CS464 TERM Project

Section 2 - Group 19

Spring 2020

Music Genre Classification

Final Report

Team Members

Doğukan Köse 21602375
Meryem Efe 21601779
Muhammed Musab Okşaş 21602984
Fatih Çakır 21601370
Nehir Kırtaş 21502024

Instructor: Ercüment Çiçek

Teaching Assistant: Oğuzhan Karakahya

1.Introduction

Music shops and especially online platforms such as Spotify, Apple Music, etc. require a music genre classification algorithm when suggesting and keeping different kinds of genres in their platforms. There are quite a number of music tracks in these platforms and organizing every music tracks by their genres is significantly difficult. By organizing each music track by its genre, the platforms can develop further systems to suggest users new tracks by predefined genres.

In our term project, we have decided to find the best music classification algorithm by genres. The best is defined by the highest accuracy. The accuracies are given for each music genre, average accuracy and top 3 accuracies. For the purpose of classification, we have used the following methods: 1D Convolutional Neural Network (1D CNN), 2D Convolutional Neural Network (2D CNN), Long Short-Term Memory (LSTM) and SVM.

The results (average accuracies) obtained from these methods are the following:
1D CNN with Raw Waveform: 51% and Top 3 accuracy: 79%
1D CNN with Mel Spec: 64% and Top 3 accuracy: 86%
2D CNN with Mel Spec: 41% and Top 3 accuracy: 74%
Long Short-Term Memory (LSTM): 62% and Top 3 accuracy: 85%
SVM: 64%

1.1. Description of the Data

The dataset we have used for our project is GTZAN Genre Collection dataset. This dataset includes 1000 music tracks for 10 different genres: blues, classical, country, disco, hip-hop, jazz, reggae, rock, metal, and pop. Each genre has 100 audio tracks inside and each of the tracks consists of 30 seconds.

The audio tracks are preprocessed with the LibROSA package of python that extracts features from spectrogram of audio tracks. For our methods, we have used these extracted features and the pictures of the spectrogram.

1.2. Problem Description

We are trying to classify audio tracks into different music genres therefore, we are trying to address the problem of classification of song genres. Every music genre has different characteristics that make them separable and different from other music

genres such as frequency, bandwidth, etc. By using these features and characteristics, the audio samples we have can be classified and organized. By using the music genre classification, especially online music platforms such as Spotify and Apple Music can differentiate the song genres from each other and can suggest users specific types of genres by their needs.

2.Methods

In this project various methods are used to classify music genres. Firstly, music audio files are pre-processed, then 4 different methods; 1D Convolutional Neural Network (1D CNN), 2D Convolutional Neural Network (2D CNN), Long Short-Term Memory (LSTM) and SVM trained on these pre-processed data.

2.1. Data Pre-Processing

The dataset consists of 1.000 songs with 10 different genres. Each music audio is 30 seconds. Songs are in the different genre files, but they were not labelled at the beginning. That is why firstly, songs are loaded on python as 30 seconds waveform with Librosa library.[1] We load data with 22050 sample rate and load all of 30 seconds of songs. So, one of our samples includes 30*22050 = 661500 features at the beginning. Then, each of wave sample is labelled with its folder name. However, folder names are string, that is why we enumerated each class from 0 to 9. (Raw waveforms are used with 1D CNN.)

These waveforms converted into Mel Spectrogram Images (MSI) to use them for other classifying methods. While waveforms are being converted into MSI, Librosa.feature.melspectrogram function is used by using parameters n_fft = 2048, hop_length=512, sample_rate = 22050 and amplitudes of waveforms are converted into decibels in MSI. Then, each of MSI is saved into folders to be used for classifier methods.[2]

However, these images were too large at the beginning. Their size was 2160x720. It was difficult to train these images. Therefore, we decided to rescale them. Firstly, we convert them into 120x120 and 360x120 images. 360x120 images are used for SVM classifier. However, for neural networks we decided to rescaled them to a little bit bigger images. So, we rescaled 2160x720 images to 400x400 images and used them for 1D CNN, 2D CNN and LSTM.

As a result, we used 3 different pre-processed data from 1 dataset. These are, 1000x661500 raw waveforms, 1000x360x120x3 with 3 channel Mel Spectrogram Images and 1000x400x400x3 Mel Spectrogram Images. Also, PCA is applied on 400x400x3 Mel Spectrogram Images for better understanding of distribution of data.

2.2. 1D Conv with Raw Waveform

While implementing 1D convolutional neural network on waveforms, we used *pytorch* library. We tried to train all of 30 seconds of song files which have 661.500 features each. However, it was difficult to train on 661.500 features. For that reason, we used 25 seconds of wave files which have 551.250 features each.

We divided 1000 data samples into 3 separate dataset which are training set: 700, validation set: 100 and test set: 200. After we found best model on validation set, we concatenated training set and validation set and tried on test set.

Details of one of the best model that we found on raw waveforms, will be given below:

- 6 Layers: 4 of them for convolutional, 2 of them fully connected layers

Layer 1:

Conv1	Input channel: 1	Output channel:8	Kernel size: 27	Stride: 2
Mp1	Pooling size: 4			
ReLu				

Layer 2:

Conv2	Input channel: 8	Output channel:16	Kernel size: 52	Stride: 1
Mp2	Pooling size: 8			
ReLu				

Layer 3:

Conv3	Input channel: 16	Output channel:48	Kernel size: 97	Stride: 2
Mp3	Pooling size: 4			
ReLu				

Layer 4:

Conv4	Input channel: 48	Output channel:128	Kernel size: 127	Stride: 1
ReLu				

Layer 5:

Fc1	Input size: 119936	Output size: 128
ReLu		

Layer 6:

Fc2	Input size: 128	Output size: 10
Softmax		

- Batch size: 50
- Learning rate: 0.001
- Optimizer: Adam

- Number of epochs: 100 or until training accuracy larger than 90% (to avoid overfitting)
- Activation Function: All ReLu (Last one softmax)
- Loss Function: Negative Log Likelihood
- Train set ratio: 80% (70 for training 10 for validation)
- Test set ratio: 20%

We tried different models on validation set, but the given model has the best results.

In this model, we select Adam optimizer instead of SGD because Adam optimizer converges faster than SGD and linear data is too long, that's why training it with SGD takes too many epochs compared to Adam optimizer. Also, Adam optimizer gives the better result for this model.

2.3. 1D Conv with Mel Spec

1D Convolutional Neural Network is generally used on the sequential input like audio or text. We used 1D Convolutional Neural Network with Pytorch library. As mentioned before, we used 3 different processed data from 1 dataset.

We shuffled the data and separated 700 songs for train set, 100 songs for validation set and 200 songs for test set. Number of layers and parameters are tried with different values to find the best result.

There are 5 layers. It includes 3 convolution and 2 fully connected layers.

Layer 1: Conv1d(3,8,4,2), AvgPool1d(4), ReLU
    Conv1d:
        input channel: 3
        output channel: 3
        kernel size: 4
        stride: 2
    Average Pool:
        size: 4

Layer 2: Conv1d(8,16,32,2), AvgPool1d(4), ReLU
    Conv1d:
        input channel: 8
        output channel: 16
        kernel size: 32
        stride: 2

Average Pool:
size: 4

Layer 3: Conv1d(16,64,16), MaxPool1d(4), ReLU
Conv1d:
input channel: 16
output channel: 64
kernel size: 16
Max Pool:
size: 4

Layer 4: Linear(39680,100), ReLU
input size: 39680
output size: 100

Layer 5: Linear(100,10), Softmax
input size: 100
output size: 10

- Batch Size: 70
- Learning Rate: 0.001
- Epoch: 100 (break early if training accuracy exceed %80)
- Optimizer: Adam

The values of parameters are decided after several trials to get best results. Adam optimizer is used for this model after trying different optimizers such as SGD, Adagrad, AdamW, Adamax.

2.4 2D CNN with Mel Spectrogram

To implement 2D convolutional neural network, we used both *pytorch* and *keras* libraries separately. We trained our models on three different extracted versions of dataset which are 120x120x3 rescaled dataset, 400x400x3 rescaled dataset, 720x2160x3 dataset. Additionally, to find best model, we tried different models, batch sizes, optimizers, and epochs.

In this report, one of the models which gives the best result is taken as reference. In this model, we trained 400x400x3 rescaled dataset by using *pytorch*. These following are the model parameters:

- 13 layers:

Layer 1: Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1))

Layer 2: Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1))

Layer 3: MaxPool2d(kernel_size=2, stride=2, padding=0)

Layer 4: Dropout(p=0.25, inplace=False)

Layer 5: Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1))

Layer 6: MaxPool2d(kernel_size=2, stride=2, padding=0)

Layer 7: Dropout(p=0.25, inplace=False)

Layer 8: Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1))

Layer 9: MaxPool2d(kernel_size=2, stride=2, padding=0)

Layer 10: Dropout(p=0.25, inplace=False)

Layer 11: Linear(in_features=36864, out_features=120, bias=True)

Layer 12: Dropout(p=0.5, inplace=False)

Layer 13: Linear(in_features=120, out_features=10, bias=True)

- Optimizer: SGD
- Activation functions: All ReLU
- Train-test ratio: 80% train, 20% test
- Number of epochs: 100
- Batch size: 20
- Learning rate: 0.001
- Criterion: Cross entropy loss

In this model, we preferred SGD optimizer instead of Adam optimizer. This is because SGD is better at generalizing even though Adam converges faster than SGD.

Secondly, we used ReLU activation functions instead of sigmoid functions because of computational efficiency. In this way, we also aimed to avoid vanishing gradient problems due to the number of layers. We did not use any additional softmax layers because the implementation of cross entropy loss, which is also called as softmax loss, already includes logistic.

The details of this model is given in *Appendix C*. The model summary of 2D CNN with *Keras* implementation is also available in *Appendix D*.

2.5 Long short-term memory (LSTM) with Mel Spectrogram Images

Recurrent Neural Network (RNN) is special type of network which helps training a dataset depending on the information of sequences. Long short-term memory networks are one types of Recurrent Neural Networks (RNN). LSTM basically helps learning longer sequences in a more efficient way compared to RNN.[3] We wanted to try RNN and LSTM because when we observe our mel spectrogram images, we noticed there might be some sequences for different genres that can be learned. We tried both implementing RNN and LSTM, We chose LSTM since it had better results. We used Pytorch with Google Colab to implement LSTM to train our dataset. Specifically, we used LSTM class from Pytroch.nn package. We used rescaled mel spectrogram dataset with dimensions 400x400x3. In order to find better models, we split our dataset into three parts which are training dataset, validation dataset, test dataset. By finding training dataset accuracy and validation dataset accuracy on every epoch and deriving information from these values, we tuned our LSTM model parameters.

As a result, our parameters and tools are as shown below:

- Batch size: 64
- Number of epochs: 120 (If our training accuracy exceeds 87%, we stop epoch iteration in order to avoid overfitting)
- Input size: 1200
- Step size: 400
- Hidden layer size: 400
- Layer dimension: 2
- Output dimension: 10
- Optimizer: Adam with learning rate 0.001
- 70% training dataset, 10% validation dataset, 20% test dataset (We are aware of the fact that sparing 10% of the dataset for validation could be inadequate; however, since we have only 1000 sample for dataset we wanted training dataset to be as much as possible.)

We selected Adam optimizer instead of SGD because SGD optimizer converged very slowly and required lots of epochs compared to Adam and also SGD optimizer had bigger oscillations. One other thing to note is that by experience and intuition, we can say that splitting an image row by row or column by column and feeding to the model as inputs at every step (size of the steps is equal to column pixel size if splitting is row by row, or row pixel size if splitting is column by column) was the best way to train LSTM for music genre classification by using mel spectrogram images obtained from GZAN music dataset. In between row by row or column by column splitting we could not find a huge difference; however, we chose column by column splitting since we had slightly better results.

2.6 Support Vector Machines with Mel Spectrogram Images

To implement SVM we used sklearn library from Python. We wanted to try Support Vector Machines (SVM) with Radial Basis Function(RBF) Kernel even though they are less complex model compared to our other models. We thought SVM with RBF kernel could give us good results. The reason behind our expectation was since we use mel spectrogram images with a lot of pixels as features, by using RBF kernel we could use samples (we only had 1000 sample) as features and reduce the downside of having pixels as features and maybe achieve good results. After trying different rescaled versions of mel spectrogram images with different parameters our best results was with the options below:

- Used rescaled mel spectrogram images with 360x120x3 dimensions
- C value: 100
- Gamma value: 2^-14
- 80% training dataset 20% test dataset

3. Results

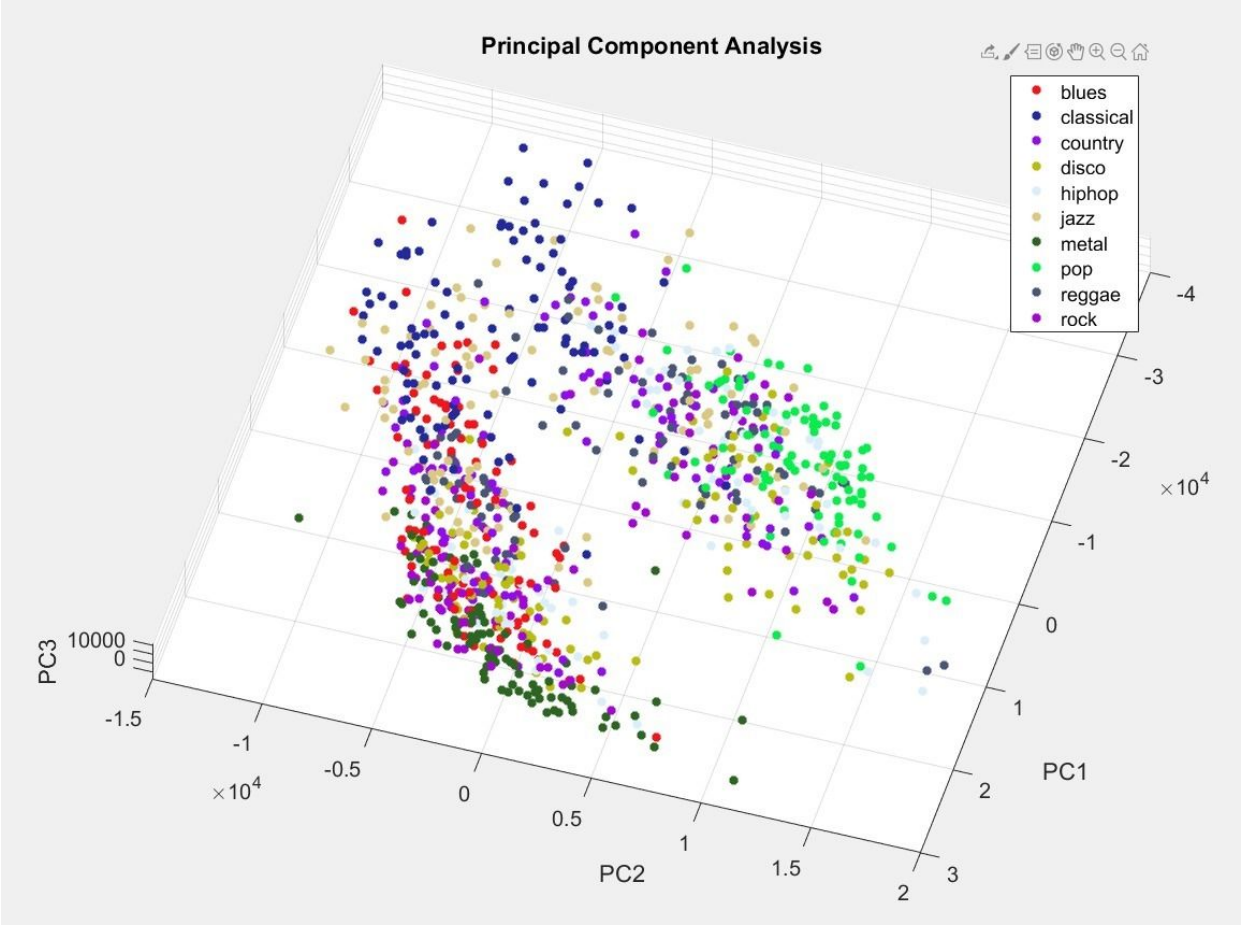


Figure 1. PCA Application on 400x400x3 Mel Spectrogram Images
PVE: 51.56% (Other perspectives of PCA is also available in Appendix E)

3.1. 1D Conv with Raw Waveform

As it is written in the methods section, to train 1D CNN, we used raw waveforms which have 551.250 features for each sample. While we are training model, we test model on validation set to find the best parameters for training model. The following 2 graphs show validation accuracy, training accuracy and training loss values.

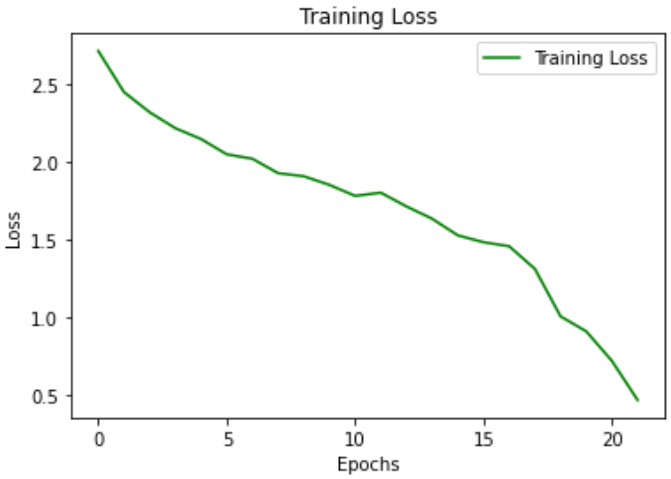


Figure 2. Training Loss vs. Number of Epochs in 1D CNN with Raw Waveform

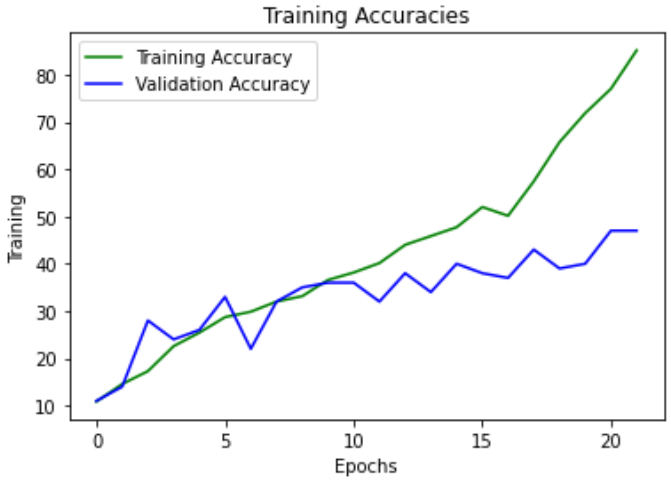
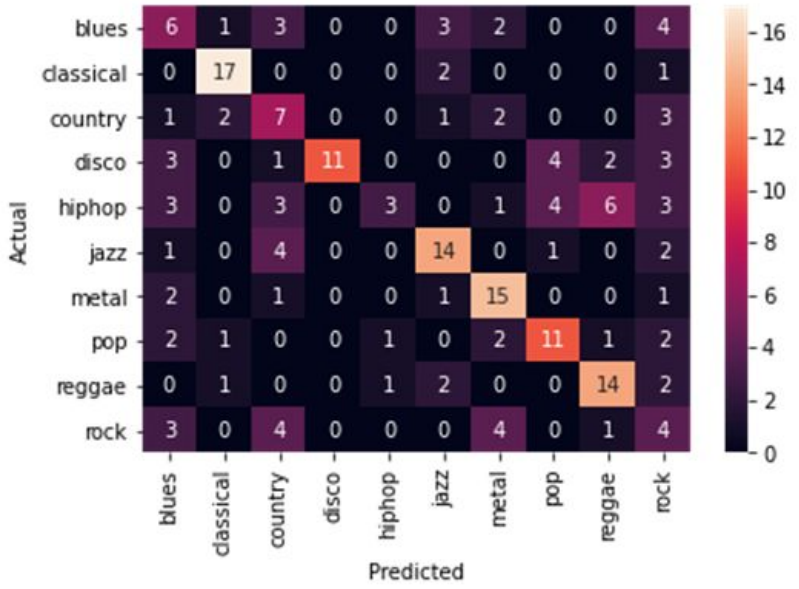


Figure 3. Training & Validation Set Accuracies vs. Number of Epochs in 1D CNN with Raw Waveform

As it can be seen from Figure 2, our model is learning while number of epoch is increasing. Although we set 100 epochs for this model, it reaches 90% training accuracy rate at 22th epoch. We stop to train the model at this point because we want from model to be generalizable, not to overfit to training dataset.

From Figure 3, it can be seen that while training accuracy is increasing, also validation accuracy is increasing with it. However, after %40 training accuracy, validation accuracy is starting to be flatten compared training accuracy. It shows that our model starts to fit training data.

Table 1. Confusion Matrix of Test Dataset for 1D CNN with raw waveforms



	precision	recall	f1-score	support
blues	0.29	0.32	0.30	19
classical	0.77	0.85	0.81	20
country	0.30	0.44	0.36	16
disco	1.00	0.46	0.63	24
hiphop	0.60	0.13	0.21	23
jazz	0.61	0.64	0.62	22
metal	0.58	0.75	0.65	20
pop	0.55	0.55	0.55	20
reggae	0.58	0.70	0.64	20
rock	0.16	0.25	0.20	16
accuracy			0.51	200
macro avg	0.54	0.51	0.50	200
weighted avg	0.57	0.51	0.51	200

```
Accuracy for blues is: 0.2857142857142857

Accuracy for classical is: 0.7727272727272727

Accuracy for country is: 0.30434782608695654

Accuracy for disco is: 1.0

Accuracy for hiphop is: 0.6

Accuracy for jazz is: 0.6086956521739131

Accuracy for metal is: 0.5769230769230769

Accuracy for pop is: 0.55

Accuracy for reggae is: 0.5833333333333334

Accuracy for rock is: 0.16
```

Top 1 accuracy: 51%

Top 2 accuracy: 65%

Top 3 accuracy: 79%

By looking confusion metric and class based accuracies, we can see that 1D CNN model with raw waveforms is good at predicting classical, jazz, metal and reggae songs. By looking at Top 2 and Top 3 accuracies, we can say that even though the classifier cannot predict a genre correctly, it includes the correct genre in its top 3 prediction with %79 accuracy.

Although, the model gives 51% accuracy for this time, it can be changed according to how we splitted data into training and test sets. Because, our dataset consists of 1000 songs which can be small for training. Since our dataset is small, effect of distribution of each song is large for training and testing.

3.2. 1D Conv with Mel Spectrogram

We used 400x400x3 rescaled dataset in 1D CNN with mel spectrogram model as we mentioned it before. We tried different parameters for each function and different optimizers, layers. The best result was Adam optimizer for our trials in terms of accuracy and speed. We have tried also different batch sizes, pooling sizes to catch the best results.

The following results includes training accuracy, validation accuracy and training loss.

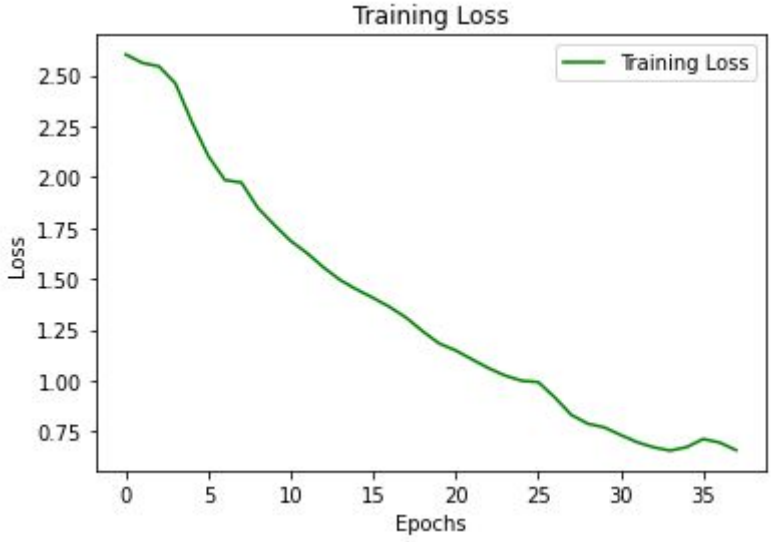


Figure 4. Training Loss vs. Number of Epochs in 1D CNN with Mel Spectrogram

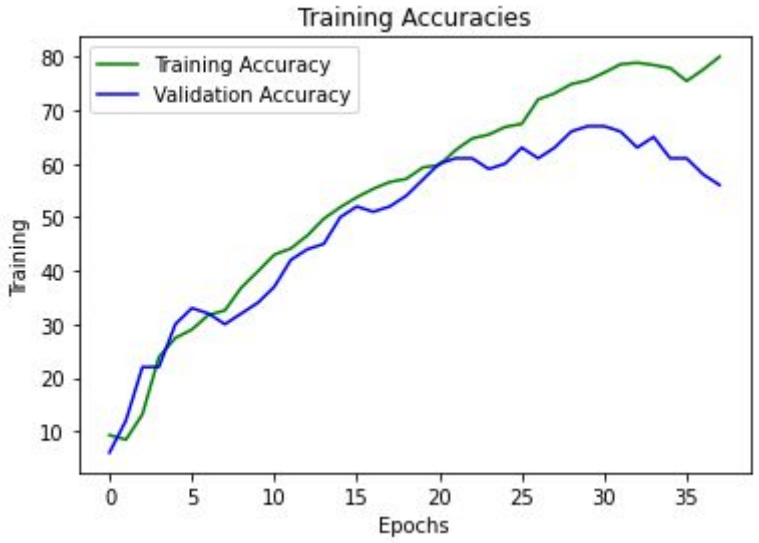
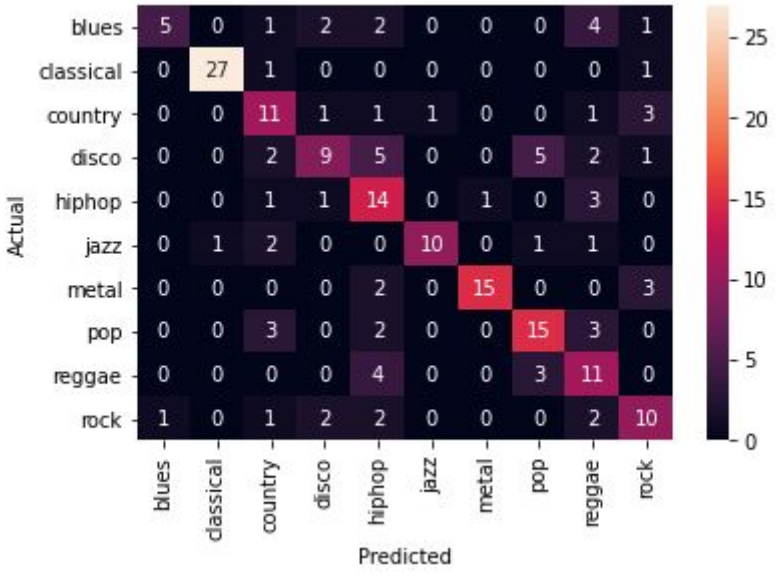


Figure 5. Training & Validation Set Accuracies vs. Number of Epochs in 1D CNN with Mel Spectrogram

As it can be seen at both Figure 4 and 5, the training is stopped before 40th epoch when training accuracy exceeds 80% to prevent overfitting. If the training accuracy would not exceed 80%, training would stop at 100th epoch. Also, we tried it 90% at first, but our model would approach overfitting and accuracy for test dataset would not be near to each other. The decrease in this limitation provided faster and better results. On the other hand, as it can be seen at Figure 5, even though training accuracy reaches 80%, validation accuracy locked between 65-55%, it shows that our model is fitting the training data.

The following table includes confusion matrix of our test:

Table 2. Confusion Matrix of 1D CNN with Mel Spectrogram



	precision	recall	f1-score	support
blues	0.83	0.33	0.48	15
classical	0.96	0.93	0.95	29
country	0.50	0.61	0.55	18
disco	0.60	0.38	0.46	24
hiphop	0.44	0.70	0.54	20
jazz	0.91	0.67	0.77	15
metal	0.94	0.75	0.83	20
pop	0.62	0.65	0.64	23
reggae	0.41	0.61	0.49	18
rock	0.53	0.56	0.54	18
accuracy			0.64	200
macro avg	0.67	0.62	0.62	200
weighted avg	0.68	0.64	0.64	200

```
Top 1 prediction accuracy: 127/200 (64%)
Top 2 prediction accuracy: 79.0%
Top 3 prediction accuracy: 86.0%
```

It can be seen at Table 2 as top 1, top 2 and top 3 genre prediction accuracies are 64%, 79%, 86% respectively. Also, classical, jazz, metal songs can be distinguished by our model more than other classes.

The results can differ for each execution because of shuffle and size of our dataset. Shuffle process affects our data critically. We generally obtained between 50-66% accuracy with our model.

3.3 2D CNN with Mel Spectrogram

We trained our 2D CNN model on 400x400x3 rescaled dataset with the parameters explained in the section 2.3. For 100 epochs, we obtained the following losses and accuracies:

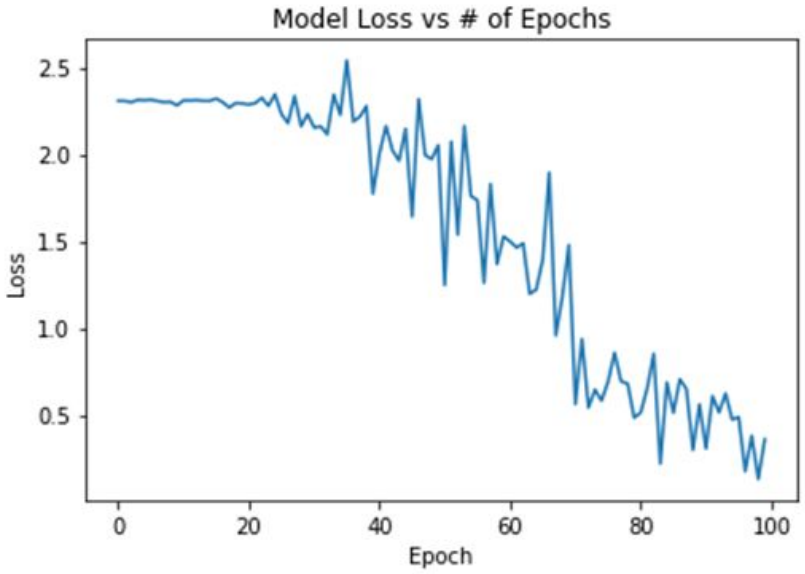


Figure 6. Training Loss vs. Number of Epochs in 2D CNN

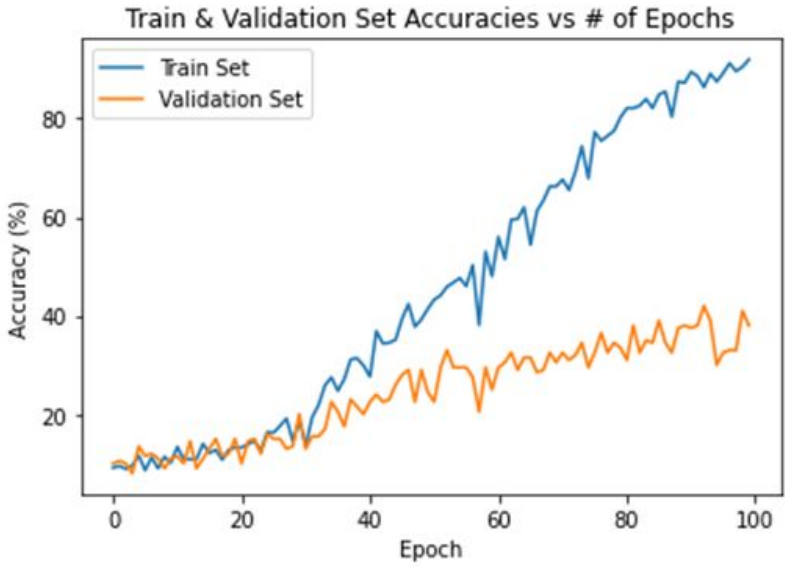


Figure 7. Training & Validation Set Accuracies vs. Number of Epochs in 2D CNN

As it can be seen from Figure 6, the loss is decreasing significantly while number of epochs is increasing. This means that the model was trained in 100 epochs and it learnt the training set. However, the success of the model is determined by the accuracies on validation set.

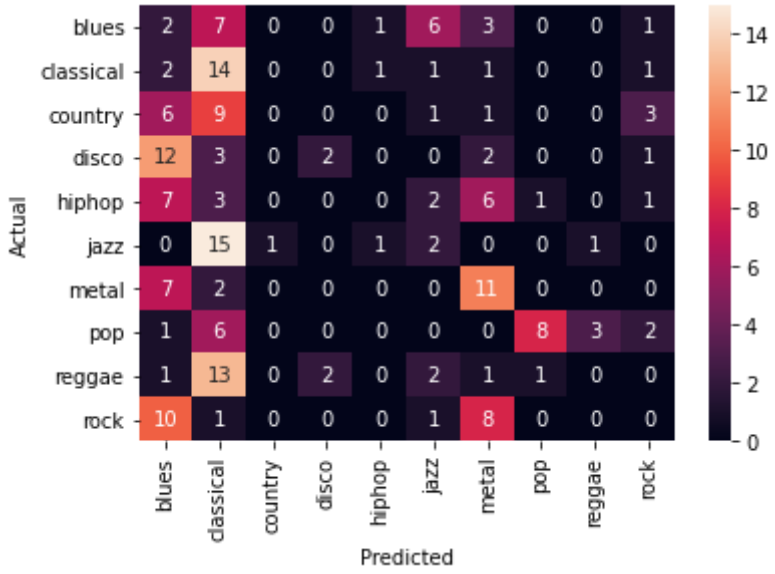
In Figure 7, the maximum accuracy on the validation set is about 40%. Until about the first 50 epochs, the validation set accuracy is increasing in accordance with the

training set accuracy. After the first 50 epochs, the difference between the training set accuracy and the validation set accuracy is increasing dramatically. Hence, overfitting occurs in this model.

There can be several reasons of overfitting such as inadequate number of data and complexity of model. To decrease the effect of overfitting, we added dropout layers and also, we tried different parameters. However, this is one of the best results we could obtain. We think the model could give better results if we would have more data. There are 800 data for training and 10 classes. To learn one class, the model has only 80 data and this may not be adequate.

To observe training success of the model, we compared performances in the epoch 25 and after training is finished.

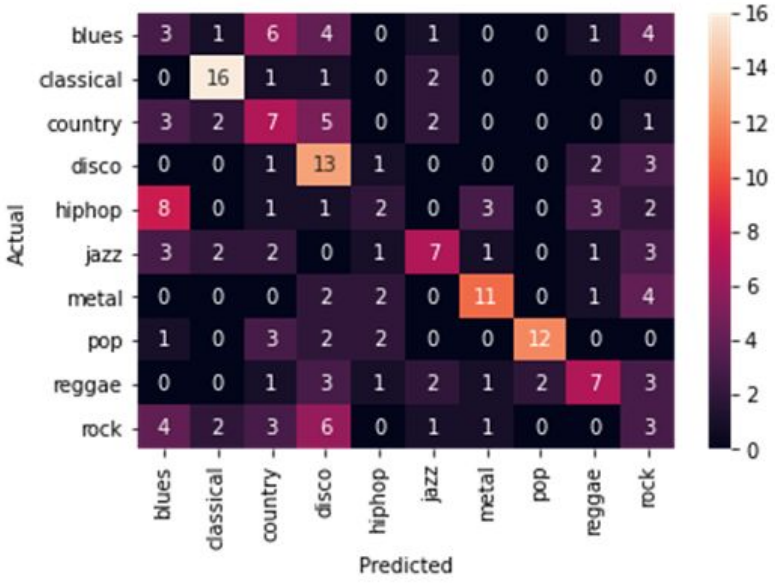
Table 3. Confusion Matrix of Test Dataset for 2D CNN with 25 Epoch Training



	precision	recall	f1-score	support
blues	0.04	0.10	0.06	20
classical	0.19	0.70	0.30	20
country	0.00	0.00	0.00	20
disco	0.50	0.10	0.17	20
hiphop	0.00	0.00	0.00	20
jazz	0.13	0.10	0.11	20
metal	0.33	0.55	0.42	20
pop	0.80	0.40	0.53	20
reggae	0.00	0.00	0.00	20
rock	0.00	0.00	0.00	20
accuracy			**0.20**	200
macro avg	0.20	0.20	0.16	200
weighted avg	0.20	0.20	0.16	200

```
Accuracy for blues is: 0.04
Accuracy for classical is: 0.19
Accuracy for country is: 0.0
Accuracy for disco is: 0.5
Accuracy for hip hop is: 0.0
Accuracy for jazz is: 0.13
Accuracy for metal is: 0.33
Accuracy for pop is: 0.8
Accuracy for reggae is: 0.0
Accuracy for rock is: 0.0
```

Table 4. Confusion Matrix of Test Dataset for 2D CNN after 100 Epoch Training



	precision	recall	f1-score	support
blues	0.14	0.15	0.14	20
classical	0.70	0.80	0.74	20
country	0.28	0.35	0.31	20
disco	0.35	0.65	0.46	20
hiphop	0.22	0.10	0.14	20
jazz	0.47	0.35	0.40	20
metal	0.65	0.55	0.59	20
pop	0.86	0.60	0.71	20
reggae	0.47	0.35	0.40	20
rock	0.13	0.15	0.14	20
accuracy			**0.41**	200
macro avg	0.43	0.40	0.40	200
weighted avg	0.43	0.41	0.40	200

```
Accuracy for blues is: 0.14
Accuracy for classical is: 0.70
Accuracy for country is: 0.28
Accuracy for disco is: 0.35
Accuracy for hip hop is: 0.22
```

```
Accuracy for jazz is: 0.47
Accuracy for metal is: 0.65
Accuracy for pop is: 0.86
Accuracy for reggae is: 0.47
Accuracy for rock is: 0.13

Top 2 prediction accuracy: 59%
Top 3 prediction accuracy: 74%
```

If these two tables are compared, we can observe that the model has shown a greater success than 50% in estimating classical, metal, and pop music. However, it couldn't be trained successfully to estimate especially blues, hiphop, and rock. These results are also coherent with PCA of the dataset. We have observed classical and pop can be classified with ease compared to other genres.

After the training, the model has 41% accuracy on the test set. However, top-3 accuracy is higher than 70%. This result also support that model can be improved by using more data.

3.4 Long short-term memory (LSTM) with Mel Spectrogram Images

We have trained our LSTM network with the parameters given in the previous section. In order to find best model we can, we splitted dataset into training, validation and test. Nevertheless, since we have only 1000 data sample, we trained our data without using validation dataset after we found the best model we can. We have recorded the training accuracy, validation accuracy and training loss. We have obtained the following results:

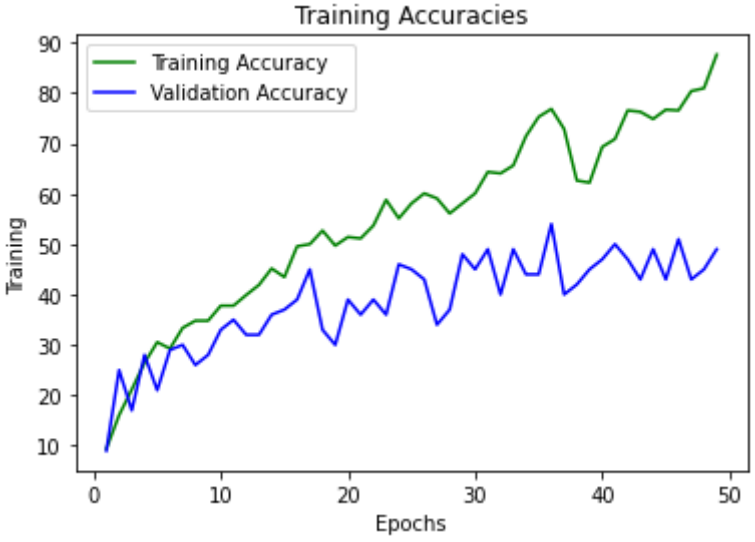


Figure 8. Training & Validation Set Accuracies vs. Number of Epochs in LSTM

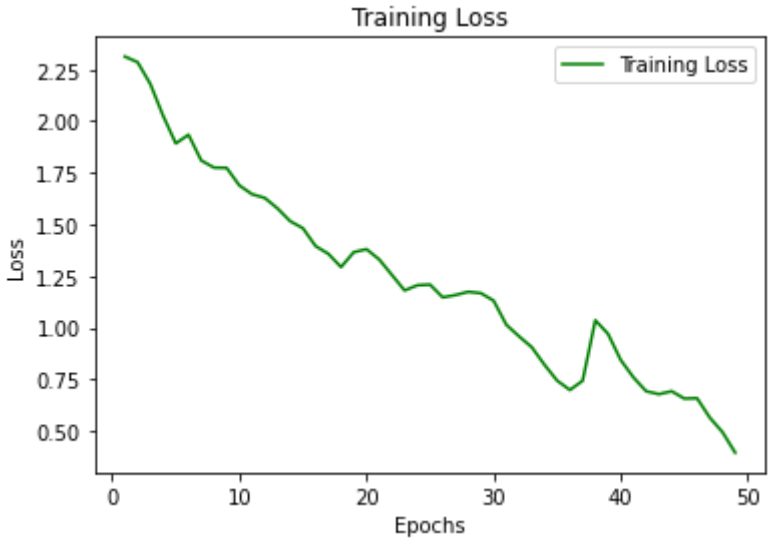
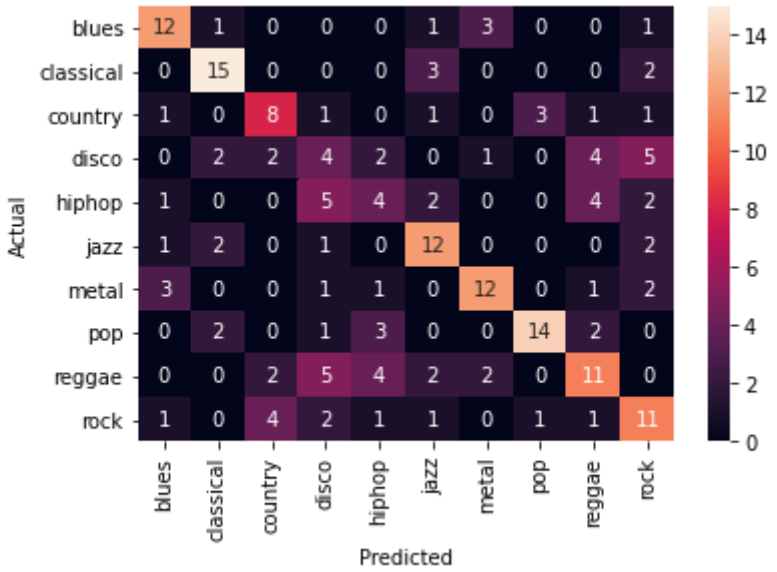


Figure 9. Training Loss vs. Number of Epochs in LSTM

By looking at Figure 8 it can be stated that as our model learns predicting training dataset, it slows down learning predicting values other than training dataset. One of the reasons for this can be that our dataset sample size is inadequate. Moreover, it can have noisy data as we can infer from PCA analysis at Figure 1. Lastly, maybe we could find better parameters for our model, but this was the best we can achieve at the moment.

After training the model with 70% training dataset, the performance metrics results for our test dataset is as shown as below:

Table 5. Confusion Matrix for Test Dataset of LSTM with 70% Training Dataset



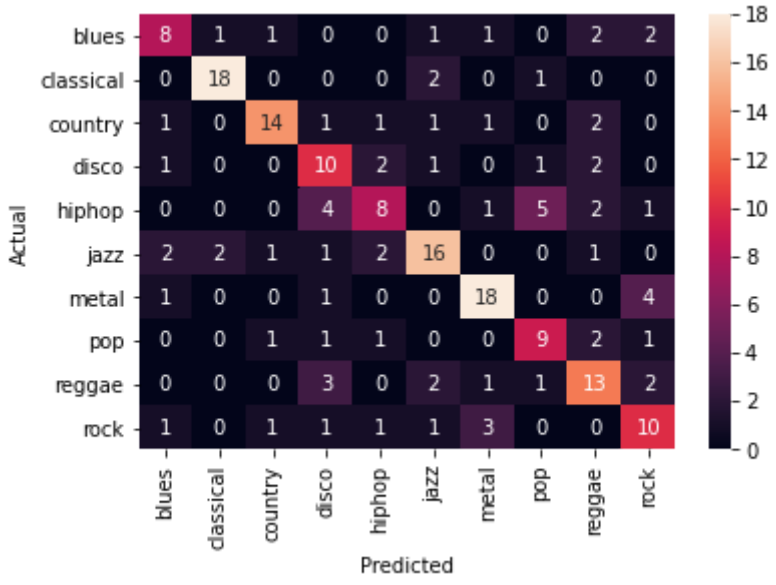
	precision	recall	f1-score	support
blues	0.63	0.67	0.65	18
classical	0.68	0.75	0.71	20
country	0.50	0.50	0.50	16
disco	0.20	0.20	0.20	20
hiphop	0.27	0.22	0.24	18
jazz	0.55	0.67	0.60	18
metal	0.67	0.60	0.63	20
pop	0.78	0.64	0.70	22
reggae	0.46	0.42	0.44	26
rock	0.42	0.50	0.46	22
accuracy			**0.52**	200
macro avg	0.52	0.52	0.51	200
weighted avg	0.52	0.52	0.51	200

```
Accuracy for blues is: 0.63
Accuracy for classical is: 0.68
Accuracy for country is: 0.5
Accuracy for disco is: 0.2
Accuracy for hip hop is: 0.26
Accuracy for jazz is: 0.54
Accuracy for metal is: 0.66
Accuracy for pop is: 0.77
Accuracy for reggae is: 0.45
Accuracy for rock is: 0.42
```

From these results we can say that best classes we learned to predict are classical, pop, and blues; the classes we could not exactly learn to predict are disco and hip hop.

In order to have a better model we trained our model again without separating a validation dataset with same parameters. The performance metrics results for our test dataset is as shown as below:

Table 6. Confusion Matrix for Test Dataset of LSTM with 80% Training Dataset:



	precision	recall	f1-score	support
blues	0.57	0.50	0.53	16
classical	0.86	0.86	0.86	21
country	0.78	0.67	0.72	21
disco	0.45	0.59	0.51	17
hiphop	0.53	0.38	0.44	21
jazz	0.67	0.64	0.65	25
metal	0.72	0.75	0.73	24
pop	0.53	0.60	0.56	15
reggae	0.54	0.59	0.57	22
rock	0.50	0.56	0.53	18
accuracy			**0.62**	200
macro avg	0.62	0.61	0.61	200
weighted avg	0.63	0.62	0.62	200

```
Accuracy for blues is: 0.5714285714285714
Accuracy for classical is: 0.8571428571428571
Accuracy for country is: 0.7777777777777778
Accuracy for disco is: 0.45454545454545453
Accuracy for hiphop is: 0.5333333333333333
Accuracy for jazz is: 0.6666666666666666
Accuracy for metal is: 0.72
Accuracy for pop is: 0.5294117647058824
Accuracy for reggae is: 0.5416666666666666
Accuracy for rock is: 0.5
```

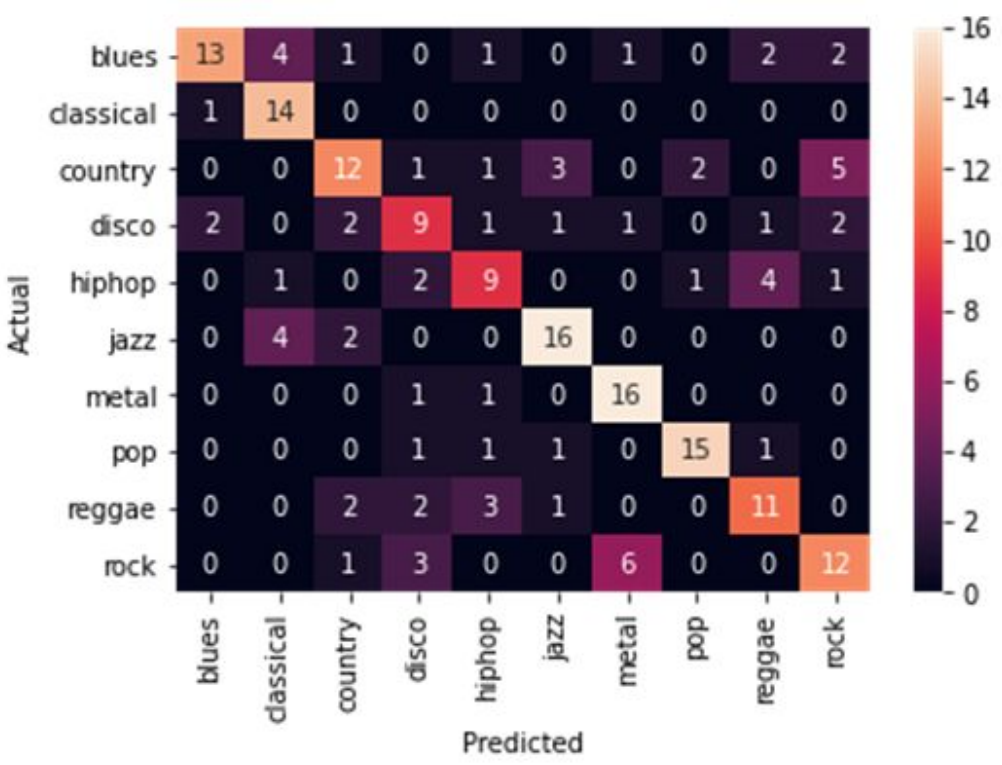
By looking at the results above we can derive that as we guessed earlier having more samples tends to give better results. We still have problems with some classes such as hip hop. Thus we also found the accuracies for predicting top 2 class and top 3 class. The results are as shown below:

```
Top 2 prediction accuracy: 78%
Top 3 prediction accuracy: 85%
```

3.5 Support Vector Machines with Mel Spectrogram Images

We have trained our SVM with RBF kernel by using the parameters given in the previous section. The performance metrics results for our test dataset is as shown as below:

Table 7. Confusion Matrix for Test Dataset of SVM with RBF kernel



	precision	recall	f1-score	support
blues	0.81	0.54	0.65	24
classical	0.61	0.93	0.74	15
country	0.60	0.50	0.55	24
disco	0.47	0.47	0.47	19
hiphop	0.53	0.50	0.51	18
jazz	0.73	0.73	0.73	22
metal	0.67	0.89	0.76	18
pop	0.83	0.79	0.81	19
reggae	0.58	0.58	0.58	19
rock	0.55	0.55	0.55	22
accuracy			**0.64**	200
macro avg	0.64	0.65	0.63	200
weighted avg	0.64	0.64	0.63	200

Accuracy for blues is: 0.8125
Accuracy for classical is: 0.6086956521739131
Accuracy for country is: 0.6
Accuracy for disco is: 0.47368421052631576
Accuracy for hiphop is: 0.5294117647058824
Accuracy for jazz is: 0.7272727272727273
Accuracy for metal is: 0.6666666666666666
Accuracy for pop is: 0.8333333333333334
Accuracy for reggae is: 0.5789473684210527
Accuracy for rock is: 0.5454545454545454

From these results we can say that best classes we learned to predict are classical, metal, and pop; the classes we could not exactly learn to predict are disco, hip hop and reggae. One interesting thing to note is this model occasionally predicts rock as metal and as a human we cannot differentiate the two always.

Another interesting result which we encountered when we trained SVM with RBF kernel, we had better results when we used rescaled data with 360x120x3 dimension in comparison to with 400x400x3 dimensions (See Appendix-B). We believe that one reason for this could be when we used lower dimension data, the noise was lower also, so we had better results. Another reason could be that we didn't find best parameters for the data with 400x400x3 dimensions, but we highly doubt that.

4. Discussion

By gathering all of the top 1, top 2 and top 3 accuracies, we can compare different models we have used for classification of music genres.

By looking at top 1 accuracies from previous sections, we can see that 1D CNN with MSI, LSTM and SVM are good at predicting music genres correctly compared to 2D CNN and 1D CNN with Raw waveform. Since the audio data is a time domain 1D data, we expected better accuracies from 1D CNN and LSTM models. Because each pixel on mel spectrogram image and each feature on audio wave is strongly related with its neighboring points, these models are good at finding relation between these points. However, we did not expect SVM to give close results these methods, but surprisingly it gives close accuracy rate to 1D CNN and LSTM models. Moreover, although top 2 and top 3 accuracies are higher than top 1 accuracy rates, performance of models do not change with respect to each other.

When we look at performances of each model, we found maximum accuracy as 64% with SVM and 1D CNN, and we found minimum accuracy as 41% with 2D CNN. When we think the class size, which is 10, our accuracies are not small. Also, we have small number of data samples for each genre, we think that if we can enlarge our dataset, we expect to see higher accuracies with these models. Because, in this dataset, even small changes on distribution of music files into train and test sets may affect the performance of models.

Also, when we look at class based F1 scores of music genres, some of music genres have more distinguishable features than other genres. For example, classical and metal music genres have highest F1 scores for each classifier. So, every classifier that we trained are good at predicting these classes and it shows that these genres are more distinguishable. However, some genres have lowest F1 score such as hip hop or disco. For each method that we trained, these genres give low F1 score. It might show that, our classifiers has difficulty predicting these genres. We think that

these genres might have common many features with other genres or our data size is not large enough to distinguish these genres from other genres.

5. Conclusion

In this project, we aimed to classify music tracks by their genres. We have used GTZAN Genre Collection dataset that includes 1000 music tracks having 30 second samples for 10 different genres. Music genre classification algorithms can be used by online music platforms and music stores for organizing different music tracks and suggesting users music tracks from predefined genres.

In order to find the suitable music classification methods, we have contacted our TA and decided to use the following methods: 1D Convolutional Neural Network (1D CNN), 2D Convolutional Neural Network (2D CNN), Long Short-Term Memory (LSTM) and SVM. After preprocessing of the dataset, we have trained the dataset on these methods and revised continuously in order to get higher accuracy results. We have also obtained top 1, top 2 and top 3 accuracies. The methods, accuracy results and the reasons why we might have obtained these accuracies and results were further discussed in their corresponding parts in this report. The workload is further discussed in Appendix-A.

With such results and such an algorithm, we can further classify new music files, music samples in online streaming platforms by their genres.

6. References

1- Librosa - Feature Extraction. (n.d.). Retrieved from https://librosa.github.io/librosa/feature.html

2- Pandey, P. (2018, December 19). Music Genre Classification with Python. Retrieved from https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8

3-"Empirical Evaluation of Gated Recurrent Neural Networks on ...." 11 Dec. 2014, https://arxiv.org/abs/1412.3555. Access date: 10 May. 2020.

4- PyTorch Documentation. (n.d.). Retrieved from https://pytorch.org/docs/stable/

5- Li, H., Xue, S., & Zhang, J. (2019). *Combining CNN and Classical Algorithms for Music Genre Classification*. Retrieved from http://cs229.stanford.edu/proj2018/report/19.pdf

6- Synced. (2019, March 7). ICLR 2019: 'Fast as Adam & Good as SGD' - New Optimizer Has Both. Retrieved from https://medium.com/syncedreview/iclr-2019-fast-as-adam-good-as-sgd-new-optimizer-has-both-78e37e8f9a34

7. Appendix

Appendix-A: Work Division

Doğukan Köse: I extracted waveform data, and mel-spectrogram images from audio files. I rescaled them according to test accuracies of my teammates. I also worked on training 1D CNN with raw waveforms with pytorch library. Also, I helped Fatih have better accuracy on 1D CNN with mel-spectrogram images.

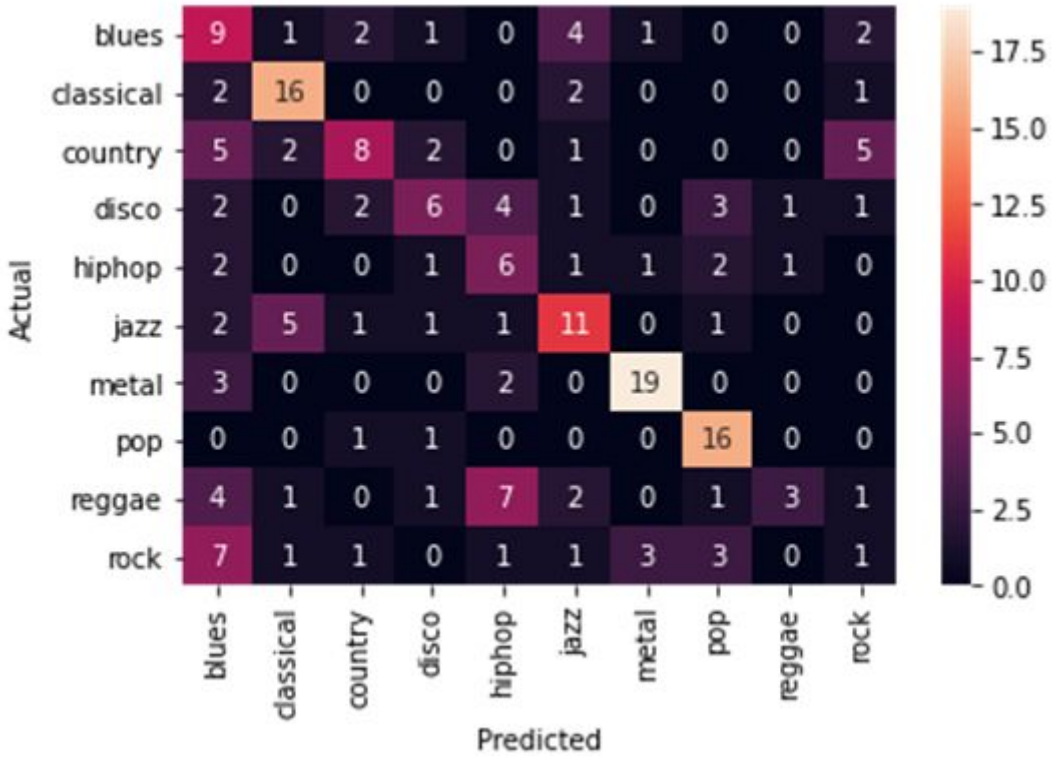
Meryem Efe: I trained 2D CNN on mel-spectrogram images by using Pytorch. I have tried different models on 3 different extracted versions of dataset to reach maximum accuracy. Also, I compared my results with the results Nehir obtained, then I analyzed and reported 2D CNN results.

Muhammed Musab Okşaş: I trained RNN and LSTM with rescaled mel-spectrogram images, but I only put the results of LSTM. I also trained SVM with RBF kernel with rescaled mel-spectrogram images. I applied PCA to rescaled mel-spectrogram images. I wrote a module to print performance metrics when predictions and labels given as parameters, which is used by me and my teammates during project.

Fatih Çakır: I worked on 1D CNN with mel-spec images with Doğukan to reach max accuracy which we could. I have tried this with every dataset which are extracted with my friends. I have used Pytorch library.

Nehir Kırtaş: I trained 2D CNN on mel-spectrogram images and two other extracted versions of the dataset. I have only put mel-spectrogram 2D CNN results. I have trained the dataset by using Keras. In addition, I compared my results with the results Meryem had obtained and decided to use the results having the most accuracy. I also have written the introduction and conclusion parts of the report.

Appendix-B: SVM with RBF kernel with using 400x400x3 reshaped mel-spectrogram



	precision	recall	f1-score	support
blues	0.25	0.45	0.32	20
classical	0.62	0.76	0.68	21
country	0.53	0.35	0.42	23
disco	0.46	0.30	0.36	20
hiphop	0.29	0.43	0.34	14
jazz	0.48	0.50	0.49	22
metal	0.79	0.79	0.79	24
pop	0.62	0.89	0.73	18
reggae	0.60	0.15	0.24	20
rock	0.09	0.06	0.07	18
accuracy			0.48	200
macro avg	0.47	0.47	0.44	200
weighted avg	0.49	0.47	0.46	200

Appendix-C: Summary of 2D CNN Model with Pytorch

```
----------------------------------------------------------------
        Layer (type)               Output Shape       Param #
================================================================
            Conv2d-1         [-1, 16, 398, 398]             448
            Conv2d-2         [-1, 16, 396, 396]           2,320
         MaxPool2d-3         [-1, 16, 198, 198]               0
           Dropout-4         [-1, 16, 198, 198]               0
            Conv2d-5         [-1, 16, 196, 196]           2,320
         MaxPool2d-6           [-1, 16, 98, 98]               0
           Dropout-7           [-1, 16, 98, 98]               0
            Conv2d-8           [-1, 16, 96, 96]           2,320
         MaxPool2d-9           [-1, 16, 48, 48]               0
          Dropout-10           [-1, 16, 48, 48]               0
           Linear-11                 [-1, 120]       4,423,800
          Dropout-12                 [-1, 120]               0
           Linear-13                  [-1, 10]           1,210
================================================================
Total params: 4,432,418
Trainable params: 4,432,418
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.83
Forward/backward pass size (MB): 56.77
Params size (MB): 16.91
Estimated Total Size (MB): 75.51
----------------------------------------------------------------
```

Appendix-D: Summary of 2D CNN Model with Keras

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 396, 996, 8)       608
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 198, 498, 8)       0
_________________________________________________________________
dropout (Dropout)            (None, 198, 498, 8)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 194, 494, 8)       1608
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 97, 247, 8)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 97, 247, 8)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 95, 245, 16)       1168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 47, 122, 16)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 45, 120, 32)       4640
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 22, 60, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 42240)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              43254784
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 43,273,058
Trainable params: 43,273,058
Non-trainable params: 0
_________________________________________________________________
```

Appendix-E: PCA WITH DIFFERENT PERSPECTIVE