FINAL REPORT

Music Genre Classification

Group 96

So Hee Yoon	1004327604
Ho Seok (David) Lee	1004112177
Myeong Hun (David) Song	1004815961
Ting Wei (Sherman) Lin	1004835413

Word count: 2251

1. Introduction

Classifying music by ear is a fairly trivial human task; you can show a child one of Chopin’s études and they will be able to recognize it as a classical piece. Formulating explicit musical criteria that defines a classical piece, however, is very difficult and requires detailed descriptions of musical properties such as instrumentation, tempo, and arrangement. Even with these descriptions, identification of a genre without listening to the music can become ambiguous. Despite these shortcomings, we observed that these explicit musical criteria reflect a pattern-like consistency within genres that machines can possibly learn to detect.

An image recognition algorithm can detect a dog by recognizing features that are distinctive of a canine. These machine-learned features, however, cannot easily be translatable for humans to understand. In the same manner, algorithms can learn some mechanized version of those explicit musical criteria.

An algorithm that can classify music is extremely helpful for music platforms such as Spotify or Apple Music. Users are more inclined to search for and explore through genres instead of groupings like “songs with guitars”, since these are classifications that can not only be easily understood by humans, but are often reflective of one’s music preference. Considering all of these factors, we believe that a machine that can recognize genres is both viable and valuable.

2. Illustration / Figure

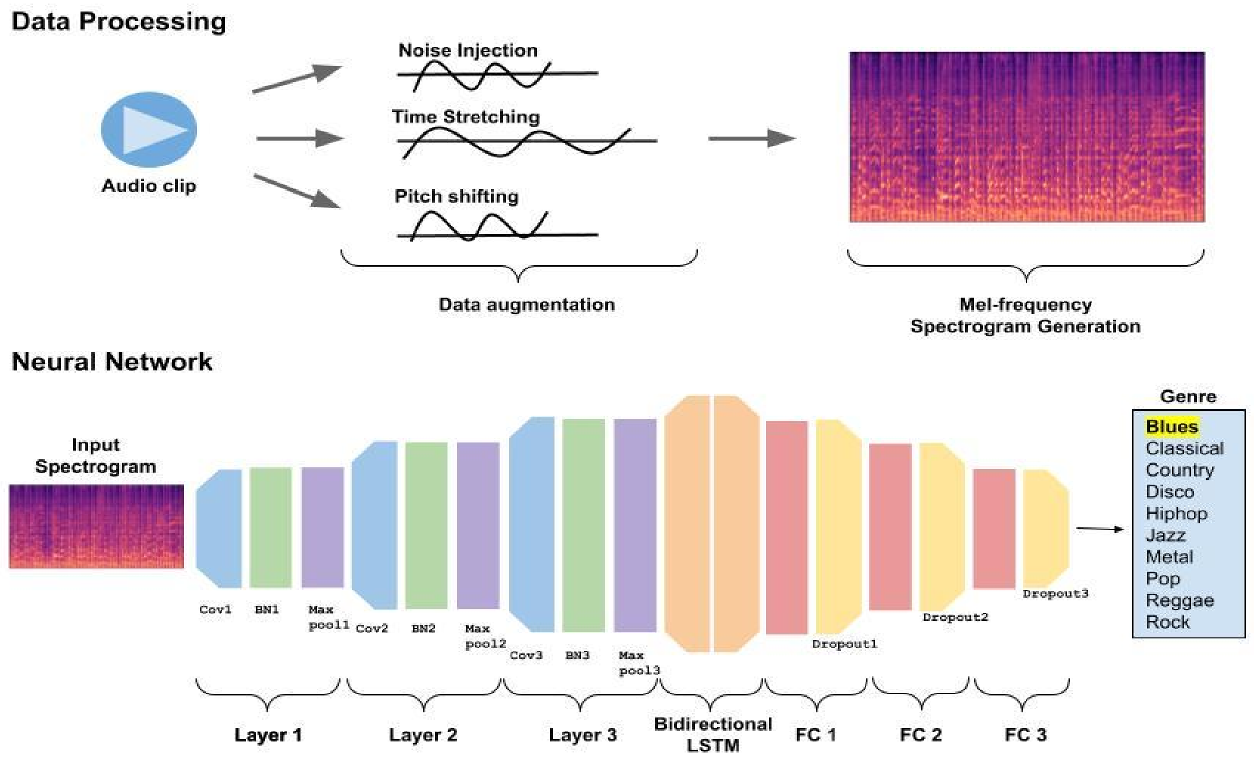


Figure 1: Illustration showing the core idea and architecture of the project

3. Background & Related Work

Analyzing lyrics and chords to classify music genre *by Timothy Greer and Shrikanth Narayanan [1]*

This paper uses natural language processing to classify music from the Billboard charts and represent the relationship between chord progression and lyric sequence in predicting genres.

Music genre classification *by Derek A. Huang [2]*

Convolutional Neural Networks are used to predict genres of the music, out of 10 distinct music genres. The model inputs two different data: raw amplitude data and their transformed mel-spectrograms to determine which input resulted in a higher accuracy.

4. Data Processing

Source of data

We decided to use the GTZAN genre dataset [3], which feature clips of music (of file type .wav) of 30 seconds in length of categorized genres: 10 genres with 100 audio clips in each. Data augmentation tripled the dataset by generating three new clips from each through noise injection, pitch shift, and time stretching.

Mel-Spectrogram

1. Load audio file using (librosa.load)
2. Create mono audio signal (librosa.to_mono)
3. Generate mel spectrogram (librosa.feature.melspectrogram)
4. Convert power spectrogram into decibel units (librosa.power_to_db)
5. Save figures as image file in the genre folder
6. Resize the 1200*800 pixel image to 240*160 pixel

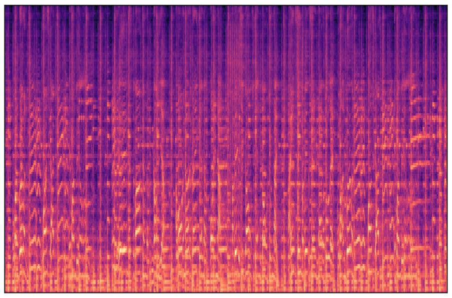


Figure 2: Example of generated spectrogram from audio file in blues genre.

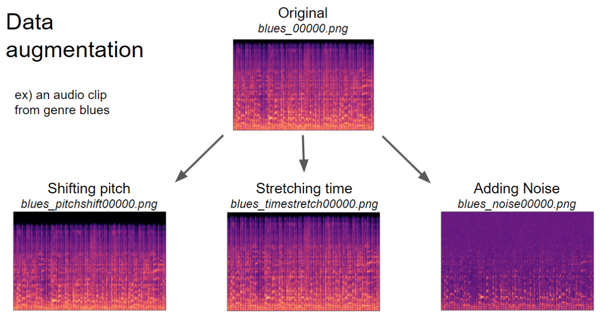
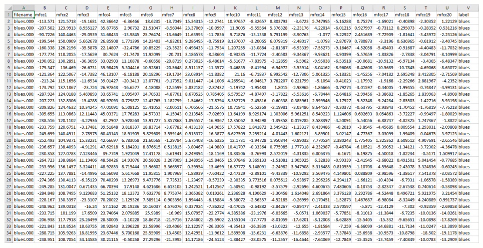


Figure 3: Spectrograms generated from an audio clip through data augmentation.

Mel-Frequency Cepstrum Coefficient

1) Load audio file using (librosa.load)
2) Extract mfcc features (librosa.feature.mfcc)
3) Save mfcc values into data.csv file



filename	mfcc1	mfcc2	mfcc3	mfcc4	mfcc5	mfcc6	mfcc7	mfcc8	mfcc9	mfcc10	mfcc11	mfcc12	mfcc13	mfcc14	mfcc15	mfcc16	mfcc17	mfcc18	mfcc19	mfcc20	label
blues.000	-113.571	121.5718	-19.1681	42.36642	-6.36466	18.6235	-13.7049	15.34315	-12.2741	10.97657	-8.32657	8.803793	-3.6723	5.747995	-5.16288	0.75274	-1.69021	-0.40898	-2.30352	1.22129	blues
blues.000	-207.502	123.9913	8.955127	35.87765	2.90732	21.51047	-8.56044	23.37069	-10.0997	11.9005	-5.55564	5.376328	-2.23176	4.22014	-6.01215	0.927997	-0.73112	0.295073	-0.28352	0.531216	blues
blues.000	-90.7226	140.4463	-29.0939	31.68433	-13.9845	25.76474	-13.6649	11.63993	-11.7836	9.71876	-13.1338	5.791199	-8.90763	-1.077	-9.22927	2.451689	-7.72909	-1.81641	-3.43972	-2.23126	blues
blues.000	-199.544	150.0909	5.662678	26.85908	1.771399	14.23403	-4.83201	9.286495	-0.75919	8.137607	-3.20065	6.079319	-2.48017	-1.0793	-2.87079	0.780873	-3.3196	0.636965	-0.61912	-3.40745	blues
blues.000	-160.338	126.2196	-35.5878	22.14807	-32.4786	10.85229	-23.3523	0.498433	-11.7934	1.207255	-13.0884	-2.81387	-6.93339	-7.55273	-9.16467	-4.52058	-5.45403	-0.91687	-4.40483	-11.7032	blues
blues.000	-177.774	118.2055	-17.5659	30.7624	-21.7478	11.92099	-20.731	3.186578	-8.58604	-0.93285	-11.7724	-2.40583	-9.34367	-9.93621	-3.90399	-5.57659	-1.83826	-2.7838	-3.04791	-8.10999	blues
blues.000	-190.052	130.2891	-36.3695	33.02903	11.10878	-0.60558	-20.8719	0.273025	-6.48614	-5.51677	-7.83975	-3.12859	-6.5962	-9.95038	-6.53518	-10.0681	-10.9132	-6.97134	-3.4365	-6.48347	blues
blues.000	-179.347	136.469	-26.6731	39.98425	5.304416	10.92861	-20.5648	8.511157	-11.3572	-3.46835	-8.41994	-6.94972	-3.53914	-8.04162	-8.96068	-8.42608	-10.5689	-10.7845	-4.69068	-8.63072	blues
blues.000	-121.364	122.5067	-14.7382	46.13107	-8.18188	20.18296	-19.1734	23.03914	-11.8382	21.16	-6.71837	6.992542	-12.7306	5.061325	-5.18321	-1.45256	-7.04182	2.695248	3.412305	-2.71569	blues
blues.000	-213.24	115.1656	-11.6934	39.01427	-20.3413	13.07761	-9.17352	9.011447	-14.1006	4.265941	-6.04617	3.782207	0.221799	-5.1094	-0.41023	-1.17992	-1.9168	-2.29266	2.881967	-4.2352	blues
blues.000	-173.792	137.1867	-23.724	26.97843	-16.6577	4.18088	-12.5599	5.832182	-2.87432	-1.19742	-1.95483	1.8015	-2.98965	-1.86666	-0.79274	-3.01597	-0.44005	-1.99455	-0.74847	-4.99131	blues
blues.000	-287.924	124.0188	5.469893	33.65741	1.095497	14.70533	-4.87701	8.670525	0.785405	6.579527	-4.87497	-3.17822	-5.53616	-6.78444	-2.64816	-2.99456	-3.38682	-1.85265	1.839963	-4.8908	blues
blues.000	-207.223	132.8306	-15.4288	60.97093	0.729872	12.43765	1.182799	-1.54662	-17.8794	8.352729	-2.45816	-0.60338	0.385961	2.599546	-5.17927	-9.52348	-9.24284	-2.85503	-1.42716	-5.93198	blues
blues.000	-209.826	124.4632	10.34245	47.01091	6.508125	15.41052	-2.00511	6.706566	-21.5576	10.71841	-5.52369	-2.19981	-3.03486	8.644537	-0.30372	-0.63795	-2.93843	-1.70452	-1.76819	-7.76218	blues
blues.000	-305.655	113.0863	12.11443	45.03371	17.76283	14.57333	4.15943	0.213545	-7.02699	13.64199	8.929174	1.303006	5.961251	8.549223	1.124606	0.602693	0.054863	-3.72227	-0.99497	-1.80029	blues
blues.000	-238.516	120.1102	-4.22936	42.2907	5.928563	13.91727	5.357868	1.895557	-16.9367	12.35062	3.94598	-1.39358	0.019285	3.588397	-4.50091	-5.54056	-6.88747	-4.82325	1.747367	-1.8822	blues
blues.000	-233.759	120.6751	-3.17481	39.51848	3.818337	18.83714	-3.67782	4.433138	-14.9655	17.57822	1.841872	2.549422	-1.23317	8.439486	-0.2019	-3.8945	-4.45685	0.009554	1.293031	-2.09808	blues
blues.000	-245.699	140.4911	-2.78975	60.43143	18.91905	9.829609	3.559146	0.515372	-16.3877	6.627509	7.259214	-4.01443	1.602121	5.89501	-3.02147	-4.77367	-3.03099	-1.99409	-0.04675	-3.97123	blues
blues.000	-252.493	104.9371	15.33231	42.66697	6.783058	21.60568	-1.29784	5.675408	-16.6326	12.1731	-1.94728	1.950844	-0.89032	7.770524	2.380243	1.775405	1.325362	3.609512	4.907691	-2.65546	blues
blues.000	-236.657	138.4093	-4.91291	47.62918	5.184201	8.876615	0.513815	-3.80467	-14.9889	10.41707	-0.33364	0.775985	1.777318	4.232987	-0.46784	-6.10521	-5.39052	-1.34121	-3.72302	-4.36478	blues
blues.000	-230.358	127.0783	7.123446	39.7749	1.921049	17.41178	-5.61941	8.249394	-18.1169	13.83506	-3.76993	2.572019	-4.31835	6.806376	-6.1671	-6.19875	-4.50018	-1.82234	-0.5171	1.509917	blues
blues.000	-264.723	138.8684	11.19406	48.50426	14.93076	20.58028	3.207009	1.248956	-15.8465	15.97846	3.369133	-1.51081	1.905925	6.32838	-0.99339	-2.41945	-3.68022	0.491501	5.041454	-0.77685	blues
blues.000	-233.956	136.1457	3.324411	48.92853	8.715444	13.96602	5.366597	0.39954	-13.4699	16.87772	5.148091	-2.24982	3.947908	5.316488	0.810559	-1.10708	-4.55648	-2.43078	3.324836	-0.60245	blues
blues.000	-227.225	137.7881	-14.4996	63.56093	5.617668	11.95815	1.907969	-1.88939	-7.60422	-2.47329	-2.89101	-9.43339	-10.9292	3.569476	4.149801	0.088809	-2.98596	-1.38617	7.541378	-3.03572	blues
blues.000	-274.366	130.4113	-6.35129	70.40299	13.26973	9.473776	7.73533	-1.23497	-0.57239	-2.30335	3.773516	0.075612	-0.93897	2.296226	4.294117	-1.66121	-10.4394	-6.7931	-1.06578	-1.58389	blues
blues.000	-249.285	131.0047	0.671435	66.70394	17.9148	4.621686	6.613105	1.242521	1.412567	-1.58981	-0.98192	-3.57579	-2.92696	4.600675	7.480606	-0.18753	-2.82347	-2.67538	0.740614	-0.53098	blues
blues.000	-284.848	108.7495	9.129683	51.25132	18.12372	7.632778	8.775374	2.365382	0.019261	2.236928	4.190629	-3.30458	1.614048	2.691864	3.376128	3.292786	-4.52448	0.496721	5.921975	3.21454	blues
blues.000	-228.167	130.3397	-23.3107	70.20022	1.129326	7.589114	0.903396	1.994443	-6.15884	-9.38072	-2.56357	-4.52185	-0.26999	0.170451	-1.52873	1.467687	-6.98084	-8.32449	4.240889	0.991737	blues
blues.000	-248.962	139.0318	-16.24	57.1162	20.19236	10.16037	4.576076	0.337924	-7.86282	-3.47025	-2.64882	-2.84267	-0.89477	-2.41338	2.570597	-5.871	-12.4129	-7.302	-0.92359	-2.69858	blues
blues.000	-233.715	101.199	17.6509	23.74064	2.079885	25.9389	-16.969	15.07957	-22.2774	4.385386	-23.1976	-6.03665	-5.0571	1.069037	-5.77851	-8.31013	-11.3844	-6.7235	-10.0136	-14.0261	blues
blues.000	-206.938	117.7918	23.26499	28.30005	-1.10228	18.86718	-21.9716	17.84802	-25.5902	2.135104	-17.7773	-8.01059	-17.6201	-8.12008	-8.62889	-15.5405	-15.332	-9.65651	-10.0898	-17.8269	blues
blues.000	-221.843	110.8524	18.63755	30.92843	3.296228	22.58996	-20.4066	12.12297	-26.3305	-4.35413	-26.3839	-13.0322	-12.655	-1.81584	-7.259	-6.66099	-14.6881	-11.7134	-11.0247	-13.3899	blues
blues.000	-288.715	105.9263	18.81995	23.67446	5.700168	25.55909	-13.4505	12.42951	-11.9612	1.589508	-15.6231	-6.63876	-11.6858	-2.93577	-7.37843	-15.6938	-10.9573	-10.8798	-18.502	-19.1178	blues
blues.000	-238.951	108.7054	34.14585	30.21115	-0.50258	27.29296	-21.3995	14.17186	-24.5123	-1.88427	-28.0575	-11.2557	-16.4644	-7.64069	-12.7849	-15.3525	-13.7459	-7.40849	-10.0783	-13.2909	blues

Figure 4: Example of generated MFCC(mel-frequency cepstrum coefficient) logged in excel sheet.

Statistics

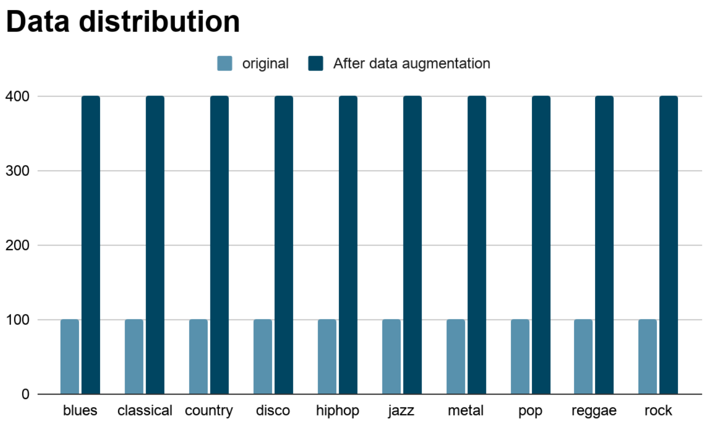


Figure 5: Cleaned data distribution for each genre.

Total Number of dataset
- 400 audio clips in each genre
- Total of 4000 audio clips

Dataset split ratio
- 70% training data (2800 audio clips)
- 15% validation data (600 audio clips)
- 15% test data (600 audio clips)

5. Architecture

Between the option of training with mel-spectrogram images or MFCC numbers, we believe that mel-spectrograms offer more possible information for the machine to learn from. Building upon this, we decided to use a convolutional neural network with a fully-connected classification neural network with ReLU activation functions.

Convolutional Neural Network

There are three increasing convolutional neural networks layers channel sizes of 32, 64, and 128 respectively, each with a kernel size of 3 and stride and padding of 1. After each convolutional layer, we decided to perform batch normalization to improve training efficiency and maximum pooling layers to help consolidate learned information.

Fully-Connected Neural Network

The classifier is a simple fully-connected neural network with hidden units of 128, 64, and 10 respectively, with dropout layers of 50% to avoid overfitting to the training data. The last output layer represents the classifications for each of the genres, and must be applied cross entropy loss.

Long Short-Term Memory

After noticing that mel-spectrograms not only contained information about the frequencies but also the order in which they appear, or the arrangement of the song, we decided to incorporate a bidirectional long short-term memory module between the CNN and the classification ANN. The feature maps extracted through the CNN are fed in sequence to the LSTM, which are converted into the embedding for the classifier. We decided to use 128 units in the hidden layer, as making the model too complex resulted in overfitting or slower training. This allows the overall model to learn as much about the mel-spectrograms as possible.

6. Baseline Model

To compare the effectiveness of our model, we will compare the accuracy with a few rudimentary supervised baseline models. The two baseline models that we will be using are the k-nearest neighbours algorithm (k-NN) and a support-vector machine (SVM). Both of these models will be implemented in the scikit-learn Python library [4][5].

k-Nearest Neighbours Algorithm

The k-nearest neighbours algorithm is a classification algorithm that processes labeled data [6]. Given an unlabeled data point, the algorithm finds the closest labeled data point within the parameter vector space.

Each mel-spectrogram from the labeled training data will be converted into 20 mel-frequency cepstral coefficients (MFCC) to populate the vector space. For each unobserved datum, using the Minkowski distance

$$dist(x, z) = \left(\sum_{r=1}^{d} \left| x_r - z_r \right|^p \right)^{1/p},$$

the algorithm will find 5 nearest neighbours (i.e. closest sounding clips). The predicted label will be the most commonly occurring label from the 5 neighbours, weighted by their distance.

Support Vector Machine

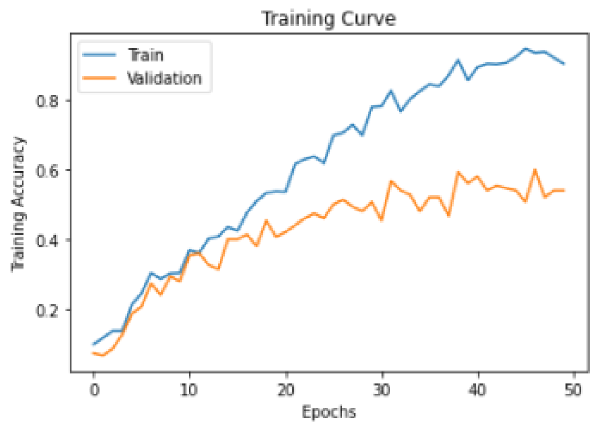
A support vector machine attempts to maximize the separation between clusters of labeled vectors [7]. Since we do not know the general shape and behaviour of our data, we can blindly assume that the classes are not linearly separable. Changing the SVM kernel from a linear function to the Gaussian radial basis function (RBF)

$$rbf(x, z) = exp\left(- \gamma||x - z||^2\right),$$

allows the vector space to be mapped to an infinite dimensional feature space for the SVM to create non-linear decision boundaries. Once trained, the SVM can begin predicting the labels of unobserved data.

7. Quantitative Results

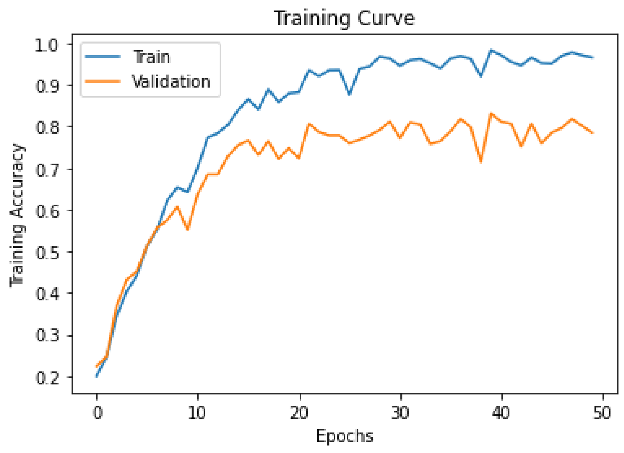
The KNN algorithm had a testing accuracy of 66.85%, and validation accuracy of 44.67%. The SVM had a testing accuracy of 47.57%, and validation accuracy of 42.67%, which are lower than K-nearest Neighbours algorithm. We have decided to use the KNN algorithm performance as the baseline model.



```
Final Training Accuracy: 0.9027181688125894
Final Validation Accuracy: 0.54
Total time elapsed: 1060.66 seconds
```

Figure 6: Training and validation accuracy curves with 1000 samples, 50 epochs, learning rate 1×10^{-4}.

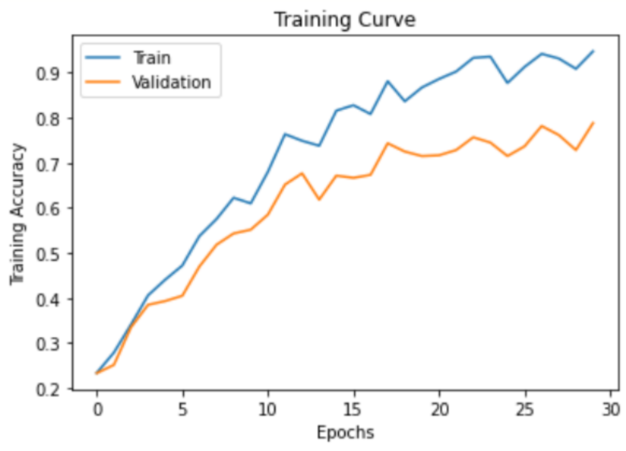
The first trial run had 1000 samples in dataset, and the model had 50 epochs with Adam optimizer which had learning rate of 1×10^{-4}. We obtained training accuracy of 90%, but the validation accuracy was only 54%, indicating overfitting.



```
Final Training Accuracy: 0.9660714285714286
Final Validation Accuracy: 0.785
Total time elapsed: 3760.61 seconds
```

Figure 7: Training and validation accuracy curves with 4000 samples, 50 epochs, learning rate 1×10^{-4}.

To help generalize, we decided to increase the dataset to 4000 samples through data augmentation. The hyperparameters of the model were not changed for the second trial, and we obtained training and validation accuracy of 96.6% and 78.5%. However, the training time was longer than an hour, and we realized that accuracies are not improving significantly after epoch 30.



```
Final Training Accuracy: 0.9475
Final Validation Accuracy: 0.7883333333333333
Total time elapsed: 1868.63 seconds
Test Accuracy:  0.7783333333333333
```

Figure 8: Training and validation accuracy curves with 4000 samples, 30 epochs, learning rate 1×10^{-4}.

In the third trial, we decided to use 30 epochs, which took 31 minutes for training and validation. The training and validation accuracy was 94.75% and 78.83%, which was not too different from the result we obtained in the second trial. This final model produced a test accuracy of 77.83%.

8. Qualitative Results

In order to check if its prediction is correct, the model has compared the actual label of the song and predictions, and increased the number of correct predictions when the actual label corresponds to predictions.

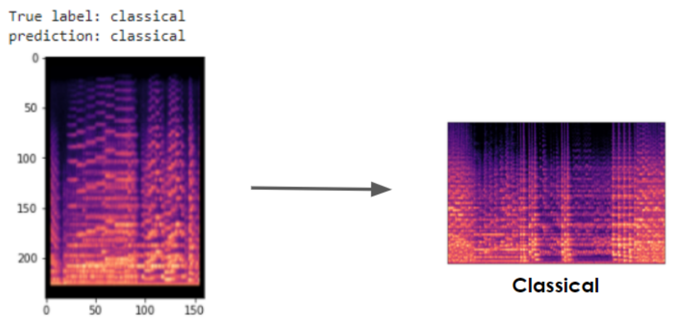


Figure 9: Sample prediction with correct classification.

This image shows sample prediction with correct classification. It is labelled as classical, and prediction of our model was aligned with the label. To compare it with another mel-spectrogram from classical, they look similar to each other. Both of them are having some black area at the above, followed by a long range of purple and orange, and a short range of orange below. They also showed similar patterns.

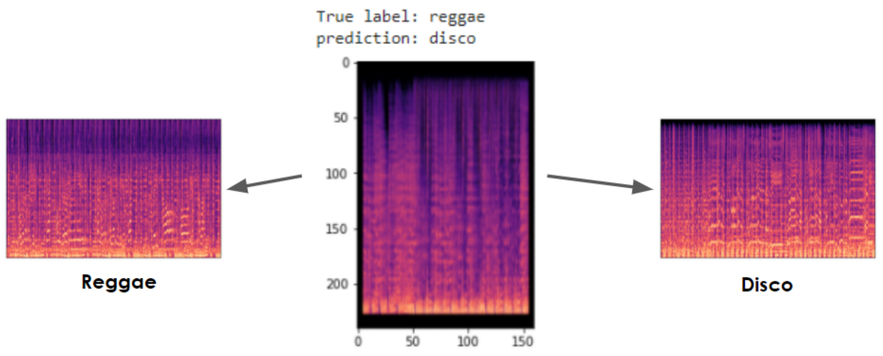


Figure 10: Sample prediction with wrong classification.

This image shows sample prediction with wrong classifications. It was labelled reggae, but the model predicted it as disco. Comparing it with examples from reggae and disco, all three were showing similar patterns, but reggae does not have black region at the top, while disco does. The image had a black region on top, so it appears like the music clip had some disco traits which could confuse our model.

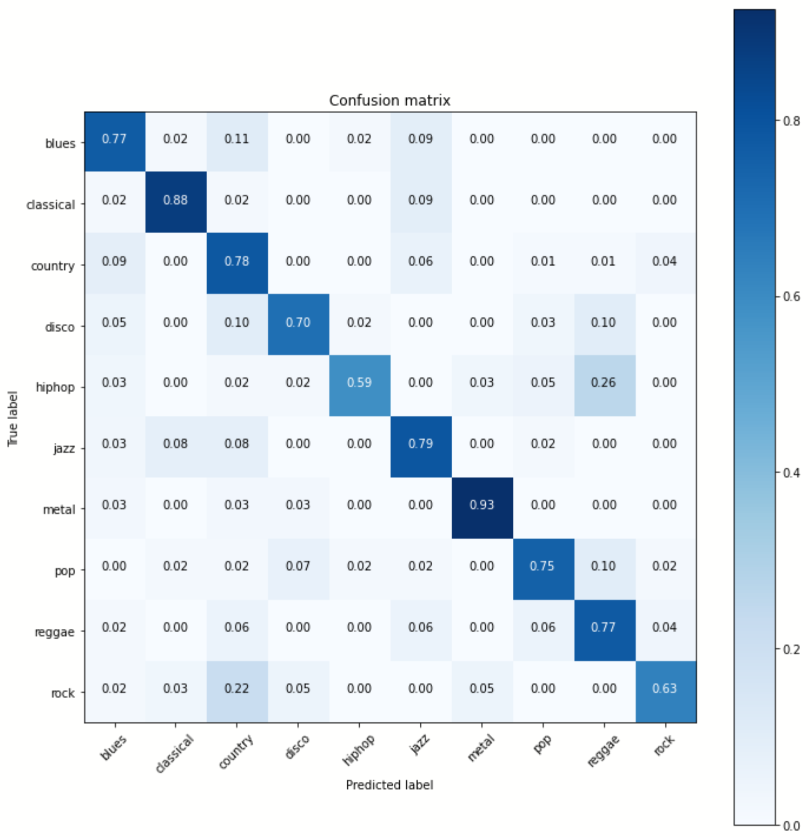


Figure 11: Confusion Matrix.

According to Figure 11, our model has predicted classical and metal clips accurately, while it showed lower accuracy on hip-hop and rock. The most inaccurate prediction was predicting hip-hop as reggae (26%), followed by predicting rock as country (22%), blues as country (11%), and disco as reggae (10%). It confirms that the model could be confused by music clips which show traits of different genres.

9. Evaluate Model on New Data

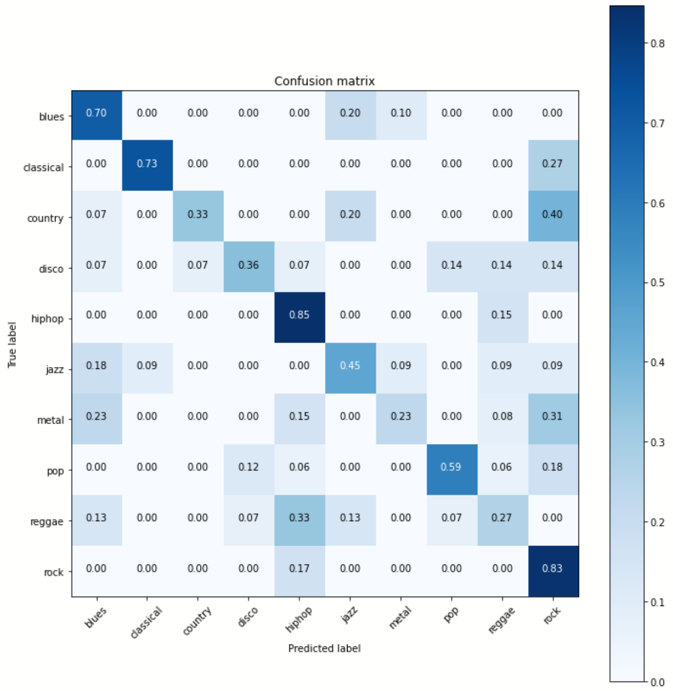


Figure 12. Confusion matrix from new dataset

To test our model on a completely new dataset, we created a new dataset consisting of 3 songs each for each genre, which is 30 songs in total. In the new dataset, we obtained an accuracy of 61.725%. Even though it is slightly lower than test accuracy in [Figure 11], it is still considered better than the baseline test accuracy of 50.67% provided by K-nearest Neighbours algorithm. For this new dataset, hip-hop and rock are the two genres that were predicted most accurately, while our model doesn't perform as well when trying to identify country, metal and reggae genres.

The accuracy from the new dataset is lower than validation or test accuracy, and it can be a consequence of having "bad" data. GTZAN is the dataset that is popular for machine learning projects involving music audio, because it is processed and labeled properly. We used GTZAN datasets for training our models and hyperparameters tuning. However, for the new dataset we obtained our data from other sources on the internet, which has risks of data not being labelled properly or data that are of low quality. If the music clips are not labelled properly, then there is a chance that the accuracy from the dataset is lower. For example, if a song is labeled as "metal" when in reality it is more similar to being classified as "rock", then this would result in a wrong prediction from our model if our model classifies it as "rock". Since music can have traits of multiple genres, labelling a song into one genre may not represent the song properly. If we chose songs which show traits of multiple genres, then it is likely that the model would not classify the song's genre properly.

10. Discussion

As mentioned in the introduction, we believe that genre-detection is not a trivial task. There are large diversities between genres itself, caused by differences in musicians and creative liberties taken by certain composers. Despite the patterns formed by musical properties within genres, these factors often cause ambiguity in mel-spectrograms and cannot be clearly parsed by the machine. For example, in [Figure 10], the mel spectrogram was labelled as reggae, but our model classified it as disco, since it had similar characteristics with disco's mel-spectrogram. This can justify our low testing accuracy, and may also justify the existence of a performance plateau of genre-detection machines in general.

One of the more important things we learned was the importance of having large datasets in complex problems, as it can greatly improve a model's accuracy. We were able to increase validation accuracy by 24.5% after creating a larger dataset using data augmentation. Since data augmentation was done by simply manipulating existing data to create new data, we didn't expect large improvements in performance. The new data, however, introduced enough diversity to allow the model to generalize efficiently.

11. Ethical Consideration

Music is a form of artistic and creative expression. There isn’t a definite description of how a certain genre of music has to sound like. A song can be very dynamic, composed of elements from many genres of music. By trying to give a definite answer and classifying a song to be only one genre of music, we are restricting its artistic expression. This puts a preconceived notion on a song that may go against the ideas of its creator. For example, our model may classify a song to be Rock music, but its author may claim otherwise.

Most music streaming platforms have playlists of different music genres. If a song is classified as only one genre and is only placed in one of the many genre-specific playlists, it will lose many potential listeners who don’t listen to other playlists because of its labeled genre. As a result, an artist may lose profits on music royalties or the reach of a bigger audience. Furthermore, there is a potential for our project to replace jobs in the music industry that involves music classification, like playlist curators on music streaming platforms.

12. Project Difficulty

Throughout the report, we have repeatedly noted how the difficulty in identifying genres stems from the ambiguous similarities between genres. For example, the two genres rock and metal are often misclassified by even human ears, a decision that usually comes down to personal opinion.

We attempted to thwart this inherent and unavoidable property by including an LSTM module to learn from musical arrangement, as humans don’t actively consider arrangement when determining a genre. With a fairly deep network and the LSTM in place, we only managed to acquire a 77.83% accuracy.

At the start of hyperparameter tuning, while attempting to deepen the network, we noticed that we were not only overfitting to the training dataset but severely slowing down the training speed. To mitigate this, we worked with a smaller network and implemented dropout layers to help generalize the model, and batch normalization layers for training efficiency.

We only managed to achieve a validation accuracy of 50% with this architecture, and decided that the only way to improve generalization was to generate more data. With a limited audio file data set, we used data augmentation techniques of noise injection, pitch shifting and time stretching to increase the amount of data that we have to train our network. As a result, we were able to drastically increase the validation performance from 50% to around 75%.

13. References

[1] T. Greer and S. Narayanan, “Using shared vector representations of words and chords in music for genre classification,” SMM19, Workshop on Speech, Music and Mind 2019, 2019.

[2] D. A. Huang, S. A. Arianna, and P. J. Eli, “Music Genre Classification,” 2018.

[3] Index of /sound/genres. [Online]. Available: http://opihi.cs.uvic.ca/sound/genres/. [Accessed: 12-Feb-2021].

[4] “1.4. Support Vector Machines¶,” scikit. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html. [Accessed: 12-Feb-2021].

[5] “sklearn.neighbors.NearestNeighbors¶,” scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html#sklearn.neighbors.NearestNeighbors. [Accessed: 12-Feb-2021].

[6] “Lecture 2: k-nearest neighbors / Curse of Dimensionality,” Cornell Computer Science - CS4780. [Online]. Available: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html. [Accessed: 12-Feb-2021].

[7] R. Berwick, “An Idiot’s guide to Support vector machines (SVMs),” in 6.034 Artificial Intelligence - MIT.