Music Generation using Style Transfer by Deep Learning Methods

Akhilesh Gupta, Anavi Kaushik, Celine Lee, Hanrong Zhu

May 8, 2020

Project Drive Link

1 Introduction

Music is among one of the first domains to be digitized and processed by modern computers and algorithms. We were fascinated by the research done by Noam Mor et al. [4] which was kind of the first time where they produced high fidelity musical translation between instruments, styles, and genres. This opened up directions of music generation by using style transfer mechanisms. Style transfer has been one of the topics of interest in the area of image and computer vision but applying these to music and other areas like NLP has recently seen a growing trend.

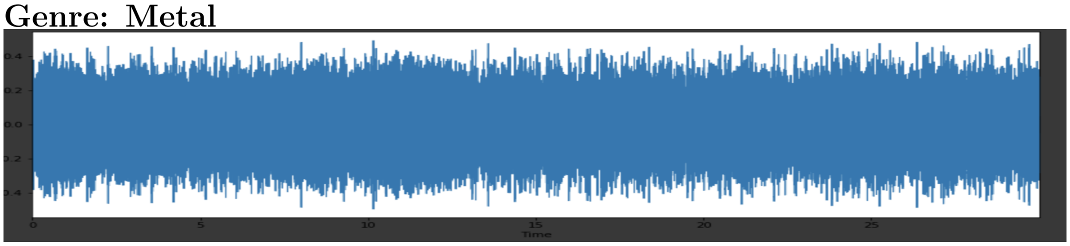
In this research we aim to building a new machine learning technique for generating music and audio signals. The focus of this work is to develop new techniques parallel to what has been proposed for image style transfer by Zheng Xu et al.[6]. The work in artistic style transfer became revolutionary after the seminal idea by Gatys et al. [3] in how we can in some sense alter the "style" of an image while generally preserving its "content". In this work, we present a method for creating new sounds using a similar approach by applying the techniques discussed by Zheng Xu et al.

1.1 Dataset

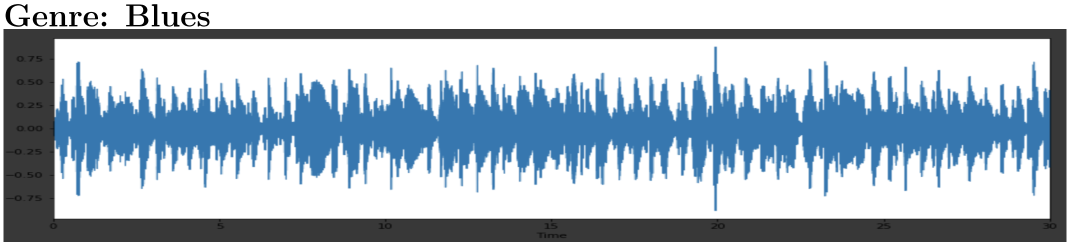
We used the Free Music Archive (FMA) [1], an open and easily accessible dataset suitable for evaluating several tasks in MIR (Music Information Retrieval). This dataset has 10 genres namely: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. Each of these genres has 100 music files in wav file extension of 30 seconds each.

Visualizing the dataset:
We plan to use the dataset in the form of a numpy array using the LibROSA [1], a python package for music and audio analysis. Also it can be visualized in the waveform as below:

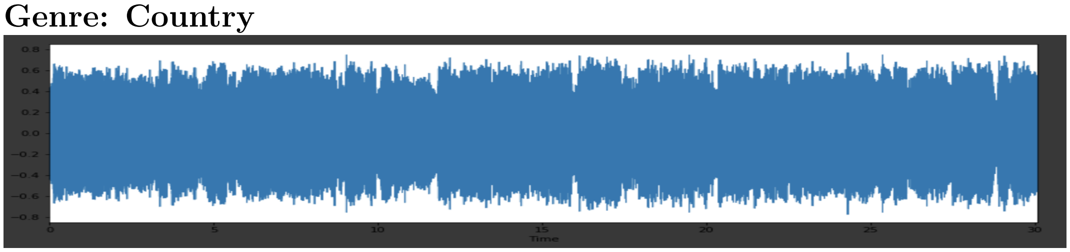
Genre: Metal



Genre: Blues

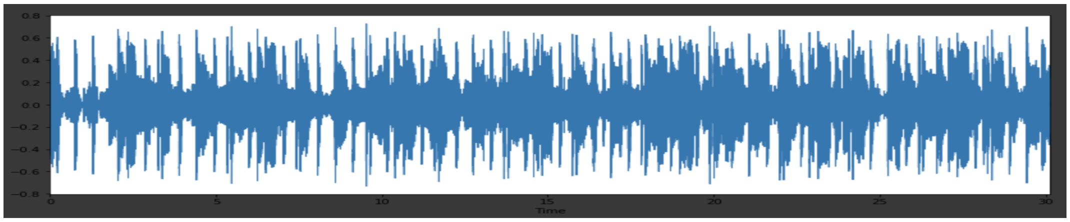


Genre: Country



Genre: Disco

[1]https://librosa.github.io/librosa/



Genre: Jazz

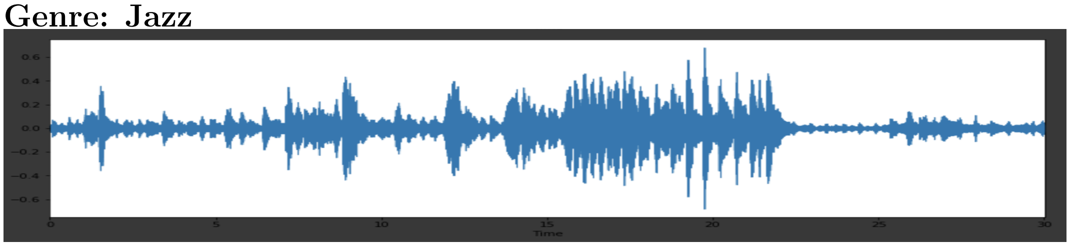


Fig 1. Waveform of audio clips for different genres

The spectogram visualization of the data is as below:

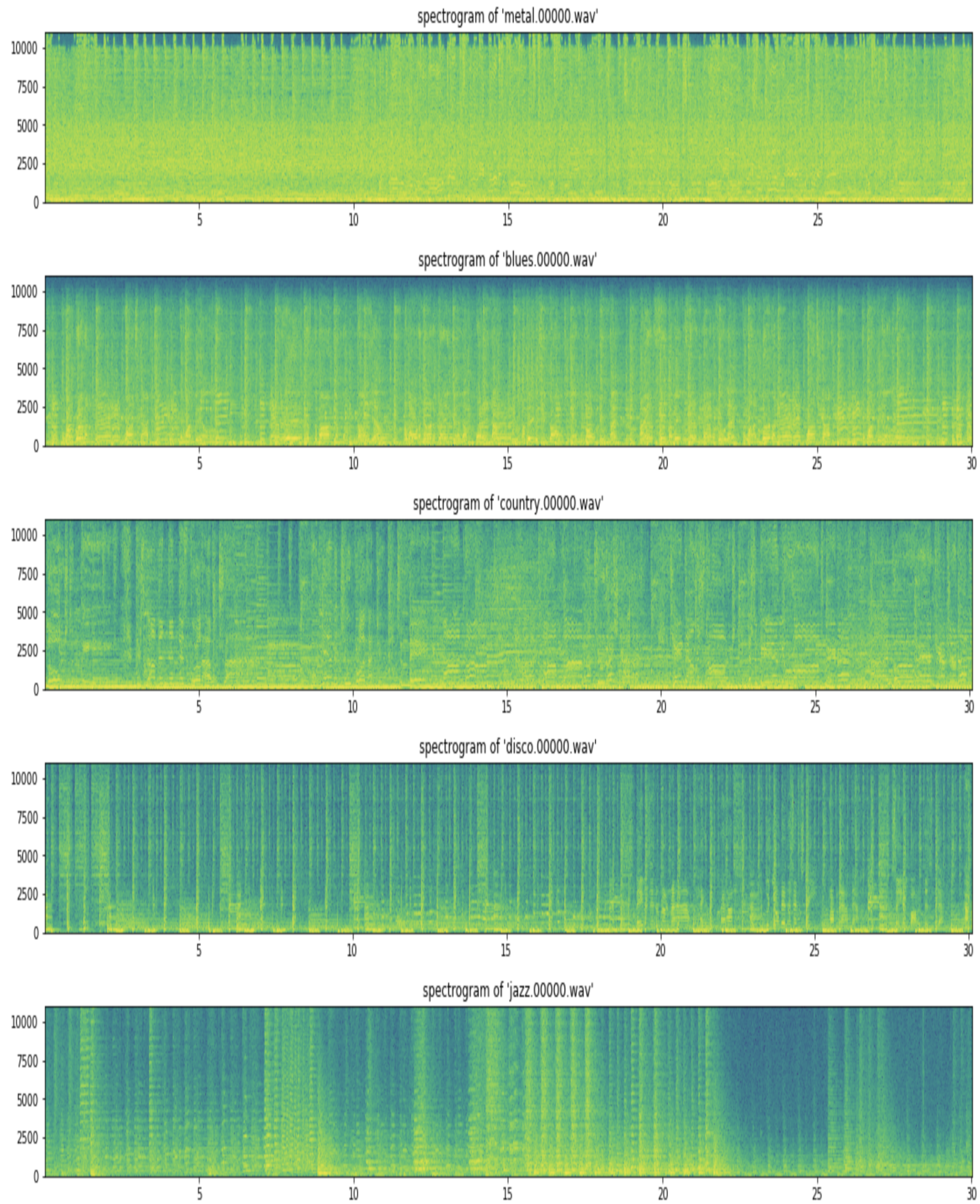


Fig 2. Spectogram visualization of audio clips for different genres

1.2 Hypothesis

The hypothesis for this research is to be able to generate a completely new music form, say "Audio 1" by taking the style component from one of the given audio, say "Audio 2" as an input and applying this style to another audio input, say "Audio 3". We can then let a classifier predict the genre or type of this music. The evaluation metric can be determined by the inability of this classifier to distinguish the style of "Audio 1" from "Audio 2". They will both have the same style, the content however will be same for "Audio 1" to "Audio 3". This would be done by training the deep learning network on various genres and types of data to help the model learn the characteristics of one genre and hence be able to mix the content and style of these audio inputs.

2 Related Works

Our hypothesis is a variation of Image style transfer applied to audio synthesis, inspired from readings we have found. A summary of the related works, their relation to our project, and key insights found in their papers are below:

A Universal Music Translation Network (Noam Mor et al.)[4]: In this paper, the authors explore an unsupervised learning method to translate music across musical instruments, genres, and styles. The novelty in this paper is that they found that by reducing the size of the latent space, the decoders become more "creative" and can produce more original outputs that actually lose exact association with the original input. A disentangled latent space is combined with a multi-domain wavenet autoencoder trained on waveforms. In their results, they were able to translate from musical domains not even seen during training. The NSynth dataset is used for evaluation, measuring correlation of embeddings across pitch, timbre, and envelope for multiple instruments. We can use inspiration from this paper's feature extraction and findings on latent space, to guide our data collection/transformation.

Audio texture synthesis and style transfer (Ulyanov):[2] Ulyanov focuses on extending style transfer to audio. He starts by converting raw audio to a spectrogram via short-term Fourier Transform, thus treating the input as an

[2]https://dmitryulyanov.github.io/audio-texture-synthesis-and-style-transfer/

image. Texture synthesis, a method used in object recognition, and style transfer are then used to in defining a network that can produce a new spectrogram output, which is then reconstructed into an audio wave. We will explore these featurization methods in extracting information from our dataset, and embedding the information in a "content layer" and "style layer".

A Neural Algorithm of Artistic Style[2]: In this paper, they have introduced an artificial system based on a Deep Neural Network that creates artistic images of high perceptual quality. The system uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images. The key finding of this paper is that the representations of content and style in the Convolutional Neural Network (CNN) are well separable. It is possible to manipulate both representations independently to produce new, perceptually meaningful images. The results presented were generated on the basis of the VGG network [5], which was trained to perform object recognition and localisation. The feature space that was provided by a normalised version of the 16 convolutional and 5 pooling layers of the 19-layer VGG network was used. Also, they normalized the network by scaling the weights such that the mean activation of each convolutional filter over images and positions is equal to one. Such re-scaling can be done for the VGG network without changing its output, because it contains only relu's and no normalization or pooling over feature maps. They did not use any of the fully connected layers.

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks[7]: In this paper, the authors proposed CycleGAN, a new type of GAN network that utilizes the ideas of both adversarial loss and cycle consistency to realize style transferring between images.Like normal GANs, CycleGAN also has a structure of generator and discriminator. The difference is in CycleGAN, we train on two separate generators with one mapping style A to style B and the other vice versa. Accordingly, there are also two discriminators for both style A and B.The final objective is to minimize the loss of these two GAN pairs, as well as cycle consistency loss,which is the loss of transferring style A to B and then back to A. This method has proved to have amazing performance on image style transferring, and we hope to explore the possibility of applying to audio style transferring.

We draw inspiration from these findings as the audio files in wav format can be represented as an image using spectogram and the same style transfer technique can be applied to an audio file as well. The input to this model would be an input wav file in one genre along with an input wav file from another genre so that the music can that is output will be of second genre with the content of the first genre.

3 Methods

Our methods are comprised of three phases: during first phase, we implement a simple CNN model as our baseline genre transfer model and a separate CNN as our baseline classifier. In the second phase, we design an RNN to perform the same task of music transfer, and consider improvements to our classifier using autocorrelograms. In the final phase, we will improve the model by adding in elements of GAN.

3.1 Baseline Approach (CNN)

In order to train a CNN on audio files, we need to first do data preprocessing to convert our audio data to spectrogram representation using Short-Time Fourier Transform (STFT). Since the Pytorch framework does not handle complex tensors, we instead capture the magnitude and phase of the complex array returned by the STFT function, train on the magnitude, and recover with the phase. We then feed the 2D representation of signals in the STFT to a shallow CNN with 1D convolutional layers.

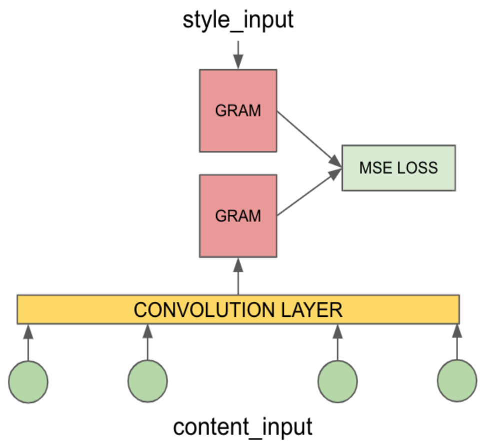


Fig 3. CNN Network overview

We feed into the CNN the content audio and extract style from the style audio using the Gram matrix calculation, and try to minimize the style loss, which is the mean squared error between the gram matrices of style audio and the generated audio.[3] We found that with this approach, a shallow CNN performed best. Our final baseline CNN is structured as follow:

Layer	Details
Conv1d	in channels=1025; out channels=4096; kernel size=3; stride=1; padding=1
Gram Matrix	multiplication of itself with its transpose, normalized

The output of the CNN is then computed against the gram matrix output of the style audio using mean-squared-error loss, weighting the target style input at some certain higher value; this is a hyperparameter we tune. The optimizer is then stepped forward, the CNN is backpropagated using that loss value. Once the model has run for the number of epochs desired, the signal is reconstructed from the last output and the original phase extracted from the STFT.

Through testing on various pairs of content and style inputs, we found that the set of hyperparameters that seemed to generally perform most consistently was:

[3]https://software.intel.com/en-us/articles/neural-style-transfer-on-audio-signals

Optimizer	Adam lr=0.005
Style Weight	500
Num Steps	2500

3.2 Deep Learning Approach (RNN, BiLSTM)

In this approach, we implemented 2 models, namely Recurrent Neural Networks and Bidirectional long short-term memory (biLSTM). Since these models take into account the output from the previous inputs, they help a lot in the area of music generation.

A deep neural network is one that has more than one hidden layer of units (neurons) between its input and output layers. Essentially, a neural network transforms an input by a series of cascaded non-linear operations.

It can be difficult to train standard RNN because the gradient of the loss function decays exponentially with time (vanishing gradient problem). LSTM tackles this problem by introducing both cell states and a hidden states. BiLSTM networks are a type of RNN that uses special units in addition to standard units. BiLSTM units include a "memory cell" that can maintain information in memory for long periods of time. Hence LSTM can learn longterm dependencies and perform way better than RNN. The primary reason seems to be directly related to the training of the underlying time series processes (first from left-to-right and then right-to-left). As a result, the BiLSTM based learning model needs to fetch additional data batches to tune its parameters. Hence it takes longer to converge, but makes sure that the generated style transferred music has less noise factor.

A recurrent neural network (RNN) has looped, or recurrent, connections which allow the network to hold information across inputs. These connections can be thought of as similar to memory. RNNs are particularly useful for learning sequential data like music.

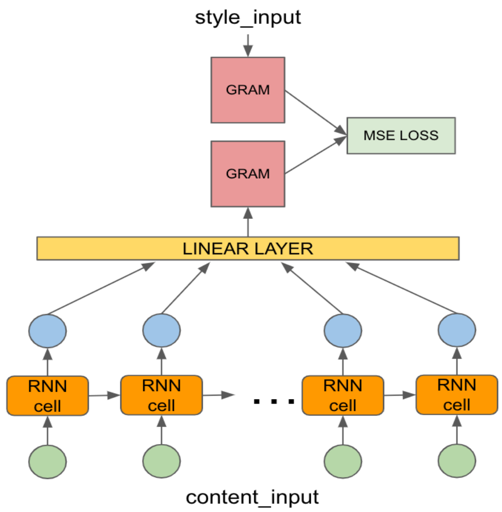


Fig 4. RNN Network overview

The RNN Network which is used is as given below:

Layer	Details
Recurrent Layer	input size=430; hidden size=100; num layer=1;
Linear Layer	input size = 100; output size = 4096; bias = True
Gram Matrix	multiplication of itself with its transpose, normalized

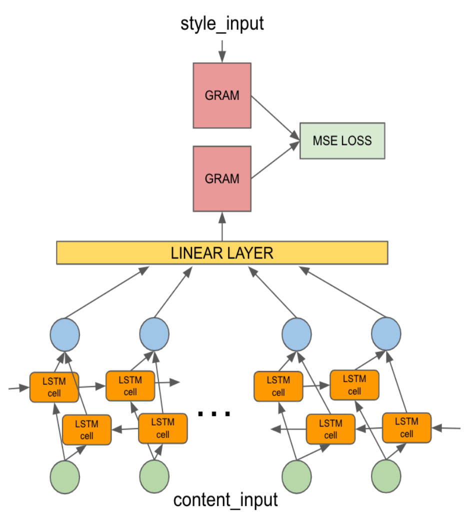


Fig 5. BiLSTM Network overview

The biLSTM Network which is used is as given below:

Layer	Details
LSTM	input size=430; hidden size=100; num layer=1; biderectional = True
Linear Layer	input size = 100; output size = 4096; bias = True
Gram Matrix	multiplication of itself with its transpose, normalized

We tried to tune the hyperparameters for both the recurrent neural networks, BiLSTM and RNN. We tried with a range of learning rates starting from 0.5 to 0.00005. With larger learning rates, it was seen that the loss was always 0 and did not increase or decrease as the number of steps increased. A learning rate of 0.005 for RNN and a learning rate of 0.00005 for BiLSTM produced the best results in terms of the style loss and the quality of music generated in the end. We also tried changing the number of steps for generating the music in RNN and BiLSTM. RNN outputs with fewer number of steps of 3000 were filled with noise and the input content or input style was not audible in the output. Increasing the number of steps to 10000 for both RNN and BiLSTM produced good results. BiLSTM gave similar results even with a fewer number of steps of 8000. We also tried using different optimizer of Stochastic Gradient Descend (SGD), but this optimizer did not help in convergence even after many number of steps. The style loss did not decrease on using SGD. Adam optimizer worked very well in this case

The differences in their basic fundamental phenomenon made the LSTM to perform better with a lower learning rate and it converged faster as compared to RNN.

We tried number of different parameters for both the models. We found out that except the learning rate everything else seemed to be same for both the networks. The hyperparameters which worked the best for biLSTM were :

Optimizer	Adam lr=0.00005
Style Weight	2500
Num Steps	10000

RNN used a higher learning rate of 0.005 .

3.3 Advanced Deep Learning Approach (GAN)

Inspired by the CycleGAN approach on computer vision, which is proposed in "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks"[7], we decided to explore its effectiveness on audio style transferring. Following the data processing method from previous models, we also first transformed the audio files into Short-time Fourier transform (STFT) matrices, and fed them into our GAN networks as if they had been grey-scale images. The CycleGAN networks consist of 2 generators with one transferring style A to style B and the other vice versa, and also two discriminators for both style A and B. The generators are encoder-decoder network that transforms an input STFT matrix to another after style transferring. The network looks as following:

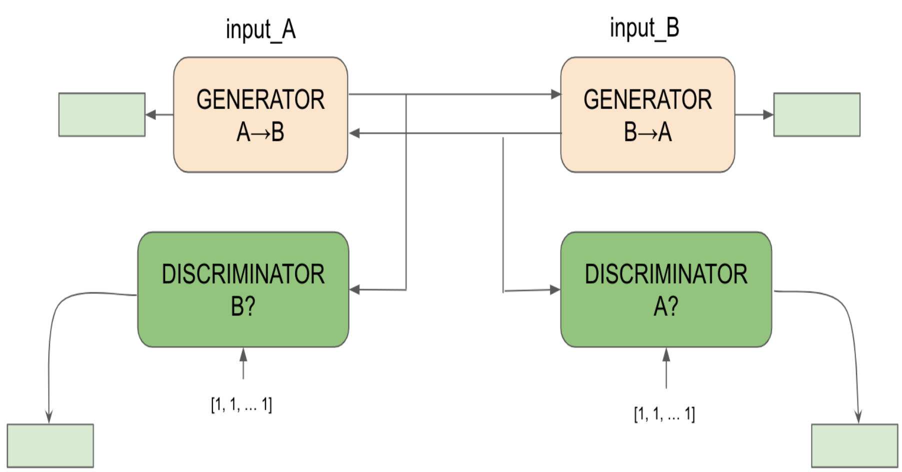


Fig 6. GAN Network overview

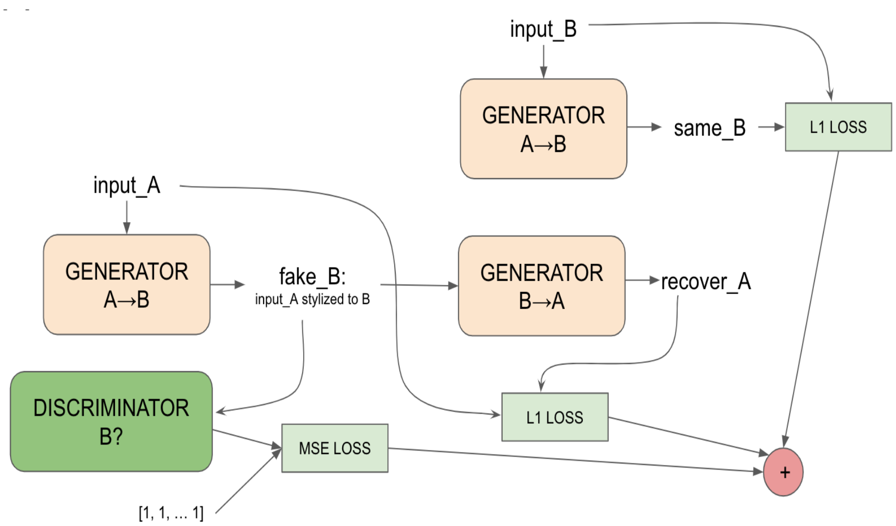


Fig 7. GAN Generator overview

Layer	Details
RefectionPad2d Conv2d InstanceNorm2d Relu	padding = 3 in channels=1; out channels=4; kernel size=7; stride=0; padding=0 in channels = 4 in place = True
Conv2d InstanceNorm2d Relu	in channels=4; out channels=8; kernel size=3; stride=2; padding=1 in channels = 8 in place = True
Conv2d InstanceNorm2d Relu	in channels=8; out channels=16; kernel size=3; stride=2; padding=1 in channels = 16 in place = True
Residual Blocks	num = 3
ConvTranspose2d InstanceNorm2d Relu	in channels=16; out channels=8; kernel size=3; stride=2; padding=1 in channels = 8 in place = True
ConvTranspose2d InstanceNorm2d Relu	in channels=8; out channels=4; kernel size=3; stride=2; padding=1 in channels = 4 in place = True
RefectionPad2d Conv2d Tanh	padding = 3 in channels=4; out channels=1; kernel size=7; stride=0; padding=0

Of these layers, residual blocks are added after down-sampling layers to retain the characteristics of the original input.The structure of the residual blocks is as following:

Layer	Details
RefectionPad2d	padding = 1
Conv2d	in channels=8; out channels=8; kernel size=3; stride=0; padding=0
InstanceNorm2d	in channels = 8
Relu	in place = True
RefectionPad2d	padding = 1
Conv2d	in channels=8; out channels=8; kernel size=3; stride=0; padding=0
InstanceNorm2d	in channels = 8

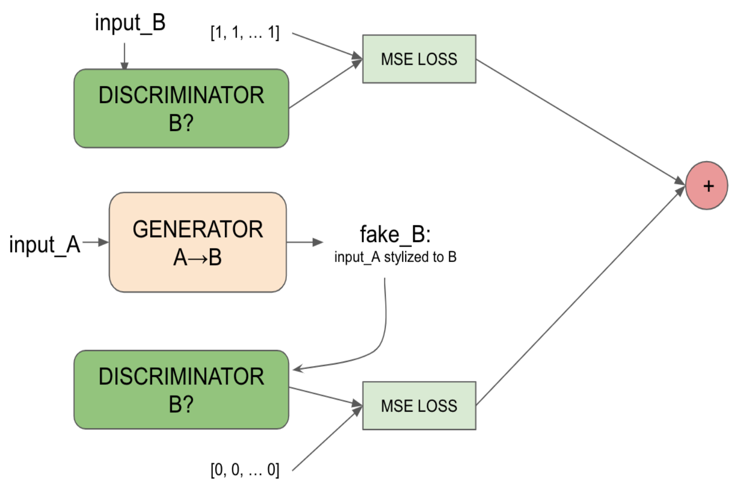


Fig 8. GAN Discriminator overview

The discriminators are 2D convolutional classifiers, and their structure is as following:

Layer	Details
RefectionPad2d	padding = 1
Conv2d	in channels=1; out channels=8; kernel size=4; stride=2; padding=1
Relu	in place = True
MaxPool2d	kernel size = 3, stride = 2
Conv2d	in channels=8; out channels=16; kernel size=3; stride=2; padding=2
Relu	in place = True
MaxPool2d	kernel size = 3, stride = 2
Conv2d	in channels=16; out channels=32; kernel size=3; stride=2; padding=2
MaxPool2d	kernel size = 3, stride = 2
Linear	in dim = 32×10×16, out dim = 512
Relu	in place = True
Linear	in dim = 512, out dim = 1
Sigmoid	

The object is to minimize the loss of 2 GAN pairs: generator $G_{A,B}$ with discriminator D_B, and generator $G_{B,A}$ with discriminator D_A, plus the cycle consistency loss, which is loss incurred from $G_{B,A}(G_{A,B}(x_A)) - x_A$.

Our chosen hyperparameters include Adam optimizer with learning rate 1e-3, batch size 10 (we have only 100 audios for each genre) and epoch 100.

3.4 Music Genre Classifier

In the baseline style transfer CNN, we use a gram matrix as the determining metric for style. However, we cannot assume that this perfectly captures style, so we also take steps to develop a music classifier using deep learning. This classifier is used throughout our experimental process as a more quantitative checkpoint of how well our models perform (as opposed to letting our ears make that call). For the baseline genre classifier, reconstruction is not needed, so we focus on extracting features we believe are relevant to how music genre is perceived. We settled on the following features: chromagram, spectral centroid, spectral bandwidth, spectral roll-off, zero-crossing rate, and mel-frequency cepstral coefficients. These features are then compiled together to represent the data points for a particular audio file. The classification CNN is structured as follows:

Layer	Details
Conv1d	in channels=1; out channels=32; kernel size=4
LeakyReLU	
BatchNorm1d	channels=32
MaxPool1d	kernel size=2
Dropout	probability=0.2
Conv1d	in channels=1, out channels=64, kernel size=3
LeakyReLU	
BatchNorm1d	channels=64
Dropout	probability=0.5
Flatten	
Linear	in=1572, out=256
LeakyReLU	
Linear	in=256, out=64
LeakyReLU	
Linear	in=64, out=10
Softmax	

The CNN was tuned by trying different numbers of convolutional layers, size of convolution kernels, differnt numbers and sizes of linear layers, amount of dropout and pooling, and different activation functions. The best-performing CNN design worked with the following hyperparamters:

Optimizer	Adam lr=1e-4
Loss function	Cross entropy loss
Num epochs	500
Batch size	32

The resulting F1 score on a test dataset from this model is 0.63. The training loss and f1 are shown in figures 9 and 10.

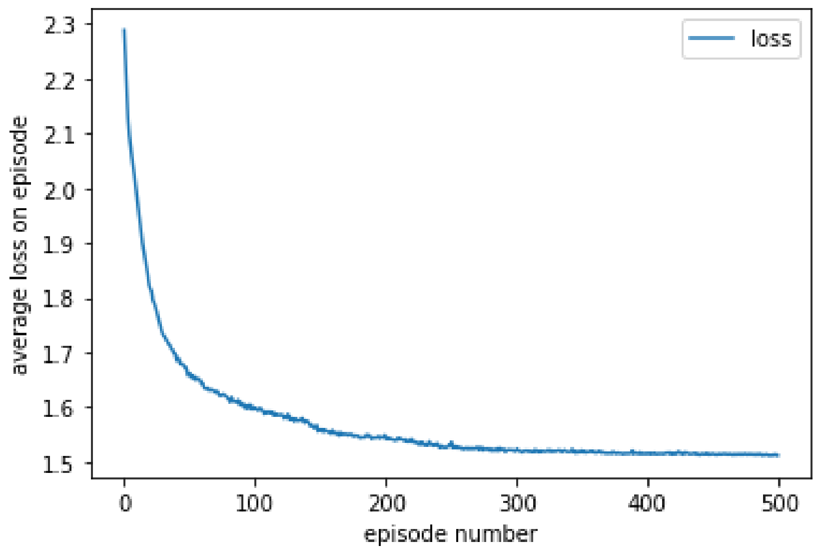


Fig 9. Baseline classifier training loss

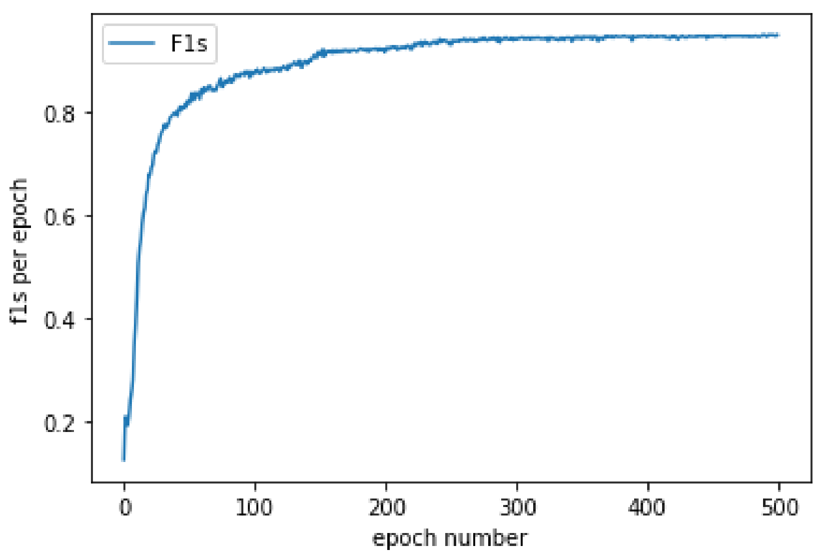


Fig 10. Baseline classifier F1 scores over training

In considering improvements to our classifier, we also explore the potential improvement that autocorrelograms may bring to our music genre classifier. Since autocorrelograms provide a method to capture rhythm in a time series data, we believe that this might be better suited with a CNN because it captures relationships across "distance" and forms them into features more local to each other. A major weakness in applying CNNs to music analysis is that harmonic relationships are frequently at patterned distances from one another, which convolutional filters are not as effective at extracting. See figure 11 for an example of autocorrelograms for each genre.

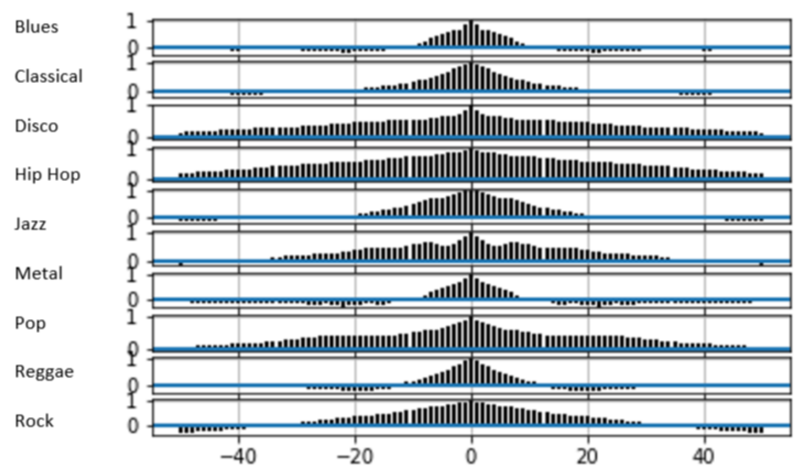


Fig 11. Autocorrelograms for each of the 10 genres.

We design a dataloader to compute the autocorrelograms of each music file, and tune a CNN to perform classification. The resulting model is designed as follows:

Layer	Details
Conv1d	in channels=1; out channels=32; kernel size=4
Tanh	
BatchNorm1d	channels=32
Dropout	probability=0.2
Flatten	
Linear	in=1536, out=128
Tanh	
Linear	in=128, out=64
Tanh	
Linear	in=64, out=10

Unfortunately, after significant efforts in tuning the optimizer, learning rate, number and shape of linear layers, number and shape of convolutional layers, and number of epochs, the best performance achieved was a 0.254 F1 score on the test data with the following hyperparameters and training plots:

Optimizer	Adam lr=1e-3
Loss function	Cross entropy loss
Num epochs	6

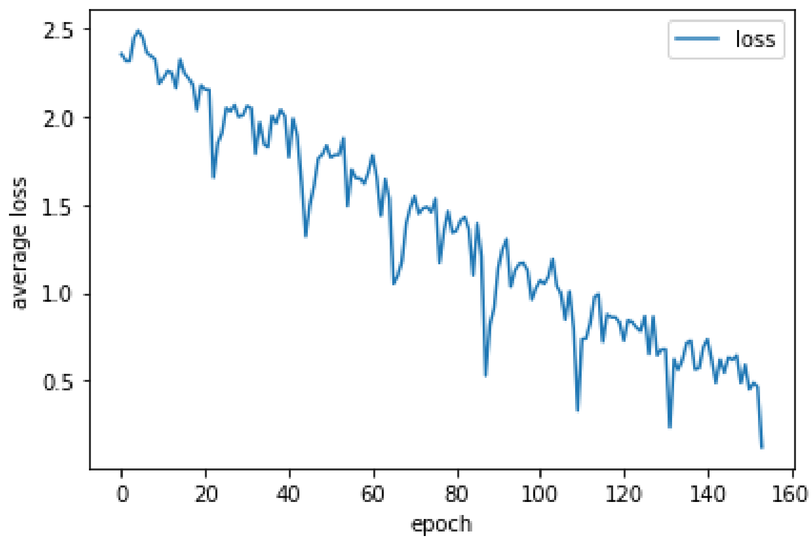


Fig 12. Loss graph of training using autocorr data.

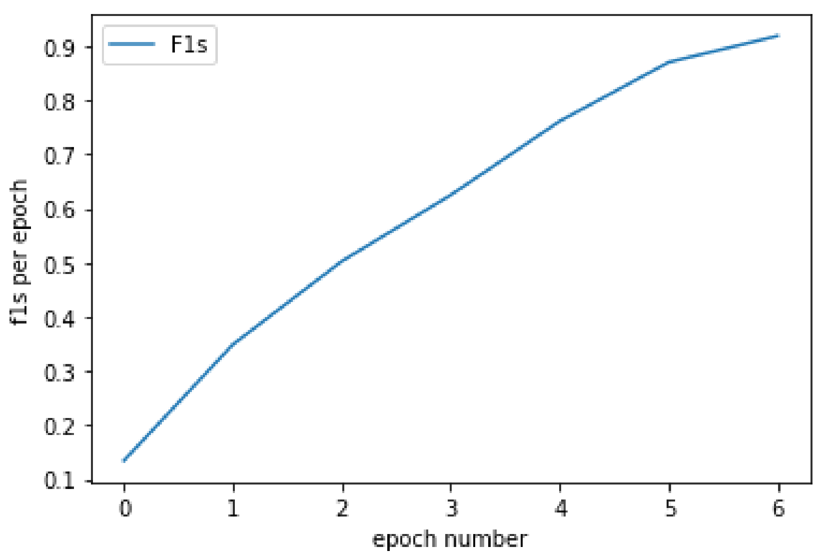


Fig 13. F1 graph of training using autocorr data.

Consequently, we do not use this model for our classification, but do find the exploration into autocorrelograms to be a valuable topic for the question at hand.

4 Analysis

Various approach have been tried for this interesting problem. We started with a basic CNN model and advanced to RNN, biLSTM and to GAN based approaches. To compare all the models we benchmarked these with 2 audios, specifically classical music genre to jazz music genre. We will be comparing the style loss of these models, how the models converged for these specific inputs. In addition to style loss, as our basis of the study was spectogram analysis. We analysed the spectogram of the generated music from all the models and compared them to see which performed the best.

The other way of benchmarking is to give the generated music to the classifier model which tries to predict the genre. We used that but didn't get a lot of differences in the result as the outputs are noisy and the model itself gave 0.61 testing accuracy after training on the clear music sounds. We will discuss more on this later in this section.

For the baseline model, we obtain the following loss curves as calculated by the Gram matrix loss:

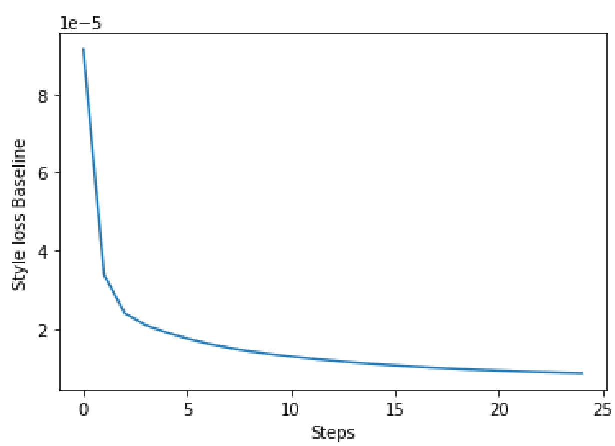


Fig 14. Baseline CNN Style loss vs Steps

Style Loss Plot for RNN model:

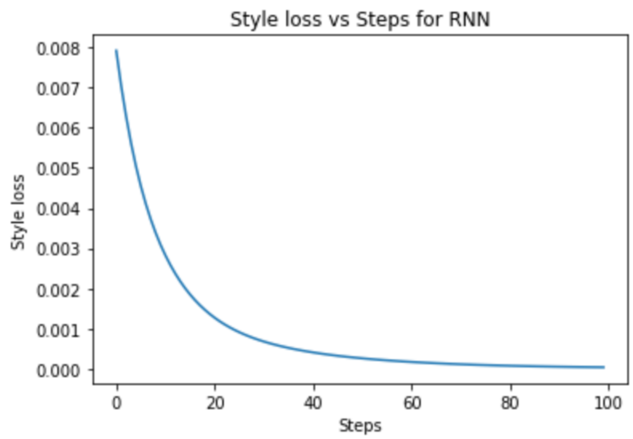


Fig 15. RNN Style loss vs Steps

Style Loss Plot for biLSTM model:

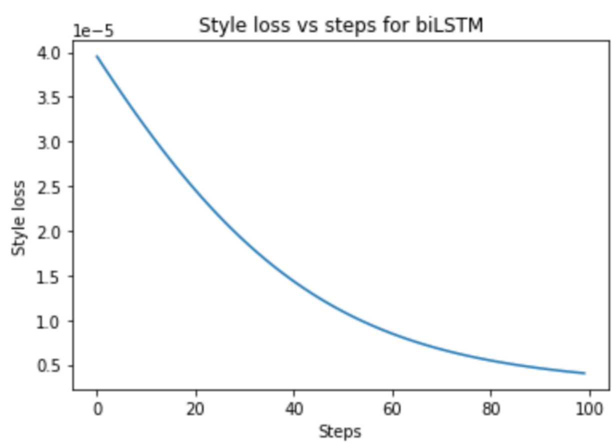


Fig 16. BiLSTM Style loss vs Steps

After running CycleGAN on our data to transfer between Jazz and Classical, we got loss curves as following:

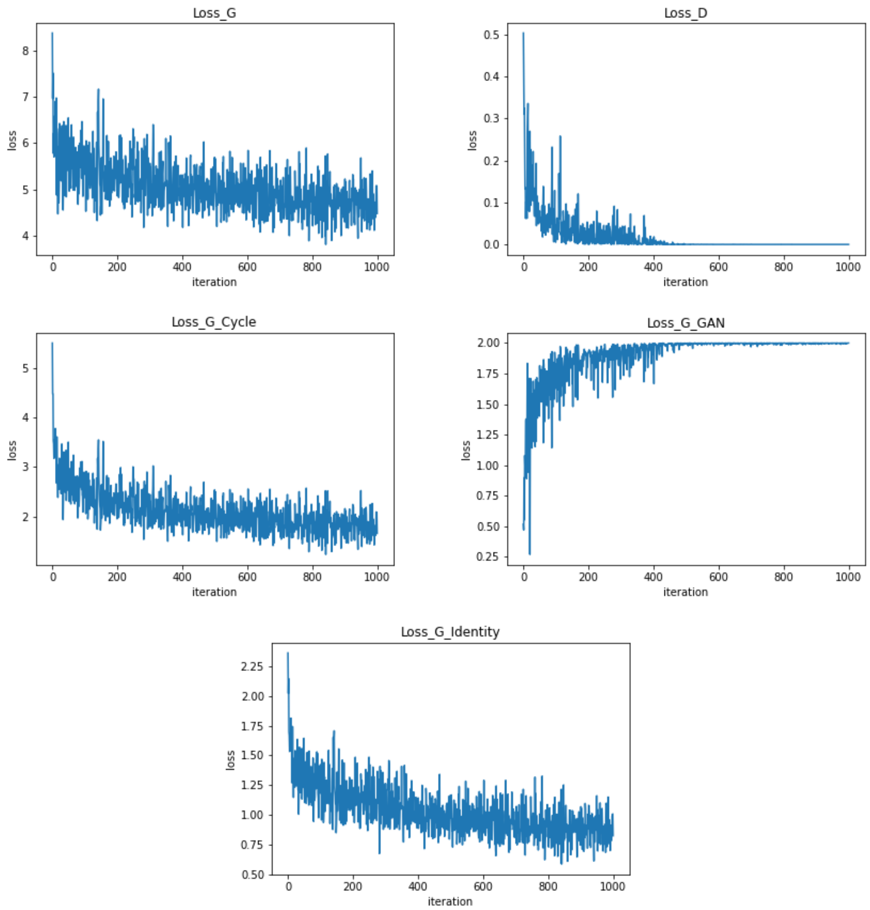


Fig 17. GAN loss plots

We went further to analyse the spectograms of the music generated after the style transfer for RNN and biLSTM models. We did it only for the recurrent models as we got the least noisy and decent output from these models. We wanted to prove our hypothesis from the spectogram.

Below is the spectogram of the inputs of content and style and then the generated music from RNN:

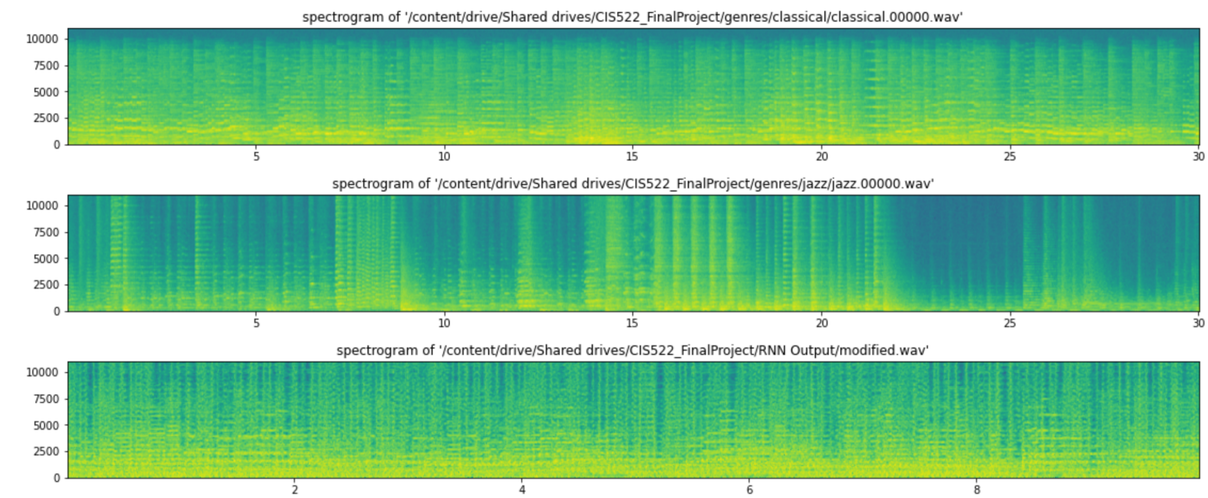


Fig 18. Spectogram for RNN output: (top to bottom: content, style, output)

We can see that the generated music spectogram has characterstics from both. It does conserve the content and gets the style from the style audio. The audio output can be heard from RNN

Below is the spectogram of the inputs of content and style and then the generated music from biLSTM:

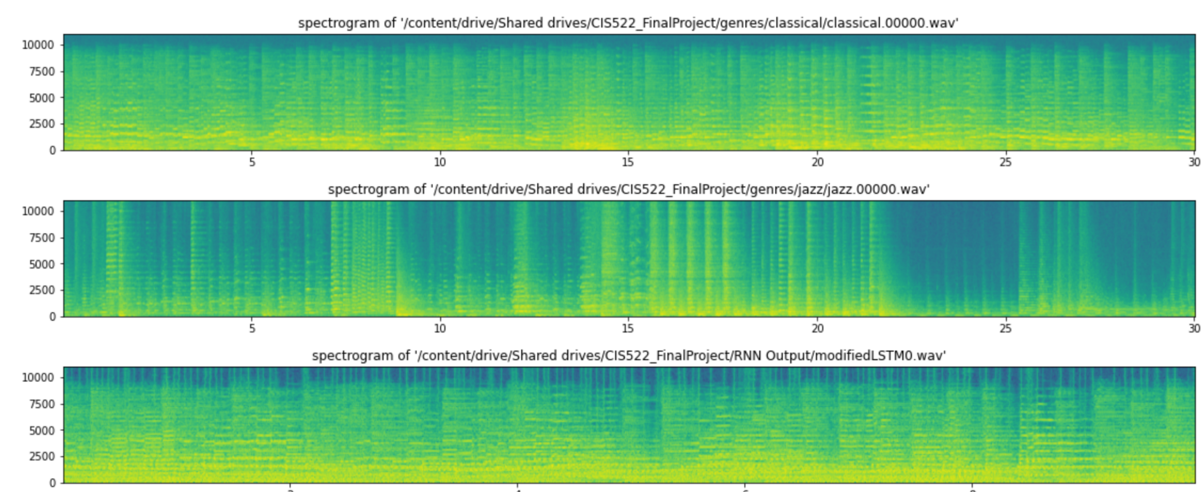


Fig 19. Spectogram for BiLSTM output: (top to bottom: content, style, output)

The audio output can be heard from biLSTM

We can observe the output spectrogram of the baseline as well. The first is the content, second is the style and third is the spectogram of the generated music

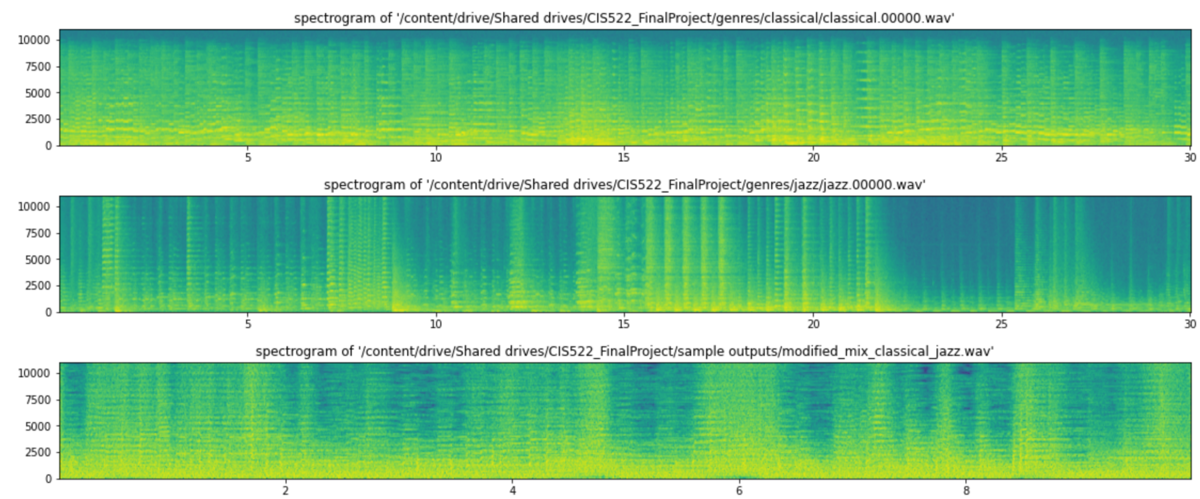


Fig 20. Spectogram for baseline CNN output: (top to bottom: content, style, output)

The spectrogram of the generated music from GAN is as given below. The first is the content, second is the style and third is the spectogram of the generated music:

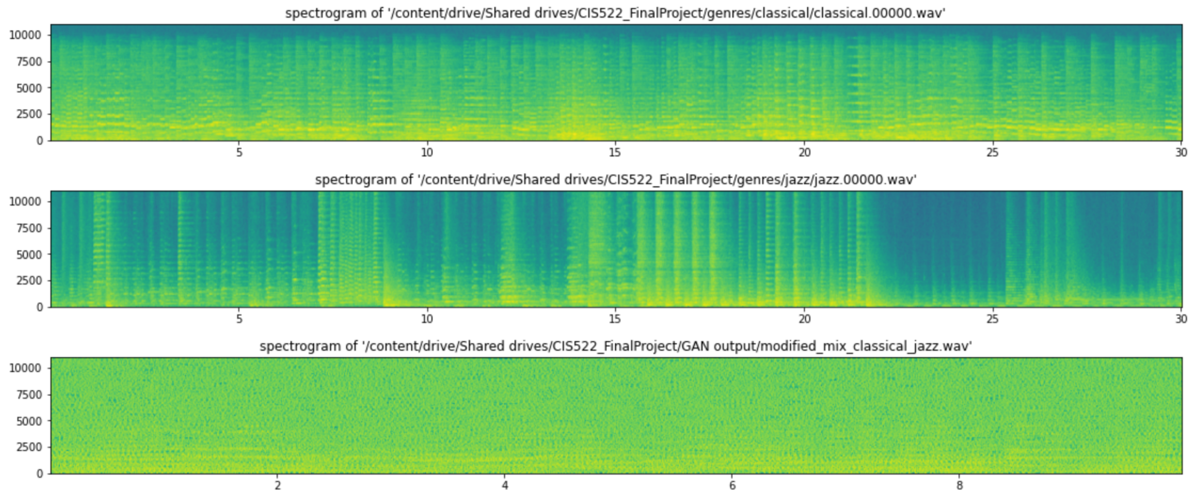


Fig 21. Spectogram for GAN output: (top to bottom: content, style, output)

We can see that there is almost white noise with little content in the audio for the GAN output. The outputs from this model are also given in the end of this section.

As we tested different models over the course of this project, we observed that the one that performed best was biLSTM. The reason for this is likely the fact that recurrent networks do take the component of time series and as we know in music the next note is dependent on the previous one and has various different characterstics as per the genre of the music. Hence, we did observe this to be true and in comparison to baseline model which was a CNN based model recurrent models performed better with least noise in the outputs and quicker convergence. We took the approach of a GAN network as our final advanced deep learning model. The reason for which we discussed in the model section. GAN being difficult to train and hence we got noisy outputs in comparison to the RNN and biLSTM outputs. The function GAN networks try to optimize is a loss function that essentially has no closed form (unlike standard loss functions like log-loss or squared error). Thus, optimizing this loss function is very hard and requires a lot of trial-and-error regarding the network structure and training protocol. This is also one of the reasons why there is very few research or approaches using GANs to anything more complex than images, such as music or even text.

For content music as Classical Music and style as Jazz Music, the output with each of the model using these different techniques for the models discussed above are in below links:

1. Baseline CNN.
2. RNN
3. biLSTM
4. GAN

5 Discussion

Since our project ended up being a two-pronged approach to applying deep learning to music style, our efforts ended up being split between developing a music genre transfer model and developing a music genre classifier. Based on the results we have found, it is clear that there is much work to be done on how to capture the way that humans perceive sound and music, and how to transfer that knowledge to deep learning approaches.

We found the best results with the BiLSTM model, which hints that a recurrent model approach that can maintain long-term memory is appropriate for this application. However, since the results in all models are still notably noisy, we believe that the next step would be to first find the source of the "noise", and design a way to eliminate its randomness and/or form it into the sense of rhythm and tune that defines the target style genre. One model we have considered, but did not have time to yet explore, is one that first focuses on extracting the content (ie melody, cadence) out of the content input before trying to apply style to it. We would also want to combine in the flexible elements of an RNN so that the newly-stylized audio output could also benefit from difference in timing and rhythm. We think that the extraction of "pure content" could drastically decrease the distracting static in our output. This could allow us to inject rhythm in a flexible manner, which will also add to the ability to perceive a new genre in the output.

We can further improve on evaluation mechanism by using a state of the art genre classification mechanism. This can help to find the gap in the style transfer and also help to realize which genres are difficult to train and do the style transfer on.

References

[1] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. *arXiv preprint arXiv:1612.01840*, 2016.

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[3] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[4] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *arXiv preprint arXiv:1805.07848*, 2018.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Zheng Xu, Michael J. Wilber, Chen Fang, Aaron Hertzmann, and Hailin Jin. Beyond textures: Learning from multi-domain artistic images for arbitrary style transfer. *CoRR*, abs/1805.09987, 2018.

[7] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A.Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv:1703.10593v6*, 2017.