Autoregressive Customizable Bach Chorale Generation Using a Lightweight Hybrid Model

Umut Olmez

CMU-CS-25-157
December 2025

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Roger Dannenberg, Chair
Chris Donahue

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Keywords: Bach Chorales, Autoregressive Sequence Modeling, Symbolic Music Generation, Temporal Convolutional Neural Network, Transformer, Self Attention, RoPE

To my dear family and friends.

Abstract

Music, especially Bach's music which is a staple in the teaching of fundamentals of musical composition, exhibits various patterns and rules throughout various parameters, such as harmony, melody and structure, that give it a defining structure when we listen to it and make it different than just noise. Many of these parameters are simplified in Bach chorales, making them great for studying the fundamentals of tonal composition and making them a perfect candidate to be generated by a computer. Looking at both the earlier autoregressive methods and the recent hybrid models, this thesis produced a lightweight and customizable hybrid model that can take in any initial sequence of chords and start generating a chorale on top of it. With an augmented symbolic dataset of 4584 chorales, using a TCNN and a transformer with self attention and RoPE, the model captures local voice leading patterns and more global structural cues to generate a chorale autoregressively. The final model was able to train under 1.5 hours in most cases and is very easy to customize with adjustable layer numbers, model dimension, kernel size and dilation, and extra steps during fusion of transformer and TCNN layers. The generated chorales exhibited interesting musical material, some mimicking Bach's style successfully through 2 bar phrases that end sensibly, 4 bar phrases with half cadences and perfect cadences, modulations to the correct relative key, and idiomatic four-part writing patterns. But it was also observed that the chorales started to degenerate especially after the first few phrases. The final model can be used by students or musicians being introduced to four-part writing as an assistant and is even able to create interesting ideas for experienced composers.

Acknowledgments

First and foremost, I would like to thank my advisor, Roger Dannenberg, for his otherworldly patience with me and not giving up on me during some of the busiest and most uncertain periods of my life that happened to occupy the same time frame with my thesis work. I further thank him for his guidance in the area of symbolic music composition and for introducing me to the research that formed the basis of this work.

I also thank Chris Donahue for accepting to be a committee member with short notice and thank Angy Malloy for her support throughout the entire process.

Finally I present my deepest gratitude to my family and close friends. I would not be able ot come this far without them backing me and supporting me. Thank you my parents, my sister, Arka Sokaklar and Yingyi Zhang.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Motivation

Music, especially Bach's music which is a staple in the teaching of fundamentals of musical composition, exhibits various patterns and rules that give it a defining structure when we listen to it and make it different from just noise. When notes are played at the same time we get chords and when we get multiple chords that follow each other we get harmonic progressions. Depending on the specific notes inside the chords, the rhythms, the non-chord tones and their placement, the timbres being used, and other characteristics such as melody, texture and dynamics we get different styles of music with different sets of rules and patterns. Bach chorales are a type of music where many of these parameters are simplified which makes them great for studying the fundamentals of tonal composition like harmony, voice leading and phrase structure. This also makes them a perfect candidate to be generated by a computer. A process where the aim is to produce original musical material by learning statistical and structural patterns from existing compositions.

Traditional algorithmic composition relies heavily on rule-based systems grounded in music theory[1]—for example, implementations of species counterpoint or harmonic rule sets designed to imitate Bach's style. These systems offer interpretability but struggle to generalize beyond explicitly encoded rules. Deep learning based methods are an alternative rule based algorithmic composition with its ability to learn musical structure directly from data without manual rule specification. The music data used for generation typically falls into two categories: symbolic representations and audio representations. Symbolic music data encodes musical events—pitches, durations, chords, and voice assignments—through formats such as MIDI, MusicXML, piano-roll matrices, or custom token sequences. Symbolic representations are especially useful for modeling the compositional and harmonic structure of polyphonic works, such as Bach chorales, and enable models to operate at the level of notes and voices rather than raw sound. Audio representations, in contrast, capture timbre, dynamics, and expressive performance through waveforms or spectrograms. These representations support generative tasks focused on sound design and expressive realism but require significantly larger models and datasets due to the high dimensionality and temporal density of audio signals.

Different architectures and methods have been used for both symbolic and audio data based music generation throughout the decades. This paper focuses on symbolic music generation with a custom representation scheme for a specific type of music, which allows for a small and lightweight customisable hybrid model to achieve interesting generated musical material.

1.2 Related Work

As mentioned earlier, music generation methods have evolved significantly over time, moving from early rule-based systems and probabilistic models to neural network based models like RNNs and LSTMs, which enabled data driven model estimation, but still struggled with long-range structure. Then convolutional approaches were introduced, which excelled at capturing local voice-leading and harmonic patterns through large receptive fields and parallel training. The field advanced further with Transformer-based architectures, where self-attention enabled global context modeling and began to show long-term musical coherence. Recent research has shifted toward hybrid and hierarchical models that combine different architectures or incorporate explicit structural representations to manage both local detail and large-scale form to better capture music's multi-level organization.

An early paper is by Allan and Williams (2005) [2], where they present one of the earliest data-driven approaches to Bach chorale harmonization using a Hidden Markov Model (HMM) framework, where the observed sequence is the soprano melody and the hidden states represent harmonic functions and chord voicings. By modeling chorales at a beat-level grid and representing harmonies through pitch-interval structures augmented with harmonic labels, their system learns transition and emission probabilities directly from annotated Bach chorales, enabling full harmonization through Viterbi decoding and alternative realizations via posterior sampling. A secondary HMM adds ornamentation at a finer temporal resolution to simulate passing notes and melodic motion. Quantitative evaluation using negative log-probabilities shows that their full HMM outperforms simpler Markov models and independent baselines in predicting Bach's harmonic sequences. Their work demonstrates that probabilistic sequence modeling can learn harmonic rules from examples, providing a foundational predecessor to modern neural generative models.

More than a decade later DeepBach (Hadjeres et al., 2017) [3] introduces a steerable graphical model for generating four-part Bach-style chorales using a pseudo-Gibbs sampling procedure rather than left-to-right sequential generation. The model represents each voice separately and conditions predictions on both local musical context and metadata such as beat subdivisions and fermata positions, allowing it to capture characteristic rhythmic and harmonic features of Bach's style. Its architecture combines forward and backward LSTMs with feedforward networks to model past, future, and simultaneous notes, enabling flexible conditioning and high-quality reharmonization. DeepBach allows users to fix specific notes, voices, or cadences, and the system regenerates the remaining parts while respecting these constraints. Human evaluation results show that listeners often judge DeepBach outputs as convincingly Bach-like, outperforming Maximum Entropy and MLP baselines. Overall, DeepBach demonstrates that Gibbs-style iterative sampling paired with local context modeling can generate coherent, stylistically faithful polyphonic music with strong user control.

That same year Google Magenta used a different method to approach Bach chorale generation with COCONET (Huang et al., 2017) [4], a deep convolutional generative model that completes and rewrites polyphonic musical scores, particularly Bach chorales, using an orderless NADE formulation that enables conditioning on arbitrary subsets of notes. Instead of generating music chronologically, COCONET treats composition as a nonlinear inpainting task and uses blocked Gibbs sampling to iteratively refine partial scores, just like a professional composer might do, allowing the model to correct earlier mistakes and achieve more coherent contrapuntal structure. The convolutional architecture exploits the local, translation-invariant nature of contrapuntal rules while remaining flexible enough to capture broader polyphonic textures. Quantitatively, the model achieves strong framewise negative log-likelihoods across multiple temporal resolutions, and human listening tests show that Gibbs-sampled outputs are perceived as significantly more Bach-like than samples from ancestral NADE procedures. COCONET demonstrates that convolutional architectures, combined with iterative sampling mimicking a composer, can effectively model harmonic and voice-leading constraints in multi-voice symbolic music.

Not long after came Music Transformer (Huang et al., 2018) [5] which extended the standard Transformer architecture to symbolic music generation by introducing a memory-efficient relative positional attention mechanism that better captures musical relationships such as relative timing and recurring motifs. The model operates on both grid-based chorale data and event-based expressive piano performances, enabling sequence lengths far longer than previous RNN-based approaches. By reducing the memory cost of relative attention from O(L^2D) to O(LD), the authors demonstrated coherent minute-long generations that preserve phrase structure, timing regularity, and motif development. Across datasets, the model achieved state-of-the-art negative log-likelihood and is consistently preferred in human listening tests over baseline Transformers and LSTMs. Music Transformer showed that relative attention improved modeling large-scale musical form, establishing a foundation for architectures that aim to balance local voice-leading and global harmonic structure.

In recent years, many hybrid music generation models that employ different techniques emerged. One such example is MusicFrameworks (Dai et al., 2021) [6] which proposes a controllable deep melody generation system built on a hierarchical music structure representation that encodes section-level and phrase-level repetition patterns, basic melodies, and basic rhythmic forms. Instead of relying solely on autoregressive next-note prediction, the method decomposes melody generation into sub-tasks—basic melody creation, rhythm pattern generation, and final melody realization—all conditioned on interpretable musical features such as contour, positional metadata, and chord context. Each sub-process uses a Transformer-LSTM architecture to learn local dependencies while maintaining global structural coherence. This design enables strong user controllability: one may freely edit or mix structural components (e.g., rhythm templates, phrase contours) to shape the generated output. Objective evaluations showed significant improvements when using hierarchical frameworks compared to direct melody generation, and subjective listening tests revealed that roughly half of the generated melodies are rated as good as or better than human-composed POP909 melodies. MusicFrameworks demonstrated that explicit hierarchical structure and controllable intermediate representations greatly improve long-term consistency and user-guided generation in symbolic music modeling.

1.3 Goal and Contribution

This paper decides to step back a little bit, looking at not only the recent hybrid models and custom methods that are being employed, but also the older autoregressive simpler models. Observing instrumentalists in music schools and how they approach introductory four-part writing courses revealed that non-composition students have little experience with composing four-part chorale style music and struggle with it more than other areas of music. Their approach consists of trying to satisfy larger scale harmonic progressions and trying to satisfy localized voice-leading rules at the same time, looking at what they wrote before. This is a valid approach to writing four-part chorales, even though it might not be the best one, and it does yield good results on occasion, especially for smaller scale class assignment-like four-part music. All of these observations led to the idea of a small-scale, lightweight and customizable model that thinks autoregressively like an instrument major and that tries to balance broader harmonic progressions with localized voice leading patterns looking at what it wrote prior. A model like this could help music majors and musicians in general interact with composition and serve as an almost assistant-like idea source. All of this culminated in a hybrid model that is trainable in under an hour and half in most cases, that makes use of a temporal convolutional network to capture localized patterns and rules, and a simple transformer with self attention to capture global harmonic progressions. The model can produce convincing phrases that could help a musician with their beginning journey in four-part writing or even give ideas to experienced composers.

2 Chorale Generation

2.1 Introduction and Outline

The process of building a lightweight chorale generator was mainly composed of 3 parts. Finding the necessary data and its preprocessing, designing and implementing the model, and improving the model while making sure it stays lightweight and easy to customize. The crucial steps for these parts is determining the representation for the data, picking architectural choices that would match the goal and making sure the model can generate interesting musical material while also keeping it very easily trainable. For this study free GPU time was utilized in Google Colab. In the next sections these steps are explained in more detail.

2.2 Data and Preprocessing

The dataset that was used is an augmented version of an existing dataset "ageron/handson-ml2/datasets/jsb_chorales" on github, which was based on czhuang's JSB-Chorales-dataset. The original dataset contains 382 Bach chorales in their original keys where each chorale is a csv file with 4 integers depicting the MIDI indices for the 4 voices. Rests are denoted by the integer 0. Each chorale was transposed to all other 11 keys to increase the dataset size. The original chorales range from length 100 to length 640, this was modified to make each chorale length 641 with necessary EOS padding at the end. Each chord was also extended with an extra index to signal if it is an EOS token or not which made each chord representation length 5. The final dataset consisted of 4584 chorales with each being represented by a sequence of length 641, where each token/chord in the sequence is a vector of length 5 including the 4 voice and the EOS index.

2.3 Model Design and Implementation

The initial idea for the model was that it would mimic how a music student would write a Bach style chorale and it would treat Bach chorales as a text generation model would treat language. The way a text generation model generates the next token, it would generate the next chord. This resulted in the broad design of an autoregressive hybrid model that makes use of a transformer with self attention to capture the long term dependencies and a temporal convolutional network that captures the local patterns and voice leading rules that exist in Bach's music.

Initially, to represent the chords throughout the model, the idea was using a concatenation of one-hot vectors that would determine which note is being played in each voice plus the EOS index, so a combination of 5 separate categorical variables. To keep the model small, each voice would only have enough classes to fit the minimum and maximum note value that is observed in the augmented dataset. It quickly became clear that this would result in problems. The concatenated one-hot vectors behave like dictionaries where the same key could mean different values since each one-hot vector represents a different range. Pitch identity is broken across voices, the 7th index in the first one-hot vector could mean a C, for example whereas the 7th index in the second one-hot vector could mean an A. Then the projection layer after this which essentially works as an embedding would use a single projection matrix and this matrix would lose information via merging the one-hot encoding information between the voices. This would mean the model would have to learn the explicit relationships like two one-hots depicting the same note or depicting the note an octave apart, implicitly, through co-occurrence patterns, which would introduce unnecessary challenge. An initial fix to this issue could be to standardize each one-hot vector's range to be between the minimum of all notes used in the chorales and maximum of all notes used in the chorale, creating one-hot vectors over a global shared pitch space. This would mean the final embedding would be formed from concatenating 4 equal length one-hot vectors and an EOS index. This does make the model a bit larger but makes it so that the 7th index on all of the sub-vectors correspond to the same note value, saving the model from having to learn an implicit mapping like it did before. This still can be made cleaner, however. The notes C4 and C5 for example, are an octave apart meaning they are very closely related, but their representations are orthogonal to each other in the embedding space from an initial look. Similarly notes a fifth or a third apart are also orthogonal to each other in the embedding space even though they would occur simultaneously on lots of the chords throughout a chorale. With these in mind, the final representation that was used was very close to the initial dataset representation. Each voice was represented as an integer class fed through an embedding. This aligns pitch semantics and is parameter efficient and faster compared to big sparse one-hot vectors. Through the embedding portion, the model can learn a musically meaningful space, easier, that captures relationships such as octave or fifth relationships or interval similarity, just as language models learn word embeddings that capture relationships between words. An important thing to note here is that the model learns an embedding matrix for each voice separately, if it were to use one single shared embedding matrix for all voices there would be information loss due to the remapping before the embedding.

The concatenation followed by a linear layer after the embeddings serve to fuse the embeddings together before feeding them into the hybrid encoder, so that the transformer and the TCNN layers see the chords as a single object and not separate voices.

The hybrid encoder is made up of two parallel configurations, a Transformer branch with causal self attention and RoPE, and a Temporal CNN branch with dilation option. The reasoning behind not serializing these and using them in parallel mostly comes down to wanting to avoid a hierarchy where one module acts like a preprocessor and has to capture the things the second will need. The parallelized structure ensures that both operate on the original learned embedding and learn their respective patterns and rules from it, which can then be fused afterwards.

The first branch is the Temporal Convolutional Neural Network (TCNN) branch. TCNNs have been widely used in sequence modeling tasks such as audio generation and time series forecasting, where local, translation-invariant structures play an important role. In the context of Bach-style chorales, most voice-leading rules, such as avoidance of parallel fifths and octaves, resolution of tendency tones, and characteristic cadential patterns, depend only on short windows of consecutive chords. The TCNN branch provides a strong inductive bias towards such local dependencies with the goal of allowing the model to create a toolbox for voice leading rules and idiomatic movement, just like how a student might learn in a four-part writing introduction class. The TCNN here consists of a stack of causal 1D convolutional layers applied along the temporal axis. Each convolutional layer uses shared filters that slide over short windows of successive chords, optionally with dilation and residual connections, thereby inducing a hierarchical representation of local temporal patterns. Because the convolutions are causal, the representation at time t depends only on inputs at times $\leq t$, which is compatible with the autoregressive next-chord prediction goal.

The second branch is a Transformer-based sequence modeling branch, which is designed to capture long-range and global dependencies across the entire chorale. Transformers have become the dominant architecture for sequence modeling tasks such as language modeling due to their ability to relate distant elements in a sequence via attention mechanisms. In the context of Bach-style chorales, many musically important structures extend beyond local chord-to-chord transitions, including phrase-level harmonic motion, modulation between keys, repetition of harmonic patterns, and the placement and preparation of cadences across longer spans. The Transformer branch is therefore intended to complement the TCNN by modeling these higher-level, non-local dependencies that govern the global coherence of a chorale. It consists of a stack of causal self-attention layers followed by position-wise feedforward networks. Autoregressive generation is enforced through the use of a causal attention mask, which ensures that, at time step t, each chord representation may attend only to chords at positions $\leq t$. This preserves temporal causality and prevents information leakage from future chords during next-chord prediction. Within each attention layer, self-attention computes content-dependent weighted combinations of all past chord embeddings, allowing the model to select which earlier musical events are most relevant for predicting the next chord. This mechanism enables the model to capture relationships that span large temporal distances, such as recalling an earlier harmonic region when completing a phrase or coordinating voice-leading decisions with long-term tonal context. To encode positional information, the Transformer branch employs Rotary Positional Embeddings (RoPE), which inject relative positional structure directly into the query and key representations used by self-attention. Unlike absolute positional embeddings, RoPE encodes positions through position-dependent rotations in embedding space, allowing attention scores to depend on relative distances between time steps rather than their absolute indices. This property is particularly well-suited for music, where the functional relationship between chords often depends more on their position relative to each other than on their absolute position within the piece. A predominant function chord (IV) being right before a dominant function chord (V) is more important positional information as opposed to it being on index 5 in the sequence, for example. By combining causal self-attention with RoPE, the Transformer branch aims to model long-range harmonic structure in a position-aware manner, providing a global structural perspective that complements the TCNN's focus on local voice-leading patterns.

After the two branches produce their output, we have the fusion block where the outputs, which live in different representation spaces at this point, of the Transformer and TCNN branches are concatenated and then passed through a linear layer with GELU to actually learn how much global versus how much local context matters in each time step. The learned matrix here is $Wfuse \in R384\times768$. Each output dimension i computes:

$$H_{\text{fused}}^{(i)} = \sum_{j=1}^{384} w_j^{(i)} H_{\text{trf}}^{(j)} + \sum_{k=1}^{384} w_{384+k}^{(i)} H_{\text{cnn}}^{(k)}$$

This allows every fused feature to rely on Transformer information mostly, TCNN information mostly, or combine information from both of them in any ratio. GELU here also matters, as it selectively attenuates weak or ambiguous fused features while preserving strongly activated ones, allowing the model to dynamically emphasize either long-range harmonic structure or short-range voice-leading patterns depending on context, without hard switches. In continuing a phrase it might make the dominant signal be the TCNN one while at a cadence point it might make the Transformer signal to be the dominant one.

The fusion layer does learn a mixture of global and local information, and can weight and gate individual features. However, it cannot model nonlinear interactions between features, as each output dimension depends only on a scalar linear combination of inputs. To enable joint reasoning over fused features, a post-fusion MLP is applied, introducing intermediate nonlinear transformations that allow higher-order feature interactions within each time step. We can describe it as such: fusion decides what information is present; the post-fusion MLP decides what it means together. The goal here is to capture statements such as "If feature A is strong **and** feature B is strong, then feature C should activate".

The final step is to actually predict the next chord through output heads. A problem emerges from the idea of treating every chord as a token and predicting the next token/chord: not counting the EOS index, there are 33 * 33 * 35 * 42 = 1600830 different possible chords. Treating each chord as a token, this would make the vocabulary size 1600830 if we ignore the EOS index, and we would have to assign probabilities to each chord in our vocabulary. This is, of course, computationally infeasible, especially for our goal. It also does not make musical sense since most chords in this vocabulary would be composed of a combination of voices that would almost never appear together. So the model employs separate output heads for each voice and the EOS index, learning separate logit matrices for all of them. This reduces the computation requirements drastically and also adds the option to add voice specific regularization if needed. It also has the added benefit of separating EOS prediction, which eliminates EOS competing with pitch probabilities, preventing the model from potentially using EOS as an "escape" from difficult predictions, and allows for independent EOS loss weighting.

Figure 1 below is a summary of the entire model in a diagram format.

```
Data: X ∈ ℝ[B, T, 5]              ┌──────────────────────────────┐
                                  │ X[:, 0:640] = inputs         │
B = batch size                    │ X[:, 1:641] = targets        │
T = 641                           └──────────────────────────────┘

                        ┌────────────────────────────────────────────────┐
                        │                  INPUT TOKEN                   │
                        │             x ∈ ℤ^{B×T×5} (raw)                │
                        │     [S_midi, A_midi, T_midi, B_midi, EOS]      │
                        └────────────────────────────────────────────────┘
                                          │
                                          ▼
              ┌──────────────────────────────────────────────────────────────────────┐
              │                 REST-AWARE TOKENIZATION + EMBEDDING                  │
              │ MIDI → class indices per voice:                                      │
              │   REST=0, NOTE=(midi - MIN + 1)                                      │
              │ Embeddings per time step:                                            │
              │   e_S,e_A,e_T,e_B ∈ ℝ^{64},  e_EOS ∈ ℝ^{16}                          │
              │ Concat: [e_S;e_A;e_T;e_B;e_EOS] ∈ ℝ^{272}                            │
              │ Linear projection: 272 → 384                                         │
              │ Output: H ∈ ℝ^{B×T×384}                                              │
              └──────────────────────────────────────────────────────────────────────┘
                            │                                    │
                            │                                    │
                            ▼                                    ▼
┌────────────────────────────────────────────────┐   ┌────────────────────────────────────────────────┐
│          TRANSFORMER BRANCH (GLOBAL)           │   │          TEMPORAL CNN BRANCH (LOCAL)           │
│         8 × TransformerBlock (Pre-LN)          │   │          5 × CausalConvBlock (TCN)             │
│                                                │   │                                                │
│ Each block:                                    │   │ Each block:                                    │
│  LN → RoPE Causal Self-Attn → Residual         │   │  Conv1d (causal) → GELU                        │
│  LN → MLP(384→1536→384) → Dropout → Residual   │   │  Conv1d (causal) → GELU → Dropout → Residual   │
│                                                │   │ Dilations (if enabled): 1,2,4,8,16             │
│ Causal mask: prevents attending to future      │   │ Kernel size: 7                                 │
│ RoPE: rotates Q,K with relative position info  │   │                                                │
│                                                │   │ Final LayerNorm                                │
│ Final LayerNorm                                │   │ Output: H_cnn ∈ ℝ^{B×T×384}                    │
│ Output: H_trf ∈ ℝ^{B×T×384}                    │   └────────────────────────────────────────────────┘
└────────────────────────────────────────────────┘
                            │                                    │
                            └─────────────────┬──────────────────┘
                                              │
                                              ▼
                    ┌────────────────────────────────────────────────┐
                    │                    FUSION                      │
                    │ Concat: [H_trf ; H_cnn] ∈ ℝ^{B×T×768}          │
                    │ Linear: 768 → 384                              │
                    │ GELU                                           │
                    │ Output: H_fused ∈ ℝ^{B×T×384}                  │
                    └────────────────────────────────────────────────┘
                                              │
                                              ▼
                    ┌────────────────────────────────────────────────┐
                    │        POST-FUSION REFINEMENT (SHARED)         │
                    │ MLP: 384 → 768 → 384                           │
                    │ Residual: H_fused + MLP(H_fused)               │
                    │ LayerNorm                                      │
                    │ Output: H_refined ∈ ℝ^{B×T×384}                │
                    └────────────────────────────────────────────────┘
                                              │
                                              ▼
┌──────────────────────────────────────────────────────────────────────────────┐
│                          OUTPUT HEADS (PER t)                                │
│                                                                              │
│  S head:   Linear 384 → P_S=34   → logits_S   ∈ ℝ^{B×T×34}                   │
│  A head:   Linear 384 → P_A=34   → logits_A   ∈ ℝ^{B×T×34}                   │
│  T head:   Linear 384 → P_T=36   → logits_T   ∈ ℝ^{B×T×36}                   │
│  B head:   Linear 384 → P_B=43   → logits_B   ∈ ℝ^{B×T×43}                   │
│  EOS head: Linear 384 → 1        → logits_EOS ∈ ℝ^{B×T×1}                    │
└──────────────────────────────────────────────────────────────────────────────┘
```

Figure 1: Model Diagram Including the Removable Post-Fusion MLP layer

2.4 Results and Discussion

Before analyzing some of the results, it should be noted that the chorale generator generates chords for each 16th note time step. This means if we take the literal generation without any processing we get a score with a new chord being played at each 16th note time step, inevitably sounding very out of style. To prevent this, the direct output of the model gets post-processed using a Python script utilizing the **music21** library, and a little bit of manual adjustment after that at the end. This process results in a MuseScore file that is listenable, that better resembles a proper chorale score.

The results that were generated were generated via giving the chorale just an initial 16th note portion of the starting chord. There were 3 different configurations of the model that were tested.

The first configuration uses d_model = 256 for the model dimension (main hidden size), 4 transformer layers, 3 convolution layers and kernel size of 5. It is trained for 20 epochs with a batch size of 16, and uses greedy decoding during prediction. It does not employ post-fusion MLP for refinement. The produced chorales show promise but they degenerate into chaotic voice leading patterns devoid of harmonic sense after a while sometime or tend to meander around in the home key without direction. In Figure 2 below we can observe some properties we wanted to capture, and some properties that are lacking.

Figure 2: Example output #1 from Configuration 1

Right from the start we can observe an idiomatic Tonic (I) - Subdominant (IV) - Dominant (vii6) - Tonic (I) progression with passing tones in the first measure, granted the second beat is missing an important note in the chord, the 3rd scale degree. Similarly on measure 3 there is a IV - V64 - V - V7 progression. These suggest the model did learn some idiomatic cadential movement as intended. A problem that can be observed, however, is that in lots of cases one voice of the chord picks a note that is not a part of the chord and that can't be interpreted as a proper usage of a non-chord tone such as the B natural on measure 2 beat 1, or measure 5 beat 4. This could be the result of the fact that when generating the next chord, each note is being generated independently from one another, just looking at the previous chords. This will be further explained later. Another important observation here is that the model has some sense of more global phrase structuring, it performs a perfect authentic cadence (PAC) at the end of measure 3 and modulates to the fifth scale degree and tries to make a PAC at the end of measure 8. These suggest the model was able to plan a convincing modulation and execute it at a proper measure number. However, the first PAC is on measure 3 and is not very appropriate for its position. One reason for this could be that even though the dataset is mostly in 4/4 time signature there are some chorales in other time signatures such as 3/4. This could confuse the model and make it predict cadences in inappropriate locations especially when the generated chorales are all being treated as 4/4. Below is another generation example.

Figure 3: Example output #2 from Configuration 1

Here the model demonstrates significant understanding of modulation, in this case, to the relative minor key and interesting polyphony and voice leading. It correctly captures secondary dominant relationship on measure 4 and knows to resolve the F# leading tone to a G. It almost arrives at a cadence in measures 2 and 4 but misses it, which is a pattern that emerges in the generated music throughout examples. This might point toward the need for extra care in teaching cadence points to the model through other means in future work. Another observation here is that once the generation goes off track and introduces chords that don't make harmonic sense, degeneration starts and music becomes worse and worse. This might be a result of the autoregressive nature of the model and the model overemphasizing local voice leading rules over broader harmonic context during generation, especially when the harmonic context starts to get lost. As we can see, once the model can find its way to a c minor chord on measure 6, it starts to make more musical sense getting closer to the original key of Eb major. This particular chorale actually finds its way back to

Eb major with a Authentic Cadence in the next measure. In Figure 4 below we see another problem.

Figure 4: Example output #3 from Configuration 1

The model is able to generate a very successful 2 bar phrase ending with an IAC followed by an answering 2 bar phrase ending with a HC, with idiomatic soprano and bass lines, and usage of 3rd scale degree minor chord to start a phrase. However, the harmonic rhythm is off. In measure 3, the arrived chord from the previous measure is still being held and in measure 4 beat 3 the harmony changes in a very unexpected and off-putting beat. Once again, these issues might be because of the existence of different time signatures in the dataset during training, or they may also be a result of the model not learning what a new measure is during training.

The second configuration uses d_model = 384 for the model dimension (main hidden size), 8 transformer layers, 5 convolution layers and kernel size of 7. It is trained for 6 epochs with a batch size of 8, and uses temperature and sampling during prediction. Once again it does not employ post-fusion MLP for refinement. Compared to the first configuration it seems that this one can continue a coherent structure a bit longer. However, there also seems to be an emergence of heavily repeating soprano voice patterns. Figure 5 below demonstrates both good coherence and how it falls into the repeating soprano pitfall.

Figure 5: Example output #1 from Configuration 2

Here we once again observe the model is able to capture an interesting phrase structure, if the rhythmic problem had not occurred on measure 3 beat 1, and measure 3 beat 2 was played on beat 1, we would get an avoided cadence on beat 4 of measure 4 and measures 5 and 6 would be an extension to the phrase. Another notable thing here is the similarity between the soprano melody in the first 2 measures and the soprano melody from beat 2 of measure 5 to beat 1 of measure 7, suggesting the model might have some capacity to learn short motifs. On measure 9 the excessively repeating soprano melody emerges and continues on for a while. One reason for why this configuration might have resulted in this problem is that both the Transformer and Temporal CNN branches are well suited to learning strong local continuity priors, including the tendency to repeat recently observed pitches. So the depth increase of both branches may amplify this effect by strengthening confidence in short-term predictive heuristics such as this one. When generating the next chord autoregressively, such high-confidence local priors might dominate the output distribution, causing repetition to become a stable attractor that reinforces itself across successive time steps. The deeper branches also seem to capture phrase structure better, however. Figure 6 below demonstrates the model almost capturing a musical narrative with 4 successful phrases.

Figure 6: Example output #2 from Configuration 2

The relatively good planning and execution of the two IACs followed by a secondary dominant leading to a HC in the 6th measure makes it seem like the deeper branches did help the model gain better understanding of longer musical coherence. However, as encountered in above examples, with a wrong chord pick at the beginning of 7th measure, the model gets into a repeating HC pattern and loses the narrative. We can also observe parallel fifths on measure 2 which could again be a result of two voices not having information on each other when the next chord is being generated or which might suggest the model still couldn't learn through other means that this type of movement is very unlikely.

The third configuration uses the same number of layers as the second one for both branches and uses the same model dimension as the second one. During training early stopping is employed, with max epoch number being 20 and batch size being 6. This time the post-fusion MLP layer is activated. Unfortunately the empirical results revealed that the post-fusion MLP layer did not improve the model performance as expected above in the model design section. In fact, the generation quality went down drastically. A typical output is shown below through Figure 7.

Figure 7: Example output from Configuration 3

This is one of the outputs from configuration 3 with more movement in it but even here we can observe a tendency for the model to not change notes from chord to chord. This is more extreme in other examples, even sacrificing generating harmonically sensible chords in favor of repeating notes. This could be a result of the post-fusion MLP introducing a contractive mapping in hidden space that pushes diverse contexts toward a small set of stable representations. Once generation enters such a region, the resulting logits become nearly the same across time steps, causing the autoregressive process to converge to a fixed point characterized by exact chord repetition, even using sampling.

The issue of converging to a fixed chord and going countless measures with that chord until an EOS token is reached was actually observed in all 3 configurations, just amplified by the post-fusion MLP.

Some further quantitative analysis also has been performed on the generated chorales from Configuration 1 and Configuration 2 (Config 3 has been omitted due to very poor performance). The average training and validation loss for each epoch for Configuration 1, from 5 different training runs, have been computed and reported in Table 1 below.

Table 1: Training and validation loss for each epoch averaged over 5 training runs for Configuration 1.

Epoch	Train	Val	Epoch	Train	Val
1	1.936	1.050	11	0.506	0.736
2	0.947	0.863	12	0.475	0.747
3	0.828	0.800	13	0.447	0.767
4	0.766	0.762	14	0.419	0.783
5	0.718	0.741	15	0.392	0.802
6	0.678	0.725	16	0.367	0.830
7	0.640	0.719	17	0.344	0.852
8	0.605	0.716	18	0.322	0.875
9	0.571	0.718	19	0.302	0.905
10	0.538	0.725	20	0.284	0.933

A classic overfitting trend can be observed from the data, with the training loss decreasing further and further while the validation loss reaching its lowest point around epoch 11 and then starting to increase. Similarly, the average training and validation loss for each for Configuration 2, from 5 different training runs, have been computed and reported in Table 2 below.

Table 2: Training and validation loss for each epoch averaged over 5 training runs for Configuration 2

Epoch	Train	Val
1	1.3484	0.8549
2	0.7933	0.7527
3	0.6846	0.7089
4	0.5893	0.6980
5	0.4924	0.7173
6	0.3982	0.7557

Training for Configuration 2 was manually set to end at 6 epochs with the reasoning that the considerably larger model size would learn much quicker which seems to be correct as it reaches minimum validation loss very quickly.

To further test overfitting and potential memorization for both configurations, a basic memorization analysis was done. For configuration 1, 11 “good” generated chorale examples were picked and their first 8 measures were compared against each measure of the entire augmented training dataset. Figure 8 below shows a heatmap for Configuration 1, that shows what is the highest percentage match with each measure of the example chorales and any measure in the augmented training data. To make it clearer, measure 1 corresponds to the first 16 tokens/chords in the chorale data representation, measure 2 corresponds to tokens between 17 and 32 inclusive and so on. To calculate the percentage match between two measures, the note played by each voice in each 16th note time index in the data representation in measure 1 was compared to the same note in the same voice and 16th note time index in the data representation in measure 2. A 100% match suggests the two measures play the same note at every voice on every 16th note time index.

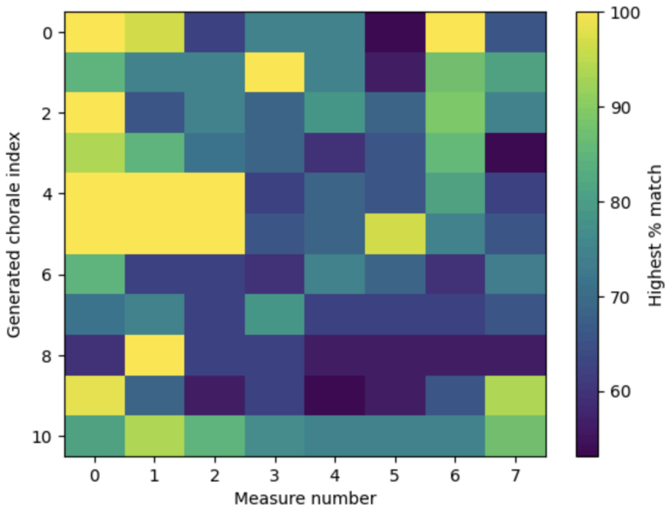

Figure 8: Heatmap for percentage match between first 8 measures of “good” chorale outputs from Configuration 1 and any measure from the augmented training data

Here we can indeed see that there are measures that have been completely memorized by the model. It is important to note that a bunch of these chorales had the same seed chord to start generation, such as chorale 4 and chorale 5, where we can see both have their first 3 measures at a 100% match. Similar analysis was done on Configuration 2 as well. Figure 9 below shows the results.

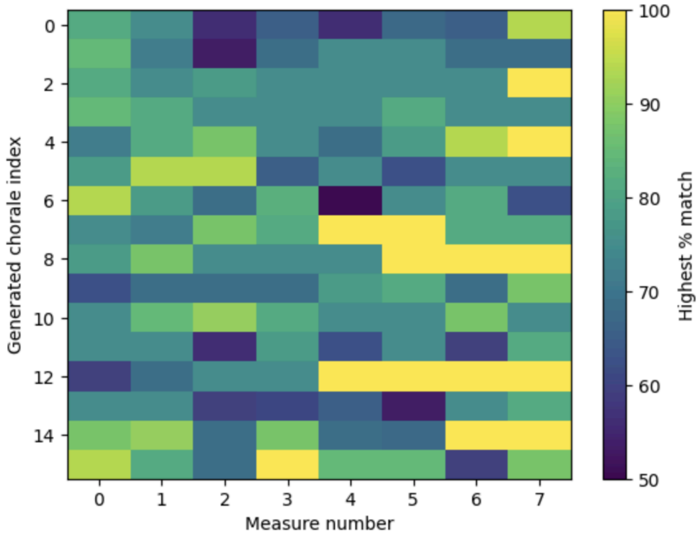

Figure 9: Heatmap for percentage match between first 8 measures of "good" chorale outputs from Configuration 2 and any measure from the augmented training data

Again, we can see there were measures that were completely memorized, but surprisingly these measures do not occur at the beginning. The deeper branches of configuration 2 promoting stable attractors which causes repetition that would not be seen in Bach's chorales could also be lowering the match percentages here, combined with the fact that the training process for this configuration was manually cut short to prevent overfitting.

Finally, Shannon Entropy Analysis was performed on a set of 20 chorale generations in the key of F major for both configurations. The generations were seeded with 4 different starting chords, all a variation of the F major chord, with each seed being used to generate 5 different chorales. Shannon Entropy was calculated for each voice through the first 8 measures, for each of the 20 chorales, and each voice's average was taken. Similarly, the Shannon Entropy for each voice was calculated for the first 8 measures, throughout all of the existing chorales in the key of F major in the augmented dataset and then averaged. To calculate Shannon Entropy, the following formulas was used:

$$H(X) = -\sum_{k} p(k) \log_2 p(k)$$

The mean entropies for all the voices for both configurations' generations and the F major chorales in the training data are reported below in Table 3.

Table 3: Average Shannon Entropy for Voices Across Generated Chorales and Augmented Training Data

	Soprano	Alto	Tenor	Bass
Config 1	2.11779	1.735776	1.903012	2.51663
Config 2	1.5808	0.86719	0.936197	1.79019
Augmented Training Data	2.5471	2.43523	2.58385	3.07095

The results back up the previous claims on how deeper branches may have caused repetition to be a stable attractor as Configuration 2 generations exhibit the least entropy out of all three. Another interesting result here is the bass voice having higher entropy than the soprano for all three. This is unexpected but might be due to the fact that we are calculating entropy on our chord representations/tokens in the first 8 measures, meaning a held soprano note in the melody is considered 4 different soprano notes playing the same voice. We can also see that both configurations did capture the bass entropy being the largest among the voices just like in the training data. In principle, it would be informative to compute entropy within each chorale and use a t-test to compare the machine-created chorales to those of Bach. However, since we are only looking at the first 8 measures, where compositional quality is highest, we do not think the within-composition entropy is very meaningful or comparable. Therefore, only the entropy computed over the entire collection of outputs was compared.

In the next section we explore some ideas that could potentially solve some issues that seem to plague the model such as the repetition problem or the memorization problem.

3 Future Work

There are many future potential routes that can be taken on top of this project to improve the generated chorales while also keeping the lightweight and easy to play with and customizable nature of the model. One immediate change would be to introduce some pre-determined domain rules into the system to eliminate some extremely uncharacteristic passages such as parallel fifths and octaves in Bach's four part writing.

Another very useful improvement could be the introduction of measure numbers and beat numbers for the chords, though this would require a lot of preprocessing of the current data. This might be a crucial step as it can help the model learn about harmonic rhythm and phrase structure which are key things for generating convincing music and are a problem point in the current model. It could help the model learn things like cadences mostly occur at the end of 2 or 4 measure phrases, or harmony almost always changes from beat 4 of measure *t* to beat 1 of measure *t+1*. It could also be a fix to problems like getting stuck on the same chord, which occurred quite frequently towards the ends of chorales, and meandering without arriving at a cadence point. Overall this improvement would have a lot of potential.

The current generation of the next chord also only relies on all of the chords before but not the current chord being generated, which actually most of the time is not how students or composers would approach this problem. A more common approach would have been to generate the bass or soprano voice first, and then to fill in the alto and tenor voices to complete the chord. This change would not require too much extra work and could eliminate incomplete chords without the third scale degree of the chord, for example, and add more interesting and stylistic bass and melody lines.

The model, due to its small scale and not too complicated nature and the simplicity of its data, is very open to improvement and experimentation. The potential routes mentioned above are only a small subset of the possible future work that could be done. Changes like adding an LSTM layer towards the end or swapping the existing architectures that lead to the fusion with different architectures can also be explored in the future, maybe scaling the project up.

4 Conclusion

This thesis examined symbolic Bach-style chorale generation through the design of a lightweight, autoregressive hybrid model that balances local voice-leading constraints with broader harmonic structure. Rather than pursuing state-of-the-art performance using large-scale architectures, the goal was to explore whether a compact and interpretable system could learn musically meaningful patterns while remaining easy to train and customize. Bach chorales were selected as the target domain due to their pedagogical importance, constrained musical vocabulary, and availability of the data, making them well suited for experimentation in generative music modeling.

The proposed model combines a Transformer branch for capturing long-range harmonic dependencies with a Temporal Convolutional Neural Network branch for modeling localized voice-leading patterns. These components operate in parallel on a shared embedding space and are fused in a learned, context-dependent manner, allowing the model to dynamically balance global and local musical information. Additional design choices, such as voice-specific embeddings, separate output heads per voice, and explicit EOS prediction, were motivated by both musical structure and computational efficiency. The resulting system is compact enough to be trained on modest hardware while still producing non-trivial generative output.

Experimental results demonstrate that the model is capable of learning several stylistically relevant musical features, including idiomatic cadential motion, convincing local voice-leading, correct modulation patterns and occasional phrase-level coherence. At the same time, the experiments reveal clear and informative failure modes. Autoregressive degeneration, excessive repetition, unstable harmonic rhythm, and sensitivity to architectural depth emerged across configurations. In particular, deeper models seemed to amplify short-term continuity heuristics, causing repetition to become a stable attractor during generation. The introduction of a post-fusion refinement MLP further degraded generation quality, suggesting that nonlinear refinement after fusion can introduce contractive dynamics that lead to output collapse.

Several limitations remain, including the lack of explicit metrical awareness, independent voice prediction, and reliance on post-processing to produce musically listenable scores. Nevertheless, these limitations point to clear directions for future improvement. Overall, this work demonstrates that musically meaningful symbolic generation can be achieved with compact, interpretable models, and that small-scale hybrid architectures offer a valuable framework for studying generative music. In conclusion, the final model reaches our goal of creating a lightweight model that can serve as an assistant or tool for experimentation.

References

[1] K. Ebcioglu, “An expert system for harmonizing chorales in the style of J. S. Bach,” *J. Logic and Algebraic Programming*, vol. 8, no. 1, pp. 145–185, 1990.

[2] M. Allan and C. K. I. Williams, “Harmonising Chorales by Probabilistic Inference,” in *Advances in Neural Information Processing Systems 17*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds., Cambridge, MA, USA: MIT Press, 2005, pp. 25–32.

[3] G. Hadjeres, F. Pachet, and F. Nielsen, “DeepBach: a Steerable Model for Bach Chorales Generation,” in *Proc. 34th Int. Conf. on Machine Learning*, PMLR vol. 70, Sydney, Australia, 2017, pp. 1362–1371.

[4] C. Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, “Counterpoint by Convolution,” in *Proc. 18th Int. Soc. Music Information Retrieval Conf. (ISMIR)*, Suzhou, China, 2017, pp. 211–218.

[5] C. Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, “Music Transformer: Generating Music with Long-Term Structure,” in *Proc. Int. Conf. on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[6] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, “Controllable deep melody generation via hierarchical music structure representation,” in *Proc. 22nd Int. Soc. Music Information Retrieval Conf. (ISMIR)*, Online, 2021, pp. 143–150.

[7] Y. Wu, *Motif-based Music Representation Learning and Structure Analysis*, M.S. thesis, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, PA, 2023.