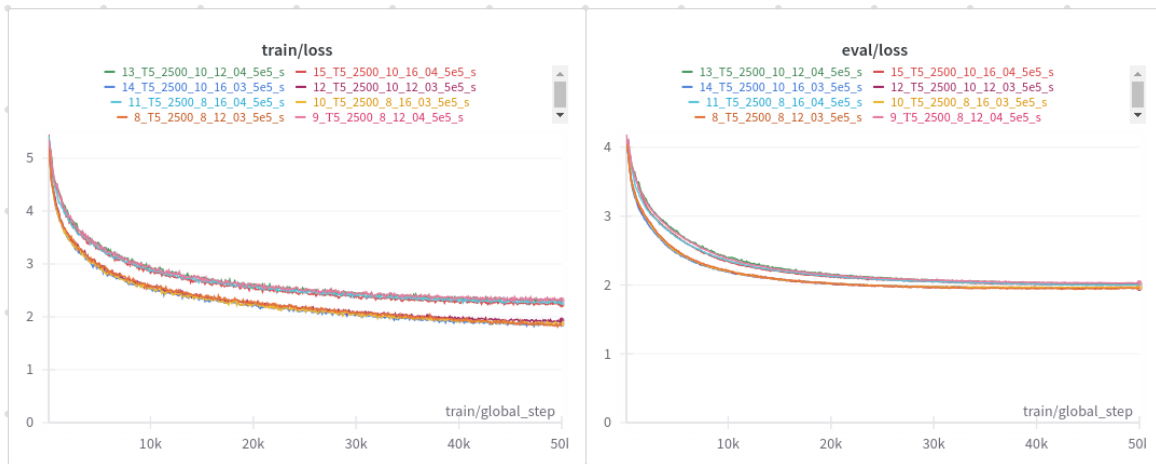


Figure 1.2.5: Καμπύλη απωλειών των μοντέλων που εκπαιδεύτηκαν με TSD συμβολοποιητή και μέγεθος λεξιλογίου = 2500. Οι υπόλοιπες παράμετροι περιγράφονται στο 1.2

Παραγωγή

Για να περάσουμε στη δοκιμή των παραπάνω μοντέλων, αποφασίσαμε να παράγουμε έξοδο από το μοντέλο εφαρμόζοντας θερμοκρασία στο τελικό αποτέλεσμα. Σκοπός της θερμοκρασίας είναι να παράγεται πρόταση στην έξοδο, η οποία δεν ακολουθεί την συνάρτηση softmax, επιλέγοντας το σύμβολο με τη μεγαλύτερη πιθανότητα εμφάνισης. Αυτή η προσθήκη παρέχει μεγαλύτερη ευελιξία και δημιουργικότητα στη διαδικασία παραγωγής μουσικής, επιτρέποντας στο μοντέλο να επιδείξει μεγαλύτερη δημιουργικότητα και να αποφύγει τον βρόχο "πιο πιθανό επόμενο σύμβολο".

Εφαρμόζοντας την προτεινόμενη μέθοδο αξιολόγησης

Για να αποκτήσουμε μια συνολική επισκόπηση της απόδοσης των μοντέλων μας, θα χρησιμοποιήσουμε τη μετρική που ορίστηκε ωρίτερα στο 1.1.5.

Για να αναλύσουμε διεξοδικά το συνολικό αποτέλεσμα συγκρίνοντας την ευθυγράμμιση της εισαγόμενης φωνής με τη δεδομένη μελωδία και αξιολογώντας συγκεκριμένες πτυχές, όπως ο πλούτος, η τήρηση της κλίμακας και η αυτόνομη απόδοση. Αυτή η δομημένη προσέγγιση επιτρέπει την ενδελεχή εξέταση τόσο των ολιστικών όσο και των επιμέρους χαρακτηριστικών της παραγόμενης φωνής.

Για να οπτικοποιήσουμε αυτά τα ευρήματα, δημιουργούμε ραβδογράμματα που απεικονίζουν την κατανομή των αποστάσεων από τα δείγματα του συνόλου δοκιμών που παράγονται από κάθε μοντέλο 1.2.6. Αυτή η προσέγγιση επιτρέπει την εύκολη σύγκριση μεταξύ διαφορετικών μοντέλων 1.2.7 1.2.8.

Όπως επιβεβαιώνεται και από τις καμπύλες απωλειών, τα μοντέλα τείνουν να συγκλίνουν και παρουσιάζουν παρόμοια αποτελέσματα. Ωστόσο, οι διαφορές γίνονται πιο εμφανείς κατά την αξιολόγηση της ποιότητας της παραγόμενης φωνής σε σύγκριση με την δοσμένη. Σε αυτό, οι διαφορές είναι πιο έντονες, κάτι που είναι αναμενόμενο, καθώς το πρώτο συγκρίνει αρχεία που έχουν μέρη κοινού πλαισίου (οι δεδομένες φωνές).

Ομοίως, αξιολογώντας τα αποτελέσματα της πρόσθετης παραγωγής φωνής, αυτή τη φορά με μεταβαλλόμενη θερμοκρασία, παρατηρούμε ότι οι μέσες τιμές των αποστάσεων δεν αλλάζουν, οι κατανομές των αποστάσεων έχουν παρόμοια μορφή και επομένως καταλήγουμε στο συμπέρασμα ότι η μεταβολή της θερμοκρασίας κατά τη διάρκεια της παραγωγής δεν μεταβάλλει σημαντικά τη συνολική αξιολόγηση. Αυτό υποδηλώνει ότι, εντός του επιλεγμένου εύρους, η παράμετρος της θερμοκρασίας μπορεί να έχει περιορισμένο αντίκτυπο στην ποιότητα των παραγόμενων φωνών. Τα ευρήματά μας δείχνουν επίσης ότι μια τιμή θερμοκρασίας 0,2 παράγει σταθερά ικανοποιητικά αποτελέσματα.

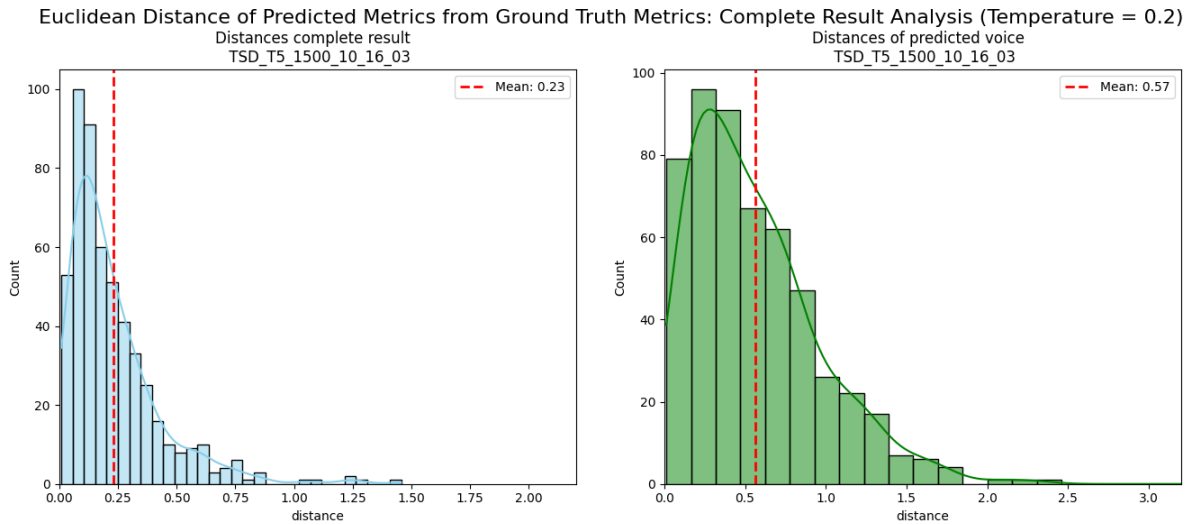


Figure 1.2.6: Σύγκριση των κατανομών απόστασης μεταξύ των μετρικών που μετρήθηκαν στις παραγόμενες εκροές και της βασικής αλήθειας σε ολόκληρο το μουσικό κομμάτι (αριστερά) και ειδικά στη παραγόμενη φωνή (δεξιά).

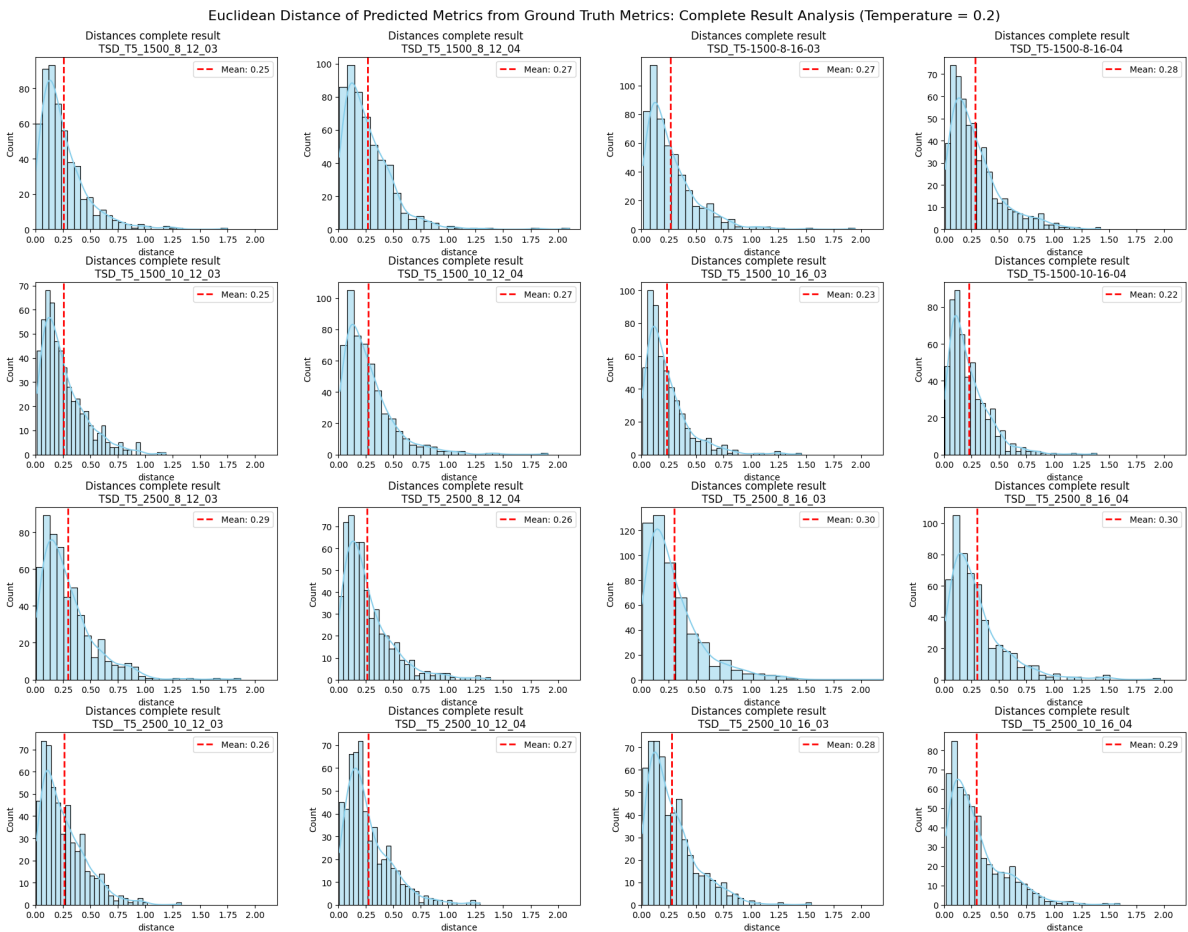


Figure 1.2.7: Ανάλυση 16 μοντέλων, με χρήση του TSD tokenizer και των παραμέτρων που περιγράφονται στο 1.2: Κατανομή των μετρικών αποστάσεων μεταξύ των παραγόμενων εξόδων και της βασικής αλήθειας, που υπολογίστηκαν στο σύνολο των μουσικών κομματιών του συνόλου δοκιμών, με ρύθμιση της θερμοκρασίας στο 0,2

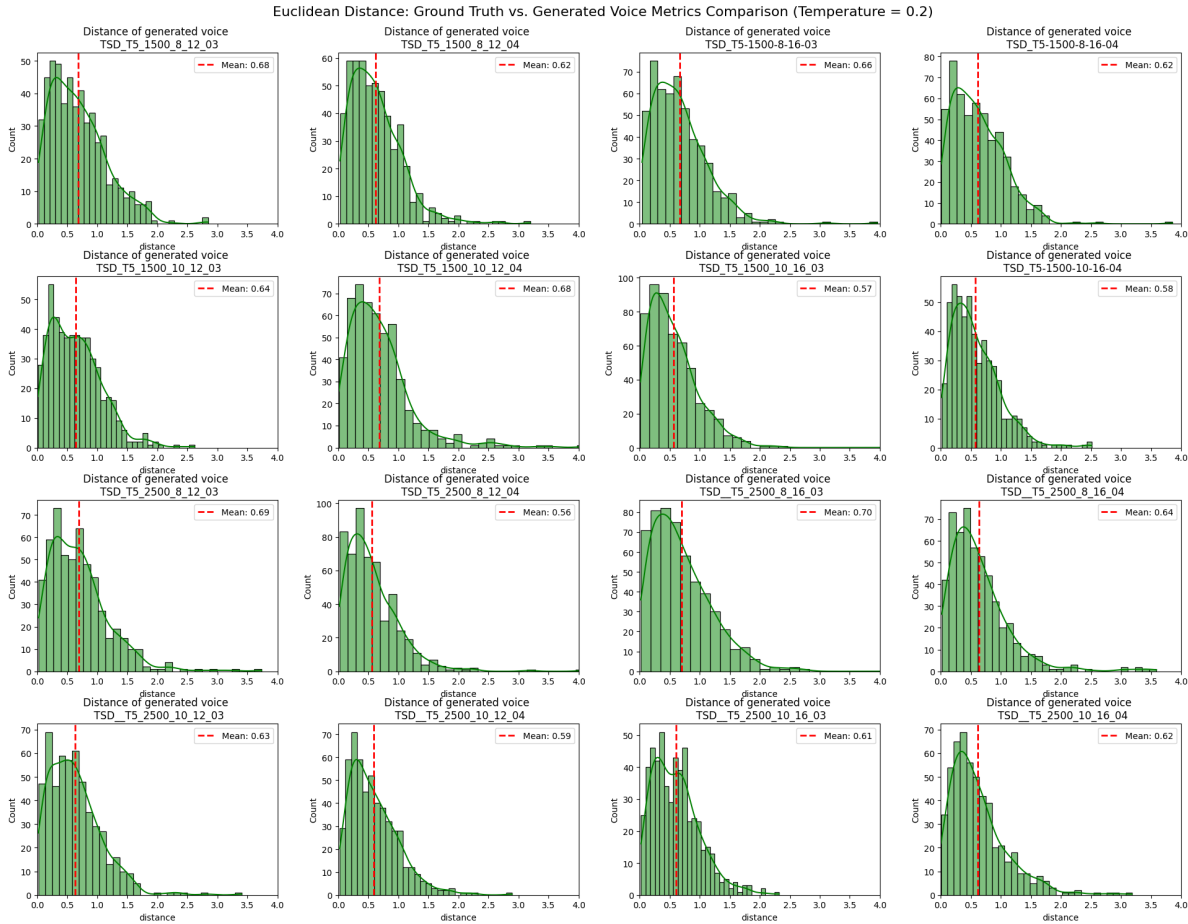


Figure 1.2.8: Analysis of 16 models using TSD tokenizer and parameters described on 1.2: Distribution of metric distances between generated outputs and ground truth, calculated specifically on the generated voice in the test set, with temperature settings ranging from 0.05 to 0.35.

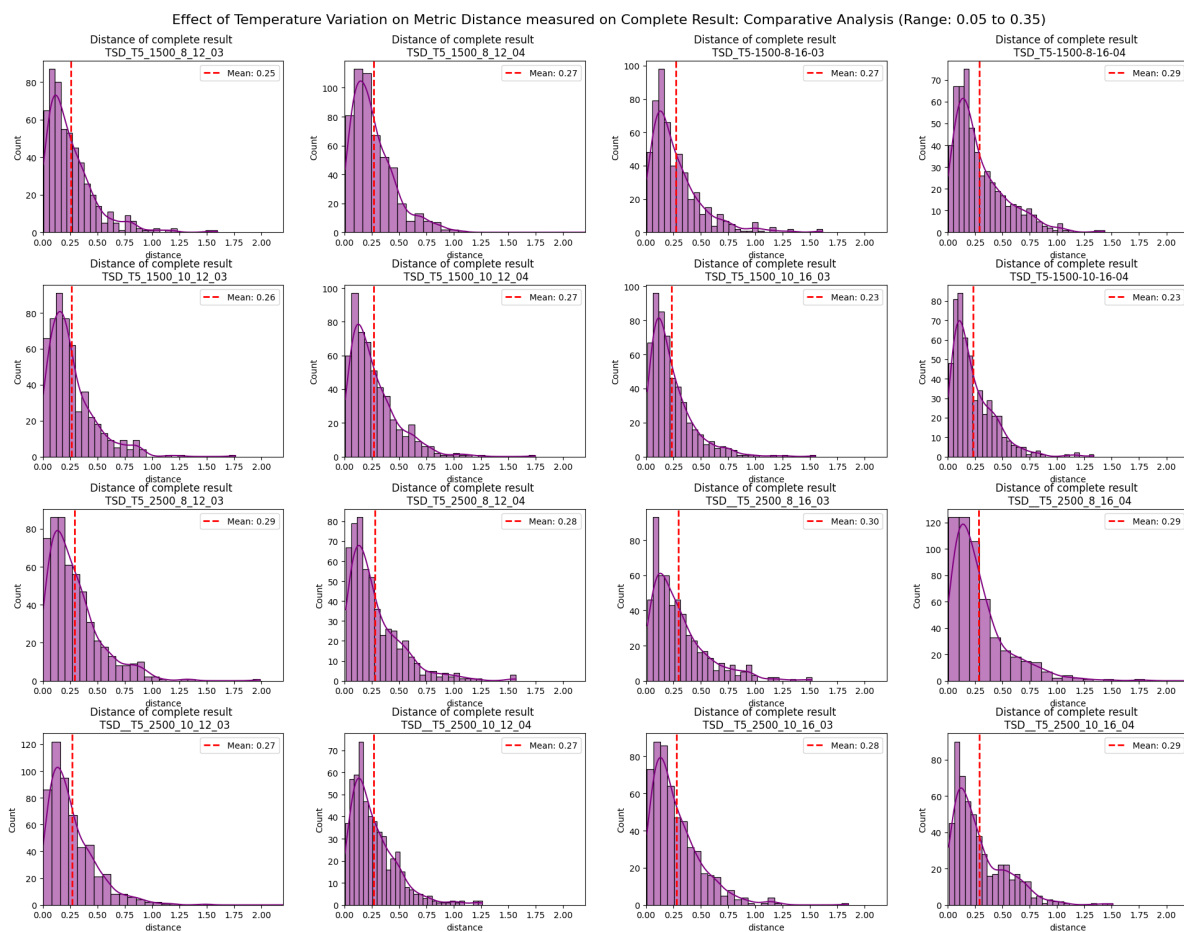


Figure 1.2.9: Ανάλυση 16 μοντέλων με τη χρήση TSD συμβολοποιητή και παραμέτρων που περιγράφονται στο 1.2: Κατανομή των μετρικών αποστάσεων μεταξύ των παραγόμενων εξόδων και του δοσμένου που υπολογίστηκαν στο σύνολο των μουσικών κομματιών του συνόλου δοκιμών, με ρύθμιση της θερμοκρασίας στο 0,2.

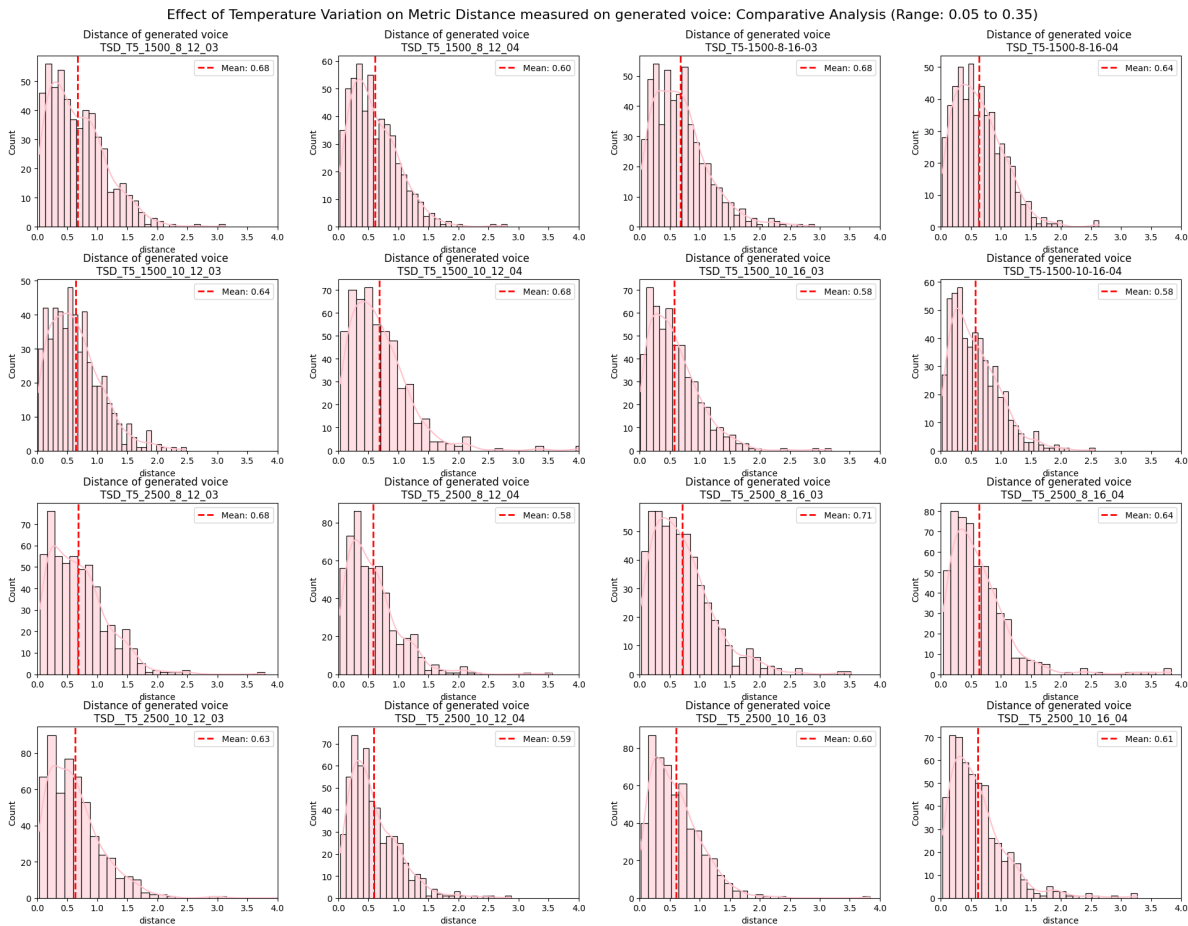


Figure 1.2.10: Ανάλυση 16 μοντέλων με τη χρήση TSD συμβολοποιητή και παραμέτρων που περιγράφονται στο 1.2: Κατανομή των μετρικών αποστάσεων μεταξύ των παραγόμενων εξόδων και του δοσμένου, που υπολογίστηκαν ειδικά για την παραγόμενη φωνή στο σύνολο δοκιμών, με ρυθμίσεις θερμοκρασίας που κυμαίνονται από 0,05 έως 0,35

Περαιτέρω πειραματισμοί: REMI & Structured Συμβολοποιητές Λόγω των υπολογιστικών περιορισμών και των περιορισμών στην ισχύ της GPU, επεκτείναμε το πεδίο εφαρμογής των πειραμάτων μας για να διερευνήσουμε μοντέλα με λιγότερες παραμέτρους, μικρότερα μεγέθη και διαφορετικούς tokenizers. Υλοποιώντας πολυάριθμα πειράματα, χρησιμοποιήσαμε ένα μοντέλο με 4 στρώματα, 12 κεφαλές προσοχής, ποσοστό διακοπής 0,3 και διάφορους tokenizers, καθένα με μέγεθος λεξιλογίου 1000.

Comparing evaluation metrics of two small models trained with REMI tokenizer and custom augmentation

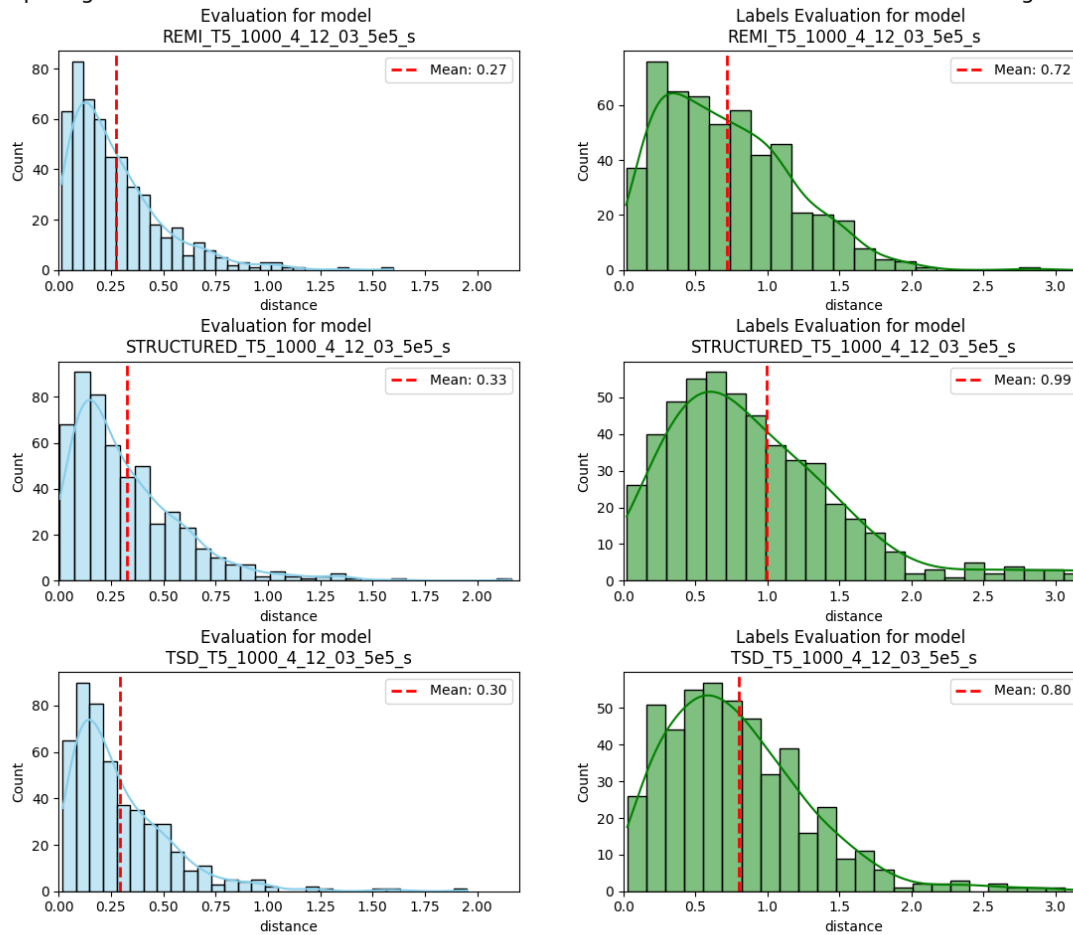


Figure 1.2.11: Μικρά μοντέλα που εκπαιδεύονται σε δεδομένα που έχουν επισημειωθεί με τους συμβολοποιητές REMI, Structured & TSD: Σύγκριση των κατανομών των αποστάσεων μεταξύ των μετρικών που μετρώνται στις παραγόμενες εξόδους και της βασικής αλήθειας σε ολόκληρο το μουσικό κομμάτι (αριστερά) και συγκεκριμένα στην παραγόμενη φωνή (δεξιά)

Παρατηρήσαμε ότι ο TSD και ο REMI φαίνεται να ξεπερνούν τον Structured όσον αφορά τις μετρικές μας. Ωστόσο, για να διεξάγουμε πειράματα που απεικονίζουν ολοκληρωμένα τα πλεονεκτήματα και τα μειονεκτήματα του καθενός, πρέπει να αναλάβουμε μια πιο εξαντλητική διαδικασία πειραματισμού, επιπλέον, η διασφάλιση ότι οι πόροι μας είναι επαρκείς για να χειριστούμε αυτόν τον διευρυμένο πειραματισμό είναι επιτακτική ανάγκη.

1.3 Συμπεράσματα

Συνοψίζοντας, η παρούσα διατριβή ασχολήθηκε με τον εμπλουτισμό των πολυφωνικών χορωδιών του Μπαχ μέσω ενός μοντέλου παραγωγής μουσικής βασισμένου σε μετασχηματιστή Seq2Seq. Η προσεκτική προεπεξεργασία, η οποία περιλάμβανε τη συμβολαιοποίηση, την επαύξηση δεδομένων και την τυποποίηση, υπογράμμισε τον κρίσιμο ρόλο του χειρισμού των δεδομένων σε αυτό το έργο. Ο μετασχηματισμός της συμβολικής μουσικής σε διαδοχική μορφή αναδείχθηκε ως ένα δυνητικά ξεχωριστό πεδίο μελέτης.

Το πειραματικό μας ταξίδι διερεύνησε διάφορες διαμορφώσεις μοντέλων, αποκαλύπτοντας γνώσεις σχετικά με τις επιπτώσεις των παραμέτρων στην απόδοση, συμπεριλαμβανομένης της σύγκλισης, της υπερπροσαρμογής και της πιστότητας της φωνής. Οι συγκριτικές αναλύσεις παρείχαν αποχρώσεις της συμπεριφοράς του μοντέλου, διευκρινίζοντας τις αντισταθμίσεις και τις βέλτιστες διαμορφώσεις. Προσοχή δόθηκε στην εξέταση των συμβολικών μουσικών αναπαραστάσεων και των τεχνικών αξιολόγησης, συμπεριλαμβανομένων πλαισίων για ποσοτική και ποιοτική αξιολόγηση.

Προτείνουμε μια καινοτόμο προσέγγιση αξιολόγησης που ευθυγραμμίζεται με την ανθρώπινη αντίληψη, ρίχνοντας φως στις διαφορές των μοντέλων και αναδεικνύοντας τη σημασία της επιλογής της θερμοκρασίας.

Συνολικά, η παρούσα διατριβή συμβάλλει στην προώθηση της παραγωγής μουσικής με βάση την τεχνητή νοημοσύνη, αναδεικνύοντας τις δυνατότητες για μετασχηματιστική καινοτομία σε αυτόν τον διεπιστημονικό τομέα.

1.4 Συζήτηση

Η παρούσα έρευνα παρουσιάζει πολύτιμες γνώσεις σχετικά με την παραγωγή μουσικής και τα μοντέλα που βασίζονται σε μετασχηματιστές, ανοίγοντας το δρόμο για μελλοντική εξερεύνηση και καινοτομία.

Προχωρώντας, σκοπεύουμε να αξιοποιήσουμε τις τεχνικές προεκπαίδευσης με μάσκα για να βελτιώσουμε την απόδοση και την προσαρμοστικότητα του μοντέλου. Επιπλέον, σκοπεύουμε να διερευνήσουμε πρόσθετες αρχιτεκτονικές παραλλαγές και να επεκτείνουμε την έρευνά μας σε διάφορους μουσικούς τομείς.

Επιπλέον, η μελλοντική μας δουλειά θα επικεντρωθεί στην ανάπτυξη μιας προσαρμοσμένης συνάρτησης απωλειών που θα ενημερώνεται από τις μεθόδους αξιολόγησης που περιγράφονται σε προηγούμενες μελέτες, με στόχο τη βελτίωση της προσκόλλησης του μοντέλου στις αρμονικές αρχές. Η ενσωμάτωση εφαρμογών επεξηγηματικότητας θα εμβαθύνει την κατανόηση της διαδικασίας λήψης αποφάσεων του μοντέλου, επιτρέποντας τον πειραματισμό με αντιπαραθετικές επεξηγήσεις και την παραγωγή υπό όρους.

Η διερεύνηση διαφορετικών μεθόδων και μοντέλων συμβολικοποίησης, συμπεριλαμβανομένων των μεγάλων γλωσσικών μοντέλων και των νευρωνικών δικτύων γράφων, θα προσφέρει ευκαιρίες για τη βελτίωση της συμβολικής αναπαράστασης και επεξεργασίας της μουσικής. Η κατανόηση των περιπλοκών της διαδικασίας συμβολικοποίησης θα επιτρέψει την πρόοδο τόσο στις συμβολικές όσο και στις ηχητικές αναπαραστάσεις μουσικής, προωθώντας διεπιστημονικές συνεργασίες στη δημιουργία και σύνθεση μουσικής.

Τέλος, σκοπεύουμε να διερευνήσουμε ποικίλες μεθόδους αξιολόγησης για την ολοκληρωμένη αξιολόγηση της ποιότητας, της συνοχής και της εκφραστικότητας των παραγόμενων μουσικών αποτελεσμάτων, προωθώντας τελικά τις δυνατότητες των συστημάτων παραγωγής μουσικής με βάση την τεχνητή νοημοσύνη.

Chapter 2

Introduction

The realm of music generation has long been a captivating domain, blending the intricacies of artistry and technology to create harmonious compositions that resonate with audiences on emotional, intellectual, and aesthetic levels. Within this landscape, the pursuit of polyphonic music generation stands as a pinnacle aspiration, seeking to craft musical pieces that not only possess pleasant harmonic qualities but also harbor a complexity that stimulates the listener across multiple dimensions.

We draw inspiration from the advancements in Natural Language Processing (NLP), particularly in the domain of language generation tasks, we propose a novel approach to polyphonic music generation.

At the heart of our methodology lies a Seq2Seq generation framework, crafted to enrich musical compositions by integrating additional voices into the given melody. By extending the traditional Seq2Seq paradigm to the domain of music, we aim to embed compositions with layers of complexity and depth, elevating the listening experience for both creators and audiences alike.

Central to our approach is the transformation of symbolic music, typically stored in MIDI files, into a sequential format suitable for transformer models. Here, we employ modern encoding tools, as introduced in prior research [9], which include techniques such as byte pair encoding and data augmentation. By incorporating these techniques to music information, we expect to unlock new avenues of expression and innovation, as seen in the field of Natural Language Processing.

While objectively evaluating musical compositions remains an ongoing challenge, our research takes strides towards bridging the gap between quantitative assessments and human perception. Through a novel evaluation framework that implements quantitative metrics with qualitative criteria, we offer a comprehensive understanding of the compositional quality and aesthetic appeal of our generated music.

In the subsequent sections of this paper:

- We will firstly provide all the background needed in basic Machine Learning algorithms and concepts as well as transformer models. After doing so, we will provide a thorough description of crucial music concepts in order to explain our train of thought, regarding the specification of the concept.
- We will break down the appropriate tools that bridge symbolic music representations with transformer models, allowing us to approach music generation tasks through the prism of natural language processing, use state-of-the-art tools and add our own vision to the trajectory of the abilities of the field of generative music. We explore a creative task for polyphonic music enrichment with the use of a T5 transformer, in order to test the limits of machine understanding and creativeness in terms of musical context.
- After developing and testing our model, explore certain aspects of each evaluation category, discuss the evaluation process in detail, and propose a composite metric utilizing metrics from the MusPy library

Chapter 3

Machine Learning

For centuries, building a machine that is capable of replicating human reasoning and perception has been an aspiration of humanity. Artificial intelligence has been envisioned throughout history by multiple civilizations. In the rapid technological evolution of the 20th century, scientists seek to comprehend the mechanism of human recognition systems, interpret it with mathematical principles and translate it into logical sequences. [20]. Thus the field of Machine Learning came into existence.

Machine learning revolves around the concept of creating computational systems that can learn and understand the world. It's a field of study within artificial intelligence that utilizes probabilities, statistics, differential equations in order to create models that are capable of completing certain tasks. These tasks include decision making, predicting, labeling, generating, optimizing and more. In simpler terms, a machine learning model is "trained" on data, aiming to identify patterns and relationships within the input and extract the mathematical curves that define them - both in an overview and in detail. The ultimate goal is to improve the model's performance in time, by adjusting its parameters with each epoch of training, similar to a human person learning from experience.

Throughout the years, the field has evolved from originally aiming to solve mathematically demanding processes, instead of humans, to now aspiring to acquire human perception and mimic human reasoning processes. Machine Learning has evolved to a versatile tool that can be applied across numerous scientific domains, services and individual daily life. The early 21st century has been a time of revolution, as research is being incorporated in a large number of applications, such as image classification, recommendation systems, optimization algorithms, natural language processing and more.

3.1 Learning Categories

In order to provide a clearer understanding of the diverse methods and applications of machine learning research, we will analyze the three fundamental categories of algorithms in this field. In the following section, we will examine the concepts of supervised, unsupervised and reinforcement learning.

Supervised Learning

Supervised Learning in machine learning involves developing knowledge about the data by understanding the relationship between input and corresponding output. The training data, real or synthetic, are presented as a set of paired samples, also referred to as labelled data. The objective of a supervised model is to construct an artificial system capable of mapping the connections between input and output, to estimate the mathematical curves that describe them, enabling the model to make accurate and well-informed decisions.

Supervised learning can be categorized into regression, which involves tasks with continuous output needs like forecasting, and classification, that includes tasks where data points are assigned to predefined classes, such as sentiment analysis or image classification. However, it should be noted that the above distinction is not always straightforward, as supervised learning includes a diverse range of tasks, that cannot be fit into two categories, Sequence-to-sequence tasks, multi-label classification, task that reach to the extend of complex real-world scenarios are considered supervised tasks, without be distinct in the above mentioned categories.

Unsupervised Learning

Opposing supervised learning, unsupervised learning is the type of learning that operates with unlabeled data. In this case, the learning algorithm utilizes the probability distribution of the data points in order to discover relationships, patterns and structures within the dataset, with minimal human interaction. Two of the most popular methods used in supervised learning are principal component analysis for dimensionality reduction and clustering. Anomaly detection, self-organizing maps and word embeddings represent some of the tasks the field engages in.

Reinforcement Learning

Reinforcement Learning is the area of machine learning that is concerned with optimal control in dynamic systems. Within this type of learning, an agent interacts with an environment to take sequential decisions, getting feedback in form of rewards and penalties, in order to optimize its actions and maximize the reward. During training, the agent develops adaptive strategies, applicable in scenarios where dynamic decision making and adaptability is required. Thus, this type of learning is optimal for game playing, autonomous robotics and recommendation systems tasks.

Later in this thesis we will be diving deeper in the fields of supervised and unsupervised learning, focusing on Natural Language Processing tasks and examining in detail how and which techniques are utilized in the state of the art technologies in this field.

3.2 Training a Machine Learning Model

This section is dedicated to explain the fundamental aspects of constructing and training a neural network. In the following pages we will go into detail on the architecture and the parameters of a machine learning model, on the requirements of data the model is trained on as well as the basic concepts of the training and learning process.

3.2.1 Architecture of a Neural Network

A neural network consists of interconnected layers of artificial neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron applies an activation function to the weighted sum of its inputs, producing an output signal. The connections between neurons have associated weights that are adjusted during the training process to optimize the network's performance. This layered structure and the adaptive nature of the connections enable the neural network to process complex information.

The architecture of a neural network is inspired by the biological neural networks in the human brain, aiming to replicate its information processing, learning, and adaptation capabilities. During training, the weights of the connections between neurons are adjusted in order to optimize the network's performance on a specific task, just like the connections in the human brain are strengthened or weakened based on experience and learning. [22]

In supervised learning the goal is to minimize the difference between the desired and the predicted output, while in unsupervised learning, it is to identify patterns within the data without explicit guidance.

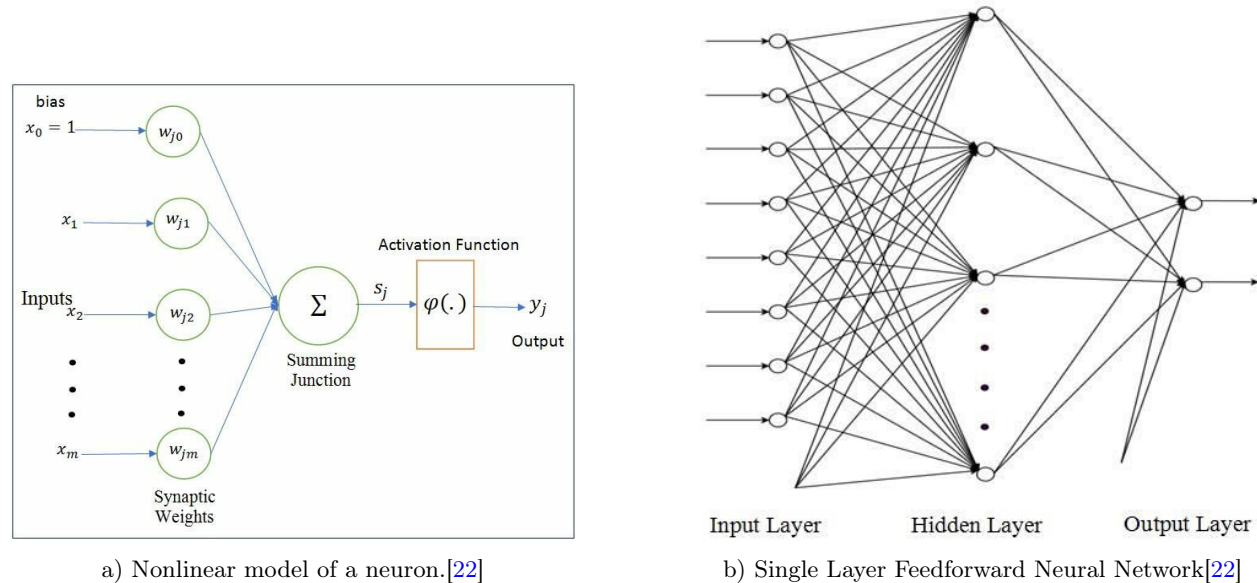


Figure 3.2.1

3.2.2 The role of the architecture of a model

The architecture of a neural network is a key factor to the model's ability to extract information from the data. According to the task, the nature and volume of the available data, and the available computational resources, it is crucial to select a suitable model architecture, considering specific parameters, in order to achieve optimal training cost and performance. This subsection explains how the number of nodes, layers and connections influences the training process of a neural network.

Nodes: this term refers to the number of neurons on a neural network. This number is correlated to the model's complexity: The greater the number of neurons, the greater the model's capacity to retrieve complex information from the input data. However, an excessive number of nodes might not lead to optimal training, as it might lead to overfitting, and also increase the computational cost of the model, both in training and inference.

Layers: The number of layers of a neural network represents the ability of the model to capture hierarchical representations and abstract features. Typically, a neural network has an input layer, one or more hidden layers and an output layer. Shallow networks, e.g. a single layer feedforward NN in Figure 1.2.1b are effective for simple tasks, such as regression. By adding more hidden layers in a neural network, the model will have the **ability to represent increasingly complex features**. The domain focused on complex models trained on diverse data, requiring networks with many layers, is known as **Deep Learning**. Although, like in the case of adding more nodes, adding more layers to a neural network can provide the model with more capacity to memorize the training data, thus lead to an overfit model.

Connections: The number of connections between the nodes of layers of a neural network are correlated with the networks **learning capacity**. The weights of the connections in a neural network determine the emphasis put on different features or details on the training data, influencing the model output. The weights are adjusted during the training process. Too many connections in a neural network may lead the model to

memorize the input data or get biased on specific patterns, thus, regularization techniques, such as dropout or weight decay, are employed, in order to avoid relying on specific connections and to create a model that generalizes well.

The concept of an overfit and underfit model will be discussed in detail later, in section *****

3.2.3 Loss Functions, Backpropagation and Activation Functions

In addition to the architectural and topological parameters mentioned in the previous subsection, the learning and expression path of the neural network output is controlled by the loss and activation functions. To be able to explain the operation of these, let us focus on Figure 1.2.1, which shows a shallow neural network.

The **loss function** is responsible to **guide the learning process** [25]. It calculates the difference between the model's prediction and the desired outcome. During training, the goal is to make the predictions as close to the actual values as possible.

This is done through **backpropagation**, an optimizing algorithm that calculates the gradient of the loss with respect to the parameters and **minimizes the loss function** by adjusting the weights of the neural network.

Neural models undergo multiple iterative rounds of training, known as **epochs**, continuing until the loss function reaches a minimum, or the model reaches the maximum predefined number of iterations.

It is important to select the appropriate loss function and evaluation metrics in relation to the task and the characteristics of the available dataset.

Mean Squared Error(MSE) and its variations are often suitable for regression and estimation tasks, due to their simplicity and interpretability.

$$MSE = \frac{\sum_{i=1}^n (y_i - f(x_i))^2}{n} \quad (3.2.1)$$

Some commonly used variations are Root Mean Squared Error (RMSE), Max-Error , Mean Average Error (MAE).

Binary Cross Entropy Loss, measures the difference between the predicted probability distribution and the actual probability distribution, making them suitable for optimizing classification models. The predicted probability is represented by a vector of predicted probabilities for each class, where the predicted probability of the true class is denoted by $p(y=1/x)$ and the predicted probability of the other class is denoted by $p(y=0/x)$.

$$BCE = L(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \quad (3.2.2)$$

Despite being easy to compute, differentiable and providing a probabilistic interpretation of the model's output, Binary Cross Entropy Loss is sensitive to class imbalances. Some robust alternatives for situations that involve an imbalanced dataset are also weighted binary cross entropy loss functions.

Beyond the loss function, **evaluation metrics** are an index of the model's performance in classification tasks. These metrics are based on calculations on the correct and incorrect predictions of a model. The following confusion matrix contains the parameters considering a binary classification problem.

Table 3.1: Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Where:

- True positives(TP): the number of samples correctly classified as given label.

- True Negatives(TN): the number of samples correctly classified as not given label.
- False positives (FP): the number of samples incorrectly classified as given label.
- False negatives (FN): the number of samples incorrectly classified as not given label.

Using the above values we can calculate the performance labels, such as accuracy, precision, recall and F1-score.

Accuracy is the ratio of correctly classified samples to the total number of samples.

$$Accuracy = \frac{\text{Number of Correctly Classified Samples}}{\text{Number of Total Samples}} \quad (3.2.3)$$

Using the classified samples categories in the confusion matrix above, accuracy is mathematically described as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.2.4)$$

As simple and intuitive metric as it is, it can be misleading when used on imbalanced datasets, as it favors the majority class, and it would be an ineffective index of the model's performance in recognizing the minority class.

Precision is a metric used to measure the positive samples that are correctly predicted from the total predicted samples in a positive class and is mathematically defined as:

$$Precision = \frac{TP}{TP + FP} \quad (3.2.5)$$

Recall, or True Positive Rate, is the ratio of true positive classifications out of the total number of positive samples. Mathematically,

$$Recall = \frac{TP}{TP + FN} \quad (3.2.6)$$

What is observed in the above equations and 3.1 is that Precision involves the expected positive samples (TP and FP), and outputs the ratio of successful positive classifications out of the expected positive samples. Optimizing the precision value translates as minimizing the model's false alarms.

Recall, on the other hand, involves TP and FN values and focuses on how many of the samples that the model classified as positive, were actually positive. A higher recall value indicates the model's ability to capture relevant instances.

Generally, metrics emphasizing on True Positive and False Positive values without considering the Negatives may produce high scores that can be misleading for tasks where the minority classes are significant [18].

This suggests that the inverse relationship between the two metrics, also discussed as the precision-recall trade-off, makes them evaluate different aspects of the model's performance. Maximizing both metrics is only sometimes possible, therefore, the optimal solution would be to maintain a balance between them.

F1-score, is a combined metric that represents the harmonic mean between recall and precision values.

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (3.2.7)$$

F1-score considers both the ability to correctly identify positive examples (precision) and the ability of the model to identify all the positive examples (recall), making it a robust and safe metric for model evaluation and optimization.

The final stage before a network produces an output involves the **Activation Function**. This function **determines the result on the output of a neuron**, taking the samples from the training dataset, mapping the input data to the corresponding labels.

Among the most essential characteristics of an activation function are non-linearity, differentiability and boundedness. A non-linear activation function gives the model the ability to approximate any complex pattern within the data, while differentiability plays a crucial role in backpropagation, as it is required to the calculations of the gradients. A linear activation function has a constant gradient, making the backpropagation process impossible, as the weighted sum would remain unchanged.

Depending on the range of an activation function, different results are achieved in training. Bounded functions, like **tanh** and **sigmoid function** (3.2.2) prevent extreme output values, providing numerical stability to the system. However, when these functions take values close to their boundary values, the gradient is minimized and vanishes during backpropagation (vanishing gradient problem).

The vanishing gradient is a significant challenge in deep learning algorithms as it slows down and sabotages the training process, as the weights are updated minimally. Thus, function that avoid saturation are desired in complex tasks. ReLU 3.2.3 is a widely used activation function that is non-linear, differentiable, has an unlimited range of values in the positive plane and avoids some of the saturation and vanishing gradient issues. However, for input values below zero, the ReLU function undergoes very has a small gradient, which, in backpropagation, can also lead to "dead nodes", that will remain in the network but contribute little to training.

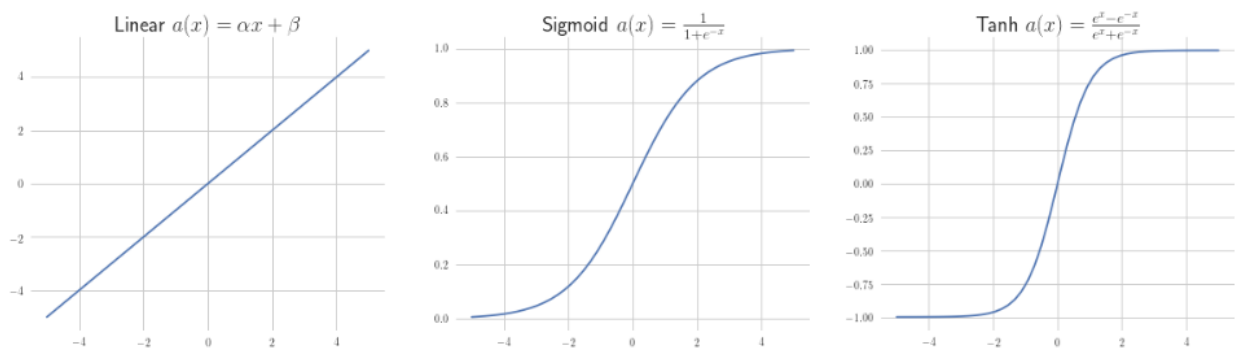


Figure 3.2.2: Examples of activation functions.

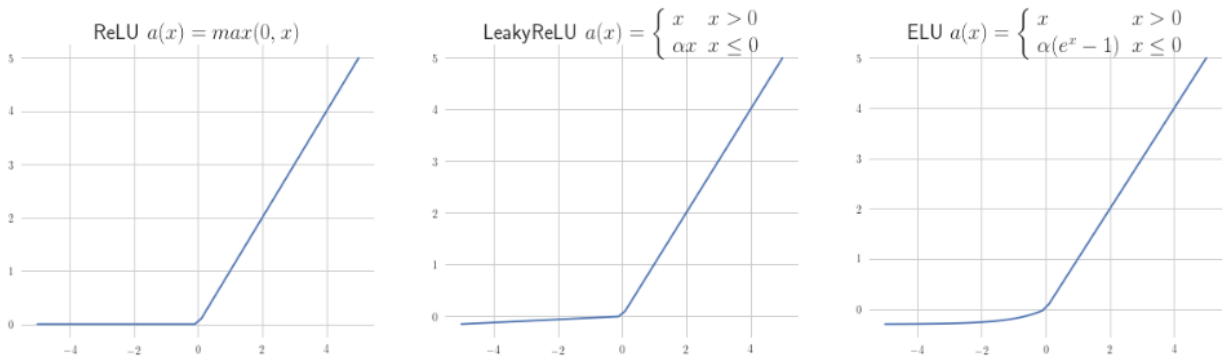


Figure 3.2.3: ReLU activation function and variants.

The role of the activation function is crucial and it should be chosen depending on the nature of the task. While there is no fixed rule on selecting, the scientific community has arrived at some preferences, giving some guidelines [23]:

- **ReLU** is a widely used function that performs well in many contexts. Nevertheless it is only to be used in the hidden layers of a neural network.
- In case there are "dead" nodes in the network, it's preferred to use **leaky ReLU**.

- Considering the vanishing gradient problem, the use of sigmoid and tanh function needs to be considered carefully. The functions are frequently applied in classification problems, image classification but avoided in Deep Networks.

3.2.4 Overfitting and hyperparameter fine-tuning

The main objective of a machine learning algorithm is to retrieve useful information and recognize patterns from data on the training set and make decisions on the test set. The ability of a model to perform well on data that it has not encountered on the training phase is called generalization. An indicator that a model generalizes well is a low value in loss function and error rate or high values evaluation metrics. The inability of a model to generalize is commonly associated with overfitting or underfitting.

Overfitting and Underfitting

Overfitting occurs when the model memorizes the details (noise) of the samples during training, resulting into making decisions on false criteria later on unseen data. An indicator of overfitting is substantial differences between training loss and test loss values, considering that the training loss is already low. Underfitting on the other hand occurs when the model lacks complexity in order to retrieve and learn information on the data, resulting in high training error.

The key to effectively training and constructing a model lies in striking the balance between overfitting and underfitting – achieving a favorable tradeoff between training error and test error. The paragraphs below are dedicated in discussing the factors that influence the training process and how we can handle them in order to ensure the model’s optimal performance.

Architecture

Firstly, as discussed thoroughly in 3.2.2, the architecture and capacity of a model are of great significance when it comes to generalization. The number of nodes, layers and connections and how they influence the model’s training process, a more complex model is prone to memorizing the data, thus overfitting while a simpler model may underperform.

Hyperparameter tuning

Following the architecture of the model, comes the hyperparameter tuning process. Hyperparameters are external configuration variables that manage the machine learning model training. Due to the rapid evolution of the field, the tasks machine learning is called upon to solve are becoming more and more complex, and the need to control the training process results in an escalating number of hyperparameters. Here, we will mention some of the most important and how we can handle them optimally.

Learning Rate (LR): a positive scalar that determines the step size during optimization process of a model [11]. In many cases, the learning rate is not a fixed constant but is varied and scheduled to be adjusted dynamically during the learning process in response to the model’s performance. These adjustments are carried out by an optimizer, such as Gradient Descent(GD, SGD) or Adaptive Learning Rate (ADAM).

Batch Size: refers to the number of training examples utilized in one iteration during model training. Choosing the appropriate batch size involves considering the model architecture, dataset characteristics and also computational resources. In general, a larger batch size provides a smoother gradient, although it requires more memory and could potentially lead to the model overfitting. Smaller batch sizes introduce more noise to the training process, aiding as a regularizer to the training, although excessively small batches prevent the model from generalizing well and might influence the training duration.

These two parameters receive much focus during fine-tuning processes, as they significantly influence the speed at which the model converges.

Regularization techniques

In contrast to controlling the complexity of the model beforehand, through model parameters, architecture and functions of the model, we can also use regularization to reduce overfitting [32]. Regularization methods reduce the complexity of the model during training.

L1 and L2 Regularization techniques penalize larger complex models by adding an extra term λ in the loss function. The general mathematical expression term is:

$$cost = lossfunction + regularizationterm \quad (3.2.8)$$

This technique reduces larger weights and prevents some connections from becoming dominant. The λ term, which is included in the *regularizationterm* above, can oversimplify the structure of a deep learning model when receiving very large values, whereas extremely small values makes the regularization inefficient. L1 Regularization, also called Lasso Regression, penalizes the sum of absolute value of weights, while L2 Regularization, or Ridge Regression, penalizes the sum of square weights. L2 is more commonly used due to its computational efficiency.

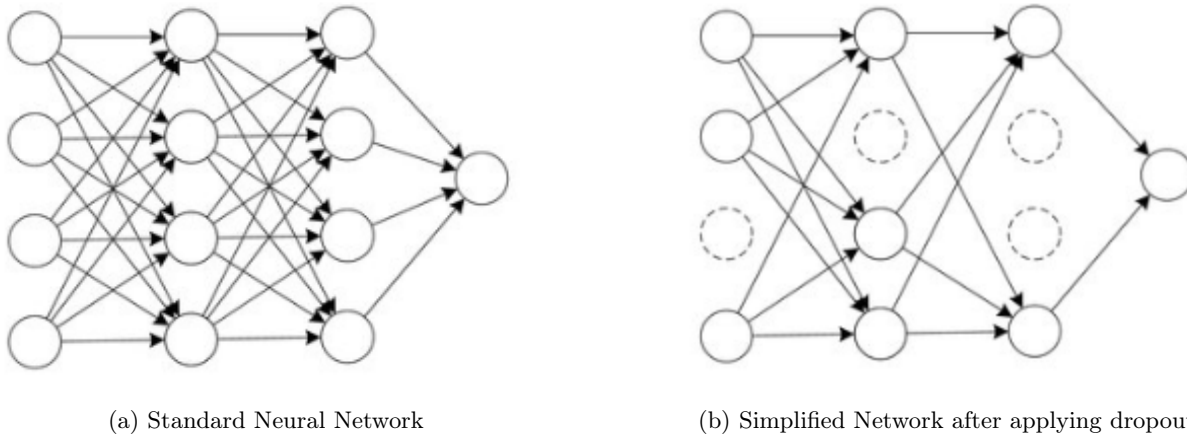


Figure 3.2.4: Comparison of a neural network before and after dropout is applied

Adding to the above, dropout and data augmentation are also commonly used regularization techniques.

Dropout is a technique that deactivates nodes of the network randomly during training, making the network less sensitive to specific weights. The result is a simplified network that is less prone to overfitting ?? . The dropout probability for each weight updated step is suggested to be around 20% - 50% .

Data augmentation is a method that creates fake data by applying random transformations to the *training dataset* to equalize imbalances, add data variations and, thus, avoid overfitting. It is a straightforward regularization technique, widely used in image classification, NLP and generative tasks.

Among the above-written and other regularization techniques, for the needs of this thesis we will be experimenting with dropout rate, data augmentation, weight decay and early stopping, and will be discussed in depth in the later chapters.

3.3 Natural Language Processing

The recent technological revolution and rapid development of machine learning has laid the foundations for integration with other scientific fields. Natural Language Processing (NLP) is an interdisciplinary subfield of computer science and linguistics [3]. It is the field of machine learning that revolves around the representing, understanding and generating human language through computational algorithms, using rule-based or

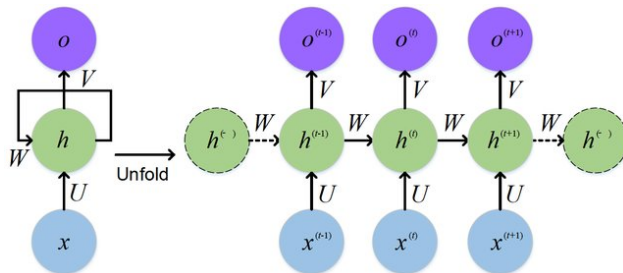


Figure 3.3.1: Standard RNN and Unfolded Version [6]

probabilistic models. The field includes language representations, speech, syntax, semantics of words and expressions in order to interpret and manipulate natural language data in order to use it in various algorithms, tools, and methods.

3.3.1 Brief Course of Evolution

The field emerged in the late 1940s with Machine Translation as its main field of activity, however, it was later established as Natural Language Processing as the scope of application became broader [16]. For the later part of the 20th century, the field was relatively inactive. That was until the introduction of the feed-forward Neural network, a probabilistic language model, in 2001 [2]. The feed-forward NN uses a lookup table of n previous words in a sequence to predict the next one. Later, multitask learning for language processing was proposed, with a multitask model that assigned grammatical categories and classified named entities. This model was the middle step that led to the introduction of word embedding, a process that addressed representing text as dense vectors in a continuous vector space. This representation encapsulates semantic information about words and their contextual relationships in a compact and meaningful manner. Later a general framework for sequence-to-sequence word mapping was proposed, using encoder-decoder architecture.

3.3.2 Recurrent Neural Networks

Neural networks in the field of NLP were discussed a lot at the beginning of the 21st century. In 2013, Recurrent Neural Networks were introduced to the field for language modeling, and very quickly became a popular choice for modeling long text sequences.

Recurrent Neural Networks [21], [6], as the name states, use feedback loops and go through the training data in recurrent cycles during training. RNNs fall into the class of supervised machine learning models and require a training dataset of input-target pairs.

A simple RNN consists of an input layer $x^{(t)}$, a recurrent hidden layer $h^{(t)}$ and an output layer $o^{(t)}$, as seen in 3.3.1. The input layer has N input units, that process the data sequentially, one element at a time. The input units are connected through weights to the hidden units in the hidden layer, where the weights are mapped in a weight matrix U . The hidden layer has M hidden units, that share their parameters through recurrent connections. This helps the network share information across different elements in the sequence. When initializing the hidden units we use small non-zero elements in a bias vector b_1 . As the training proceeds, the information flows through a feedback loop recurrently in the hidden layer. At each step, the hidden state $h^{(t)}$ is updated based on the input $x^{(t)}$ and the previous hidden states $h^{(t-1)}$. Thus, at each time step, $h^{(t)}$ contains information from data of past time steps, giving the network the ability to "remember". Mathematically the training process can be summed up in the following equations:

where b_1 and b_2 are bias vectors, σ is an activation function and U , V and W are the weighting matrices of the input-to-hidden connection, hidden-to-output connection, and hidden-to-hidden connection, respectively.

3.3.3 LSTMs and GRUs

Despite their novelty, standard RNNs face the challenge of vanishing gradient during training. Long-Short Memory Models (LSTM), a modified version of RNNs, address this challenge by replacing a hidden layer of

Figure 3.3.2: RNN mathematical equations as given in [6]

$$\begin{aligned}\alpha^{(t)} &= b_1 + Wh^{(t-1)} + Ux^{(t)} \\ h^{(t)} &= \sigma(\alpha^{(t)}) \\ o^{(t)} &= b_2 + Vh^{(t)}\end{aligned}$$

the Standard RNN with a memory cell to store and output information. The flow of information in the cells is controlled by a set of "gates", that, depending on the previous inputs and state of the cell, allow the network to selectively store or forget data. LSTMs became very prominent in the field, as they retain important information for a much longer time and disregard the less important information. Further improvements constructed the Gated Recurrent Unit (GRU), a simpler version of the LSTMs that replaces the set of gates with a single "update gate" [13].

3.4 Transformers

Further development on Machine Translation systems lead to the enhancement of Attention Mechanism in encoder-decoder systems. This turned out to be a significant advancement to the field, with a great focus on sequence to sequence models. Attention, together with Transformer models, is currently the state of the art in language processing tasks, with Large Language Models dominating the research field.

3.4.1 Attention Mechanism

The concept of attention was first introduced to encoder-decoder models for machine translation in 2014 by Bahdanau et al [1].

To explain how the attention mechanism operates, we will firstly refer to the example of additive attention in Machine Translation and Alignment, such as the RNNsearch model uses in [1].

The model follows an encoder-decoder type of architecture, with main objective to translate an input sentence x to an output sentence y .

The encoder is a BiRNN. For every term x_i of the input \mathbf{x} , the encoder computes an annotation term \mathbf{h}_i such as:

$$(h_1, h_2, \dots, h_T) = \text{BiRNN}(x_1, x_2, \dots, x_T). \quad (3.4.1)$$

The decoder consist of an attention function and an RNN. At each time step t , the RNN is characterized by a hidden state s_t . The attention mechanism produces an embedding c_t called *context vector* using the input's annotation term 3.4.1.

The context vector c_t is derived in the following steps. The attention function takes as input the previous hidden state of the decoder RNN, s_{t-1} and the annotations h_i described in 3.4.1, thus computing an alignment model:

$$e_{ij} = f(s_{i-1}, h_j) \quad (3.4.2)$$

which scores the matching level of the inputs around position j to the outputs around position i .

The scores are then passed through a Softmax function to obtain a set of *attention weights* α_{ij} , that express the probability of the target word y_i being aligned to, or translated from, a word x_j from the input.

$$\alpha_{ij} = \frac{e^{ij}}{\sum_k e^{ik}} \quad (3.4.3)$$

Finally, c_t is computed as a weighted sum of the annotations h_i based on the weights α_{ij} :

$$c_t = \sum_i \alpha_{ij} h_i \quad (3.4.4)$$

The most probable output symbol y_t is computed as follows:

$$p(y_t|y_1, \dots, y_{t-1}, x) = \text{RNN}(c_t) \quad (3.4.5)$$

In the context of [1], the attention scores e_{ij} have to go through $m * n$ iterations, where m the length of the encoder sequence and n the length of the decoder sequence. Vaswani et al. 2017 [27] proposed a more efficient model to compute the attention scores e_{ij} , that first projects s_t and h_t onto a common space and then calculates the dot-product of the two to obtain e_{ij} .

$$e_{ij} = f(s_i)g(h_j)^T \quad (3.4.6)$$

where $g(h_j)$ is the encoder projection vector and $f(s_i)$ is the decoder projection vector.

The differences between 3.4.2 and 3.4.6 is that in the later case, we only need to calculate $f(s_i)$ m times and $g(h_j)$ n times followed by matrix multiplication calculation of the two vectors, making dot-product attention faster and more space efficient in practice.

In the context of [27], the encoder projection vector $g(h_j)$ is called query \mathbf{Q} , the decoder projection vector $f(s_i)$ is called key \mathbf{K} , and the input annotations \mathbf{h}_i is called values \mathbf{V} .

Thus the attention is calculated as follows:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.4.7)$$

where d_k is a scaling factor.

Multi-Head Attention

Multi-head attention is an extension of the attention mechanism also introduced by [27]. In this concept the queries Q , keys K and values V are projected into the same subspace and then the attention function is performed in parallel. This allows the model to attend to information at different positions between different projections of the input.

Self-attention

Self-attention is a method utilized by transformers, in which case the query, key and value are all derived from the same set, allowing each element to attend to others within the same set. Self-attention connects all positions of the input with a constant number of sequentially executed operations, without needing the input-target corresponding.

3.4.2 Transformers Architecture

As described in 3.4.1, the original transformer model follows the encoder-decoder architecture.

- Input representation: The input sequence is mapped into a continuous vector.
- Encoder: is composed by a multi-head self-attention layer and a position-wise feed-forward network.
- Decoder: Similar to the encoder, the decoder is composed by multi-head self-attention layer for the targeted output with added mask, that prevents the model to attend to future elements during training. The additional sub-layer of multi-head attention feeds to the decoder the encoder output, aligning and finding correspondence between input-target sequences. The two attention layers are followed by a feed-forward neural network.
- Output: The decoder outputs a probability distribution over the output vocabulary, and after a linear transformation and a Softmax activation, the final output will be the most probable element of the output vocabulary.

The complexity and the approach of projecting the input-target correlations into a common geometric space, gives transformers great versatility and allows them to be converted into variations with multiple usability.

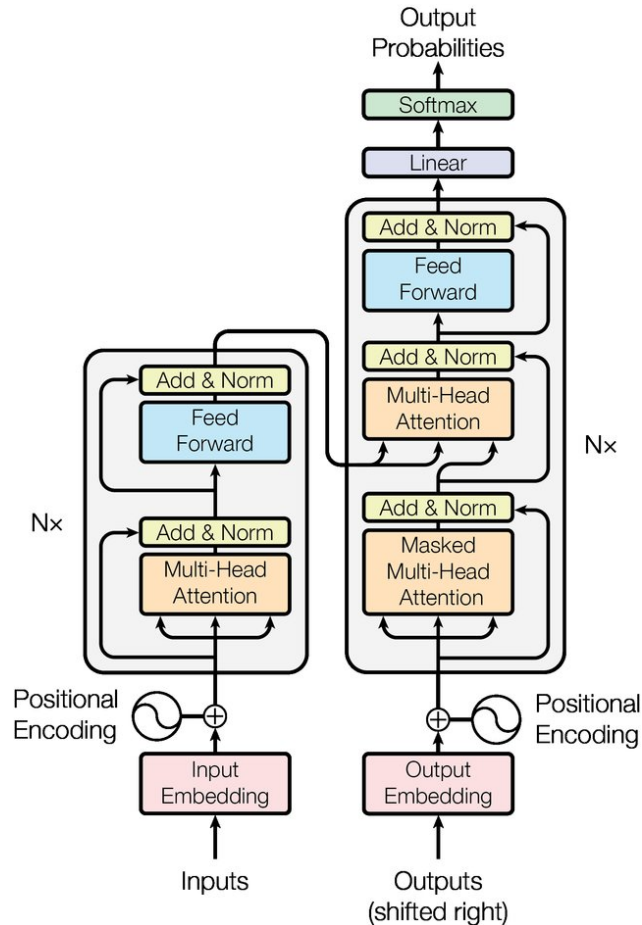


Figure 3.4.1: The Transformer - a model architecture proposed by [27]

Encoder-Decoder Models

Encoder-Decoder models are constructed according to the full transformer architecture 3.4.1 described above. The encoder processes the input and creates context representations that are later used by the decoder that generates the output sequence step-by-step. **T5** is a state-of-the-art encoder-decoder Transformer, typically used in sequence-to-sequence modeling tasks, such as machine translation, sequence-to-sequence generation etc.

Decoder Only Models

Decoder-only models are constructed by decoder blocks and removed the cross-attention module between the encoder and decoder, since they lack an encoder to connect to. Without context information, these models use positional encoding and masked multi-head attention on the input. The attention focuses on the previous (left) and not the forthcoming parts of the input. A distinctive decoder-only language model is **GPT2**, which is the predecessor of GPT-3. It's objective is to predict the next most suitable token of a sequence, gradually completing the sentence. Decoder-only models are typically used for tasks such as text generation and summarization.

Encoder Only Models

Encoder-only models are built using transformer encoder blocks [10]. Due to the self-attention mechanism, encoder models extract information within the input, making them excellent to develop language understanding systems. Encoder-Only transformer models, like **BERT**, are the state-of-the-art methods on NLP, and they are suitable for language understanding tasks (sequence labeling, sentiment analysis, text classification).

3.4.3 Pretraining

Large Language models (LLMs) are mainly transformer models. These models are often pre-trained on large corpora without a predefined task, in order to extract dependencies in large texts. These dependencies will later be used as a knowledge base for further fine-tuning on specific tasks with smaller datasets. There are numerous techniques used in the pretraining phase of a language model.

- **Masked Language Modeling:** Certain words of the input sequence are randomly masked by a designated token and the model is trained to predict the original word based on the context provided by the surrounding text. This technique is mostly used on encoder-only models
- **Causal Language Modeling:** This technique is often used on autoregressive models, mostly decoder-only or encoder-decoder models, that predict the next token given a prior sequence.
- **Multitask Learning:** The model is trained to perform multiple related tasks simultaneously. This pretraining technique is most suitable for encoder-decoder models. The distinction between each task can be done with a prompt or a task specific token given at the input. Throughout training, tasks share common parameters, enabling the model to employ the inherited relationships in the entirety of the model's objectives, without the need for specific fine-tuning.

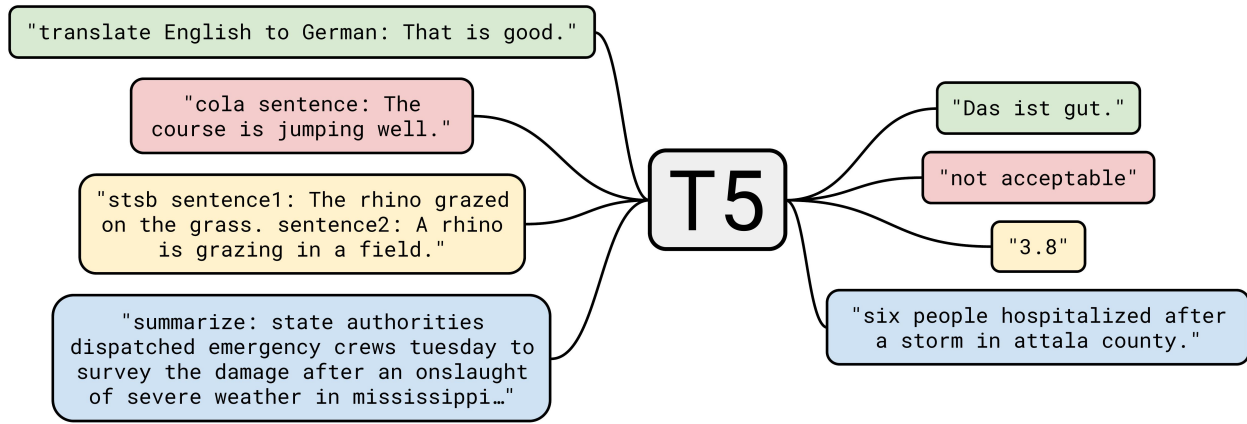


Figure 3.5.1: T5 Transformer: a flexible transformer suitable for multitask learning

3.5 T5 Model

The T5 model is a state-of-the-art Seq2Seq model, developed by Google AI, based on Encoder-Decoder architecture. The Encoder employs self-attention mechanisms to capture contextual information within the input sequence, which allows the model to effectively understand and encode the nuances of the input text. In contrast, the Decoder utilizes self-attention while also attending to the output of the Encoder, enabling it to generate coherent and contextually relevant output sequences.

Notably, the T5 architecture excels in capturing long-term interdependencies, making it well-suited for tasks requiring an understanding of extended context.

Operating on fixed-length input and output sequences, the T5 framework represents both input and output as text sequences, facilitating seamless text-to-text processing. Its multitask nature enables simultaneous training on diverse tasks, promoting knowledge sharing across domains and facilitating the utilization of an objective loss function.

All of the above make the T5 transformer model a flexible tool for multitasking exploration, well-suited for diverse tasks and offering ample opportunities for exploration.

Pretraining methods

Similar to many other transformers, T5 can be pretrained in a self-supervised or supervised manner and later be fine-tuned for specific tasks.

A self-supervised approach, is pretraining on a large corpus of sequential data, in order to learn from a vast array of recurring patterns and structures.

A more supervised approach to pretraining, is the Text-to-Text Pretraining Format: the data is formatted into a text-to-text framework, where each training instance contains both input and output text sequences. This approach facilitates the model's ability to understand and generate text across various tasks.

Masked Language Modeling (MLM) serves as an additional pretraining technique aimed at increasing the model's grasp of language context. During this process, a random subset of tokens within the input sequence is obscured, prompting the model to predict these masked tokens by drawing upon contextual information embedded in the surrounding text. By engaging in this exercise, the model sharpens its capacity to infer missing information and elevate its overall language comprehension ability.

Chapter 4

Music Theory and Data Representation

Music, as a general term, is the manipulation of sound and the means it travel through, intertwining with rhythm, harmony and melody, in order to create expressive content. The ability of the human brain to develop music emerged alongside with linguistic competence.

The intentional production of sound arose to assist in organizing labor, improve long-distance communication, it was used as a coherence tool in communities and as a defence mechanism against potential threats. As an evolutionary consequence, music became means of human expression, creativity, communication and part of civilizations' cultural attributes.

Melody and lyricism became the way in which people narrate their daily lives, worship deities, promote intellectual culture and therefore build and preserve history, customs and traditions. Today music has transformed into an art form, a platform of expression and a subject of interest in creative composition and interpretive analysis.

4.1 Theoretical Background of Music

The reasons behind humans' interest in sonic impulses are the morphology of sound signals and the biological structure of the human nervous system. In the following subsection we will explore the fundamental elements of a **harmonious sound** and the **brain's cognitive ability** to interpret them.

4.1.1 Math, Sound and Perception

Human's audible spectrum is between 20Hz - 20kHz. Our ears are especially sensitive to sounds from 1kHz - 6kHz, which is the fundamental frequency range of the human voice. The brain has the ability to distinguish and process in detail sounds within that range, allowing us to hear clearly when a person whispers, recognize different pitches and musical instruments and generally comprehend the details of a sound.

A **sound** is a set of waves that originates from a vibrating source. Sound waves can be produced by the surface of a drum, the strings of a guitar, the human vocal chords or, in case of wind instruments, the air column inside the instrument. All of the above can be modeled as a standing wave on *vibrating string*.

A vibrating string produces sound waves at frequencies that are integer multiples of the fundamental frequency. This means that when a string or chord vibrates at frequency f , it also emits sound waves that at frequencies $2f$, $3f$, and so on, as seen in 4.1.1. These sound waves are called **harmonics** [30] and includes all pitches in a harmonic series (including the fundamental frequency). The term "overtone" refers to the pitches above the fundamental.

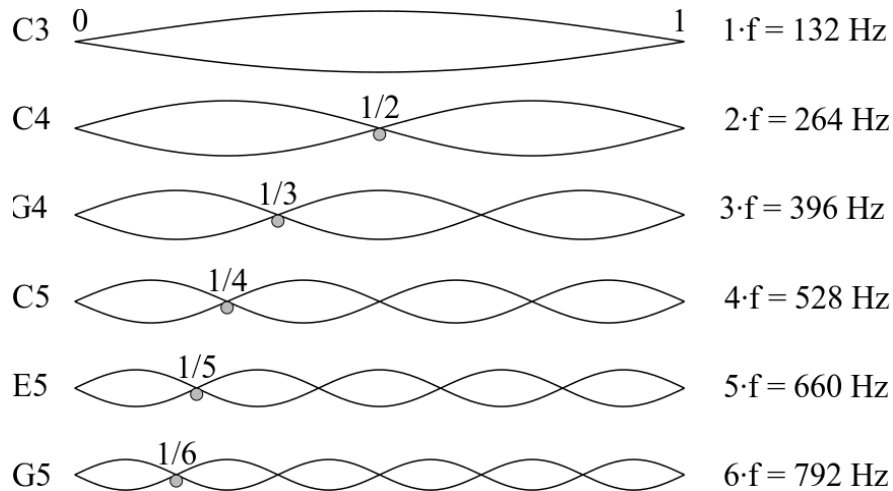


Figure 4.1.1: Harmonic partials on strings

Harmonics form an arithmetic progression and can be represented by a linear function, however, we respond to sounds non-linearly. The human brain perceives low-pitch harmonics as farther apart and high-pitch harmonics as closer together, reflecting a logarithmic function in human perception.

A way of organizing pitch relationships in a natural and intuitive way for nonlinear human perception is the **octave scale**. The aforementioned, models the frequencies in a geometric sequence ($f, 2f, 4f, 8f, 16f$), with each element having twice or half the frequency of its successive elements. This system aids in our comprehension of music by simplifying complex functions with high variations and non-linearity, making musical understanding more accessible.

4.1.2 Harmonic resonance in music

We interpret a note's pitch as the fundamental frequency f of its periodic waveform. The second harmonic, whose frequency is twice the fundamental, sounds an octave higher; the third harmonic, three times the frequency of the fundamental, sounds a perfect fifth above the second harmonic. The fourth harmonic vibrates at four times the frequency of the fundamental and sounds a perfect fourth above the third harmonic (two octaves above the fundamental). This phenomenon is illustrated in musical notation at 4.1.2.

The harmonics of a note resonate together simultaneously, each at a lower volume. When we hear a C2 note, we simultaneously hear a C3, G4, C5, E5, each diminishing in intensity.

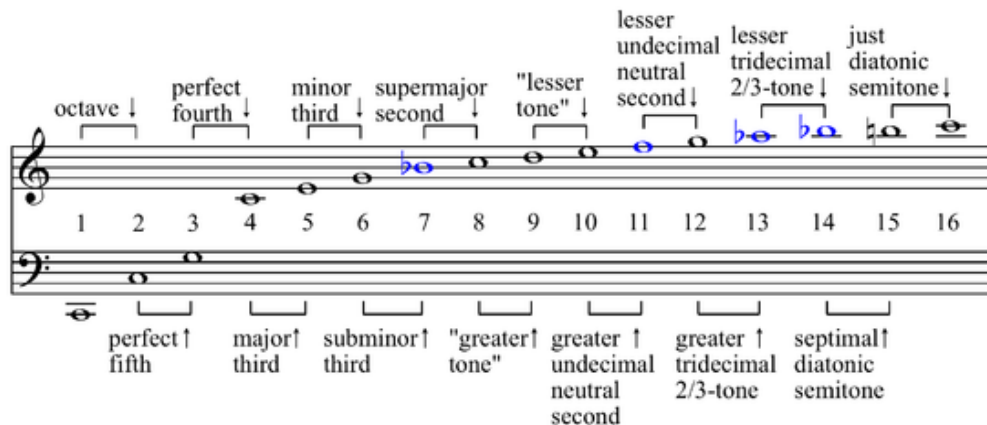


Figure 4.1.2: Harmonic Intervals of note C2

Pitches with a simple ratio also share common overtones. When these pitches are heard together or in succession, the overtones of each pitch harmonize, which the human ear finds pleasing. In musical terminology, this is known as consonance. The octave, perfect fifth and perfect fourth, with frequency ratio 1 : 2, 2 : 3 and 3 : 4 respectively, are great examples of consonances. The more complex a ratio gets, audible common overtones become fewer and the final result is cacophonous. These sounds are called dissonances. Among them is the infamous **tritone**, also known as *the devil's interval*, which has a fairly complex ratio of 32 : 45.

Table 4.1: Harmonious intervals of C and their frequency ratios

Interval	Example notes	Frequency ratio
Octave	C-C'	1:2
Perfect Fifth	C-G	2:3
Perfect Fourth	C-F	3:4
Major third	C-E	4:5
Minor third	A-C	5:6

Dissonant sounds cannot be avoided and, nonetheless, contribute to the interest and diversity of a musical composition. However, to achieve harmony, it is essential that dissonances be resolved by consonances.

The mathematical relationships between underlying musical intervals were formalized and systematized by ancient greek mathematician and philosopher Pythagoras. He experimented and observed the fundamental importance of simple numerical ratios that define musical intervals and developed Pythagorean Tuning [31], a framework that contributed to the Western music theory and practice and influenced musical compositions.

4.2 Overview of Historical Periods

Mesopotamia

Signs of musical and lyrical competence are dated back to Mesopotamian history. Mesopotamia is among the earliest and most well-documented civilizations that have rich musical tradition. There are artifacts of instruments made of animal bones, depictions of musicians and musical instruments appear in the 4th millennium BC and clay tablets that recorded song, genres and also included instructions on how to play instruments. This evidence suggest that humans developed musical notation and theoretical understanding of music as early as the 3rd millennium BC.

Music in Mesopotamia serves ceremonial, religious and entertainments purposes, and was also intertwined with poetry and storytelling. Musicians were highly regarded and associated with religion and royalty.

Ancient Greece

An important milestone in the development of music theory is found in ancient Greece. The etymology of the word itself is derived from the Muses, daughters of Zeus and patron goddesses of creative and intellectual endeavors. It was believed that music evoked emotions, inspired courage and promoted moral values. It was considered essential for cultivation of mind and soul. For the above-mentioned reasons, musical competence was essential for people in Ancient Greece. Education emphasized the importance of musical literature and arts, Greek drama incorporated choral odes that complemented the spoken dialogue of the play.

Musical theory evolved alongside with mathematics and philosophy, concepts that were inevitably linked in the process. Pythagoras, a Greek philosopher credited with discovering the mathematical relationships between musical intervals, observed the lengths of vibrating strings that produce harmonic ratios, a discovery which is considered the foundation of harmonic theory and consonance. The Pythagorean tuning system, laid the groundwork for the mathematical understanding of musical harmony and the development of musical scales.

Medieval Period (500-1400)

Medieval music refers to the music that developed in the Middle Ages, the longest period in the history of Western music. Two genres are dominant in this era, secular, which was intended for a non-religious audience and sacred, that mostly served a religious purpose. Much medieval music was purely vocal or used only instruments, until the 9th century, a time that witnessed the beginning of polyphonic music. In polyphony, two or more notes sounding simultaneously are called a *chord*, and a sequence of chords is called a *harmonic progression*. Counterpoint, the art of combining multiple independent melodic lines, intersects with the rules of harmonic progression in creating rich and cohesive musical textures. While counterpoint focuses on the interplay of individual voices, following harmonic principles ensures that these voices move in a harmonically satisfying manner, contributing to a pleasant acoustic result. Singers in churches and monasteries began experimenting with adding an accompanying voice to sing many notes in parallel to the melody of the organ, emphasizing the perfect consonances, perfect fourths, fifths and octaves. Monasteries provided a great place for experimentation in polyphony, musical tradition survived through manuscripts of liturgical music.

Renaissance (1400-1600)

The Renaissance sparked a renewed focus on the ideas and values of ancient Greece. Intellectual waves explored philosophy, literature, arts and notably music. The new interest in the past came with significant innovations for the future. Music was freed from medieval constraints, there was room for variations in rhythm, range, harmony, notation. Polyphony was advancing, and counterpoint developed to study the interdependence and harmonic coherence of voices, regarding dissonances and consonances. Viol and keyboard instruments, such as the harpsichord, became components of the consort, from which early opera emerged. Music composition and performance attracted a lot of attention and the interest of intellectuals and both the upper and domestic class.

Baroque (1600-1750)

The Baroque style followed the Renaissance period. Composers aimed to achieving a fuller sound, that was rich and challenging, both to the audience and the performers. In this period, major and minor scales were solidified, music was composed following common-practice tonality, an approach to writing a song in a particular key. This approach is extensively used in classical music, later to be the foundation of western popular music. Composers revived new genres and forms of musical pieces, leading to the birth of suite, sonata, concertos etc, and established the mixed vocal-instrumental forms of opera, cantata and oratorio.

Among the most important composers of this period is **Johann Sebastian Bach**. The youngest of a family with great musical tradition, he wrote numerous works of both sacred and secular pieces that present didactic interest. The four-part harmony system formed the core of Bach's style, and his works are considered to establish the rules of this evolving schema, that would later dominate musical expression. Among sonatas, cantatas, fugues and an immense variation of works, Bach's four-part choral compositions managed to prescribe the principles of the four-part harmony system. They are studied among musicians until today.

Classic (1750-1820)

Despite its shorted duration, classical period had a pivotal effect on western musical style. Composers now emphasize on balance, clarity, virtuosity and take interest in instrumental music. The harpsichord is replaced by the fortepiano, the precursor of the piano, which allows the performer to play louder or softer, thus enhancing the works with emotional depth. Music is mainly *homophonic*, meaning there is a clear melody line over a subordinate chordal accompaniment, that follows the rules of counterpoint established in the previous eras. Classical period composers created highly esteemed works, like symphonies, sonatas and concertos, performed by orchestras or featuring a virtuoso solo performer (piano, flute, violin) accompanied by an orchestra. Among the key figures of this era were, Joseph Haydn, Wolfgang Amadeus Mozart, Ludwig van Beethoven, and Franz Schubert.

Romantic (1820-1900)

Romanticism emerged alongside with the Industrial Revolution, and the ascent of the middle class. The middle class's growing interest in music composition moved it from elite settings, opera houses and theaters, to public venues, like concerts and festivals. Romantics sought to compose music that provoked emotions, breaking away from strict rules and conservative norms. They tended to make greater use of longer, more fully defined and more emotionally evocative themes, used more elaborate harmonic progressions, increased chromaticism and tonal range in order to sustain musical interest in their works. Less common musical structures such nocturne, concert etude and arabesque emerged. Despite his classical influence, Ludwig van Beethoven was the first composer to introduce typical romantic elements in his works. Notable figures of the era include Polish composer Frédéric Chopin, Robert Schumann and Franz Liszt.

Modern and Post Modern (1900 - Present)

Novelty styles, techniques, and genres that have emerged since the late 19th century are enclosed in the modern and contemporary era. From revolutionary experiments to minimalist compositions, modern music has continually pushed the boundaries of traditional tonality and form. The advent of electronic music, spurred by innovations in technology, has opened up new sonic possibilities and expanded the scope of musical expression. Contemporary music reflects the multiculturalism and globalization of the 21st century, with artists drawing inspiration from a wide range of cultural traditions and blending genres to create innovative fusions. From avant-garde experimentation to mainstream pop, modern and contemporary music continue to evolve, challenging conventions and reflecting the ever-changing landscape of human experience. Moreover, mathematical principles are increasingly employed in music composition, resulting in intricate structures and novel sonic experiences.

Music in the time of "now"

Today music has no borders and no limitations. It is not restricted within the lines of a disc or a CD and it does not require a lot of means to be played or produced. Aspiring musicians have opportunity to create and share their work, with affordable recording equipment and interactive software. Tools and gadgets that provide the capability to manipulate sounds and deliver dynamic performances, even on the stage, are revolutionizing the music industry.

The development of digital platforms and streaming services has a profound impact on how music is distributed and consumed. YouTube, Spotify, Soundcloud and more are some worldwide scale platforms, where artists and listeners meet and interact. However, despite giving everyone the ability to produce, share and listen to music on demand, large discographies and curated playlists wield significant influence over music trends, determining commercial success and thus shaping the trajectory of overall artistic expression.

In this rapidly evolving musical landscape, technological innovation and artistic creativity meet, collide and interact, to redefine the boundaries of musical expression, creating room for more and more possibilities over time.

4.3 Music notation

Humans required a language to communicate and represent the auditorily perceived music, leading to the development of a notation system. These systems have varied across civilizations, cultures and time, with not particularly comprehensive pre-medieval iterations. Driven by the Christian Church's goal for consistency in chants throughout services, a notation system emerged, that closely resembles what we know as musical notations today. Later the Western Music notation system was established, which will be described in the following section.

The foundation of musical notation lies in the **staff**, a set of five horizontal lines and four spaces upon which musical symbols are placed. Each line and space on the staff corresponds to a specific pitch, determined by the **clef** symbol placed at the beginning of the staff. The clef symbol at the start of a musical passage establishes the pitch range for that passage.

The clef is followed by the **key signature** of the piece. The key signature is a set of symbols, placed on the beginning of the section to convey the tonality of the music. Each major and minor scale has its corresponding key signature, showing up to seven flats or seven sharps, called accidentals, representing the notes used within the scale. The accidental symbols are placed on a line or space of the staff and affect all the subsequent occurrences in the same pitch class within the section.

Time signatures are placed next to the key signatures of a musical piece, and play a crucial role in defining the meter and rhythm of a musical composition. A time signature consists of two numbers written as a fraction at the beginning of a piece of music. The top number indicates the number of beats in each measure, while the bottom number represents the type of note that receives one beat. For example, in 4/4 time, commonly known as "common time," there are four beats per measure, with a quarter note receiving one beat.

Notes are the oval symbols placed on the lines or spaces of the pentagram. A note's **pitch** is related to the vertical positioning on the staff, i.e. which line or space they are on¹.

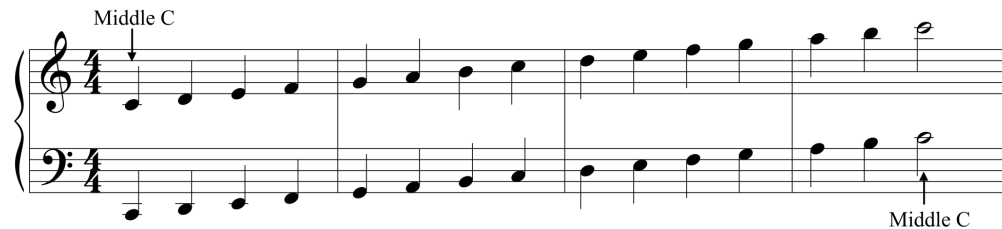


Figure 4.3.1: Basic notes on a staff. The pitch is defined by the vertical position between the staff lines and spaces.

The pitch can be modified by using a corresponding accidental (a sharp, flat or natural symbol) to the left of the note.

The **duration** of a note is symbolized by the note-head or with the addition of a note-stem plus beams or flags². The duration symbols were introduced in the context of the "common time", or 4/4 time signature. In the context of a 4/4 measure, a whole note takes up a whole measure, a half note takes up a half measure, a quarter note takes up a quarter of the measure, and so on 4.3.3. The presence of articulation marks, such as staccato dots, ties or accents, can affect the perceived duration and articulation of a note.

A musical piece is read and written from left to right. According to the instrument the musical piece is referring to, the musical notation could include harmonic intervals called chords or melodic intervals called melody. For example, a monophonic instrument such as a flute, is restricted to play one note at a time, so the musical notation consists of just a melody written on one staff. On the other hand, the piano utilizes a wider range of pitches at a time within a musical piece. Typically piano staves are characterized by their dual-staff structure, flexibility for polyphonic writing, and inclusion of key signatures, clef changes, pedal

¹<https://musictheory.pugetsound.edu/mt21c/OctaveRegisters.html>

²<https://www.schoolofcomposition.com/music-notation/>

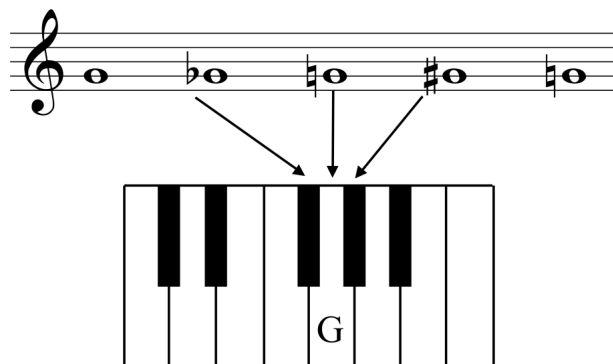


Figure 4.3.2: Accidentals of a note











	Note Value	Equivalent Rest
Whole		
Half		
Quarter		
Eighth		
Sixteenth		

Figure 4.3.3: Note duration values and equivalent rests

markings. Several musical compositions also include performance directions, such as dynamics, tempo and expression markings, to facilitate accurate and expressive piano performance.

4.4 Symbolic Music Representation

Music can be represented within the context of computational machines using various storage formats that utilize different data modalities. In the following sections, we offer a brief overview of the most commonly employed representation formats in the research area of Automatic Music Synthesis.

4.4.1 MIDI files

MIDI (Musical Instrumental Interface) [29] is a technical standard that describes a communication protocol, digital interface, and electrical connectors that connect a wide variety of electronic musical instruments, computers, and related audio devices for playing, editing, and recording music.

In a simplistic view, a MIDI file can be considered a score with additional optional annotations. As a result, you can count on getting a transcription of the song, as well as meter information such as beats and downbeats [26].

MIDI files primarily consist of digital instructions that represent musical events, such as note-on and note-off messages, velocity, pitch, and duration, as seen in 4.4.1. It serves as a format for music writing, enabling the simulation of compositions and facilitating variety and communication between different instruments, allowing them to control one another [24].

Figure 1.13 from
[Müller, FMP, Springer 2015]

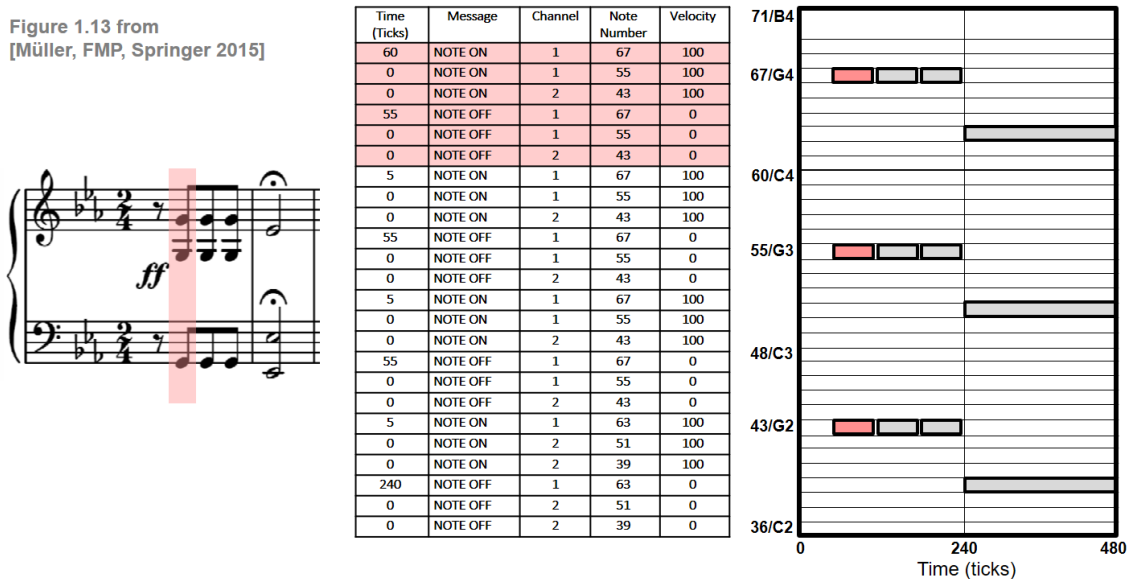


Figure 4.4.1: Symbolic music representations: a sheet music representation, a MIDI representation (in a simplified, tabular form), and a piano-roll representation. [24]

The variables equipping the note events describe the features of the note that is played at a certain time.

The MIDI note number is an integer ranging between 0 and 127 that encodes the pitch of a note. Mirroring the alignment of the acoustic piano, where there are 88 keys that correspond to musical pitches ranging from A0 to C8, the MIDI number encodes the musical pitches C0 to G9 in increasing order. For example, the middle C corresponds to the MIDI number 60.

The key velocity is again an integer ranging from 0 to 127, that controls the intensity of the note that is played. In note-on events, it regulates volume, while in note-off events, it manages decay during the release phase. However, its precise effect varies based on the instrument or synthesizer being used.

The MIDI channel acts as a means to organize and route MIDI data to control various instruments or sounds within a MIDI setup. Usually a channel is associated with a particular instrument, although it is not mandatory. Channel is also an integer ranging from 0-15.

Finally, the time stamp is an integer value that represents how many clock pulses or ticks to wait before the respective note-on or note-off command is executed.

The standardization of the MIDI protocol made it possible for complex productions to be realized on systems as compact as synthesizers with integrated keyboards and sequencers. This, along with the stabilization of the market for personal computers, provided people, ranging from music producers to home recorders, with the ability to record music. This simultaneous development played a significant role in the revival of music in the 1980s.

The MIDI file format is particularly used in music production and composition through computers, sound design for video games and films. It is also utilized in live performances, allowing musicians to control samplers, synthesizers and other electronic instruments in real-time.

Due to their precise control over pitch, duration and dynamics, lightweight nature and universal compatibility, MIDI files are suitable for computer music generation, analysis and transcription. Notable datasets include Lakh MIDI Dataset, with 176k unique MIDI files, MuseData, JSB Chorales and many more valuable resources in advancing AI-driven approaches to music generation.

4.4.2 MusicXML files

MusicXML is a versatile mark-up language, designed specifically for representing music notation. Files include symbols for beams, key signatures, and time signatures, along with a wide range of other musical symbols, that are capable of capturing and preserving the full range of musical elements. This makes MusicXML suitable for encoding complex musical scores with precise notation and formatting details. The largest publicly available MusicXML dataset is Theorytab Dataset (TTD), which contains 16K musical pieces, stored in MusicXML format.

Nevertheless, due to their detailed content, these files require more storage space and computational resources in order to be utilized. Due to licensing restrictions, the MusicXML datasets available are very limited and not as rich as the MIDI counterparts.

These practical and technical challenges make this file type a less suitable and popular choice among the research community, when it comes to music generation, as MIDI files make a more compatible choice.

4.4.3 Pianoroll

The Pianoroll file is an image-like music representation, structured according to the alignment of the piano keyboard. It visualizes music in a grid-like matrix, where the piano's pitches are depicted along the vertical axis and time is represented in the horizontal axis, formatted in discrete time steps.

The name "pianoroll" was inspired by the old storage medium, a continuous roll of paper with punched holes, that represented note control data, and was used in old piano players. MIDI files emerged as the digital successor of the traditional pianorolls, thus, the formats of the two digital filetypes are very similar. Consequently, only a few datasets are stored explicitly in pianoroll format, as MIDI files can easily be converted to pianoroll and vice versa. *Pypianoroll* is an open source python library that bridges the gap between these music representations, offering tools that can manage and manipulate both formats efficiently.

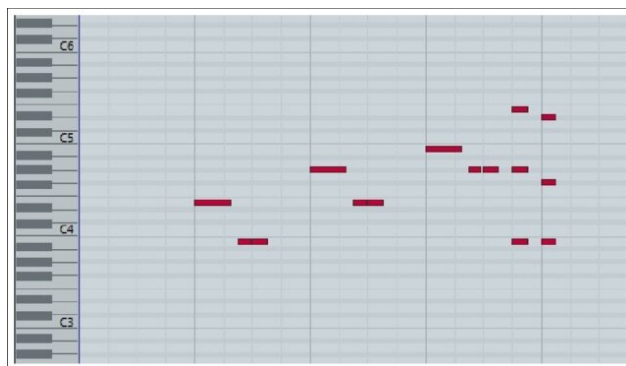


Figure 4.4.2: Pianoroll Music Representation [28]

The aforementioned symbolic music representations play a major role in machine learning applications, especially in tasks focused around music generation, transcription and analysis. Following the flourishing of the field, numerous specialized tools have been developed that greatly enhance our ability to handle and manipulate these representations effectively. Among these tools are Pypianoroll, MusPy, Music21, mido, miditoolkit and miditok, some of which will be utilized and further analyzed below.

Chapter 5

Tokenization

5.1 MidiTok

The advent of Language Models (LMs), changed the landscape of generative music tasks, mandating a shift to the approach of music representation, from a complex and multidimensional notation system to a more linguistic-like form. Thus, the concept of tokenization for symbolic music was proposed, as a crucial preprocessing step in preparing MIDI data for input into Neural Networks. The concept became more and more popular during the late 2010s and early 2020s, as the transformer architecture emerged.

In late 2021, Fradet et. al [9] introduced Midityok, a python package that serves as a tool to transform music language from MIDI files, to token sequences. It is among the first open source toolkits to offer a friendly and convenient way to tokenize MIDI files, encouraging the participation of more researchers in the music generation field.

The platform is built around the idea that all MIDI tokenizations share common methods. It features the most known MIDI tokenizers, along with functions crucial for preprocessing, such as data augmentation, dataset tokenization, and Byte Pair Encoding.

Midityok acts as a connecting link between Generative Music and Natural Language Processing tasks, allowing researchers to utilize the tools of the latter, particularly Language Models (LLMs), to address the requirements of the former. As of today the platform is actively maintained, receives regular updates, bug fixes and improvements, based on user feedback [7].

5.1.1 The Tokenization Process

In order to understand the utility and importance of the tokenization process, we need to dive into the form of the stored information in a MIDI file.

The contents of the file fall into categories; tracks of instruments, tempo changes, time signature changes, key signature changes, lyrics etc.

The musical information is contained within the track of the file, in the form of note events that occur at a certain moment in time. As described in 4.4.1, a note event has five characteristics; an onset and offset time, pitch, velocity and time step (tick).

The base time unit in MIDI files is the tick, with a resolution known as the time division expressed in ticks per quarter note. Events in MIDI files occur at specific ticks, represented by a high value, typically ranging from 384 to 480. Pitch range and velocity values are ranging from 0 to 127.

Using wide ranges of possible values with a language model is usually not optimal, as the later is a discrete model and so could struggle to efficiently capture the differences between two consecutive values.

The main idea of tokenization in the context of MIDI files, is to address the challenge posed by the wide ranges of these values present in the data.

By tokenizing the information, the continuous and varied attributes such as velocity and time, are down-sampled and quantized into discrete tokens. This quantization helps reduce the complexity of the data and optimizes model performance by providing a more manageable and structured input for the models to process.

Velocity affects the dynamics and volume of a note when played. It plays a significant role in denoting the musical expression and emotion. Although, a difference between values 100 and 101 cannot capture differences efficiently, therefore downsampling velocity values can help capture the nuances of dynamics while still reducing complexity.

Unlike velocity, pitch range directly impacts the sound and expression of a note. Reducing the pitch range will limit the musical variety and expression that can be captured within the musical piece. Therefore, maintaining the full pitch range is essential for a sufficient representation of the music.

Concluding, tokenizing MIDI files, allows for the representation of music data in a format that is more suitable for training models, enabling them to learn patterns effectively, generalize information, and make predictions in a more efficient manner.

5.2 MidiTok Tokenizers

Tokenization techniques offer different ways to represent musical information as tokens, each with its own advantages in terms of capturing musical attributes, reducing complexity, and optimizing model performance. The choice of tokenization method can significantly impact the effectiveness of training models on symbolic music data.

To address this, MidiTok introduces a diverse selection of tokenizers, each employing distinct methods of tokenization. These tokenizers offer varying degrees of specialization, with some of them being suitable for specific datasets and others designed for more generalized use cases.

In this subsection, we will present the most notable characteristics and functionalities of some of the framework’s tokenizers. To showcase the operations of each tokenizer, we will illustrate the tokenization process using the musical sheet below, as provided in ¹



Figure 5.2.1: Lead Sheet for tokenizer illustration, as presented in ²

5.2.1 REMI



Figure 5.2.2: Illustration of REMI tokenizer

REMI tokenizer, was introduced with the Pop Music Transformer in [14]. REMI represents musical notes sequentially, tokenizing MIDI data into Pitch, Velocity, and Duration tokens for notes, and Bar and Position tokens for time.

The extended version REMI+ can handle multiple instruments by adding Program tokens before Pitch tokens. Tempo information is represented as a single Tempo token. When decoding multiple token sequences, only the tempos and time signatures of the first sequence are decoded for the entire MIDI.

5.2.2 MIDI-Like Tokenizer

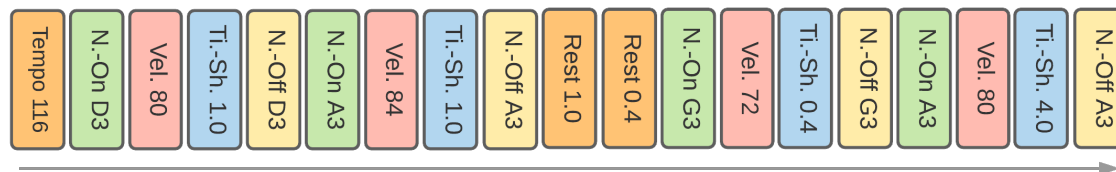


Figure 5.2.3: Illustration of REMI tokenizer

The MIDI-Like tokenizer, as introduced in [19] and utilized in subsequent works like Music Transformer and MT3, simplifies tokenization by directly converting MIDI messages (NoteOn, NoteOff, TimeShift, etc.) into tokens. It operates on a FIFO (First In First Out) logic during decoding. Like REMI, it can include Program tokens to specify instrument information and treat all tracks as a single stream of tokens.

¹<https://miditok.readthedocs.io/en/v3.0.1/index.html>

As MIDI-Like tokenizer incorporates NoteOff messages into its tokenization process, it employs a strategy to manage overlapping notes. This may lead to alterations in the durations of overlapping notes, in order to prevent conflicts with NoteOff tokens.

5.2.3 TSD

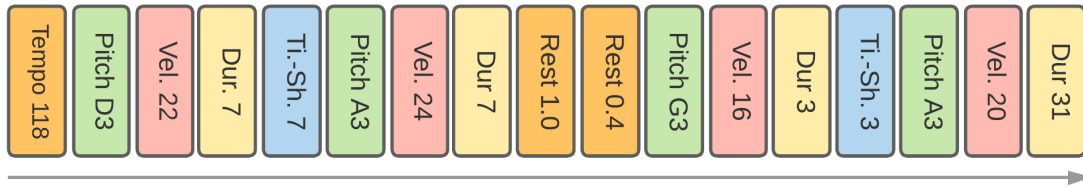


Figure 5.2.4: Illustration of TSD tokenizer

Addressing the challenges of Note-Off tokens in the previous tokenizer, the TSD tokenizer, or Time Shift Duration tokenizer, represents MIDI data using explicit Duration tokens for note durations. Same as the previous two, it can include Program tokens before each Pitch token to specify instrument information if configured to do so.

5.2.4 Structured tokenizer

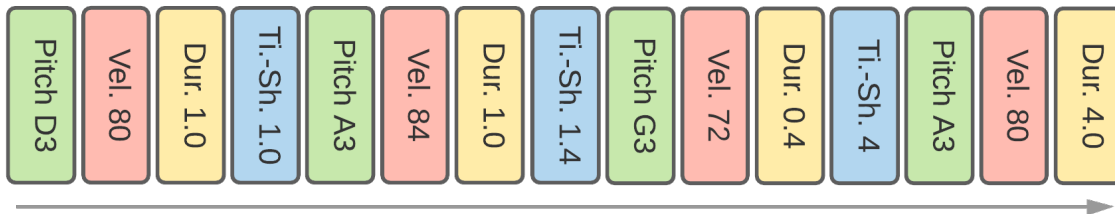


Figure 5.2.5: Illustration of Structured tokenizer

The Structured Tokenizer, introduced with the [12], operates similarly to the TSD tokenizer but follows a consistent token type succession (*Pitch*, *Velocity*, *Duration*, *TimeShift*). This structured approach is applied in every element of the musical piece and provides uniformity to the tokenized file.

Unlike REMI and TSD, which allow for additional tokens like TimeSignature, the Structured Tokenizer restricts the insertion of additional tokens except for Program tokens when configured to do so.

5.2.5 Embedding-Based Tokenization: Octuple, CPword, and MuMIDI

Further research and more novel approach to the tokenization process, introduces tokenizers incorporating embedding pooling operations to effectively make sequences shorter. Miditytok incorporates Octuple, CPword, and MuMIDI tokenizers. These tokenizers use embeddings to represent musical elements in a more compact and efficient manner.

Octuple employs embedding pooling to merge individual note tokens into compound tokens, that also exhibit uniformity, facilitating a reduction in overall sequence length while capturing essential musical information. Similarly, CPword utilizes embedding pooling to represent various musical elements, optimizing the tokenization process for multitrack MIDI data. Meanwhile, MuMIDI takes embedding pooling a step further by integrating learned positional encoding and representing all tracks in a single token sequence, handling multitrack data in a very efficient manner.

The aforementioned tokenizers present considerable complexity during training and decoding, making them suitable for small models.

The framework offers a wide variety of tokenizers, each presenting unique differences both in approach and complexity. Tokenizers such as REMI, TSD, Structured, and MIDI-Like adopt a traditional sequential

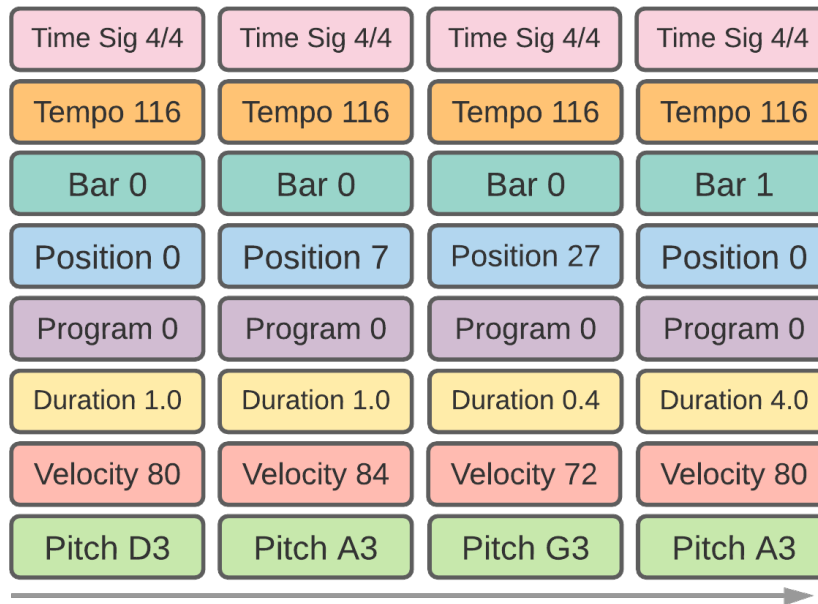
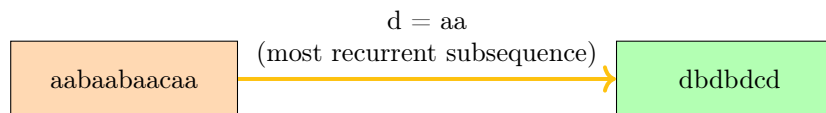


Figure 5.2.6: Illustration of Octuple, an embedding-based tokenizer

representation of musical elements. While effective in capturing the structure of MIDI data, these tokenizers are straightforward and suitable for simple musical pieces, but may lack the efficiency and compactness needed in large and complex tasks and may require longer sequences, leading to increased computational complexity and training times. On the other hand, tokenizers such as Octuple, CPword and MuMIDI that utilize embeddings, offer a more complete and versatile tokenization process, making them more suitable for more complex tasks. However, both embedding-based and traditional tokenization methods remain vital for generative music tasks, each offering distinct advantages tailored to different use cases.

5.2.6 Byte Pair Encoding (BPE)



After exploring tokenization techniques, we will investigate how vocabulary techniques commonly used in natural language processing are applied to music generation tasks using tokenized sequences. Music generation presents unique challenges due to the complexity of musical patterns, simultaneous note play, and multiple tracks. Initially, only small vocabularies of tokenizers were proposed, resulting in large token sequences. Recent research aims to address this issue by reducing sequence length, leading to the adoption of Byte Pair Encoding (BPE) in tools like miditok [8].

BPE (Byte Pair Encoding) is a compression technique used in natural language processing (NLP) to replace recurring byte sequences in a corpus with newly created symbols.

For example, in the sequence "aabaabaaca," the subsequence "aa" occurs frequently and is replaced with a new symbol, resulting in a compressed sequence. This process continues until a target vocabulary size is reached.

BPE is widely utilized in building tokenization vocabularies, enabling the encoding of rare words and the segmentation of unknown or composite words into sub-word units.

In symbolic music, tokens from a tokenizer without Byte Pair Encoding (BPE) are considered as the base

vocabulary, similar to characters in text. Thus, initially, the vocabulary consists of unique characters present in the data, which are then automatically grouped, learned as tokens by the BPE algorithm and added to the new vocabulary. BPE is computed using the Hugging Face tokenizers Rust library, enabling fast training and encoding.

Internally, base tokens are represented as characters, each having three unique forms: a textual description (e.g., *Pitch₅8*), an integer ID, and a byte form represented by a character or sequence of characters. Tokens learned with BPE are represented by the unique characters of the base tokens they represent.

It is crucial to emphasize that Byte Pair Encoding (BPE) should be initially learned and then applied consistently across all data interacting with a model. When performing data augmentation on the training set, BPE should be learned collectively on the training, validation, and test sets, followed by its application to each of these datasets respectively.

BPE has been shown to enhance the performance of Transformer models in symbolic music tasks, facilitating the learning of more isotropic embedding representations. It can be applied on top of any tokenization method except those based on embedding pooling.

Chapter 6

Evaluation Methods

The *Evaluation process* for music generation is complex and challenging task, due to the subjective nature of music perception and the multi-dimensional aspects of musical quality. Despite the existence of objective rules for counterpoint, composition and music harmonization, the tasks of defining an objective plane for comparison and evaluation of music pieces is often a subject of disagreement among members of the research community, yet a field that remains an active area for academic exploration.

Evaluation in generative music tasks typically involve both qualitative listening evaluations and quantitative assessments. In this chapter, we will explore certain aspects of each evaluation category, discuss the evaluation process in detail, and suggest a composite metric utilizing metrics from the MusPy library.

6.1 Qualitative Evaluation

Music is a deeply subjective matter and, among individuals, preferences can vary widely. What one person finds enjoyable or musically satisfying in terms of melodic and rhythmic content may not resonate with another. This subjectivity makes it challenging to establish universal criteria for evaluating the quality of generated music.

Despite the above, musical competence is a biologically inherited ability, as mentioned in 4, which makes humans capable of retrieving information and understanding basic concepts of a musical piece, such as melodicty, harmonicity, rhythmic coherence, emotional expression, that can work as empirical evaluation metrics.

Human evaluation provides researchers with the ability to explore more complex aspects of the musical experience, beyond simple quantitative metrics. Humans can intuitively explore in-depth factors such as aesthetic appeal, cultural context, historical background, influences among musical pieces through time and expression styles.

In most related work, human evaluation involves a mixture of trained musicians, composers, musicologists, as well as individuals with varying levels of experience in the field, ranging from casual listeners to music enthusiasts.

In the area of study of generative music, as demonstrated by [8], metrics employed for human evaluation involve:

- fidelity on pitch scale and rhythm regarding the prompt
- correctness, i.e. featuring good note succession and harmony
- coherent diversity, i.e. featuring diverse correct melodies and harmonies
- their overall subjective preference

In order to systematically assess the different dimensions of music quality, it is optimal to break down the evaluation process into specific questions and criteria, as demonstrated by [5]:

- *coherence*—“Is it temporally coherent? Is the rhythm steady? Are there many out-of-context notes?”
- *richness*—“Is it rich and diverse in musical textures? Are there any repetitions and variations? Is it too boring?”
- *arrangement*—“Are the instruments used reasonably? Are the instruments arranged properly?”

6.2 Quantitative Evaluation

As mention in the sections above, quantitative metrics are subject to irregularities

As mentioned beforehand, the objective characteristics of a musical text, easily perceived by humans, are complex to translate in objective rules and to project in a computational plane outside human perception. However, the use of quantitative metrics is necessary to straightforwardly evaluate the performance and creative capacity of both the generative model and the data pre-processing architecture.

Quantitative evaluation metrics, derived from the predicted model hardly lead us to valuable conclusions by themselves. Although, when applied to the generated output and compared to ground truth values, these metrics offer a quantitative basis for assessing the fidelity, accuracy, and overall performance of the generative model.

6.2.1 Loss Functions

Ground truth similarity is utilized in the model training and validation through the *loss functions*.

For sequence-to-sequence models used in sequence generation tasks, loss functions such as the sequence cross-entropy loss are employed. The output generated file is compared with the original file to find out the deviations from the input sequence [15].

This kind of loss function measures the discrepancy between the predicted sequence and the ground truth sequence [15], considering the entire output sequence rather than individual tokens. In this context, the model's weights are updated to ensure that it mimics the exact style and expression of the artist's work in the training set, taking into consideration the accompanying musical composition in the input. Therefore, when producing "irregular" output, the model is penalized for both generating music that does not suit the accompanying input and for lacking coherence, melody, rich context, and a stable rhythmic style.

6.2.2 Tokenization Errors

As proposed in [8], another method to secure that the model outputs coherent and valid results, is to measure the syntax errors in the output. For a more in depth elaboration, Fradet et. al propose that, the model outputs might include token sequences that have a pitch token as the last sequence value, or tokens that "go back in time", tokens that have incorrect successions and do not follow the *pitch*, *velocity*, *duration* pattern. To test this possibility, the authors of the paper propose the application of a syntactic error metric to the output, that can provide several indications about the quality and efficiency of the model training, and whether it can ultimately learn the music representation efficiently and yield coherent results, or not.

6.2.3 Evaluation Metrics

In addition to loss functions and Tokenization Syntax Errors, various evaluation metrics and Python libraries offer valuable tools and provide mathematical representations of certain characteristics of a musical piece that are often perceived intuitively by listeners. By analyzing these metrics, researchers and musicians can gain quantitative insights into various aspects of musical pieces that often emerge from qualitative research.

One such library is MusPy, a Python package designed for symbolic music generation and evaluation. MusPy provides a range of evaluation metrics, including pitch accuracy, rhythm accuracy, and harmonic consistency, allowing for comprehensive assessment of generated music quality. Among recent authors, [5] utilize these metrics to evaluate the output of their model, in comparison to the ground truth. A closer value to that of the ground truth is considered better.

6.2.4 Proposed Evaluation Metric

For the needs of this diploma thesis, we decided to utilize a custom evaluation metric, in order to develop one metric that captures both the efficacy of model training and its ultimate performance. We aim to derive quantitative measures that enable us to evaluate the quality of training and pre-processing, as well as the final outcomes of the model.

We decide to employ the following metrics, using the MusPy library:

- Pitch class entropy: measures the randomness of the distribution of musical notes within a note class, with lower entropy indicating less diversity and higher entropy indicating more diversity.
- Pitch Entropy: measures the randomness of the distribution of musical notes within a note class, with lower entropy indicating less diversity and higher entropy indicating more diversity.
- Scale consistency: the largest pitch-in-scale rate over all major and minor scales

After extracting the aforementioned metrics, from both the ground truth and the predicted output, we calculate their euclidean distance in the following manner:

$$\text{distance} = \sqrt{(\text{or_PCE} - \text{pr_PCE})^2 + (\text{or_PE} - \text{pr_PE})^2 + (\text{or_SC} - \text{pr_SC})^2}$$

By applying this metric to both the ground truth and the predicted outcomes of our model, and calculating the Euclidean distance between them for the test set, we aim to achieve two main objectives:

- Evaluate the Discrepancy between Ground Truth and Predicted Outcomes: By measuring the Euclidean distance 6.2.4 between the ground truth and the generated model outcomes, we gain insights into more qualitative characteristics of the generated melodies, than their similarities to the ground truth. This

analysis enables us to understand how closely the model's predictions align with the original musical compositions, showing any variations or deviations in terms of melodic structure, harmony, and overall musical quality.

- **Determine the Model with Optimal Performance:** By computing the average Euclidean distance across all models, we can identify which model *consistently* produces the most accurate and faithful results compared to the ground truth. This comparative analysis allows us to discern the relative performance of different model configurations and make informed decisions regarding model selection and optimization for future iterations of our study.

Chapter 7

Experiments

The experimental setup for our study involved the use of state-of-the-art deep learning frameworks, including HuggingFace Transformers and PyTorch, for model development and training. For MIDI preprocessing routines such as track merging, custom data augmentation and output format, we utilized libraries such like miditoolkit, and mido. We utilized a MidiTok library ¹ for the tokenization process, which provided us with a comprehensive set of functions and tools essential for encoding MIDI events into sequence format. This encoding was necessary for the pretraining of the transformer model.

We also made use of a computational setup powered by high-performance GPUs on Kaggle to enhance the training pace and enable seamless experimentation with large models and large-scale datasets.

We began by conducting experiments using the architecture and model parameters outlined in our bibliography. We explored a variety of parameters to determine which ones were most suitable for our task and the available data. Eventually, we steered towards the direction of the optimal parameters based on our findings, facilitating further experimentation.

Ultimately, we applied the proposed evaluation method on the generated results of each model to investigate their plausibility and objectivity.

We showcase the evaluation metrics through bar plots, alongside their mean values, to facilitate a comparative analysis of the proposed models. Additionally, we delve into examining the rate at which our model generates audibly pleasing music, that also serves the purpose of harmonization, regarding the Bach Chorales input.

¹<https://miditok.readthedocs.io/en/v3.0.2/>

7.1 Dataset - JSB Chorales

For the needs of this thesis, we will explore the possibilities of Generative Music using the renowned Johann Sebastian Bach Chorales dataset, or else referred to as JSB Chorales. This dataset, comprised of harmonically rich chorales by Bach, serves as a fertile ground for studying generative music systems due to its intricate melodies and well-defined musical structure.

The Bach Chorales, in particular, represent a collection of hymn-like compositions that are a great example of Bach's mastery of harmony and counterpoint. As mentioned in 4, these chorales are of great didactic significance, they often used as teaching tools in music theory education, provide a fertile ground for studying musical structure and compositional techniques. Could there be a more fitting domain than this to impart the rules of musical composition to a model?

This dataset comprises a collection of 372 four-part chorales, each containing melodic lines for soprano, alto, tenor, and bass voices. To prepare the dataset for generative music experiments, we performed preprocessing steps, which will be thoroughly analyzed in the next section, including data cleaning and normalization, to ensure consistency and accuracy in the musical data.

7.2 Data Preprocessing

We began by acquiring the JSB dataset and isolating the 372 four voice chorales, out of 514 of the dataset total chorales. Subsequently, we partitioned the 372 MIDI files into distinct training, validation, and test sets, ensuring that there will be no ensuring that there is no information leakage during the model's training process.

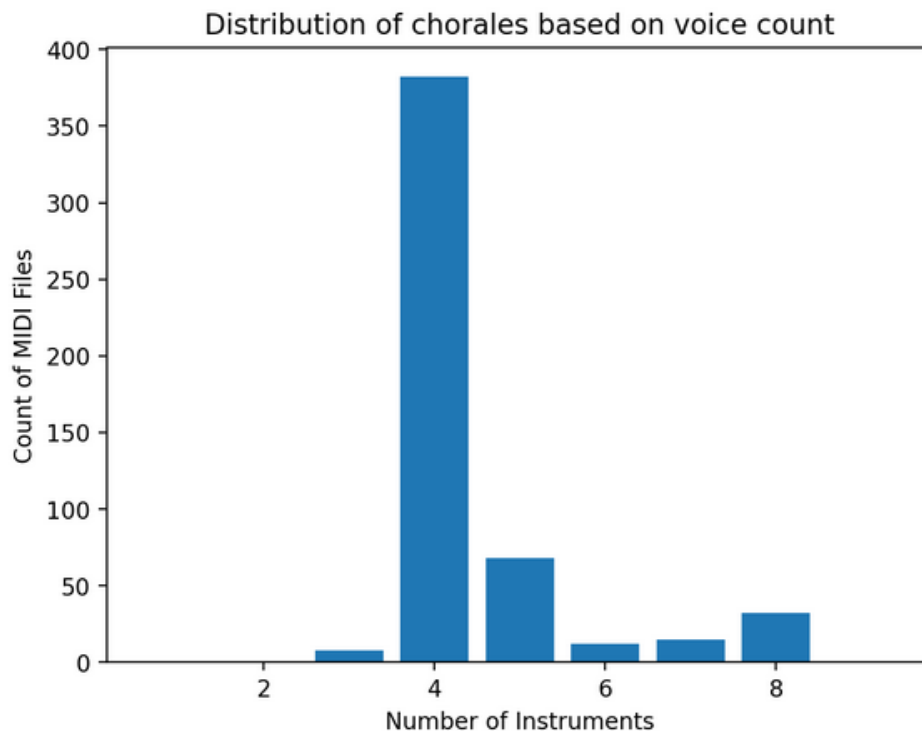


Figure 7.2.1: Distribution of JSB Chorales Dataset

In the subsequent preprocessing steps, we will generate "child files" from the original files, each containing a specific task type. We must ensure that no "child file" becomes separated from its corresponding "parent."

7.2.1 MIDI file formatting

The original MIDI files contained four tracks, each representing a distinct musical voice: Basso, Tenor, Alto, and Soprano. The goal was to transform these files into a format conducive to training a Seq2Seq model like the T5 transformer.

This translates to merging neighboring tracks based on specific task requirements to create MIDI files with only two tracks. The MIDI file that contains only two tracks, will later be tokenized effectively into I/O sequences.

To facilitate the tokenization process and streamline the implementation of the "Add upper harmonic voice" and "Add lower harmonic voice" tasks, we structured the input-output format directly within the MIDI files. This translates to merging neighboring tracks based on specific task requirements to create MIDI files with only two tracks.

For each chorale file, we generated 12 modifications, split evenly between the two tasks. For the "Add upper harmonic sequence" task, the modifications were as follows:

- Adding a Tenor voice to a piece that contains only Basso.
- Adding an Alto voice to a piece that contains only Tenor.
- Adding an Alto voice to a piece that contains both Tenor and Basso (combined in one track).
- Adding a Soprano voice to a piece that contains only Alto.
- Adding a Soprano voice to a piece that contains both Alto and Tenor (combined in one track).
- Adding a Soprano voice to a piece that contains Alto, Tenor, and Basso (combined in one track).

Similarly, modifications were made for the "Add lower harmonic sequence" task.

These modified MIDI files were then saved in the output of the notebook, with each set (training, validation, test) stored in its respective folder for ease of access and organization.

Figure 7.2.2 consists of two musical score snippets. Snippet (a) shows a four-voice Bach chorale with four staves labeled 'Piano, Instrument 1' through 'Piano, Instrument 4'. The tempo is marked as $J = 92$. Snippet (b) shows a modified file with two staves labeled 'Piano, Instrument 0' and 'Piano, Instrument 1'. The tempo is marked as $J = 96$.

(a) Original overview of a four-voice Bach chorale.

(b) Modified file, with isolated soprano voice.

Figure 7.2.2: Visual representation of the modifications that the MIDI files underwent during the data preprocessing part described in 7.2.1.

7.2.2 Tokenization

Following the previous modifications, we proceed into the tokenization process of the modified MIDI files.

Utilizing the tokenizers and routines of the miditok library, described in 5.1, convert these musical compositions into structured sequences of tokens.

For the needs of the initial experiments of our idea, we make use of the TSD tokenizer, described in 5.2.3.

The tokenizer parameters define various aspects of the tokenization process for MIDI files. They include specifications such as the pitch range considered, beat resolution, number of velocity bins, and the inclusion of special tokens for padding, sequence start, end, and masking.

Miditok provides a variety of tokenizer configuration parameters, that control the incorporation of musical elements like chords, rests, tempos, time signatures, program changes, and pitch bends. Some of these parameters, such as pitch range and velocity bins, refer to downsampling of the information contained in MIDI note-events, allowing them to be captured by fewer tokens.

The tokenizer can adapt to handle multi-voice MIDI files and offers options for organizing token streams, whether as a single stream or separate streams for each voice. We make use of that advantage to extract the I/O format of the tokenized sequences described earlier.

7.2.3 Prefix tokens

To achieve the task specificity during training, we introduced two task specific tokens to the tokenizer: the "Add Upper" token and the "Add Lower" token. These tokens signify the augmentation of upper and lower harmonic voices, respectively. This strategic addition, utilized as a *prefix token*, guides the T5 transformer, facilitating its comprehension and execution of the specific music generation tasks outlined in our experimental framework.

7.2.4 Data Augmentation

The tokenizer proceeds with a data augmentation step to enrich the dataset further. This augmentation involves manipulating pitch, velocity, and duration tokens to introduce variations in the musical sequences. Specifically, pitch augmentation is applied with an offset of 2 tokens, resulting in pitch transposition by 2 octaves up and down. Velocity and duration augmentation are applied with offsets of 1 token each. By systematically adjusting these parameters, the tokenizer enhances the diversity of the dataset, enabling the model to learn and generalize better across different musical contexts.

7.2.5 Byte Pair Encoding

Additionally, employing byte pair encoding (BPE) further enhances the efficiency and robustness of our tokenization method, ensuring optimal performance throughout the training and generation phases [8].

As described in 5.2.6, Byte Pair Encoding needs to be learned in the entirety of the data (training, validation, and test sets, respectively). This ensures consistency and coherence in tokenization across all datasets, facilitating uniform representation of musical sequences for the model. A new, augmented vocabulary is created, that consists of the most frequent subsequences.

Later, the training, validation and test datasets are encoded with the new BPE vocabulary, and, thus, compressed in a smaller size that efficiently represents the information needed to train the model effectively.

Length Standardization

Before continuing with the training and data loading process, a length criterion is applied to the tokenized data. This criterion ensures that both input and output tokens conform to a standardized format, simplifying processing within the scope of the study. Standardizing the length is crucial because T5 utilizes standard input-output phrases, necessitating uniformity in the data.

Preparing for training

Later, the data sequences are loaded into a DataLoader, with the appropriate prefix added as the first token of each sequence. This prefix serves as a cue to the model regarding the type of task it is about to be trained on. Furthermore, right padding is applied to ensure consistent sequence lengths across the dataset.

7.3 Results

7.3.1 Parameter Selection

In the initial phase of our experiments, we sought to explore the impact of dropout rate and Byte Pair Encoding (BPE) vocabulary size on model training with our dataset. We conducted training sessions with six models using standard parameters, including $n_{layers} = 8$ and $n_{heads} = 16$, while varying the parameters: vocabulary size = 2500, 5000 and dropout rates = 0.2, 0.4, 0.5.

Table 7.1: Parameter settings for initial experiments

Tokenizer	Vocab Size	Num Layers	Num Attention Heads	Dropout Rate
TSD	2500	8	16	0.2
	5000			0.4

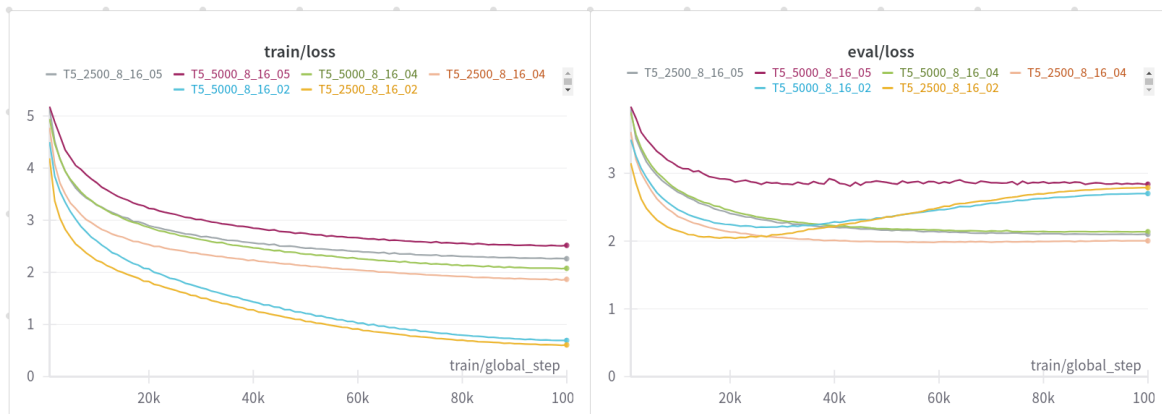


Figure 7.3.1: Loss functions of first experiments 7.1. We initially run experiments with model parameters according to the bibliography [8]. We chose a rather wide range of values, in order to explore the direction of further experimenting.

Analysis of the results, depicted in 7.3.1, revealed notable trends. Models trained with a dropout rate of 0.2 exhibited rapid overfitting on our data, indicating insufficient regularization. Conversely, models with a dropout rate of 0.5 demonstrated signs of underfitting, suggesting excessive regularization. Consequently, we determined that the optimal dropout rate lies around 0.4, striking a balance between overfitting and underfitting.

Regarding vocabulary size, observations indicated that models with a smaller vocabulary size (2500) achieved lower loss values compared to those with a larger size (5000). This phenomenon is justifiable, as a larger vocabulary on a relatively small dataset may lead to valuable information being hidden behind pair encodings. Consequently, the model may struggle to effectively capture the underlying patterns of the data.

With the groundwork of our experimental process established, we now narrow the scope of parameter values to delve deeper into our investigation. In this phase, we aim to assess how the model captures information under varying conditions, focusing on specific parameter configurations.

For this segment of experimentation, we select the parameter values described in 7.2

By systematically varying these parameters, we seek to gain insights into how different architectural configurations impact the model’s performance in capturing and understanding the underlying patterns of the data. This focused approach allows us to refine our understanding and optimize the model’s architecture for enhanced performance and efficacy in subsequent phases of our study.

After running the experiments with the aforementioned parameters, we notice that, in terms of loss, the models seem to be reaching similar results. Models with lower dropout rate seem to catch up on information

Table 7.2: Further investigation on mentioned parameters to investigate improvements.

Tokenizer	Vocabulary Size	Model Layers	Model Attention Heads	Dropout Rate
TSD	1500	8	12	0.3
	2500	10	16	0.4

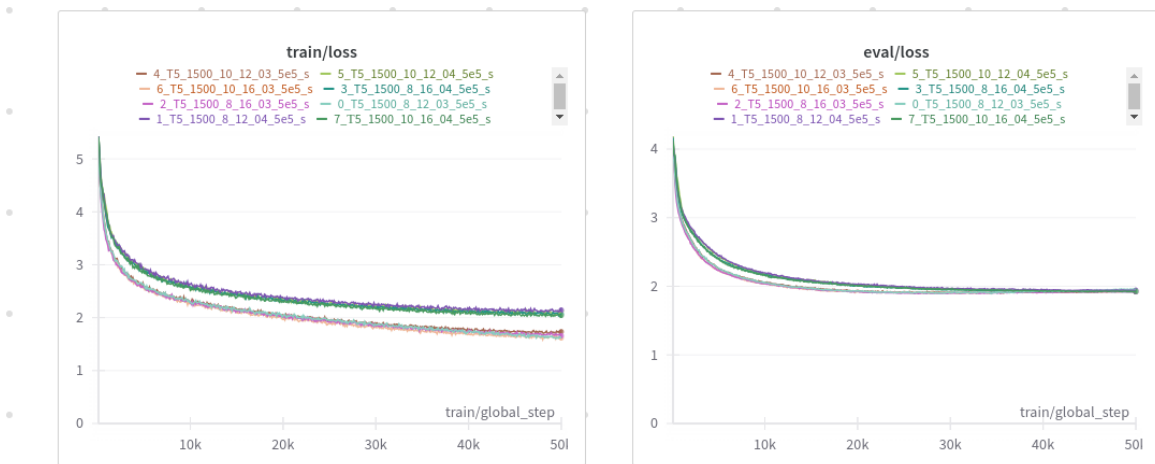


Figure 7.3.2: Loss curve of models trained with TSD tokenizer and vocabulary size = 1500. The rest of the parameters are described in 7.2

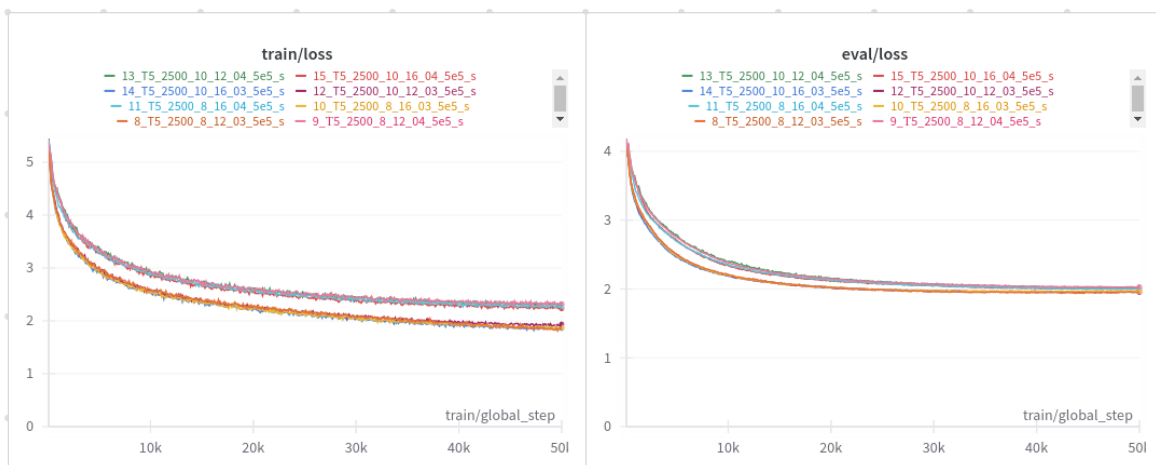


Figure 7.3.3: Loss curve of models trained with TSD tokenizer and vocabulary size = 2500. The rest of the parameters are described in 7.2

and patterns in the desired output more easily, displaying steeper curves.

7.3.2 Test Set Generation

In order to test the proposed model, we decided to utilize the model's ability to generate output with **temperature**. Adding temperature when generating output from a T5 model is a technique used to control the randomness and diversity of the generated text. The temperature parameter adjusts the softmax function during sampling, affecting the distribution of probabilities assigned to each token in the vocabulary. This addition provides greater flexibility and control over the text generation process, allowing for the model to display more creativity and avoid the "most probable next token" loop.

For the initial experimentation phase, we chose to produce output samples using a fixed temperature of 0.2. Subsequently, during the testing phase, we varied the temperature values across a range from 0.05 to 0.35 to explore the impact of this variation on the final auditory output

7.3.3 Applying proposed Evaluation Metric

To obtain a comprehensive overview of our models' performance, we will utilize the metric defined earlier in 6.2.4.

To gain a comprehensive understanding of the overall outcome, we conduct a comparative analysis at two levels. Firstly, we assess whether the inserted voice aligns with the given melody, providing a global overview of the total result. Additionally, we delve into more specific insights, evaluating aspects such as the richness of the additional voice, its adherence to a consistent scale, and its standalone performance. This structured approach allows us to examine both the holistic result and the individual characteristics of the generated voice.

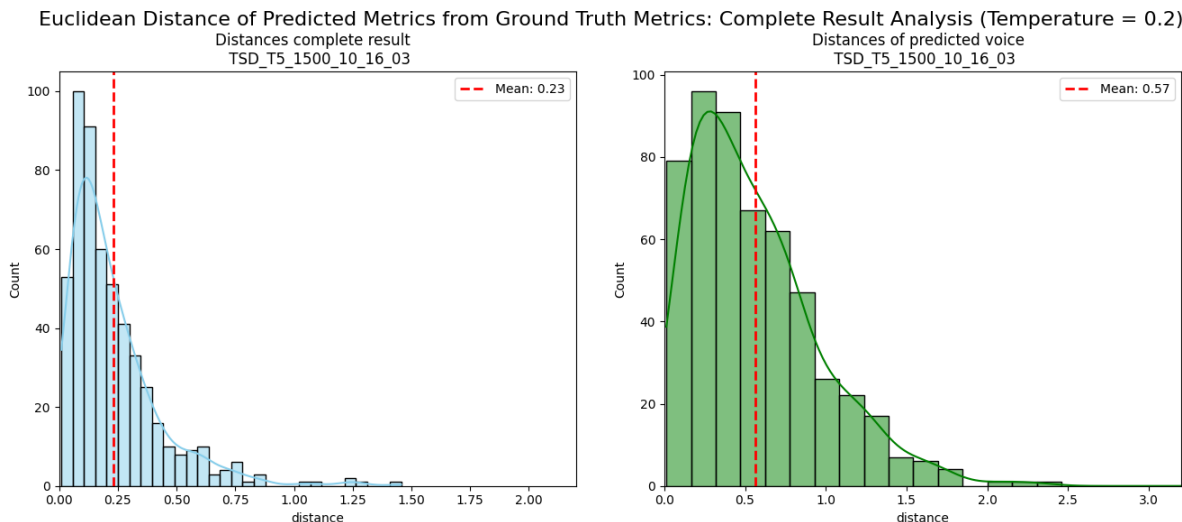


Figure 7.3.4: Comparison of distance distributions between metrics measured on generated outputs and ground truth across the entire musical piece (left) and specifically on the generated voice (right).

We visualize these findings, we create bar plots illustrating the distribution of distances from the samples of the test set generated by each model 7.3.4. This approach allows for easy comparison between different models 7.3.5 7.3.6.

As evidenced by the loss curves shows, the models tend to *converge* and exhibit similar results. However, the disparities become more apparent when evaluating the quality of the generated voice in the comparison to the ground truth. In this, the differences are more pronounced, which is anticipated since the first compares files that have parts of common context (the given voices).

Similarly, evaluating the results of additional voice generation, this time with variable temperature, we observe that the mean values of the distances do not change, the distances distributions are of similar shape, and therefore we conclude that varying the temperature during generation does not significantly alter the overall evaluation. This suggests that, within the selected range, the temperature parameter may have limited impact on the quality of the generated voices. Our findings also indicate that a temperature value of 0.2 consistently produces satisfactory results.

Through this analysis, the model trained on vocabulary size equal to 1500, $n_{layers} = 10$, $n_{heads} = 16$ and $dropout_{rate} = 0.3$, seems to produce the most prominent results.

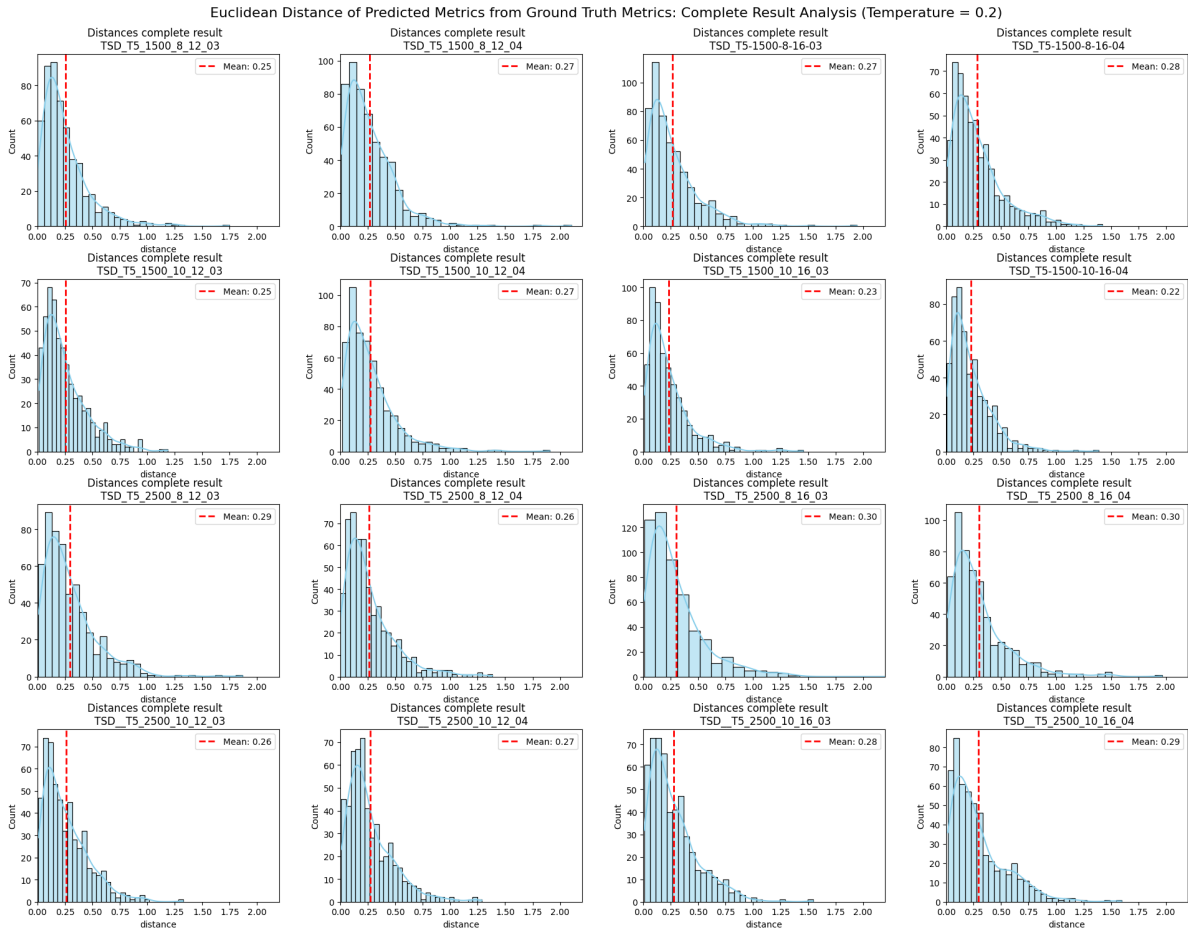


Figure 7.3.5: Analysis of 16 models, using TSD tokenizer and parameters described on 7.2: Distribution of metric distances between generated outputs and ground truth, calculated across the entirety of musical pieces in the test set, with a temperature setting at 0.2

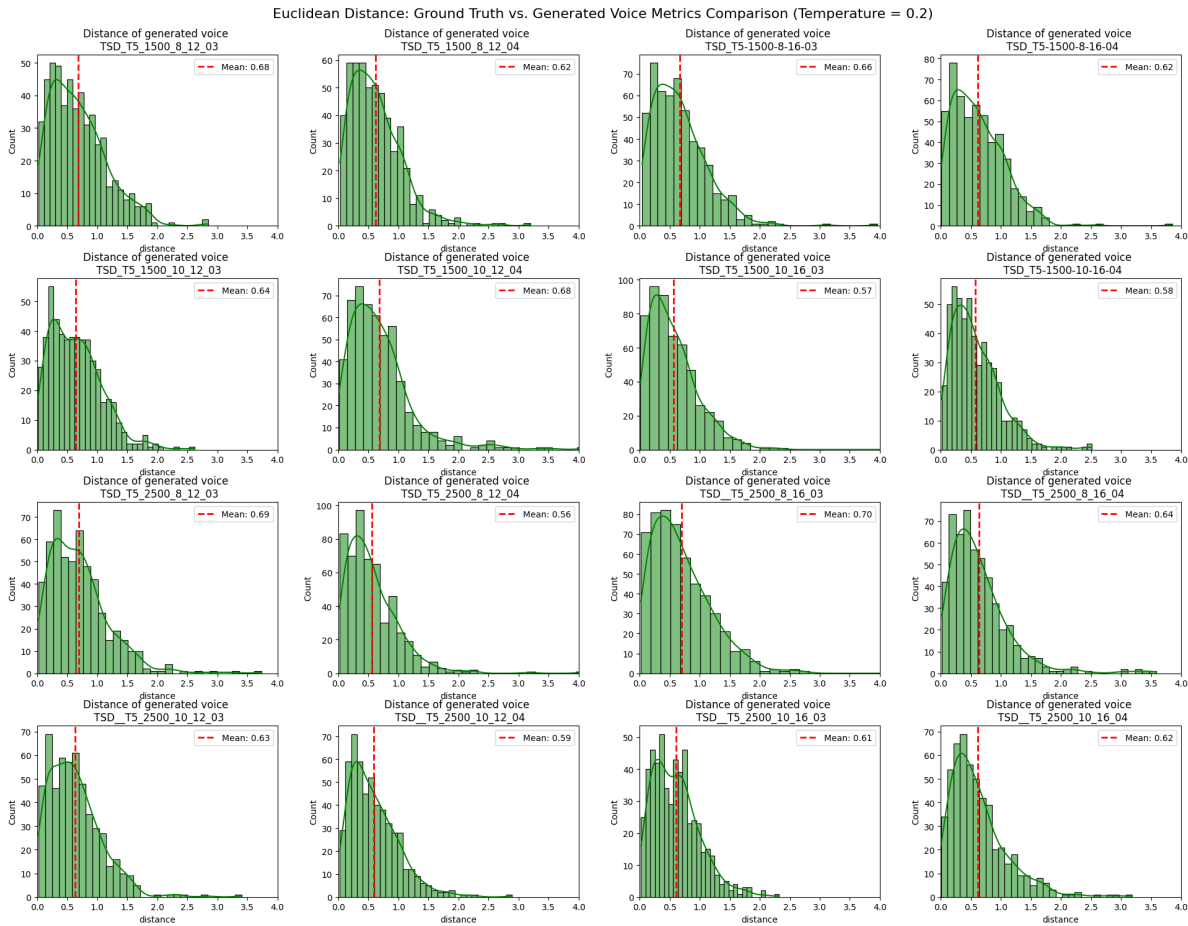


Figure 7.3.6: Analysis of 16 models using TSD tokenizer and parameters described on 7.2: Distribution of metric distances between generated outputs and ground truth, calculated specifically on the generated voice in the test set, with temperature settings ranging from 0.05 to 0.35.

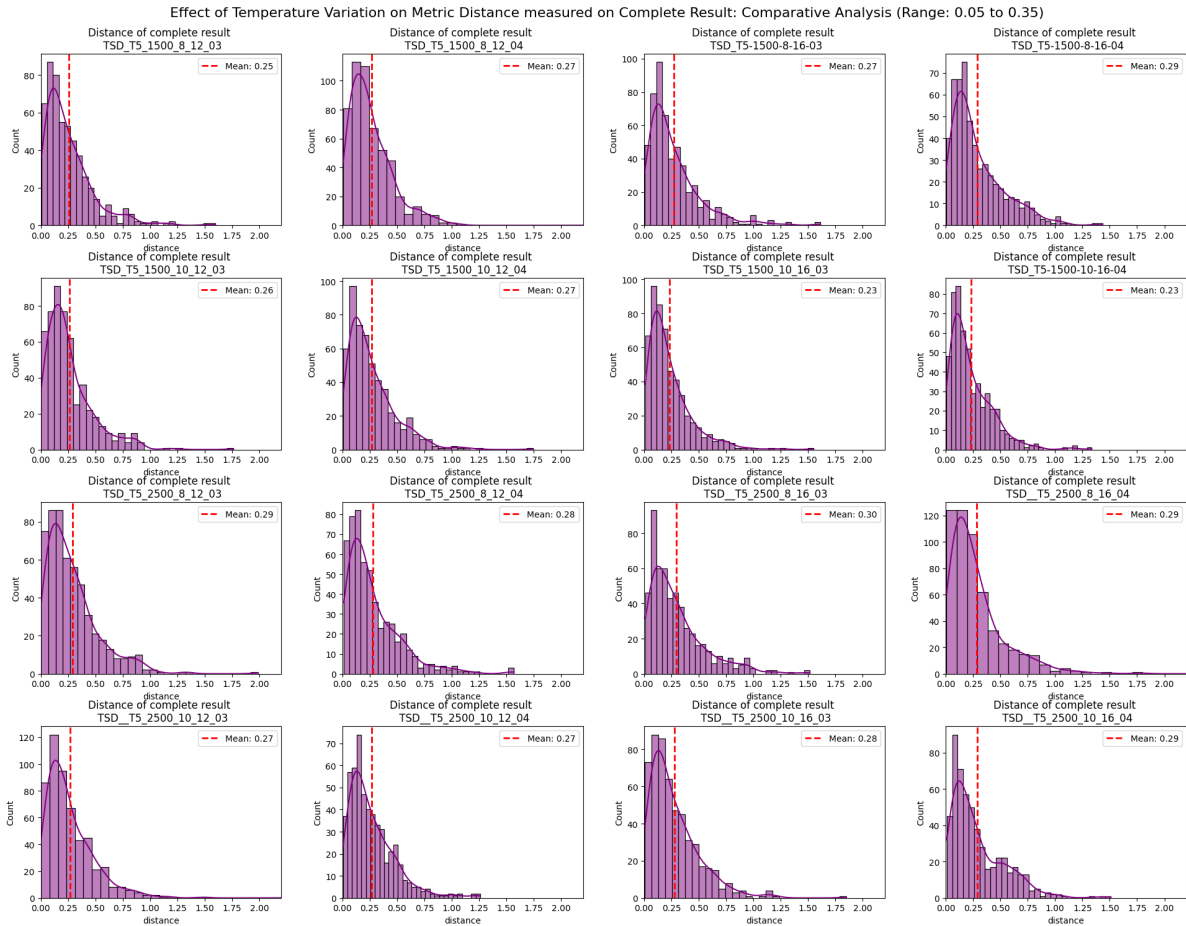


Figure 7.3.7: Analysis of 16 models using TSD tokenizer and parameters described on 7.2: Distribution of metric distances between generated outputs and ground truth, calculated across the entirety of musical pieces in the test set, with a temperature setting at 0.2.

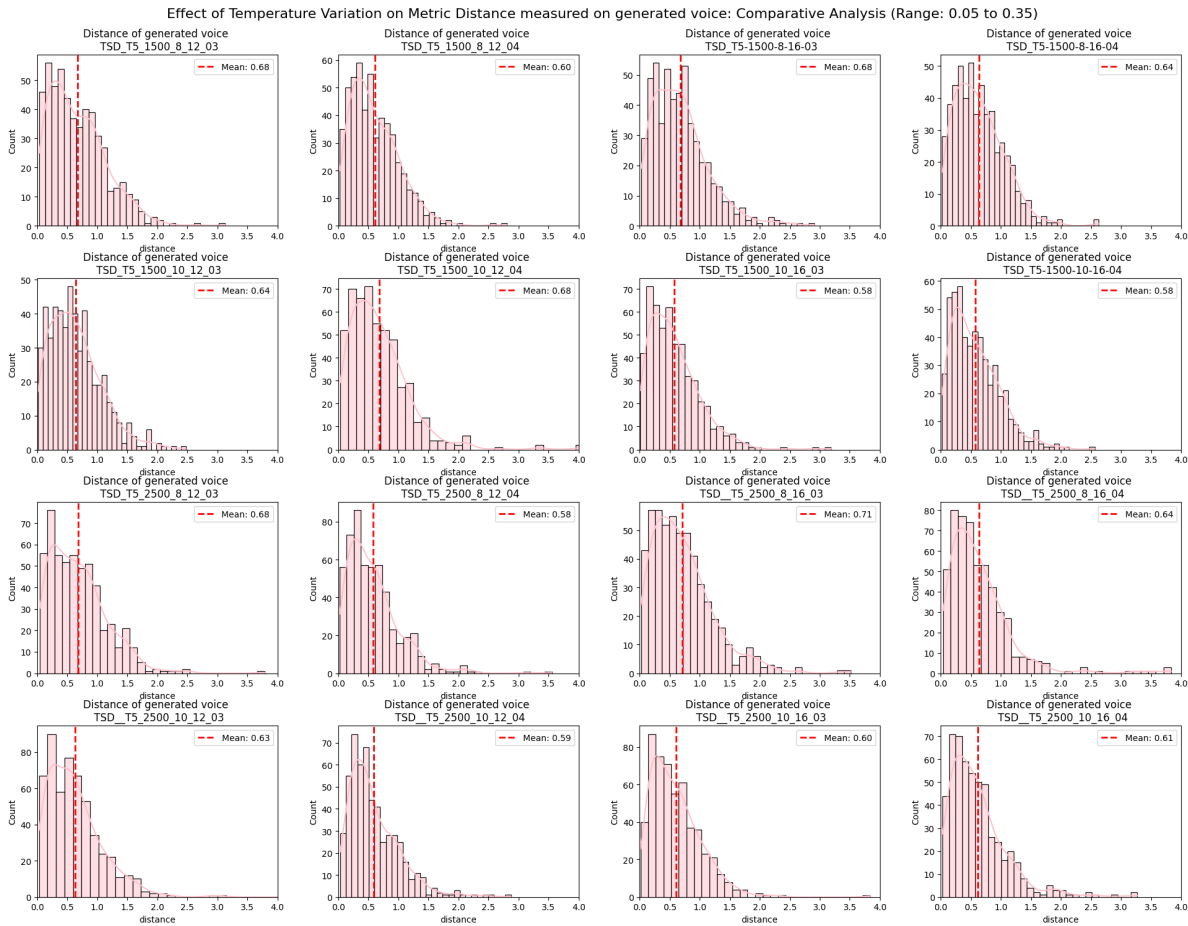


Figure 7.3.8: Analysis of 16 models using TSD tokenizer and parameters described on 7.2: Distribution of metric distances between generated outputs and ground truth, calculated specifically on the generated voice in the test set, with temperature settings ranging from 0.05 to 0.35.

7.3.4 Further Experiments: REMI, Structured & Tokenizers

Due to computational limitations and constraints in GPU power, we expanded the scope of our experiments to explore models with fewer parameters, smaller sizes, and different tokenizers. Implementing numerous experiments, we utilized a model with 4 layers, 12 attention heads, a dropout rate of 0.3, and various tokenizers, each with a vocabulary size of 1000.

Comparing evaluation metrics of two small models trained with REMI tokenizer and custom augmentation

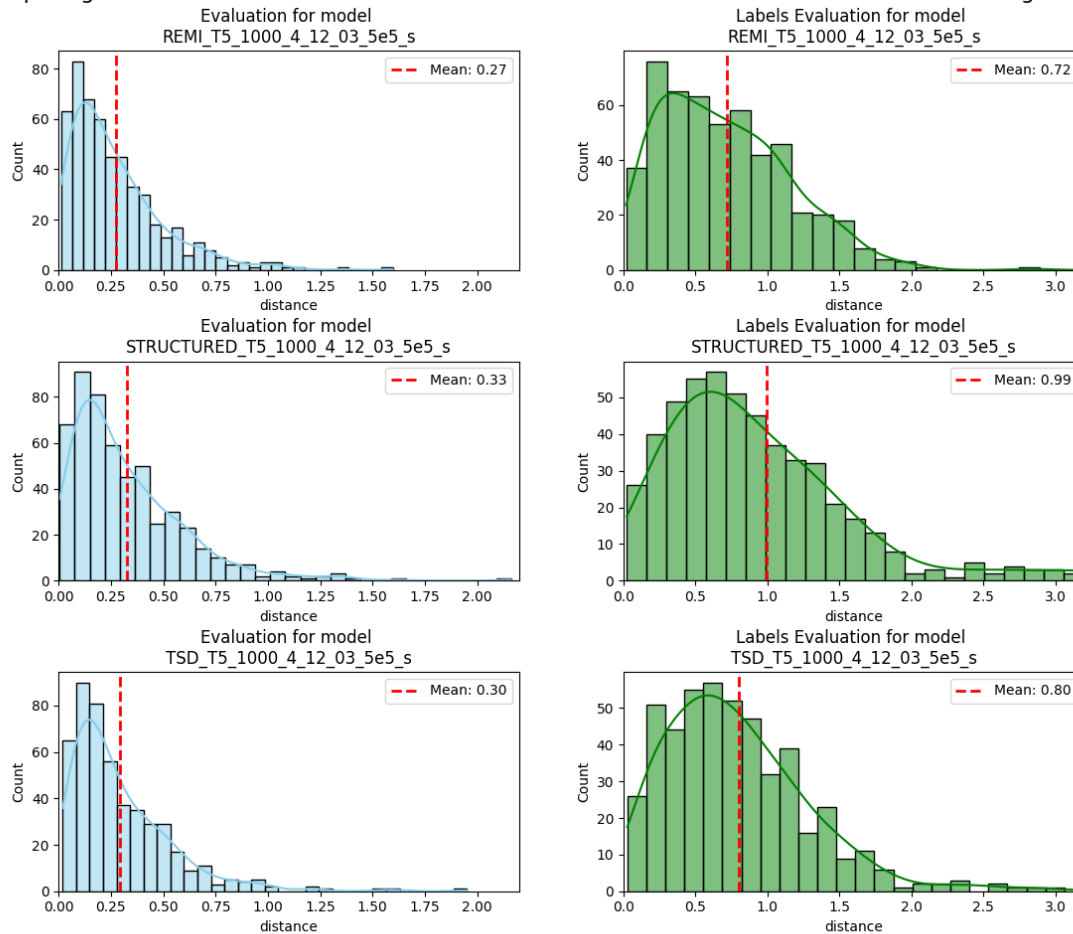


Figure 7.3.9: Small models trained on data tokenized with REMI, Structured & TSD tokenizers: Comparison of distance distributions between metrics measured on generated outputs and ground truth across the entire musical piece (left) and specifically on the generated voice (right).

We observed that the TSD tokenizer and REMI tokenizer appear to surpass the Structured Tokenizer in terms of our metrics. However, to conduct experiments that comprehensively illustrate the advantages and disadvantages of each, we need to undertake a more exhaustive experimentation process, additionally, ensuring that our resources are adequate to handle this expanded experimentation is imperative.

Chapter 8

Conclusion

8.1 Discussion

In conclusion, this thesis explored a music generation task for the enrichment of Polyphonic Bach Chorales through the development and evaluation of a Seq2Seq transformer-based model. Through careful preprocessing, including, tokenization, data augmentation, standardization and more, we laid the foundation for a modeling of how the careful handling of data is of great importance. The process involving transforming symbolic music content into sequential form could potentially constitute a distinct field in itself.

Our experimental journey included a thorough examination of various model configurations, including parameter variations such as dropout rates, BPE vocabulary sizes, and temperature values. The results revealed insights into the impact of these parameters on model performance, with notable observations regarding convergence, overfitting, and the fidelity of generated voices.

Our comparative analyses provided subtle understandings of the model's behavior, clarifying the trade-offs and optimal configurations for achieving desired outcomes. Notably, the examination of loss curves highlighted the convergence of models and underscored the importance of fine-tuning parameters to balance performance and generalization.

We devoted a proportional amount of attention and resources to examine Symbolic Music Representations, including their inherent structures and characteristics, alongside an exploration of the relevant tools and frameworks facilitating interdisciplinary research. This investigation explores symbolic music encoding and decoding processes, as well as the development of innovative techniques for effectively leveraging symbolic music data in machine learning applications.

Additionally, we thoroughly review the evaluation techniques applied to generative music. This includes a detailed exploration of the frameworks and libraries utilized for both quantitative and qualitative assessment of musical compositions. We proposed an innovative evaluation approach that seems to converge with that of human perception in musical terms. This evaluation shed light into the differences and qualities of the developed models, and revealed the significance of temperature selection in influencing the quality and consistency of outputs, with a temperature value of 0.2 emerging as a reliable choice.

Overall, this thesis represents a significant step towards advancing the capabilities of AI-driven music generation and underscores the potential for transformative innovation in this interdisciplinary field.

8.2 Future Work

Through systematic experimentation and rigorous analysis, this research contributes valuable insights to the field of music generation and transformer-based models.

Moving forward for further exploration, we plan to leverage masked pretraining and fine-tuning techniques to further enhance the performance and adaptability of our model to specific musical tasks. Along with that, we would include exploring additional architectural variations, and extending the application of voice augmentation to diverse musical domains.

In addition to our investigation into harmonic understanding within the proposed model, our future work will explore how evaluation methods outlined in [4] can inform the development of a **custom loss function**. This aims to establish a method for penalizing the model based on criteria more closely aligned with *harmonic relevance* rather than traditional cross-entropy distance metrics. By utilizing insights from these evaluation techniques, we aim to enhance the model’s ability to capture and adhere to harmonic principles, thereby further refining the quality and coherence of its generated musical compositions.

Given that the model displays an ability for understanding specific harmonic rules inherent in the given melody, we aim to integrate explainability applications, allowing for a deeper understanding of the underlying processes and decisions made during music generation.

An explainable model will provide us the opportunity to explore the use of counterfactual explanations or attacks for further experimentation on conditional generation. By employing counterfactuals, we aim to investigate how changes in specific features or conditions can influence the model’s output, thereby gaining deeper insights into its creative abilities.

On the context of generative symbolic music, investigating different tokenization methods and models, including both large language models (LLMs) and graph neural networks (GNNs), will provide opportunities to explore diverse avenues for encoding and processing symbolic music data.

Although, exploring the intricacies of the tokenization process not only enhances our understanding of symbolic music representation but also offers opportunities to model information and *time dynamics* effectively. This may allow us to develop insights into how musical information is structured and conveyed over time, thereby contributing to efforts that adopt the reverse path—converting audio signals into tokens. Understanding the tokenization process from this perspective enables us to bridge the gap between symbolic and audio-based music representations, facilitating advancements in both domains and fostering interdisciplinary collaborations in the field of music generation and synthesis [17].

Lastly, we plan to explore different evaluation methods, such as in [4] to assess the quality, coherence, and expressiveness of the generated musical outputs comprehensively. These future endeavors will contribute to advancing the capabilities and understanding of AI-driven music generation systems.

Chapter 9

Bibliography

- [1] Bahdanau, D., Cho, K., and Bengio, Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: [1409.0473 \[cs.CL\]](#).
- [2] Bengio, Y., Ducharme, R., and Vincent, P. “A neural probabilistic language model”. In: *Advances in neural information processing systems* 13 (2000).
- [3] contributors, W. *Natural language processing*. en. Jan. 2024. URL:
- [4] Dervakos, E., Filandrianos, G., and Stamou, G. “Heuristics for Evaluation of AI Generated Music”. In: *2020 25th International Conference on Pattern Recognition (ICPR)* (2021), pp. 9164–9171. URL:
- [5] Dong, H.-W. et al. *Multitrack Music Transformer*. 2023. arXiv: [2207.06983 \[cs.SD\]](#).
- [6] Feng, W. et al. “Audio visual speech recognition with multimodal recurrent neural networks”. In: May 2017, pp. 681–688. DOI: [10.1109/IJCNN.2017.7965918](#).
- [7] Fradet, N. et al. “MidiTok: A Python package for MIDI file tokenization”. In: *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*. 2021. URL:
- [8] Fradet, N. et al. *Byte Pair Encoding for Symbolic Music*. 2023. arXiv: [2301.11975 \[cs.LG\]](#).
- [9] Fradet, N. et al. *miditok: A Python package for MIDI file tokenization*. 2023. arXiv: [2310.17202 \[cs.LG\]](#).
- [10] Ghojogh, B. and Ghodsi, A. “Attention mechanism, transformers, BERT, and GPT: tutorial and survey”. In: (2020).
- [11] Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.
- [12] Hadjeres, G. and Crestel, L. *The Piano Inpainting Application*. 2021. arXiv: [2107.05944](#).
- [13] Harsha, A. *The Ultimate Showdown: RNN vs LSTM vs GRU – Which is the Best?* [Accessed: (2/2/2024)]. 2023.
- [14] Huang, Y.-S. and Yang, Y.-H. *Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions*. 2020. arXiv: [2002.00212](#).
- [15] Keerti, G. et al. *Attentional networks for music generation*. 2020. arXiv: [2002.03854 \[eess.AS\]](#).
- [16] Khurana, D. et al. “Natural language processing: state of the art, current trends and challenges”. In: *Multimedia Tools and Applications* 82.3 (July 2022), pp. 3713–3744. ISSN: 1573-7721. DOI: [10.1007/s11042-022-13428-4](#). URL:
- [17] Lyberatos, V. et al. “Perceptual Musical Features for Interpretable Audio Tagging”. In: *ArXiv abs/2312.11234* (2023). URL:
- [18] Müller, D., Soto-Rey, I., and Kramer, F. *Towards a Guideline for Evaluation Metrics in Medical Image Segmentation*. 2022. arXiv: [2202.05273 \[eess.IV\]](#).
- [19] Oore, S. et al. *This Time with Feeling: Learning Expressive Musical Performance*. 2018. arXiv: [1808.03715](#).
- [20] Pouyanfar, S. et al. “A Survey on Deep Learning: Algorithms, Techniques, and Applications”. In: *ACM Comput. Surv.* 51.5 (Sept. 2018). ISSN: 0360-0300. DOI: [10.1145/3234150](#). URL:
- [21] Salehinejad, H. et al. *Recent Advances in Recurrent Neural Networks*. 2018. arXiv: [1801.01078 \[cs.NE\]](#).
- [22] Sharkawy, A.-N. “Principle of Neural Network and Its Main Types: Review”. In: *Journal of Advances in Applied Computational Mathematics* 7 (Aug. 2020), pp. 8–19. DOI: [10.15377/2409-5761.2020.07.2](#).

- [23] Sharma, S., Sharma, S., and Athaiya, A. “Activation functions in neural networks”. In: *International Journal of Engineering Applied Sciences and Technology* 04.12 (2020), pp. 310–316. DOI: [10.33564/ijeast.2020.v04i12.054](https://doi.org/10.33564/ijeast.2020.v04i12.054).
- [24] *Symbolic Format: MIDI*.
- [25] Terven, J. et al. *Loss Functions and Metrics in Deep Learning*. 2023. arXiv: [2307.02694](https://arxiv.org/abs/2307.02694) [[cs.LG](#)].
- [26] *The lakh MIDI dataset v0.1*. URL:
- [27] Vaswani, A. et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [28] Weiß, C. “Computational Methods for Tonality-Based Style Analysis of Classical Music Audio Recordings”. PhD thesis. Apr. 2017.
- [29] Wikipedia. 2024.
- [30] Wikipedia. *Harmonic series (music)* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 18-February-2024]. 2024.
- [31] Wikipedia. *Pythagorean tuning* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 18-February-2024]. 2024.
- [32] Yu, T. and Zhu, H. *Hyper-Parameter Optimization: A Review of Algorithms and Applications*. 2020. arXiv: [2003.05689](https://arxiv.org/abs/2003.05689) [[cs.LG](#)].