CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A Chalmers University of Technology Master's thesis template for LaTeX

Investigation of Curriculum Learning in Deep Generative Modelling Using Western Classical Music

Master's thesis in Computer science and engineering

Qi Chen
Gritta Joshy

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis Final report 2025

A Chalmers University of Technology Master's thesis template for LaTeX

Investigation of Curriculum Learning in Deep Generative Modelling Using Western Classical Music

Qi Chen
Gritta Joshy

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

A Chalmers University of Technology Master's thesis template for LaTeX
Investigation of Curriculum Learning in Deep Generative Modelling Using Western Classical Music
Qi CHen
Gritta Joshy

Supervisor: Kivanc Tatar, Data Science and AI
Stefano Sarao Mannelli, Data Science and AI
Examiner: Dag Wedelin, Data Science and AI

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

A Chalmers University of Technology Master's thesis template for LaTeX
Investigation of Curriculum Learning in Deep Generative Modelling Using Western Classical Music
Gritta Joshy
Qi Chen
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Current deep learning approaches for symbolic music generation typically train on randomly ordered musical sequences, which can hinder the development of coherent musical structure and effective learning of long-term dependencies. While curriculum learning has demonstrated significant benefits in natural language processing and computer vision by progressively introducing task complexity, its application to symbolic music generation remains largely unexplored. The hierarchical nature of Western classical music, with structures ranging from simple motifs to complex compositions, makes it an ideal candidate for progressive learning strategies.

This thesis investigates the effectiveness of curriculum learning strategies for symbolic music generation using the Music Transformer architecture. A loss-based complexity ranking system is implemented to order musical sequences from simple to complex, combined with progressive data exposure schedules. Our experimental framework compares baseline training against three curriculum learning variants (CL-60%, CL-80%, and CL-60% with learning rate adaptation) using MAESTRO dataset of Western classical piano compositions.

While curriculum models demonstrated faster early convergence, they produced broader loss distributions with higher variance compared to the baseline's concentrated performance. The CL-60% LR variant emerged as a good performer, achieving superior results in multiple musical features including polyphony, qualified note ratio, and rhythmic structure. Importantly, curriculum learning preserved generalization capability while offering enhanced musical expressiveness.

Although curriculum models did not significantly outperform the baseline in loss metrics, they generated outputs that were more harmonically rich, structurally coherent, and aligned with real music characteristics. These findings demonstrate that curriculum learning offers valuable trade-offs in symbolic music generation, producing more musically compelling outputs when properly designed. This work establishes curriculum learning as a promising training paradigm for music generation and highlights the importance of moving beyond loss-based evaluation toward music-informed assessment metrics. All code and experiments are available at: `Curriculum_learning` [1]

Keywords: curriculum learning, symbolic music generation, transformers, deep learning, MIDI, Western classical music

[1] `https://github.com/Lynn00Chen/Curriculum_learning`

Acknowledgements

We would like to express our sincere gratitude to our supervisors, Kvanç Tatar and Stefano Sarao Mannelli, from the Department of Data Science and AI at Chalmers University of Technology, for their invaluable guidance, insightful feedback, and unwavering support throughout the course of our thesis work.

Their expertise in Machine Learning and Musical AI was instrumental in shaping our research direction and deepening our understanding of curriculum learning and symbolic music generation. We are especially thankful for their encouragement and patience as we navigated the technical and artistic challenges of this project.

We would also like to thank the Chalmers faculty and staff for providing an environment that fostered learning, experimentation, and collaboration.

Lastly, we are grateful to our friends and families for their continuous encouragement and support during this academic journey.

Qi Chen, Gritta Joshy, Gothenburg, 2025-06-16

Contents

List of Figures

List of Tables

1

Introduction

Imagine a young pianist learning to play Beethovens Moonlight Sonata. They don't start by playing the entire piece at once. Instead, they begin with simple scales, progress to basic chords, and then try to tackle more complex passages. This structured approach allows them to build a solid foundation before mastering the complex harmonies and rhythms of the full composition. This is the core of curriculum learning — a training strategy that mimics how humans learn by gradually increasing the complexity of tasks.

Curriculum learning has shown promise in improving both training efficiency and model generalization in deep learning. While it has been applied successfully in domains like natural language processing[1] and computer vision[2], its application to symbolic music generation remains largely unexplored. Music, particularly Western classical music, is hierarchical in nature, with structures ranging from short motifs to full compositions. This structured nature makes it an ideal candidate for progressive learning strategies like curriculum learning.

Traditional training approaches introduce models to randomized musical sequences, leading to challenges in capturing long-term dependencies and maintaining coherent structure. By progressively increasing task difficulty, curriculum learning could enable models to learn simpler musical patterns first, before dealing with more complex harmonic and structural relationships. This thesis investigates whether a curriculum-based training approach can improve the ability of generative models to produce coherent and structured symbolic music.

1.1 Context

Deep learning has significantly advanced symbolic music generation, with models such as Museformer[3] and Music Transformer [4] demonstrating the ability to generate structured compositions. However, existing training approaches still face fundamental challenges such as lack of a structured way of learning. Standard training methods present musical sequences in a random order, making it difficult for models to learn basic musical structures before progressing to complex ones. Another challenge is the computational inefficiency when training on complex sequences from the beginning as it may lead to slow convergence and minimal learning dynamics.

Curriculum learning offers a potential solution to these issues. Inspired by human

learning processes, where musicians start with simple exercises before advancing to more complex compositions, this approach aims to guide model training in a structured and hierarchical manner. Recent research, such as [5], has demonstrated how effective curriculum learning is in improving the model's convergence and generalization, but its impact on music generation model is not yet explored.

Recent advancements like JEN-1 [6] have improved generation quality through multi-task learning and latent diffusion models. However, these approaches focus mainly on architectural innovations rather than structuring the learning process itself. In contrast, a curriculum learning-based training strategy could allow models to develop an understanding of musical complexity in a more systematic manner.

Given the gap in research on curriculum learning for symbolic music generation, this thesis explores how a well-structured approach like curriculum learning can improve training efficiency, musical coherence, and long-term structure generation in deep generative models.

1.2 Research Questions

This research aims to investigate the following primary questions:

1. How can curriculum learning strategies be effectively designed and implemented for symbolic music generation to progressively introduce musical complexity?
2. What impact does curriculum learning have on model training efficiency and convergence when applied to music generation models?
3. To what extent does curriculum learning improve the quality and coherence of generated musical compositions compared to standard training approaches?

Additionally, a secondary research question explores:
4. Which architectural variations could enhance the baseline Music Transformer model while maintaining computational efficiency?

1.3 Problem Statement

Problem: Current generative models struggle to maintain coherence over longer compositions and fail to utilize hierarchical structures effectively. Most symbolic music generation studies focus on uniform or random training data selection, ignoring progressive training strategies.

Existing Knowledge:

- **Curriculum Learning:** The paper [5] first introduced the concept of curriculum learning, demonstrating its benefits for progressively complex tasks.
- **Music Generation Models:** Studies on Music Transformer [4] and Museformer [3] have shown the power of self-attention for capturing musical structures but highlighted memory and coherence issues.

- **Data Representation:** MIDI provides a structured, expressive way to represent music captures essential musical details. The event-based format is particularly efficient for catching the key musical elements needed for generation tasks.

1.4 Limitations

This research focuses specifically on Western classical piano music, utilizing datasets such as MAESTRO and piano-e-competition recordings. The choice of piano music provides a controlled environment for studying curriculum learning effects, as it avoids variables from multiple instruments while still containing rich musical complexity.

Regarding model architecture, our research focuses on the Music Transformer and its variations. While other architectures exist for music generation, this limitation allows for detailed analysis of how curriculum learning affects a specific architecture. This focused approach enables more meaningful comparisons between standard and curriculum-based training methods.

For evaluation purposes, measurable aspects of musical structure and coherence were focused more, such as harmonic consistency and pitch accuracy. While subjective metrics like aesthetic appeal are important in music, they are excluded from formal evaluation due to their subjectivity and difficulty in measurement. This limitation ensures reproducible and comparable results across different experimental conditions.

2

Background

The intersection of deep learning and music generation has seen remarkable progress in recent years, going from simple melody generation to complex, multi-track compositions. This advancement is because of several key developments in both model architectures and training approaches. Central to this progress lies symbolic music generation through MIDI format, which provides a structured representation of musical elements including pitch, duration, velocity, and timing information. This structured format has proven particularly suitable for machine learning applications, transitioning from early approaches like rule-based systems and simple probabilistic models to sophisticated deep learning techniques.

2.1 Introduction to Symbolic Music Generation

2.1.1 Symbolic vs Audio

A fundamental distinction in the field of automatic music generation lies in the choice of data representation: audio-based versus symbolic-based approaches. These two differ not only in the nature of the data they handle, but also in the type of information they emphasize, the techniques used for processing, and the kinds of musical tasks they are best suited for.

Audio-based music generation operates directly on raw or transformed audio signals. These can be represented as waveforms, spectrograms, or Chromagrams [7]. While audio representations capture the full richness and timbral characteristics of soundincluding tone color, articulation, and spatial positioningthey are typically high-dimensional and computationally intensive. Consequently, models working on audio must deal with extremely long sequences and subtle signal variations. WaveNet [8], for instance, is a prominent deep learning architecture that generates raw audio samples autoregressively, one value at a time.

In contrast, symbolic music generation[7] deals with an abstract, structured representation of music. Instead of working with sound, symbolic models operate on musical events like notes, durations, chords, rests, and dynamics. The fundamental unit in Symbolic Music is the note, defined by:

- Pitch: Represented as frequency (Hz), vertical position on a score, or pitch

notation (e.g., A = 440 Hz).

- Duration: Specified in milliseconds (absolute) or as a fraction of a whole note (e.g., quarter note = ij).
- Dynamics: Expressed in decibels (dB) or qualitative terms (e.g., ppp to fff).

A rest denotes silence and has only a duration, specified absolutely (ms) or relatively (e.g., whole rest = same as a whole note).

An interval measures the pitch difference (in semitones) between two notes (e.g., major third = 4 semitones). While useful for relative pitch representation, it is rarely used in deep learning due to tonality shift risks.

A chord consists of at least three notes and can be represented:

- Implicit: Listing all notes (e.g., C-E-G for C major).
- Explicit: Using chord symbols (e.g., Cmaj for C major).

Rhythm provides structure through pulsation and stress, organizing notes into beats and cycles essential for musical flow and dance.

2.1.2 Brief History of Symbolic Music Generation

Symbolic music generation has evolved significantly over the past few decades, progressing from rule-based systems to modern deep learning architectures. Early systems used hand-crafted rules based on music theory, such as grammars, templates, and constraints to generate compositions. These rule-based systems offered precise control but lacked the flexibility to generalize across styles or learn from data [9].

The introduction of probabilistic models, including Markov chains and Hidden Markov Models (HMMs), allowed for style imitation by capturing statistical regularities in music. However, their reliance on limited context windows and inability to model hierarchical structure restricted their expressive power [10].

The shift to neural network-based models, particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, marked a major breakthrough. These models could model temporal dependencies and generate more coherent sequences. Notable examples include DeepBach [11] and BachBot [12], which produced stylistically accurate chorales and harmonizations.

More recently, Transformer architectures have redefined symbolic music generation by leveraging self-attention to capture both local and long-range dependencies without sequential bottlenecks. Models like Music Transformer [4], Museformer [3], and JEN-1 [6] have demonstrated improved global coherence, thematic consistency, and stylistic adaptability. These architectures represent the state of the art in symbolic music generation, enabling more flexible and expressive musical outputs.

2.1.3 Challenges in Symbolic Music Generation

Despite rapid advancements in deep learning-based symbolic music generation, several core challenges persist. A primary issue is the lack of long-term structural coherence in generated compositions. Most existing systems have a tendency to generate music with no clear structure or sense of direction [7]. Especially when using recurrent sequence models (like RNN and LSTM), the generated music gradually becomes boring if there is no external guidance during the generating process[13]. While models such as Transformers capture local dependencies well, maintaining global formsuch as recurring themes, motif development, or verse-chorus structureremains difficult, especially in longer sequences[14]. This limitation becomes particularly pronounced in the context of Western classical music, where musical ideas often develop and recur over large time scales.

Another challenge lies in musical creativity and originality. Current models often rely on learning statistical patterns from data, favoring interpolation (combining/blending existing data) over extrapolation (inventing beyond the data)[15]. As a result, generated music may lack originality and instead closely imitate the training set. Defining and evaluating creativity computationally also remains an issue.

Different music genres have distinct harmonic, rhythmic, and melodic conventions, yet training a single model that can generate diverse stylesor transfer between themremains an open research area. Similarly, emotional expressiveness, interactive control, and evaluation consistency present ongoing challenges[13]. Most generation systems offer limited user interaction and provide only some or no control over aspects such as emotion, rhythm, or instrumentation. Evaluating generated music is equally difficult: objective metrics such as pitch accuracy or loss values often differ from subjective musical quality, making consistent benchmarking across systems problematic. Lastly, training deep generative models on large datasets of complex music can be computationally expensive and time-consuming.

These challenges highlight the need for continued exploration of architectures, learning strategies (such as curriculum learning), and evaluation frameworks that are better aligned with the unique properties of symbolic music.

2.2 MIDI as a Structured Representation of Music

The Musical Instrument Digital Interface (MIDI) is a widely adopted technical standard that allows electronic instruments, software, and devices to communicate by exchanging performance-related data[16]. Rather than encoding raw audio, MIDI captures music symbolically through a series of discrete events, such as when a note is played, released, how loud it is, and when it occurs. This makes MIDI an ideal format for symbolic music generation, offering a structured, compact, and interpretable way to represent complex musical information.

MIDI encodes each note using a pair of events : Note On (to indicate a note is played) and Note Off (to indicate the note has ended), each accompanied by

attributes such as pitch (via MIDI note number), velocity (indicating intensity), and timing (given in delta-time ticks from the previous event). For example, the event Note On, 0, 60, 90 denotes playing Middle C on channel 1 with a velocity of 90. Modern event-based representations [13] extend this scheme to include Time-Shift events (to represent silence or rests), Velocity changes, and control messages such as Tempo see Figure 2.1. These events can be serialized into a 1D token sequence, enabling their direct use with deep learning architectures like Transformers.

Event Type	Definition
Note On	indicates the start of a note, one for each pitch
Pitch	indicates the pitch value of a note
Note Off	indicates the end of a note, one for each pitch
Note Duration	inferred from the time gap between a Note On and the corresponding Note Off, by accumulating the Time Shift events in between
Time Shift	shifts the current time forward by the corresponding number of quantized time steps
Position/Sub-beat	points to different discrete locations in a bar
Bar	marks the bar lines
Piece Start	marks the start of a piece
Chord	one for each chord
Program/Instrument	sets the MIDI program number at the beginning of each track
Track	marks each track
Time Signature	marks the change of time signature
Note Velocity	sets the velocity for subsequent note-on events
Tempo	accounts for local changes in tempo

Figure 2.1: Musical Event

2.3 Music as Time series

A time series is a sequence of data points that happen one after another over time, typically spaced at uniform intervals. In statistical and machine learning contexts, time series analysis focuses on modeling and forecasting temporal dynamics by identifying trends, patterns, dependencies, and noise in data over time [17], [18]. Common applications include financial forecasting, climate modeling, and sensor data analysis.

Music, when represented symbolically or as audio, can naturally be treated as a multivariate time series. Each musical attribute such as pitch, duration, intensity, timbre, and rhythm can be seen as a variable that evolves over time. These variables are often highly interdependent, with changes in one aspect (e.g., rhythm) affecting others (e.g., phrasing or harmony) [19]. Unlike some domains, music features both short-term dependencies (e.g., note-to-note transitions, rhythmic motifs) and long-term structure (e.g., themes, verse-chorus forms), making it particularly challenging to model.

Traditional time series modeling techniques, such as Hidden Markov Models (HMMs) and Kalman filters, were among the first to be applied to music generation and analysis tasks [10]. These methods were used to learn statistical patterns in melodic or rhythmic progression. However, due to the non-linear and hierarchical nature of music, these linear models often fell short in capturing complex musical relationships.

Subsequently, non-linear and deep learning-based approaches such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformers have proven more effective in modeling music as a time series. These models can learn temporal patterns and structure directly from data, handling long-range dependencies more effectively. For instance, the Music Transformer uses relative positional encoding to preserve musical structure across long sequences [4].

2.4 Recent Advancements in Music Generation

2.4.1 Transformer-Based Models

Recent architectural innovations have significantly improved music generation capabilities. The Music Transformer [4] marked a significant breakthrough by introducing relative attention mechanisms specifically designed for music, enabling better modeling of long-term dependencies and musical patterns. This architecture uses self-attention to capture relationships between distant musical events, making it particularly effective for generating coherent and structured compositions. Building on this, Museformer [3] further refined this approach by implementing fine- and coarse-grained attention mechanisms, allowing the model to capture both local musical details and global structure. Museformers architecture is novel in that it uses hierarchical attention to model both local details (e.g., individual notes) and global structures (e.g., musical form) simultaneously.

2.4.2 Diffusion Models

Most recently, JEN-1 [6] showed the effectiveness of applying musical domain knowledge through omnidirectional diffusion models. The system enables fine-grained control over individual tracks while ensuring cross-track coherence through a progressive curriculum training strategy. This training approach gradually increases task difficulty, allowing the model to learn basic patterns before tackling more complex musical relationships. JEN-1 Composer's design demonstrates how structured training can improve quality in music generation, showing elements that parallel curriculum learning principles.

MusicLDM further explored the use of diffusion techniques for text-to-music generation, with a particular focus on enhancing novelty through beat-synchronous mixup strategies.[20] The researchers introduced two methodsbeat-synchronous audio mixup and beat-synchronous latent mixupto encourage the model to interpolate between musical training samples rather than copy them directly. This approach addressed a common challenge in music generation models: avoiding plagiarism while maintaining musical coherence. The research demonstrated that curriculum-like strategies could help generative models explore a broader creative space without sacrificing quality.

2.4.3 RNN-Based Models

Building on LSTM's capabilities for music generation, Roberts et al. introduced MusicVAE [21], a hierarchical latent vector model designed to capture long-term

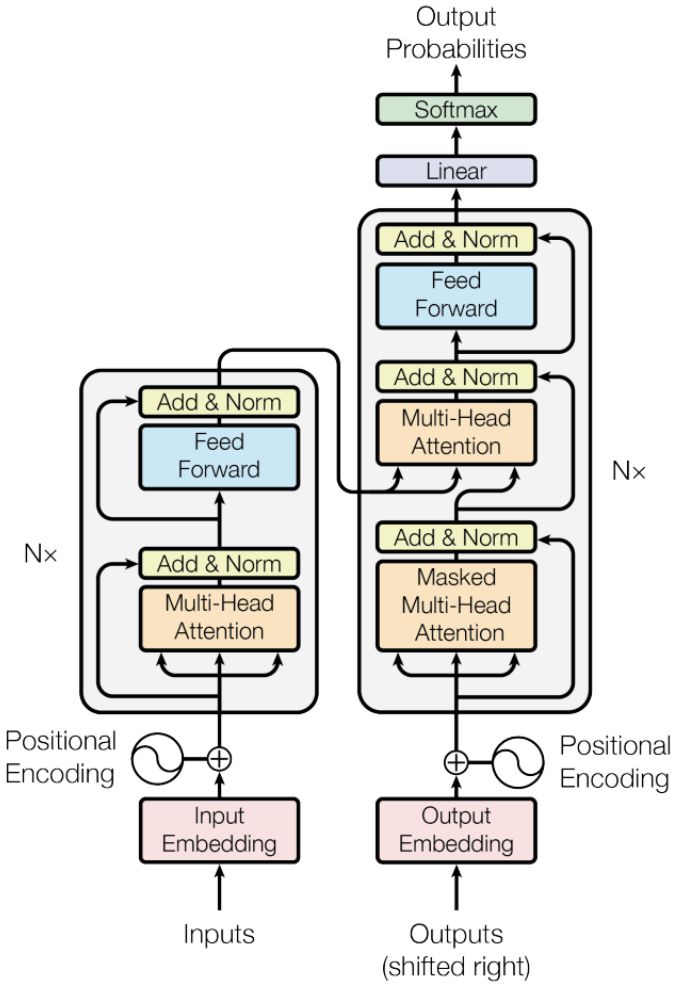


Figure 2.2: Transformer Architecture

structure in music. This approach combined variational autoencoders with LSTMs to address the challenge of modeling extended musical sequences. The authors identified that "music composed by standard recurrent neural networks (RNNs) often lacks global coherence" because "RNNs cannot keep track of temporally distant events that indicate global music structure." Their novel hierarchical decoder architecture enabled the model to generate musically coherent sequences of up to 256 tokens (16 bars), significantly longer than previous approaches. By compressing musical information into a structured latent space, MusicVAE also facilitated creative applications like interpolation between different pieces and attribute vector manipulation of musical characteristics.

Sturm et al. further explored the potential of LSTM networks for music transcription modeling and composition [22], with a particular focus on folk music. Their approach used a character-based LSTM and a token-based model trained on a large corpus of ABC notation transcriptions. The researchers demonstrated that these models could not only generate stylistically consistent music but also assist human composers in creating new works both within and outside the stylistic conventions of the training

data. As they noted, "LSTM networks are able to take a transcribed musical idea and transform it in meaningful ways," showing the models' potential as creative tools. Their practical implementation revealed that deeper networks trained on larger datasets produced more coherent musical outputs than earlier, smaller models.

2.5 Music Information Retrieval Foundations

Music Information Retrieval (MIR) provides critical foundations for our work in music generation. As established by Downie [23] and extended by Schedl et al.[24], music can be understood through multiple facets including pitch, temporal, harmonic, timbral, editorial, textual, and bibliographic elements. These facets represent different dimensions of music's inherent complexity that computational systems must address.

Traditional MIR research has focused on extracting meaningful features from music to enable tasks such as similarity measurement, classification, and retrieval. The evolution of MIR from simple feature extraction to sophisticated modeling approaches parallels the type of progression aimed for in this work through curriculum learning. In particular, the ability to represent and understand musical structures at different levels of detail from individual notes to complete compositions informs our approach to designing complexity metrics for curriculum learning.

The challenges identified in MIR evaluation, particularly around musical similarity and relevance judgments, parallel the difficulties in evaluating generated music. As noted by Schedl et al.[24], music ultimately exists in the mind of its perceiver, making objective evaluation challenging. These insights inform the strategy adopted in this thesis, which integrates both objective metrics and perceptual considerations.

The extraction of high-level music features from low-level audio processes has been another central aim of MIR research. As Casey et al.[25] explain, music is organized both horizontally (in time) and vertically (in frequency), with information constructed in hierarchical schemas. This hierarchical organization of music, from individual notes to complex compositions, provides a natural structure that can inform curriculum learning approaches. Casey et al.[25] also discuss music retrieval tasks ranging from exact audio matches to stylistically similar piecesa concept that reflects the varying levels of detail targeted in symbolic music generation.

2.6 Musical AI

Musical Artificial Intelligence (Musical AI) refers to the application of artificial intelligence techniques to the understanding, composition, performance, and interaction with music. This field lies at the intersection of computational creativity, music theory, cognitive science, and machine learning. Over the past decade, advancements in deep learning have significantly accelerated progress in Musical AI, enabling machines to generate, analyze, and even perform music in increasingly human-like ways.

Musical AI seeks to address tasks such as:

- Symbolic Music Generation: Producing sequences of notes or events using models trained on large corpora of music (e.g., Music Transformer [4], Museformer [3]).
- Audio Synthesis: Generating raw audio waveforms using models like WaveNet [8] or Diffusion-based architectures [26].
- Style Transfer: Altering one piece to reflect the style of another composer or genre [27].
- Harmonization and Accompaniment: Automatically generating harmony or orchestration for a given melody (eg. DeepBach) [11].
- Interactive Composition: Assisting human musicians by offering suggestions or completing musical ideas [28].

Musical AI systems often use deep neural networks trained on symbolic representations such as MIDI. These models learn the statistical patterns of musical structures like melody, harmony, rhythm and generate new material that mimics a particular style or composer. For instance, OpenAIs MuseNet and Googles Magenta project have showcased multi-instrument, multi-genre composition with minimal human input.

While these technologies have advanced rapidly, the integration of creativity, emotional expressiveness, and human-AI interaction remains a central challenge. Current systems tend to imitate rather than innovate, raising questions about origin and artistic intent. The goal of musical AI is not just to generate music that sounds plausible, but to engage in a process that resembles creative decision-making.

Moreover, Musical AI is increasingly being applied to fields other than academic contexts, including in film scoring, video games, generative soundtracks for social media, and music education tools. Startups and platforms such as AIVA, Amper Music, and Sonys Flow Machines are exploring commercial paths for AI-assisted music creation.

As this field evolves, it is likely that future systems will become more interactive, adaptive, and capable of creative collaboration with human composers changing the role of AI from tool to creative partner.

2.7 Curriculum learning

Curriculum Learning (CL) is a training paradigm inspired by the way humans learn: starting with simpler concepts and gradually progressing to more complex ones. This idea was first formalized by Bengio in [5], who showed that the order in which training examples are presented significantly influences how well and how quickly a model learns. Instead of training on randomly shuffled data, CL guides the model through a learning trajectory, much like how a student progresses from basic to advanced concepts in school. In technical terms, the study showed that models trained using curriculum-based strategies not only learned faster but also reached

better local minima, particularly in non-convex optimization landscapes like those found in deep learning, leading to better convergence rates and generalization in machine learning models.

2.7.1 Curriculum Learning Frameworks

Curriculum learning strategies generally fall under a "Difficulty Measurer + Training Scheduler" framework[29]:

- The Difficulty measurer decides the relative easiness of each data example.Common metrics include model-based loss, entropy of outputs, hand-crafted heuristics, or structural properties (e.g., in graphs or sequences).
- The Training scheduler decides the sequence of data subsets throughout the training process based on the judgment from the Difficulty Measurer. This can be a linear increase, staged release, or more adaptive methods depending on model feedback.

CL methods fall broadly into two categories:

Predefined Curriculum Learning: Here, both difficulty and schedule are fixed prior to training. Sequences are ordered according to a static criterion (e.g., loss from a pretrained model, sentence length[1]), and the curriculum unfolds according to a preset schedule. This is simple, interpretable, and computationally efficient.

Automatic Curriculum Learning: The curriculum is determined during training, often using reinforcement learning, meta-learning, or self-paced learning. These methods dynamically adapt the sample order based on model feedback or performance.

- *Self-paced learning*[30] is one such example, where the model selects easier samples early on and automatically advances as it improves.
- *Transfer Teacher*[31] approaches use a pretrained model to assign difficulty rankings to samples .
- *Meta-CL and Hierarchical CL* where curriculum itself is learned or applied at multiple levels of abstraction.

2.7.1.1 Curriculum Learning vs. Traditional Machine Learning

In traditional machine learning, data is typically shuffled randomly before training, under the assumption that all samples are equally informative. However, this ignores the natural development of learning complexity and may lead to unstable optimization, especially early in training. Curriculum learning challenges this, by organizing training examples from simple to complex shaping the models learning. This is particularly helpful in tasks with structured outputs (like language, graphs, or music), where learning fundamental patterns first leads to better generalization and interpretability.

2.7.2 CL Strategy for Symbolic Music Generation

Western Classical Music provides an ideal domain for studying curriculum learning due to its well-documented historical progression from the relatively structured forms of the Baroque period to the more complex compositions of the Romantic era, mirroring the gradual increase in complexity needed for curriculum learning. This natural evolution in compositional complexity offers a structured way to investigate whether progressive learning strategies can improve model performance.

Given the hierarchical and structured nature of symbolic Western classical music, a predefined curriculum is a natural fit. Music sequences vary in complexity based on rhythm, harmony, and melodic progression which can be approximated using model loss as a substitute for musical difficulty.

This thesis employs the following approach:

- **Predefined Difficulty**: Estimated using loss values from a pretrained Music Transformer. Lower loss sequences are treated as easier.
- **Continuous Training Scheduler**: A linear ramp-up strategy that gradually increases the proportion of high-complexity sequences over training epochs.

3

Theory

3.1 Symbolic Music as a Sequence Modeling Problem

Symbolic music, such as MIDI, encodes musical performances as sequences of discrete eventsnote onsets, note offsets, time shifts, and dynamicsrather than raw audio waveforms. These events can be tokenized similarly to words in natural language, allowing the application of sequence modeling architectures traditionally used in NLP like transformer-based models, which are widely used in deep generative modeling .

Let $M = \{m_1, m_2, \ldots, m_n\}$ represent a sequence of symbolic music events. The goal is to train a model to predict each event based on the ones before it, which can be written as: :

$$P(M) = \prod_{t=1}^{n} P(m_t \mid m_1, m_2, \ldots, m_{t-1}) \tag{3.1}$$

This means the model learns to generate music step-by-step, using the full history of events to predict what comes next. This approach helps capture the timing, structure, and musical flow needed for coherent compositions. Compared to audio, symbolic music keeps the musical structure like rhythm, melody, and harmony intact, which is especially useful for training models. Since transformer models are excellent at learning patterns over long sequences, they are a natural fit for generating symbolic music, where understanding context and structure over time is essential.

3.2 Self-Attention Mechanisms

The transformer architecture is based on the concept of self-attention, which allows the model to weigh the importance of different elements in a sequence. [32]This is particularly useful for tasks like music generation, where long-term dependencies and relationships between distant events are crucial.

3.2.1 Self Attention

The self-attention mechanism computes a weighted sum of all elements in a sequence, where the weights are determined by the similarity between elements. Given an input sequence $X = (x_1, x_2, ..., x_L)$, where each x_i is a vector representation of a token (e.g., a musical event), the self-attention mechanism computes three vectors for each token:

- Query (Q): Represents the token for which the attention is computed.
- Key (K): Represents the tokens to which it is attending.
- Value (V): Represents the information to be aggregated.

The self-attention mechanism is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{3.2}$$

where d is the dimension of the key vectors. The scaling factor $\frac{1}{\sqrt{d}}$ prevents the dot products from growing too large, which would push the softmax function into regions with extremely small gradients.[32]

3.2.2 Relative Attention in Music Transformer

The Music Transformer [4] extends the standard self-attention mechanism by introducing relative attention, which models the relative timing and pitch relationships between musical events. This is particularly important for music generation, where the relative positions of notes (e.g., intervals, rhythms) are more meaningful than their absolute positions.

To achieve this, the model learns a relative position embedding E_r that encodes the distance between tokens in a sequence. The attention weights are then computed by modifying the standard self-attention formula to include a relative position bias:

$$\text{RelativeAttention} = \text{softmax}\left(\frac{QK^T + S_{rel}}{\sqrt{d}}\right)V \tag{3.3}$$

where S_{rel} is a learnable matrix representing pairwise relative distances. This technique improves long-range musical coherence while reducing the memory overhead from $O(L^2D)$ to $O(LD)$ for sequences of length L and hidden dimension D.

3.3 Music Transformer Architecture

The Music Transformer follows the standard Transformer decoder architecture see Figure3.1 with key modifications for musical sequence modeling. The model uses relative attention mechanisms and learned position information, with each layer consisting of a self-attention sub-layer followed by a feedforward sub-layer.[4]

The Music Transformer builds upon the standard Transformer decoder but replaces absolute position representations with relative attention mechanisms throughout.

The architecture consists of:

- Relative Embedding Dimension: Typically 512 dimensions for token representations
- Layers: Multiple stacked decoder layers
- Attention Heads: Multi-head relative attention
- Hidden Dimension: Feedforward layers with larger intermediate dimensions

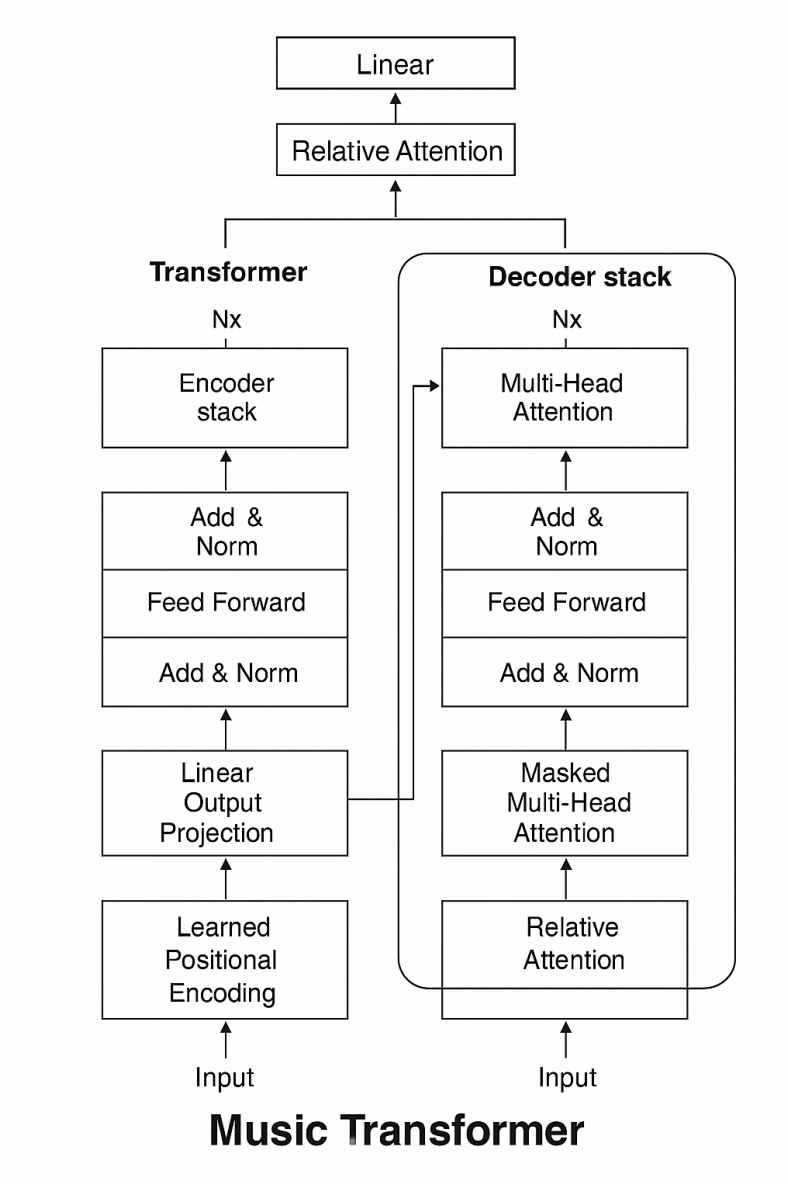


Figure 3.1: Music Transformer Architecture

3.3.1 Stacked Decoder Architecture

The Music Transformer employs a stack of identical decoder layers, where each layer builds upon the representations learned by the previous layer. Each decoder layer processes input through two sub-layers with residual connections and layer normalization:

$$\text{DecoderLayer}_l(x) = \text{LayerNorm}(x + \text{FFN}(\text{LayerNorm}(x + \text{MultiHeadAttention}(x))))$$

This stacked architecture enables hierarchical learning: lower layers capture local musical patterns while upper layers model long-term dependencies and musical structure.

3.3.2 Multi-Head Attention Mechanism

The attention mechanism uses multiple heads (typically $H = 8$) to allow the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this capability.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)W^O$$

where each head is computed as:

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

The projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

This multi-head structure enables the model to focus on different aspects of musical relationships simultaneously, such as melodic contour, harmonic progression, and rhythmic patterns.

3.3.3 Position-wise Feedforward Networks

Each decoder layer contains a fully connected feedforward network that is applied to each position separately and identically. This sub-layer consists of two linear transformations with a ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

where W_1, W_2, b_1, b_2 are the weights and biases of those two layers.

3.4 Curriculum Learning

Curriculum learning (CL) is inspired by the human learning process, where simpler concepts are learned before tackling complex ones. In deep learning, this corresponds to progressively increasing the difficulty of training data. CL has been successfully applied to NLP[33] and image generation[34] to improve optimization and generalization dynamics . Curriculum learning is the training strategy that trains a machine learning model with a curriculum.

3.4.1 Formal Definition[5], [29]

A curriculum is defined as a sequence of training criteria over T training steps:

$$C = \langle Q_1, Q_2, \ldots, Q_T \rangle$$

Each criterion Q_t is a reweighting of the target training distribution $P(z)$:

$$Q_t(z) \propto W_t(z)P(z), \quad \forall z \in D$$

where $W_t(z)$ is a weight function applied to training example z at step t, and D is the dataset. The curriculum satisfies the following conditions:

1. The entropy of the weighted distribution increases over time: $H(Q_t) < H(Q_{t+1})$. Meaning, the training data becomes progressively more diverse and informative
2. The importance of each sample increases or remains the same: $W_t(z) \leq W_{t+1}(z), \quad \forall z \in D$. More (or harder) examples are introduced gradually.
3. The final weighting equals the original data distribution: $Q_T(z) = P(z)$. ie. Eventually, the full dataset is used with uniform weighting

3.4.2 Complexity Metrics

In the context of curriculum learning, quantifying the difficulty of data samples is crucial for structuring the training schedule. Let D=$d_1, d_2, , d_n$ be a dataset where each d_i represents a symbolic music sequence (e.g., tokenized MIDI events) . The complexity of a sequence is defined by a function C(d_i) which estimates how difficult the sequence is for a model to learn. Sequences are sorted based on this score to form a learning path from simple to complex.

Loss as a Proxy for Difficulty :

A practical and empirically grounded approach to estimating sequence complexity is to use model-based loss values. The primary loss function used is the masked cross-entropy loss:

$$C(d_i) = 1/T \sum_{t=1}^{T} L_t(d_i) \tag{3.4}$$

where : T is the number of training epochs, and $L_t(d_i)$ is the masked cross-entropy loss for sample d_i at epoch t.

The standard cross-entropy loss measures the difference between the predicted probability distribution and the true distribution of the next event. For a sequence of length n, the loss is computed as:

$$-\sum_{t=1}^{n} \log P(x_t \mid x_1, x_2, \ldots, x_{t-1}) \tag{3.5}$$

This loss function is minimized during training to improve the model's ability to generate coherent and musically plausible sequences.

Masked Cross Entropy[4] is used to ignore non-informative tokens like padding tokens or silence when computing loss thus improving learning.

$$L_{masked} = {}^{n}_{t=1} mask_t log P(x_t x_{<t})$$

This allows for more accurate complexity estimation and model feedback in curriculum schedules.

3.5 Kullback-Leibler Divergence for Model Comparison

The Kullback-Leibler Divergence (KLD) is widely used in deep generative models, including VAEs, transformers, and diffusion models, to either:

- Compare the models output distribution against a reference (e.g., real music data)[35], or
- As a regularization term, ensuring that learned latent spaces maintain smoothness or statistical consistency[36]).

In this paper, KLD is an information-theoretic metric that quantifies the difference between two probability distributions.

3.5.1 Definition

Given discrete distributions P and Q, the KLD from Q to P is:

$$D_{KL}(P||Q) = \sum_{x} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{3.6}$$

3.5.2 Overlap Area (OA)

While KLD captures divergence, Overlap Area (OA) quantifies similarity. It measures the overlapping region between two probability density functions:

$$\mathrm{OA}(p, q) = \int \min(p(x), q(x)) dx \tag{3.7}$$

OA ranges from 0 (no overlap) to 1 (identical distributions), offering a more interpretable view of distributional similarity.

3.5.3 Application to Training Dynamics

Here KLD is used to compare the loss distributions of models trained under different regimesspecifically, a baseline model versus curriculum learning variants.

- P: loss distribution from the baseline model
- Q: loss distribution from a curriculum learning model
- Metric: $D_{KL}(P||Q)$ and $\text{OA}(P, Q)$

A higher KLD value indicates greater divergence, meaning the models behave differently across the complexity spectrum of musical sequences. Both KLD and OA is applied to the loss distributions of the baseline and curriculum models. This dual-metric approach enables us to evaluate:

- Whether curriculum learning shifts the model's behavior across sequence complexities.
- How much of the performance profile is similar or distinct between models.

Together, KLD and OA provide a subtle view of how curriculum learning affects generalization and sequence handling.

3.5.4 Application to Musical Quality Evaluation

Beyond loss-based comparisons, Kullback-Leibler Divergence is also applied to assess the quality of generated music by comparing the distribution of extracted musical features with those from real compositions.

- P: probability distribution of a musical feature from real samples
- Q: probability distribution of a musical feature from generated samples
- Metric: $D_{KL}(P||Q)$

A lower KLD value indicates that the generated music closely mimics the statistical characteristics of real music in that dimension.

In our evaluation, KLD is computed for each feature category:

- **Pitch**: Entropy, pitch class histogram
- **Rhythm**: Inter-onset interval, rhythmic intensity
- **Harmony**: Consonance, polyphony
- **Structure**: Self-similarity matrix, information rate

This distributional comparison provides a rich measure of how well the generative model captures real-world musical characteristics, beyond surface-level accuracy.

4

Methodology

4.1 Dataset

The MAESTRO dataset will be used, which contains high-quality MIDI files aligned with piano performances. This dataset is ideal for training music generation models due to its expressive and diverse compositions. Introduced by Hawthorne et al.[37], MAESTRO consists of classical solo piano performances, providing a musical domain that captures the full spectrum of classical piano repertoire spanning multiple periods and composers.

The dataset is built from live recordings of piano competitions hosted by the International Piano-e-Competition. Each recording is accompanied by both high-quality audio and time-aligned MIDI files . The MIDI data provides high-resolution symbolic representations that capture note onsets, offsets, velocities, and pedal information with precise temporal accuracy. Importantly, the dataset includes only piano, with no other instruments, vocals, or ensembles.

For this research, an expanded version of the MAESTRO dataset was utilized, incorporating six years of high-quality classical piano compositions (2011, 2013-2015, 2017-2018) totaling 727 pieces with approximately 113 hours of performance duration. This expanded dataset provides comprehensive coverage of Western classical piano compositions.

4.1.1 Data Preprocessing

The MIDI files are preprocessed into token sequences following a methodology provided in an open-source implementation in `music-transformer` [1]. Although the preprocessing pipeline from the repository is used, the key steps are summarized here for completeness:

- **Tokenization & Vocabulary Encoding:** Musical events are converted into discrete tokens representing:

[1] `https://github.com/spectraldoy/music-transformer`

Index Range	Token Type	Description
[0]	<pad>	Padding token for batch processing
[1-128]	Note-on events	Corresponding to MIDI note numbers
[129-256]	Note-off events	Signals to stop playing specific notes
[257-381]	Time-shift events	Representing temporal gaps between events
[382-413]	Velocity events	Encoding dynamics (note intensity)
[414-415]	<start>, <end>	Special tokens to mark sequence boundaries

Table 4.1: Vocabulary structure for music event tokenization

- **Sequence Preparation:**

  Each musical composition was processed by segmenting it into multiple sequences of variable length. For each composition, two types of sequences were extracted: a randomly selected middle section and an ending section. This approach allowed to capture different structural elements from the same composition.

  Each sequence was targeted to be approximately 2048 tokens long, with some controlled variability introduced by randomly sampling the exact length:

  $$length = randint(lth - lth//factor, lth + lth//factor)$$

  Where lth is 2048 and factor is 6. This results in sequence lengths varying between approximately 1707 and 2389 tokens. This approach helps the model generalize better to sequences of different lengths while maintaining a manageable context window.

- **Data Augmentation:** To enhance model generalization and increase training data diversity, two key augmentation techniques are applied:
  - **Pitch Transposition:** Sequences are transposed by ±2 semitones.
  - **Time Stretching:** Events are temporally stretched by factors of 1.0, 1.05, and 1.1.
  - **Result:** Approximately 37,850 tokenized sequences.

The preprocessed sequences were compiled into a single tensor for efficient batch processing during model training.

4.2 Baseline Model Implementation

4.2.1 Model Architecture

The Music Transformer will serve as the baseline model for this research. This architecture is well-suited for music generation due to its self-attention mechanisms, which effectively capture long-term dependencies in sequential data. The model will be configured with an embedding dimension of 512, 8 layers, and 8 attention

heads. It uses a relative attention mechanism to model relative timing and pitch relationships, which is crucial for maintaining musical structure over long sequences. The cross-entropy loss function will be used for training, as it is appropriate for token-based sequence prediction tasks. Training efficiency will be monitored through time-to-convergence and validation loss. The model is trained using the cross-entropy loss function with a padding mask to ignore non-musical tokens. The Adam optimizer was employed with parameters

$$\beta_1 = 0.9,\ \beta_2 = 0.98,\ \text{and}\ \epsilon = 1 \times 10^{-9}$$

, following the standard transformer learning rate schedule with warm-up steps. Training is conducted with a batch size of 16 sequences. The model uses cross-entropy loss with padding mask and applies a dropout rate of 0.1 to prevent overfitting.

All models baseline and curriculum learning variantsshare this exact configuration to ensure consistency and isolate the effects of curriculum scheduling. The only distinction lies in their training schedule and data exposure strategy.

Two training schemes are used across different stages:

- A 300-epoch model is used to pretrain a baseline Music Transformer for the purpose of computing sequence-wise loss values for curriculum ranking.
- A standard 100-epoch baseline model is used as the primary reference for comparison against curriculum learning variants, trained on the full dataset without any curriculum.

4.3 Curriculum Learning Implementation

This section outlines the training framework developed to implement curriculum learning (CL) for symbolic music generation using the Music Transformer architecture. The framework is based on the "Difficulty Measurer + Training Scheduler" paradigm (see section in Theory/Background) introduced by Wang in [29], which separates the curriculum process into how difficulty is defined and how it is introduced over time. Our implementation employs a predefined difficulty metric (masked cross-entropy loss) and a progressive data exposure strategy, allowing for structured learning progression from simple to complex musical sequences.

4.3.1 Difficulty Measurement and Data Sorting

The difficulty measurer in our curriculum learning framework is based on a predefined loss ranking. A Music Transformer model, pretrained on the full training dataset is used as an external evaluator of sequence complexity. Each training sequence is passed through this model to compute its masked cross-entropy loss, which reflects how well the model can predict the continuation of the sequence. Sequences that yield higher loss values are interpreted as being more complex, i.e. harder for the model to predict while lower-loss sequences are said to be easier.

These loss values are used to create the sorted dataset with sequences from easiest to hardest. This ranked list serves as input to the training scheduler. This method provides a consistent, model-driven way to estimate how difficult each music sequence is.

4.3.2 Training Scheduler

Curriculum Progression Strategy

Training begins with the easiest 20% of the dataset, determined by a predefined sequence complexity ranking (see Section 4.3.1). As training advances, the portion of the dataset accessible to the model is gradually expanded, following a linear curriculum schedule. This expansion is governed by a scheduling function that maps training progress to the percentage of data exposure:

$$\text{curriculum\%} = \begin{cases} 0.2, & \text{if batchesprocessed} \leq \text{phase}_1\text{end} \\ 0.2 + 0.8 \cdot \frac{\text{batchesprocessed} - \text{phase}_1\text{end}}{\text{phase}_2\text{end} - \text{phase}_1\text{end}}, & \text{if phase}_1\text{end} < \text{batchesprocessed} \leq \text{phase}_2\text{end} \\ 1.0, & \text{if batchesprocessed} > \text{phase}_2\text{end} \end{cases} \tag{4.1}$$

Where:

- $phase_1end$ is the start of curriculum expansion (typically 0),
- $phase_2end$ is a configurable milestone (a percentage of total training batches) at which full dataset exposure is reached,
- batchesprocessed is the cumulative number of batches completed during training.

This process enables flexible curriculum pacing by adjusting the transition point (phase_2_end) depending on experimental needs. Once the training reaches this point, the model continues using 100% of the dataset for the remainder of training. This approach is applied at the batch level, allowing smoother transitions between difficulty levels and consistent comparison across models.

Training Implementation and Data Subset Selection

At each training step, the model is trained on a dynamically selected subset of the dataset, defined by the current curriculum percent. The number of sequences used is calculated as:

$$\text{subset\_size} = \lfloor \text{curriculum\%} \times \text{total\_training\_sequences} \rfloor \tag{4.2}$$

This subset is selected from the start of the sorted training data (from easiest to hardest), and then shuffled to preserve diversity in training batches. Training, therefore, progresses through three conceptual phases:

- Initial phase: Limited to the simplest 20% of sequences.

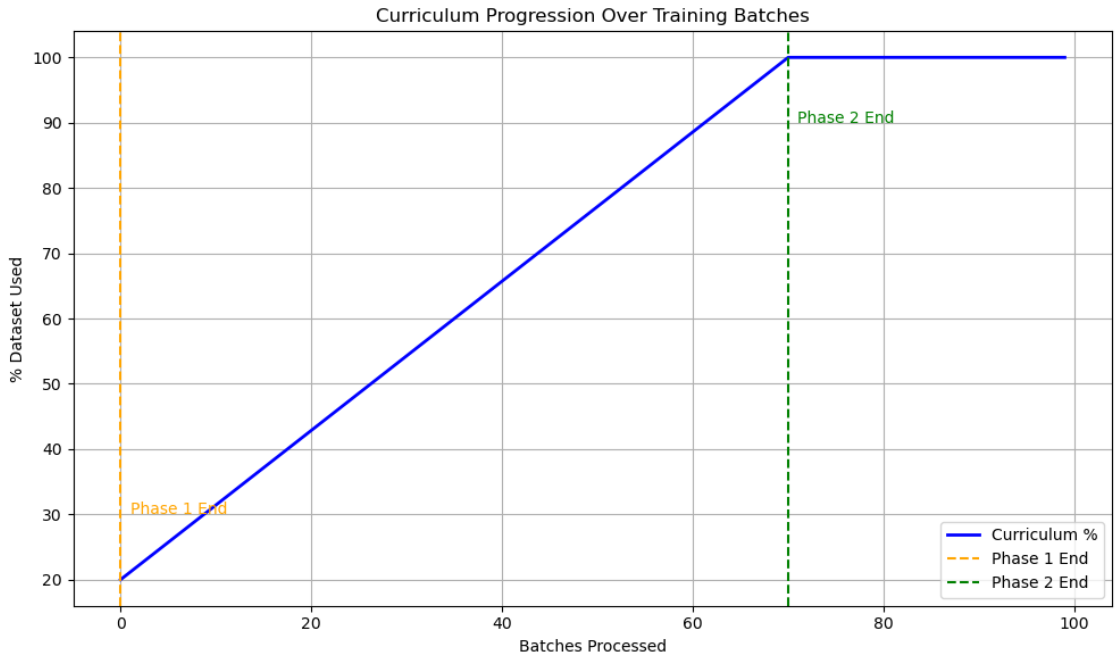


Figure 4.1: Curriculum Progression Scheduler

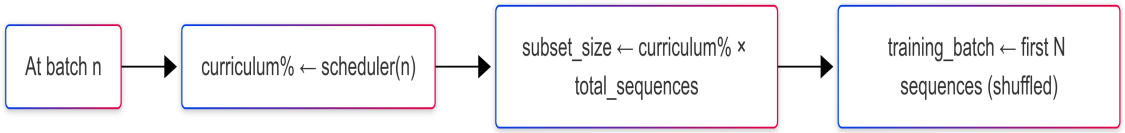


Figure 4.2: Training Step Data Selection Logic

- Growth phase: Gradual inclusion of more complex sequences as curriculum percentage increases.
- Full-data phase: The entire dataset is used for training once the growth phase completes.

This curriculum strategy is applied at the batch level rather than the epoch level, enabling finer control over training progress. This allows the model to experience smoother and more continuous transitions between different difficulty levels.

4.4 Evaluation Framework

4.4.1 Training Dynamics Analysis

Comprehensive monitoring of learning progression across all training approaches:

Convergence Metrics

- Training loss trajectories: Per-batch loss tracking throughout training
- Validation loss curves: Regular evaluation on held-out sequences

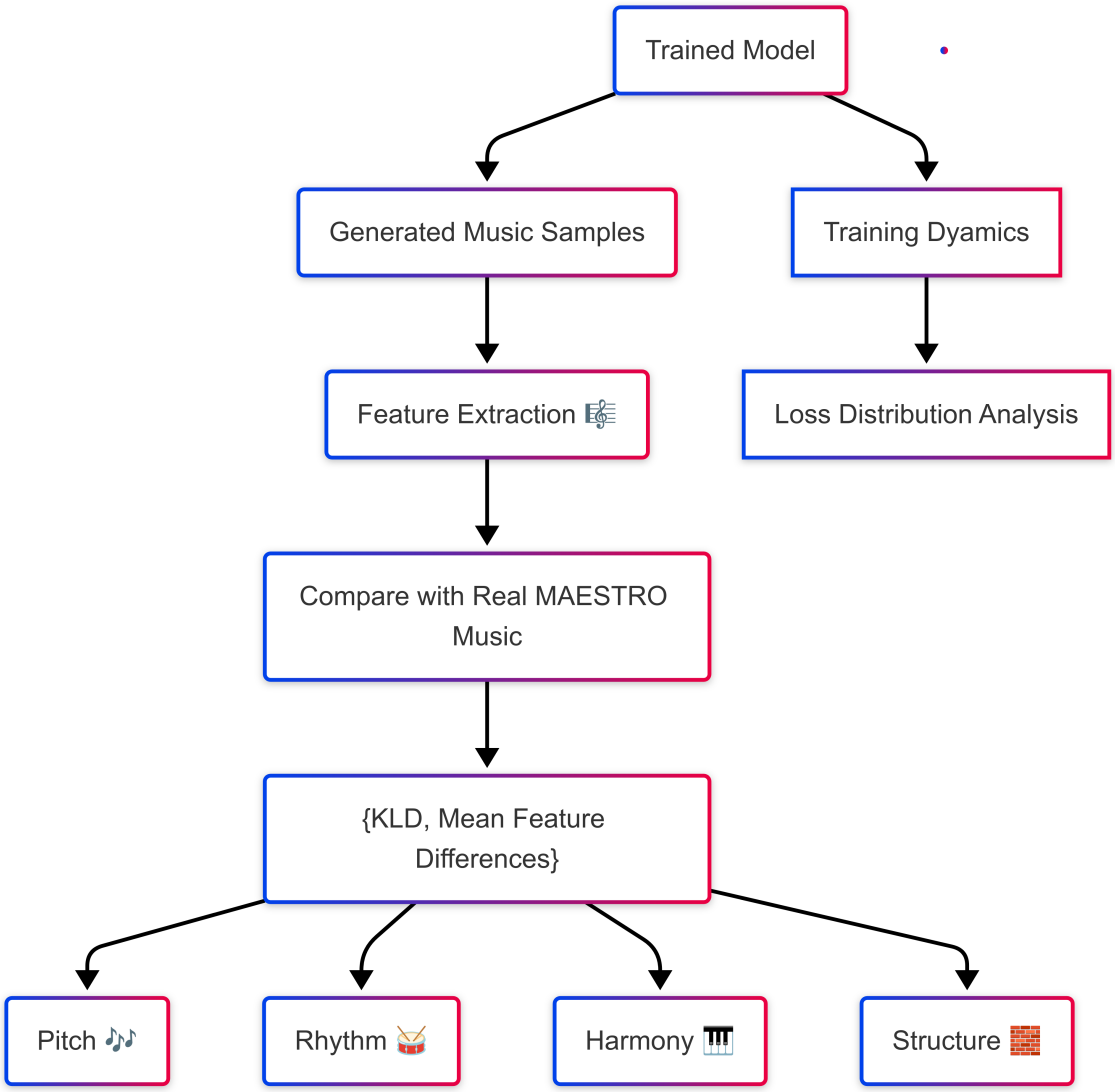


Figure 4.3: Evaluation WorkFlow

- Final performance comparison: Training and validation loss at convergence

Learning Pattern Assessment

- Early stage performance: Initial 20% of training batches
- Transition phase analysis: Curriculum expansion periods
- Final convergence characteristics: Last 20%-40% of training batches

Although training and validation losses are often logged per epoch, this can lead to misleading comparisons when evaluating curriculum learning models. Because CL models begin with a smaller portion of the dataset and expand gradually, they typically require more epochs to process the same total number of training batches as a baseline model.

To address this, in addition to epochs, all loss metrics in this study are also logged

and visualized per batch as well. This ensures a fair comparison by aligning training progress across models according to the number of batches processed, ie. both baseline and curriculum models are evaluated after processing the same total number of training batches. This avoids the misleading appearancecreated by epoch-based plotsthat curriculum learning models take longer to converge. In reality, they process the same number of training batches, just spread across more epochs due to the gradual dataset expansion.

Batch-aligned loss plots thus provide a more reliable basis for assessing training dynamics, as they reflect equivalent data exposure across models.

4.4.2 Loss Distribution Analysis

To evaluate and compare model behavior across different training strategies, a per-sequence loss analysis is performed on the full training dataset. For each trained model, the dataset is passed through the final model one sequence at a time, and the masked cross-entropy loss is computed for every sequence independently. This produces a fine-grained view of how well each model handles sequences of varying difficulty. The resulting loss values form a distribution that reflects the models performance across the entire dataset. To compare these distributions across models, Gaussian Kernel Density Estimation (KDE) is applied to create smooth probability density functions. These distributions are then compared using two key divergence metrics:

- KullbackLeibler Divergence (KLD): Measures the information difference between two probability distributions, highlighting where models diverge in behavior.
- Overlap Area (OA): Quantifies the shared region between two distributions, with higher values indicating greater similarity in performance across sequences.

By examining these distributional characteristics alongside visual plots, a deeper insight into how different models generalize is gained, particularly in how they handle sequences of varying complexity

4.4.3 Performance Analysis

To evaluate the quality of generated music, features across four key musical dimensions: pitch, rhythm, harmony, and structure was assessed. [13]. Formal definitions for all metrics are provided in Appendix B.

Pitch-Based Features
Pitch-focused metrics assess tonal distribution, diversity, and melodic behavior:

- Pitch Entropy: Measures the unpredictability of pitch class usage, indicating tonal diversity. Higher entropy suggests more evenly distributed pitch material.
- Pitch Class Histogram Similarity [38]: Compares the pitch class distribution of generated music to that of real music, evaluating tonal alignment.

- Pitch Range: Represents the interval in semitones between the lowest and highest pitch, reflecting melodic span.
- Empty Bar Ratio: Percentage of bars with no note events, reflecting structural density and continuity.
- Qualified Note Ratio: The proportion of notes with musically meaningful durations, filtering out brief or ornamental events.

Rhythm-Based Features

- Average Inter-Onset Interval (IOI) :[38]: Quantifies rhythmic stability by analyzing time gaps between note onsets. Higher values indicate slower rhythms.
- Rhythmic Intensity [39] : Represents the percentage of beats with at least one note onset, capturing rhythmic activity.
- Qualified Rhythm Frequency: Measures how often rhythmic events align with standard metric subdivisions.

Harmony Based Features

- Pitch Consonance Score [40]: Evaluates harmonic stability by quantifying how consonant or dissonant the intervals are within a tonal context.
- Polyphony [41] : Average number of simultaneously sounding notes, indicating harmonic complexity.

Structure-Based Features

- Self-Similarity Matrix (SSM) Score [41]: Captures recurring patterns or motifs within a piece. Higher scores suggest stronger internal structure.
- Information Rate: Represents the balance between repetition and novelty in the musichigher values indicate richer but more coherent musical progression.

To compare the statistical alignment between generated music and real compositions, the mean feature values across 100 samples per model is calculated and compare them with 100 excerpts from the MAESTRO dataset. Additionally, the KullbackLeibler Divergence (KLD) is computed for each feature distribution to quantify how closely the generated outputs resemble the distributional properties of real music. Lower KLD values indicate greater similarity to the reference dataset. This multi-level analysis enables a comprehensive comparison of model outputs, capturing both low-level musical realism and high-level structural and stylistic features.

4.5 Evaluation Criteria

4.5.1 Training Dynamics Assessment:

Convergence Pattern Assessment: Models will be evaluated based on their training and validation loss trajectories throughout the training process. Curriculum learning models should demonstrate distinct learning patterns, particularly during

early training phases and curriculum transition periods. Per-batch loss tracking will be used to ensure fair comparison across different training approaches, accounting for varying dataset exposure rates in curriculum models.

Learning Progression Stability: The smoothness of transitions between curriculum phases will be assessed, with successful curriculum implementation showing controlled adaptation to increasing data complexity without significant performance disruption. Models should maintain consistent learning momentum across curriculum milestones.

Final Performance Competitiveness: All training approaches should achieve comparable final performance levels, with curriculum models expected to converge within reasonable proximity to baseline performance.

4.5.2 Musical Quality Criteria

Symbolic Feature Alignment: Generated music will be evaluated against real music characteristics using the MAESTRO dataset as reference. Models achieving closer statistical alignment across multiple musical dimensions will be considered superior, including rhythmic, pitch, harmonic, and structural properties.

Feature Distribution Similarity: Kullback-Leibler Divergence values between generated and real music feature distributions will serve as the primary quantitative measure of musical quality. Models achieving lower KLD values across multiple features demonstrate superior alignment with real musical characteristics.

4.5.3 Distributional Learning Characteristics

Learning Pattern Differentiation: KLD analysis between curriculum and baseline models should reveal statistically meaningful differences in loss distributions, indicating that curriculum learning fundamentally alters how models process sequences of varying complexity. Significant divergence values suggest successful curriculum integration.

Baseline Similarity Preservation: Models maintaining high overlap area values when compared to baseline distributions are considered to preserve baseline-like consistency while incorporating curriculum benefits.

5

Experiments

This chapter presents a comprehensive exploration of the experimental setup designed to evaluate the effectiveness of curriculum learning strategies for symbolic music generation. The following sections detail the dataset configuration, baseline model implementation, curriculum learning variants, complexity measurement methodology, and preliminary findings related to training dynamics and curriculum efficacy.

5.1 Dataset and Experimental Setup

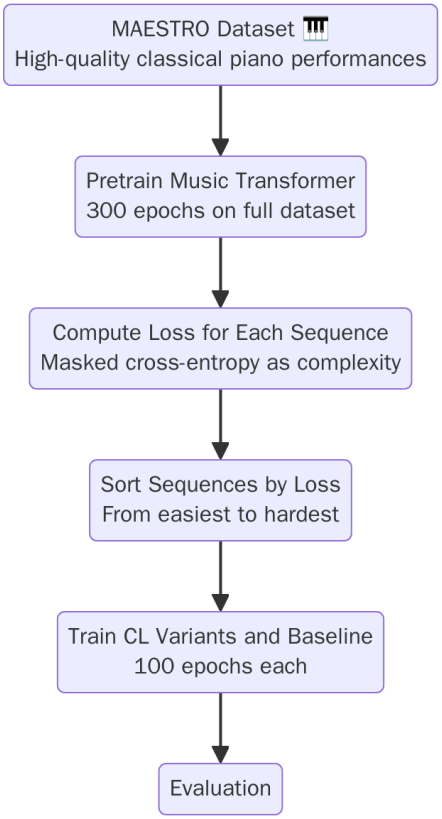


Figure 5.1: Experimental Setup

5.1.1 Dataset Configuration

To evaluate curriculum learning strategies, the preprocessing pipeline was applied to the expanded MAESTRO dataset described in Section 4.1. After preprocessing and augmentation, the dataset comprised 37,850 sequences. A stratified sampling strategy was employed to ensure a balanced distribution of sequence complexity across training and validation splits. Specifically, every fifth sequence i.e., 20% (7,570 sequences) was selected for the validation set, while the remaining 80%(30,280 sequences) were used for training. This approach maintained diversity across complexity levels, which is crucial for fair evaluation of both baseline and curriculum learning models. Each sequence had a fixed length of 2,048 tokens corresponding to roughly 15 - 40 seconds of music depending on the local tempo and event density.

5.1.2 Implementation of the Baseline Model

The Music Transformer architecture was implemented as the baseline model and was trained for 300 epochs on the preprocessed MAESTRO dataset.
The model was trained with the following hyperparameters:

- Batch size: 16
- Learning rate: Adaptive, using the transformer learning rate schedule with warmup
- Optimizer: Adam ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\varepsilon = 1e-6$)
- Loss function: Cross-entropy with padding mask
- Dropout rate: 0.1

A custom learning rate schedule was utilized as described in Attention is all you need [32], which increases the learning rate during an initial warmup phase and then decreases it proportionally to the inverse square root of the step number:

$$\text{lr} = (d_{\text{model}})^{-0.5} \cdot \min\left(\text{step\_num}^{-0.5}, \text{step\_num} \cdot \text{warmup\_steps}^{-1.5}\right)$$

This schedule has proven effective for training transformer-based models, allowing for efficient optimization during the initial training phases and stability during later phases.

5.2 Curriculum Learning Variants

To systematically assess the impact of curriculum pacing and optimization strategies, three distinct curriculum learning (CL) variants were implemented. All variants used a predefined curriculum based on loss-based complexity ranking, but differed in terms of pacing and optimization scheduling.

- **CL-60% Model**: This model introduced the curriculum in a linear fashion, beginning with the simplest 20% of the training data and expanding to the full dataset by 60% of total training steps. Difficulty was measured using

pre-computed cross-entropy loss values from the pretrained baseline model. The learning rate followed the standard transformer schedule throughout.

- **CL-80% Model**: This variant featured a more gradual pacing, reaching full dataset exposure at 80% of the total training steps. Like CL-60%, it used the same loss-based complexity ranking and maintained the standard learning rate schedule. This model was designed to explore the effect of slower curriculum expansion on stability and final convergence.
- **CL-60% LR model**: The CL-60% LR model builds directly upon the curriculum scheduling strategy employed in the CL-60% variant, where in the full dataset is introduced by 60% of the total training batches. This variant combines the curriculum scheduling strategy of CL-60% with an adaptive learning rate approach specifically designed for curriculum learning. As illustrated in 5.2, the learning rate remains elevated and constant during the curriculum phase until the model reaches full dataset exposure at 60% of training steps. Only then does the standard transformer learning rate decay begin, allowing for optimized learning dynamics during the critical transition from curriculum to full dataset training.

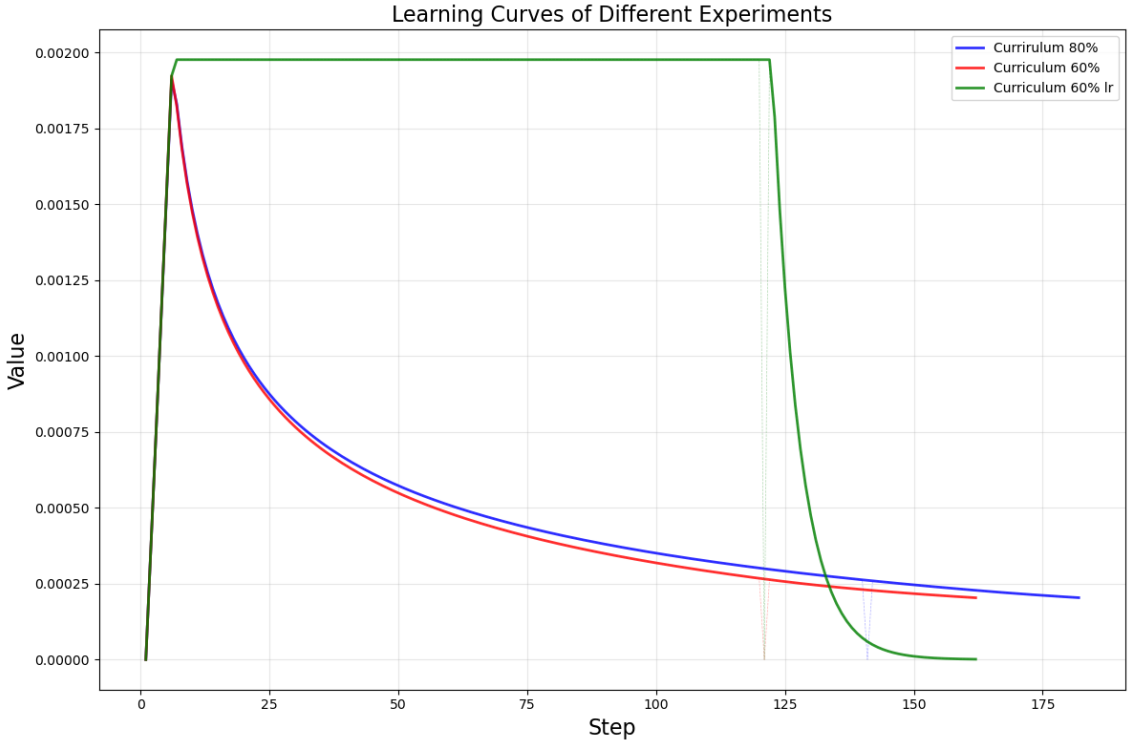


Figure 5.2: Learning Curves of Different Experiments

Learning rate schedules across different curriculum variants. The CL-60% LR model (green line) maintains constant elevated learning rate during curriculum phases, contrasting with the continuous decay used in other variants.

The decision to introduce a dynamic learning rate adjustment was motivated by preliminary observations comparing the CL-60% and CL-80% models (see chapter 8

). Although both curriculum variants adhered to the same loss-based complexity ordering, the CL-60% model demonstrated more stable convergence and better final validation loss than its slower-paced counterpart, CL-80%. This suggested that reaching full data exposure earlier may allow the model to benefit from longer training on the complete distribution without compromising stability.

To further enhance this advantage, the CL-60% LR model introduces learning rate modulation aligned with curriculum transitions. The hypothesis is that adapting the learning rate during key curriculum milestones particularly during the transition from low-complexity to high-complexity sequences would improve optimization dynamics.

5.3 Curriculum Design

All curriculum models were driven by the complexity ranking and sequence sorting process as defined in Section 4.3.1. The curriculum progression followed a linear expansion schedule with the training beginning with the easiest 20% of the dataset. As training progressed, more complex sequences were gradually introduced. Each model reached full dataset exposure at its designated milestone (60% or 80% of total training steps). To ensure fair evaluation across models, the total number of training batches was fixed at 189,300. This number was derived from:
Total training sequences: 30,280
Batch size: 16
Epochs: 100

$$TotalBatches = \frac{30,280}{16} * 100 = 189,250 \approx 189,300$$

This fixed batch count ensures that all curriculum variants are trained for the same overall data exposure and computational cost, regardless of curriculum pacing.

5.4 Early Training Dynamics

This section highlights early-stage observations related to training dynamics and curriculum integration. Detailed quantitative evaluations and musical output assessments are presented in Chapter 8. Initial experiments confirmed that all three curriculum learning models successfully integrated progressive complexity scheduling as designed. In comparison to the baseline model trained on randomly shuffled data, the curriculum-based models demonstrated faster convergence during the initial training epochs, which is a commonly observed benefit of curriculum learning, as models initially face a smoother optimization landscape. Notably, transitions to more complex sequence sets were seen as subtle shifts in training loss curves, indicating that the models were responsive to curriculum phase changes.

5.5 Exploratory Loss Studies

Several exploratory studies were conducted prior to finalizing the expanded MAESTRO dataset and curriculum strategy to better understand the relationship between loss-based complexity and sequence predictability. These experiments were performed on a smaller subset of the MAESTRO dataset (2018) and involved three different dataset configurations. A pretrained Music Transformer was used to compute the per-sequence masked cross-entropy loss, which served as a proxy for musical complexity. Loss values were then used to sort training sequences from easiest to hardest, forming the foundation of our curriculum design. The datasets used in these explorations were constructed as follows:

- Dataset 1: Loss computation and sequence sorting were performed using the same dataset that the pretrained model was originally trained on.
- Datasets 2 & 3: Loss values were computed using the pretrained model from Dataset 1, but applied to two separate, unseen datasets to evaluate generalization across varying data distributions.

These Curriculum Learning experiments revealed that:

- Sequences located toward the ends of musical pieces tended to be more predictable (i.e., lower loss), aligning with conventional compositional closure patterns.
- Using mismatched datasets for loss computation and sorting (Datasets 2 and 3) introduced greater variability in loss distributions, frequently resulting in bimodal profiles that reflected uneven complexity perception by the model.
- Dataset 1, where sorting and evaluation were performed on the same data, exhibited a unimodal loss distribution and more stable training behavior, suggesting a stronger alignment between curriculum ordering and model familiarity.

Although these exploratory setups are not central to our final experimental conclusions, they were instrumental in shaping our methodological choices. Specifically, they motivated the transition to a larger, multi-year MAESTRO dataset for curriculum learning experiments to ensure consistent complexity evaluation and improved generalization. Full results, figures, and statistical comparisons are provided in Appendix A.

6

Results & Analysis

This chapter presents a comparative analysis of the results obtained from training a baseline model and a curriculum learning (CL) model for symbolic music generation. The evaluation focuses on three key aspects: training and validation performance over time, analysis of loss distributions across the dataset, and musical feature-based metrics extracted from the generated MIDI outputs. Together, these evaluations provide an overview of how curriculum learning influences the learning dynamics and musical quality of the model compared to standard training.

For this analysis, three models are evaluated on the same dataset:

- **Baseline model**: Trained using standard procedures with random batch selection from the full dataset.
- **Curriculum 60% LR model**(CL-60% LR): Trained using curriculum learning where the model reached the full dataset at 60% of the total training batches, with learning rate adjustments during the curriculum transition phases.
- **Curriculum 60% model** (CL-60%): Trained using curriculum learning where the model reached the full dataset at 60% of the total training batches.
- **Curriculum 80% model**(CL-80%): Trained using curriculum learning where the model reached the full dataset at 80% of the total training batches.

6.1 Training Dynamics

Across all four models, the training loss curves ultimately converge to similar final values, with all models achieving training losses around 1.7-1.8 by the end of training (see Figure 6.1). Note that while we compare models based on batch counts for consistency, batches are not directly proportional to computing time. Early in training, all three curriculum learning models (CL-60%, CL-60% LR, and CL-80%) demonstrate significantly lower training losses compared to the baseline model. This is expected, as the curriculum models begin training on a restricted subset of simpler examples before gradually expanding to more complex ones.

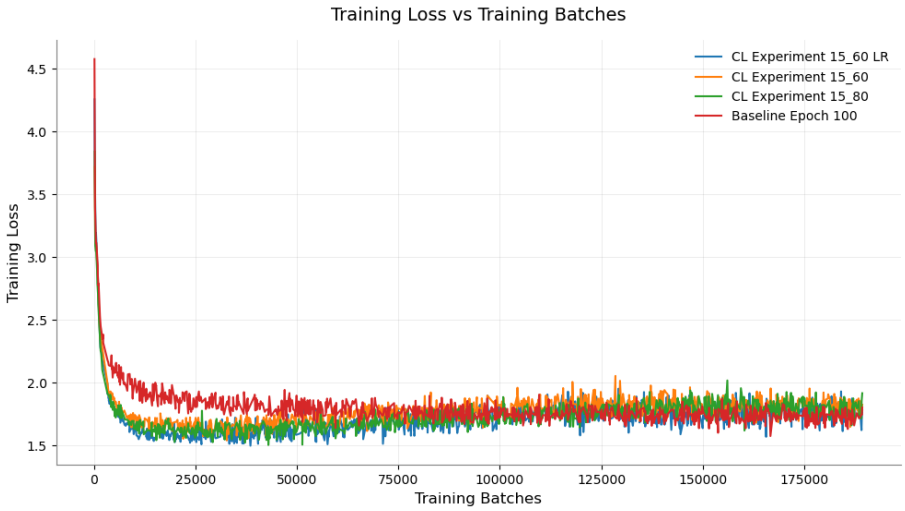


Figure 6.1: Training Losses per batch

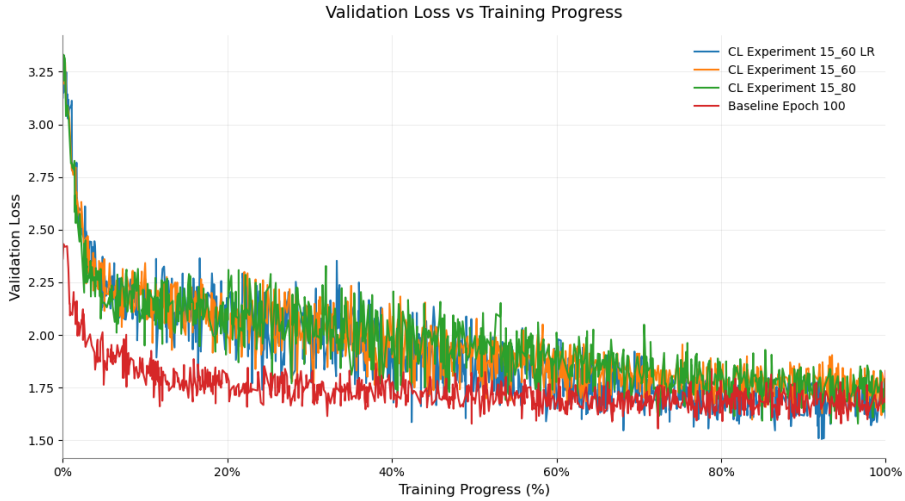


Figure 6.2: Validation Losses for training process

The curriculum learning models maintain their superior training performance through the early and middle stages of training (approximately 0-100,000 batches). However, a distinctive pattern emerges in the later stages: as the curriculum models transition to training on the full dataset, their training losses show slight increases and greater fluctuations, eventually converging with the baseline model's performance. This behavior reflects the models' adaptation to the complete complexity spectrum of the dataset.

The validation loss curves reveal a more consistent pattern across all models (see Figure 6.2). All four models show a clear decreasing trend in validation loss throughout training, with the baseline model achieving the most stable descent.

Despite the baseline model's slower start in training loss, it ultimately achieves the best final validation performance, as shown in Table 6.1. This suggests that the baselines early and consistent exposure to the full dataset complexity, unlike

curriculum learning models, which initially train on subsets, may contribute to its superior generalization by the end of training.

Table 6.1: Final Training and Validation Losses by Model

Model	Final Train Loss	Final Val Loss
CL-60%	1.8039	1.7466
CL-60% LR	1.7549	1.6850
CL-80%	1.7994	1.7443
Baseline	1.7279	**1.6738**

The results present a nuanced picture of curriculum learning's impact on generalization. While the baseline model achieves the lowest final validation loss (1.6738), the curriculum learning models demonstrate notably different training dynamics and competitive performance.

The validation loss curves reveal distinct learning patterns: the curriculum learning models show more variable performance in the early-to-mid training phases, with higher fluctuations compared to the baseline's smoother descent. However, as training progresses, the CL models stabilize and approach performance levels competitive with the baseline. The CL-60% LR model achieves the closest validation performance to the baseline (1.6850), while CL-60% and CL-80% settle at slightly higher but comparable validation losses (1.7466 and 1.7443 respectively).

These fluctuations in the curriculum learning validation curves are partly attributable to the progressive training strategy, where the model continuously adapts to increasingly complex data throughout training. Unlike the baseline model, which sees the full dataset distribution from the beginning, curriculum models must adjust their learned representations as new complexity levels are introduced, leading to the observed variability.

Importantly, despite the baseline achieving the lowest final validation loss, the curriculum learning approaches demonstrate remarkably similar generalization gaps. The difference between training and validation loss remains consistent across all models (ranging from 0.05-0.07), suggesting that curriculum learning does not compromise the model's fundamental ability to generalize to unseen data. Instead, it appears to alter the learning trajectory while maintaining comparable generalization characteristics.

Among the curriculum learning variants, CL-60% LR (which incorporates learning rate adjustments) shows the most promising results, achieving validation performance closest to the baseline while potentially offering other training benefits such as improved stability in the final training phases.

6.2 Loss Distribution Analysis

To gain deeper insights into model performance, the distribution of per-sequence loss values across the entire dataset is examined for both our baseline and curriculum

learning approaches. This analysis reveals how different training strategies affect the model's performance across sequences of varying complexity.

For each model, the per-sequence loss values are calculated for both training and validation sets to analyze their performance distributions.

6.2.1 Loss Statistics Comparison

The comparative statistics of the four models reveal interesting patterns in how curriculum learning affects model performance:

Table 6.2: Mean and Standard Deviation of Training and Validation Losses by Model

Model	**Training Data**		**Validation Data**	
	Mean	Std. Dev.	Mean	Std. Dev.
Baseline	**1.6548**	0.1861	**1.6669**	0.1864
Curriculum 60%	1.6849	0.2297	1.6997	0.2271
Curriculum 60% LR	1.6652	0.2645	1.6853	0.2597
Curriculum 80%	1.7125	0.2631	1.7293	0.2584

As shown in Table 6.2, the baseline model exhibits the lowest average loss values for both training (1.6548) and validation (1.6669) sets, followed closely by the Curriculum 60% LR model (1.6652 training, 1.6853 validation). However, a distinct pattern is observed where curriculum learning models demonstrate notably different loss distribution characteristics compared to the baseline.

The baseline model maintains the smallest standard deviation (0.1861 for training, 0.1864 for validation), indicating highly consistent performance across sequences. In contrast, all curriculum learning models show substantially larger standard deviations, with values ranging from 0.2271 to 0.2645. This increased variance suggests that curriculum learning models exhibit more diverse performance across different sequences, potentially reflecting their ability to handle a broader range of musical complexity levels.

Among the curriculum learning approaches, the Curriculum 60% LR model achieves performance closest to the baseline in terms of mean loss while maintaining the highest standard deviation (0.2645 for training, 0.2597 for validation). The Curriculum 60% model shows intermediate performance with moderate variance (0.2297/0.2271), while the Curriculum 80% model exhibits the highest mean losses (1.7125/1.7293) but similar standard deviation patterns.

The consistent pattern of higher standard deviations in curriculum learning models, despite their competitive mean performance, suggests a fundamental difference in how these models learn to process musical sequences. Rather than converging to uniform performance across all sequences like the baseline, curriculum-trained models appear to develop more specialized responses to different types of musical content, resulting in greater performance variability but potentially more nuanced musical understanding.

6.2.2 Loss Distribution Analysis

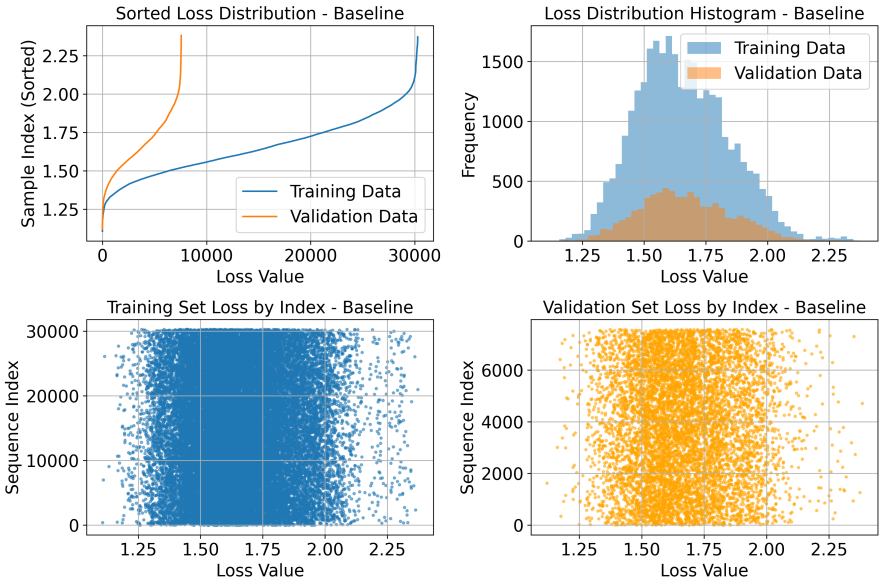


Figure 6.3: Loss distribution analysis for baseline model (standard training)

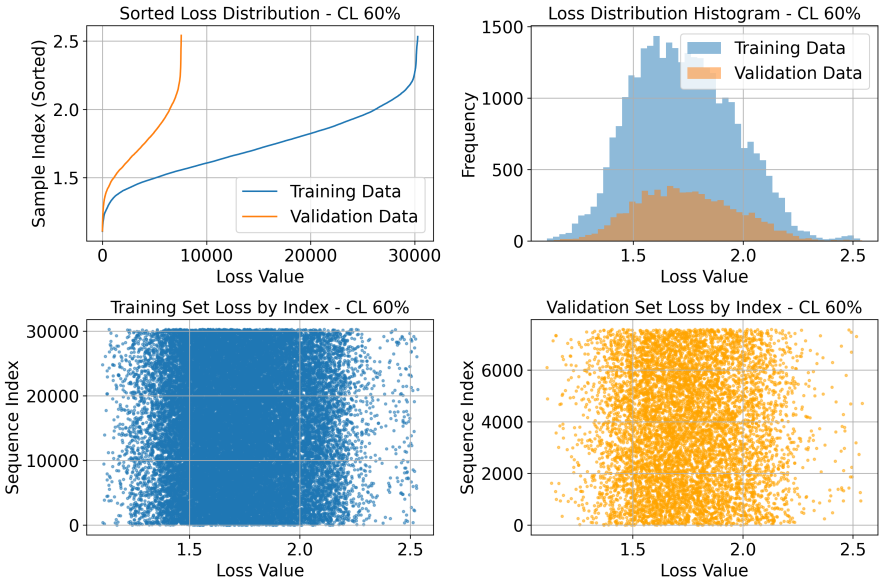


Figure 6.4: Loss distribution analysis for CL-60% model

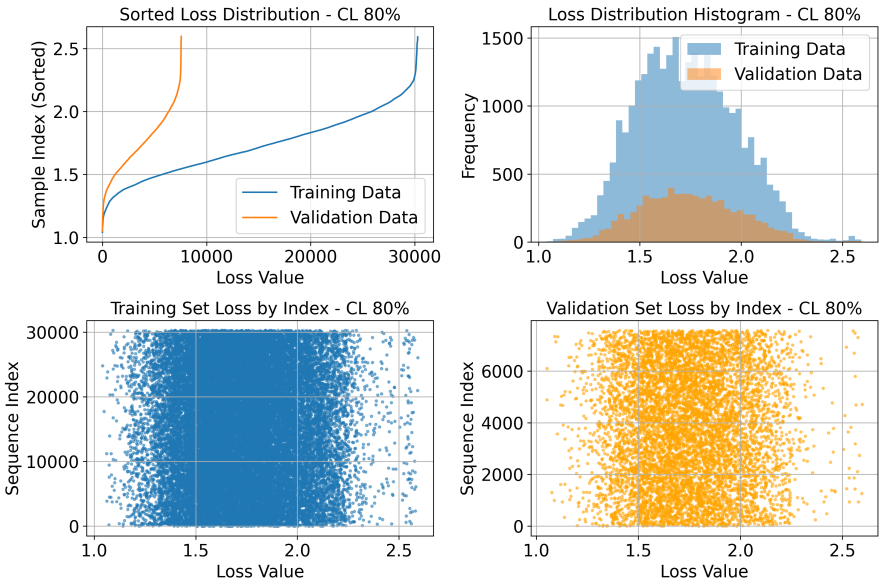


Figure 6.5: Loss distribution analysis for CL-80% model (more gradual curriculum)

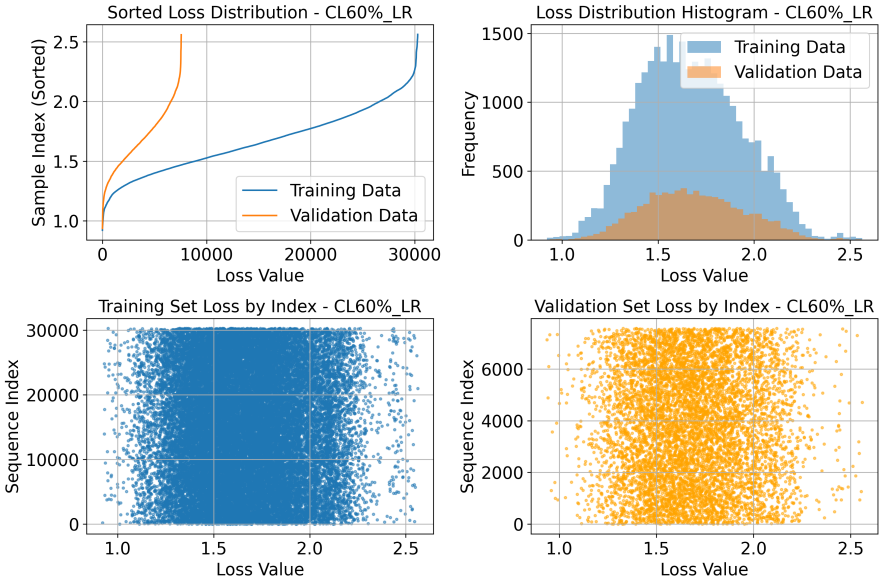


Figure 6.6: Loss distribution analysis for CL-60% LR model.

In Figure 6.3 for the baseline model the sorted loss distribution shows the smoothest progression with the lowest absolute values. The histogram reveals the most concentrated distribution, with sharp peaks around 1.5-1.6 for both training and validation data. The scatter plots demonstrate the lowest variability among all models, with fewer extreme outliers and more consistent performance across sequences. This concentrated pattern reflects the baseline model's uniform exposure to all complexity levels throughout training.

In Figure 6.4 for CL-60% model, the sorted loss distribution is showing gradual increase from easiest to most difficult sequences. Histogram revealing right-skewed

distributions with training data concentrated around 1.5-1.7 and validation data showing similar pattern with slight shift toward higher values. Scatter plots of individual sequence losses demonstrating substantial variability across the dataset, with most sequences clustering in lower loss regions but notable outliers extending beyond 2.5. In Figure6.5 for the CL-80% model the sorted loss distribution shows a similar overall pattern to other curriculum models but with slightly higher absolute values. The histogram displays a broader distribution compared to CL-60%, with both training and validation data showing increased spread. The scatter plots demonstrate greater variability, particularly in the higher complexity sequences, reflecting the impact of the more gradual curriculum transition on model performance consistency. In Figure 6.6 for CL-60% LR model, the sorted loss distribution shows steeper progression in complexity transitions compared to other models. The histogram displays broader, more dispersed distributions with training data spread across 1.2-2.2 range and validation data showing similar but slightly shifted patterns. The scatter plots reveal the highest variability among all models, with substantial spread across sequence complexities, reflecting the impact of learning rate adaptation during curriculum transitions on model performance diversity.

Examining the loss distribution histograms reveals important differences between training approaches. The baseline model shows a more concentrated distribution with a sharper peak, indicating more uniform performance across different sequences. In contrast, the curriculum learning models display broader distributions with more variance.

All models show similar overall trends, with loss values ranging approximately from 1.0 to 2.6 . This suggests that curriculum learning affects how the model performs on sequences of moderate complexity, potentially allowing for better performance on some challenging sequences at the expense of slightly higher average loss.

6.2.3 Training vs. Validation Loss Correlation

A notable observation is the strong correlation between training and validation loss distributions across all models. The mean difference between training and validation loss remains remarkably consistent across different training approaches: approximately 0.012 for the baseline model, 0.015 for CL-60%, 0.020 for CL-60% LR, and 0.017 for CL-80%.

This consistency in generalization gap suggests that while curriculum learning affects the overall performance profile and loss distribution characteristics of the models, it does not fundamentally compromise their ability to generalize to unseen data. All models maintain similar relationships between their training and validation performance, indicating that the benefits or changes introduced by curriculum learning do not come at the cost of generalization capability.

The slightly larger generalization gap observed in the CL-60% LR model (0.020) compared to others may reflect the impact of the learning rate adjustments, but this difference remains within a narrow range that suggests robust generalization across all training approaches. This finding is particularly significant as it demonstrates

that curriculum learning provides a viable training strategy that maintains the fundamental generalization properties of the model while potentially offering other benefits in terms of training dynamics and performance distribution characteristics.

6.2.4 Kullback-Leibler Divergence Analysis

To quantitatively compare the loss distributions of the curriculum learning models with the baseline, two distributional similarity metrics is computed : KullbackLeibler Divergence (KLD) and Overlap Area (OA) see Figure 6.7 and Table 6.3. KLD measures the divergence between two probability distributions, where a lower value indicates a closer match. OA quantifies the overlapping region between two distributions; a higher OA value implies greater similarity.

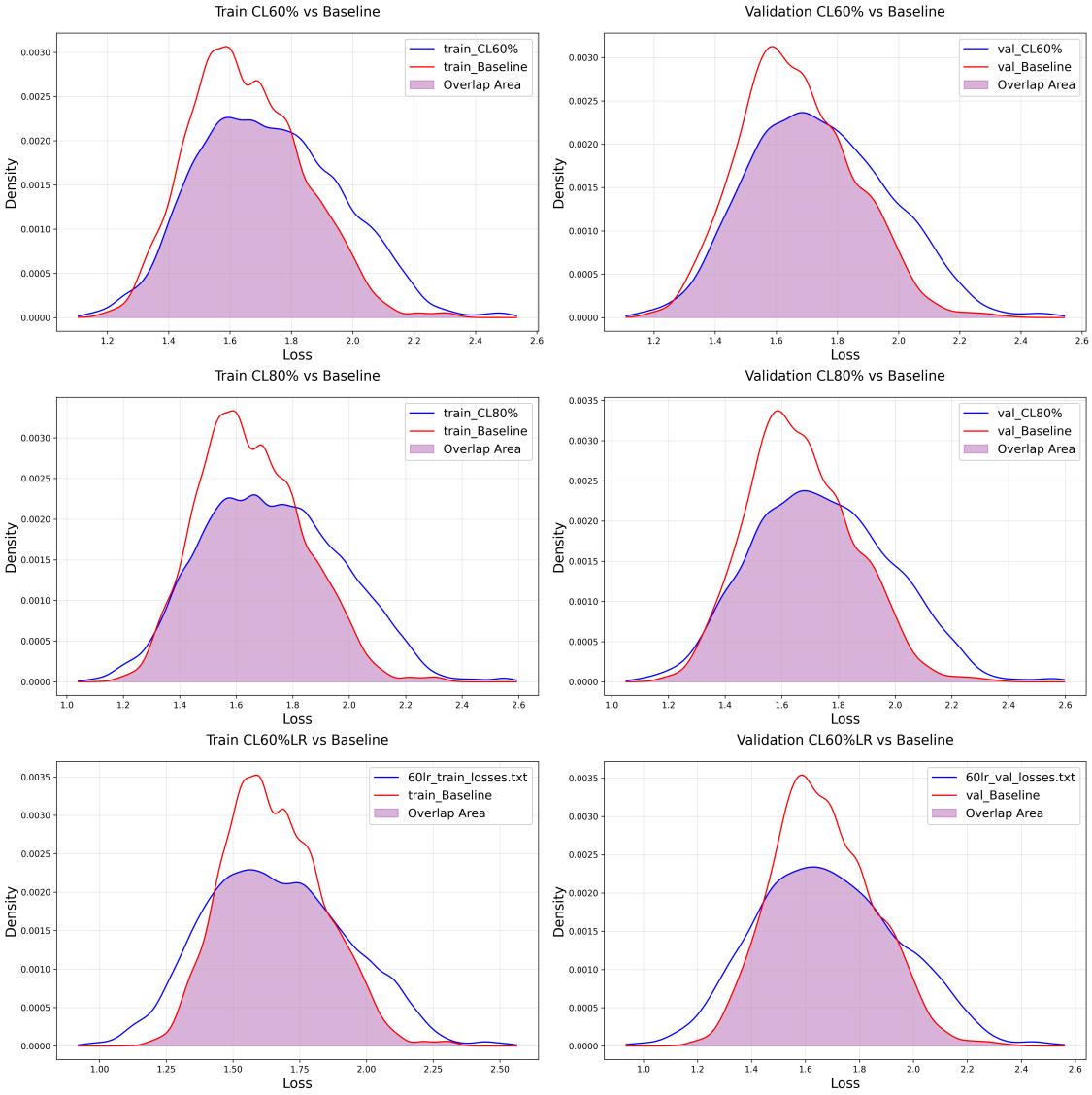


Figure 6.7: Analysis of Loss distributions between Curriculum models vs Baseline

These results reveal distinct patterns in how different curriculum learning approaches

Table 6.3: Distribution Similarity Between Curriculum and Baseline Models

Comparison	**KLD**	**Overlap Area (OA)**
Train CL60% vs Baseline	0.1646	0.8413
Validation CL60% vs Baseline	0.1323	0.8454
Train CL80% vs Baseline	0.2379	0.8221
Validation CL80% vs Baseline	0.1824	0.8289
Train CL60%LR vs Baseline	0.2908	0.8114
Validation CL60%LR vs Baseline	0.2009	0.8302

affect loss distributions relative to the baseline. As shown in Figure 6.8, CL-60% demonstrates lower KLD values (approximately 0.16 for training, 0.18 for validation) and higher overlap area (OA) values (around 0.84 for both training and validation) compared to CL-80% (KLD 0.24 for training, 0.19 for validation; OA 0.82 for training, 0.83 for validation).

Notably, CL-60% LR shows the most substantial divergence from the baseline, with the highest KLD value (approximately 0.28 for training) and lowest overlap area (0.81 for training). This pattern suggests that while CL-60% LR achieves competitive performance metrics, it develops the most distinct loss distribution characteristics compared to standard training.

The scatter plot reveals an interesting trade-off relationship: models with higher distributional similarity to the baseline (higher OA, lower KLD) tend to cluster in the upper-left region, while those with more divergent learning patterns occupy the lower-right space. CL-60% appears to strike a balance, maintaining relatively high similarity to baseline distribution while introducing moderate curriculum-based changes. In contrast, CL-80%'s more gradual curriculum approach produces intermediate divergence, while CL-60% LR's learning rate modifications result in the most distinctive distributional profile.

This analysis suggests that different curriculum learning configurations not only affect final performance but also fundamentally alter how models distribute their performance across sequences of varying complexity, with earlier transition to full dataset (CL-60%) preserving more baseline-like characteristics than more gradual approaches.

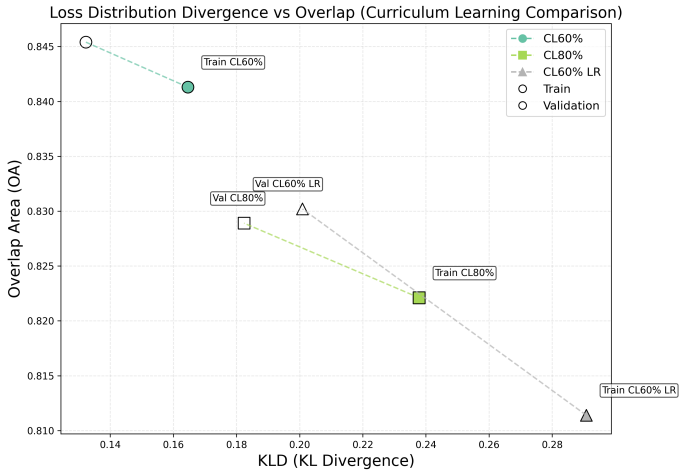


Figure 6.8: KLD-OA Scatter Plot of Loss Distributions

6.3 Musical Analysis

To assess the quality of symbolic music generated by different models, statistical and music-theoretic properties of the generated samples are compared with real-world music data from the MAESTRO dataset. The goal is to measure how closely the generated distributions match real music and to find which features show significant differences, using distributional and statistical testing techniques.

To ensure fair and meaningful comparison between real and generated symbolic music, duration alignment is performed across datasets. The generated music samples, produced by each of the four models (baseline, CL-60%, CL-60%LR, and CL-80%), are approximately 2048 tokens longcorresponding to about 15 to 40 seconds of music depending on the temporal resolution. In contrast,the real music samples from the MAESTRO dataset are full-length performances, often several minutes long, and therefore unsuitable for direct comparison without preprocessing. To address this, 3040-second segments were extracted from 100 different MAESTRO MIDI files by selecting a random start time within each piece and clipping the subsequent window while preserving any overlapping notes. This randomized clipping approach avoids biases from always sampling the beginning of pieces which may include intros or low-complexity segments and instead provides a more representative distribution of musical features.

Once the datasets were ready, a set of symbolic music descriptors were computed for each sample to evaluate musical characteristics at scale. These features include rhythmic activity, pitch diversity, harmonic complexity, and structural regularity, with mathematical formulations and computational details presented in Appendix A.

Table 6.4 presents the average values of various symbolic music features computed across all generated samples. All models exhibit comparable rhythmic intensities, with CL-60% LR producing slightly lower values, suggesting a slightly more sparse rhythmic structure. Pitch entropy and pitch range values are relatively consistent across models, indicating that all architectures capture a reasonable diversity and

Table 6.4: Model Comparison Across Musical Features

Feature	**Baseline**	**CL60**	**CL60LR**	**CL80**	**Maestro**
Rhythmic Intensity	0.6722	0.6318	0.5573	0.6438	0.6707
Pitch Entropy	0.7988	0.8359	0.7948	0.8337	0.8642
Note Density	10.8942	10.6342	9.4803	10.7008	11.6104
Pitch Range	59.41	58.00	58.27	58.85	58.29
Avg Pitch Interval	12.8402	11.8073	10.9275	11.5227	11.1728
Avg IOI	0.1033	0.1064	0.1375	0.1212	0.1077
UPC Mean	6.1025	6.3432	5.6015	6.6386	6.5233
Polyphony	2.7313	3.5573	4.0414	4.2687	2.3351
Empty Bar Ratio	7.4989	5.7452	8.1785	4.7568	2.0294
Qualified Note Ratio	39.2856	49.8201	51.9439	51.7356	42.7770
Qualified Rhythm Freq	73.4209	64.6778	59.6776	62.3121	69.0383
Pitch Consonance Score	0.4035	0.3427	0.3812	0.2772	0.3065
Information Rate	2.1895	2.2020	2.0708	2.0683	2.3016
SSM	0.4003	0.4417	0.4849	0.5088	0.3704
Duration	36	40	48	55	39
Note Count	356	371	340	422	464

Note: Values represent means across all samples. Higher values indicate greater presence of each feature, except for Avg IOI (lower = faster transitions) and Empty Bar Ratio (lower = fewer empty bars). Maestro serves as the reference dataset.

spread of pitches. However, CL-60% LR shows the lowest average pitch interval and highest average IOI, which may imply a more conservative and spaced-out melodic motion compared to other models. In terms of texture, CL-80% and CL-60% LR demonstrate higher polyphony, indicating their tendency to generate richer, more harmonically layered outputs. Notably, CL-60% LR and CL-80% exhibit a higher qualified note ratio, implying better retention of musically meaningful note durations. On the other hand, the Baseline model tends to produce more fragmented compositions, as reflected in its lower qualified note ratio and higher empty bar ratio. Pitch consonance scores are higher in the Baseline and CL-60% LR models, suggesting a preference toward more harmonically stable note combinations. Finally, while information rate and SSM values are relatively similar, CL-80% shows a slightly higher SSM score, possibly reflecting a more patterned or repetitive structural tendency in its generated sequences.

To quantify how closely the distributions of generated symbolic music align with those of real music, Kullback-Leibler (KL) divergence is computed between the feature distributions of each model and those of the real MAESTRO dataset. This divergence measures the dissimilarity between two probability distributionslower values indicate a closer match to real music.

Table 6.5 presents the KL divergence values for eight symbolic music features, comparing the distribution of generated outputs from each model to that of real MAESTRO data. The CL-60% LR model emerges as the most consistent performer, achieving the lowest divergence in polyphony (0.0046), qualified note ratio (0.0041), information rate (0.0017), and qualified rhythm frequency (0.0063). It also shows

Table 6.5: Kullback-Leibler Divergence (KLD) Values Across Models

Feature	**Maestro**	**CL60**	**CL60LR**	**CL80**	**Baseline**	**Best Model**
Pitch Entropy	0	0.0182	**0.0036**	0.0048	0.0120	CL60LR
Polyphony	0	0.0573	**0.0046**	0.0119	0.0279	CL60LR
Note Density	0	0.0272	0.0413	0.0106	**0.0085**	Baseline
Qualified Note Ratio	0	0.0236	**0.0041**	0.0110	0.0152	CL60LR
Pitch Consonance Score	0	**0.0023**	0.0030	0.0062	0.0123	CL60
Information Rate	0	0.0066	**0.0017**	0.0060	0.0038	CL60LR
SSM	0	0.0224	0.0140	0.0630	**0.0109**	Baseline
Qualified Rhythm Freq	0	0.0118	**0.0063**	0.0146	0.0106	CL60LR
Rhythmic Intensity	0	0.0082	0.0067	**0.0030**	0.0051	CL80
Avg Pitch Interval	0	**0.00625**	0.0080	0.0073	0.0129	CL60
Avg IOI	0	0.0273	0.0377	0.0369	**0.0102**	Baseline
Empty Bar Ratio	0	0.0101	0.0460	**0.0091**	0.0436	CL80

Note: KLD values measure divergence from the Maestro reference (lower is better). Values rounded to 8 decimal places. Maestro serves as the ground truth (KLD = 0 for all features).

strong alignment in pitch entropy (0.0036). These results suggest that CL60LR effectively captures both the harmonic richness and temporal structure of the training dataset. CL60 performs best in pitch consonance score (0.0023), indicating it produces harmonically pleasing results more aligned with real music in that specific aspect. CL-80% , demonstrates competitive alignment with real music in rhythmic intensity (0.0030) and empty bar ratio (0.0091), suggesting its outputs maintain a well-balanced rhythmic structure and consistent note activity. The Baseline model achieves the lowest KLD for note density (0.0085) and SSM (0.0109), suggesting some structural similarity to real music but generally higher divergence in core musical features like pitch consonance and qualified note ratio, indicating its limitations in capturing deeper musical structure.

7

Conclusion

7.1 Summary of Findings

All models including the baseline and curriculum learning variants (CL-60%, CL-60% LR, and CL-80%) successfully converged to similar final training loss values, confirming stable optimization across training regimes. The baseline achieved the lowest final validation loss, followed closely by CL-60% LR, indicating strong performance consistency. Importantly, all models exhibited stable generalization gaps, demonstrating that curriculum learning introduces no adverse effects on generalization and does not increase overfitting risk. From the perspective of loss distribution, the baseline model exhibited the most concentrated performance across sequences with the lowest variance , suggesting uniform behavior regardless of sequence complexity. In contrast, all curriculum learning models produced broader distributions with higher standard deviations, indicating more varied and specialized responses to sequences of differing complexity. This was further reflected in the Kullback-Leibler Divergence (KLD) analysis, where CL-60% showed the highest similarity to the baseline, while CL-60% LR exhibited the most divergent distribution, highlighting how curriculum pacing and learning rate modulation shape learning behavior.

These results lead to three key findings. First, curriculum learning fundamentally changes learning dynamics, encouraging specialized behaviors across musical complexity levels rather than merely improving average performance. Second, there is no generalization trade-off curriculum models preserve the generalization capability of the baseline while offering richer performance dynamics. Third, the timing and design of curriculum exposure are crucial. The CL-60% LR configuration, which transitions to the full dataset at 60% of training and includes learning rate adjustments, emerges as the most effective strategy offering a balanced blend of baseline-like stability with the expressiveness enabled by curriculum learning.

Coming to the evaluation of musical output, our study revealed that CL-60% LR achieved the lowest KL divergence values across several symbolic music metrics, including polyphony, qualified note ratio, information rate, and qualified rhythm frequency. These features point to greater harmonic density, richer structural variation, and enhanced rhythmic fluency compared to other models. Curriculum- trained models, particularly CL-60% LR, demonstrated increased musical complexity and expressiveness, producing compositions that exhibited greater pitch diversity, more

layered textures, and stronger alignment with real music characteristics found in the MAESTRO dataset. The higher qualified note ratios also indicate that these models produced more musically meaningful and rhythmically valid note durations. We find that the CL-60%LR model performs well across several symbolic music metrics, but want to be careful in how it is interpreted it's not entirely clear why this happens. One hypothesis is: the CL-60%LR model benefits from earlier full data exposure combined with a more adapted optimizer schedule giving it more chances to generalize and fine-tune, especially for complex musical patterns. We also want to point out that these metrics dont necessarily reflect what listeners perceive as musical or expressive. So while the results are encouraging, we treat them as a sign rather than a final verdict.

7.2 Conclusion

This study investigated the effects of curriculum learning (CL) strategies on symbolic music generation, with a particular focus on training dynamics, loss distribution, and the musical quality of generated outputs. Three types of Curriculum learning models were experimented on:CL 80%, CL 60 % and CL 60 % with adapted learning rate. The results reveal a distinct trade-off between consistency and expressiveness across the models.

First, while the baseline model achieved the lowest final loss and the most concentrated loss distribution, curriculum models introduced greater variance across sequences. This increased variance reflects more specialized responses to musical complexity and was most pronounced in the CL-60% LR model, which emerged as the best performer across several musical featuressuch as polyphony, qualified note ratio, rhythmic structure, and information rate. These findings suggest that although baseline models may appear superior in lossbased metrics, they do not necessarily generate outputs that are richer or more musically compelling.

Second, the results highlight a fundamental limitation of using training loss as a substitute for musical complexity in curriculum design. In the context of symbolic music, higher loss does not always correspond to greater musical complexity; it may instead reflect prediction difficulty. This indicates a need for future curriculum strategies to consider alternative difficulty metricssuch as pitch entropy, harmonic irregularity, or learned musical difficulty embeddingsto better align curriculum progression with musically meaningful complexity.

Importantly, the study emphasizes the role of curriculum design choices in shaping model performance. Specifically, the timing of the transition to full data complexity and the use of learning rate scheduling were shown to significantly affect both training dynamics and musical output. The success of the CL-60% LR configuration demonstrates that moderate curriculum pacing, when combined with tuned optimization dynamics, strikes an effective balance between learning stability and expressive outputs.

That said, this work should be viewed as an investigative study, not a conclusive

benchmark. Curriculum models did not significantly outperform the baseline in raw loss values; however, their training and validation curves showed consistent improvement trends, and the generated music often surpassed the baseline in terms of stylistic diversity and structure. This suggests that extended training over additional epochs, along with broader hyperparameter explorationparticularly in learning rate schedules and curriculum pacing could further enhance performance.

In summary, while the baseline model demonstrated stronger loss convergence, the curriculum models especially CL-60% LR produced outputs that were more expressive, harmonically rich, and structurally aligned with real music. These results support the broader potential of curriculum learning in symbolic music generation and encourage future work to move beyond loss-based evaluation, embracing more music-informed metrics to guide model development.

7.3 Future Work

Future work in curriculum learning for symbolic music generation can explore several directions. One key area is the development of learned difficulty metrics. While this study relied on model loss as a substitute for complexity, future research should investigate representations that better reflect musically meaningful featuressuch as harmonic irregularity, pitch entropy, or rhythmic unpredictability. Embedding-based approaches, such as learning semantic embeddings of musical input[42], offer a more musically grounded alternative to loss-based ranking. Another promising direction lies in multi-modal curriculum learning. As symbolic music models increasingly include inputs such as audio, lyrics and textual or visual prompts , curriculum strategies can be extended to beyond symbolic-only sequences. Recent models like VT2Music[43] illustrate how symbolic and non-symbolic embeddings can be integrated in training pipelines. Finally, beyond static scheduling, research can explore meta-learning approaches[29], [44] where the curriculum policy itself is learned. Such adaptive strategies could customize training progression for specific music styles, tasks, or even individual model checkpoints. Together, these advancements could lead to curriculum systems that are not only more effective but also more musically aware and expressive.

8

Ethics and Acknowledgements

This research involves several ethical considerations that warrant careful attention. First, regarding the use of AI tools in the preparation of this Planning report, Claude was used for paraphrasing and Chatgpt was used for enhancing the clarity of sentences, with all AI-generated content being carefully reviewed and validated.

The research utilizes the MAESTRO dataset[37], which is available under a Creative Commons Attribution Non-Commercial Share-Alike 4.0 license. This license restricts the use of generated music to non-commercial applications and requires appropriate attribution. We acknowledge the creators and contributors of this dataset, including the International Piano-e-Competition organization.

The source code for this project is adapted from the open-source repository `music-transformer` [1], developed by Aditya Gomatam. The repository is licensed under the GNU General Public License v3.0. The following copyright and license notice applies:

> © 2021 Aditya Gomatam.
>
> This file is part of `music-transformer`, an open-source project to build and train a Music Transformer. It is licensed under the GNU General Public License v3.0, which allows redistribution and modification under the same license terms. The project is provided without warranty of any kind. Full license details are available at `https://www.gnu.org/licenses/gpl-3.0.html`.

We thank Aditya Gomatam for making this code publicly available and acknowledge that our implementation builds upon and significantly extends the original codebase as part of our thesis project.

Additionally, we recognize the potential societal implications of AI-generated music, particularly regarding intellectual property rights and the impact on human musicians. While this research focuses on technical aspects of music generation, we maintain awareness of these broader ethical considerations throughout our work.

[1] `https://github.com/spectraldoy/music-transformer`

Bibliography

[1] E. A. Platanios, O. Stretcu, G. Neubig, B. Poczos, and T. M. Mitchell, "Competence-based curriculum learning for neural machine translation," *arXiv preprint arXiv:1903.09848*, 2019.

[2] Ç. Gülçehre and Y. Bengio, "Knowledge matters: Importance of prior information for optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 226–257, 2016.

[3] B. Yu, P. Lu, R. Wang, *et al.*, "Museformer: Transformer with fine-and coarse-grained attention for music generation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1376–1388, 2022.

[4] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, *et al.*, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[6] P. P. Li, B. Chen, Y. Yao, Y. Wang, A. Wang, and A. Wang, "Jen-1: Text-guided universal music generation with omnidirectional diffusion models," in *2024 IEEE Conference on Artificial Intelligence (CAI)*, IEEE, 2024, pp. 762–769.

[7] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep learning techniques for music generation – a survey*, 2019. arXiv: `1709.01620 [cs.SD]`. [Online]. Available: `https://arxiv.org/abs/1709.01620`.

[8] A. van den Oord, S. Dieleman, H. Zen, *et al.*, *Wavenet: A generative model for raw audio*, 2016. arXiv: `1609.03499 [cs.SD]`. [Online]. Available: `https://arxiv.org/abs/1609.03499`.

[9] D. Cope, "Computer modeling of musical intelligence in emi," *Computer Music Journal*, vol. 16, no. 2, pp. 69–83, 1992.

[10] D. Ponsford, G. Wiggins, and C. Mellish, "Statistical learning of harmonic movement," *Journal of New Music Research*, vol. 28, no. 2, pp. 150–177, 1999.

[11] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: A steerable model for bach chorales generation," in *International conference on machine learning*, PMLR, 2017, pp. 1362–1371.

[12] F. Liang, "Bachbot: Automatic composition in the style of bach chorales," *University of Cambridge*, vol. 8, no. 19-48, pp. 3–1, 2016.

[13] S. Ji, X. Yang, and J. Luo, "A survey on deep learning for symbolic music generation: Representations, algorithms, evaluations, and challenges," *ACM Computing Surveys*, vol. 56, no. 1, pp. 1–39, 2023.

[14] K. Bhandari, G. A. Wiggins, and S. Colton, "Yin-yang: Developing motifs with long-term structure and controllability," *arXiv preprint arXiv:2501.17759*, 2025.

[15] S. H. Hakimi, N. Bhonker, and R. El-Yaniv, "Bebopnet: Deep neural models for personalized jazz improvisations.," in *ISMIR*, 2020, pp. 828–836.

[16] U. M. Packet, "Midi 2.0,"

[17] A. Kalos, "Modeling midi music as multivariate time series," in *2006 IEEE International Conference on Evolutionary Computation*, 2006, pp. 2058–2064. DOI: 10.1109/CEC.2006.1688560.

[18] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020.

[19] D. Herremans, C.-H. Chuan, and E. Chew, "A functional taxonomy of music generation systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–30, 2017.

[20] K. Chen, Y. Wu, H. Liu, M. Nezhurina, T. Berg-Kirkpatrick, and S. Dubnov, "Musicldm: Enhancing novelty in text-to-music generation using beat-synchronous mixup strategies," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2024, pp. 1206–1210.

[21] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International conference on machine learning*, PMLR, 2018, pp. 4364–4373.

[22] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," *arXiv preprint arXiv:1604.08723*, 2016.

[23] J. S. Downie, "Music information retrieval," *Annual review of information science and technology*, vol. 37, no. 1, pp. 295–340, 2003.

[24] M. Schedl, E. Gómez, J. Urbano, *et al.*, "Music information retrieval: Recent developments and applications," *Foundations and Trendső in Information Retrieval*, vol. 8, no. 2-3, pp. 127–261, 2014.

[25] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[26] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," *arXiv preprint arXiv:2009.09761*, 2020.

[27] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer," *arXiv preprint arXiv:1809.07600*, 2018.

[28] G. Hadjeres and F. Nielsen, "Interactive music generation with positional constraints using anticipation-rnns," *arXiv preprint arXiv:1709.06404*, 2017.

[29] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 4555–4576, 2021.

[30] M. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," *Advances in neural information processing systems*, vol. 23, 2010.

[31] D. Weinshall, G. Cohen, and D. Amir, "Curriculum learning by transfer learning: Theory and experiments with deep networks," in *International conference on machine learning*, PMLR, 2018, pp. 5238–5246.

[32] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1706.03762.

[33] S. Singh, K. Patel, P. Bhattacharyya, K. Bhattacharjee, H. Darbari, and S. Verma, "Does curriculum learning help deep learning for natural language generation?" In *Proceedings of the 15th International Conference on Natural Language Processing*, 2018, pp. 92–98.

[34] C. Gong, D. Tao, S. J. Maybank, W. Liu, G. Kang, and J. Yang, "Multi-modal curriculum learning for semi-supervised image classification," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3249–3260, 2016.

[35] F. Loyola, "Polyphonic music generation using neural networks," M.S. thesis, Universitat Politècnica de Catalunya, 2021.

[36] M. Boi and M. Horvat, "A survey of deep learning audio generation methods," *arXiv preprint arXiv:2406.00146*, 2024.

[37] C. Hawthorne, A. Stasyuk, A. Roberts, *et al.*, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1lYRjC9F7.

[38] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.

[39] S.-L. Wu and Y.-H. Yang, *Musemorphose: Full-song and fine-grained piano music style transfer with one transformer vae*, 2022. arXiv: 2105.04090 [cs.SD]. [Online]. Available: https://arxiv.org/abs/2105.04090.

[40] Y.-C. Yeh, W.-Y. Hsiao, S. Fukayama, *et al.*, *Automatic melody harmonization with triad chords: A comparative study*, 2021. arXiv: 2001.02360 [cs.SD]. [Online]. Available: https://arxiv.org/abs/2001.02360.

[41] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015, vol. 5.

[42] M. Bretan, S. Oore, J. Engel, D. Eck, and L. Heck, "Deep music: Towards musical dialogue," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[43] J. Zheng, M. Cao, and C. Zhang, "Vt2music: A multimodal framework for text-visual guided music generation and comprehensive performance analysis," *IEEE Access*, 2025.

[44] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," in *international conference on machine learning*, Pmlr, 2017, pp. 1311–1320.

A

Appendix

A.1 Dataset Evaluation and Complexity Outline

In parallel with the curriculum learning experiments, we explored multiple dataset construction strategies for curriculum learning using sequence-wise loss representing the complexity .

- Dataset 1: Loss computed and sorting performed using the same dataset that the pretrained model was originally trained on.
- Dataset 2 & 3: Sorting performed using loss values computed by the pretrained model, but on two different datasets.

A.1.1 Loss Range Analysis

The distribution of sequence-wise loss values varied significantly across datasets:

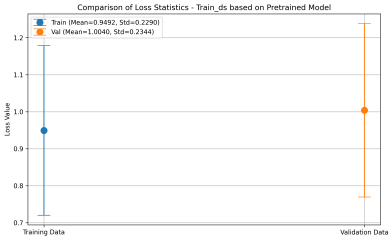
- Dataset 1: Both training and validation loss values were tightly concentrated in the range of 0.4 to 1.8.
- Datasets 2 & 3: These showed a much broader distribution, with loss values ranging from 0.8 to 4.5.

A.1.2 Statistical Comparison

From the table A.1 and the Figures A.1a and A.1b we could see that the training and validation splits are well-aligned within each dataset with similar mean and standard deviation values within each dataset.

Dataset	**Train Mean ± Std**	**Val Mean ± Std**
Dataset 1	0.95 ± 0.23	1.00 ± 0.23
Dataset 2	2.53 ± 1.04	2.47 ± 1.03
Dataset 3	2.58 ± 1.07	2.56 ± 1.08

Table A.1: Mean and standard deviation values for training and validation across datasets.



(a) Dataset 1

(b) Dataset 2

Figure A.1: Comparison of training performance on different datasets

A.1.3 Dataset-wise Loss Distribution

To further evaluate our dataset we visualized the loss distributions of all three datasets using the same pretrained baseline model. Interestingly, **Dataset 1**, where the sorting and evaluation were both performed on the same data, showed a unimodal (single-peaked) distribution. This suggests a more consistent complexity evaluation, likely due to the model being better familiarized with that specific data.

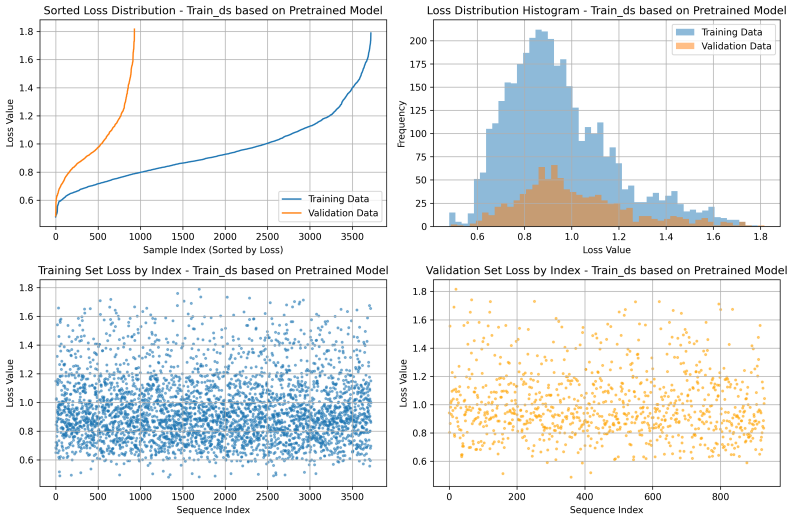


Figure A.2: Loss distribution of Dataset 1

In contrast, **Datasets 2 and 3**, where sorting was done on different data than evaluation, showed bimodal (two-peaked) loss distributions. This may reflect the model's confidence when predicting sequences that are less familiar. One mode corresponds to simpler, more predictable sequences (low loss), while the other captures more complex, harder-to-predict segments(higher loss). The bimodal shape can indicates that theres meaningful variance in how the model evaluates complexity.

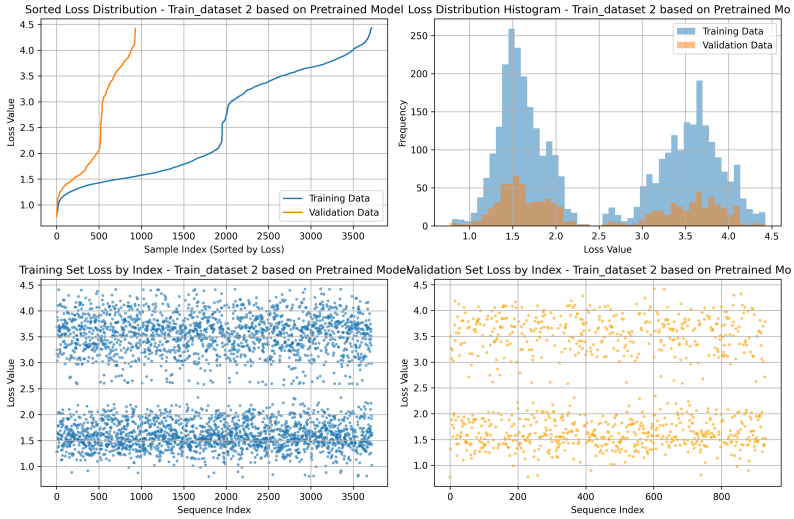


Figure A.3: Loss distribution of Dataset 2

A.2 Preliminary Curriculum Learning Experiment Results and Analysis

A.2.1 Curriculum Learning Implementation

This curriculum learning implementation builds on the theoretical framework from Section 3.2 through a carefully designed progressive training strategy. The system applies a linear curriculum schedule, expanding the training dataset from 20% to 100% over the course of training epochs. For each epoch, a subset of the training data is created with size proportional to current training percent. Subsets are drawn sequentially from pre-sorted data (low to high loss) to ensure gradual complexity progression.

To maintain fair comparison with the baseline model, we ensured that the total number of parameter updates (batches) remained consistent across experiments. Specifically, the baseline model trained for 100 epochs with the full dataset resulted in approximately 23,300 batches. Therefore, we set a limit on the total number of batches in the curriculum learning setup to this same target.
Since curriculum learning begins with smaller subsets and gradually includes more data, we adopted a strategy that runs for a fixed large number of epochs while linearly increasing the subset size from 20% to 100%. At each epoch, the subset of training data is selected from the beginning of the pre-sorted dataset (easiest to hardest), proportional to a computed progress ratio:

$$current\_percent = min(0.2 + 0.8 * \frac{batches\_processed}{TARGET\_BATCHES}, 1.0) \tag{A.1}$$

This approach ensures that early epochs see only simpler sequences thereby supporting gradual learning. Also all sequences are eventually seen, as the subset grows to 100%. The curriculum training stops automatically once the total number of batches reaches 23,300, aligning with the baseline models training.

A.2.2 Experimental Setup

In this experiment, we implemented a curriculum learning approach to evaluate its impact on model training efficiency and performance. The experimental design included:

- Baseline Model: Trained for 100 epochs using Dataset 1 with standard training protocol Curriculum Learning Models:
- Version 1: Trained on Dataset 1 for 23,300 batches using curriculum learning Version 2: Trained on Dataset 2 for 23,300 batches using curriculum learning
- Curriculum Structure: Training examples were sorted by complexity using a pretrained model that was previously trained on Dataset 1 for 300 epochs

A.2.3 Results

Training Performance Analysis By Batches

The three approaches showed similar convergence patterns in training:

- Baseline Training: The baseline model showed steady training loss reduction, starting around 3.2 and gradually decreasing to about 1.5 by the end of training. The curve exhibited minimal fluctuations after step 15,000, demonstrating stable convergence.
- Curriculum Learning with Dataset 1: This model showed a rapid initial decrease in training loss, with similar early performance to the other approaches. While its loss reduction tracked closely with curriculum learning on Dataset 2 until approximately step 5,000, it subsequently displayed higher variance with frequent small oscillations. The final training loss settled around 1.7-1.8, positioning it between the other two approaches.
- Curriculum Learning with Dataset 2: This approach started with slightly higher initial loss (3.4) compared to the baseline. While following a similar early path, it maintained higher training loss throughout and showed more pronounced fluctuations, including occasional spikes in loss (notably around step 20,000). Final training loss stabilized around 1.8-1.9.

Validation Performance Analysis By Batches

The validation metrics revealed distinct patterns across the three training approaches:

- Baseline Training: Exhibited a standard exponential decay in validation loss, gradually decreasing from approximately 3.8 to 1.4 over 23,000 steps.
- Curriculum Learning with Dataset 1: Showed the most rapid initial convergence, with validation loss dropping sharply in the first 2,000 steps. This approach maintained comparable performance to the baseline model in later stages, eventually converging to a similar final validation loss value.
- Curriculum Learning with Dataset 2: Demonstrated an atypical learning pattern with an initial decrease in validation loss followed by a significant

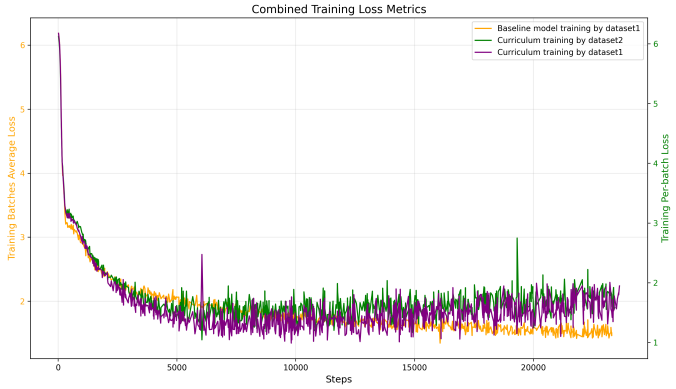


Figure A.4: Training Loss

increase around step 5,000. The loss subsequently decreased gradually but remained consistently higher than the other approaches, ending approximately at a loss value of 2.0 .

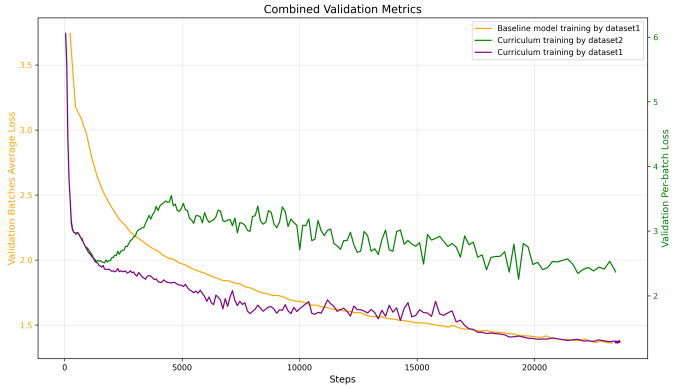


Figure A.5: Validation Loss

Training Performance Analysis By Epochs

The training loss curves showed distinct learning behaviors across the three approaches:

- Baseline Training: Shows rapid initial convergence, decreasing from approximately 6.2 to 1.5 over 100 epochs. The baseline approach consistently maintains lower training loss during the first 50 epochs compared to both curriculum learning approaches.
- Curriculum Learning with Dataset 1: Demonstrates a more gradual descent compared to the baseline in early epochs but achieves the lowest final training loss by epoch 200, suggesting excellent optimization in later training stages.
- Curriculum Learning with Dataset 2: Follows a nearly identical pattern to Experiment 1 for the first 100 epochs, but shows slightly higher loss values in later epochs, converging to approximately 1.6 by the end of training.

Figure A.6: Training Loss

ValidationPerformance Analysis By Epochs The validation metrics shows notable differences in generalization capability:

- Baseline Training: Exhibits a steady exponential decay in validation loss, gradually decreasing from approximately 3.8 to 1.4 over 100 epochs.
- Curriculum Learning with Dataset 1: Initially shows higher validation loss than the baseline but achieves consistent improvement throughout training, eventually reaching the lowest validation loss by epoch 200. The consistent downward trend suggests the model may benefit from additional training epochs.
- Curriculum Learning with Dataset 2: Demonstrates an learning pattern with an initial decrease in validation loss until epoch 30, followed by a significant increase peaking around epoch 50. Though the loss gradually decreases thereafter, it remains higher than both alternative approaches, suggesting potential overfitting to certain curriculum subsets.

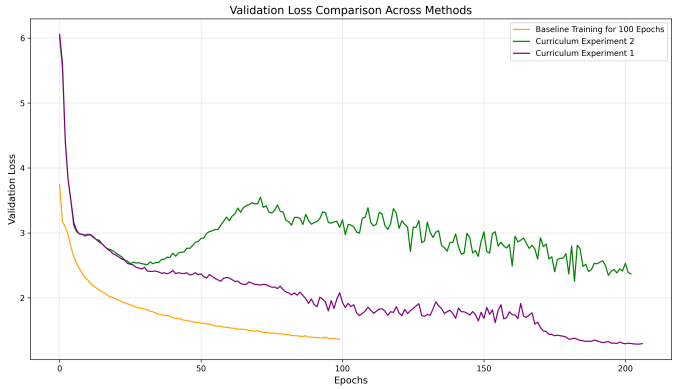


Figure A.7: Validation Loss

A.3 Next Steps

To build upon the preliminary findings and fully realize the potential of curriculum learning in this setting, these are the next steps we will take:

- **Extended Training:** The current results are based on relatively short training cycles. We plan to continue training the model for 300+ epochs to better evaluate the long-term benefits of curriculum learning and its impact on model generalization and sequence quality.
- **Model Evaluation Phase:** Following extended training we will initiate the evaluation phase which will focus on three key areas: Basic Quality Metrics, Musical Structure and Generalization Ability.
- **Dataset Distribution Improvements:** We will modify our dataset splitting function to ensure training and validation datasets have similar loss distributions. This modification will provide more reliable validation metrics and prevent potential biases in our evaluation.

- **Dataset Expansion:** For the baseline, we will use more sequences by incorporating six years of data from the MAESTRO dataset. This expansion will provide a more comprehensive baseline and potentially improve model performance
- **Reproducibility Enhancements:** We will implement fixed random seeds across all preprocessing steps to ensure reproducibility of our experiments and enable more accurate comparisons between different approaches.

B

Appendix

- **Rhythmic Intensity**: Measures the percentage of beats with at least one note onset, providing a normalized score between 0 and 1 that indicates how rhythmically active the music is. Higher values indicate greater rhythmic density.
- **Pitch Entropy**: A measure of pitch distribution diversity calculated using a normalized 12-dimensional pitch class histogram. Higher values (closer to 1) indicate more uniform use of pitch classes across the octave, while lower values suggest focus on fewer pitch classes.
- **Note Density**: The average number of notes per second, calculated as the total note count divided by the duration in seconds. This metric quantifies how busy or sparse the musical texture is.
- **Pitch Range**: The interval in semitones between the lowest and highest notes in the composition. Larger values indicate a broader pitch range being explored.
- **Avg. Pitch Interval**: The mean absolute distance in semitones between consecutive pitches in the sequence. Larger values indicate more disjunct melodic motion (larger jumps between notes).
- **Avg. IOI (Inter-Onset Interval)**: The average time in seconds between consecutive note onsets. Lower values indicate faster or more rhythmically dense passages.
- **UPC (Used Pitch Classes) Mean**: The average number of different pitch classes (from the 12 possible classes in an octave) used per bar. Higher values indicate greater pitch variety within each bar.
- **Polyphony**: The average number of simultaneous notes being played at any given moment, evaluated only at time steps where at least one note is active. Higher values indicate denser harmonic texture.
- **Pitch Consonance Score**: Heuristic of harmonic consonance based on interval class.
- **Qualified Rhythm Frequency**: % of rhythms fitting common metric divisions
- **Qualified Note Ratio**: % of notes with musically meaningful durations

- **Empty Bar Ratio**: % of bars with no notes
- **Information Rate**: The average amount of information (in bits) conveyed per musical event or time unit, measuring the entropy or unpredictability of the musical sequence.
- **SSM(Self-Similarity Matrix)**: A computational analysis tool that measures temporal structure by comparing each musical segment with every other segment in the composition, creating a matrix of similarity scores.
- **Duration (sec)**: The total playback duration of the composition in seconds.
- **Note Count**: The total number of note events in the composition.