Performance Analysis of Selected Generative Models

Bryan Healy
bhealy8@gatech.edu

***Abstract*—To build a responsive interface for interacting with a music-generating AI model, the model itself must have low inference latency. By choosing a model with low inference latency, the musician will spend less time waiting for the model and more time creating with it. The goal of the project thus far was to find a model capable of multiple music generation tasks that fits within the constraints of the Coral Dev Board system resources. By minimizing system resource usage, more resources will be left for other music generation features and the UI.**

1 INTRODUCTION

1.1 Magenta Framework

Google has created an open source library of AI models that can be used for various musical tasks, along with pipelines to handle various types of data. This library was chosen for its thoroughness, ease of use, and continued development (Magenta/Magenta/Pipelines at Main · Magenta/Magenta, n.d.).

1.2 Melody RNN

The first models examined were a set of Recurrent Neural Networks (RNNs) called Melody RNN, which were based upon Long Short-Term Memory (LSTM) cells (Magenta/Magenta/Models/Melody_Rnn at Main · Magenta/Magenta, n.d.).

1.2.1 Lookback

The first variant of Melody RNN examined used custom inputs and labels to recognize longer-term musical patterns over 1-2 bars.

In addition to information about the previous timestep, the model also adds several inputs, including the events from 1-2 bars beforehand, a feature which signaled if the previous event was a repetition of what was generated 1-2 bars before, and the current position within the measure.

They also added two new custom labels, one to repeat the event from 1 bar ago and another to repeat the event from 2 bars ago. By including this feature in the training data, the model will learn when to repeat the previous events (Generating Long-Term Structure in Songs and Stories, 2016).

1.2.2 Attention

Another of the Melody RNN models created by Magenta is also based around LSTM cells. However, this one uses the Attention mechanism to store information about previous events and learn longer-term musical structure (Bahdanau et al, 2014). Attention learns two weight vectors, one is applied to the n (chosen constant) previous outputs and the other is applied to the current step's LSTM state. An "attention mask" is then created using a softmax to determine the amount of attention received by each of the n previous steps. After applying the mask a vector $h_t^{'}$ representing the combination of previous steps is created summing the contribution of each. The $h_t^{'}$ vector is then concatenated with the current RNN output and combined in a linear layer to produce the past-informed output. The $h_t^{'}$ vector is also used as additional input for the next step, maintaining longer-term memory as the generation progresses (Generating Long-Term Structure in Songs and Stories, 2016).

1.3 Polyphony RNN

Polyphony RNN is another LSTM-based music generation model from Magenta. In this model, they focused on generating music with multiple simultaneous notes. This was done by representing the music as a stream of quantized steps (4 per quarter note) (Magenta/Magenta/Models/Polyphony_Rnn at Main · Magenta/Magenta, n.d.). For example a C Major chord played for a single eighth note would be represented as follows:

START

NEW_NOTE, 67

NEW_NOTE, 64

NEW_NOTE, 60

STEP_END

CONTINUED_NOTE, 67

CONTINUED_NOTE, 64

CONTINUED_NOTE, 60

STEP_END

CONTINUED_NOTE, 67

CONTINUED_NOTE, 64

CONTINUED_NOTE, 60

STEP_END

CONTINUED_NOTE, 67

CONTINUED_NOTE, 64

CONTINUED_NOTE, 60

STEP_END

END

1.4 Music VAE

MusicVAE is a set of models created by Magenta based on a Variational Autoencoder, which learns a latent space for the musical characteristics of the dataset. This is done by first encoding the input sequence into a latent code to reduce its dimensionality and learn fundamental characteristics of the dataset. Next, the latent code is passed to a "conductor" RNN which produces an embedding for each bar of the sequence. Finally, these bar encodings are passed to a note decoder to produce the output sequence.

The MusicVAE model can also be trained to learn a latent space for multiple simultaneous instruments by passing the encoding for each bar to multiple decoders, each producing a sequence for its corresponding instrument.

These models are also capable of more than just conditioned generation. They are also capable of interpolating between two melodies, and allow for controlling various musical characteristics such as note density by using attribute vector arithmetic. The hierarchical architecture of MusicVAE also makes it more adaptable to varying length sequences and produces sequences with more

long-term structure than the LSTM-based models (MusicVAE: Creating a Palette for Musical Scores With Machine Learning., 2018).

2 PERFORMANCE TESTING

Each model tested used weights from pretrained checkpoints and was tasked with generating a 16 bar (256 sample) melody given the first 14 notes of "Twinkle Twinkle Little Star" as a primer sequence. Each test was run for 100 iterations on the same system, without GPU or TPU acceleration.

2.1 Inference Time

Table 1 shows the average inference time and memory usage of the generation task over 100 repetitions.

Table 1—Average performance of various music generation models over 100 repetitions.

	Melody RNN (Lookback)	**Polyphony RNN**	**Melody RNN (Attention)**	**Music VAE (Melody)**	**Music VAE (Trio)**
Time (ms) per generation	614.42	788.70	832.87	1,049.21	2,238.74
Memory Usage (MB)	201	241	248	308	670

2.2 Memory Usage

The Scalene Python package was used to profile the memory usage of the different models, the results are shown in Table 1. Along with memory usage, Scalene also provides other statistics such as the percentage of time each line spent in Python, native, or system code; and disk usage. These results are shown in the Appendix (Berger, 2020).

2.3 Architectural Analysis

The performance analysis showed that the VAE-based models were more computationally intensive than those based on LSTM cells, requiring both more memory and more processing time.

The attention mechanism requires more sequential processing than lookback mechanism due to each step's reliance on the $h_t^{'}$ vector of the previous step, along with its ability to consider more events into the past (depending on n).

Unsurprisingly, the Trio Music VAE model took much more time and resources than the Melody Music VAE model, due to the Trio variant's duplication of the output RNN for each instrument.

Also as expected the inference time increased linearly with an increase in memory usage. This was due to the sheer number of additional calculations needed for each new layer/parameters which also require more memory.

3 CONCLUSIONS

Although the Lookback Melody RNN model was the most responsive, the lookback mechanism limited its memory to only 2 bars of previous music. Attention Melody RNN, while having lower throughput, is capable of producing melodies of several bars with much more coherent musical structure.

While MusicVAE took longer to generate and required more memory, it still fits within the system resource usage criteria. With its added features and adaptability to varying length sequences, the model's reduced responsiveness is an acceptable trade-off, making it a likely candidate to use as the foundational model for the remainder of this project.

4 REFERENCES

1. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
2. Berger, E. D. (2020). Scalene: Scripting-language aware profiling for python. arXiv preprint arXiv:2006.03879.
3. magenta/magenta/pipelines at main · magenta/magenta. (n.d.). GitHub. Retrieved September 27, 2022, from https://github.com/magenta/magenta/tree/main/magenta/pipelines
4. magenta/magenta/models/melody_rnn at main · magenta/magenta. (n.d.). GitHub. Retrieved September 27, 2022, from https://github.com/magenta/magenta/tree/main/magenta/models/melody_rnn
5. magenta/magenta/models/polyphony_rnn at main · magenta/magenta. (n.d.). GitHub. Retrieved September 27, 2022, from https://github.com/magenta/magenta/tree/main/magenta/models/polyphony_rnn
6. MusicVAE: Creating a palette for musical scores with machine learning. (2018, March 15). Magenta. Retrieved September 27, 2022, from https://magenta-staging.tensorflow.org/music-vae
7. Generating Long-Term Structure in Songs and Stories. (2016, July 15). Magenta. Retrieved September 27, 2022, from https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn/

5 APPENDIX

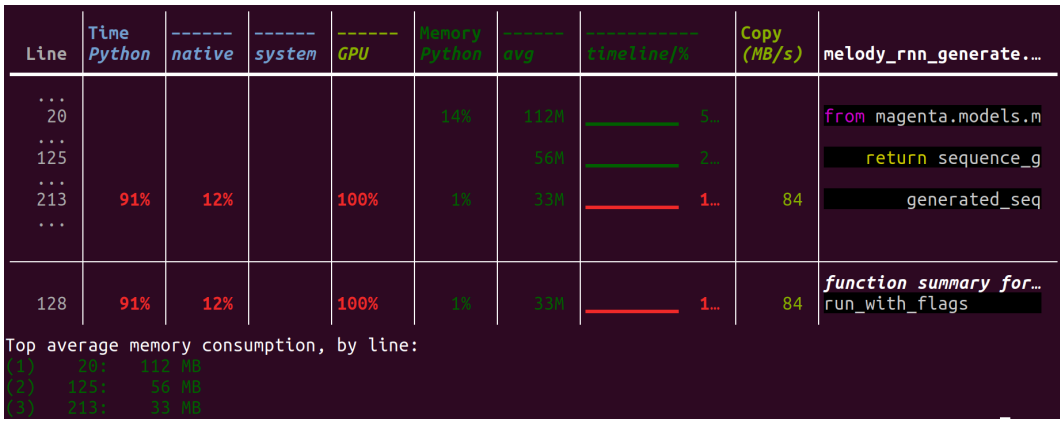

Figure 1—Scalene performance analysis of Melody RNN (Lookback).

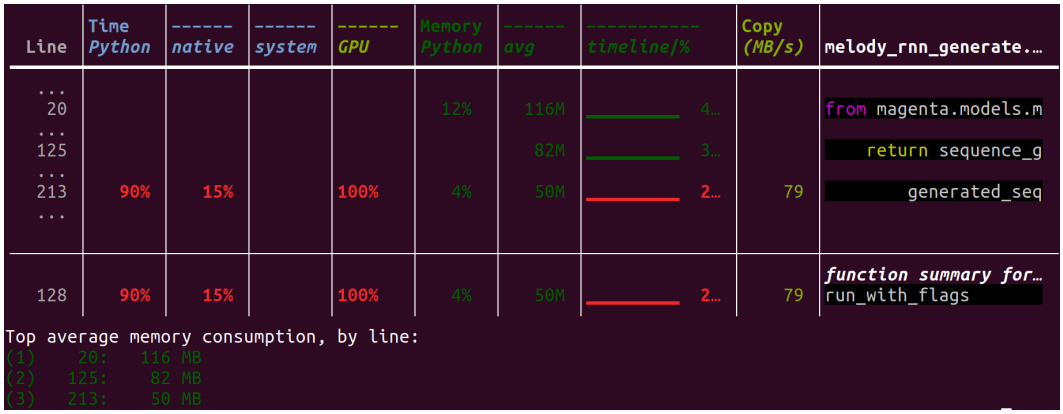

Figure 2—Scalene performance analysis of Melody RNN (Attention).

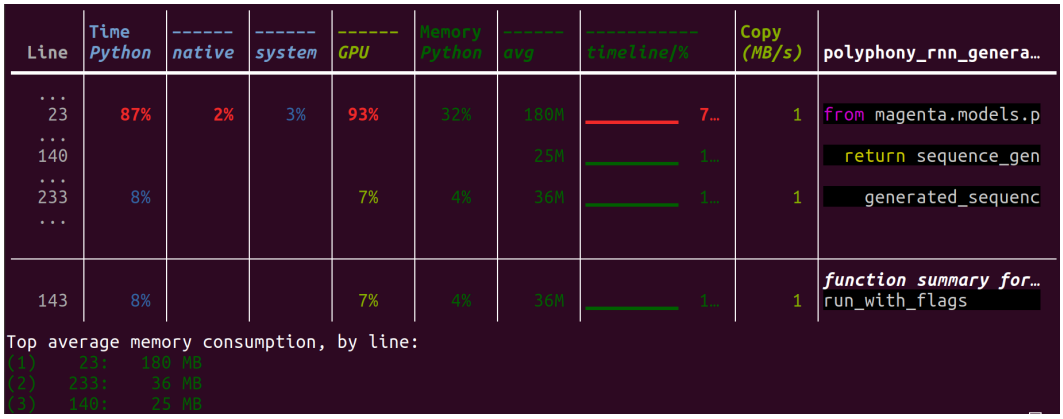

Figure 3—Scalene performance analysis of Polyphony RNN.

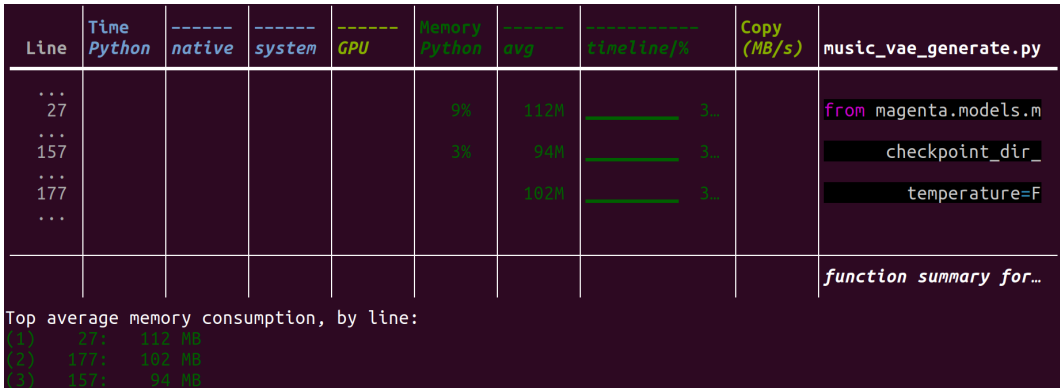

Figure 4—Scalene performance analysis of Music VAE (Melody).

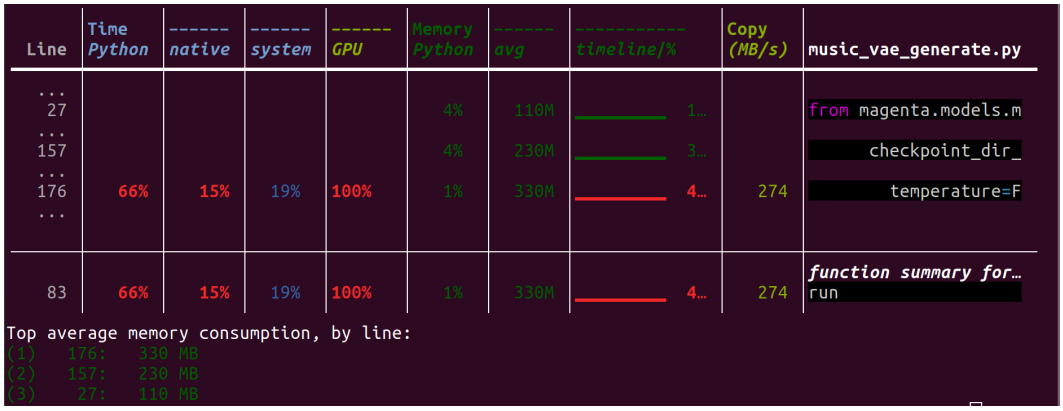

Figure 5—Scalene performance analysis of Music VAE (Trio).