Jim Sears
ADS654Z1 - Deep Learning
Final Project Proposal
11/21/21

I Know It's Only Rock 'N' Roll (But Does TensorFlow?)

Music Genre Classification with Neural Networks

ADS654Z1 - Deep Learning

James Sears

BAY PATH UNIVERSITY 1897

Applied Data Science Program, Graduate School, Longmeadow, MA

Submitted To: Dr. Rida Moustafa

Fall 2021

Jim Sears
ADS654Z1 - Deep Learning
Final Project Proposal
11/21/21

Abstract

The object of this project is to utilize deep learning techniques to classify a song's music genre from the following options: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock. The data used is from the GTZAN dataset with one thousand labeled audio files: one hundred 30-second-long samples for each of the ten genres listed. An audio file can be represented visually as a time series of its frequencies. Discrete Fourier Transformations of simple waveplots produce spectrograms, which are heatmaps that depict the changing energy of a song. Mel-scaling and Discrete Cosine Transformation of spectrograms produce real-valued coefficients called Mel-Frequency-Cepstral-Coefficients (MFCC), the first thirteen of which represent the simplest and most descriptive and interpretable aspects of the spectrogram. They represent the acoustic signature of the songs and as they are in graphical form, are input into a neural network.

The basis for this project is the work of Marc-Henry Saint-Félix, described in a Towards Data Science article and available on his GitHub repository. His preprocessing of the audio files to spectrograms and creation of neural networks were replicated. The author presents his best Multiple Layer Perceptron (MLP) and Convolutional Neural Network (CNN) models, which served as the baselines in terms of classification accuracy and processing time. The goal was to rebuild layer by layer and/or modify and tune the author's models and improve upon his best accuracy rating of 60% and 78% for his MLP and CNN, respectively.

MLPs and CNNs were constructed, modified, and evaluated in a progressive trial an error fashion. Modifications included adding, removing and tuning layers: Dense layers with varying nodes and activation functions; convolutional layers with different kernel sizes and activation functions; pooling and normalizing layers, dropout layers with varying rates. Different optimization functions were evaluated as well. While the batch sizes and learning rates were held constant, the number of epochs were varied. An MLP with a 71% accuracy rating and a CNN with an 81% accuracy rating were constructed, outperforming those of Marc-Henry Saint-Félix.

Jim Sears
ADS654Z1 - Deep Learning
Final Project Proposal
11/21/21

Introduction

A person's preference in music genre can be narrow or varied. These preferences are – or at least seem to be – unique to each person. Some may prefer only jazz or classical, and some may like rock, country and blues. Streaming services such as Spotify or Sonos generate millions in revenue by providing instant access to music. But what separates one service from the other is how well they cater to the users' preferences and generate playlists of suggested songs and artists based on what the user has listened to previously. A foundational building block of these streaming services' algorithms is classifying the genre of music of a song. It is the first step to ensure that their suggestions are appropriate to and appreciated by the user; so that, for example, a metal song is not suggested to a jazz aficionado.

The object of this project is to utilize deep learning techniques to classify a song's music genre from the following options: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock. The data used is from the GTZAN dataset and was downloaded from Kaggle. This dataset was used for the well-known paper in genre classification "Musical Genre Classification of Audio Signals" by G. Tzanetakis and P. Cook, published in IEEE Transactions on Audio and Speech Processing, Volume 10, Issue 5, from July 2002. It consists of one thousand labeled audio files: one hundred 30-second-long samples for each of the ten genres listed. "The GTZAN dataset is the most-used public dataset for evaluation in machine listening research for music genre recognition (MGR). The files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions" (Olteanu).

Audio files are essentially time series of soundwaves. Features of audio files can be extracted, and subsequent visualizations generated with the Python package Librosa. For this project, the visual representations created for each song segment are spectrograms and Mel-Frequency-Cepstral-Coefficients (MFCC) depictions, which are essentially heatmaps of the various frequencies (instruments, vocals, etc.) over time. With the data converted from an audio to a visual format from this data preprocessing, neural networks are trained on the labeled images to classify new images as one of the ten music genres. The types of neural networks implemented in this project are Multiple Layer Perceptrons (MLP) and Convolutional Neural Networks (CNN) built on the TensorFlow framework.

The basis for this project is the work of Marc-Henry Saint-Félix, described in a Towards Data Science article and available on his GitHub repository. His preprocessing of the audio files to spectrograms and creation of neural networks were replicated. The author presents his best models (MLP and CNN), which will serve as the baselines in terms of classification accuracy and processing time. The goal is to rebuild layer by layer and/or modify and tune the author's models and improve upon his best accuracy rating of 60% and 78% for his MLP and CNN, respectively. The methods used include removing and adding different layers and types (Dense, Pooling, Normalization), using different activation and optimization functions, varying the number of epochs, and implementing regularization in the form of dropout.

Methods

Data Preprocessing

To prepare the audio files for image classification, they are subjected to a series of transformations made by the tools of the Librosa library. The details of the process are beyond the scope of this project but are summarized below; they are best explained in the *Music Genre Detection With Deep Learning* article from Towards Data Science.

An audio file can be represented visually as a time series of its frequencies (waveplots in Figures 1, 2 and 3). They can be broken into a defined number of frames and "over each frame, a frequency vector (of n frequency bins) is calculated by applying the *Discrete Fourier Transform* (DFT). The frequency vector is the instant representation of our audio signal in the frequency domain. You can see it as a description of the sound in terms of energy distribution across all frequency bins at a given time (e.g a given frame)." As the song progresses, the frames change, as does the energy distribution, generating heat maps known as spectrograms (Figures 1, 2 and 3). The red lines of the spectrograms represent higher energy and volume (Saint-Félix).

The sample rock and disco songs (Figure 1) and metal song (Figure 3) sound louder and more energetic than the thinner sounding blues and reggae songs (Figure 1); the former three have more red coloring in their spectrograms than the latter. The tempo of the songs can also be seen in the spacing of the spectrograms' vertical lines. Lines that are closer together, such as the disco song (Figure 1), indicate a faster tempo. Another indicative feature of the spectrograms are the red lines that span many frequencies (vertical axis) of the spectrogram. These indicate that all the different frequency ranges

Figure 1. **Waveplot, Spectrogram and MFCC of Rock (rock.00028.wav), Disco (disco.00001.wav), Blues (blues.00001.wav) and Reggae (reggae.00003.wav) song samples.**

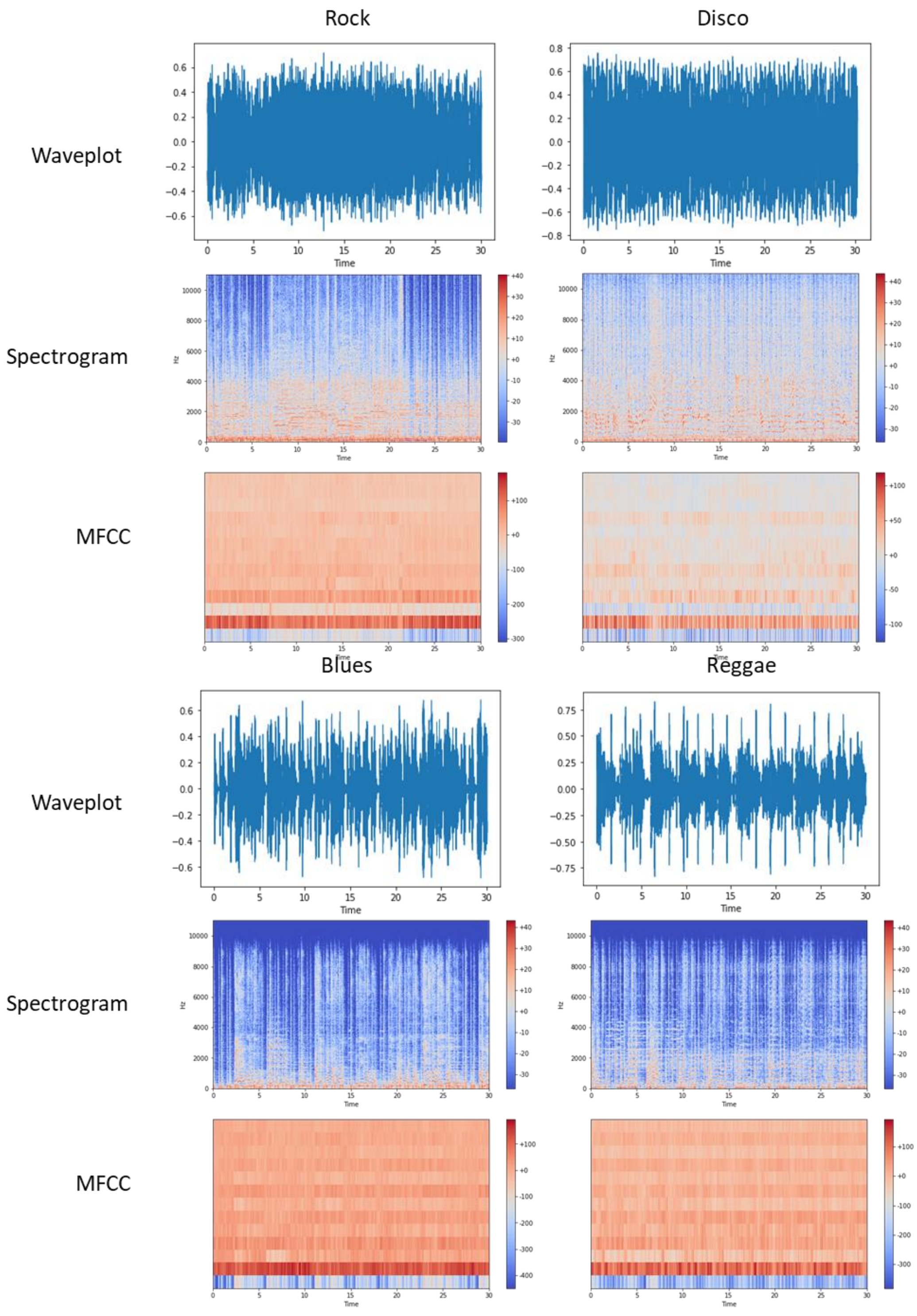


Figure 2. Waveplot, Spectrogram and MFCC of Classical (classical.00003.wav), Country (country.00028.wav), Hip-Hop (hiphop.00001.wav) and Jazz (jazz.00012.wav) song samples.

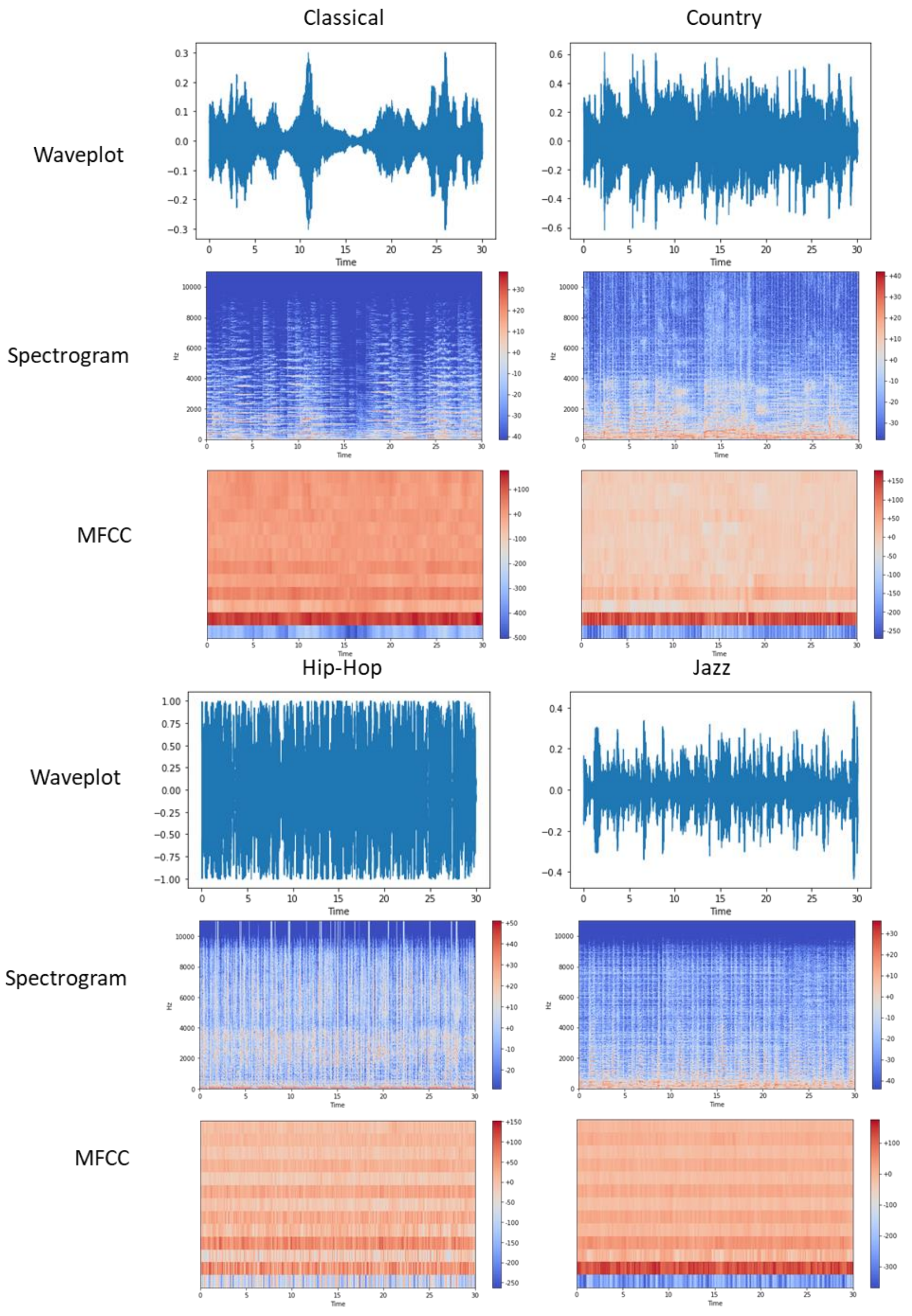
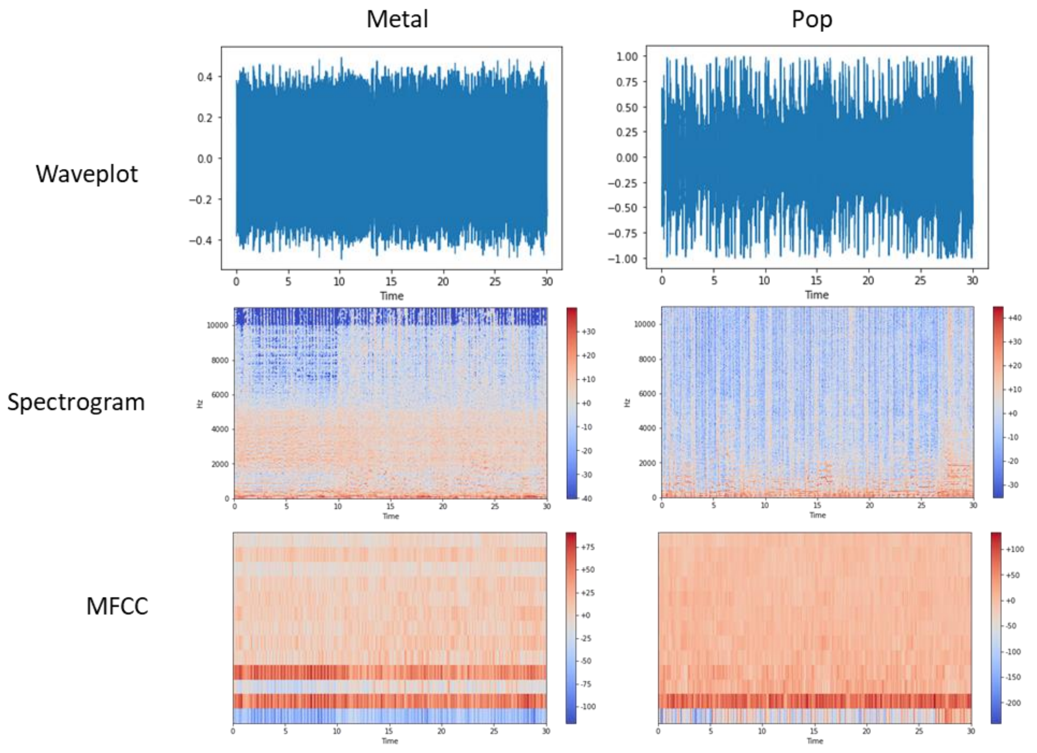


Figure 3. **Waveplot, Spectrogram and MFCC of Metal (metal.00000) and Pop (pop.00028.wav) song samples.**

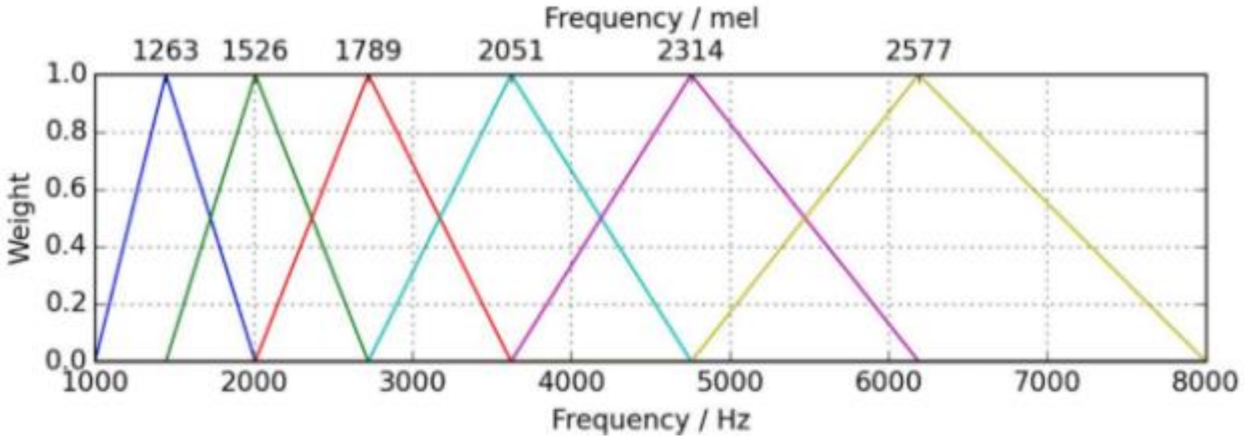


(instruments and vocals) are playing at the same time, or on the same beat in other words. This is a strong characteristic of rock (Figure 1) and hip-hop (Figure 2) music that is represented in their respective spectrograms. Reggae music is notably different in that it's distinctive sound comes from some instruments playing on the beat, and others off it. The result is shorter vertical red lines (Figure 1).

The next step in data preprocessing converts the spectrogram to a format suitable for deep learning. "Humans don't hear sounds linearly: the perceived pitch difference between A1 (55Hz) and A2(110Hz) is the same as A4 (440Hz) and A5 (880Hz). These intervals are both octaves but there is a difference of 55Hz for the first octave, whereas there is a difference of 440Hz for the other octave. The Mel-Scale defines frequency bands that are evenly distributed with respect to perceived frequencies. Mel filter banks are calculated so that they are more discriminative for lower frequencies and less so for higher ones, just like human ears. The harmonic structure is smoothened by performing weighted discriminations" (Saint-Félix) per Figure 4.

After Mel-scaling, a Discrete Cosine Transformation is performed to generate only real-valued coefficients called Mel-Frequency-Cepstral-Coefficients (MFCC). The first 13 MFCCs are retained as they typically represent the simplest and most descriptive and interpretable aspects of the spectrogram; the remaining coefficients tend to describe “noise” and are left out (Saint-Félix).

Figure 4. Triangular Mel filter banks.



Examples of the graphical representation of the MFCCs are seen in Figures 1 and 2. Each of the 13 coefficients is represented by a distinct horizontal band. They represent the acoustic signature of the songs and as they are in graphical form, are ready to be input into a neural network.

Most of the example song samples generate spectrograms and MFCCs that vary in appearance, based on their assigned genre, such that image classification seems feasible. This is more apparent when comparing examples of music genres that are obviously different, such as classical and metal. However, some music genres have similar traits, such as disco and pop, and produce similar spectrograms and MFCCs. This overlap in music styles does raise concern for the accuracy of a classifier.

Prior to the data preprocessing steps described above, an integrity check was performed, followed by regularization and augmentation. In the initial loading of the data, a corrupt file (jazz.00054.wav) was identified and removed from the local data source. Next, to account for any variation and ensure that each song sample was of the same length, all .wav files were cut to 29 seconds in length. Lastly, each sample was then sliced into smaller samples of 3 second lengths. The latter augmentation step served to generate a data set ten times its original size, from 999 to 9,990 samples. A larger data set helps prevent the generation of overfitting models. The larger sample set was divided into a training set (70%; 6,993 samples) to build the neural network models and a test set (30%; 2,997 samples) to evaluate their accuracy. The selection of samples and distribution of the ten genres was random but averaged 300 ± 30 (10%) for each genre.

Model Architecture

Two types of neural networks built on the TensorFlow framework were used to generate the models for this project: Multiple Layer Perceptron (MLP) and Convolutional Neural Networks (CNN). The methods for their construction and the options that distinguish the various models created are described here.

Multiple Layer Perceptron

A Multiple Layer Perceptron (MLP) is a feedforward neural network with at least one hidden layer between the input and output layers. Each layer contains nodes. In this case, the input layer contains 13 MFCC bands for each 125 frames per 3 second sample) and the output layer contains 10 nodes that represent the ten genres of music to be classified. The number of nodes in each hidden layer – and the number of hidden layers - is discretionary and typically done by experimenting with these variables. A model may be improved by adding more hidden layers and/or varying the number of nodes per layer. Each additional layer may increase the processing time required to train the model, but the cost is warranted with significant increases in classification accuracy.

Except for those of the input layer of an MLP, each node is a neuron that utilizes an activation function to map the weighted inputs to the output of each neuron. Linear activation functions do not allow backpropagation and reduce an MLP to a single layer because a linear combination of linear functions is still a linear function (Little). Therefore, as they can only create linear separation boundaries for classification (essentially linear regression), non-linear activation functions are used to solve more complex problems. Popular non-linear activation functions are Sigmoid and ReLU (Rectified Linear Unit) Function. While the latter sigmoid based, it is preferred as it overcomes the noted issues of the Sigmoid approach, especially with deeper neural networks:

> *A general problem with both the sigmoid and tanh functions is that they saturate. This means that large values snap to 1.0 and small values snap to -1 or 0 for tanh and*

> *sigmoid respectively. Further, the functions are only really sensitive to changes around their mid-point of their input, such as 0.5 for sigmoid and 0.0 for tanh.*
> *The limited sensitivity and saturation of the function happen regardless of whether the summed activation from the node provided as input contains useful information or not. Once saturated, it becomes challenging for the learning algorithm to continue to adapt the weights to improve the performance of the model.*
> *Finally, as the capability of hardware increased through GPUs' very deep neural networks using sigmoid and tanh activation functions could not easily be trained. Layers deep in large networks using these nonlinear activation functions fail to receive useful gradient information. Error is back propagated through the network and used to update the weights. The amount of error decreases dramatically with each additional layer through which it is propagated, given the derivative of the chosen activation function. This is called the vanishing gradient problem and prevents deep (multi-layered) networks from learning effectively (*Brownlee, *ReLU*).

The MLP models evaluated in this project used standard densely connected (Dense) layers with either Sigmoid or ReLU activation functions. The output layer for all models, including the CNNs, utilize the Softmax activation function as it enables the multiclass classification of the ten music genres.

Once the layers of the MLP are defined, they are compiled "into a highly efficient series of matrix transforms in a format intended to be executed on your GPU or CPU" (Moustafa, Lecture 3). The compiler's optimization algorithm (or functions), loss function and learning rate are selected. For a multiclass classification project, Categorical Cross Entropy is selected as the loss function to evaluate the models. Two optimization functions were evaluated, Adam and RMSprop; the latter was used in Saint-Félix's models. "The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated (Brownlee, *Learning Rate*). A learning rate that is too small may never achieve the optimal node weights; one that is too large may overshoot the optimal weights. In either scenario, convergence is not achieved and a subpar model result. For this project, due to the number of model architectures to be evaluated, learning rates were help constant at a moderate rate of 0.001 to balance the desire for convergence without extensive processing times.

Other variables that may be tuned in a neural network are the number of epochs and batch size. Epochs are the number of iterations of the training data through which the training model will run. This enables the network to learn more with each pass but has the potential to overfit the training data if the number of epochs is too great. Epochs are partitioned into groups of input and target data pairs with the defined number of pairs per group (batches). This can aid in efficient use of computational memory and tune the neural network as the batch size determines the number of input and target data pairs seen before the weights are updated within each epoch (Moustafa, Lecture 3). For all models in this project, the batch size was kept constant at 32, but the number of epochs were varied.

Convolutional Neural Network

A Convolutional Neural Network (CNN) consists of convolutional layers that filter their input matrices (tensors) to enhance the important and distinctive features of an image, such as detecting edges or drastic changes and the removal of noise such as an irrelevant background. As such, this form of feature identification is ideal for examining MFCCs. The numeric representations of the pixels of an image are traversed in overlapping – typically square - groupings of a designated dimension, called kernels. At each point, "all of the corresponding input values and weights are multiplied together, and then summed to produce a single output value at that point" (Moustafa, Lecture 6). The result is the best and most succinct representation of the pixels. While the dimensions of the kernel can be varied from model to model, it was set to 3x3 or 2x2 in each CNN created for this project.

A convolutional layer's output can be further filtered, and subsequently focused, by a pooling layer which also serves to reduce dimensionality. Pooling traverses the convolutional layer's output in designated groupings – typically square – and represents the group by a single output value. The output value is typically either the maximum (Maxpooling) or average (Average-Pooling) value of the group. All pooling layers implemented in this project utilize Maxpooling in 2x2 windows with a stride width of 2.

Convolution and pooling have the potential to produce outputs with features with higher values than others. A neural network can show bias towards those higher values, so a normalization layer can be introduced to maintain the contribution of every feature. There are multiple options for normalization layers, including weight, layer, batch and others. For this project, when normalization was implemented, it was of the batch variety. "Batch normalization is a method that normalizes activations in a network across the mini-batch of definite size. For each feature, batch normalization computes the mean and variance of that feature in the mini-batch. It then subtracts the mean and divides the feature by its mini-batch standard deviation" (Bindal).

Neural networks are prone to relying on particular connections and therefore overfitting the training data. Regularization in the form of dropout can be utilized to overcome this limitation. Dropout layers randomly leave out a desired percentage of nodes in the preceding layer. This enables many simulations of the same network as variations of the architecture are achieved when different nodes are dropped or kept.

The CNN models created and evaluated for this project used at least one convolutional layer prior to a flattening layer. Between those layers, different architectures of one or more convolutional sequences that used various combinations of pooling, normalization, and/or dropout layers were trialed. In addition, some models included standard densely connected (Dense) layers with ReLU activation functions after the flattening layer. The final output layer used the same Softmax activation function as the MLPs. The CNN models' compilers all also employ a Categorical Cross Entropy loss function and an Adam or RMSprop optimization function. For all models in this project, the learning rate (0.001) and batch size (32) were kept constant, but the number of epochs varied.

Model Building

All MLP and CNN models were trained and evaluated on the same training and test data sets. The primary evaluation metric is classification accuracy of the unseen test data; processing time is also considered. The process of building the models was done in a progressive trial and error fashion. A basic starting model was run; for the MLPs, two basic models were run and evaluated, and the better performing model then served as the starting model. The starting model was modified by adding an additional hidden layer or changing the number of epochs. If that model's performance improved, it was further modified in the same manner to form a new model; if performance did not improve, the preceding model was modified in a different manner to form a new model. This procedure continued until the subsequent models ceased to exhibit performance improvements.

Results & Discussion

Multiple Layer Perceptron

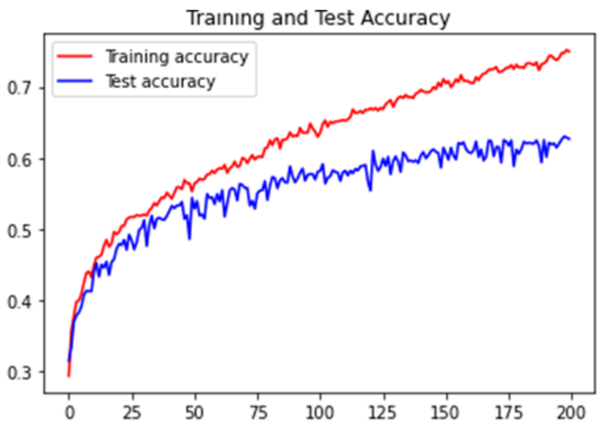
A series of models with the RMSProp optimization function and another series of models with the Adam optimization function were created separately and with the same rationale. Each series has two starting models, both with 50 epochs and the same single hidden Dense layer with 512 nodes, but a different activation function: Sigmoid or ReLU. The activation of the better performing model for each series was then used in each of the hidden Dense layers of the subsequent models.

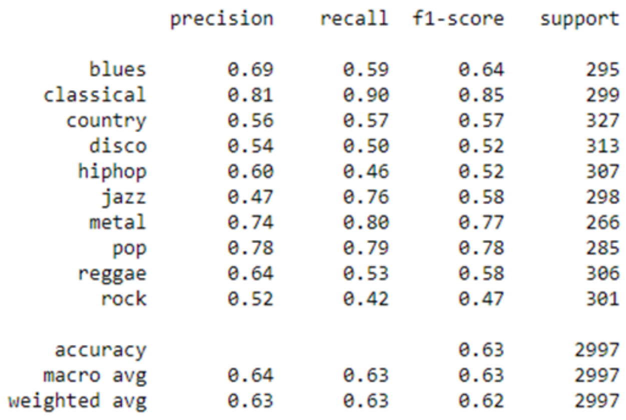
Table 1. Summary of MLP models with Dense hidden layers, RMSProp optimization function, Batch Size of 32 and 0.001 Learning Rate.

Model Name	Hidden Layers	Nodes per Layer	Activation Function	Epochs	Proc. Time	Accuracy Rate %
mlp_base	5	512, 512, 256, 128, 64	ReLU	50	2.50	56.86
mlp_01	1	512	ReLU	50	2.73	44.28
mlp_02	1	512	Sigmoid	50	2.82	47.05
mlp_04	1	512	Sigmoid	60	3.31	48.55
mlp_03	1	512	Sigmoid	100	5.50	46.81
mlp_05	2	512, 256	Sigmoid	60	3.78	53.39
mlp_06	3	512, 256, 128	Sigmoid	60	3.78	52.35
mlp_07	3	512, 256, 128	Sigmoid	80	4.63	53.92
mlp_12	3	512, 256, 128	Sigmoid	100	6.22	55.96
mlp_12.1	3	512, 256, 128	Sigmoid	200	15.82	62.70
mlp_11	4	512, 512, 256, 128	Sigmoid	80	7.39	49.15
mlp_08	4	512, 256, 128, 64	Sigmoid	50	3.07	48.11
mlp_09	4	512, 256, 128, 64	Sigmoid	80	4.99	48.21
mlp_09.1	4	512, 256, 128, 64	Sigmoid	200	12.75	59.43
mlp_10	5	512, 512, 256, 128, 64	Sigmoid	50	4.35	44.44

For the series of models with the RMSProp optimization function (Table 1), the starting model with a Sigmoid activation function (*mlp_02*, 47.05% accuracy) outperformed its ReLU activation function counterpart (*mlp_01*, 44.28% accuracy). Increasing the former's number of epochs to 100 (*mlp_03*) saw a reduction in accuracy (46.81%) but scaling back to 60 epochs (*mlp_04*) improved the accuracy to 48.55%. Adding an additional Dense layer with 256 nodes (*mlp_05*) further improved accuracy to 53.39%. Adding a third hidden layer with 128 nodes improved performance when the number of epochs were increased to 80 (*mlp_07*, 53.92% accuracy), 100 (*mlp_12*, 55.96% accuracy) and 200 (*mlp_12*.1, 62.70% accuracy). Subsequent models with four and five hidden Dense layers did not improve the performance of the MLP, with the exception of *mlp_09.1* with 200 epochs (59.43% accuracy).

Figure 5. Metrics for model *mlp_12.1*. Training and test accuracy and loss charts; classification report; confusion matrix of recall.

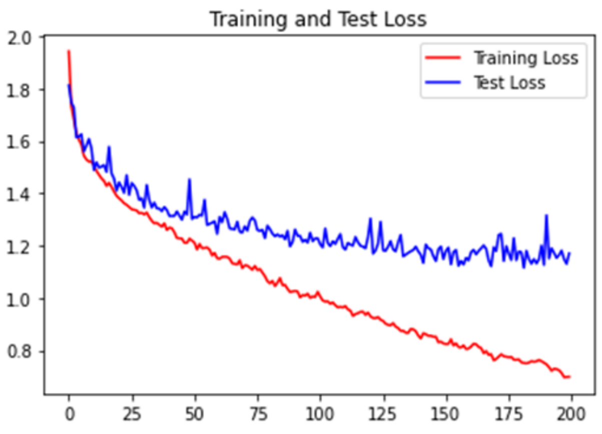


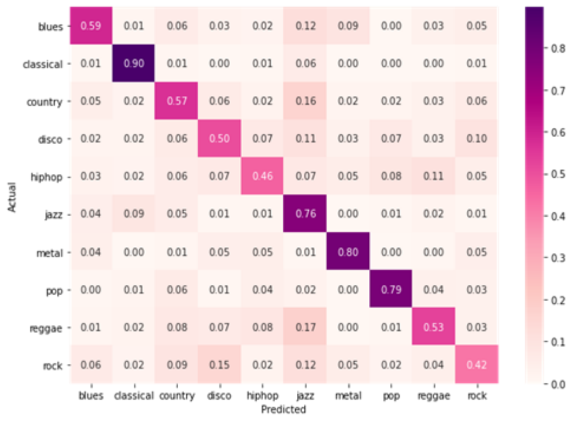


```
              precision    recall  f1-score   support

       blues       0.69      0.59      0.64       295
   classical       0.81      0.90      0.85       299
     country       0.56      0.57      0.57       327
       disco       0.54      0.50      0.52       313
      hiphop       0.60      0.46      0.52       307
        jazz       0.47      0.76      0.58       298
       metal       0.74      0.80      0.77       266
         pop       0.78      0.79      0.78       285
      reggae       0.64      0.53      0.58       306
        rock       0.52      0.42      0.47       301

    accuracy                           0.63      2997
   macro avg       0.64      0.63      0.63      2997
weighted avg       0.63      0.63      0.62      2997
```





	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.59	0.01	0.06	0.03	0.02	0.12	0.09	0.00	0.03	0.05
classical	0.01	0.90	0.01	0.00	0.01	0.06	0.00	0.00	0.00	0.01
country	0.05	0.02	0.57	0.06	0.02	0.16	0.02	0.02	0.03	0.06
disco	0.02	0.02	0.06	0.50	0.07	0.11	0.03	0.07	0.03	0.10
hiphop	0.03	0.02	0.06	0.07	0.46	0.07	0.05	0.08	0.11	0.05
jazz	0.04	0.09	0.05	0.01	0.01	0.76	0.00	0.01	0.02	0.01
metal	0.04	0.00	0.01	0.05	0.05	0.01	0.80	0.00	0.00	0.05
pop	0.00	0.01	0.06	0.01	0.04	0.02	0.00	0.79	0.04	0.03
reggae	0.01	0.02	0.08	0.07	0.08	0.17	0.00	0.01	0.53	0.03
rock	0.06	0.02	0.09	0.15	0.02	0.12	0.05	0.02	0.04	0.42

The best MLP model (*mlp_12.1*) with the RMSProp optimization function had three hidden layers (with 512, 256 and 128 nodes) and 200 epochs. It had a processing time of 15.82 minutes and an accuracy rating of only 62.70%, meaning that overall, it correctly classified the genre of 63% of the song samples in the test data. It predicted classical songs very well at 90% recall, metal (80%) and pop (79%) songs adequately but performed poorly on hip-hop (46%) and rock (42%) songs (Figure 5).

For the series of models with the Adam optimization function (Table 2), the starting model with a ReLU activation function (*mlp_13*, 46.95% accuracy) outperformed its Sigmoid activation function counterpart (*mlp_14*, 42.48% accuracy). Increasing the former's number of epochs to 100 (*mlp_15*) saw a reduction in accuracy (44.58%), but instead adding another Dense layer with 256 nodes (*mlp_16*) significantly improved accuracy to 58.63%. More models (*mlp_18*, *mlp_19*, *mlp_20*, *mlp_21*) were trialed that added more Dense layers and ran with 50 and 100 epochs, of which, model *mlp_19* performed the best with 3 hidden layers and 100 epochs, resulting in an accuracy rating of 62.43%. The next series of models trialed successive duplication of layers with the same number of nodes; for example, 512 nodes in the

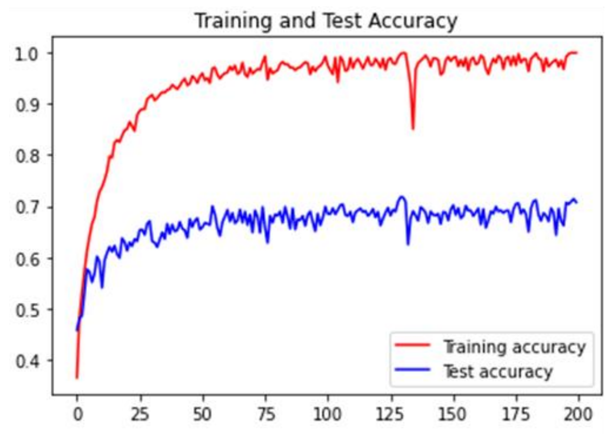
Table 2. Summary of MLP models with Dense hidden layers, Adam optimization function, Batch Size of 32 and 0.001 Learning Rate. Base model uses the RMSProp optimization function instead of Adam.

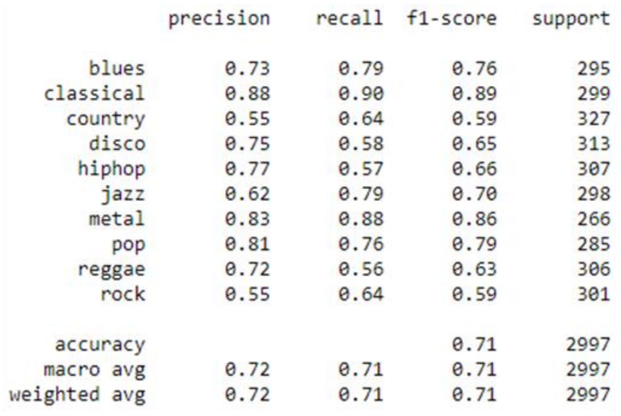
Model Name	Hidden Layers	Nodes per Layer	Activation Function	Epochs	Proc. Time	Accuracy Rate %
mlp_base	5	512, 512, 256, 128, 64	ReLU	50	2.50	56.86
mlp_13	1	512	ReLU	50	1.48	46.95
mlp_15	1	512	ReLU	100	2.97	44.58
mlp_14	1	512	Sigmoid	50	1.55	42.48
mlp_16	2	512, 256	ReLU	50	1.80	58.63
mlp_17	2	512, 256	ReLU	100	3.44	56.82
mlp_18	3	512, 256, 128	ReLU	50	1.87	61.29
mlp_19	3	512, 256, 128	ReLU	100	3.97	62.43
mlp_20	4	512, 256, 128, 64	ReLU	50	1.91	59.66
mlp_21	4	512, 256, 128, 64	ReLU	100	3.79	62.36
mlp_22	5	512, 512, 256, 128, 64	ReLU	50	2.80	66.37
mlp_23	5	512, 512, 256, 128, 64	ReLU	90	5.15	65.70
mlp_24	6	512, 512, 256, 256, 128, 64	ReLU	50	2.93	65.83
mlp_25	6	512, 512, 256, 256, 128, 64	ReLU	100	6.16	68.30
mlp_27	6	512, 512, 256, 256, 128, 64	ReLU	200	13.77	70.77
mlp_26	7	512, 512, 256, 256, 128, 128, 64	ReLU	100	5.87	67.37

first two layers, 256 nodes in the third and fourth layers, and so on. Model *mlp_27* performed the best of this group with six hidden layers (with 512, 512, 256, 256, 128 and 64 nodes) and 200 epochs and achieved an accuracy rating of 70.74%.

Model *mlp_27* outperformed not only the other Adam-optimized models, but all MLPs evaluated. It's processing time of 11.52 minutes was the second longest of all MLPs due to having more layers and twice as many epochs than most of the other models. A 70.74% accuracy rating means that overall, the model correctly classified the genre of 71% of the song samples in the test data. It predicted classical (90% recall) and metal (88%) songs very well, blues (79%) and jazz (79%) songs adequately, and relatively poorly on hip-hop (57%) and reggae (56%) songs (Figure 6).

Figure 6. Metrics for model *mlp_27*. Training and test accuracy and loss charts; classification report; confusion matrix of recall.

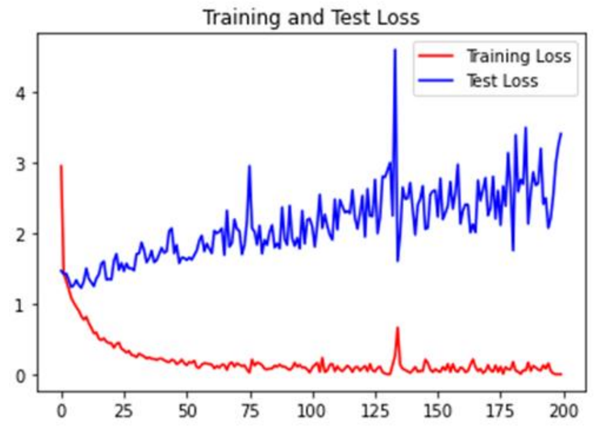


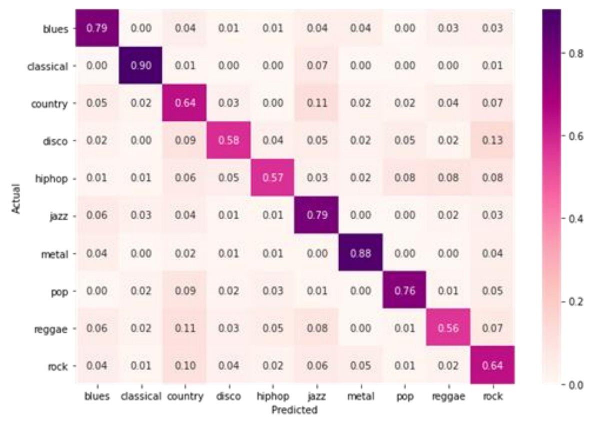


```
              precision    recall  f1-score   support

       blues       0.73      0.79      0.76       295
   classical       0.88      0.90      0.89       299
     country       0.55      0.64      0.59       327
       disco       0.75      0.58      0.65       313
      hiphop       0.77      0.57      0.66       307
        jazz       0.62      0.79      0.70       298
       metal       0.83      0.88      0.86       266
         pop       0.81      0.76      0.79       285
      reggae       0.72      0.56      0.63       306
        rock       0.55      0.64      0.59       301

    accuracy                           0.71      2997
   macro avg       0.72      0.71      0.71      2997
weighted avg       0.72      0.71      0.71      2997
```

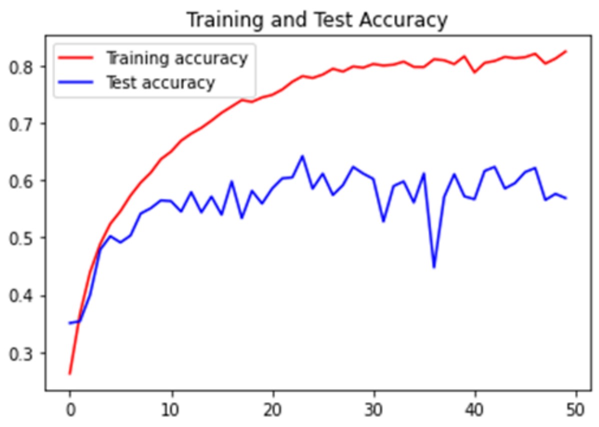


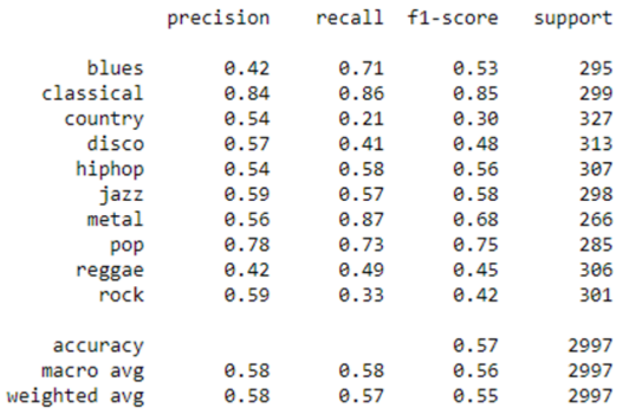


	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.79	0.00	0.04	0.01	0.01	0.04	0.04	0.00	0.03	0.03
classical	0.00	0.90	0.01	0.00	0.00	0.07	0.00	0.00	0.00	0.01
country	0.05	0.02	0.64	0.03	0.00	0.11	0.02	0.02	0.04	0.07
disco	0.02	0.00	0.09	0.58	0.04	0.05	0.02	0.05	0.02	0.13
hiphop	0.01	0.01	0.06	0.05	0.57	0.03	0.02	0.08	0.08	0.08
jazz	0.06	0.03	0.04	0.01	0.01	0.79	0.00	0.00	0.02	0.03
metal	0.04	0.00	0.02	0.01	0.01	0.00	0.88	0.00	0.00	0.04
pop	0.00	0.02	0.09	0.02	0.03	0.01	0.00	0.76	0.01	0.05
reggae	0.06	0.02	0.11	0.03	0.05	0.08	0.00	0.01	0.56	0.07
rock	0.04	0.01	0.10	0.04	0.02	0.06	0.05	0.01	0.02	0.64

The author of the article and python code for which this project is modeled, Marc-Henry Saint-Félix, presented only his best MLP (*mlp_base*). It has five hidden layers (with 512, 512, 256, 128 and 64 nodes) that used the ReLU activation function, a compiler with the RMSProp optimization function, a batch size of 32 and run with 50 epochs. His model's accuracy was 59.84%; when replicated on this project's training and test data, it achieved a slightly lower accuracy rating of 56.86% and processing time of 2.50 minutes. This accuracy rating means that overall, the model correctly classified the genre of 57% of the song samples in the test data. It predicted classical (86% recall) and metal (87%) songs very well, pop (73%) and blues (71%) songs adequately, and relatively poorly (less than 60%) on all other genres (Figure 7).

Figure 7. **Metrics for model *mlp_base*. Training and test accuracy and loss charts; classification report; confusion matrix of recall.**

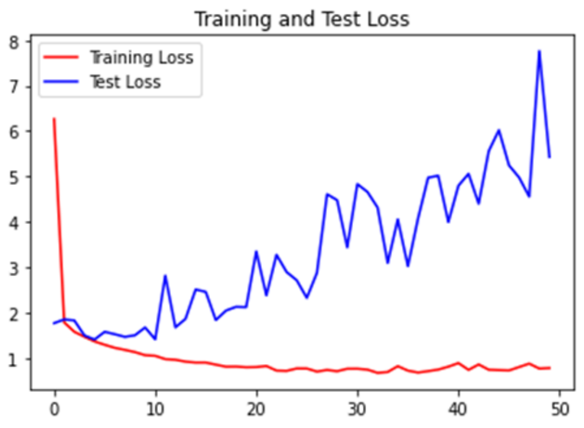


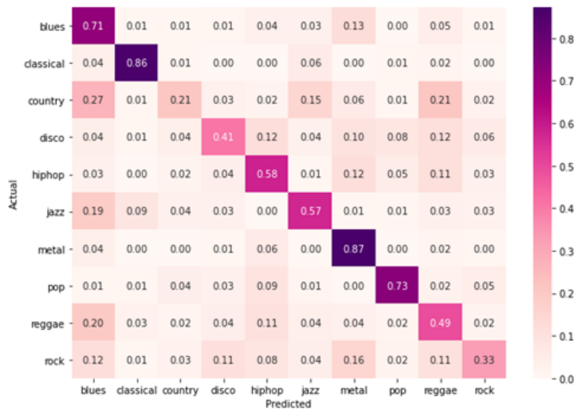


```
              precision    recall  f1-score   support

       blues       0.42      0.71      0.53       295
   classical       0.84      0.86      0.85       299
     country       0.54      0.21      0.30       327
       disco       0.57      0.41      0.48       313
      hiphop       0.54      0.58      0.56       307
        jazz       0.59      0.57      0.58       298
       metal       0.56      0.87      0.68       266
         pop       0.78      0.73      0.75       285
      reggae       0.42      0.49      0.45       306
        rock       0.59      0.33      0.42       301

    accuracy                           0.57      2997
   macro avg       0.58      0.58      0.56      2997
weighted avg       0.58      0.57      0.55      2997
```





	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.71	0.01	0.01	0.01	0.04	0.03	0.13	0.00	0.05	0.01
classical	0.04	0.86	0.01	0.00	0.00	0.06	0.00	0.01	0.02	0.00
country	0.27	0.01	0.21	0.03	0.02	0.15	0.06	0.01	0.21	0.02
disco	0.04	0.01	0.04	0.41	0.12	0.04	0.10	0.08	0.12	0.06
hiphop	0.03	0.00	0.02	0.04	0.58	0.01	0.12	0.05	0.11	0.03
jazz	0.19	0.09	0.04	0.03	0.00	0.57	0.01	0.01	0.03	0.03
metal	0.04	0.00	0.00	0.01	0.06	0.00	0.87	0.00	0.02	0.00
pop	0.01	0.01	0.04	0.03	0.09	0.01	0.00	0.73	0.02	0.05
reggae	0.20	0.03	0.02	0.04	0.11	0.04	0.04	0.02	0.49	0.02
rock	0.12	0.01	0.03	0.11	0.08	0.04	0.16	0.02	0.11	0.33

Both best RMSProp optimized (*mlp_12.1*, 63% accuracy) and Adam optimized (*mlp_27*, 71% accuracy) MLPs created for this project outperformed the base model provided by Marc-Henry Saint-Félix (*mlp_base*, 57% accuracy). The former two models have a much greater number of epochs 200 and subsequent processing times of 15.82 and 13.77 minutes respectively, compared to the 50 epochs and

2.5 minute processing time of base model. (The *mlp_base* model was run at 100 epochs, but the accuracy rate plummeted to 14.18%.)

While the overall accuracy of the models is important, more so is the recall rate of each class. Recall is essentially the accuracy rate of each class. A low variation of recall rates indicates a model that correctly classifies all classes well and one that will likely perform better on an unseen music sample. A high variation indicates that the model correctly classifies some classes very well and others poorly. Not only is the overall accuracy of *mlp_27* the best of the Multiple Layer Perceptrons, but it's recall variance is also the lowest at 1.5% (Table 3). Further, it correctly classified all but two genres of music (hip-hop, pop) at better rates than *mlp_12.1* and *mlp_base*. In summary, this model is a significantly better classifier than the baseline MLP model.

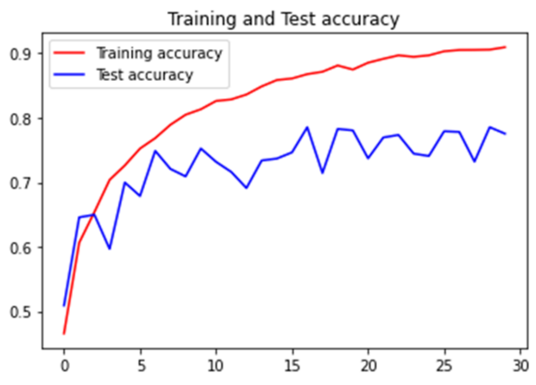
Table 3. Recall (accuracy by class) rates for the best RMSProp and Adam optimized MLP models and base model.

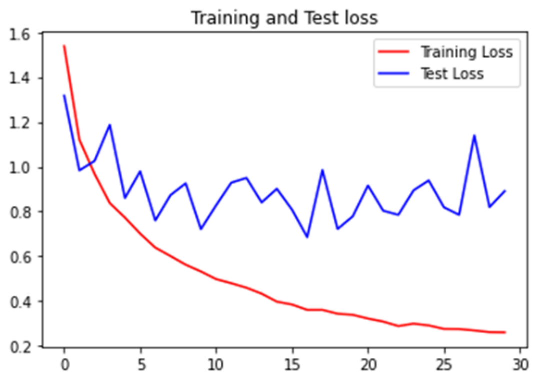
Model Name		***mlp_12.1***	***mlp_27***	***mlp_base***
Activation Function		***Sigmoid***	***Relu***	***Relu***
Optimization Function		***RMSProp***	***Adam***	***RMSProp***
Epochs		***200***	***200***	***50***
	Test Samples	recall	recall	recall
blues	295	59%	79%	71%
classical	299	90%	90%	86%
country	327	57%	64%	21%
disco	313	50%	58%	41%
hiphop	307	46%	57%	58%
jazz	298	76%	79%	57%
metal	266	80%	88%	87%
pop	285	79%	76%	73%
reggae	306	53%	56%	49%
rock	301	42%	64%	33%
Accuracy		63%	71%	57%
Variance		2.5%	1.5%	4.4%
Processing Time (m)		15.82	13.77	2.50

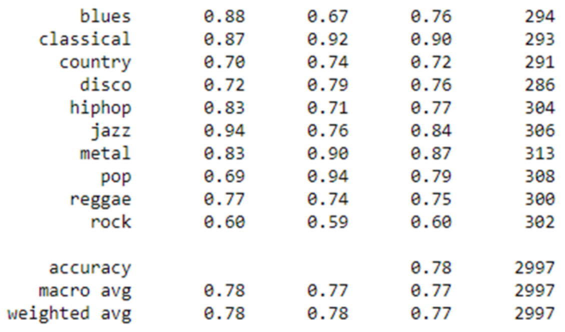
Convolutional Neural Network

Marc-Henry Saint-Félix's base CNN model (*cnn_base*) utilizes a series of three convolutional sequences (2x ((3x3 kernel with ReLU activation function), pooling, normalization), 1x ((2x2 kernel with ReLU activation function), pooling, normalization, dropout 30%)) before a flattering layer, followed by a single Dense layer with 64 nodes and a ReLU activation function, a Softmax output layer and a compiler with the RMSProp optimization function and learning rate of 0.001. With a batch size of 32 and 30 epochs, the baseline performance reported by the author is 77.83% accuracy that, when replicated, achieved 77.54% accuracy and a processing time of 6.24 minutes. (It is noted that while 78% accuracy is the baseline for CNN performance, simply increasing the number of epochs to 50 (*cnn_base_2*) improved the model's accuracy to 79.78% with a 10.77 minute processing time). This accuracy rating means that overall, the model correctly classified the genre of 78% of the song samples in the test data. It predicted classical (92% recall) pop (94%) and metal (90%) songs very well, disco (79%), jazz (76%), country (74%) and reggae (74%) songs adequately, and relatively poorly (less than 60%) on rock songs (Figure 8).

Figure 8. **Metrics for model *cnn_base*. Training and test accuracy and loss charts; classification report; confusion matrix of recall.**

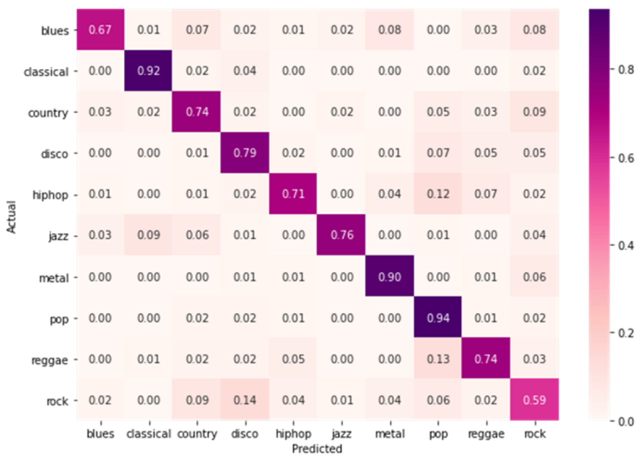






```
       blues       0.88      0.67      0.76       294
   classical       0.87      0.92      0.90       293
     country       0.70      0.74      0.72       291
       disco       0.72      0.79      0.76       286
      hiphop       0.83      0.71      0.77       304
        jazz       0.94      0.76      0.84       306
       metal       0.83      0.90      0.87       313
         pop       0.69      0.94      0.79       308
      reggae       0.77      0.74      0.75       300
        rock       0.60      0.59      0.60       302

    accuracy                           0.78      2997
   macro avg       0.78      0.77      0.77      2997
weighted avg       0.78      0.78      0.77      2997
```



Actual \ Predicted	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.67	0.01	0.07	0.02	0.01	0.02	0.08	0.00	0.03	0.08
classical	0.00	0.92	0.02	0.04	0.00	0.00	0.00	0.00	0.00	0.02
country	0.03	0.02	0.74	0.02	0.00	0.02	0.00	0.05	0.03	0.09
disco	0.00	0.00	0.01	0.79	0.02	0.00	0.01	0.07	0.05	0.05
hiphop	0.01	0.00	0.01	0.02	0.71	0.00	0.04	0.12	0.07	0.02
jazz	0.03	0.09	0.06	0.01	0.00	0.76	0.00	0.01	0.00	0.04
metal	0.00	0.00	0.00	0.01	0.01	0.00	0.90	0.00	0.01	0.06
pop	0.00	0.00	0.02	0.02	0.01	0.00	0.00	0.94	0.01	0.02
reggae	0.00	0.01	0.02	0.02	0.05	0.00	0.00	0.13	0.74	0.03
rock	0.02	0.00	0.09	0.14	0.04	0.01	0.04	0.06	0.02	0.59

As these base models utilize the ReLU activation function and RMSProp optimization function, three series of models were created with different variations of these components to seek better accuracy performance. All models perform one or more convolutional sequences prior to a flattening layer that then feeds zero, one or two Dense layers prior to the Softmax output layer.

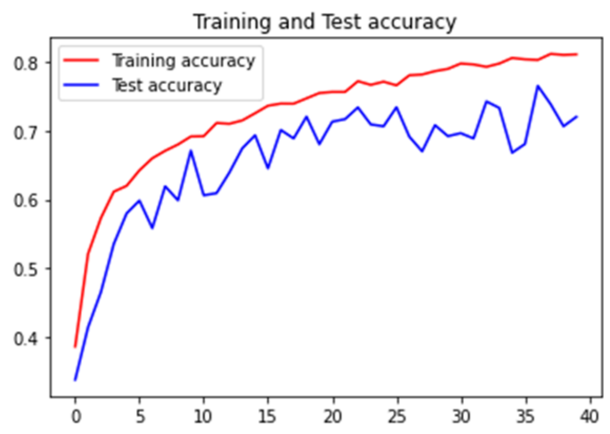
Table 4. Summary of CNN models with Sigmoid activation functions, the RMSProp optimization function, Batch Size of 32 and 0.001 Learning Rate. Base models use ReLU activation functions and the RMSProp optimization function.

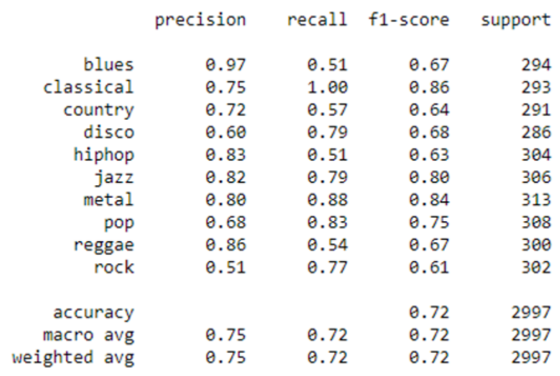
Model Name	Hidden Layers	Convolution Sequences	Dense Layer Nodes	Epochs	Proc. Time	Accuracy Rate %
cnn_base _01	11	2x (3x3 kernel, pooling, normalization), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	30	6.24	77.54
cnn_base _02	11	2x (3x3 kernel, pooling, normalization), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	50	10.77	79.78
cnn_01.0	1	1x (3x3 kernel)	NA	50	7.72	54.52
cnn_01.1	2	1x (3x3 kernel, pooling)	NA	50	8.67	57.16
cnn_01.2	5	1x (3x3 kernel, pooling, normalization, dropout 30%)	64	30	6.88	62.60
cnn_01.3	9	2x (3x3 kernel, pooling, normalization, dropout 30%)	64	30	8.50	65.60
cnn_01.4	13	2x (3x3 kernel, pooling, normalization, dropout 30%), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	30	8.90	67.97
cnn_01.5	14	2x (3x3 kernel, pooling, normalization, dropout 30%), 1x (2x2 kernel, pooling, normalization, dropout 30%)	128, 64	30	8.07	70.67
cnn_01.6	14	2x (3x3 kernel, pooling, normalization, dropout 30%), 1x (2x2 kernel, pooling, normalization, dropout 30%)	128, 64	50	15.64	71.04
cnn_01.7	15	2x (3x3 kernel, pooling, normalization, dropout 30%), 1x (2x2 kernel, pooling, normalization, dropout 30%)	128, 64	40	11.80	72.01

The *cnn_01* series models (Table 4) utilize Sigmoid activation functions and the RMSProp optimization function. Starting with a single convolutional layer and zero Dense layers with 50 epochs, *cnn_01.0* achieves just 54.52% accuracy. Adding a pooling layer to the convolutional sequence (*cnn_01.1*) improves accuracy to 57.16%. Adding a normalizing and a dropout (30%) layer to the convolutional sequence (*cnn_01.2*), a single Dense layer with 64 nodes, and reducing the number of epochs from 50 to 30 improves accuracy to 62.60%. Model *cnn_01.3* is the same as *cnn_01.2* but duplicates the same convolutional sequence and increases the accuracy to 65.60%. Adding a third convolutional sequence (same format, but with a 2x2 kernel) to create *cnn_01.4* further improved the accuracy rating to 67.97%.

The next model trialed (*cnn_01.5*) included an additional Dense layer with 128 nodes that preceded the one with 64 nodes; this change increased accuracy to 70.67%. Models *cnn_01.6* and *cnn_01.7* served to tune the previous model by increasing the number of epochs to 50 and then reducing them to 40, respectively. Model *cnn_01.7* performed the best of the *cnn_01* series by achieving an accuracy rating of 72.01% with a processing time of 11.80 minutes. This accuracy rating means that overall, the model correctly classified the genre of 72% of the song samples in the test data. It predicted classical (100% recall) perfectly, metal (90%) and pop (83%) songs very well, disco (79%), jazz (79%), and rock (77%) songs adequately, and relatively poorly (less than 60%) on blues, country, hip-hop and reggae songs (Figure 9). However, it did not outperform the CNN baseline of 78% accuracy.

Figure 9. Metrics for model *cnn_01.7*. Training and test accuracy and loss charts; classification report; confusion matrix of recall.

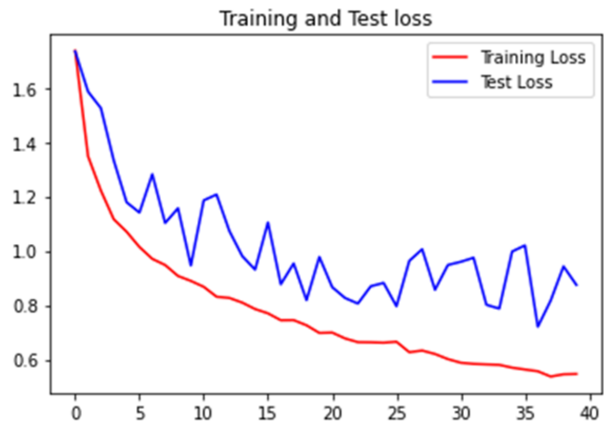


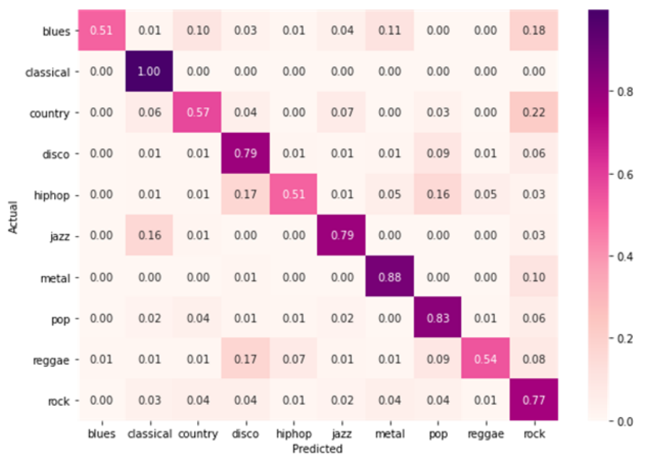


```
              precision    recall  f1-score   support

       blues       0.97      0.51      0.67       294
   classical       0.75      1.00      0.86       293
     country       0.72      0.57      0.64       291
       disco       0.60      0.79      0.68       286
      hiphop       0.83      0.51      0.63       304
        jazz       0.82      0.79      0.80       306
       metal       0.80      0.88      0.84       313
         pop       0.68      0.83      0.75       308
      reggae       0.86      0.54      0.67       300
        rock       0.51      0.77      0.61       302

    accuracy                           0.72      2997
   macro avg       0.75      0.72      0.72      2997
weighted avg       0.75      0.72      0.72      2997
```





	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.51	0.01	0.10	0.03	0.01	0.04	0.11	0.00	0.00	0.18
classical	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
country	0.00	0.06	0.57	0.04	0.00	0.07	0.00	0.03	0.00	0.22
disco	0.00	0.01	0.01	0.79	0.01	0.01	0.01	0.09	0.01	0.06
hiphop	0.00	0.01	0.01	0.17	0.51	0.01	0.05	0.16	0.05	0.03
jazz	0.00	0.16	0.01	0.00	0.00	0.79	0.00	0.00	0.00	0.03
metal	0.00	0.00	0.00	0.01	0.00	0.00	0.88	0.00	0.00	0.10
pop	0.00	0.02	0.04	0.01	0.01	0.02	0.00	0.83	0.01	0.06
reggae	0.01	0.01	0.01	0.17	0.07	0.01	0.01	0.09	0.54	0.08
rock	0.00	0.03	0.04	0.04	0.01	0.02	0.04	0.04	0.01	0.77

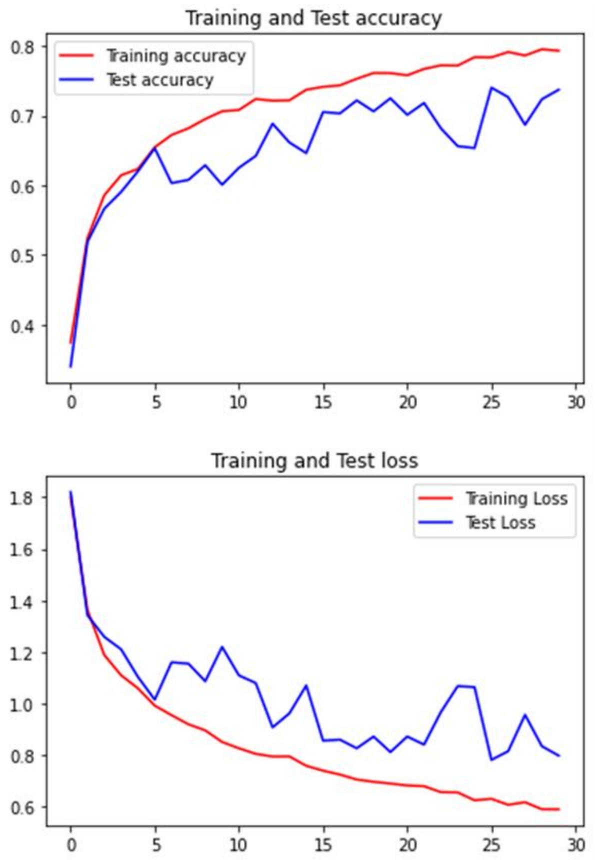
The *cnn_02* series models (Table 5) utilize Sigmoid activation functions and the Adam optimization function. Starting with a single convolutional layer and zero Dense layers with 50 epochs, *cnn_02.0* achieves just 54.42% accuracy. Adding a pooling layer to the convolutional sequence and reducing the number of epochs from 50 to 30 (*cnn_02.1*) improved accuracy to 60.53%. Adding a normalizing and a dropout (30%) layer to the convolutional sequence and a single Dense layer with 64 nodes (*cnn_02.2*)

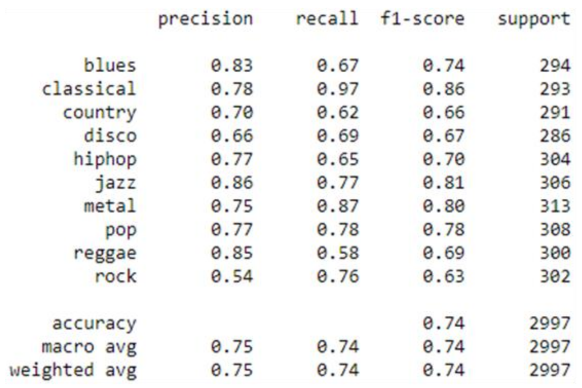
Table 5. Summary of CNN models with Sigmoid activation functions, the Adam optimization function, Batch Size of 32 and 0.001 Learning Rate Base models use ReLU activation functions and the RMSProp optimization function.

Model Name	Hidden Layers	Convolution Sequences	Dense Layer Nodes	Epochs	Proc. Time	Accuracy Rate %
cnn_base _01	11	2x (3x3 kernel, pooling, normalization), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	30	6.24	77.54
cnn_base _02	11	2x (3x3 kernel, pooling, normalization), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	50	10.77	79.78
cnn_02	1	1x (3x3 kernel)	NA	50	7.29	54.42
cnn_02.1	2	1x (3x3 kernel, pooling)	NA	30	3.83	60.53
cnn_02.2	5	1x (3x3 kernel, pooling, normalization, dropout 30%)	64	30	6.10	62.50
cnn_02.3	9	2x (3x3 kernel, pooling, normalization, dropout 30%)	64	30	5.80	69.67
cnn_02.4	9	2x (3x3 kernel, pooling, normalization, dropout 30%)	64	50	15.41	67.97
cnn_02.5	13	2x (3x3 kernel, pooling, normalization, dropout 30%), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	30	8.12	69.67
cnn_02.6	13	2x (3x3 kernel, pooling, normalization, dropout 30%), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	50	7.88	70.70
cnn_02.7	14	2x (3x3 kernel, pooling, normalization, dropout 30%), 1x (2x2 kernel, pooling, normalization, dropout 30%)	128, 64	30	8.10	73.71
cnn_02.8	14	2x (3x3 kernel, pooling, normalization, dropout 30%), 1x (2x2 kernel, pooling, normalization, dropout 30%)	128, 64	50	13.65	72.77
cnn_02.9	14	2x (3x3 kernel, pooling, normalization, dropout 40%, 30%), 1x (2x2 kernel, pooling, normalization, dropout 20%)	128, 64	30	8.05	73.24

improves accuracy to 62.50%. Model *cnn_02.3* is the same as *cnn_02.2* but duplicates the same convolutional sequence and increases the accuracy to 69.67%; however, increasing the number of nodes to 50 (*cnn_02.04*) saw a reduction in accuracy (67.97%). Adding a third convolutional sequence (same format, but with a 2x2 kernel) and reducing the number of epochs to 30 to create *cnn_02.5* further improved the accuracy rating to 69.67%, and to 70.70% with 50 epochs (*cnn_02.6*). The next model trialed (*cnn_02.7*) included an additional Dense layer with 128 nodes that preceded the one with 64 nodes; this change increased accuracy to 73.71%; a reduction to 30 epochs (*cnn_02.8*) also saw a reduction in accuracy (72.77%). Model *cnn_02.9* served to trial different dropout rates of model *cnn_02.7* by changing the rates to 40% and 20% in the first and third convolutional sequences, respectively; performance did not improve (73.24% accuracy). Model *cnn_02.7* performed the best of

Figure 10. Metrics for model *cnn_02.7*. Training and test accuracy and loss charts; classification report; confusion matrix of recall.

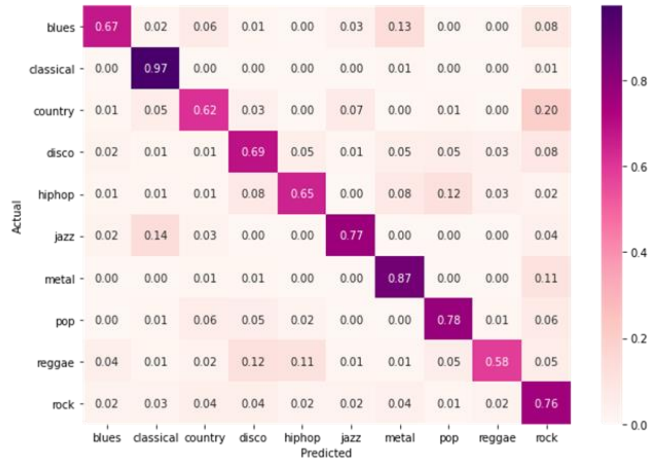




```
              precision    recall  f1-score   support

       blues       0.83      0.67      0.74       294
   classical       0.78      0.97      0.86       293
     country       0.70      0.62      0.66       291
       disco       0.66      0.69      0.67       286
      hiphop       0.77      0.65      0.70       304
        jazz       0.86      0.77      0.81       306
       metal       0.75      0.87      0.80       313
         pop       0.77      0.78      0.78       308
      reggae       0.85      0.58      0.69       300
        rock       0.54      0.76      0.63       302

    accuracy                           0.74      2997
   macro avg       0.75      0.74      0.74      2997
weighted avg       0.75      0.74      0.74      2997
```



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.67	0.02	0.06	0.01	0.00	0.03	0.13	0.00	0.00	0.08
classical	0.00	0.97	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01
country	0.01	0.05	0.62	0.03	0.00	0.07	0.00	0.01	0.00	0.20
disco	0.02	0.01	0.01	0.69	0.05	0.01	0.05	0.05	0.03	0.08
hiphop	0.01	0.01	0.01	0.08	0.65	0.00	0.08	0.12	0.03	0.02
jazz	0.02	0.14	0.03	0.00	0.00	0.77	0.00	0.00	0.00	0.04
metal	0.00	0.00	0.01	0.01	0.00	0.00	0.87	0.00	0.00	0.11
pop	0.00	0.01	0.06	0.05	0.02	0.00	0.00	0.78	0.01	0.06
reggae	0.04	0.01	0.02	0.12	0.11	0.01	0.01	0.05	0.58	0.05
rock	0.02	0.03	0.04	0.04	0.02	0.02	0.04	0.01	0.02	0.76

the *cnn_02* series by achieving an accuracy rating of 73.71% with a processing time of 8.10 minutes. This accuracy rating means that overall, the model correctly classified the genre of 74% of the song samples in the test data. It predicted classical (97% recall) and metal (87%) songs very well, pop (78%) and jazz (77%) songs adequately, and relatively poorly (less than 60%) on just reggae songs (Figure 10).

And while it outperformed the best model from the *cnn_01* series, it still did not outperform the CNN baseline of 78% accuracy.

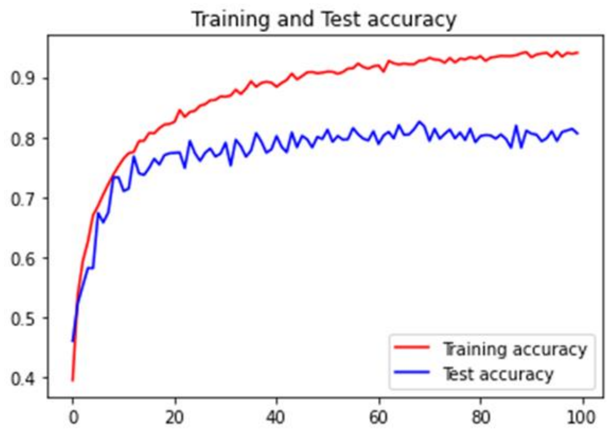
The *cnn_03* series models utilize ReLU activation functions and the Adam optimization function. Starting with a single convolutional layer and zero Dense layers with 50 epochs, *cnn_03.0* achieves just 50.85% accuracy. Adding a pooling layer to the convolutional sequence (*cnn_03.1*) reduced accuracy to 50.28%. Adding a normalizing and a dropout (30%) layer to the convolutional sequence (*cnn_03.2*) and a single

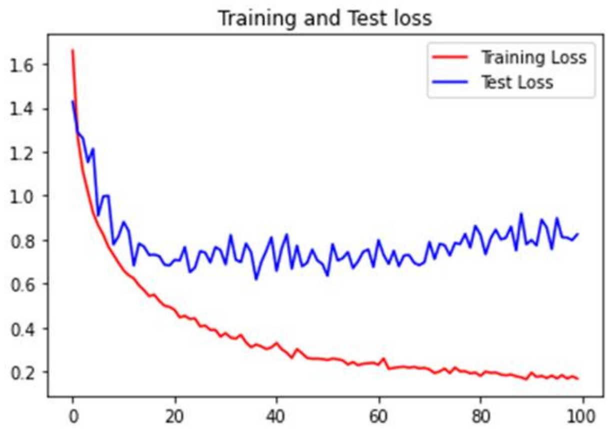
Table 6. Summary of CNN models with ReLU activation functions, the Adam optimization function, Batch Size of 32 and 0.001 Learning Rate. Base models use ReLU activation functions and the RMSProp optimization function.

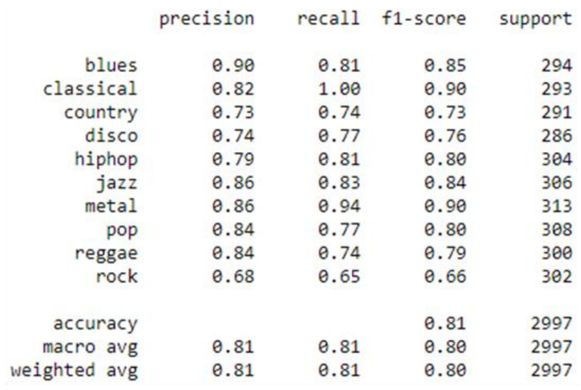
Model Name	Hidden Layers	Convolution Sequences	Dense Layer Nodes	Epochs	Proc. Time	Accuracy Rate %
cnn_base _01	11	2x (3x3 kernel, pooling, normalization), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	30	6.24	77.54
cnn_base _02	11	2x (3x3 kernel, pooling, normalization), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	50	10.77	79.78
cnn_03.0	1	1x (3x3 kernel)	NA	50	5.91	50.85
cnn_03.1	2	1x (3x3 kernel, pooling)	NA	50	7.98	50.28
cnn_03.2	5	1x (3x3 kernel, pooling, normalization, dropout 30%)	64	50	12.77	63.40
cnn_03.3	9	2x (3x3 kernel, pooling, normalization, dropout 30%)	64	30	9.75	74.67
cnn_03.4	13	2x (3x3 kernel, pooling, normalization, dropout 30%), 1x (2x2 kernel, pooling, normalization, dropout 30%)	64	30	6.67	76.81
cnn_03.5	13	2x (3x3 kernel, pooling, normalization, dropout 40%, 30%), 1x (2x2 kernel, pooling, normalization, dropout 20%)	64	30	5.93	77.54
cnn_03.6	14	2x (3x3 kernel, pooling, normalization, dropout 40%, 30%), 1x (2x2 kernel, pooling, normalization, dropout 20%)	128, 64	50	10.26	77.68
cnn_03.7	14	2x (3x3 kernel, pooling, normalization, dropout 40%, 30%), 1x (2x2 kernel, pooling, normalization, dropout 20%)	128, 64	80	17.67	79.65
cnn_03.8	14	2x (3x3 kernel, pooling, normalization, dropout 40%, 30%), 1x (2x2 kernel, pooling, normalization, dropout 20%)	128, 64	100	25.69	80.61

Dense layer with 64 nodes significantly improves accuracy to 63.40%. Model *cnn_03.3* is the same as *cnn_03.2* but duplicates the same convolutional sequence and increases the accuracy to 65.60% when the number of epochs is reduced from 50 to 30. Adding a third convolutional sequence (same format, but with a 2x2 kernel) to create *cnn_03.4* further improved the accuracy rating to 76.81%. Model *cnn_03.5* served to trial different dropout rates of model *cnn_02.4* by changing the rates to 40% and 20% in the first and third convolutional sequences, respectively; performance improved to 77.54% accuracy. The next model trialed (*cnn_03.6*) included an additional Dense layer with 128 nodes that preceded the one with 64 nodes, as well as an increase in the number of epochs to 50; these changes only slightly increased accuracy to 77.68%. Models *cnn_03.7* and *cnn_03.8* served to tune the previous model, increasing the number of epochs to 80 and 100 achieved accuracy ratings of 79.65% and 80.61%, respectively.

Figure 11. Metrics for model *cnn_02.7*. Training and test accuracy and loss charts; classification report; confusion matrix of recall



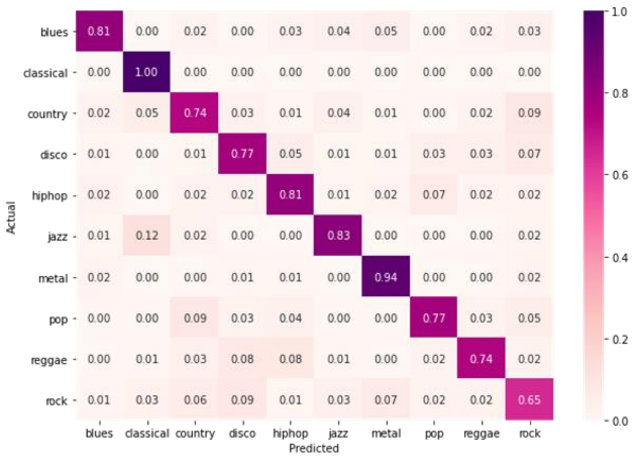




```
              precision    recall  f1-score   support

       blues       0.90      0.81      0.85       294
   classical       0.82      1.00      0.90       293
     country       0.73      0.74      0.73       291
       disco       0.74      0.77      0.76       286
      hiphop       0.79      0.81      0.80       304
        jazz       0.86      0.83      0.84       306
       metal       0.86      0.94      0.90       313
         pop       0.84      0.77      0.80       308
      reggae       0.84      0.74      0.79       300
        rock       0.68      0.65      0.66       302

    accuracy                           0.81      2997
   macro avg       0.81      0.81      0.80      2997
weighted avg       0.81      0.81      0.80      2997
```



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.81	0.00	0.02	0.00	0.03	0.04	0.05	0.00	0.02	0.03
classical	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
country	0.02	0.05	0.74	0.03	0.01	0.04	0.01	0.00	0.02	0.09
disco	0.01	0.00	0.01	0.77	0.05	0.01	0.01	0.03	0.03	0.07
hiphop	0.02	0.00	0.02	0.02	0.81	0.01	0.02	0.07	0.02	0.02
jazz	0.01	0.12	0.02	0.00	0.00	0.83	0.00	0.00	0.00	0.02
metal	0.02	0.00	0.00	0.01	0.01	0.00	0.94	0.00	0.00	0.02
pop	0.00	0.00	0.09	0.03	0.04	0.00	0.00	0.77	0.03	0.05
reggae	0.00	0.01	0.03	0.08	0.08	0.01	0.00	0.02	0.74	0.02
rock	0.01	0.03	0.06	0.09	0.01	0.03	0.07	0.02	0.02	0.65

Model *cnn_03.8* performed the best of the *cnn*_03 series with an accuracy rating of 80.61% with a long processing time of 25.69 minutes. This accuracy rating means that overall, the model correctly classified the genre of 81% of the song samples in the test data. It predicted classical (100% recall) songs perfectly, metal (94%) songs very well, jazz (83%), blues (81%), hiphop (81%), disco (77%), pop (77%), country (74%) and reggae (74%) adequately, and relatively poorly (65%) on just rock songs (Figure 11). Further, the 81% accuracy rating means that *cnn_03.8* outperformed both the baseline (*cnn_base_01*, 78% accuracy), modified baseline (*cnn_base_02*) models (Table 7) as well as all of the other CNNs and MLPs.

Table 7. Recall (accuracy by class) rates for the best CNN models and base models.

Model Name		***cnn_01.7***	***cnn_02.7***	***cnn_03.8***	***cnn_base_01***	***cnn_base_02***
Activation Function		***Sigmoid***	***Sigmoid***	***Relu***	***Relu***	***Relu***
Optimization Function		***RMSProp***	***Adam***	***Adam***	***RMSProp***	***RMSProp***
Epochs		***40***	***30***	***100***	***30***	***50***
	Test Samples	recall	recall	recall	recall	recall
blues	294	51%	67%	81%	67%	80%
classical	293	100%	97%	100%	92%	99%
country	291	57%	62%	74%	74%	80%
disco	286	79%	69%	77%	79%	71%
hiphop	304	51%	65%	81%	71%	75%
jazz	306	79%	77%	83%	76%	81%
metal	313	88%	87%	94%	90%	88%
pop	308	83%	78%	77%	94%	74%
reggae	300	54%	58%	74%	74%	81%
rock	302	77%	76%	65%	59%	69%
Accuracy		72%	74%	81%	78%	80%
Variance		2.7%	1.3%	0.9%	1.2%	0.7%
Processing Time (m)		11.80	8.10	25.69	6.24	10.77

While a best model was declared, proper evaluation of all models should be performed to confirm this statement. Cross validation of the data and/or many iterations of other sets of training and test data would be required to determine the average accuracy rates of the models and whether their performances are significantly different enough to determine which is best.

In each series of MLPs and CNNs, each additional layer usually improved on its predecessor model immediately or once tuned. Building a deeper neural network proved to increase performance. This is reasonable as there is much variation within music genres and much in common between them. Much needs to be learned from the samples in order to properly classify them. Much also needs to filtered, focused, and forgotten as evidenced by the improved performance of models that included increasing numbers of convolutional, pooling (and normalization) and dropout layers, respectively; it is reasonable that the CNNs performed better than the MLPs overall. In terms of activation functions, the best models were those that relied on ReLU; as noted earlier, the Sigmoid activation function's performance suffers in deeper neural networks due to the vanishing gradient problem. Both the Adam and RMSProp optimization functions generated models of comparable accuracy such that their performance cannot be fully evaluated.

To build a better classifier than those presented here, there are several recommended approaches. The first is to increase the sample size. While ten thousand samples seem like an adequate size, they still represent just one thousand songs, only 100 from each genre, and only thirty seconds of each of those. The relatively small sample size for each genre span decades – sometimes over a century – of that style of music. While they retain many of their signatures, music genres have gone through periods of changes and metamorphoses over their histories. Pop music from the 1950s sounds different than that of today. And rock music has come a long way from Little Richard and Elvis Presley. In fact, the rock genre was usually amongst the worst recall ratings for the better classifiers. Building a training data set with a higher percentage of rock songs could aid in improving the models' performance for that class.

Conversely, classical music was usually the models' best performing genre. This is reasonable as it is very different from the other relatively contemporary genres in terms of instrumentation (strings, woodwinds, horns), lack of vocals, variation of tempo, etc. Metal was also usually among the models' better performing genres. This is reasonable once again as metal is an extreme version of rock with a consistent style that is guitar-heavy and loud and without much influence of other music genres.

The concept of influence is something that makes music interesting. Rock was borne from blues and country and any given song can sound more like one of its parents' genres; or it can borrow themes from pop, or vice versa. Reggae can cross with hip-hop. Country can cross with hip-hop or pop, or both. Many music genres are hybridized and hyphenated, such that the gray areas between the music genres discussed here are large. This will always make building a music classifier for just ten genres a difficult task from the onset. Building up a sample set geared towards past models' genre weaknesses could help overcome them while saving time and memory by reducing the number of samples of genres with higher recall.

In terms of the actual classifiers, further varying their architectures and tuning the hyperparameters could result in a model with an accuracy rating greater than 81%. The procedure of progressive trial and error was effective. As noted, in each series of MLPs and CNNs, each additional layer usually improved on its predecessor model immediately or once tuned. Over sixty models were built and evaluated for this project with a relatively limited amount of variation. More Dense layers can be added, and their number of nodes varied. Mixing and/or introducing other types of activation functions can be trialed; other optimization functions can be introduced and trialed. Varying the order of pooling, normalizing and dropout layers within the convolutional sequences can be varied; dropout rates can be tuned as well. Batch sizes and learning rates were held constant, but these hyperparameters can be tuned as well. Attention to each variation can generate more models to evaluate; the number of configurations is virtually endless. With enough attempts, it is likely that a music genre classifier with a higher accuracy rating can be created.

Jim Sears
ADS654Z1 - Deep Learning
Final Project Proposal
11/21/21

References:

Bindal, A. (2019, February 10). *Normalization Techniques in Deep Neural Networks*. Medium. https://medium.com/techspace-usict/normalization-techniques-in-deep-neural-networks-9121bf100d8

Brownlee, Jason. (2019, January 9). *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. Deep Learning Performance. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Brownlee, Jason. (2020, September 12). *Understand the Impact of Learning Rate on Neural Network Performance*. Deep Learning Performance. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Little, Z2. (2020, May 17). *Activation Functions (Linear/Non-linear) in Deep Learning*. Medium. https://xzz201920.medium.com/activation-functions-linear-non-linear-in-deep-learning-relu-sigmoid-softmax-swish-leaky-relu-a6333be712ea

Moustafa, R. (2021). *Lecture 3: Deep Learning Using Keras* [PowerPoint slides].Bay Path University Deep Learning. Canvas: https://baypath.instructure.com/courses/1382355/pages/lecture03-multilayernnt-keras-tf?module_item_id=18491199

Moustafa, R. (2021). *Lecture 6: Convolutional Neural Networks (CNNs)* [PowerPoint slides].Bay Path University Deep Learning. Canvas: https://baypath.instructure.com/courses/1382355/pages/week-6-lecture-cnn?module_item_id=18491226

Olteanu, A. (Version1, Updated 2020, March 24). *GTZAN Dataset - Music Genre Classification*. Kaggle. https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification

Saint-Félix, M. (2020, July). *Music Genre Detection With Deep Learning*. Towards Data Science. https://towardsdatascience.com/music-genre-detection-with-deep-learning-cf89e4cb2ecc
Related GitHub Repository: https://github.com/msaintfelix/TensorFlow_MusicGenre_Classifier