LSTM based genre-specific music generation

Nimish Magre
Northeastern University, Boston, MA
magre.n@northeastern.edu

Abstract

The project has had a two-fold aim in understanding, implementing and reviewing the state-of-the-art (SOTA) machine learning (ML) models to generate user specified genre of music along with implementing a personal ML model to generate similar music. In terms of the SOTA ML models to generate music, the following sections introduce and explain the **Jukebox model** [4] presented by *HuggingFace* and the **GANSynth model** [5] presented by *Magenta*. Upon reviewing these models, it is understood that the amount of time and finance it would take to procure the training data as well as train these models (over 24 hours with GPU setup) is beyond the available resource and therefore a relatively simpler LSTM-based model has been developed and trained from scratch to generate the music for the required genre. Samples from the Jukebox model [4], the GANSynth model [5], the new LSTM model as well as the code for it are available at this Github Repository

1 The JukeBox Model (OpenAI) [4]

A typical 4-minute raw audio for music with sampling rates in the typical range of 16kHz to 48kHz contains approximately 10 million segments with each segment consisting of 16-bits of information [4]. This makes it computationally expensive to utilize typical generative models to process and work with raw audio samples. Through the use of a novel Vector Quantized Variational AutoEncoder (VQ-VAE) [10, 11] developed to generate high resolution diverse images, Dhariwal et. al [4] have been able to generate lengthy audio samples that contain voiced lyrics with user-specified genres and artists.
For the jukebox model, the authors initiate training with a three-layered hierarchical VQ-VAE [10, 11] model with three separate temporal scales as shown below:

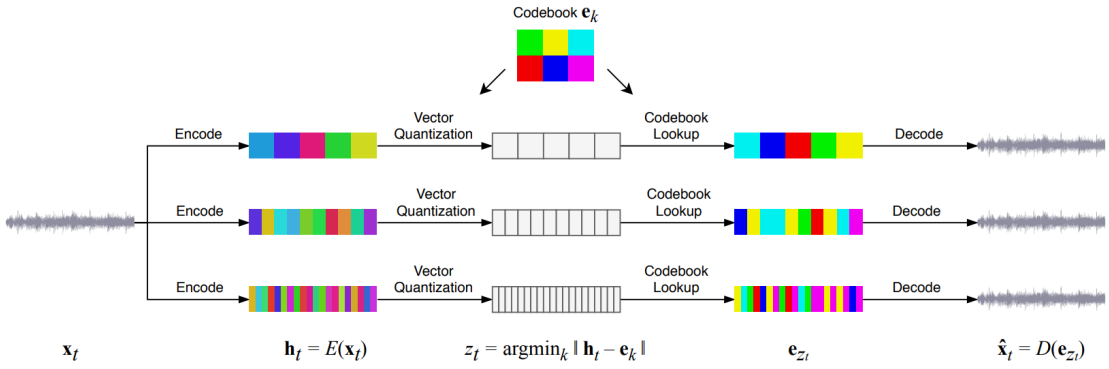

FIG. 1 – *The 3-layer hierarchical VQ-VAE model [4]*

Unlike a regular VAE, the VQ-VAE makes use of **codebooks** (a list of vectors e) to assign the vector closest to the encoded space representation of a sample h. The book or list of vectors for each sample are then fed to the decoder to output the initial sample. For all three layers, along with the encoder and the decoder, the codebook layers are also trained to better resemble sample representations in the encoded space.
Furthermore, to make use of information from all three hierarchies of the VQ-VAE models, the authors then utilize Scalable Transformers [3, 12] to learn a prior over the compressed space so that more varied samples of music can be generated.

$$p(z) = p(z^{top}, z^{middle}, z^{bottom}) \tag{1}$$

$$p(z) = p(z^{top}) p(z^{middle} | z^{top}) p(z^{bottom} | z^{middle}, z^{top}) \tag{2}$$

Eq:1 represents the combined prior with all three hierarchical layers which can be split into three separately trainable convolutional models as in *eq:2*. Finally, the authors provide conditioning information such as the artist, lyrics, genre and timing signals to upsample the codebook information from the learnt prior $p(z)$ through *eq:2* to the bottom layer of the VQ-VAE architecture which is then passed to the decoder of this layer to generate the final new music sample.

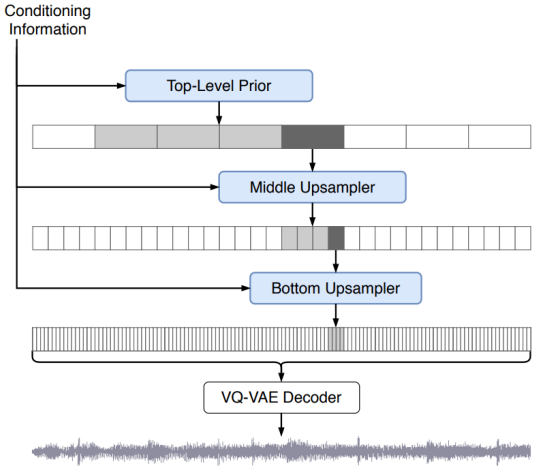

FIG. 2 – *Upsampling the 3 codebook layers from top level to coarse level along with lyrics, genre and timing section as additional conditioning information [4]*

The authors make use of 1.2 million webscraped songs along with metadata lyrics obtained through LyricWiki [1]. Overall, the VQ-VAE model involves 2 million variables and was trained for three days on a 256 Nvidia V-100 graphics card.

2 GANSynth Model (Magenta by Google-AI) [5]

This generative model improves on its predecessing auto regressive model (WaveNet[7]) through the introduction of a progressively growing GAN model[8] to generate the entire audio sample in parallel instead of sequentially generating the samples over long stretches of time. Importantly, the paper highlights the disadvantage of upsampling convolutions to align with the phase of highly periodic signals (typically found in music related audio samples). Instead, the authors focus on generating the Instantaneous Angular Frequency (derivative over time for the phase over the 2π boundary)[5] which remains constant over time and represents the coherent periodicity of the sound sample. Much like in the case of the CelebA[9] dataset that is cropped and centered during preprocessing for face generation through GANs, this model uses the NSynth dataset[6] which includes separate notes from different instruments across a range of pitches, timbres and volumes. Furthermore, while training the authors use the progressively growing GAN training technique[8] which involves progressively adding upsampling layers to both the generator and the discriminator to allow for faster training times.

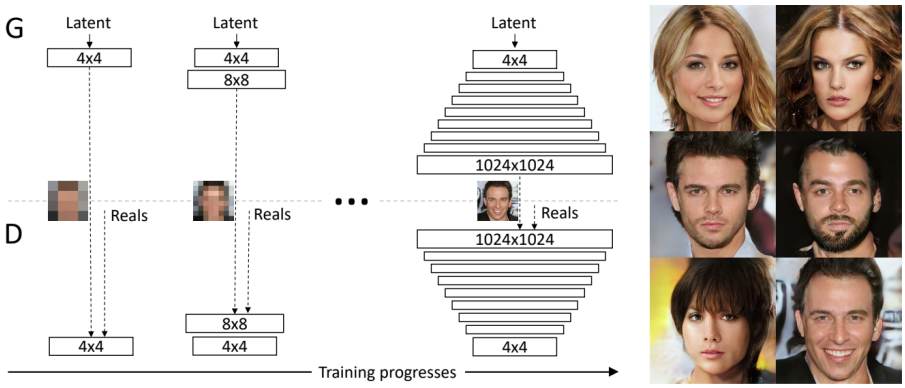

FIG. 3 – *The progessive GAN training method [8] that involves progressively adding upsampling layers to both the Generator and the Discriminator. The successfully CelebA [9] style faces generated on the right-side suggest the advantage of the this training style*

While this GANSynth model improves in terms of performance over its autoregressive sequential music generating predecessors, it certainly lacks the advantages of working on the Jukebox model [4] explained earlier due to its lack of ability in processing raw audio and synthesizing lyric vocation.

3 LSTM-based model

As seen with both the VAE and GAN based SOTA networks from the previous sections, the computative as well as the financial cost incurred to acquire datasets with millions of songs and to train them with multiple GPU based models for extended periods of time is beyond the scope of the project and therefore an LSTM based Recurrent Neural Network has been developed. It also has the advantage of being able to handle sequential data that can be found in music samples.
For the purpose of this project, all music files used for training were presented to the model in a *midi* file format [2]. These *midi* representations are compact compared to raw audio and carry music event messages specified by notes, chords, octaves and offsets. They consist of the music data being split into *chords* and *notes* as shown below for a guitar sample:

```
<music21.note.Note G> 72.0
<music21.chord.Chord E4 C5> 72.0
<music21.note.Note C> 72.0
<music21.note.Note G> 72.0
<music21.chord.Chord E4 C5> 72.0
<music21.note.Note C> 72.0
<music21.note.Note G> 73.5
<music21.note.Note G> 221/3
<music21.note.Note C> 73.75
<music21.note.Note C> 73.75
<music21.note.Note F> 74.0
<music21.chord.Chord D4 B-4> 74.0
<music21.note.Note B-> 74.0
<music21.note.Note F> 74.0
<music21.chord.Chord D4 B-4> 74.0
<music21.note.Note B-> 74.0
<music21.note.Note F> 75.5
<music21.note.Note F> 227/3
<music21.note.Note B-> 75.75
<music21.note.Note B-> 75.75
<music21.note.Note F> 76.0
<music21.chord.Chord C4 A4> 76.0
<music21.note.Note F> 76.0
<music21.note.Note F> 76.0
```

FIG. 4 – *Music21 library based reading of a guitar midi file with notes, chords and numeric offset values*

Each note object contains information about the Pitch, Octave and the Offset values. For the image above, the Pitch is represented by letters from A-G with A signifying a high frequency sound and B signifying a low frequency sound; The Octave signifies the sets of pitches used whereas the Offset signifies the gap between notes and chords. Chord objects contain sets of Notes that are played at a particular time.
In order to supply music data to the LSTM model, each music sample is read as a midi file and the obtained notes/chords are stored in an array. Once all notes/chords have been stored as sequential string arrays for all training music samples, they are then converted to numerical values through a mapping function which maps categorical data to numeric data. Based on the mapping function, each input sequence (chosen to be 100 in this case) is normalized and the network ground-truth output is chosen to be the 101^{st} note (one-hot encoded) and so on.
The model itself consists of some LSTM (inbuilt with Keras API), dropout, activation as well as dense layers as shown below:

```
model = Sequential()
model.add(LSTM(
    512,
    input_shape=(network_input.shape[1], network_input.shape[2]),
    recurrent_dropout=0.3,
    return_sequences=True
))
model.add(LSTM(512, return_sequences=True, recurrent_dropout=0.3,))
model.add(LSTM(512))
model.add(BatchNorm())
model.add(Dropout(0.3))
model.add(Dense(256))
model.add(Activation('relu'))
model.add(BatchNorm())
model.add(Dropout(0.3))
model.add(Dense(n_vocab))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='Adam')
```

FIG. 5 – *the proposed LSTM based model to train with 100 input sequence samples for predicting a single 1 hot encoded output sample*

The network was then trained on a set of 30 hip-hop based guitar midi files scraped from the *FreeMidi* [1] public domain. With the *Music21* [2] library, the training data is processed as single instrument midi files belonging to the genre required by the user. For the given guitar midi dataset, the model seemed to require approximately 200 epochs to complete training with a batch size of 64. The total training time with a Tesla P-100 GPU setup was approximately 4 hours.

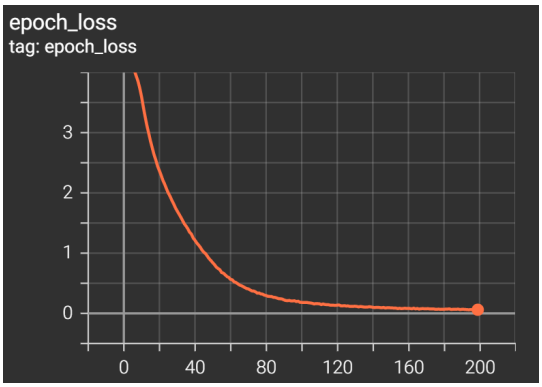

FIG. 6 – *the tensorboard epoch loss logs output for the proposed LSTM based model with the y-axis representing epoch loss (categorical crossentropy) and the x-axis representing the number of epochs. the logs are available for regeneration on the github repository*

To finally test the model, a random sequence of 100 samples either uploaded by the user or from the training set is used to generate 500 output samples with random offset values between 0.1-0.9 to generate a 2 minute output sample. All generated samples, along with experiments carried out with varying input sequence lengths as well as offset values and instruments are available at Github Repository

4 Conclusion

While the goal of the project has been to explore the best AI models for music generation, it has been difficult to individually train and experiment with these models [4, 5] due to the vast computational resources that would be required to do so. The hierarchical VQ-VAE [4] based model certainly provides the most user-options in terms of deciding on music genre and artist for lyric vocation. It is also capable of handling raw audio data while training this sequential data parallely and providing results better than the GAN-based model [5].
It would be interesting to explore the concepts of using a hierarchical VQ-VAE [10] model in the video synthesis and lyric generation domains due to its capability of parallely handling sequential data. Through experimenting with an LSTM-based model to generate genre-specific music, a user can successfully generate valid and unique music samples with a few hours of GPU resources. While the *Music21* [2] library used with the LSTM model sufficiently handles single instrument notes, future work would be focused on utilizing the same library with parallel models for separate instruments along with an additional lyric vocation model.

1. https://www.midis101.com/search/guitar
2. https://web.mit.edu/music21/

References

[1] Lyricwiki, Aug 2022.

[2] Midi, Aug 2022.

[3] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[4] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[5] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.

[6] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, pages 1068–1077. PMLR, 2017.

[7] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

[8] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[9] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018.

[10] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-resolution images with vq-vae. 2019.

[11] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.