AML Assignment
Music Generation using MFCC

Name	USN
Akhil Sundaram	01FB15ECS027
Anish Murthy	01FB15ECS039
Hardik Mahipal Surana	01FB15ECS116

Problem Statement

The goal of this project is to create sequence-to-sequence deep learning models to generate new sequences of music. We tackle the domain of polyphonic music, which is more complex in structure and in which it is more difficult to find patterns.

Potential Approaches

Music is highly structured, but difficult to create procedurally. For this reason, we have attempted to use several models in an attempt to learn to generate music. The tried approaches include:

1. Variational Autoencoders:

A variational autoencoder (VAE) resembles a classical autoencoder and is a neural network consisting of an encoder, a decoder and a loss function. They let us design generative models of data and fit them to large datasets and can also be used for image generation and reinforcement learning. For example, a practical application may be to generate trees for a forest in a video game, which are all similar but not the same.

This behaviour of VAEs allows us to use them for music generation as the audio files generated from these may be varied while still being recognised as from the same source.

2. LSTM Networks:

An LSTM Network is an improvement over a vanilla recurrent neural network. It handles the exploding and vanishing gradient problems well that traditional RNNs would struggle with. It can also deal with data where there can be lags of unknown duration between important events in time. These cells allow the network to retain memory, thereby being able to handle temporal data well.

The behaviour of LSTM which allows it to retain information of the current and previous state make LSTM a good choice for music generation, as music tends to follow a trend of having the same kinds of periods of delay between important events, namely, beats.

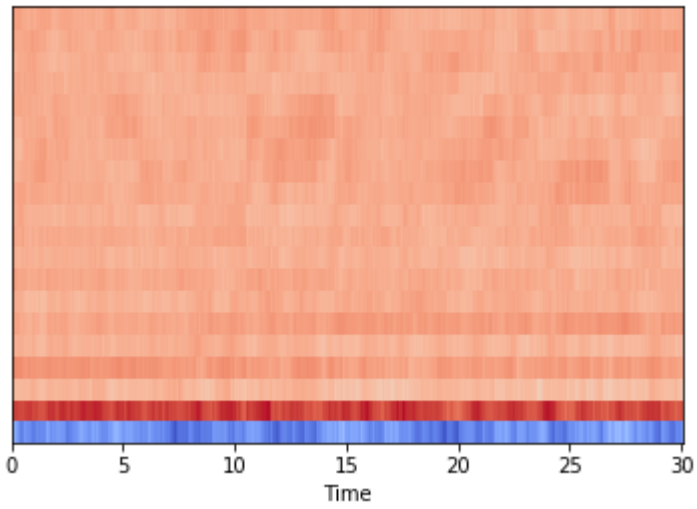
Methodology Followed

The dataset used in this project are the Free Music Archive (https://github.com/mdeff/fma) with size 8 GB and 1000 songs per type and the GTZan (https://marsyasweb.appspot.com/download/data_sets/) genre classification dataset whose size is around 1.2 GB with around 100 songs and 10 genres.

To generate new music sequences, the input audio is first converted to numerical form. In contrast to easier but limited options like using MIDI or Piano Rolls as the representation of the input data, we use MFCCs (Mel Frequency Cepstral Coefficients) . Mel - Frequency Cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel - Frequency Cepstral Coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). In the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human ear response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

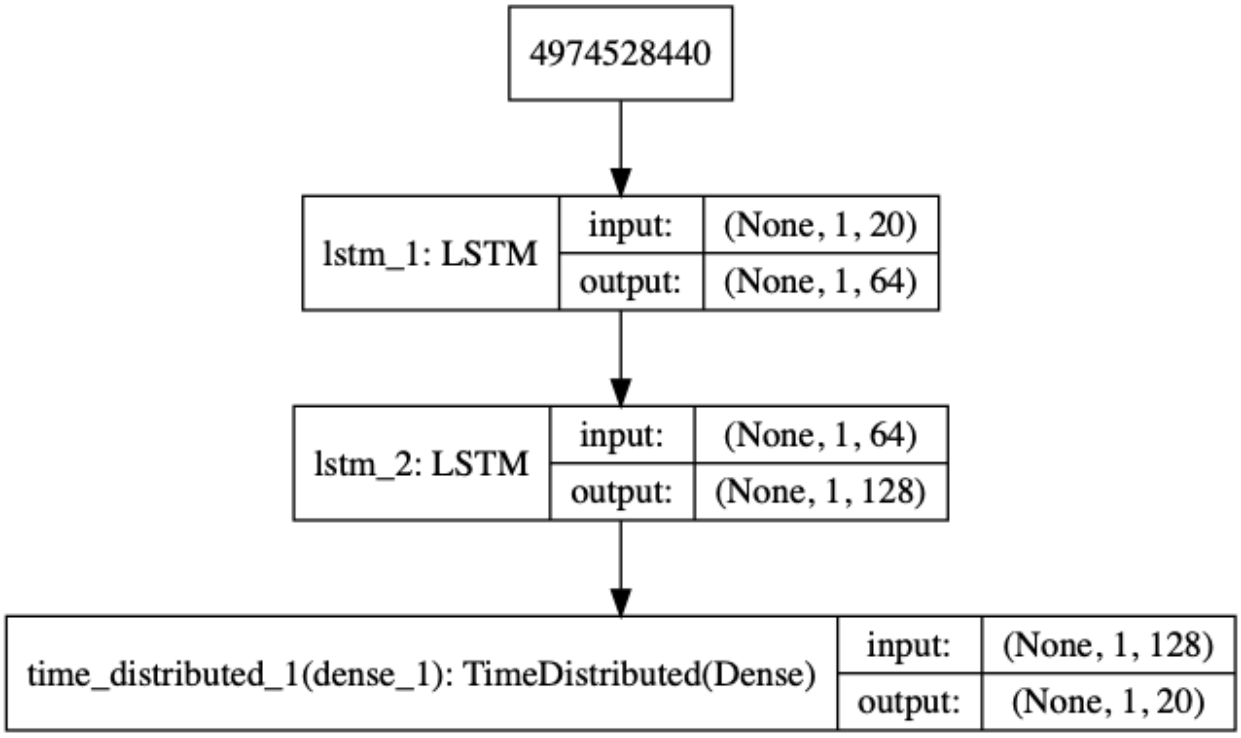
The main drawback in this representation is the lossy conversion back to audio signals. This divides the audio into small frames (typically with the width of few milliseconds) and encodes the energy information in frequency domain found in a given frame. The number of coefficients generated range from 13-40. We have chosen 20 coefficients for our dataset. The output of the model is also chosen to be an MFC representation of the desired track. Since the MFCCs are a representation of a sound, they can effectively be used to obtain a sound waveform from the output MFCCs, and hence allows for generation of music.

A spectrogram of MFCCs shown below identifies the frequency ranges which encode information of any given song's energy in the frequency domain.



During the training process, we tried several variations of the LSTM model. Due to the large size and importance of the input values, we did not apply any squashing activation function like relu, tanh etc. Instead, we followed the linear activation. Thus, we faced the exploding gradient problem. To overcome the same, we used a shallow network to reduce the number of parameters. Surprisingly, these shallow networks with 2 LSTM layers performed better than a network with 3-4 layers.

The LSTM model architecture followed is shown below:



To handle temporal information, the 2D dataset is reshaped to 3D. Each timestep has 20 features, corresponding to the 20 MFCC coefficients. The train-test split of 80-20 was followed.

The generation/synthesis of new music requires that the model be able to read and learn the input audio and generate new clips based on this input. This is accomplished by training the model on several songs, by having the model take clips of the music as input and attempt to generate the next clip, which is chosen as output. The next clip is then fed into the model as input and is used to generate the next output, and so on. This is repeated with music of similar genres, in order to obtain an effective generative model that can generate clips which sound similar. The model is then used to generate new clips by having a small sample of music input to the model and allowing it to generate the next clip, having each of these outputs fed back into the model as input and repeating the generation process until a desired length of audio is generated.

Several variations of the Variational Auto Encoder Model was trained on. The VAE model used is the standard vanilla model. The input MFCC of shape (20,1293) is flattened into a vector of size (25860,) to be passed to the model. The autoencoder model does not give a low loss value, and at times overshoots “infinity” to a gradient exploding problem in some configurations. This is especially seen when training on a higher number of inputs; ~300 music samples of 30 seconds each.

The VAE model architecture is as follows:

```
In [14]: print(vae.summary())
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 25860)	0	
dense_3 (Dense)	(None, 2580)	66721380	input_1[0][0]
dense_4 (Dense)	(None, 128)	330368	dense_3[0][0]
dense_5 (Dense)	(None, 128)	330368	dense_3[0][0]
kl_divergence_layer_1 (KLDiverg	[(None, 128), (None,	0	dense_4[0][0] dense_5[0][0]
lambda_1 (Lambda)	(None, 128)	0	kl_divergence_layer_1[0][1]
input_2 (InputLayer)	(None, 128)	0	
multiply_1 (Multiply)	(None, 128)	0	lambda_1[0][0] input_2[0][0]
add_1 (Add)	(None, 128)	0	kl_divergence_layer_1[0][0] multiply_1[0][0]
sequential_1 (Sequential)	(None, 25860)	67077480	add_1[0][0]

```
Total params: 134,459,596
Trainable params: 134,459,596
Non-trainable params: 0
```

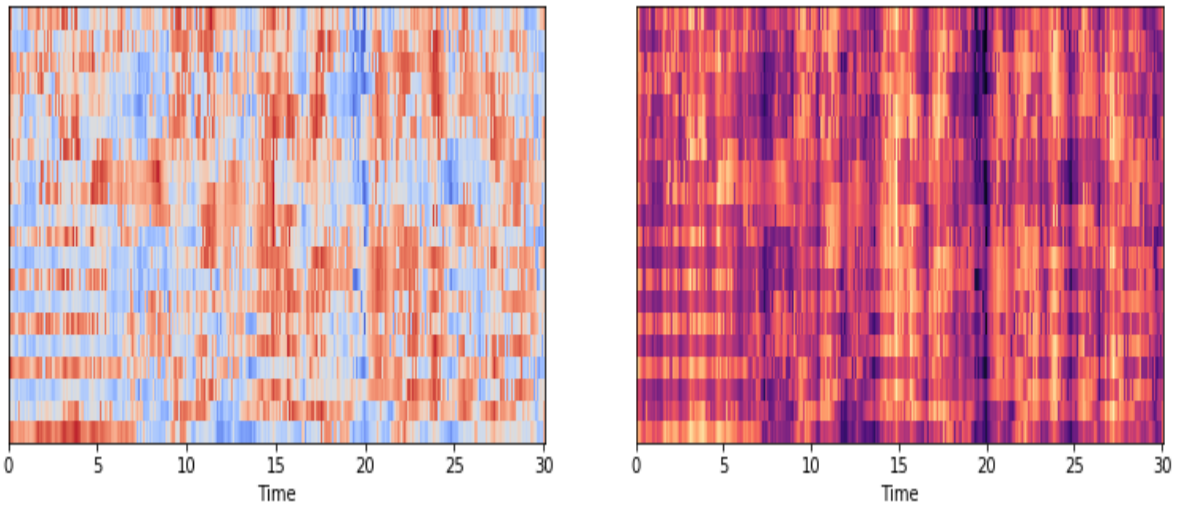
The model generates/creates new music by training the latent dimensions of the model.It is trained using samples/clips of music fed through the encoder-decoder network. The decoder is then used to create new sampling from the information encoded in the latent dimensions. A vector of values of mean 0.05, and variance 0.95 is then created to be used to generate music samples.

Tests and Results

Due to computational limitations, the training took places for the classical genre from the GTZan dataset consisting of 100 songs of 30 seconds duration each. The training loss achieved by LSTM is 22.3379 and the validation loss achieved is 22.8960, thereby showing minor overfitting. To optimize on the same, the Recurrent Dropout of 0.5 was used along with checks on the validation loss using the ReduceLROnPlateau and EarlyStopping callback techniques were employed. Along with this, the SGD, RMSProp and Adam optimizers were tried.

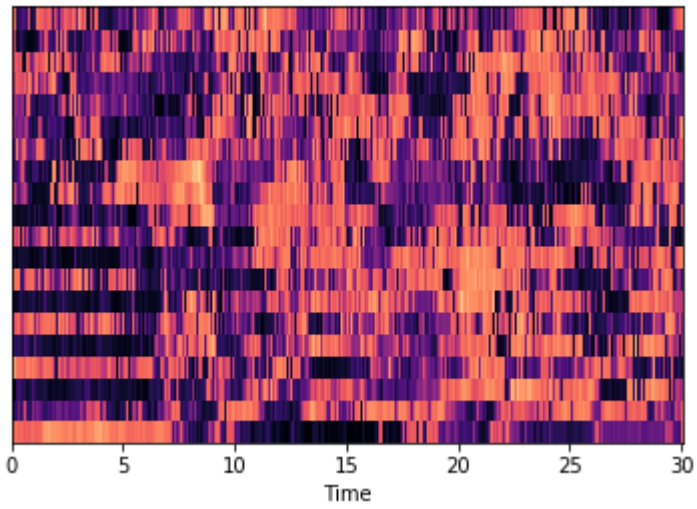
The music generated from VAE varied with every training instance. The training loss achieved was very high, but further training only exploded the gradient to NaN values:loss= 29290.578, val_loss=4319.1016 and loss = 26.837x 10^{22} , val_loss=21.743x 10^{13} . The gradient explodes the loss value to Infinity(NaN) when using using MFCC 2D data without any normalization.

Therefore, the MFCC was first normalised before passing to the network. It was found that modifying the Inverse_transform(DeNormalization) of data determined the output of the Model. To rescale the output mfcc, sklearn's MinMax Transformation was first used(fitted) on an existing MFCC to act as template. The "Inverse_transform" was then performed on the output MFCC. It is seen that rescale range also affects the output of the mfcc:



The above two images represent the same MFCC output ,with the former using a rescale range of (-1,1), while the latter uses a rescale range(-2,2). The former produces better sound.

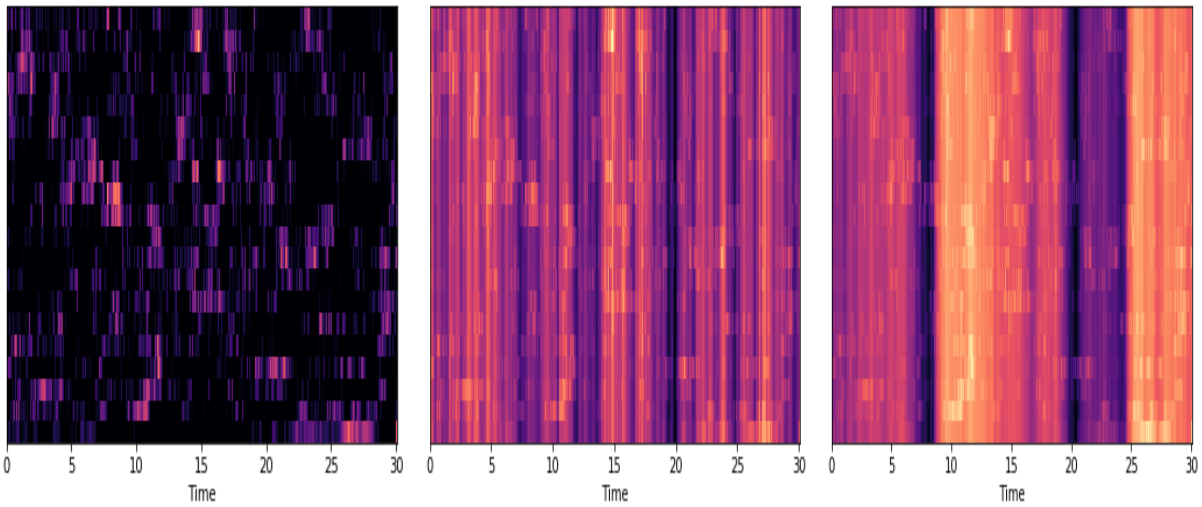
While the original MFCC through the model looks as follows:



The original MFCC portrayed above produces only noise on conversion to wav audio output.

While not all MFCC outputs produced any meaningful sounds, it was also seen that the "template" mfcc used to rescale the outputs also plays a part in reproducing better sounds.

The following images demonstrate this fact:



The Original(Left) Mfcc image shows a learned pattern that is squashed together. While trying to Rescale it back, MFCC of two other images where used as references or as “templates”. The sound generated by both(Middle, Right) vary in their quality significantly.

To conclude, it is seen that further research is needed to develop a better way to extrapolate the “Music/Sound” patterns generated by the Models. Furthermore, more parameters need to examined/tuned to synthesize and encode Music better.

Paper References

[1]
http://louistiao.me/posts/implementing-variational-autoencoders-in-keras-beyond-the-quickstart-tutorial/
[2]
https://jaan.io/what-is-variational-autoencoder-vae-tutorial/
[3]
http://colah.github.io/posts/2015-08-Understanding-LSTMs/