Music Style Transfer with Pre-trained Convolutional Neural Networks

Jiayi Wang, *MechE, Carnegie Mellon University,* Tao Chu, *MechE, Carnegie Mellon University,*

***Abstract*—Remixing music can bring a new life to old and cliche songs, however, existing song remixing methods are often highly professional, time consuming, and costly, and for laypeople who want to produce their own remixes, it is not a good sign. In this paper, we are proposing a method to have artificial intelligence help us remix music we want in a faster and more convenient fashion. The method we are using is called Style Transfer, first introduced by Gatys et al. [1], who accomplished recomposing one image in the style of another image, hence combined both images together using convolutional neural network. Now, taking advantage of this technique, we are implementing a similar method, yet we are not transferring styles of images but music. In order to apply style transfer, we have to convert our audio data into images, and spectrograms turn out to be a favorable choice, for it carries the audio information we need and at the same time, can be recognized by our convolutional neural network. After having our audio images ready, next is to build a model which suits our need closely, and we found pre-trained convolutional neural network not only saves time for training but recognizes the features we want our model to recognize already. Therefore, we selected VGG16 [2], a 16-layer convolutional neural network trained to classify images of daily objects, and VGGish [3], which is trained on millions of audio spectrograms to classify different sounds, to be the model candidates. Using the same algorithm as image style transfer, our results indicate that although we are able to mix two songs together, we have to deal with noises generated during the process of modifying our spectrograms, yet we still provide new insights into implementing style transfer in domains other than images.**

***Index Terms*—Deep Learning, CNN, VGG, VGGish, Style Transfer, Audio Analysis**

I. Introduction

Music remixing is a way to mix two music together harmonically and ends up with a song with a brand new feeling different than the original music, and can suit variable atmospheres in different occasions.

However, music remixing has long been considered a very professional and tedious task, and for average Joe who only wants to try how two music will sound when they are combined, it makes no sense to study about music from scratch, let alone going further into remixing. Besides, even though machine learning implementation in music have been around for a few years, when it comes to remixing music with machine learning, few attempts have been done. Given that using machine learning to approach music remixing is absolutely novel and unprecedented, there is no state-of-the-art in what we are trying to address. In this paper, we are proposing to remix music in the sense of transferring the style of one music to the style of another using deep convolutional neural networks, which is much faster, less knowledge-demanding, and more laypeople-friendly.

In order to achieve audio style transfer, we implement a method called Style Transfer [1]. Style transfer is a relatively mature technique thanks to the thriving development of convolutional neural network, and it is useful in transferring style of an image to style of different images. In the original paper, the author proposed a method to extract both content information of one image, called content image, and style information of another image, called style image, from output of different layers in convolutional neural network and eventually combine them on a random noise image to generate a final image that keeps the content of the content image and the style of the style image, shown as Fig 1.

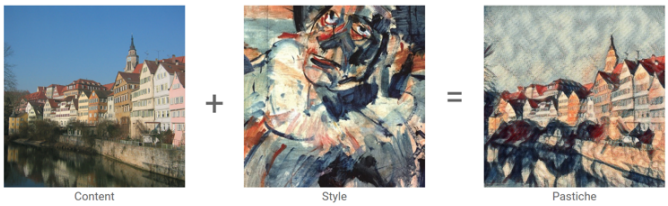

Fig. 1: From Google Codelabs. The texture of the content image is transferred to the texture of the style image after style transferring.

Besides, since convolutional neural networks work best on images, we turn our audio files into spectrograms, which carries audio information in a photographic way. After converting audio into spectrograms, we implement the same algorithm Gatys et al. [1] has proposed to minimize the difference between the content information of the content spectrogram and our input random noise image, at the same time minimize the difference between the style information of the style spectrogram and our input random noise image. These are to ensure our final output, which was originally a random noise image, matches the content of the content image and the style of the style image as closely as possible. Finally, we reconstruct the audio from the output spectrogram using the algorithm proposed by Griffin et al. [4].

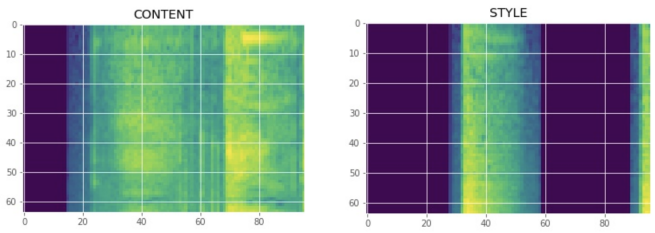

Fig. 2: Spectrograms of content audio and style audio. **X axis** represents time, **Y axis** represents mel-frequency, which is a mapping from normal Hertz frequency and approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum, and **brightness of color** carries amplitude information, the brighter the color, the higher the amplitude.

Our main contributions include:

- Provided insights into VGGish usage in style transfer.
- Approached audio synthesis using deep learning.
- Opened a new channel for style transfer implementation.

II. Related Works

A. Image Style Transfer Using Convolutional Neural Networks

Gatys et al. [1] utilized pre-trained VGG network to extract content information and style information of images and by minimizing differences between our content, style, and output images, it turned out the algorithm was able to generate an output that stores content of one image while carries the style of the other image. The authors found that higher layer in the network captures high level content in terms of objects and their arrangement in the input image but not constrained by the exact pixel values, while lower layer simply duplicates the exact pixels, so we will use higher layer to represent our content information. For style presentation, the authors used a feature space designed to capture texture information. It consists of the correlation between different filter responses, and by including the feature correlations of multiple layers, we can obtain a stationary, multi-scale representation of the input image, which captures its texture information but not the global arrangement.

This paper pioneered the development of style transfer with convolutional neural networks, and had very promising results. However, in this paper, we are dealing with audio spectrograms, and we use VGGish as our pre-trianed network, which could be more representative in audio domain, compared to the VGG the authors deployed in their paper.

B. Audio spectrogram representations for processing with Convolutional Neural Networks

Lonce Wyse [5] proposed a different way to deal with audio style transfer, first the author mentioned since audio spectrograms dont store audio information the same way as normal pictures store image content and also spectrograms have different meanings in X and Y axis, spectrograms should not be directly passed into convolutional neural networks. Instead, the author converted the original 2D spectrograms into a 1D vector by stacking up all frequency bins in the same time bin, and passed the 1D vector into the network to run the style transfer algorithm.

However, after Google released their AudioSet [**45857**], which consists of an expanding ontology of 632 audio event classes and a collection of 2,084,320 human-labeled 10-second sound clips drawn from YouTube videos, and a deep convolutional neural network that was trained on millions of audio spectrograms called VGGish [3], which achieved over 90 percent accuracy in classifying audio data, we are sure that we can pass our spectrograms straight into the network and also in this paper we use VGGish as one of our feature extractors, which suits more closely to our need than VGG-19, the one the author used in his paper.

C. Existing Machine Learning Algorithms in Music Processing

In terms of multi-layer music processing, the European Research Council has founded a research project called Flow Machine, which has created the first artificial intelligence song, "Daddy's Car".[6] The project aims at transforming music styles into computation objects so they can be applied on computer generated melodies and harmonies. It focuses more on music generation rather than style transfer. However, after generating a multi-layer song, it still needs professional musicians to refine it. Ji-Sung Kim[7] from Princeton University developed Deep Jazz by using Keras and Theano to generate jazz music. The output is extraordinary but can only process music involving only one kind of music instrument.

III. Technical Solution

In this project, we are trying to do music style transfer on popular songs which include vocalist and multiple instruments. These songs contains several music layers and huge amount of information and are difficult to transfer to MIDI files as people do with single layer music. It is particularly hard and time consuming to train a model from scratch for these songs. The way we solve this problem is similar to that in image style transfer by Gatys et al.[1] We utilized a pre-trained VGG network as feature extractor to extract the information contained in the audio files. Since the network is already trained to recognize information in images, much time can be saved.

The features in this project are the content feature and style feature. For content feature, it describes the content of the song, which can be the lyrics and the singing of the vocalist. As for style feature, it includes beats, instruments and the rhythm of the song.

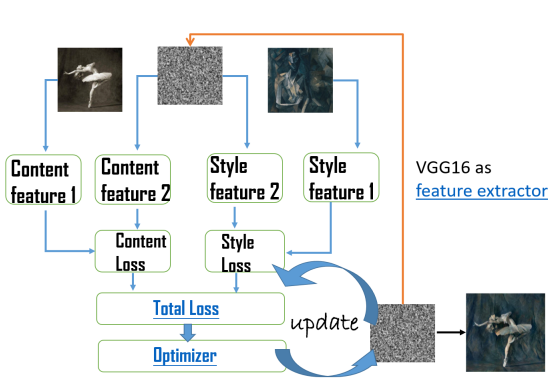

Fig. 3: Model Structure

A. Model Structure

Figure 3 shows the model structure in our music style transfer project. It can be divided into four parts which are data pre-processing, feature extraction, optimizing and data post processing. The inputs of the model are three spectrograms, the one with target style is known as style spectrogram, the one we want to transfer the style onto is called content spectrogram and the target spectrogram we are optimizing during each iteration. The taget spectrogram is initialized to random value at the beginning. The output of the model is that same target spectrogram but after several iterations which contains the information of both content audio and style audio.

In our proposed method, VGGish and VGG16 network only serve as feature extractor which helps the machine understand the content in the image. The feature extractor gives out the output of the intermediate layers of the network. The reason we are using intermediate layer to compute the content and style feature is that for a network to get the label of the image and do classification, it has to learn about what the image is about. That is why we are trying to use different network structures and use different kernels to compute next layer to get a higher accuracy. To make it easier to understand, if we pass two images of cats through an image classification network, even if the initial images look very different, after being passed through many internal layers, their representations will be very close in raw value, and at last it will be the same binary pattern distribution which can be assigned to the same class. An illustration of the intermediate layer output of VGG-F is shown in Figure 4 to help readers understand how feature extractor work.

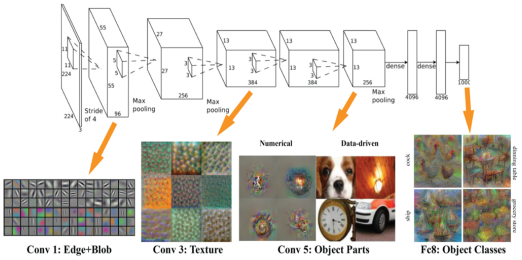

Fig. 4: AlexNet/VGG-F network visualized bymNeuron

To start the optimization, we initialize our target spectrogram to random noise. Then pass it along with the content and style spectrogram through the layers in VGGish or VGG16 network. We use the outputs of various intermediate layers to compute content loss and style loss, then compute total loss. Total loss is minimized by changing the pixels in target spectrogram. After repeating the process and optimizing for several iterations, a stylized version of the original content image will be generated.

B. Similarities and Differences between VGG16 and VGGish

In the following sections, we are going to discuss the style transfer process with VGG16 and VGGish. Both of them are pre-trained deep convolutional neural network architectures for classification. VGG16 is trained on images of daily objects belonging to 1000 classes while VGGish is trained on audio spectrograms belonging to 128 classes.

C. Style Transfer with VGG16

Convolutional Neural Network (CNN) is widely used in large-scale image processing. To obtain a better accuracy, people tend to apply different CNN architectures with various type or order of layers. VGG is one of the most typical and popular one among these architectures in terms of image classification. It usually refers to a deep convolutional network for object recognition developed and trained by Oxford's renowned Visual Geometry Group (VGG), which achieved very good performance on the ImageNet dataset [8]. a daily image dataset contains over 14 million samples belonging to 1000 classes.

D. VGG16 Feature Extraction

To obtain these features, we are using a pre-trained VGG16 [2] network, which is known as a very deep convolutional neural network for Large-Scale Image Recognition. Its structure is shown in Figure 5. VGG16 is a classification network which has a 92.7 percent test accuracy on ImageNet[8].

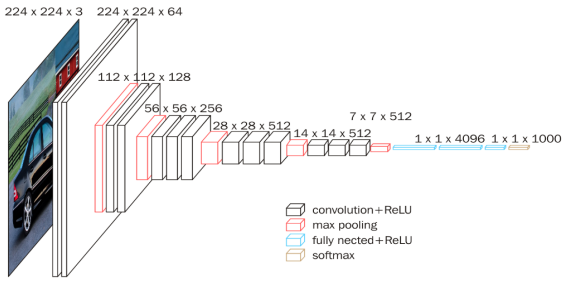

Fig. 5: Architecture of VGG-16 convolutional neural network

Figure 6 shows the layer structure of VGG16 network. It has totally 13 convolutional layers and three fully connected layers. In terms of selecting best intermediate layer to extract features, we tried different combination of the output of 13 convolutional layers in VGG16 network. We evaluated the goodness of the intermediate layer by comparing the loss after 10 iterations. We selected the combination with least content

loss as content layer, and the output of content layer will be used to extract content feature. Similarly, we choose the combination with least style loss as style layer, and the output of style layer will be used to extract style feature. We can also track if the feature extractor is effective in the same way. It turns out that the fifth layer is the best as content layer and the combination of all the 13 convolutional layers are the best to serve as style layer. If taking computational cost into consideration, the combination of 1st-4th convolutional layer is a secondary choice as style layer. Since they obtained almost the same style loss comparing to our best combination and the layer output we have to extract decreased from 13 to 4 during each loop.

Figure 3. Layer structure of VGG-16 network

Fig. 6: Layer structure of VGG16 network

In terms of data normalization, we used L2 normalization to keep the data under the same scale. This is necessary because different songs have various amplitude or loudness, it is better to put them into a same scale before processing. Or one of them will take dominant during the training.

E. VGG16 Data Preprocessing and Post Processing

Since the input of VGG16 is supposed to be a colorful image with R, G, B color channel, our spectrogram is actually only a black and white image with only one color channel. To fix this problem, we assign same value to the R, G, B channel. When the feature got extracted and our target spectrogram forms, it is still a colorful image. Which means we need to convert the colorful image back to black and white one thus we will be able to transfer it back to audio file. To convert the colorful image to greyscale image, we applied the formula Gray = (Red * 0.3 + Green * 0.59 + Blue * 0.11)[9] to get the most clear features from the spectrogram.

F. Style Transfer with VGGish

VGGish [3] is a variant of the VGG model with 11 weight layers trained on the YouTube-8M Dataset, all input audio data is put through the following preprocessing steps:

- Resampled to 16kHz mono, from the original 44.1kHz.
- Spectrogram is computed using magnitudes of the Short-Time Fourier Transform with a window size of 25ms, a window hop of 10ms, and a periodic Hann window.
- Map the spectrogram to 64 mel bins covering 125 Hz to 7500 Hz to compute for mel-spectrogram.
- Apply log(mel-spectrogram+0.01) to obtain a stabilized mel-spectrogram, where adding 0.01 is to avoid taking logarithm of zero.
- Features are then framed into non-overlapping examples of 0.96 seconds, where each example covers 64 mel bands and 96 frames of 10ms each.

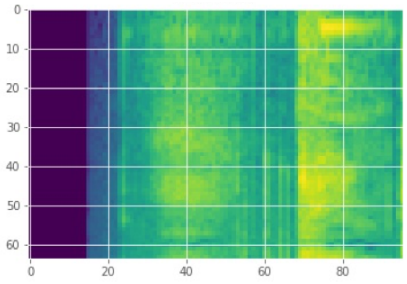

Fig. 7: After preprocessing, a 64 by 96 mel-spectrogram is computed to be input to the VGGish network.

In this part, we selected two sets of content layers and style layers to see the difference in results between different layer choices:

- Content Layer: 5th RELU, Style Layer: 1st RELU
- Content Layer: 6th RELU, Style Layer: 2nd RELU

Also, for each content-style layer combination, we exchange the weight of style loss and content loss to change our bias between the two.

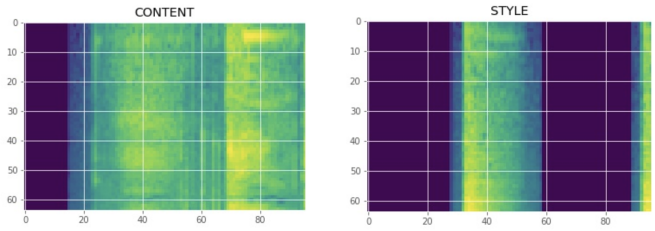

Fig. 8: Content spectrogram and style spectrogram input to the network.

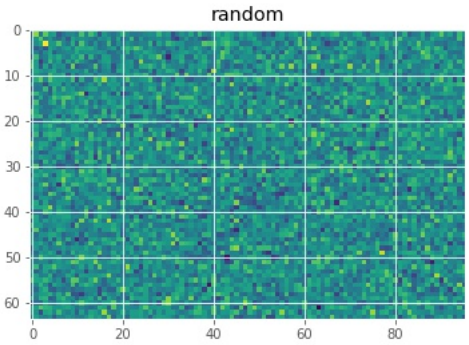

Fig. 9: Random noise image, on which our transferred result will appear.

Below is the style transferring process of the four combinations of different layers and weights.

- Content Layer: 5th RELU, Style Layer: 1st RELU

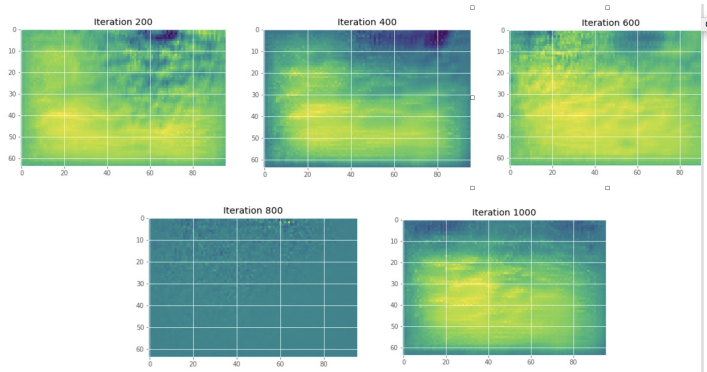

Fig. 10: Content Weight:10, Style Weight:1.5

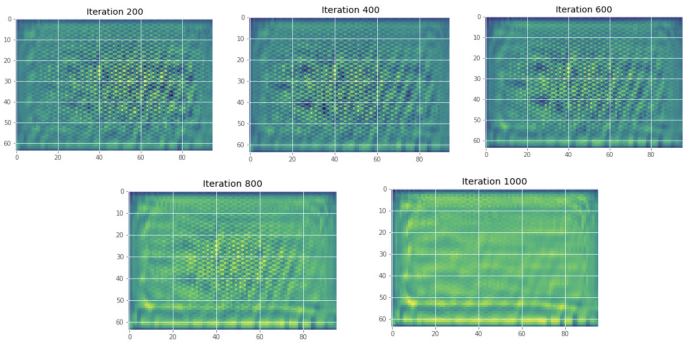

Fig. 11: Content Weight:1.5, Style Weight:10

Apparently, when content weight is much higher than style weight, the remixed spectrogram soon turns bright, due to high brighter color coverage in the content spectrogram, while when the weights are exchanged, there are certain patterns in the middle of the spectrogram that the remixed spectrogram is trying to capture and keep, and only gets covered by a large yellow area after almost 1000 iterations

- Content Layer: 6th RELU, Style Layer: 2nd RELU

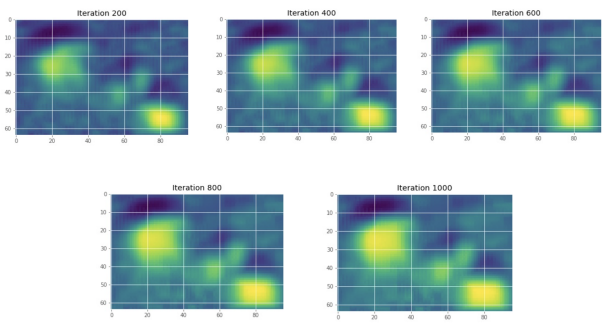

Fig. 12: Content Weight:10, Style Weight:1.5

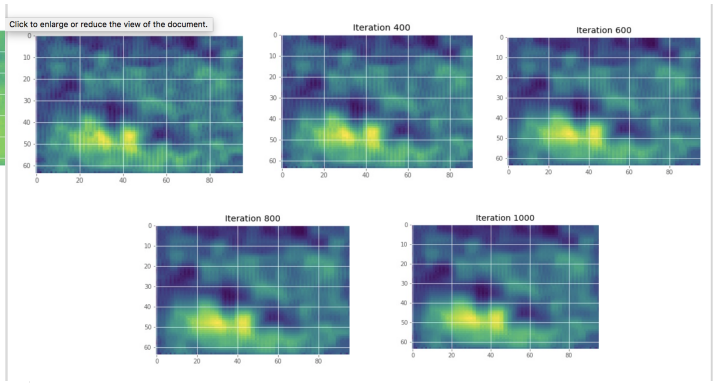

Fig. 13: Content Weight:1.5, Style Weight:10

After changing the content and style layers, we found that the transferring process is not as obvious as previous layer selections; however, for both processes, the changes from the original random noise image to the resulted spectrograms are still visible, and by swapping the content weight and style weight, we can still see the difference between higher content weight versus lower content weight, as per the style weight.

IV. Results

A. Results of Using VGG16 as Feature Extractor

Although VGG16 network is trained on daily object images instead of spectrogram, the output preserves the content of content audio and is also finely styled.

It can be easily told that the output audio is a styled version of the content audio. The reason VGG16 is working is that spectrogram also has the element of lines, edges, textures which VGG16 is trying to capture from regular images. The output is obtained from the target spectrogram, we optimize it during each iteration. Figure 12 shows the content loss within 500 iterations while Figure 13 shows the style loss within 500 iterations.

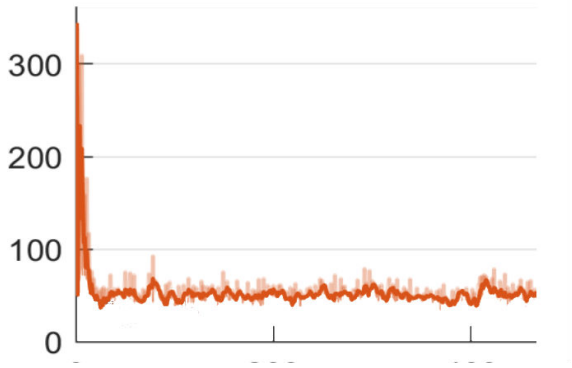

Fig. 14: Content loss in 500 iterations

Figure 14 and 15 shows both content loss and style loss keep the same after about 60 iterations, so in the following training process, we just optimize the target spectrogram for 60 iterations to save computational cost. Figure 16 to 19 show the optimizing process of our target spectrogram in 60 iterations. The left image in each figure is the content spectrogram,

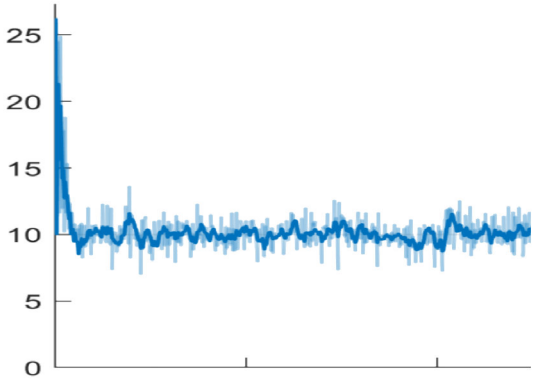

Fig. 15: Style loss in 500 iterations

the middle one is the target spectrogram which begins with random value and the left one is style spectrogram. Patterns are gradually generated in the image as more iterations are being optimized. Initially there were some background noises in our output audio file, we refined it by adding a low-pass filter to filter out the high frequencies which should not appear in a song. A mid-frequency enhancer is also added to increase the volume of vocalist. Finally we got the ideal styled version of the content audio file.

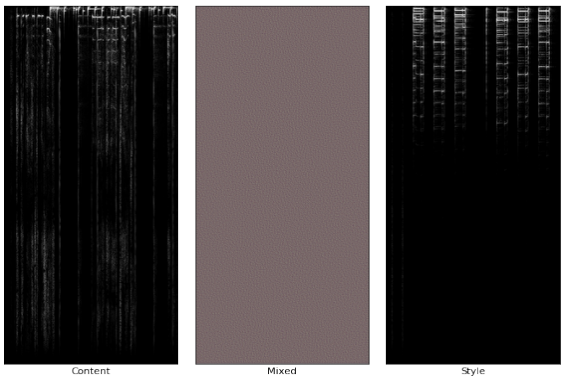

Fig. 16: Target spectrogram at 0 optimize iteration

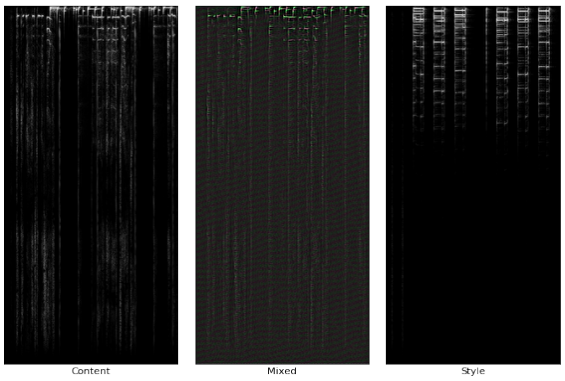

Fig. 17: Target spectrogram at 30 optimize iteration

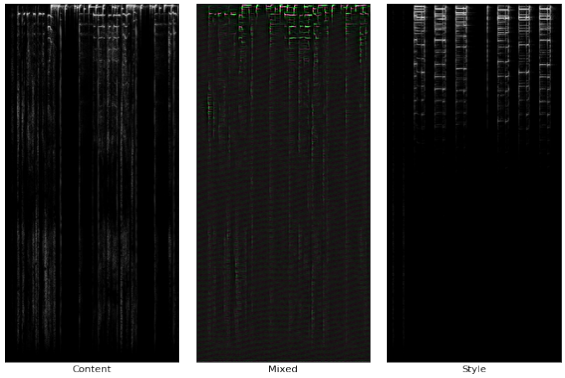

Fig. 18: Target spectrogram at 45 optimize iteration

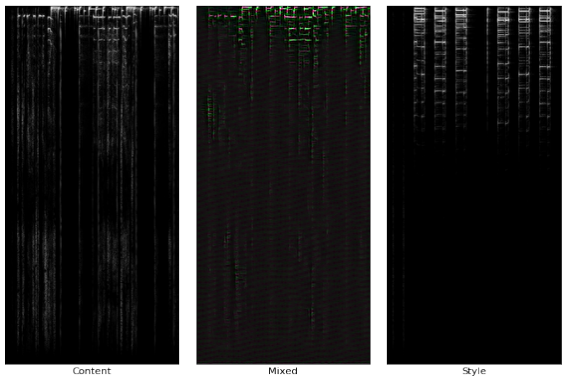

Fig. 19: Target spectrogram at 60 optimize iteration

In one of our training sample, we used two song which already has a style transferred version. One of them is *Hands to myself* by *Selena Gomez*, the other is *Love yourself* by *Justin Bieber*. The style transferred version is also available online [10].

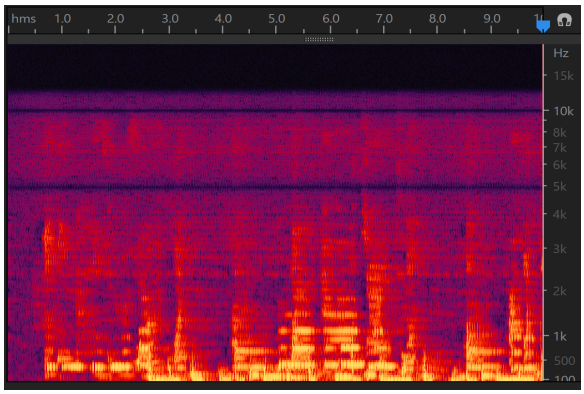

Fig. 20: Style transferred version of *Hands to myself* and *Love yourself* using our model

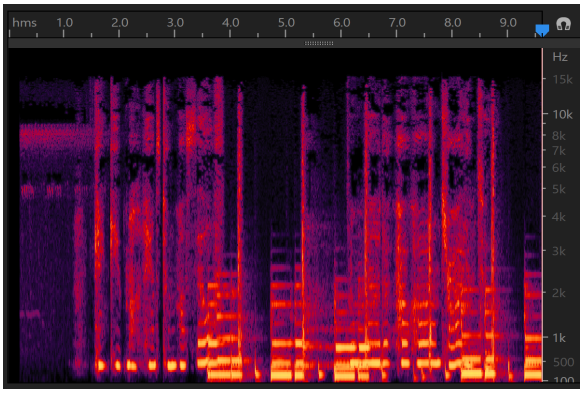

Fig. 21: Online style transferred version of *Hands to myself* and *Love yourself*

Figure 18 shows the spectrogram of our style transferred output according to these two songs, while figure 19 shows the online style transfer version which is done by professional musicians. The x axis represents for time and y axis represents for frequency. It can be seen that our style transferred version is similar to the online version at most of the frequencies. Notice the background of our output spectrogram is full of color comparing to the online version, that is background noise caused by the optimization process. Since we are optimizing the content file according to the style audio file, even the smallest sound in content file will be applied style, and the style information may make that small sound louder and noisy.

B. Results of Using VGGish as Feature Extractor

Remixing music with deep learning is definitely much faster than remixing manually, and because VGGish is trained to classify different sounds, it makes so much sense to deploy it under the scope of this project. However, due to the fact that preprocessing in VGGish that Hershey et al.[3] proposed resamples the original audio data from 44.1kHz to 16kHz, which results in a huge loss of information, and after we reconstruct the remixed spectrogram back to audio file, it is full of noise and the original contents and styles are completely shattered.

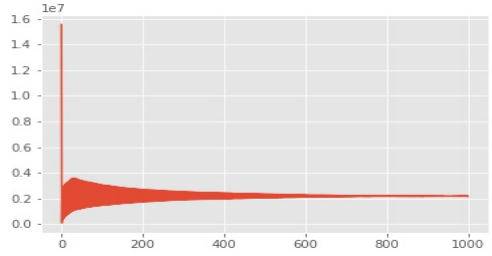

Fig. 22: Content Layer: 6th RELU, Style Layer: 2nd RELU, Content Weight:10, Style Weight:1.5

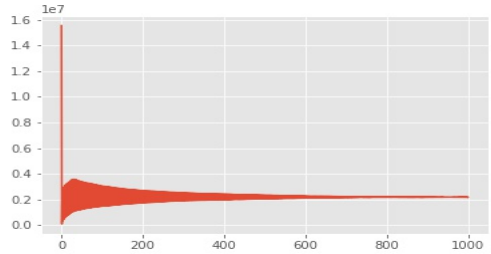

Fig. 23: Content Layer: 6th RELU, Style Layer: 2nd RELU, Content Weight:1.5, Style Weight:10

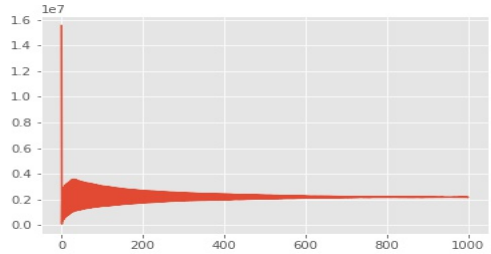

Fig. 24: Content Layer: 5th RELU, Style Layer: 1st RELU, Content Weight:10, Style Weight:1.5

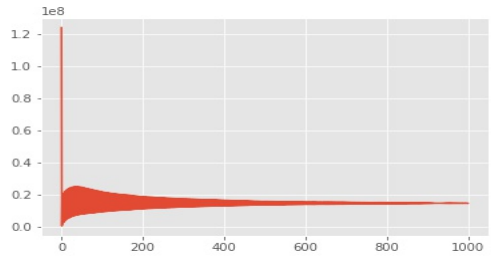

Fig. 25: Content Layer: 6th RELU, Style Layer: 1st RELU, Content Weight:1.5, Style Weight:10

In the above figures, y represents the value of our total loss function, and x is the number of iteration. We can clearly tell the loss reaches minimum after 1 iteration, and jiggles afterward, meaning that after 1 iteration of optimization, our output contains the most content information of the content spectrogram and most style information of the style spectrogram, shown as follows.

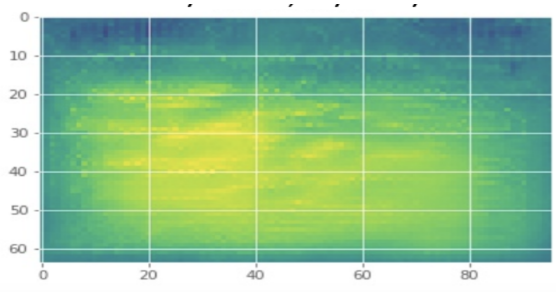

Fig. 26: Content Layer: 6th RELU, Style Layer: 2nd RELU, Content Weight:10, Style Weight:1.5

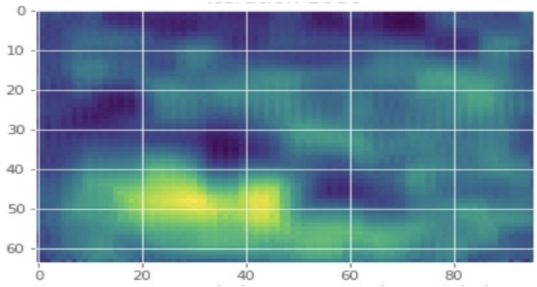

Fig. 27: Content Layer: 6th RELU, Style Layer: 2nd RELU, Content Weight:1.5, Style Weight:10

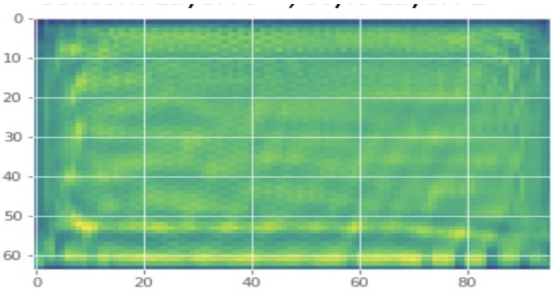

Fig. 28: Content Layer: 5th RELU, Style Layer: 1st RELU, Content Weight:10, Style Weight:1.5

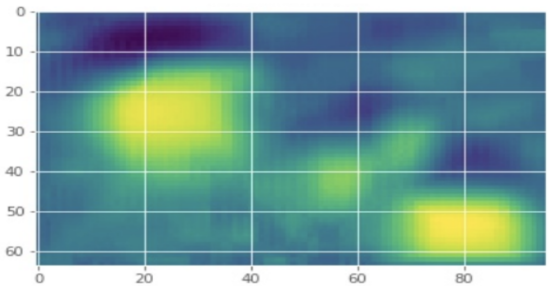

Fig. 29: Content Layer: 5th RELU, Style Layer: 1st RELU, Content Weight:1.5, Style Weight:10

In spite of the reconstruction issue with the destroyed audio data, we are still excited to see that style transfer with VGGish is plausible. Though it is still filled with unwanted noise compared to the content and style spectrograms converted from a real audio file, it did actually make the original random noise image much more arranged and patterned.

V. Limitations

Although our approach with VGG16 as the feature extractor successfully obtained a restyled audio file, there are still unwanted background noises produced during the style transfer process which we have yet found a solution to solving it. We have tried applying several noise deduction filters and reduced some of the background noise, however, it is still not perfect until all the noise is eliminated. There is a potential solution to put a threshold when reading audio files in order to filter out small sounds, and only apply style transfer to the non-zero part. Also the resampling steps in the preprocessing stage of VGGish has caused a huge issue for us to reconstruct the audio files, and to address this, we can build another convolutional neural network that is trained on spectrograms and without changing the data contents.

VI. Conclusions

Our audio style transfer project has successfully produced results on popular Youtube songs and has met our expectations. It fills the blank of multi-layer music processing using machine learning or deep learning. Although a number of single-layer music processing and generating works have been done and received excellent results, most of the multi-layer music processing algorithms generate noisy outputs. By utilizing VGG16 and VGGish, our approach is able to process and analyze multi-layer music in a short period of time and to give out a much less noisy version comparing to existing methods.

After testing on the output of different intermediate layers, we found the best performing content layer for VGG16 is the 5th convolutional layer and the best performing style layer is the combination of all layers. However, if taking computational cost into account, 1st-4th convolutional layers serve as a better choice. As for VGGish, since we have had a hard time reconstructing a clear audio file due to the resampling issue in the preprocessing steps, we can only tell the differences by looking at the spectrograms but not by listening to the audios, thus at current stage, we cannot conclude which layer combination is the best choice for the purpose of this project.

To sum up, VGG16 has clearly obtained a style transferred audio file which is close to the online style transferred version performed by professional musicians. Although VGGish failed to present a satisfactory audio output, it still managed to produce a style transferred spectrogram, and thanks to the fact that it is trained to classify audios, it is still promising that it will give a better performance on music style transfer problems if the reconstruction issue can be solved. Besides, to achieve a better result in the future, it is necessary to build a model that can recognize spectrogram features as VGGish does, yet without down-sampling the original audio data and keep the data as complete as possible.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. “A Neural Algorithm of Artistic Style”. In: *CoRR* abs/1508.06576 (2015). arXiv: 1508.06576. URL: http://arxiv.org/abs/1508.06576 (cit. on pp. 1, 2).

[2] Karen Simonyan and Andrew Zisserman. “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”. In: *ICLR 2015* (2015) (cit. on pp. 1, 3).

[3] Shawn Hershey et al. “CNN architectures for large-scale audio classification”. In: *ICASSP 2017 Changes* (2017), pp. 77–81 (cit. on pp. 1, 2, 4, 7).

[4] D. Griffin and Jae Lim. “Signal estimation from modified short-time fourier transform”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2 (1984), pp. 236–243 (cit. on p. 1).

[5] Lonce Wyse. “Audio spectrogram representations for processing with Convolutional Neural Networks”. In: *Proceedings of the First International Workshop on Deep Learning and Music joint with IJCNN* 1.1 (2017), pp. 37–41 (cit. on p. 2).

[6] Glen Tickle. “Daddy’s Car, A Song Composed by Artificial Intelligence Created to Sound Like The Beatles”. In: Flow Machines. Sept. 2016 (cit. on p. 2).

[7] Ji-Sung Kim. *Using Keras and Theano for deep learning driven jazz generation*. URL: https://deepjazz.io (cit. on p. 2).

[8] L. Fei-Fei. “ImageNet: crowdsourcing, benchmarking other cool things”. In: *CMU VASC Seminar* (Mar. 2010) (cit. on p. 3).

[9] Tarun Kumar and Karun Verma. “A Theory Based on Conversion of RGB image to Gray image”. In: *International Journal of Computer Applications (0975 fffdfffdfffd 8887)* 7.2 (2010), pp. 90–92 (cit. on p. 4).

[10] Justin Bieber and Selena Gomez. *Love Yourself / Hands To Myself*. URL: https://www.youtube.com/watch?v=3nhiY-yeCJw&app=desktop (cit. on p. 6).