Music Generation Using Autoencoders and Transformer Mixture Distribution Models

Kevin Chen
Division of Engineering Science
University of Toronto
Toronto, Canada
kevinkaiwen.chen@mail.utoronto.ca

Yina Gao
Division of Engineering Science
University of Toronto
Toronto, Canada
yina.gao@mail.utoronto.ca

David Guo
Division of Engineering Science
University of Toronto
Toronto, Canada
davidmy.guo@mail.utoronto.ca

***Abstract*—Despite recent progress in generative AI, challenges persist in creating musically coherent compositions. Building upon established techniques such as variational autoencoders (VAEs) and transformer mixture density models (MDN), we introduce a reduced-scale approach to unconditional music generation.**

The proposed model is a modification of the MusicVAE-TransformerMDN model utilized by Mittal et al. Similarly, our model utilizes transformers that operate on latent embedding to capture melodic patterns and dependencies. Our implementation differs on several fronts, namely utilizing sliding window to pair context and target embedding during data processing, along with general network architecture changes. In addition, we switch between two datasets mid-way through training for the purpose of fine tuning. The model is first trained on a large dataset of with varying genre and style, then switches to a more specific dataset with classical piano music. Notably, the results show that such a method is effective in retaining model efficacy despite a 90% reduction in dataset size and a shallower network, although it comes at the cost of a smaller variety of styles it can emulate.

Quantitative evaluation metrics such as Fréchet Distance and Mean Value Discrepancy demonstrate the retained efficacy in comparison to a prior baseline and the ground truth. Furthermore, subjective analysis by humans of varying familiarity with music theory indicates the model is effective at learning tonality. However, all models, including the baseline and the model presented by Mittal et al produce poor evidence of rhythm and cadence. The model produced mixed results for long-term rhythm and song structure. As such, limitations of the task and possible solutions to this issue are also discussed.

***Index Terms*—Automatic Music Generation, Transformer Mixture Density Model, Variational Autoencoder, MIDI**

I. Introduction

A. Overview

Generative AI has recently been applied, with great success, to art and text. Generative models and architectures, powered by diffusion and transformers, are very popular and are now being deployed and available to the general public. They are convenient because they offer significant cost and time savings. Both being more creative domains, music generation is a natural next step. However, music has its unique challenges.

Music is an ultra-high dimensional input and target. There are too many different aspects of music; pitch and rhythm objectively, but also how these musical features combine to create new ones such as playing style, harmony, and musical meter. Music composition is an artistic discipline that requires both creativity and sufficient understanding of theory to make it pleasant sounding. We hope to create a model that can emulate this understanding, and implicitly learn the guidelines that make music enjoyable to listen to.

Several different techniques have been experimented with, with various degrees of success. Many of these strategies have taken inspiration from image and text generation, such as using diffusion, transformers, and embeddings.

We demonstrate that the challenges of music generation can be overcome by using a variational autoencoder and principal component analysis to reduce the dimensionality of music and also computationally derive and extract the important qualities of a given sample of music. This autoencoder is meant to capture the essential qualities of a musical sample without extensive hand-engineering of features.

We combine this with a well-studied transformer mixture density model [1] [2]. By using self-attention, the model is able to effectively incorporate the time-series nature of music. Originally devised for language problems, we show that transformers can work well for music generation. [1].

B. Literature Review and Previous Work

Much work relating to music generation has been done, and many use common elements. We focus on models that use Diffusion, transformers, and also models that embed music for preprocessing.

Various strategies for embedding music have been formulated, but a useful one is MusicVAE [3]. This is an encoder-decoder architecture with LSTMs.

Mittal et al. formulate a neural network architecture using diffusion. [2] Taking MIDI files, the model embeds them using MusicVAE and then uses a diffusion model for unconditional generation. This is essentially generating music from scratch.

Majidi et al. [4] pursue another approach. They rely on genetic algorithms for generation, and Bi-LSTMs combined with music grammar as the objective function. Genetic Algorithm One (GA1) generates music pieces and evaluates them based on similarity to existing music and rhythm or harmony violations. The Bi-LSTMs are trained using regular and expert listeners to score input music. Genetic Algorithm

[1]See our GitHub Repo at https://github.com/davidguo123456/ECE324

Two (GA2) is a modification of GA1 which adds the human-trained Bi-LSTMs to its objective function. Generally, the largest contribution of this paper is their use of neural networks in the objective function to capture listener satisfaction as well as musical grammar.

Lam et al. combine music generation with text input to give more versatility anc customizability to the generation process, using Language Models (LM) to guide the process [5]. A problem with music generation is the difficulty behind creating coherence across the generated sequence. The VAE-GAN uses Dual-Path Diffusion (DPD) to solve this. The first path is a "long-context generation" path that predicts a multi-chunk target on larger blocks of generated music. The other path is a fine path to focus on the smaller details of the music.

JEN-1, proposed by Li et al. [6] is another model using Omnidirectional Diffusion Models. It utilizes both autoregressive and non-autoregressive diffusion modes and latent embeddings. The diffusion network itself is modified from Efficient U-Net.

Another relevant aspect of training these models is being able to recognize melodies. Zhao et al. [7] propose a modified definition for main melody, defining it as "...a set of similar but non-identical melodies that can be utilized to identify the music by entities..." along with a novel model to extract multiple main melodies from a song. Input data is represented with Compound Word, (with five tokens to describe notes), and a sliding window to identify chord likelihood.

In their paper, Tang et al. review notable deep learning focused approaches to music generation in recent years, and highlight their proposed model, RM-Transformer. [8] RM-Transformer can be described as a two part model that uses an encoder, decoder, and an attention metric.

The authors Peebles and Xie [9] propose a new class of diffusion models that utilize transformers as their backbone, coined Diffusion Transformers (DiTs). For DiTs, Peebles and Xie made promising observations when training architectures using a forward pass inspired by Vision Transformer architecture and pre-trained autoencoders from Stable Diffusion. The contribution is showcasing their scalability and the possibility of using transformers as substitute backbone for existing diffusion models.

C. Problem Statement

Composing music often requires extensive knowledge in music theory and practice. These considerations make composing a melody a difficult and time consuming task for musicians and the general population alike. Given the success of VAEs and transformers in generation tasks, our problem statement is to use these architectures to unconditionally generate independent melody lines that can emulate real music.

D. Dataset

We used two datasets. The first is a subset of the Lakh MIDI dataset [10]. This is a standard dataset used for training audio samples. Second, we used a classical piano music dataset [11].

MIDI is a file format that encapsulates pitch, duration, and instrumentation of music, making it highly versatile.

II. Model and Training Specifications

A. Data Processsing and Model Architecture

The MIDI files are first fed into MusicVAE to generate lower dimensional embeddings of the music. This is necessary to reduce training time as MIDI files are extremely high-dimensional, encoding information about pitch, duration of notes, different instruments, and tempos (speeds), all changing throughout time [3].

The VAE has an encoder-decoder architecture where the encoder is a bi-direction LSTM and the decoder is an autoregressive LSTM. See the architecture in figure 1. [2]

For every input song, MusicVAE encodes each 2 bar segment into a 512-dimension embedding. A sliding window of size 16 is then utilized to pair contexts with targets. Compared to Mittal et al, this is a reduction in context size (down from 32 to 16), but enables more of the input data to be used in training while also reducing the impact of padding [2].

Next, PCA is performed on the output of MusicVAE to transform the 16x512 data to 16x42 dimensional embeddings, which are fed into the next half of the model.

Then, a 128-dimensional sinusoidal positional encoding is added to the input samples according to (1) from Mittal. [2].

$$\omega = \left[\frac{h}{10^{-4} \times 0.63j}, \dots, \frac{h}{10^{-4} \times 63j}\right] \quad e^j = [\sin(\omega), \cos(\omega)] \tag{1}$$

The generative part of the model's architecture is an autoregressive transformer. This transformer encoder has 5 layers, with 8 self-attention heads. Each layer has 2 shortcuts. See figure 2.

The output from the transformer is fed into 2 fully-connected layers, each with 2048 neurons. The output of this model is a mixture of 100 Gaussians, to capture a wide range of distributions.

B. Training

Training was conducted for 16 hours in WSL2 on a system with a Ryzen 7 5800X3D CPU and RTX 3070 Ti GPU, both less powerful than that used by Mittal et al. Using WSL2 also added additional compute overhead.

20 Epochs of training were done on the Lakh MIDI dataset. 25 more were done on the classical dataset, with a reduced learning rate for the last 5 epochs for better fine-tuning.

This is an application of finetuning. We think that starting training on the larger dataset allows the model to first learning general musical relationships. Then, applying it on the classical dataset fine-tunes the model to generate for a specific genre and produce more consistent results.

We found that the classical dataset, being smaller, also was faster to train, allowing us to use our limited compute better.

III. Results

Unlike simpler tasks like classification, there are no standard evaluation metrics for music generation. We will quantitatively evaluate the generated samples with framewise average musical statistics, Fréchet Distance, and Mean Value Discrepancy.

Fig. 1: MusicVAE architecture based on Mittal paper [2]

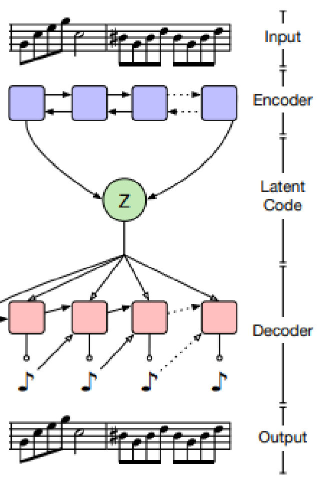
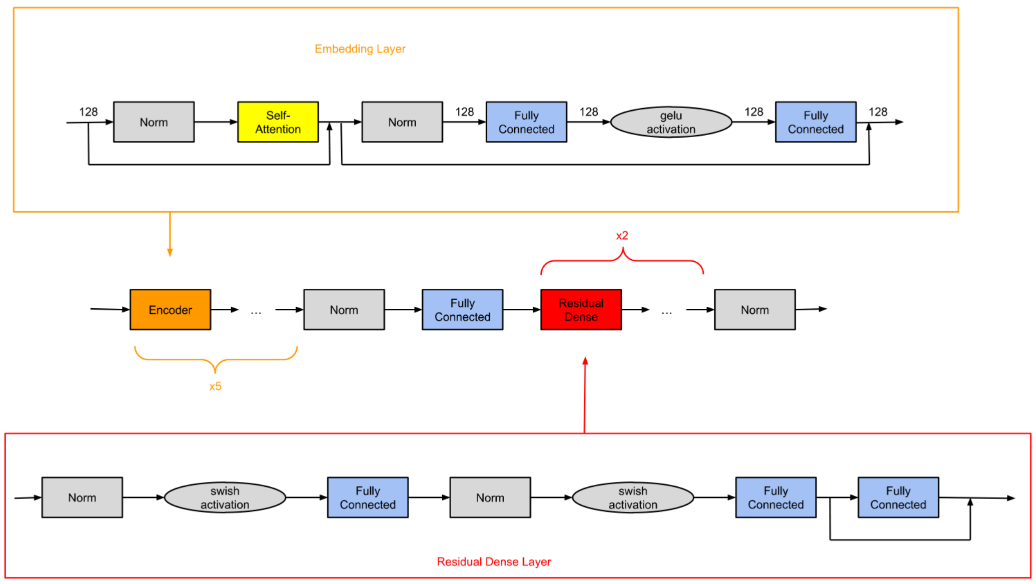


Fig. 2: Model Architecture for Transformer Mixture Density Model [2]



Our baseline model is the generation of a random embedding from a standard Gaussian distribution, which then follows the same process to get decoded into music using MusicVAE.

Since musicality is subjective and hard to quantify, we also present subjective results and record noteworthy observations from the human evaluation of qualities like evidence of music theory understanding, musicality, and pleasantness of the generated samples.

Ten music samples were unconditionally generated using our trained autoregressive Transformer model. These were compared with random samples that were generated by drawing a random vector from a standard normal of the size of the embedding and passing that to MusicVAE.

A. *Human Evaluation*

The model-generated samples were given to humans for their evaluation.

- **Track 2:** This one seems random. The track has no cohesion.
- **Track 4:**
  - 0:09 - 0:47
  - Mostly in minor key, alternating between A minor and C major
- **Track 5:**
  - 0:33 - 1:00
  - Rarely out of key, has cadence
- **Track 6:**

  - Octaves at the beginning, showing knowledge of structure
  - At 0:16, in-key triad
- **Track 8:**
  - D# diminished triad, using D#, F#, A#
  - Doesn't sustain though
- **Track 9:**
  - Overall weird mixture, but has some of the best short sections because of good use of tonality
- **Track 10:**
  - C# melodic minor
  - 0:02 - 0:40 is mostly melodic, however poor rhythm
  - Elements of a key theme or ostinato, no answer to the melody

Overall comments for all tracks: while the model is good at keeping the tonality, the samples consistently lack evidence of rhythm, and the specific combination of notes seems like it is randomly walking the scale instead of having a meaningful structure.

B. Average Note Statistics for Sliding Windows

Average framewise note statistics are calculated by considering notes within a frame 4 measures long. Then, the frame is moved measure by measure across the sample. The series of statistics is recorded and plotted for the generated, random, and testing data and averaged across the 10 samples. The purpose of using framewise statistics is to evaluate the model's ability to produce music that demonstrates long term patterns, which relates to the strength of melodic and rhythmic similarity as proposed by [2]. For the following metrics, smaller deviations in the statistics suggests melodic and/or rhythmic consistency and stability in the samples, which is desired. See the appendix for all figures in this section.

In figure 5, we compared the pitch range (difference between the highest lowest notes) of a sliding window of notes for a generated sample, real sample, and random sample. Observing the smoothness of the graphs shows the framewise pitch range is the most consistent in the testing data, second most consistent in the generated data, and least consistent in the random samples. A similar conclusion can be drawn for mean pitch in figure 3 and pitch variance in figure 6. These results show that the model outperforms the random baseline in learning the melodic patterns of real music.

For the rhythmic metrics, mean note duration and note duration variance, shown in figures 4 and 7 respectively, there is no obvious order of which samples between real music, model output, and random data are the most rhythmically consistent. A possible explanation of these results is since the testing data doesn't show strong rhythmic stability compared to even the random baseline, the embedding method using MusicVAE may perform poorly at extracting rhythmic information.

C. Fréchet Distance and Mean Value Discrepancy

The Fréchet Distance (FD) is a measure of the maximum distance between two vector trajectories at any point in time. The Mean Value Discrepancy (MVD) is the difference between the mean values of distributions generated between two models. Both are common metrics for generative models. We compared the final trained model (trained with 45 epochs) to the random baseline. For both metrics, a lower distance can be interpreted as more similarity between the sample and a training distribution.

TABLE I: Fréchet Distance and Mean Value Discrepancy for Generated Music and Random Music, comparing with a Training Example

	Generated	Random
Fréchet	1.77	11.55
MVD	0.15	0.20

As seen from table I, the lower distances for the Generated sample indicate the generated sample is more similar to the training distributions than the random sample. Let it be noted that these metrics merely evaluate latent similarity, which does not necessarily imply melodic or rhythmic consistency or musicality on its own.

D. Limitations and Potential Solutions

Future work could compare our architecture to diffusion models or GANs. These models have shown promise, especially with image generation. GANs in particular hep address issues with evaluation since they can be trained to imitate human listeners. Several papers previously mentioned [2] [5] [9] using diffusion.

Another area of improvement is the evaluation metrics. More work needs to be done in the subject of evaluating the quality of musical samples. Human evaluation is slow and does not scale, while it is difficult quantify musical "goodness". Importantly, work is still required to concretely correlate quantitative metrics with subjective ones.

A major critique of the generated samples is the model's poor understanding of rhythm. Several remedies for this have been proposed; for example, Li and Sung [12] propose separating the embedding and processing of rhythm and pitch. Other solutions include increase embedding context size even more, similar to [2].

Additionally, a modified melody extraction scheme during data processing could better separate the different parts represented by different instruments or parts and allow multi-track generation.

IV. Conclusion

We have shown that our data processing pipeline, with an autoencoder and transformer mixture model, produces music that is better than the baseline of random. This has been done through quantitative metrics, and indicates the model has learned aspects of music, notably tonality, mean note duration and pitch variance well.

We encountered severe limitations related to the availability of compute. This forced us to reduce the size of our model and length of training compared to state-of-the-art models.

Despite this, training was effective on a significantly truncated dataset and reduced model size. Furthermore, the adjusted training scheme proved effective in mitigating weaker compute capability and ensuring timely training. While more compute ability will likely enhance the result, the proposed measures lower the difficulty of training significantly.

Limitations and possible solutions have also been proposed, notably regarding rhythm and cadence, which remain to be implemented. Since the model was effective at being tonally coherent, addressing rhythm is a top priority.

References

[1] C. M. Bishop, "Mixture density networks," 1994.

[2] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, "Symbolic music generation with diffusion models," 2021.

[3] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," 2019.

[4] R. M. T. M. Majidi, "A combination of multi-objective genetic algorithm and deep learning for music harmony generation," *Multimedia tools and applications*, vol. 82, pp. 2419–2435, 01 2023.

[5] M. W. Y. Lam, Q. Tian, T. Li, Z. Yin, S. Feng, M. Tu, Y. Ji, R. Xia, M. Ma, X. Song, J. Chen, Y. Wang, and Y. Wang, "Efficient neural music generation," 2023.

[6] P. Li, B. Chen, Y. Yao, Y. Wang, A. Wang, and A. Wang, "Jen-1: Text-guided universal music generation with omnidirectional diffusion models," 2023.

[7] K. A. V. M. B. K. W. Jing Zhao, David Taniar, "Multi-mmlg: a novel framework of extracting multiple main melodies from midi files," pp. 22 687–22 704, 08 2023.

[8] J. Tang, L. Yin, and J. Yu, "Generation of western piano music based on deep learning," in *2022 International Symposium on Advances in Informatics, Electronics and Education (ISAIEE)*, 2022, pp. 524–527.

[9] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 4172–4182.

[10] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Columbia University, 2016.

[11] "Classical archives: Midi," 2024.

[12] S. Li and Y. Sung, "Mrbert: Pre-training of melody and rhythm for automatic music generation," *Mathematics*, vol. 11, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2227-7390/11/4/798

Appendix

Fig. 3: Mean Pitch for each Sliding Window Aggregated over 10 Examples

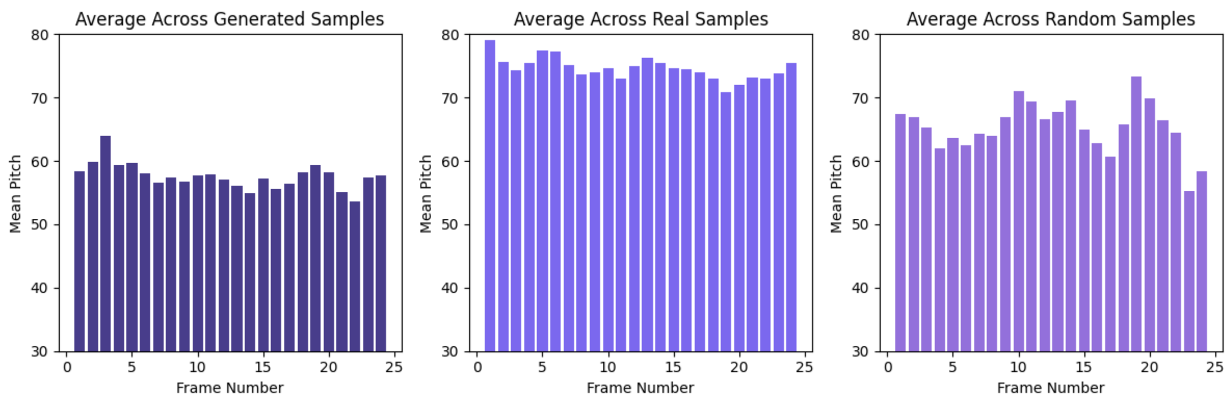


Fig. 4: Mean Note Duration for each Sliding Window Aggregated over 10 Examples

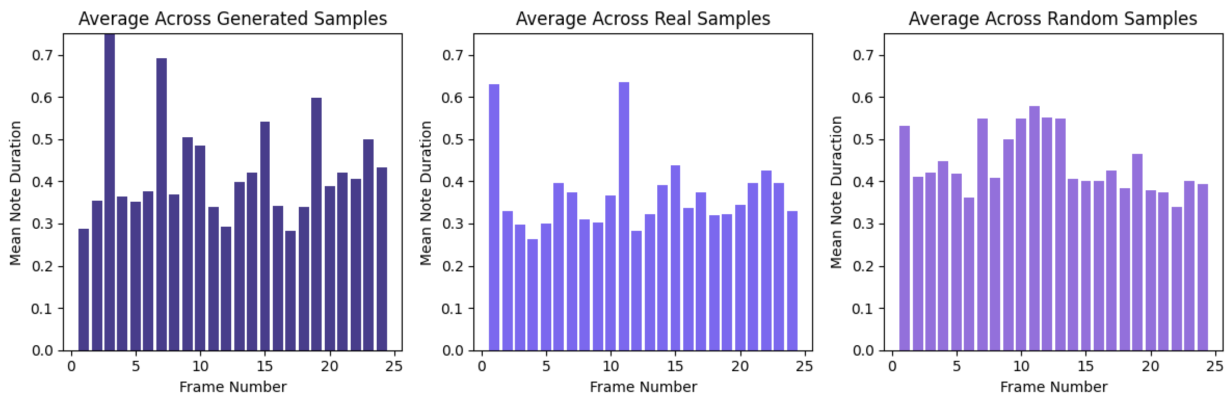


Fig. 5: Pitch Range for each Sliding Window Aggregated over 10 Examples

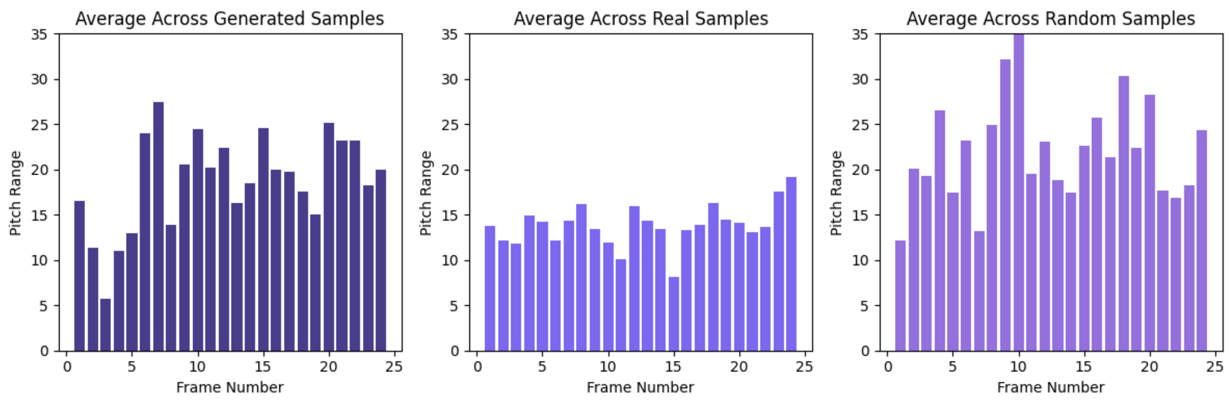


Fig. 6: Variance of Pitch for each Sliding Window Aggregated over 10 Examples

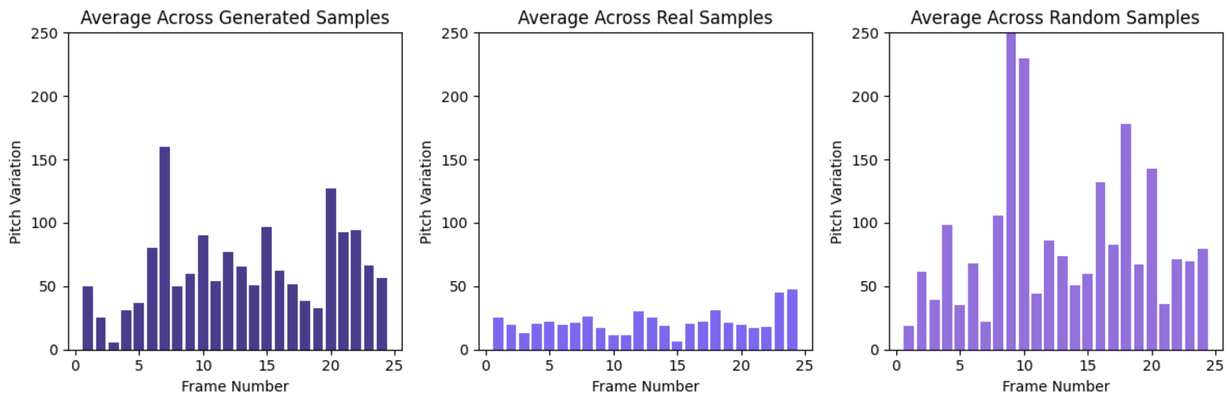
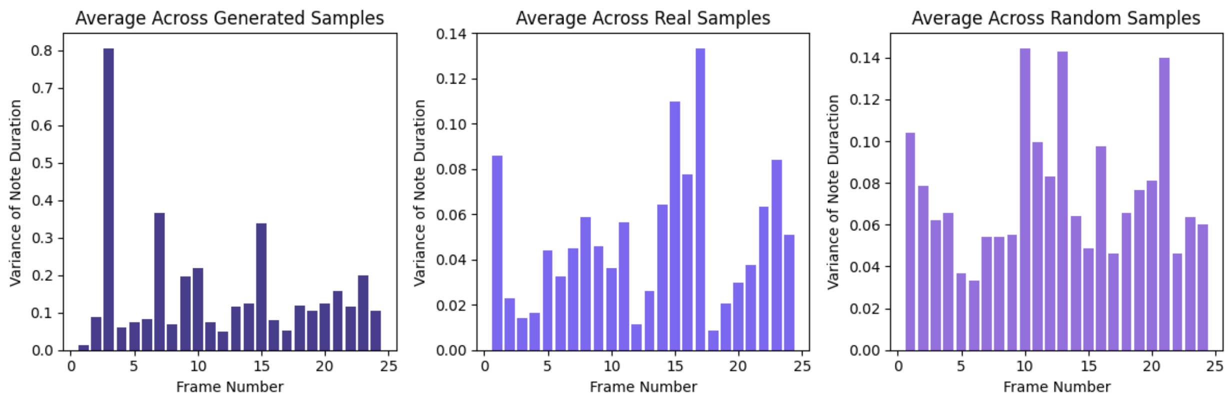


Fig. 7: Mean Variance of Note Duration for each Sliding Window Aggregated over 10 Examples


Average Across Generated Samples
Average Across Real Samples
Average Across Random Samples
Variance of Note Duration
Variance of Note Duraction
Frame Number