|  |
| --- |
| **Midpoint Report: Classifying Music**  **Subgenres Using Neural Networks** |

**Coauthor Coauthor Coauthor Coauthor**

Sam Schrader Daniel Tucek Rohith Kumar Sura Ryan Cafarelli

*saschra@siue.edu dtucek@siue.edu rosura@siue.edu rcafare@siue.edu*

**1 Introduction**

While there has been a fair amount of research into automatic classification of music genres [1, 2, 3, 4], less has been done with subgenre classification. It is a more difficult problem, as subgenres tend to have more common elements and overlap, leading to ambiguity in classification even among human experts. For our project we are tackling the problem of automatic classification of EDM subgenres by extending the results of Caparrini et al. to use neural networks on their dataset. We take raw audio files of EDM songs and process them into spectrograms and MFCCs, which are then fed into CNN and RNN-LSTM models; these models then output a classification label indicating the subgenre. The overall process flow is shown in Figure 1.

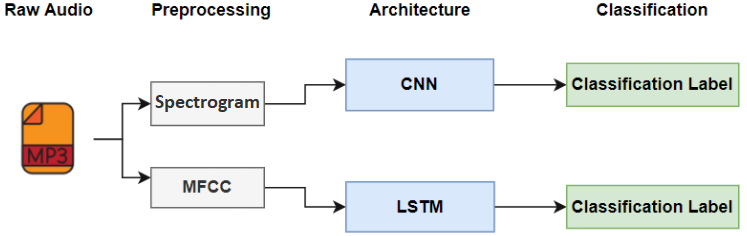


Figure 1: Process flow for EDM subgenre classification project.

**2 Related Work**

A number of different approaches have been taken to classify music by genre and subgenre. The project focused on the work of five of these approaches. In [1], the author gives the details of the genre classification using Support Vector Machine (SVM) and their comparison in terms of accuracy with other classifiers like decision trees, multilayer perceptrons, simple logistic, random tree, and Naive Bayes. The proposed classifier gives an accuracy of 92% when classifying 5 musical genres which are higher compared to other classifiers used in this experiment. In [5], a classifier was designed for jazz subgenre classification using an LSTM layer as the core and measured the performance of their new classifier with respect to the three techniques MLP, SVM, and KNN. They used Mel Frequency Cepstral Coefficients (MFCCs) for feature extraction of each song. The LSTM network achieved an accuracy of 80.30% whereas the MLP, SVM, and KNN achieved a maximum accuracy of 79.39%, 81.67%, and 77.43% respectively. Additionally, they added a MLP network before the LSTM layer that boosted the accuracy to 89.824%. In [2], the authors used a 5-layer Independently RNN with scattering coefficient for preprocessing the data to complete the music genre classification. The experimental results showed an excellent performance on the GTZAN dataset in terms of classification accuracy and training time. LSTM accuracy reaches a high accuracy of 97%, however, its training time per iteration is as long as 0.68s. IndRNN performed best with both comprehensive training time of 0.23s and classification accuracy of 96%. Compared to the state-of-the-art models, experimental results are very competitive. In [3], images of spectrograms were generated from time-slices of songs and used as the input to the neural network to classify the songs into their respective musical genres. Initially, they trained using 27 genres with no change in architecture and a 47% accuracy observed. Later, they trained on 7 genres with a change in activation function from ELU to ReLU where the accuracy improved to 67%. In [4], the authors examined the performance of convolutional neural networks (CNN) and recurrent neural networks (RNN). For pre-processing, they tested a Fast-Fourier Transform, a mel-spectrogram and mel-frequency cepstral coefficients. The combination of the GTZAN dataset, mel-spectrograms, and convolutional neural networks yielded the best result. They observed accuracy of 59% with CNN and 33% with RNN. Although the results fell behind the state-of-the-art performance, they outperformed other music classification studies that use a CNN by a large margin.

Many of the papers implemented CNN algorithms and Machine Learning techniques, including k-NN, SVMs, LSTMs for music genre classification. We decided to implement LSTM and CNN to determine which approach provides the better accuracy.

**3 Materials and Methods**

**3.1 Datasets and Features**

Our primary dataset comes from [6], in which the authors investigate the use of tree-based classifiers in determining the subgenre of an EDM track. They prepared two datasets; we are currently utilizing Set 1, which consists of the top 100 tracks in each of the 23 subgenres on Beatport as of November 29, 2016. Likely due to space and/or intellectual property concerns, the authors did not provide the raw audio for each track, instead making available the extracted audio features for each track in the form of a CSV file. Tracks can be uniquely identified in Beatport’s database using a numeric ID, which is given in the files for Set 1 but not Set 2. We wanted to be able to extract new features from the tracks if necessary, so a VBA macro was written for Excel which parses the HTML document for a track on Beatport, locates the URL for the audio sample stream, and downloads it. These sample streams are typically 2 minutes of 96 kbps audio in MP3 format, although they can occasionally be of shorter duration. We were able to retrieve audio samples for 2,258 out of the 2,300 tracks in Set 1; we have more than 97 out of 100 tracks for most subgenres, with Hip-Hop being the least represented at 93. Contact was made with Antonio Caparrini to try to acquire the track IDs for Set 2, which would afford us another 2,900 tracks from Beatport’s 2018 charts; we are still in the process of getting this information. Our VBA macro will make it straightforward to pull more audio from Beatport if we deem it necessary.

As detailed in subsequent sections, we chose to investigate two types of neural net: LSTMs and CNNs. LSTMs are often used for applications involving sequential data; however, in general it is not trivial to train them on overly long sequences (typically less than a few hundred time steps). The raw audio data we’ve collected has a sample rate of 44,100 Hz (samples per second), so a few hundred samples is on the order of milliseconds and likely wouldn’t be useful for genre classification. Thus, for the LSTM so far, we have preprocessed the audio data into MFCCs (mel-frequency cepstral coefficients). MFCCs have been used successfully in other genre classification systems such as [6]. They are derived by mapping the Fourier transform (frequency spectrum) of a signal to the mel scale (which expresses frequencies logarithmically, more similar to our own perception of pitch), followed by performing a discrete cosine transform on the logarithm of the result. Much like generating a spectrogram, we can find the MFCCs for successive windowed segments of an audio signal to generate short sequences which still encapsulate a lot of audio information. There are two major parameters that can be adjusted when generating MFCCs which affect the resolution of the data: the number of audio segments (time resolution) and the number of coefficients (frequency resolution).

**3.2 Methods**

**3.2.1 CNN**

Convolutional Neural Networks (CNNs) are a type of neural network that are typically used for image recognition. Each layer of a CNN extracts features from an image by convolving (performing a mathematical operation of the data) a subset of the image onto the next output layer. After each convolutional layer many models will format and shrink the data using normalization, activation and pooling layers. These layers together help the model to learn higher level details about sections of the input data rather than focusing on individual pixels. A typical CNN incorporates multiple sets of theses layers before flattening the data into a fully connected layer to classify the overall image.

For music classification the audio data must be translated into an image to be fed into a CNN. The typical format for this image is called a spectrogram. A spectrogram is a visual representation of the frequencies of an audio signal over time.

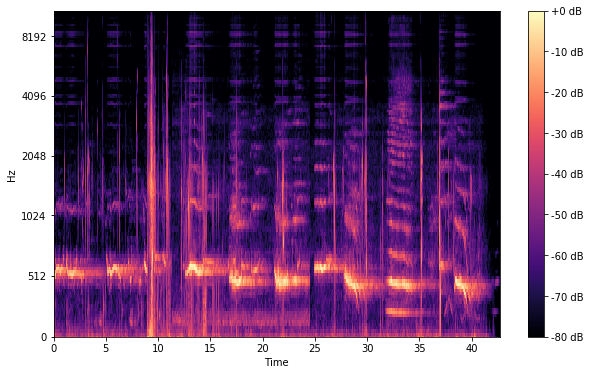


Figure 2: Spectrogram of a raw audio file.

Figure 2 shows the spectrogram as a heat map calculated from an audio signal over time translated by a Fourier transform into the frequency domain.

The choice of hyperparameters for this model (learning rate, number of epochs, batch size and steps per epoch) were influenced by the limitations of Google Colab. The number of steps was defined as a function of the batch size (input size/batch size). The learning rate was influenced by the batch size and number of epochs.

**3.2.2 RNN-LSTM**

The other method being attempted is the recurrent neural network (RNN) with long short-term memory (LSTM). LSTMs are helpful with classifying time series data. They are able to detect patterns with longer dependencies in comparison to regular RNNs. This is because an LSTM stores information from multiple previous iterations while an RNN only stores the information from the previous iteration. A LSTM can perform computations using an activation and recurrent activation function that allows it to forget and add information to its memory. Our LSTM used a tanh activation function and a sigmoid recurrent activation function. The input for the network used MFCCs, discussed earlier, which were extracted from the data. Each song was processed into an audio time series for a certain duration. Each time series was divided up into the same number of segments to calculate the MFCCs for every segment.

The RNN-LSTM model has several hyperparameters: learning rate was 0.001, batch size was 32, and 30 epochs were used. Attempting to lower the batch size led to the same accuracy but it took a longer amount of time since it’s not exploiting vectorization as much. While increasing it gave lower accuracy. So, keeping the batch size close to 32 was optimal. Then, for the number of epochs, Figure 3 shows the loss/accuracy for each epoch for the training and validation set. Figure 3 shows that the model didn’t have a trend to increase in loss or decrease in accuracy meaning the number of epochs didn’t need to be changed. It could be decreased because the validation set usually started flat-lining, but when changes are made to the model having slightly more epochs might be beneficial.

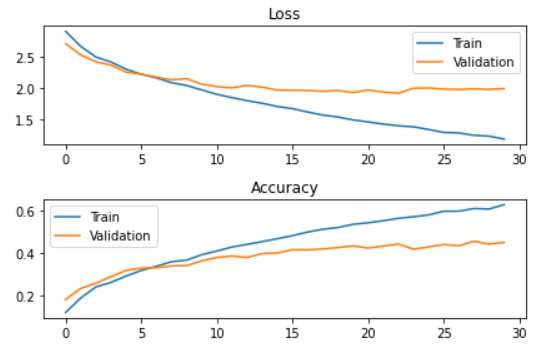


Figure 3: Training vs. validation loss (top) and accuracy (bottom) for the LSTM model.

The data was split by using 80% for the training set and 20% for the testing set. If a validation set was used, then 20% of the training set was used for the validation instead. This was chosen mostly because of convention. The other example models found had similar splits.

The main metric used was accuracy. Additionally, precision and recall were calculated. Since this is a classification problem, the AUC and confusion matrix were also found. If a validation set was used, then the loss and accuracy were plotted, for each epoch, to compare the train and validation values.

**4 Preliminary Results and Next Steps**

**4.1 CNN**

For our project we used a CNN produced for genre classification of music that was used to try and classify sub-genres of electronic dance music. The CNN was broken into groupings of layers. Each group consisted of a convolutional layer, batch normalization, relu activation layer and a max pooling layer. The groupings corresponding to the filter size of the convolutional layers (8, 16, 32, 64, 128 and 256). After the groupings the data is flattened and a dropout of 30% is applied before using a softmax function to classify the image into the 23 sub-genre categories.

Each track was broken into ten five second intervals then each interval was translated into a spectrogram with image size 360 x 360. For each sub-genre, the spectrograms were shuffled then separated into training, validation, and testing sets (70%, 10% and 20%). The model was trained for 20 epochs with a learning rate of 0.001 and batch size of 100. The hyperparameters were chosen due to memory limitations when running the model, this required a smaller batch size which led to a smaller learning rate being applied to each batch. A secondary limitation was discovered when using Google Colab which was the cell execution time of 12 hours. Due to the size and number of the images in the training set the model exceeded this time constraint.

The initial results of this experiment produced a max of 40% validation accuracy and a final validation accuracy of 29%. The testing accuracy was 28% with a loss of 3.99. The training vs. validation accuracy and loss are shown in Figure 4.

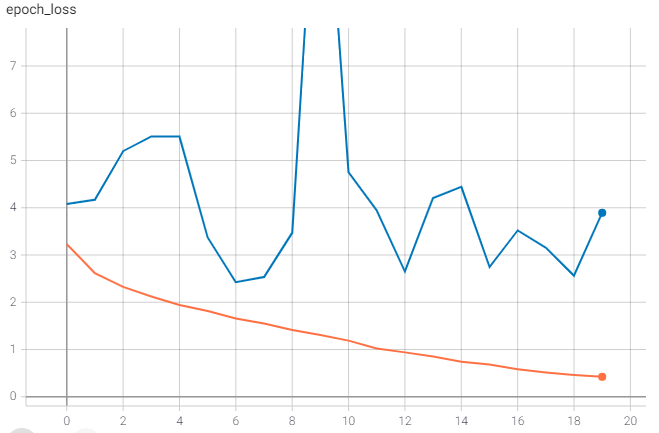
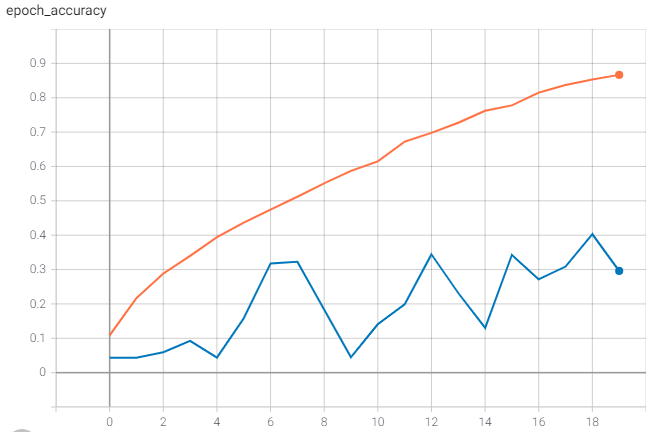


Figure 4: Training vs. validation accuracy (left) and loss (right) for the CNN.

The testing precision was 27% with a recall of 44% and a ROC AUC of 50%. The confusion matrix heatmap, shown in Figure 5, shows that the model has a strong emphasis toward index 6 corresponding to the subgenre electro house.

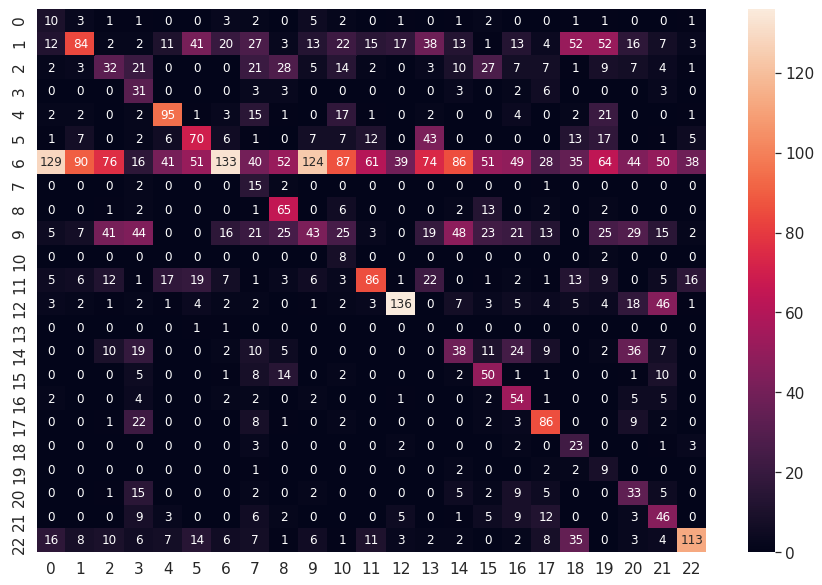


Figure 5: Heatmap of the confusion matrix for the CNN.

The next steps for this approach are to experiment with the model structure, hyperparameters, image size, and corresponding track segment duration. A failed experiment was conducted using the full track length which led to overfitting of the model, this was likely due to a large number of model parameters compared to the size of the dataset. Another failed experiment was conducted using image sizes of 720 x 720. Due to the image size the memory usage exceeded the limitations of Google Colab as well as the cell execution time limitation.

**4.2 RNN-LSTM**

The starting baseline model used for the LSTM network had five layers: two LSTM, dense, dropout, and dense respectively. The two LSTM layers used a tanh activation function, sigmoid recurrent activation function, and has 64 units for the output. The first dense layer used the relu activation function with 64 units for output. The dropout layer is used during training to attempt to stop overfitting, and it was set to drop 30% of the input. The dropout layer used 64 units for output. The output dense layer has an output of 23 which corresponds with the 23 different subgenres in the dataset, and it uses a softmax activation function.

Initially, only a training and testing set were used to first see what kind of results it could achieve. The training and testing set were split into 80% and 20% of the original data respectively. The first test of the model produced a test accuracy of 47.4% using 30 second track durations from each MP3 file. Each 30 second duration was split into 10 equal length segments to calculate the MFCCs. Next, validation was used by splitting the data the same way but 20% of the training set was used for validation instead. This model produced a test accuracy of 44.3%. Figure 6 shows a heatmap of the confusion matrix based on the model getting the 44.3% accuracy. The confusion matrix shows that there are a few subgenres that get predicted very poorly compared to the rest. This might mean there isn’t enough distinguishing characteristics for these subgenres to be classified correctly and the data needs to be preprocessed differently. Several other experiments were tried to see the different accuracies the model could produce. Table 1 shows the different changes that were made to the model and the results. Some of the changes were made to help find more optimal hyperparameters, test different network topologies, and see what would happen with different optimizers/loss functions. None of the changes positively impacted the accuracy in any significant manner.

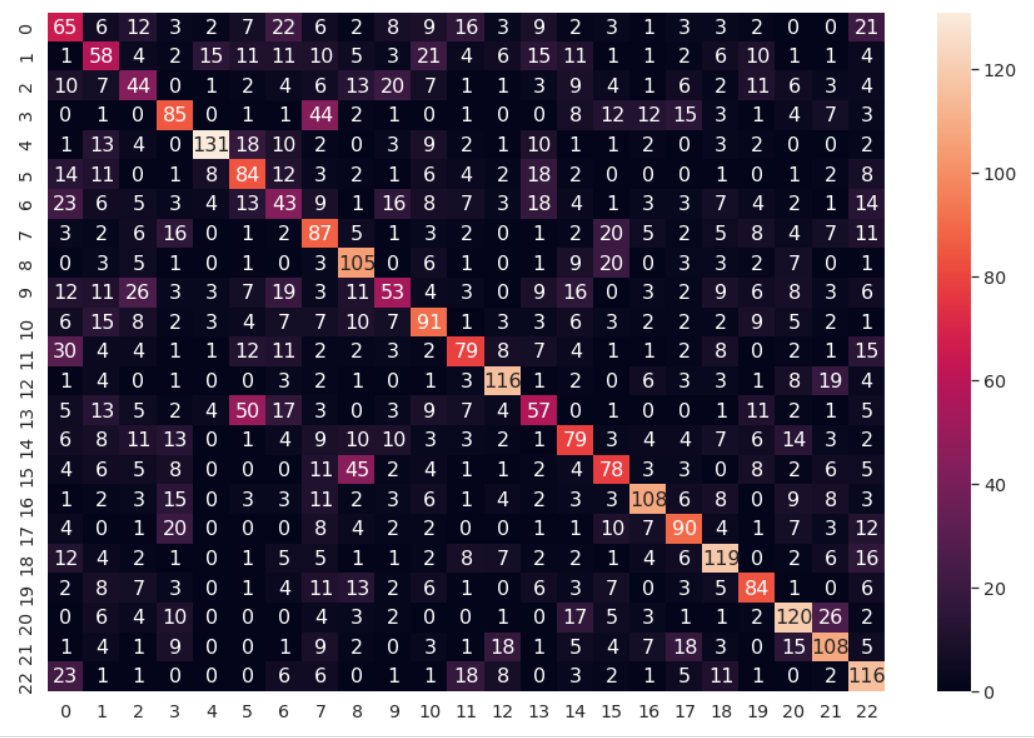
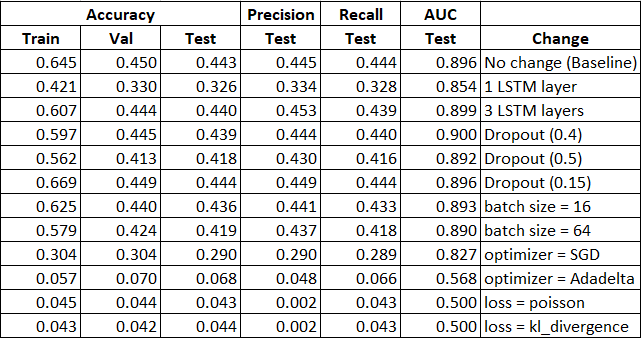


Figure 6: Heatmap of the confusion matrix for the LSTM model.

Table 1: Results attempting different changes to the LSTM baseline model.



The next steps for this approach would be to optimize the hyperparameters more. Also, another thing that could be done is increasing the initial track durations to be higher than 30 seconds. Currently, all the MP3 files are between 90 and 120 seconds. The model requires for the track duration to be the same, so the MFCCs can be calculated in equal segments. This means the track duration can be increased to 90 seconds. The downside of doing this is that it increases the amount of memory and time needed by a large margin to train the model. Lastly, different network topologies should be tested.

**References**

[1] G. Mali and S. P. Mahajan, "Audio Genre Classification Employing Support Vector Machine," 2018 4th International Conference for Convergence in Technology (I2CT), Mangalore, India, 2018, pp. 1-5, doi: 10.1109/I2CT42659.2018.9058184.

[2] W. Wu, F. Han, G. Song and Z. Wang, "Music Genre Classification Using Independent Recurrent Neural Network," 2018 Chinese Automation Congress (CAC), Xi'an, China, 2018, pp. 192-195, doi: 10.1109/CAC.2018.8623623.

[3] N. Pelchat and C. M. Gelowitz, "Neural Network Music Genre Classification," 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), Edmonton, AB, Canada, 2019, pp. 1-4, doi: 10.1109/CCECE.2019.8861555.

[4] Yang, J. “Music Genre Classification with Neural Networks: An Examination of Several Impactful Variables.” (2018).

[5] R. J. M. Quinto, R. O. Atienza and N. M. C. Tiglao, "Jazz music sub-genre classification using deep learning," TENCON 2017 - 2017 IEEE Region 10 Conference, Penang, Malaysia, 2017, pp. 3111-3116, doi: 10.1109/TENCON.2017.8228396.

[6] Antonio Caparrini, Javier Arroyo, Laura Pérez-Molina & Jaime SánchezHernández (2020): Automatic sub genre classification in an electronic dance music taxonomy, Journal of New Music Research, DOI: 10.1080/09298215.2020.1761399