Music Genre Classification using Machine Learning Techniques

Hareesh Bahuleyan
University of Waterloo

Music Genre Classification using Machine Learning Techniques

Hareesh Bahuleyan
University of Waterloo, ON, Canada
hpallika@uwaterloo.ca

Abstract

Categorizing music files according to their genre is a challenging task in the area of music information retrieval (MIR). In this study, we compare the performance of two classes of models. The first is a deep learning approach wherein a CNN model is trained end-to-end, to predict the genre label of an audio signal, solely using its spectrogram. The second approach utilizes hand-crafted features, both from the time domain and frequency domain. We train four traditional machine learning classifiers with these features and compare their performance. The features that contribute the most towards this classification task are identified. The experiments are conducted on the *Audio set* data set and we report an AUC value of 0.894 for an ensemble classifier which combines the two proposed approaches.[1]

1 Introduction

With the growth of online music databases and easy access to music content, people find it increasing hard to manage the songs that they listen to. One way to categorize and organize songs is based on the genre, which is identified by some characteristics of the music such as rhythmic structure, harmonic content and instrumentation (Tzanetakis and Cook, 2002). Being able to automatically classify and provide tags to the music present in a user's library, based on genre, would be beneficial for audio streaming services such as Spotify and iTunes. This study explores the application of machine learning (ML) algorithms to identify and classify the genre of a given audio file. The first model described in this paper uses convolutional neural networks (Krizhevsky et al., 2012), which is trained end-to-end on the MEL spectrogram of the audio signal. In the second part of the study, we extract features both in the time domain and the frequency domain of the audio signal. These features are then fed to conventional machine learning models namely Logistic Regression, Random Forests (Breiman, 2001), Gradient Boosting (Friedman, 2001) and Support Vector Machines which are trained to classify the given audio file. The models are evaluated on the *Audio Set* dataset (Gemmeke et al., 2017). We compare the proposed models and also study the relative importance of different features.

The rest of this paper is organized as follows. Section 2 describes the existing methods in the literature for the task of music genre classification. Section 3 is an overview of the the dataset used in this study and how it was obtained. The proposed models and the implementation details are discussed in Section 4. The results are reported in Section 5.2, followed by the conclusions from this study in Section 6.

2 Literature Review

Music genre classification has been a widely studied area of research since the early days of the Internet. Tzanetakis and Cook (2002) addressed this problem with supervised machine learning approaches such as Gaussian Mixture model and k-nearest neighbour classifiers. They introduced 3 sets of features for this task categorized as timbral structure, rhythmic content and pitch content. Hidden Markov Models (HMMs), which have been extensively used for speech recognition tasks, have also been explored for music genre classification (Scaringella and Zoia, 2005; Soltau et al., 1998). Support vector machines (SVMs)

[1] The code has been opensourced and is available at https://github.com/HareeshBahuleyan/music-genre-classification

with different distance metrics are studied and compared in Mandel and Ellis (2005) for classifying genre.

In Lidy and Rauber (2005), the authors discuss the contribution of psycho-acoustic features for recognizing music genre, especially the importance of STFT taken on the Bark Scale (Zwicker and Fastl, 1999). Mel-frequency cepstral coefficients (MFCCs), spectral contrast and spectral roll-off were some of the features used by (Tzanetakis and Cook, 2002). A combination of visual and acoustic features are used to train SVM and AdaBoost classifiers in Nanni et al. (2016).

With the recent success of deep neural networks, a number of studies apply these techniques to speech and other forms of audio data (Abdel-Hamid et al., 2014; Gemmeke et al., 2017). Representing audio in the time domain for input to neural networks is not very straight-forward because of the high sampling rate of audio signals. However, it has been addressed in Van Den Oord et al. (2016) for audio generation tasks. A common alternative representation is the spectrogram of a signal which captures both time and frequency information. Spectrograms can be considered as images and used to train convolutional neural networks (CNNs) (Wyse, 2017). A CNN was developed to predict the music genre using the raw MFCC matrix as input in Li et al. (2010). In Lidy and Schindler (2016), a constant Q-transform (CQT) spectrogram was provided as input to the CNN to achieve the same task.

This work aims to provide a comparative study between 1) the deep learning based models which only require the spectrogram as input and, 2) the traditional machine learning classifiers that need to be trained with hand-crafted features. We also investigate the relative importance of different features.

3 Dataset

In this work, we make use of *Audio Set*, which is a large-scale human annotated database of sounds (Gemmeke et al., 2017). The dataset was created by extracting **10-second sound clips** from a total of 2.1 million YouTube videos. The audio files have been annotated on the basis of an ontology which covers 527 classes of sounds including musical instruments, speech, vehicle sounds, animal sounds and so on[2]. This study requires only the audio files that belong to the music category, specifically having one of the seven genre tags shown in Table 1.

Table 1: Number of instances in each genre class

	Genre	**Count**
1	Pop Music	8100
2	Rock Music	7990
3	Hip Hop Music	6958
4	Techno	6885
5	Rhythm Blues	4247
6	Vocal	3363
7	Reggae Music	2997
	Total	**40540**

The number of audio clips in each category has also been tabulated. The raw audio clips of these sounds have not been provided in the *Audio Set* data release. However, the data provides the `YouTubeID` of the corresponding videos, along with the start and end times. Hence, the first task is to retrieve these audio files. For the purpose of audio retrieval from YouTube, the following steps were carried out:

1. A command line program called `youtube-dl` (Gonzalez, 2006) was utilized to download the video in the `mp4` format.

2. The `mp4` files are converted into the desired `wav` format using an audio converter named `ffmpeg` (Tomar, 2006) (command line tool).

Each `wav` file is about 880 KB in size, which means that the total data used in this study is approximately 34 GB.

4 Methodology

This section provides the details of the data preprocessing steps followed by the description of the two proposed approaches to this classification problem.

[2] https://research.google.com/audioset/ontology/index.html

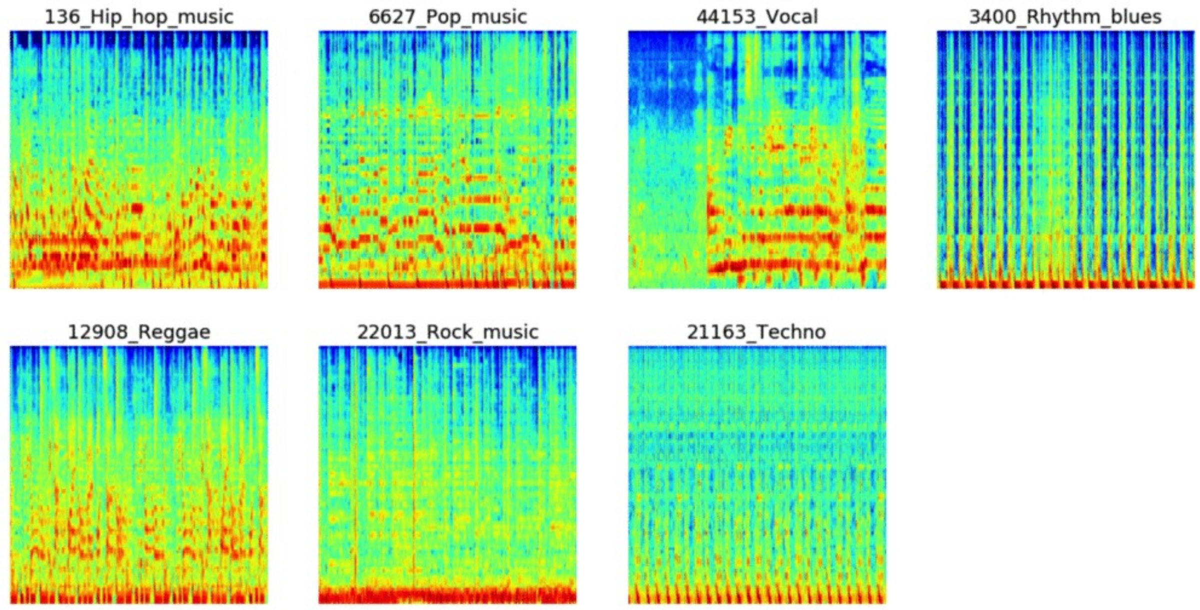


Figure 1: Sample spectrograms for 1 audio signal from each music genre

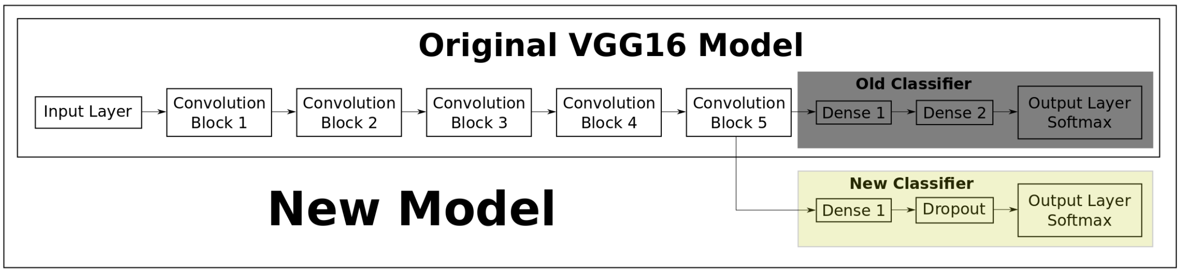


Figure 2: Convolutional neural network architecture (Image Source: Hvass Tensorflow Tutorials)

4.1 Data Pre-processing

In order to improve the Signal-to-Noise Ratio (SNR) of the signal, a pre-emphasis filter, given by Equation 1 is applied to the original audio signal.

$$y(t) = x(t) - \alpha * x(t-1) \tag{1}$$

where, $x(t)$ refers to the original signal, and $y(t)$ refers to the filtered signal and α is set to 0.97. Such a pre-emphasis filter is useful to boost amplitudes at high frequencies (Kim and Stern, 2012).

4.2 Deep Neural Networks

Using deep learning, we can achieve the task of music genre classification without the need for hand-crafted features. Convolutional neural networks (CNNs) have been widely used for the task of image classification (Krizhevsky et al., 2012). The 3-channel (RGB) matrix representation of an image is fed into a CNN which is trained to predict the image class. In this study, the sound wave can be represented as a spectrogram, which in turn can be treated as an image (Nanni et al., 2016)(Lidy and Schindler, 2016). The task of the CNN is to use the spectrogram to predict the genre label (one of seven classes).

4.2.1 Spectrogram Generation

A spectrogram is a 2D representation of a signal, having time on the x-axis and frequency on the y-axis. A colormap is used to quantify the magnitude of a given frequency within a given time window. In this study, each audio signal was converted into a MEL spectrogram (having MEL frequency bins on the y-axis). The parameters used to generate the power spectrogram using STFT are listed below:

- Sampling rate (`sr`) = 22050
- Frame/Window size (`n_fft`) = 2048

- Time advance between frames (hop_size) = 512 (resulting in 75% overlap)

- Window Function: Hann Window

- Frequency Scale: MEL

- Number of MEL bins: 96

- Highest Frequency (f_max) = sr/2

4.2.2 Convolutional Neural Networks

From the Figure 1, one can understand that there exists some characteristic patterns in the spectrograms of the audio signals belonging to different classes. Hence, spectrograms can be considered as 'images' and provided as input to a CNN, which has shown good performance on image classification tasks. Each block in a CNN consists of the following operations[3]:

- **Convolution**: This step involves sliding a matrix filter (say 3x3 size) over the input image which is of dimension image_width x image_height. The filter is first placed on the image matrix and then we compute an element-wise multiplication between the filter and the overlapping portion of the image, followed by a summation to give a feature value. We use many such filters , the values of which are 'learned' during the training of the neural network via backpropagation.

- **Pooling**: This is a way to reduce the dimension of the feature map obtained from the convolution step, formally know as the process of *down sampling*. For example, by max pooling with 2x2 window size, we only retain the element with the maximum value among the 4 elements of the feature map that are covered in this window. We keep moving this window across the feature map with a predefined stride.

- **Non-linear Activation**: The convolution operation is linear and in order to make the neural network more powerful, we need to introduce some non-linearity. For this purpose, we can apply an activation function such as ReLU[4] on each element of the feature map.

In this study, a CNN architecture known as VGG-16, which was the top performing model in the ImageNet Challenge 2014 (classification + localization task) was used (Simonyan and Zisserman, 2014). The model consists of 5 convolutional blocks (conv base), followed by a set of densely connected layers, which outputs the probability that a given image belongs to each of the possible classes.

For the task of music genre classification using spectrograms, we download the model architecture with pre-trained weights, and extract the conv base. The output of the conv base is then send to a new feed-forward neural network which in turn predicts the genre of the music, as depicted in Figure 2.

There are two possible settings while implementing the pre-trained model:

1. **Transfer learning**: The weights in the conv base are kept fixed but the weights in the feed-forward network (represented by the yellow box in Figure 2) are allowed to be tuned to predict the correct genre label.

2. **Fine tuning**: In this setting, we start with the pre-trained weights of VGG-16, but allow all the model weights to be tuned during training process.

The final layer of the neural network outputs the class probabilities (using the softmax activation function) for each of the seven possible class labels. Next, the cross-entropy loss is computed as follows:

$$\mathcal{L} = -\sum_{c=1}^{M} y_{o,c} * \log p_{o,c} \tag{2}$$

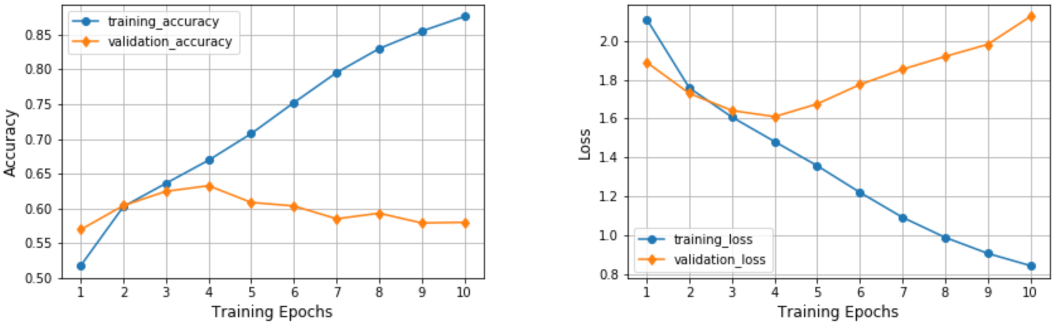
where, M is the number of classes; $y_{o,c}$ is a binary indicator whose value is 1 if observation o belongs to class c and 0 otherwise; $p_{o,c}$ is the model's predicted probability that observation o belongs to class c. This loss is used to backpropagate the error, compute the gradients and thereby update the weights of the network. This iterative process continues until the loss converges to a minimum value.

4.2.3 Implementation Details

The spectrogram images have a dimension of 216 x 216. For the feed-forward network connected to the conv base, a 512-unit hidden layer is implemented. Over-fitting is a common issue in neural

[3] https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[4] https://en.wikipedia.org/wiki/Rectifier_(neural_networks)



(a) Accuracy

(b) Loss

Figure 3: Learning Curves - used for model selection; Epoch 4 has the minimum validation loss and highest validation accuracy

networks. In order to prevent this, two strategies are adopted:

1. **L2-Regularization** (Ng, 2004): The loss function of the neural network is added with the term $\frac{1}{2}\lambda \sum_i {w_i}^2$, where w refers to the weights in the neural networks. This method is used to penalize excessively high weights. We would like the weights to be diffused across all model parameters, and not just among a few parameters. Also, intuitively, smaller weights would correspond to a less complex model, thereby avoiding overfitting. λ is set to a value of 0.001 in this study.

2. **Dropout** (Srivastava et al., 2014): This is a regularization mechanism in which we *shut-off* some of the neurons (set their weights to zero) randomly during training. In each iteration, we thereby use a different combination of neurons to predict the final output. This makes the model generalize without any heavy dependence on a subset of the neurons. A dropout rate of 0.3 is used, which means that a given weight is set to zero during an iteration, with a probability of 0.3.

The dataset is randomly split into train (90%), validation (5%) and test (5%) sets. The same split is used for all experiments to ensure a fair comparison of the proposed models.

The neural networks are implemented in Python using Tensorflow [5]; an NVIDIA Titan X GPU was utilized for faster processing. All models were trained for 10 epochs with a batch size of 32 with the ADAM optimizer (Kingma and Ba, 2014). One epoch refers to one iteration over the entire training dataset.

Figure 3 shows the learning curves - the loss (which is being optimized) keeps decreasing as the training progresses. Although the training accuracy keeps increasing, the validation accuracy first increases and after a certain number of epochs, it starts to decrease. This shows the model's tendency to overfit on the training data. The model that is selected for evaluation purposes is the one that has the highest accuracy and lowest loss on the validation set (epoch 4 in Figure 3).

4.2.4 Baseline Feed-forward Neural Network

To assess the performance improvement that can be achived by the CNNs, we also train a baseline feed-forward neural network that takes as input the same spectrogram image. The image which is a 2-dimensional vector of pixel values is unwrapped or flattened into a 1-dimensional vector. Using this vector, a simple 2-layer neural network is trained to predict the genre of the audio signal. The first hidden layer consists of 512 units and the second layer has 32 units, followed by the output layer. The activation function used is ReLU and the same regularization techniques described in Section 4.2.3 are adopted.

4.3 Manually Extracted Features

In this section, we describe the second category of proposed models, namely the ones that require hand-crafted features to be fed into a machine learning classifier. Features can be broadly classified as time domain and frequency domain features. The feature extraction was done using

[5] http://tensorflow.org/

librosa[6], a Python library.

4.3.1 Time Domain Features

These are features which were extracted from the raw audio signal.

1. **Central moments**: This consists of the mean, standard deviation, skewness and kurtosis of the amplitude of the signal.

2. **Zero Crossing Rate (ZCR)**: A zero crosssing point refers to one where the signal changes sign from positive to negative (Gouyon et al., 2000). The entire 10 second signal is divided into smaller frames, and the number of zero-crossings present in each frame are determined. The frame length is chosen to be 2048 points with a hop size of 512 points. Note that these frame parameters have been used consistently across all features discussed in this section. Finally, the average and standard deviation of the ZCR across all frames are chosen as representative features.

3. **Root Mean Square Energy (RMSE)**: The energy in a signal is calculated as:

$$\sum_{n=1}^{N} |x(n)|^2 \tag{3}$$

   Further, the root mean square value can be computed as:

$$\sqrt{\frac{1}{N}\sum_{n=1}^{N} |x(n)|^2} \tag{4}$$

   RMSE is calculated frame by frame and then we take the average and standard deviation across all frames.

4. **Tempo**: In general terms, tempo refers to the how fast or slow a piece of music is; it is expressed in terms of Beats Per Minute (BPM). Intuitively, different kinds of music would have different tempos. Since the tempo of the audio piece can vary with time, we aggregate it by computing the mean across several frames. The functionality in librosa first computes a tempogram following (Grosche et al., 2010) and then estimates a single value for tempo.

4.3.2 Frequency Domain Features

The audio signal can be transformed into the frequency domain by using the Fourier Transform. We then extract the following features.

1. **Mel-Frequency Cepstral Coefficients (MFCC)**: Introduced in the early 1990s by Davis and Mermelstein, MFCCs have been very useful features for tasks such as speech recognition (Davis and Mermelstein, 1990). First, the Short-Time Fourier-Transform (STFT) of the signal is taken with n_fft=2048 and hop_size=512 and a Hann window. Next, we compute the power spectrum and then apply the triangular MEL filter bank, which mimics the human perception of sound. This is followed by taking the discrete cosine transform of the logarithm of all filterbank energies, thereby obtaining the MFCCs. The parameter n_mels, which corresponds to the number of filter banks, was set to 20 in this study.

2. **Chroma Features**: This is a vector which corresponds to the total energy of the signal in each of the 12 pitch classes. (C, C#, D, D#, E ,F, F#, G, G#, A, A#, B) (Ellis, 2007). The chroma vectors are then aggregated across the frames to obtain a representative mean and standard deviation.

3. **Spectral Centroid**: For each frame, this corresponds to the frequency around which most of the energy is centered (Tjoa, 2017). It is a magnitude weighted frequency calculated as:

$$f_c = \frac{\sum_k S(k)f(k)}{\sum_k fk}, \tag{5}$$

   where S(k) is the spectral magnitude of frequency bin k and f(k) is the frequency corresponding to bin k.

4. **Spectral Band-width**: The p-th order spectral band-width corresponds to the p-th order moment about the spectral centroid (Tjoa, 2017) and is calculated as

$$[\sum_k (S(k)f(k) - f_c)^p]^{\frac{1}{p}} \tag{6}$$

   For example, $p = 2$ is analogous to a weighted standard deviation.

[6] https://librosa.github.io/

5. **Spectral Contrast**: Each frame is divided into a pre-specified number of frequency bands. And, within each frequency band, the spectral contrast is calculated as the difference between the maximum and minimum magnitudes (Jiang et al., 2002).

6. **Spectral Roll-off**: This feature corresponds to the value of frequency below which 85% (this threshold can be defined by the user) of the total energy in the spectrum lies (Tjoa, 2017).

For each of the spectral features described above, the mean and standard deviation of the values taken across frames is considered as the representative final feature that is fed to the model.

The features described in this section would be would be used to train machine learning algorithms (refer Section 4.4). The features that contribute the most in achieving a good classification performance will be identified and reported.

4.4 Classifiers

This section provides a brief overview of the four machine learning classifiers adopted in this study.

1. **Logistic Regression (LR)**: This linear classifier is generally used for binary classification tasks. For this multi-class classification task, the LR is implemented as a one-vs-rest method. That is, 7 separate binary classifiers are trained. During test time, the class with the highest probability from among the 7 classifiers is chosen as the predicted class.

2. **Random Forest (RF)**: Random Forest is a ensemble learner that combines the prediction from a pre-specified number of decision trees. It works on the integration of two main principles: 1) each decision tree is trained with only a subset of the training samples which is known as bootstrap aggregation (or bagging) (Breiman, 1996), 2) each decision tree is required to make its prediction using only a random subset of the features (Amit and Geman, 1997). The final predicted class of the RF is determined based on the majority vote from the individual classifiers.

3. **Gradient Boosting (XGB)**: Boosting is another ensemble classifier that is obtained by combining a number of weak learners (such as decision trees). However, unlike RFs, boosting algorithms are trained in a sequential manner using forward stagewise additive modelling (Hastie et al., 2001).

   During the early iterations, the decision trees learnt are fairly simple. As training progresses, the classifier become more powerful because it is made to focus on the instances where the previous learners made errors. At the end of training, the final prediction is a weighted linear combination of the output from the individual learners. XGB refers to eXtreme Gradient Boosting, which is an implementation of boosting that supports training the model in a fast and parallelized manner.

4. **Support Vector Machines (SVM)**: SVMs transform the original input data into a high dimensional space using a kernel trick (Cortes and Vapnik, 1995). The transformed data can be linearly separated using a hyperplane. The optimal hyperplane maximizes the margin. In this study, a radial basis function (RBF) kernel is used to train the SVM because such a kernel would be required to address this non-linear problem. Similar to the logistic regression setting discussed above, the SVM is also implemented as a one-vs-rest classification task.

5 Evaluation

5.1 Metrics

In order to evaluate the performance of the models described in Section 4, the following metrics will be used.

- **Accuracy**: Refers to the percentage of correctly classified test samples.

- **F-score**: Based on the confusion matrix, it is possible to calculate the precision and recall. F-score[7] is then computed as the harmonic mean between precision and recall.

- **AUC**: This evaluation criteria known as the area under the receiver operator characteristics (ROC) curve is a common way to judge the performance of a multi-class classification system. The ROC is a graph between the

[7] https://en.wikipedia.org/wiki/F1_score

Table 2: Comparison of performance of the models on the test set

	Accuracy	**F-score**	**AUC**
Spectrogram-based models			
VGG-16 CNN Transfer Learning	0.63	0.61	**0.891**
VGG-16 CNN Fine Tuning	**0.64**	**0.61**	0.889
Feed-forward NN baseline	0.43	0.33	0.759
Feature Engineering based models			
Logistic Regression (LR)	0.53	0.47	0.822
Random Forest (RF)	0.54	0.48	0.840
Support Vector Machines (SVM)	0.57	0.52	0.856
Extreme Gradient Boosting (XGB)	**0.59**	**0.55**	**0.865**
Ensemble Classifiers			
VGG-16 CNN + XGB	**0.65**	**0.62**	**0.894**

true positive rate and the false positive rate. A baseline model which randomly predicts each class label with equal probability would have an AUC of 0.5, and hence the system being designed is expected to have a AUC higher than 0.5.

5.2 Results and Discussion

In this section, the different modelling approaches discussed in Section 4 are evaluated based on the metrics described in Section 5.1. The values have been reported in Table 2.

The best performance in terms of all metrics is observed for the convolutional neural network model based on VGG-16 that uses only the spectrogram to predict the music genre. It was expected that the fine tuning setting, which additionally allows the convolutional base to be trainable, would enhance the CNN model when compared to the transfer learning setting. However, as shown in Table 2, the experimental results show that there is no significant difference between transfer learning and fine-tuning. The baseline feed-forward neural network that uses the unrolled pixel values from the spectrogram performs poorly on the test set. This shows that CNNs can significantly improve the scores on such an image classification task.

Among the models that use manually crafted features, the one with the least performance is the Logistic regression model. This is expected since logistic regression is a linear classifier. SVMs outperform random forests in terms of accuracy. However, the XGB version of the gradient boosting algorithm performs the best among the feature engineered methods.

5.2.1 Most Important Features

In this section, we investigate which features contribute the most during prediction, in this classification task. To carry out this experiment, we chose the XGB model, based on the results discussed in the previous section. To do this, we rank the top 20 most useful features based on a scoring metric (Figure 4). The metric is calculated as the number of times a given feature is used as a decision node among the individual decision trees that form the gradient boosting predictor.

As can be observed from Figure 4, Mel-Frequency Cepstral Coefficients (MFCC) appear the most among the important features. Previous studies have reported MFCCs to improve the performance of speech recognition systems (Ittichaichareon et al., 2012). Our experiments show that MFCCs contribute significantly to this task of music genre classification. The mean and standard deviation of the spectral contrasts at different frequency bands are also important features. The music tempo, calculated in terms of beats per minute also appear in the top 20 useful features.

Next, we study how much of performance in terms of AUC and accuracy, can be obtained by just using the top N while training the model. From Table 3 it can be seen that with only the top 10 features, the model performance is surprisingly good. In comparison to the full model which has 97 features, the model with the top 30 features has only a marginally lower performance (2 points on

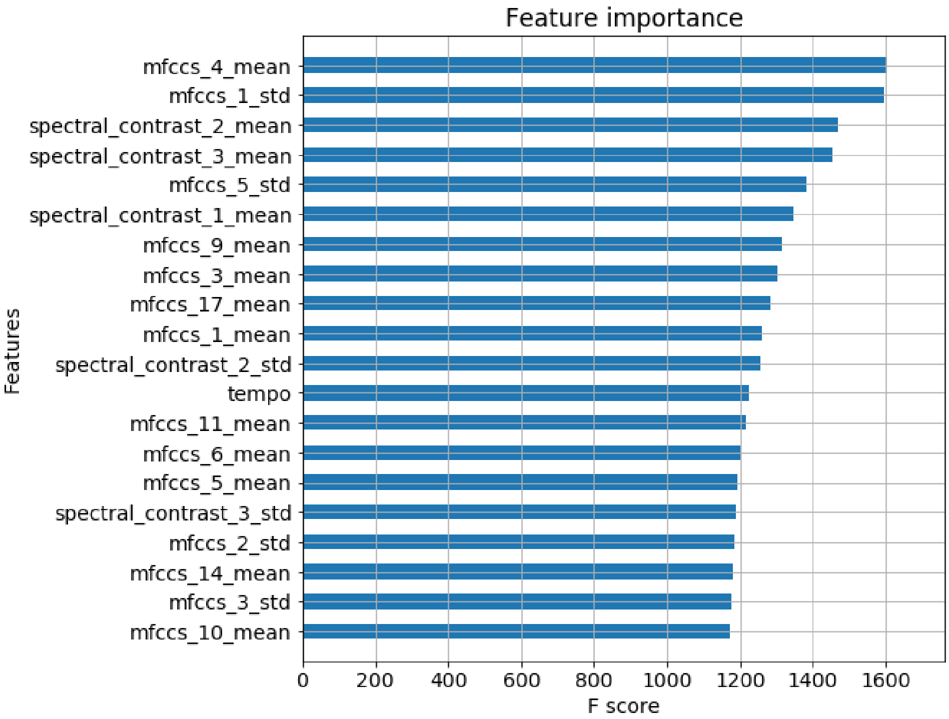


Figure 4: Relative importance of features in the XGBoost model; the top 20 most contributing features are displayed

the AUC metric and 4 point on the accuracy metric).

Table 3: Ablation Study: Comparing XGB performance keeping only top N features

N	AUC	Accuracy
10	0.803	0.47
20	0.837	0.52
30	0.845	0.55
97	0.865	0.59

The final experiment in this section is comparison of time domain and frequency domain features listed in Section 4.3. Two XGB models were trained - one with only time domain features and the other with only frequency domain features. Table 4 compares the results in terms of AUC and accuracy. This experiment further confirms the fact that frequency domain features are definitely better than time domain features when it comes to modelling audio for machine learning tasks.

5.2.2 Confusion Matrix

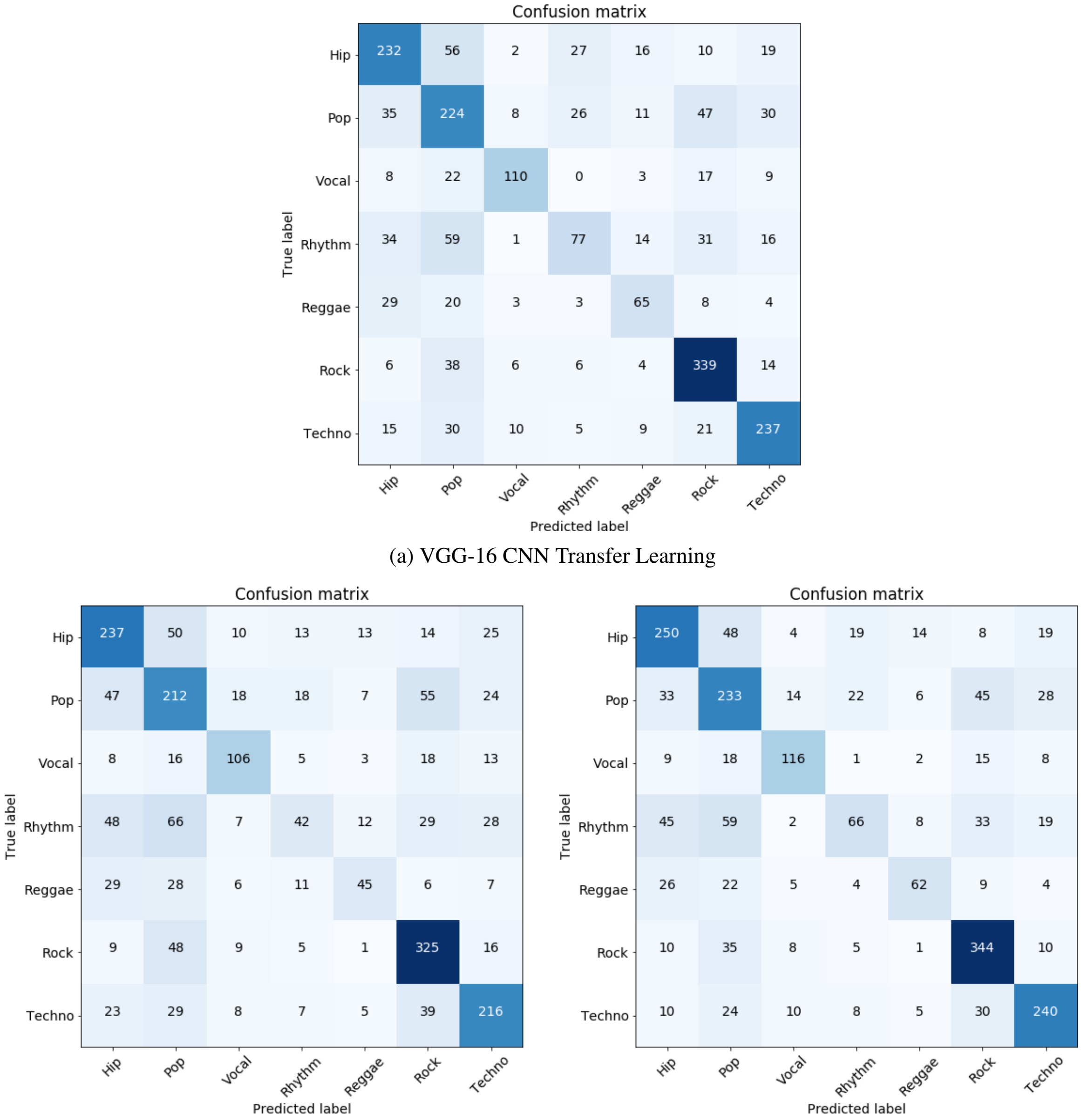
Confusion matrix is a tabular representation which enables us to further understand the strengths and weaknesses of our model. Element a_{ij} in the matrix refers to the number of test instances of class i that the model predicted as class j. Diagonal elements a_{ii} corresponds to the correct predictions. Figure 5 compares the confusion matrices of the best performing CNN model and XGB, the best model among the feature-engineered classifiers. Both models seems to be good at predicting the class 'Rock' music. However, many instances of class 'Hip Hop' are often confused with class 'Pop' and vice-versa. Such a behaviour is expected when the genres of music are very close. Some songs may fall into multiple genres, even as much that it may be difficult for humans to recognize the exact genre.

Table 4: Comparison of Time Domain features and Frequency Domain features

Model	AUC	Accuracy
Time Domain only	0.731	0.40
Frequency Domain only	0.857	0.57
Both	0.865	0.59

5.2.3 Ensemble Classifier

Ensembling is a commonly adopted practice in machine learning, wherein, the results from



(a) VGG-16 CNN Transfer Learning

(b) Extreme Gradient Boosting

(c) Ensemble Model

Figure 5: Confusion Matrices of the best performing models

different classifiers are combined. This is done by either majority voting or by averaging scores/probabilities. Such an ensembling scheme which combines the prediction powers of different classifiers makes the overall system more robust. In our case, each classifier outputs a prediction probability for each of the class labels. Hence, averaging the predicted probabilities from the different classifiers would be a straight-forward way to do ensemble learning.

The methodologies described in 4.2 and 4.4 use very different sources of input, the spectrograms and the hand-crafted features respectively. Hence, it makes sense to combine the models via ensembling. In this study, the best CNN model namely, VGG-16 Transfer Learning is ensembled with XGBoost the best feature engineered model by averaging the predicted probabilities. As shown in Table 2, this ensembling is beneficial and is observed to outperform the all individual classifiers. The ROC curve for the ensemble model is above that of VGG-16 Fine Tuning and XGBoost as illustrated in Figure 6.

6 Conclusion

In this work, the task of music genre classification is studied using the Audioset data. We pro-

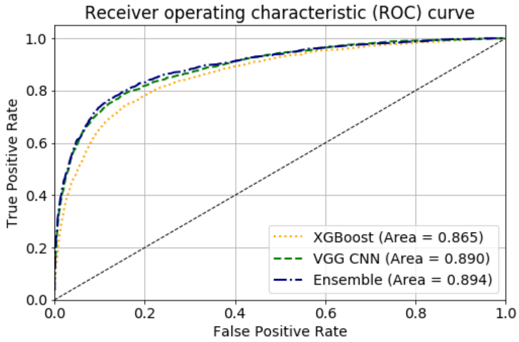


Figure 6: ROC Curves for the best performing models and their ensemble

pose two different approaches to solving this problem. The first involves generating a spectrogram of the audio signal and treating it as an image. An CNN based image classifier, namely VGG-16 is trained on these images to predict the music genre solely based on this spectrogram. The second approach consists of extracting time domain and frequency domain features from the audio signals, followed by training traditional machine learning classifiers based on these features. XGBoost was determined to be the best feature-based classifier; the most important features were also reported. The CNN based deep learning models were shown to outperform the feature-engineered models. We also show that ensembling the CNN and XGBoost model proved to be beneficial. It is to be noted that the dataset used in this study was audio clips from YouTube videos, which are in general very noisy. Futures studies can identify ways to pre-process this noisy data before feeding it into a machine learning model, in order to achieve better performance.

References

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. 2014. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing* 22(10):1533–1545.

Yali Amit and Donald Geman. 1997. Shape quantization and recognition with randomized trees. *Neural computation* 9(7):1545–1588.

Leo Breiman. 1996. Bagging predictors. *Machine learning* 24(2):123–140.

Leo Breiman. 2001. Random forests. *Machine learning* 45(1):5–32.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Steven B Davis and Paul Mermelstein. 1990. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In *Readings in speech recognition*, Elsevier, pages 65–74.

Dan Ellis. 2007. Chroma feature analysis and synthesis. *Resources of Laboratory for the Recognition and Organization of Speech and Audio-LabROSA* .

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* pages 1189–1232.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, pages 776–780.

Ricardo Garcia Gonzalez. 2006. Youtube-dl: download videos from youtube. com.

Fabien Gouyon, François Pachet, Olivier Delerue, et al. 2000. On the use of zero-crossing rate for an application of classification of percussive sounds. In *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*.

Peter Grosche, Meinard Müller, and Frank Kurth. 2010. Cyclic tempograma mid-level tempo representation for musicsignals. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, pages 5522–5525.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. The elements of statistical learnine.

Chadawan Ittichaichareon, Siwat Suksri, and Thaweesak Yingthawornsuk. 2012. Speech recognition using mfcc. In *International Conference on Computer Graphics, Simulation and Modeling (ICGSM'2012) July*. pages 28–29.

Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. 2002. Music type classification by spectral contrast feature. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*. IEEE, volume 1, pages 113–116.

Chanwoo Kim and Richard M Stern. 2012. Power-normalized cepstral coefficients (pncc) for robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, pages 4101–4104.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. pages 1097–1105.

Tom LH Li, Antoni B Chan, and A Chun. 2010. Automatic musical pattern feature extraction using convolutional neural network. In *Proc. Int. Conf. Data Mining and Applications*.

Thomas Lidy and Andreas Rauber. 2005. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *ISMIR*. pages 34–41.

Thomas Lidy and Alexander Schindler. 2016. Parallel convolutional neural networks for music genre and mood classification. *MIREX2016* .

Michael I Mandel and Dan Ellis. 2005. Song-level features and support vector machines for music classification. In *ISMIR*. volume 2005, pages 594–599.

Loris Nanni, Yandre MG Costa, Alessandra Lumini, Moo Young Kim, and Seung Ryul Baek. 2016. Combining visual and acoustic features for music genre classification. *Expert Systems with Applications* 45:108–117.

Andrew Y Ng. 2004. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, page 78.

Nicolas Scaringella and Giorgio Zoia. 2005. On the modeling of time information for automatic genre recognition systems in audio signals. In *ISMIR*. pages 666–671.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Hagen Soltau, Tanja Schultz, Martin Westphal, and Alex Waibel. 1998. Recognition of music types. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. IEEE, volume 2, pages 1137–1140.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Steve Tjoa. 2017. Music information retrieval. https://musicinformationretrieval.com/spectral_features.html. Accessed: 2018-02-20.

Suramya Tomar. 2006. Converting video formats with ffmpeg. *Linux Journal* 2006(146):10.

George Tzanetakis and Perry Cook. 2002. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing* 10(5):293–302.

Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* .

Lonce Wyse. 2017. Audio spectrogram representations for processing with convolutional neural networks. *arXiv preprint arXiv:1706.09559* .

E Zwicker and H Fastl. 1999. Psychoacoustics facts and models .