EECS 545 Project Progress Report: Tempo-Lite: Drum Beats Generated from Other Instruments

Congni Shi, Li Ye, Muye Jia, Jerry Cheng
{congni, yeliqd ,muyej, chengjry}@umich.edu

March 10, 2022

Abstract

Comparing to Vision, Audio generation has always been a field that is less studied with deep learning. In ths course project, a lightened version of a temporal music generator MuseGAN [**?**], Tempo-Lite, is proposed. The Tempo-Lite model takes the existing tracks of musical instruments other than drum and generate complementary drum tracks for the existing tracks. As a milestone of this objective, the group has also provided classification methods and model for the Pianoroll representation of music signals. Although the project is still in progress, the classification models has shown some promising results that can be used in following stages. The group has also finished the Generator and will soon be tested once the Discriminator is finished.

1 Introduction

Machine learning has been employed extensively in the field of computer vision and image processing, however,it has not been extensively investigated in the music field. For this project, the group aims to use various machine learning models to do music genre classification and music composition. More particularly, the group intends to produce a drum track generator model, which takes the the rest of the song (song tracks without the drum track) and the predicted genre as inputs, then outputs a drum track sequentially based on those inputs.

Compared to copious deep learning researches in vision, acoustic generative networks are less visited. The group believes that it would be a lucrative field since amateur composers may not be able to compose for all instruments, which means most of them are more familiar with some instruments than the others. In that case, our model, with the ability to generate a music track (drum track) based on tracks from other instruments, can provide important insights for those amateur composers with their music compositions. Moreover, since the aim of the group is to provide a computationally-light training model, it can be easily re-trained towards different musical instruments, even novel ones such as theremin or Kazoo with contemporary styles.

2 Proposed Method

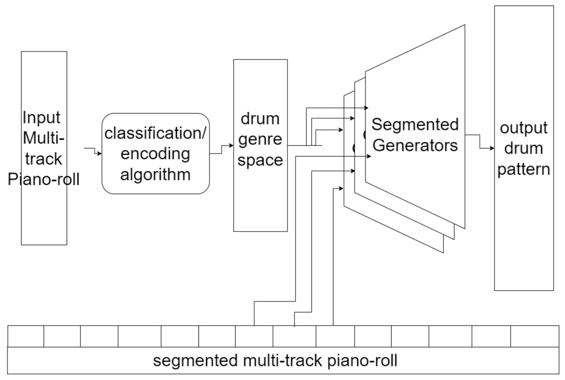
Input Multi-track Piano-roll
classification/ encoding algorithm
drum genre space
Segmented Generators
output drum pattern
segmented multi-track piano-roll

Figure 1: General Pipeline the model

Figure 1 shows the general pipeline of the proposed model. The model can be split into 2 parts: classification and generation. The output (before the last layer) of the classifier is fed into the generator as global induction. After that, the generator and the discriminator work together as a Generative Adversarial Network (GAN) to produce the final output.

2.1 Classification for Global Induction

2.1.1 VGG-based classifier

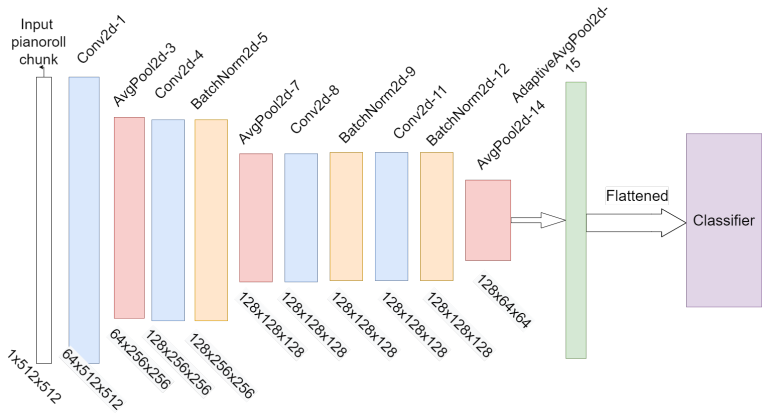

Figure 2: The feature extraction architecture of the CNN classifier

The first classification method the group proposed is a convolutional neural network (CNN) based classifier. The CNN classifier is based on the VGG-16 architecture, featuring convolution layers with small 3×3 filters. However to adapt the binary nature of the pianoroll, the team decided to replace the max pooling layers into average pooling layers to introduce more structural properties of the data as induction for the model. A more specific architecture will be discussed in the next several paragraphs.

Figure 2 shows the feature extraction architecture of the CNN classifier. The feature extraction is done by convolutional layers stride of 2 and padding of 1. We decided not to pad the input too much to maintain the structure of the binary data. The average pooling layers all have a kernel size of 2 and stride of 2 with no padding. The non-linear layer used in the feature extractor was ReLU.

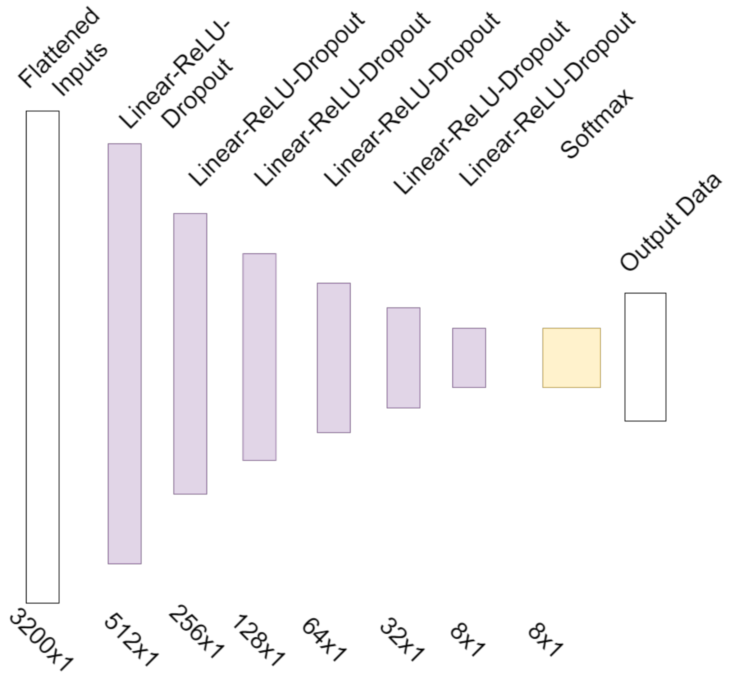

Figure 3: Linear Classifier of the CNN classifier

The output of the convolutional layers are pooled into a single vector. The output of the pooling layer is fed into a linear classifier, which is shown in Figure **??**. Dropout layers in the linear classifier shares a probability of 0.3. Since we believe that can be used to prevent overfitting due to the small training set. In the end, the output of the linear-ReLU-Dropout layers was fed into a softmax layer for classification.

2.1.2 SVD classifier

2.1.3 Perceiver Classifier

Although not implemented yet, the group would also like to try the classification task with the Perceiver [**?**] , which is a sparse attention transformer model with general perception. This classifier would be adapted from a group member's EECS-542, Advance Computer Vision course project, which is expected to be done in the next few weeks.

2.2 Generator

The team proposes to use Generator Network to generate pieces of music. Each input of the Generator has multiple sub-components, and each sub-component of the Generator has three channels-first channel is a randomly extracted song segment, the second channel contains the genre vector of the song, and lastly the third channel contains the positional encoding of the notes in the song segment. All the song segments in each individual input are extracted from the same song, meaning they share the same genre vector, which, as the name indicates, tells the Generator the genre of the song that's being fed in; in order to preserve a richer spatial connectivity, the team decides to add positional encoding for each time-step.

Adding positional encoding enriches the information fed into the Generator. It enables the Generator to take into account the spatial connectivity of the beats. Similar to processing a string of words, the ordering of the words is probably the most important information in determining the meaning of the sentence-so is the ordering of the notes in a song-the team is hoping that the positional encoding can provide this ordering information to the Generator. The advantage of this adopted positional encoding lies in: 1) its ability to provide unique coding for the note at each time-step. 2) its ability to generalize to input of arbitrary length. 3) the encoding does not explode as the input length increases.

2.3 Discriminator

3 Related Work

4 Experimental Results

4.1 Classification

4.1.1 CNN Classifier Results

The VGG-based CNN classifier was able to provide some sensible results for the classification task after training of 50 epochs. The network was able to give around 76% of accuracy upon the testing set and about 81% on the training set on average of 3 to 4 training attempt. The model was trained on the LPD-5 cleansed dataset on an Nvidia GTX 1070 first, with parallel training with different hyperparameters on a different machine with an Nvidia Tesla T4. Training of 50 epochs usually takes about 7 hours on either GPU with no significant difference. However, the Tesla T4 was able to train the model with a larger batch size with the additional graphical memories, which usually leads to a slightly better training results.

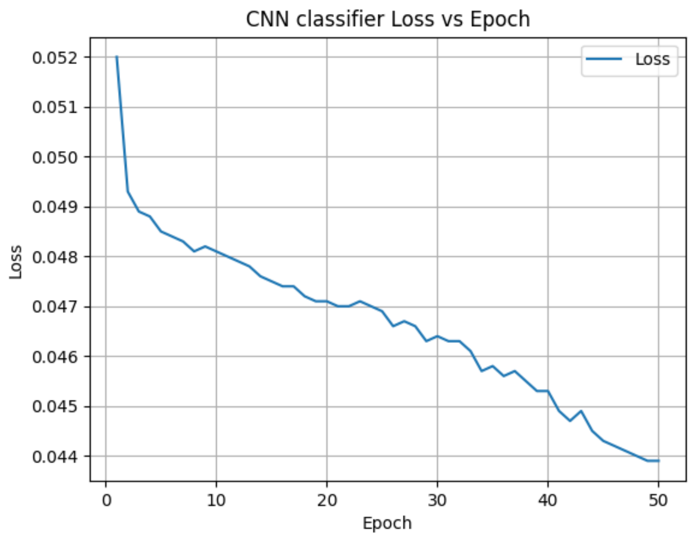

Figure 4: Loss Histogram of the CNN training

Figure **??** shows the loss histogram of training the classifier model. According to the plot, the loss of the training is not plateaued at the ending training epoch, signifying that the model is still learning, yet at the end of 50 epochs, the model was able to give an output of around 80% accuracy. Thus the group believe that with longer training time and better hardware accelerator, the model will be able to achieve better results. The group would try to investigate on further training with the next half of the project.

4.1.2 SVD Classifier

In order to retain an acceptable time-efficiency for the singular value decomposition required in the team's Sub-space Learning (SL) module, the team needs to shrink down the size of the song tracks. One challenge for this task is that the compression must make sure the prominent features of the song data stay distinguishable so an effective classification is possible. With this in mind, the team decides to adopt a feature extraction/representation method called Mel-Frequency Cepstral Coefficient (MFCC) in order to take out the most prominent patterns of the songs.

MFCC is originally used to represent the power-spectrum of an acoustic signal. Since the coefficients of MFC are able to represent the frequency features (after applying Fourier transform) of the signal, this quantity is widely used in audio compression. Although the MFCC is originally designed to transform a time-amplitude signal, and that our data set are binary values, with pitch instead of amplitude of the sound recorded at each time-step, the team can still take advantage of its ability to describe the noticeable shape and patterns of a signal-the song tracks can be seen as time-amplitude signal where the amplitude equals, in numeric value, to the pitch at each time-step.

4.2 Generator

5 Future Milestone

6 Conclusion

Author Contributions

For this course project, Jerry Cheng has proposed the network architecture and their respective ablations, collected the data and constructed the customized dataloader. Jerry also finished the VGG-based CNN classifier. Congni Shi finished the naive SVD classifier, and incorporated the MFCC features once it was finished by Muye Jia. Muye wrote and tested the Generator and the MFCC feature extractor for the SVD classifier. Jerry oversees the cloud computing platforms and is in charge of the deployment of all deep learning models. All member contributed equally on this report.