Auto-Dynamics: Transformer-Generated Interpretations of Piano Music

ABSTRACT

In the past decade, there have been large advancements in the field of music processing due to the application of neural networks to classic tasks such as music generation, transcription, and classification. This work concentrates rather on generating musical interpretation, which is a set of custom changes the performer may add to a piece of music during performance to convey personal expression. Specifically, we focus on auto-generating dynamics, or custom volumes, for a given series of notes. We evaluate our dynamics model in five categories and demonstrate that the Transformer architecture works effectively in generating artistically expressive performances. These findings suggest that the Transformer has potential to excel in subjective areas such as artistic style.

Introduction

In recent years, research projects applying deep learning techniques to the field of music have generally focused on generation and classification tasks [1-4]. There is less research on musical interpretation, which is defined as a performer's personal "style" and which may involve unwritten, personalized changes in timing, volume, or tone that heighten the emotional value of the performance. For the scope of this paper, we will consider the problem of generating a set of personalized volumes, known as "dynamics," when given a series of piano MIDI note events.

Dynamics generation is a highly complex task because performers consider a variety of factors including pitch, harmony, and thematic significance when deciding on the volume for any single note. Deep learning, and specifically the recent Transformer architecture, is particularly suited to this task given the need for each note to glean information from other notes.

In this paper, we demonstrate that the Transformer model can be used to generate remarkably human-like dynamics given a set of MIDI note events such as in online sheet music (which often lacks velocity). After converting the input into a piano roll-style format, we use that modified input as encoding and train an encoder-decoder model to decode it into a sequence of note events with personalized velocity.

Regarding further implications, this work adds on to previous research [5] in suggesting that Transformer models work effectively in abstract, subjective areas such as artistic style.

Related Work

Transformers in Music Processing

The Transformer architecture has enabled significant progress in the field of Music Information Retrieval (MIR) in recent years, with improvements in genre classification [24], instrument recognition [25], automatic music transcription [1], and music generation [18, 23]. In several of these areas, the Transformer successfully replaced older architectures such as the Recurrent Neural Network (RNN) [14] to achieve significantly improved, state-of-the-art results.

Dynamics and Style Generation

One of the earliest attempts at generating musical interpretations was the "Director Musices" project from the KTH Royal Institute of Technology, a classical algorithm published in 2000 that attempted to generate dynamics, expressive timing, swing, etc. for a given score based on a set of twenty-four rules [15]. However, such classical algorithms cannot capture the full complexity of musical expression, often mishandling areas where multiple rules conflict as well as failing to consider structural aspects such as thematic significance or melody recognition.

Since the popularization of neural networks and the attention mechanism, there have been a few projects aiming to generate stylistic performances using RNNs [16-17]. Most recently, Iman Malik's 2017 paper "Neural Translation of Musical Style" [17] details a successful attempt at generating musical dynamics using the StyleNet RNN. However, the recent Transformer architecture has several advantages over the RNN, including better handling of long-term dependencies due to having direct access to previous steps instead of relying on sequential processing.

Methods

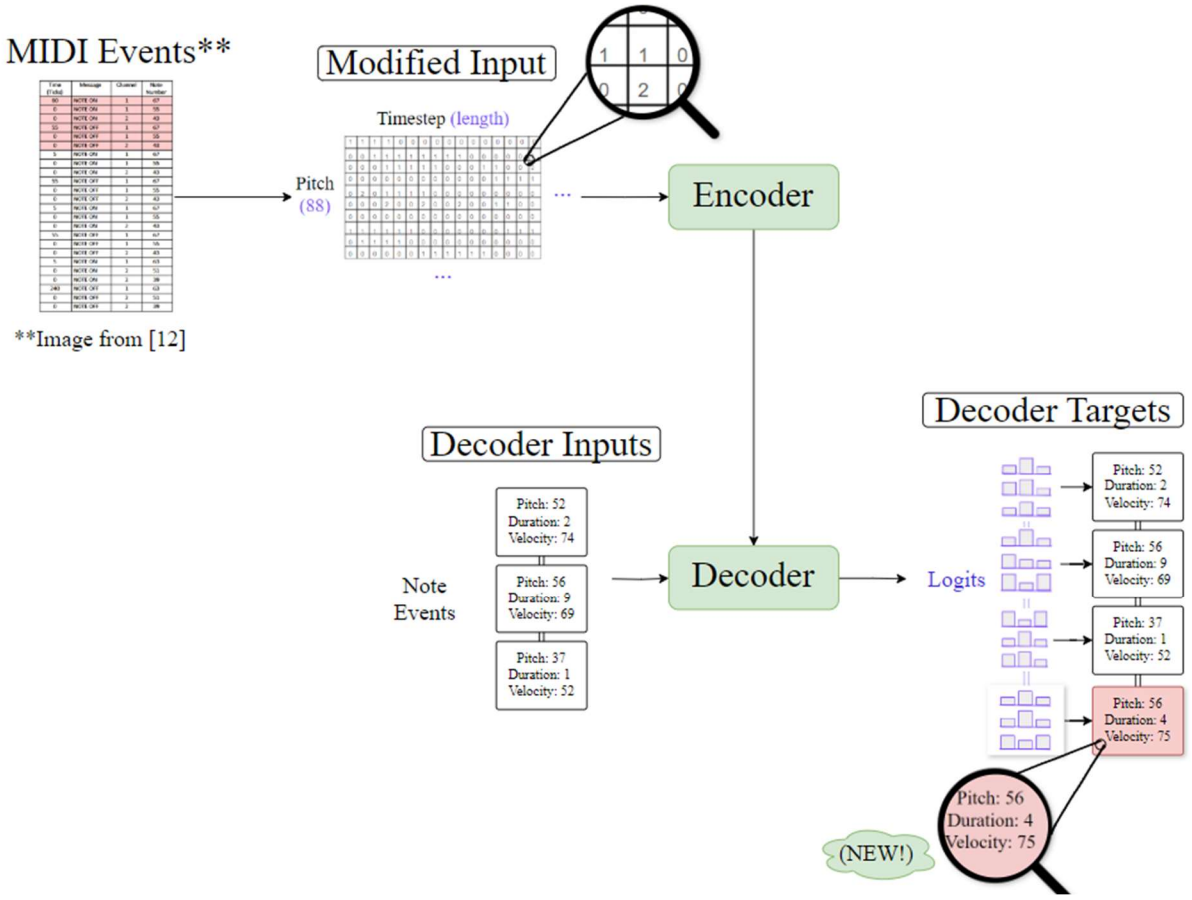

Figure 1. Diagram of Model Architecture.

Model Architecture

The model uses the T5X framework [6] which is based on JAX [7] and Flax [8] with SeqIO [6] used for preprocessing and Adafactor [9] for optimization. The parameters are largely based on T5's [10] base configuration, with a few exceptions:

Table 1. Model default parameters.

Decoder Layers	12
Encoder Layers	12
Number of Attention Heads	12
Vocabulary Size	128
Embedding Size	768
Key/Value Dimension	64

Feed-Forward Dimension	2048

This gives the model around 200M parameters, which is on the smaller side compared to larger translation models with several billion parameters. Its smaller size makes this model trainable on a consumer grade 4090 GPU card.

Dataset

We use the MAESTRO V3.0.0 dataset [21] based on the Yamaha Piano-e-Competition [22] to train our model, which contains 1276 performances and approximately 200 hours of data. The training split is used for training and contains 962 pieces. We do not model *sostenuto* or *una corda* pedal events.

Inputs and Outputs

During preprocessing, we convert the input MIDI file into a time-by-pitch matrix similar to Iman Malik's setup [17].

Timestep - Similar projects have traditionally used units of 10 ms for time due to it being the reported limit of human perception [1], but we use 50 ms to account for delays from key displacement---the world record of most piano key hits in one minute is 958, meaning one hit every ~60 ms. A larger time unit has the benefit of providing more context within the same input length.

Pitch - 88 values are used corresponding to the 88 keys on a piano.

Matrix Values - The matrix contains values in the range 0 to 2: 0 meaning note off, 1 meaning note on, and 2 meaning the note is articulated shortly.

The model output is a one-dimensional array, with each set of three predictions representing a note event. The three predictions are respectively pitch, duration, and velocity; although pitch and duration were both given in the input, the model had difficulty interpreting the input format to produce a set of pure velocity predictions. Having the model deduce pitch and duration forced it to extract features relevant to predicting velocity. However, the deduced pitch and duration are not considered during inference.

Additionally, in the case of multiple notes occurring at once, the events are ordered within the target tokens from lowest pitch to highest.

Since input files are often long, randomized samples with maximum length size are selected from each file for training, where the number of samples selected is proportional to the number of notes in the file.

Evaluation

Loss

Since this is a regression model, mean squared error loss (MSE) is used in place of T5X's provided cross entropy loss. Upon obtaining a set of logits, we take its softmax and calculate the MSE between its softmax and a one-hot vector with the target value. The MSE values are then added across the entire sequence to produce the total loss.

Result Evaluation Metric

In evaluating overall model performance in comparison to other models, we found MSE to be a better metric than other regression metrics such as mean absolute error (MAE) due to greater discouragement of outliers. Thus, MSE is used for evaluation in the Results section.

Results

Base Model

For the default model, we used a constant learning rate of 1e-3 and a dropout of 0.2. Input length was set to 256, giving an input size of 12.8 seconds, with a corresponding output length of 2048 to accommodate the large amount of notes possibly occurring within that time. Batch size was set to 2 due to memory constraints, and to prioritize a larger context size.

Training was done on a Dell Alienware Aurora R16 with NVIDIA RTX 4090 GPU, Intel i9 14900KF CPU, and 64G system memory. Models were trained for ~3M steps unless otherwise specified, which took about 2 days for the default model.

Base Model Evaluation

Since it is challenging to compare it to Iman Malik's previous work, we instead perform an in-depth evaluation of the default model's ability in five areas. All excerpts used are from the test and evaluation sets of MAESTRO V3.0.0 [21].

Melody Recognition

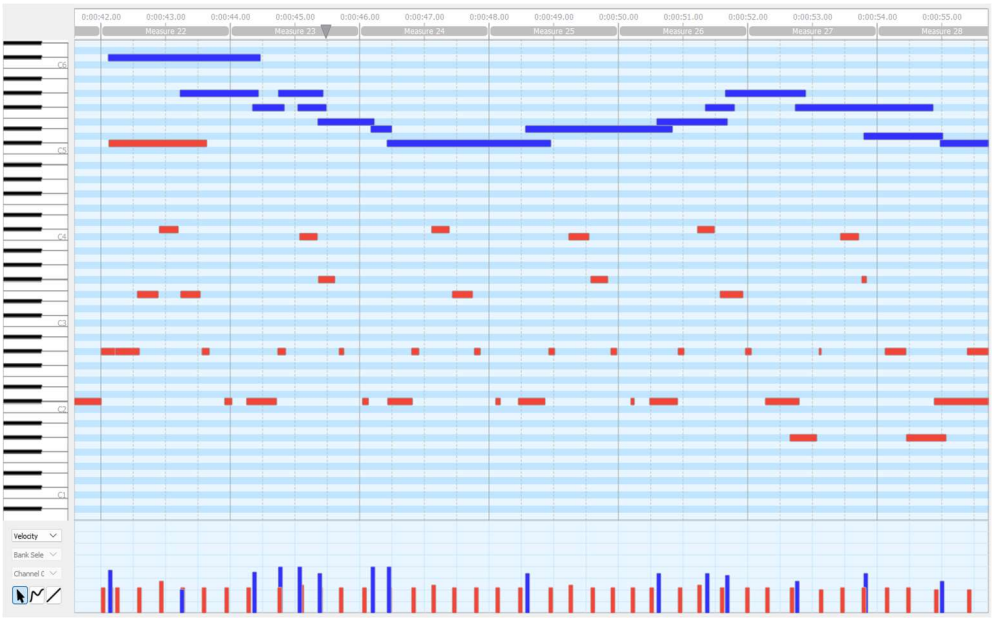

Figure 2. An excerpt from Chopin's Nocturne Op. 27 No. 1 in c-sharp minor with dynamics as generated by the base model. Visuals are from the MidiEditor app [26].

In this excerpt, the model successfully differentiates between the melody line and the accompaniment. The melody can be seen as the set of horizontal lines highlighted blue in the top third of the image, and the accompaniment consists of the notes below the melody. The line of vertical bars at the bottom of the page correspond to the volumes of the notes, with each note having a matching vertical bar.

The generated velocities for this sample are mostly in the quiet 40-50 range, except for the melody line, which is brought out clearly and loudly (see the smattering of taller blue bars at the bottom of the page). This indicates a successful attempt at recognizing and prioritizing the melody.

Voicing

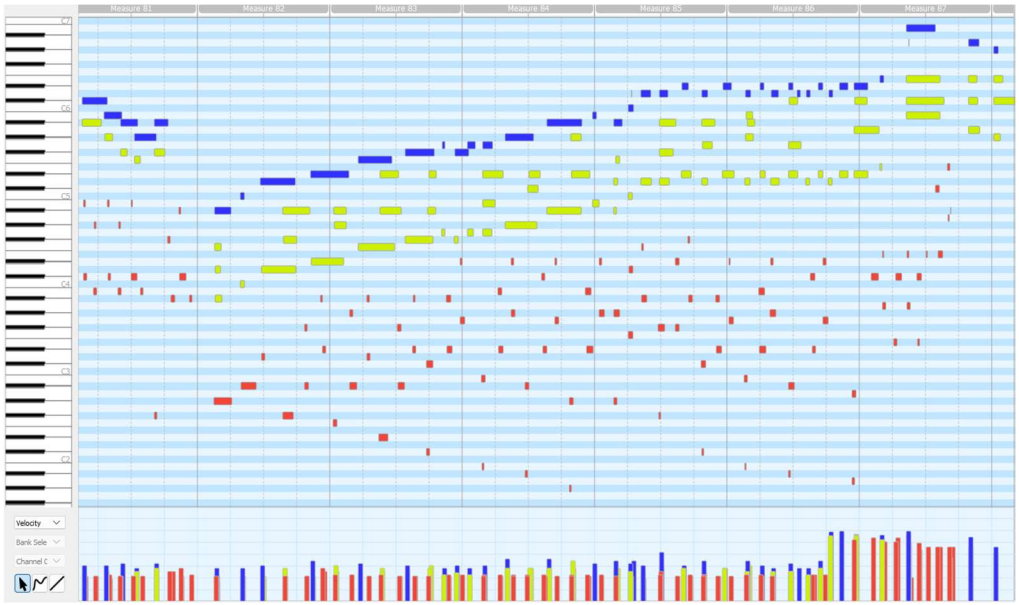

Figure 3. An excerpt from Rachmaninoff's Elegie Op. 3 No. 1, with dynamics as generated by the base model. Visuals are from the MidiEditor app [26].

In piano music, chords (three or more notes played at the same time) are generally "voiced," meaning the most melodically important note within the chord is played louder than the others. Generally, the most important note is the top note.

The excerpt has three sections highlighted in different colors for convenience: the accompaniment is depicted in red, with a line of chords above it in green and blue. The blue depicts the most melodically important note, and the green depicts the rest of the chord which helps harmonize the melody.

The model distinguishes accurately between the three and treats them appropriately. The blue melody line has the tallest bars, indicating that it is correctly "voiced." The accompaniment is represented by short red bars at the bottom, indicating that it is evenly quiet despite the large range of different pitches. The green bars are generally taller than the red and shorter than the blue.

Phrasing

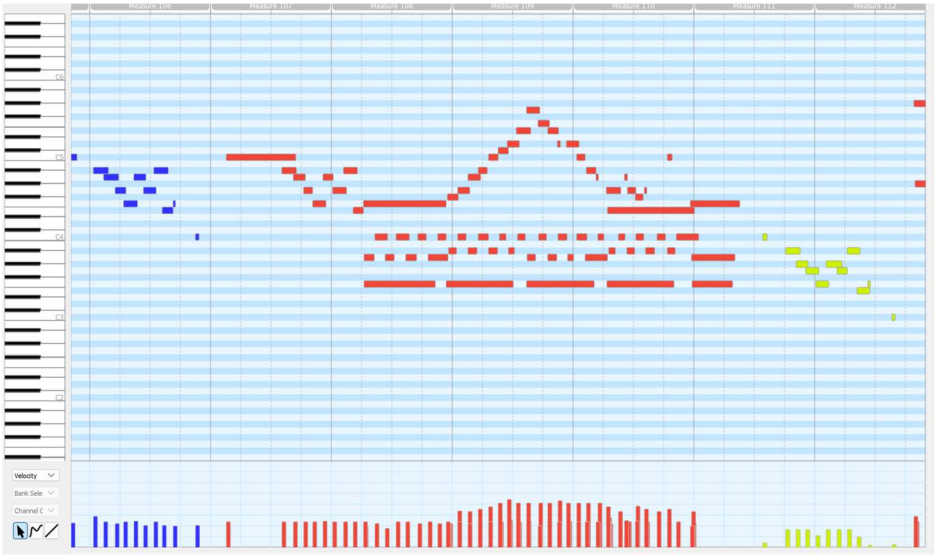

Figure 4. An excerpt from Mozart's Sonata No. 15 in F Major, K533, with dynamics as generated by the base model. Visuals are from the MidiEditor app [26].

Musical "phrasing" refers to how a sequence of notes may have different individual volumes to convey emotion and expression. A general rule of phrasing is that higher notes are played louder. In fact, "Director Musices" project [15] mentioned in the Related Work section had this rule as the first of twenty-four.

In this excerpt, the mountain-shaped notes in the middle are successfully accompanied by slightly taller bars at the bottom.

Another rule of phrasing is that the same sequence of notes should not be repeated the same way twice. The notes highlighted in blue correspond with the notes highlighted in green, and the model successfully differentiates them by making the second set softer (the green bars are shorter than the blue bars).

Harmonic Understanding

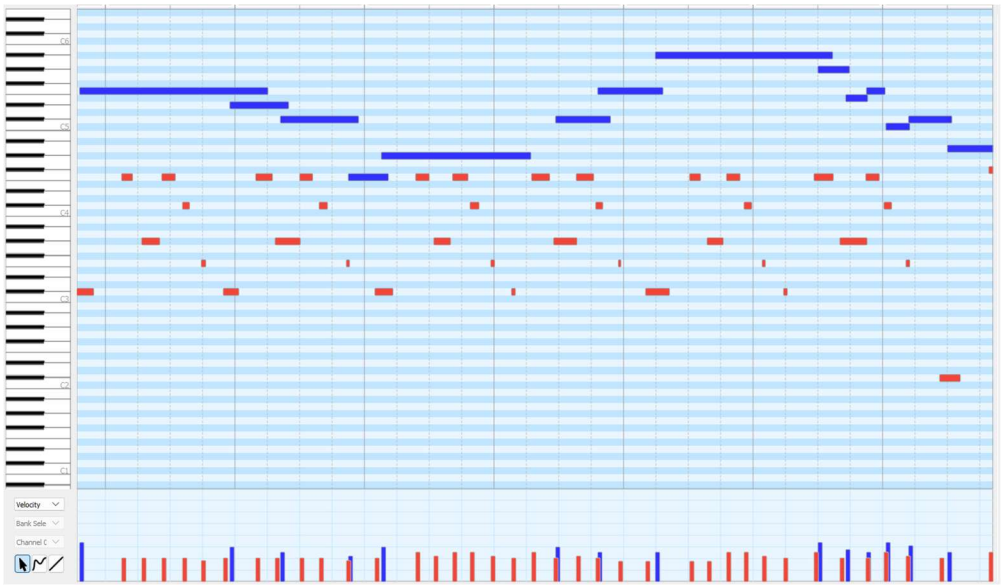

Figure 5. An excerpt from Chopin's Nocturne Op. 27 No. 2 in D-flat Major, with dynamics as generated by the base model. Visuals are from the MidiEditor app [26].

As mentioned above, a general rule of phrasing is that higher notes are played louder. However, performers often choose to override this rule if it conflicts with other rules.

In this excerpt, the highest note of the melody (a Bb5) should theoretically be played the loudest. However, there is a conflicting harmonic rule which dictates that it should be played softer: everything before this note is firmly within the D-flat Major tonic chord, and this note not only ventures outside of the tonic, but is also in the minor key.

The model successfully overrides the "higher notes are louder" rule to play this note softly, which demonstrates that it has some level of harmonic understanding. It is likely that this is not a fluke, since the previous melody notes (all highlighted in blue) follow the "higher notes are louder" rule.

Polyphonic Music and Counterpoint

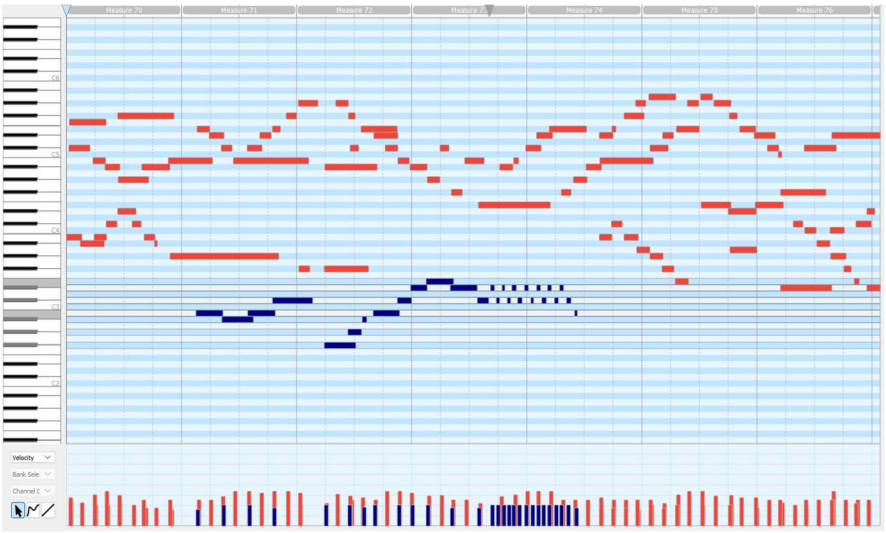

Figure 6. An excerpt from Bach's Prelude and Fugue in B Major, WTC I, BWV 868, with dynamics as generated by the base model. Visuals are from the MidiEditor app [26].

Polyphonic music is defined as music with multiple independent melodic lines played at once. The model is successfully able to distinguish between the different melodic lines by making certain melodies softer, as seen in the excerpt above (the line highlighted in blue is notably softer than the others).

However, some polyphonic compositions such as the fugue are based on a repeated motif that the melodic lines take turns performing. It is common practice within these compositions to emphasize the motif whenever it appears.

In fact, the notes highlighted in blue above represent an iteration of its fugue's motif. Unfortunately, the model did not successfully recognize this passage as a motif, possibly due to either lack of context or lack of training data. This is a possible area for improvement.

Ablation Studies

Table 2. Loss results for various parameters.

Model	MSE Value (averaged across selected "test" category performances
DEFAULT MODEL	760.5
LOSS: Double MSE with one-hot vector	770.9
LOSS: Double MSE with modified one-hot	793.2
LOSS: Single MSE with modified one-hot	905.6
SMALLER MODEL: • Encoder layers: 8 • Decoder layers: 8 • Number of heads: 6 • Embedding Size: 512	752.2
BATCH SIZE: 1	786.2
BATCH SIZE: 8	817.4
LARGER INPUT: • Input length: 512 • Output length: 4096 • Feed-Forward Dimension: 4096	915.7

• Batch size: 1 • Encoder layers: 8 • Decoder layers: 8 • Number of heads: 6 • Embedding Size: 768	

Different Loss Functions - In the default model, MSE is used a single time between the softmax'd logits and a one-hot vector containing the target value. In experimenting with alternative loss functions, we tried various combinations of "double MSE" and "modified one-hot." Double MSE refers to taking MSE twice: first between the logits and the one-hot vectors, then another time on those first MSE values at a sequence-level. Modified one-hot refers to replacing the traditional one-hot vector (a vector containing a single 1 in an array of 0s) with a vector using (0.5, 1, 0.5) to surround the target value rather than just (1). The goal of this modification was to reduce penalization if the model predicted a value which was only slightly off. However, the default loss (single MSE with one-hot) still had the best performance.

Smaller Model - This smaller model is based on T5X's small configuration [6]. Reducing the model size did not have a significant impact on performance.

Larger Input - To experiment with different context sizes, we doubled the input size, resulting in an input length of 25.6 seconds. To accommodate memory constraints, we used a smaller version of the model along with reduced batch size. The larger input did not seem to improve performance.

Conclusion

In this work, we have shown that the Transformer model can be used to generate artistically viable, human-like dynamics given a set of MIDI note events.

Related areas of research include whether it is possible to generate expressive timing and tone, in addition to volume. There are other aspects which may influence dynamics generation and are not considered in this model, such as the year when the music was written, whether it is homophonic or polyphonic, and any written instructions listed by the composer on the score. Also worth exploring is using sparse attention mechanisms to evaluate the entire piece with consideration of its long-term structure, a concept which has previously been successful in the field of music generation [18].

In a broader context, this work suggests that Transformer models in general work effectively in understanding abstract human notions such as "style," and paves the way for future research in emulating specific styles in various fields of art.

Limitations

This model is trained on a dataset of Western classical music and confined to piano only. Other instruments may rely on different aspects such as vibrato or rhythm to determine expressive dynamics, or they may rely on other metrics than dynamics for musical expression. Dynamics may also vary depending on the genre; for example, jazz has a different set of rules for expressive dynamics than classical music (see [17] for a model trained on jazz datasets). Additionally, this model lacks a way to represent several characteristics prominent in non-Western musical tradition, including ornamentation/glissandos and microtones as used in Persian, Indian, and Southeast Asian music along with Western genres such as blues. There is less available data for these musical traditions as well, although a few exist. See [19] and [20] for examples of models trained on datasets of non-Western music.

Acknowledgements

I would like to thank my teacher, Mr. Ferrante, for his support.

References

[1] Gardner, J., Simon, I., Manilow, E., Hawthorne, C., & Engel, J. (2022). MT3: Multi-task Multitrack Music Transcription. International Conference on Learning Representations. https://doi.org/10.48550/arXiv.2111.03017

[2] Dong, H., Hsiao, W., Yang, L., & Yang, Y. (2018). Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. Proceedings of the AAAI Conference on Artificial Intelligence, 2018. https://doi.org/10.48550/arXiv.1709.06298

[3] Copet, J. Kreuk, F., Gat, I. etc (2023) Simple and Controllable Music Generation. arXiv preprint arXiv:2306.05284 https://doi.org/10.48550/arXiv.2306.05284

[4] Bittner, R., Bosch, J., Rubinstein, D. etc (2022). A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation. ICASSP 2022. https://doi.org/10.1109/ICASSP43922.2022.9746549

[5] Deng, Y., Tang, F., Dong, W., Ma, C., Pan, X., Wang, L., & Xu, C. (2022). Stytr2: Image style transfer with transformers. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr52688.2022.01104

[6] Robert, A., Chung, H., Mishra, G. Levskaya, A., etc. (2022). Scaling Up Models and Data with t5x and seqio. arXiv preprint arXiv:2203.17189. https://doi.org/10.48550/arXiv.2203.17189

[7] Bradbury, J., Frostig, R., Hawkins, P., etc (2018). JAX: composable transformations of Python+NumPy programs. http://github.com/google/jax

[8] Heek, J., Levskaya, A., Oliver, A., (2023). Flax: A neural network library and ecosystem for JAX. http://github.com/google/flax

[9] Shazeer, N., Stern, M., (2018). Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. https://doi.org/10.48550/arXiv.1804.04235

[10] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text Transformer," Journal of Machine Learning Research, vol. 21, no. 140, pp. 1–67, 2020.

[11] MIDI Manufacturers Association. (1996). The complete MIDI 1.0 detailed specification. Los Angeles, CA, The MIDI Manufacturers Association, 1996. ISBN 097288310X

[12] Müller, M. (2021). Fundamentals of Music Processing: Using Python and Jupyter Notebooks. Springer International Publishing. ISBN-13 978-3030698072

[13] Vaswani, A., Shazeer, N., Parmar, N. etc (2017). Attention Is All You Need. arXiv preprint arXiv:1706.03762. https://doi.org/10.48550/arXiv.1706.03762

[14] Hochreiter, S., Schmidhuber, J. (1997) Long Short-term Memory. Neural computation 9 (Dec. 1997), pp. 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[15] Friberg, A., Colombo, V., Frydén, L., Sundberg, J. (2000) Generating Musical Performances with Director Musices. Computer Music Journal, 24:23–29. https://doi.org/10.1162/014892600559407

[16] Lauly, S. (2010). Modélisation de l'interprétation des pianistes & Applications d'auto-encodeurs sur des modèles temporels.

[17] Malik, I., Ek, C. (2017). Neural Translation of Musical Style. arXiv preprint arXiv:1708.03535. https://doi.org/10.48550/arXiv.1708.03535

[18] Huang, C., Vaswani, A., Uszkoreit, J. etc (2018). Music Transformer: Generating Music with Long-Term Structure. arXiv preprint arXiv:1809.04281, https://doi.org/10.48550/arXiv.1809.04281

[19] Reghunath, L.C., Rajan, R. (2022). Transformer-based ensemble method for multiple predominant instruments recognition in polyphonic music. J AUDIO SPEECH MUSIC PROC. 2022, 11 (2022). https://doi.org/10.1186/s13636-022-00245-8

[20] Natesan, S., Beigi, H., (2024). Carnatic Raga Identification System using Rigorous Time-Delay Neural Network. arXiv preprint arXiv:2405.16000. https://doi.org/10.48550/arXiv.2405.16000

[21] Hawthorne, C., Stasyuk, A., Roberts, A. etc. (2019). Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. International Conference on Learning Representations, 2019. https://openreview.net/forum?id=r1lYRjC9F7

[22] Yamaha International Piano-e-Competition. http://www.piano-e-competition.com/

[23] Huang, C., Cooijmans, T., Roberts, A., etc. (2019). Counterpoint by Convolution. International Society for Music Information Retrieval 2019. https://doi.org/10.48550/arXiv.1903.07227

[24] Roberts, A. Engel, J. Raffel, C. etc. (2018). A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. International Conference on Machine Learning 2018. https://doi.org/10.48550/arXiv.1803.05428

[25] Szeliga, D., Tarasiuk, P., Stasiak, B., Szczepaniak, P. (2022). Musical Instrument Recognition with a Convolutional Neural Network and Staged Training. KES 2022. https://doi.org/10.1016/j.procs.2022.09.307

[26] Schwenk, M. (2019). MidiEditor: Graphical interface to edit, play, and record Midi data. https://github.com/markusschwenk/midieditor