Eric Lin
GenEd 1125
2022.05.09

Finetuning Music Transformers to Generate Music in Various Styles

Summary 1

Motivation 2

Background / Related Work 2

Methods 3

Results 4

Conclusions and Future Work 7

References 8

Summary

In this final project I finetuned Google's Music Transformer model to generate music in various styles (classical, jazz, and pop piano music) using transfer learning techniques.

To the best of my knowledge, my work is currently the *only* working example of transfer learning on Google's Music Transformer. An extensive search through research literature, online discussion forums with the original authors (link), and GitHub repositories show that there have been previous attempts but no publicly available working version of finetuning their Transformer. In particular, Google's Magenta initiative shows many demos of using pre-trained models, but this specific customizability functionality has been a "ToDo" item of theirs since 2020.

Code

The entirety of code, datasets, and saved models can be found in this Google Drive folder: https://drive.google.com/drive/folders/1BzyzrRfYP83LsbGk_T7o2zh6lGJpwZph?usp=sharing. The Google Colab / Jupyter notebook "Music Generation Transfer Learning.ipynb" (direct link) can be directly run using Google Cloud's servers.

Generated Music

A subset of code and generated music files can be found in my GitHub repository: https://github.com/eric-z-lin/Finetuning-Music-Transformer. Due to storage constraints, I can't fit everything on GitHub (hence the Google Drive folder above), but I have included 25+ examples of generated music files (in .wav format) and saved models trained in each of the different styles (classical, jazz, and pop piano music).

Demo
I demonstrate running this code in the following Zoom video recording: https://harvard.zoom.us/rec/share/VKwNXAYeOcHk5xcRI7bvw14WHavI_HB4d9xHM09S5gh-INkYcych8RDNE3PZECs4.uBro1YHLp6T3zkEX.

Motivation

Generating music is incredibly difficult. The normal MP3 recording saves audio files at 96 kilobits per second (CDs and other formats are at much higher quality), which is already several orders of magnitude above image files. Consequently, trying to train on 10 seconds of an audio file is computationally equivalent to training on thousands of images in a computer vision task.

Moreover, music generation is much more complicated than typical image classification tasks that neural networks have excelled in. Instead of clear image labels, music generation shares the same challenges with text generation in attempting to capture the nuances of how humans communicate with each other.

Due to this difficulty, music generation models (e.g. Google's Magenta and OpenAI's JukeBox projects) have typically been trained on massive data centers that take days if not months to finish. **As a result, trained music generation models are general in nature — the researchers behind them aren't looking to personalize models to specific composers / listeners.**

However, there is great potential for personalizing music generation models. **This work is motivated by the vision for developing a music generation model that can learn individual composers' styles and aid them as a tool in music production.** To work towards this goal, I attempt in this project to show how to (1) train models that can be finetuned to different styles of music and (2) do so using transfer learning techniques that don't require too many data points nor compute power.

Background / Related Work

There are many differing machine learning architectures to generating music. The most popular ones include recurrent neural networks (RNNs) such as Google's Performance RNN, long short-term memory (LSTMs) models, and Transformer models. As was mentioned in class, these architectures all rely on the ability to detect recurring patterns and are therefore viable for tasks like music and text generation.

I chose Google Brain's Music Transformer architecture as my baseline model. It utilizes MIDI files as data inputs. MIDI is an efficient encoding technique for music that captures the pitch, length, and velocity which notes should be struck. Compared to OpenAI's JukeBox model that works with pure audio files, it is much more efficient computationally to train on MIDI files. Incorporating velocity allows the model to learn some semblance of expression; indeed, the Google Brain team trains on a dataset called MAESTRO that aims to capture the nuances of expressions in professional concert pianists.

Without going into too much more detail, their transformer model was groundbreaking due to its success in combining the traditional absolute attention approaches in the original transformer paper ("Attention is All You Need") with more recent advances in using relative attention. In terms of music, this means that the model measures relative instead of absolute differences between pitches.

Methods

Datasets

MAESTRO: this is a dataset of classical piano performances of pianists competing in the world-renowned Yamaha e-piano competition. I used 1,300 MIDI files from this dataset.
Jazz piano: I extracted this dataset from a publicly available repository of MIDI files (link). It contains 450 different snippets of jazz piano pieces.
Pop piano: I extracted this dataset from a publicly available repository of MIDI files (link). It contains 715 different snippets of piano covers of pop songs.

These datasets can be found in the Google Drive folder I linked above.

Approach

In order to train a model that can generate music in three different styles, I conducted the following steps:

1. **Classical model**: Train the Google Music Transformer model **from scratch** on the MAESTRO dataset for 100 epochs (this took about half a day using Google Colab's NVIDIA K80 GPU).
    a. Save a copy of the trained models at each epoch.
    b. Use an adaptive learning rate that accelerates then slowly decreases over time (see graph below).
2. **Jazz model**: Transfer learn the saved copy of the classical model at the 100 epoch mark to the Jazz piano dataset (this only took 2 hours).

   a. Train 35 epochs in total, with the learning rate at 0.0003 at first then declining to 0.0002.
3. **Pop model**: Transfer learn the saved copy of the classical + jazz model at the 135 (100+35) epoch mark to the Pop piano dataset (this took less than 3 hours).
   a. Train 35 epochs in total.

The saved models can be seen in my GitHub repository.

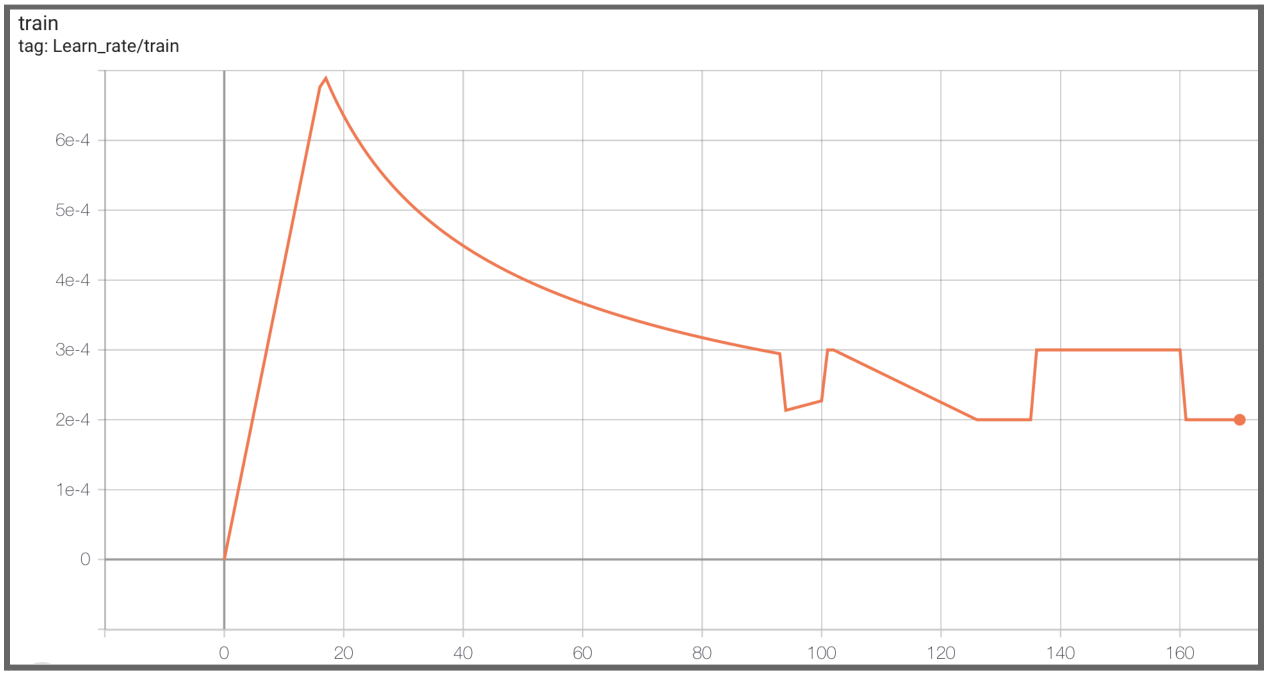

Figure 1. Learning rate over 170 epochs of training.

Results

Metrics

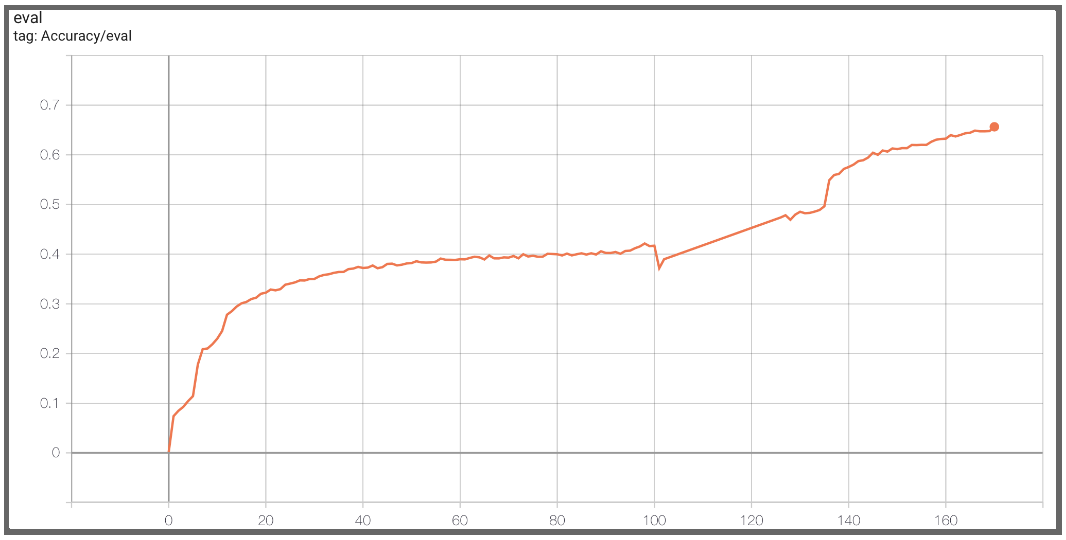

Figure 2. Validation accuracy over 170 epochs of training.

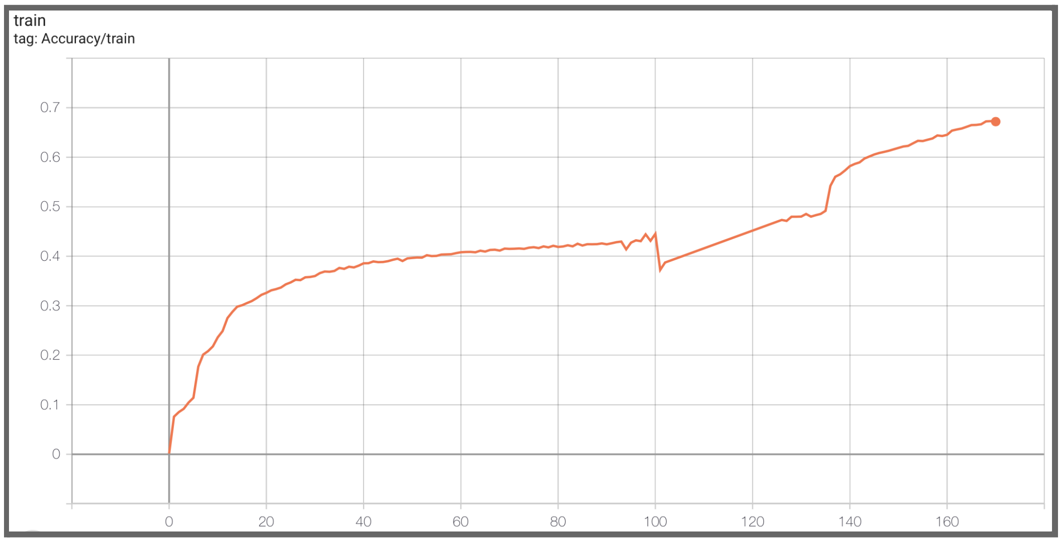

Figure 3. Training accuracy over 170 epochs of training.

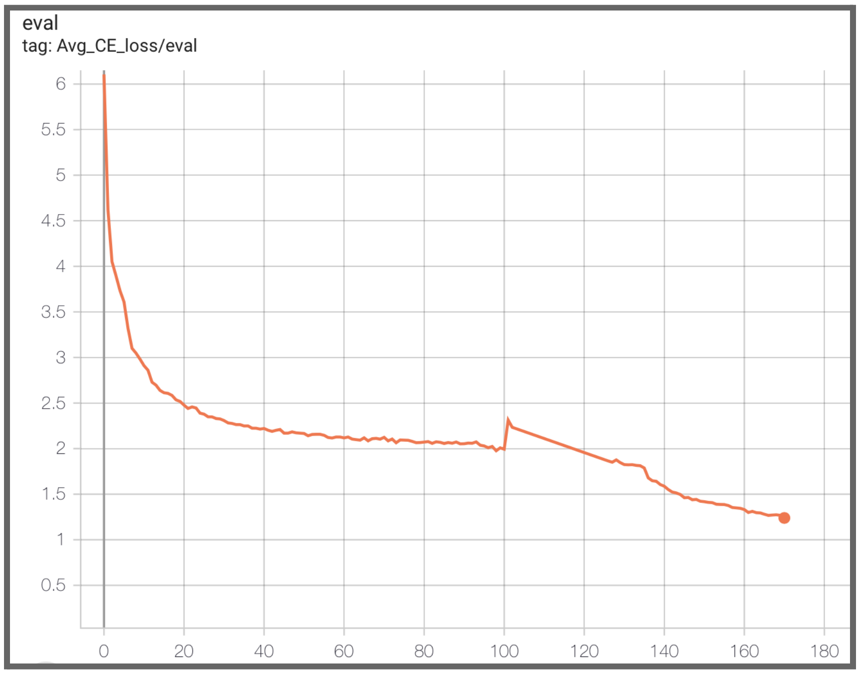

Figure 4. Cross-entropy loss over 170 epochs of training.

Music Generation

I evaluated the model's output by generating music samples in two ways:

1. "improv": randomly initiating the starting sequence of notes and letting the models run.
2. "primed": continuing a set sequence of notes (e.g. the first few seconds of Happy Birthday).

Please see 25+ examples of generated music from my model in my GitHub directory "sample_generated_music."

Musical Analysis
Music theory is out of the scope of this project, so I will refrain from too much in-depth analysis / will not claim results from a few examples. There are, however, a few trends I observed in the musical output from the model:

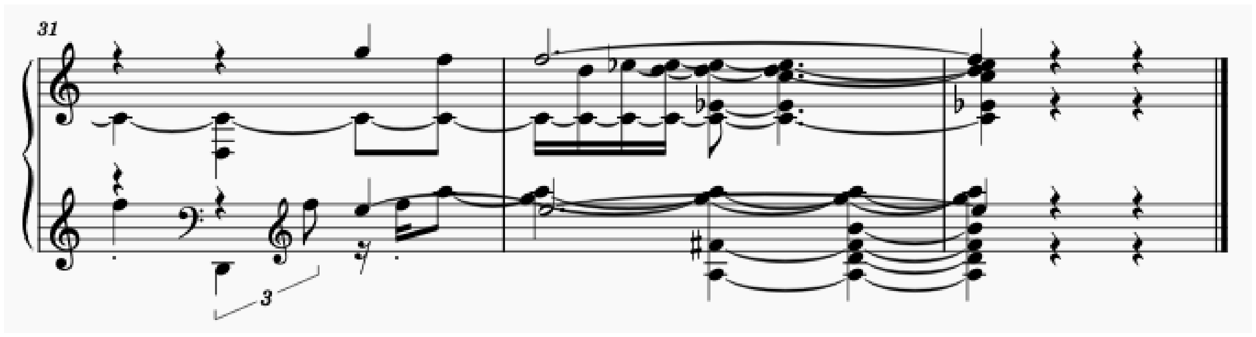

Figure 5. Final measures of "TwinkleTwinkleLittleStar-classical-100epochs-ex2.wav" show long lines with complex melodies (present in both hands).

In general, the **Classical model** exemplified many common attributes of classical music — for example, longer passages with more buildup in harmony. There are also more variations in musical ideas (instead of just repeating the same patterns). There were also several instances of musical trills, which appear far more frequently in classical music than other genres.

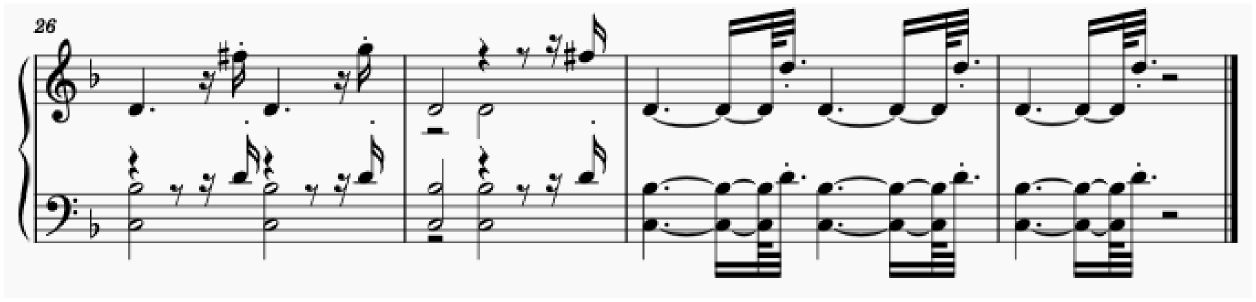

Figure 6. Final measures of "TwinkleTwinkle-jazz-135-epochs-ex2.wav" show repetition of the same "jazzy" chords (rare to find in other genres like classical music).

The **Jazz model** tended to repeat the same musical patterns over and over again. Other than that, there were also prevalence of more scales associated with Jazz tunes (not proper major/minor scales but rather harmonic, diminished, etc.).

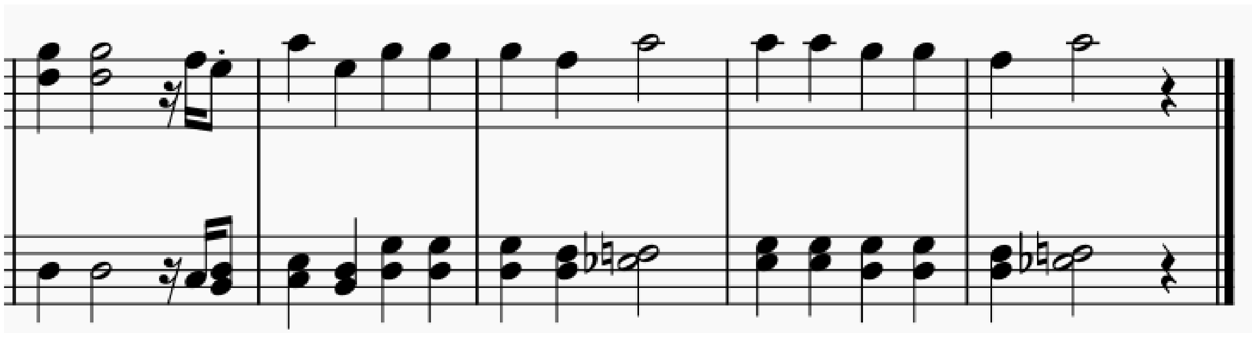

Figure 7. Final measures of “ClairDeLune-pop-170epochs-ex1.wav” show simple chord progressions with a single obvious melody in the upper register.

The **pop model** frequently utilized blocked or rolled chord progressions. Of the three models, it had the least sophistication and also fewer notes.

Conclusions and Future Work

Challenges

I encountered many challenges during this project.

- As mentioned above, I found no working examples of transfer learning on Google’s music transformer models. In fact, most code repositories were outdated by 1-2 years (and utilized versions of TensorFlow or PyTorch that were not supported anymore).
- I had to throw out a lot of data due to differing standards in how music is encoded in the MIDI format and how PyTorch’s Transformer models took in data. Moreover, some of the data needed manual sanity checks (for example, some recordings were filled with mostly rests).

Conclusions

This project presented the only known currently working finetuning of Google Brain’s Music Transformer model to different styles of music. Although the performance is far below Google Brain’s version, the model is able to achieve promising outputs with less than 3 hours of finetuning. Moreover, this project showed that transfer learning on relatively small datasets (less than 1,000 data points compared to tens of thousands) not only reduces training time but also is able to learn certain stylistic attributes of different music genres.

Future Work

This project was limited by computational resources as well as time constraints. It would be interesting for future work to explore the ability to hyper-personalize a model to an individual composer’s work (which would require meticulous data curation and cleaning). Another direction would be to evaluate the efficacy of the model to capture certain musical styles or emotions via a human computer-interaction (HCI) study.

References

Google Brain Music Transformer Paper https://arxiv.org/pdf/1809.04281.pdf

- Forum thread on author discussing finetuning Magenta music transformer (authors claimed to have done this but supplied no working code and weren't able to help researchers trying to implement finetuning) https://groups.google.com/a/tensorflow.org/g/magenta-discuss/c/tRrth7wXF6U

Transfer Learning Transformer Models: https://arxiv.org/pdf/1911.03090.pdf

Datasets

- Computer-Generated Music for Tabletop Role-Playing Games
    - https://arxiv.org/abs/2008.07009
    - https://github.com/lucasnfe/adl-piano-midi
- GiantMIDI-Piano: A large-scale MIDI dataset for classical piano music
    - https://arxiv.org/abs/2010.07061

MIDI visualization

- https://nbviewer.org/github/craffel/pretty-midi/blob/main/Tutorial.ipynb