Anthony Galczak
Tristin Glunt
CS529 - Project 3
Music Genre Classification

**I. Introduction**

The following report is our analysis and classification of the music-genre dataset. The dataset consists of 900 genre labeled training examples, where there are 10 classes of genres and where an example is a 30s audio clip. There are 100 validation audio clips, where our goal is to classify each unknown validation file with a genre. We use three feature extractions: converting .au files to colored spectrograms, converting .au files to mel-spectrograms, and converting the files to find the central-spectroid of each clip. We use two classifiers, the first being the canonical artificial neural network (ANN) with 5 fully-connected layers and the second being a deep convolutional neural network (CNN). Both classifiers use SGD with momentum, learning rate decay, and nesterov optimization. We conclude the report with a comparison of the two classifiers and our overall findings of using different feature extractors.

**II. Data**

**a. Analysis / Setup**

The most interesting aspect of this project is how we represent the music files. We decided to initially represent the files as images. To represent the music files as images, we plot the files as a spectrogram, this gives us a fairly unique image for every audio file.

At first we were using specgram from matplotlib. This generated us some color spectrograms that mostly came out as greenish-yellow. These pictures looked pretty low in contrast and this was reflected in our accuracy on validation and testing data. We were getting around 40-50% accuracy with this data. At this point we definitely thought it was time to do something else with our data.

Since we are using 3 channels (RGB) on our color spectrograms, but we are only visually seeing green and yellow we might as well use grey-scale spectrograms. So at this point, we switched to scipy.signal.spectrogram to make use some grey-scale spectrograms. In addition to doing mono-channel spectrograms we decided to split our images into various clips instead of giant 30 second clips. A major difficult of this data is that we only have 1000 total sound files so we need to somehow get more data. We have grey-scale data splits of 1s (30 files per sound file), 2s (15 files), 3s, 5s, and 10s. After doing some additional research on sound analysis, it appears that 3s clips are somewhat the standard so we mostly used these clips for training our networks. We train on the 3 second clips and then at test time we classify each 3 second clip. After they are all classified, there is a voting mechanism among the 10 clips to come to a classification for the full 30s clip. This process is very similar to the voting process that is done within random forests. This got us to 70% accuracy on the testing data.

After doing some more research, it turns out the “gold-standard” for audio classification is making color mel spectrograms. I can’t begin to explain the full theory of how mel works, but it is some way of

capturing the most significant parts of the signal (frequency, db, etc.). So far, we have achieved 70% with the mel spectrograms, but we are looking to make more advanced networks to better utilize this data.

Doing data augmentation has heightened our testing accuracy to 77%. Data augmentation includes: mirroring the images, rotating the images vertically and horizontally, stretching them slightly, blurring the images, and zooming in on images. Currently, this is in-line with our data, meaning some of our data is being construed like this based on some probability. Ideally, we would like to do this but instead of construing our current data, save each new augmented image as a new image that's labeled. This will grow our training data, and by growing the data, we believe our accuracy could increase even more.

For our third feature extractor, we implemented librosa's central-spectroid. We then ran the data through our ANN network. Unfortunately, this feature did not seem to represent the data as well as our pixel-images or spectrograms. This was illustrated by our accuracy on the validation sets which was around 40%. Thus, we decided to not do further analysis on it and went with images and mel-spectrograms to try and get the best accuracy we could.

**b. Bias**

An interesting part about music genre classification is that it is an always evolving field and falls prey to some subjectivity. Our confusion matrices showed common misclassifications for pop, rock, and metal. The problem with these classifications is that it is pretty difficult, even as a human listener, to truly identify the line between pop and rock or rock and metal. Thus, an argument for bias in the data is that patterns and concepts may be shared among two songs that don't directly classify in the same genre. For instance, a really slow rock song could be mistaken for a pop song.

Regular research is done in music genre classification and it is kind of a slippery slope to really identify one song from another. I found some very useful articles and blog posts from Spotify researchers (link) that showed the various features and separations between music genre data. An important part to note about this type of research is that often what classifies one song from another isn't necessarily a genre separation. Therefore, classifying music genres is a bit more subjective and fuzzy than, for example, detecting a particular gene expression in a genome.

**III. Classifiers**

**a. Artificial Neural Network**

Using neural networks to classify images is a well-known solution to the classic problem of image classification. Neural networks work extremely well for image classification specifically because of the way they can automatically extract features from the images based on pixel values. The ANN extracts features for us by tuning weights with gradient descent, each gradient helping push the weights towards the correct value for what that neuron is looking at. Although it is extremely difficult to tell what the network is learning, it is clear that it is learning by the increasing accuracy in the training dataset and the split dataset.

The overall architecture of our network is a deep 5-layer neural network with relu activations in the hidden layers and a 10 neuron output layer using the softmax activation. The optimizer used is Adam, which takes advantage of momentum and RMS prop. We first flatten the data, that is, we make it a 1D array to feed into the ANN as the first input layer. The input layer is then connected to a 512 neuron layer, which is connected to a 256 neuron layer, which is connected to a 128 neuron layer, connected to a 64 neuron layer, finally connected to the softmax output layer.

The network is being trained by using a minibatch of size 32, with usually 100 training epochs. This helps our network taking in a group of inputs and then updating the weights of the network. A mini-batch gradient descent is is important so the weights don't jump all over due to slightly different gradients.

The ANN is a good immediate classifier due to the way it extracts features without us needing to determine any on our own. We can simply pass in the pixel values from the spectrograms and the depth and size of the ANN learns due to the stochastic-gradient descent learning algorithm.

**b. Convolutional Neural Network**

Although the canonical models of neural networks work quite well, it has been discovered that using convolution functions on images works even better to extract features, making it the goto classifier for image classification. It uses small filters, usually 3x3, and proceeds to cover the entire image applying convolutions using the filter values. The filters in this case contain the weights the network is trying to learn, and most networks have an ensemble of filters each learning to detect different features. Some filters will detect vertical edges, other will detect diagonal edges going from left-to-right and so on. As you add more convolution layers, the filters begin detecting higher and higher abstractions of features, until eventually some filters are looking for such extremely abstract features.

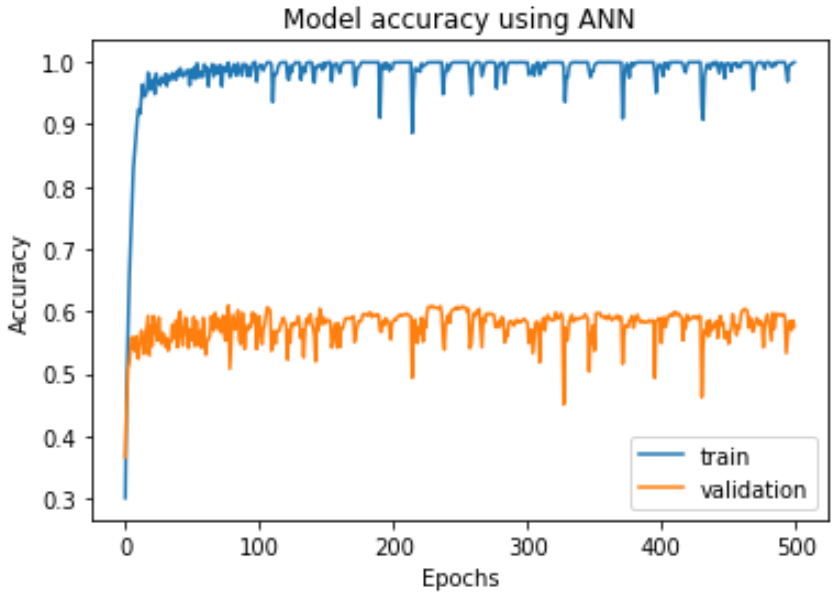
The architecture we use is:

```
model.add(BatchNormalization(input_shape=X_train.shape[1:]))

model.add(Conv2D(64, (7,7), activation = 'relu', strides=(1,1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Conv2D(128, (7,7), activation = 'relu', strides=(1,1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Conv2D(256, (7,7), activation = 'relu', strides=(1,1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Flatten())
```

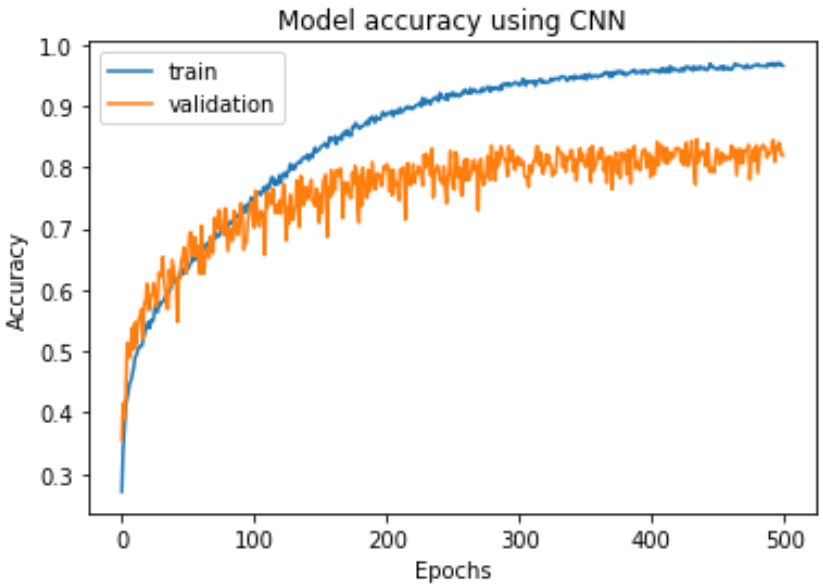
```
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

The CNN is one of the start-of-the art classifiers due to its superior feature extracting methods. The methods are perfect for generalizing to nearly any problem space that contains image classification. It is clear that this classification method works excellently for our problem. The mel spectrograms we are converting from the wav files show unique edges and beat patterns that the higher-level filters of the CNN can learn.

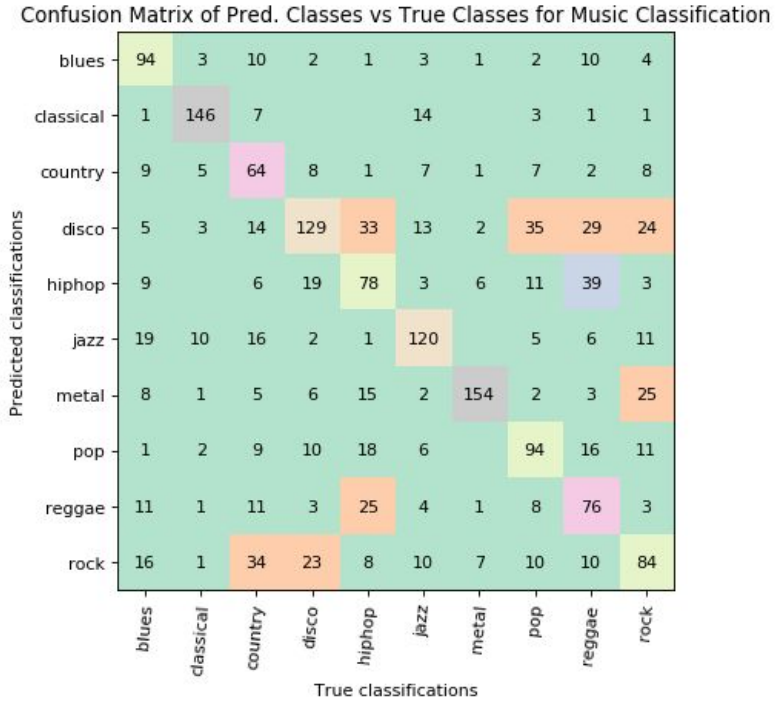
**IV. Classifier Analysis**



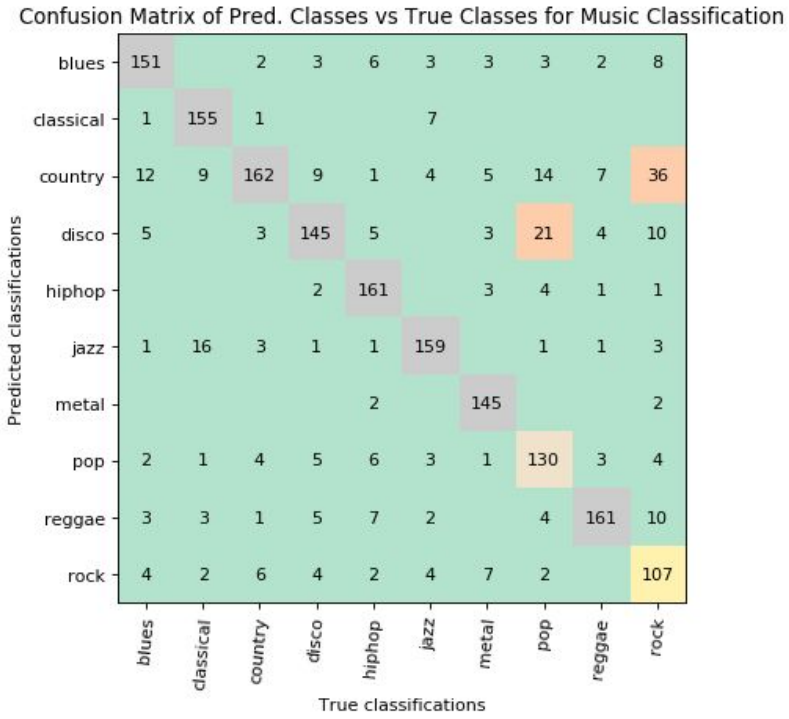

Looking at the above graph for accuracy and epochs ran for the Artificial Neural Network, you can clearly see the network start to overfit around 55% accuracy, very early on around epoch 10. This graph immediately tells us we should be introducing lots of regularization into our network, as well as possible dropout. To combat this further, one may try to incorporate data augmentation, which is one of our possible future implementations discussed in the conclusion.




The above graph is the accuracy vs epoch variables for the CNN model. In this model, it is clear that the network takes longer to begin overfitting, which is around 70% at epoch 100. However, it is also more difficult to judge the magnitude of this overfitting, as the validation slope overall is still increasing, all the way up to an average of 80% at epoch 400. There still may be some regularization terms we could incorporate into the network to avoid overfitting and possibly stay with the training accuracy beyond 70%. Looking at the two model accuracy graphs, one can immediately tell the CNN is outperforming the ANN.



Above is the confusion matrix for the ANN. We can immediately see there isn't an ideal diagonal correlation, with many outliers showing up strongly. There are also just generally mis-classifications throughout the matrix; this is most likely to do with the low accuracy of this network, and shows us some of the initial difficulties simple classifiers will have when differentiating certain classes. From this confusion matrix, we can see the immediate difficulties a classifier will have: predicting rock when it's country, predicting rock when it's disco, predicting disco and reggae when it's hiphop, predicting disco when it's pop, predicting disco and hiphop when it's reggae, and predicting disco and metal when it's rock. This informs us what are some of the mis-classifications we should be looking out for.




Above is the confusion matrix for the CNN model. We can immediately see due to the higher accuracy that there is far less misclassifications throughout the matrix. This informs us that the network is doing an overall better job at classifying the music genres. We can also see that the network is able to differentiate some of the bigger difficulties that the ANN had. Specifically, the network is predicting country, disco, and hip hop much more accurately. However, the network is still clearly struggling with predicting disco when it's actually pop. The most interesting issue with this classifier is it's now error in predicting country when it's actually rock; this was not a problem with the ANN, however, it was saying many other things were rock that were not. When thought about intuitively, this misclassification error in the CNN is much more reasonable and understandable. Even for myself, I may struggle depending on the era of music saying whether a song is disco when it's pop, or it's country when it's rock.

Using the confusion matrices and the plots of accuracy are extremely helpful in gaining more insight into what our networks are doing, and why one is actually outperforming another. The confusion matrices

tell us directly which genres may need more feature extraction to be able to differentiate further, and the accuracy plots help us know if more regularization should be brought in to help our networks avoid overfitting.

**V. Conclusion**

In this report we implemented two classifiers for the music-genre classification problem. The first classifier was the canonical ANN model, which surprised us with extremely quick convergence. However, the point at which it converged would not be as high as our second implemented classifier; the CNN model. The convolutional neural network learned the mel-spectrograms for each genre much better than the ANN due to the convolutional filters it uses, the max-pooling layers, and the dropout and batch normalization methods. The convergence rate takes longer due to the size of the CNN, but it's worth the time to converge due to the accuracy it gives. If the reader were to want to improve on our findings, it is recommended that data augmentation be implemented to double the size of the data. With more data, there is a far greater chance of converging to a higher accuracy on the testing data. Unfortunately, we did not have time to implement this add-on, although we believe it will be a good way to gain accuracy.

It is worth noting that we also implemented K-fold cross-validation, specifically 10-fold cross-validation. Unfortunately, the time to run and train 10 networks would be multiple days, as one network with one validation set is taking us 3-4hrs, going for the most optimal accuracy. Due to this, we don't use this validation method in practice but have it implemented in all_models.ipynb.