Project 3: Music Genre Classification

Esther Rodriguez
Department of Computer Science
University of New Mexico
Albuquerque, USA
resther@math.unm.edu

***Abstract*—A sample of the Free Music Archive (FMA) dataset, was classified by genre. There are 2400 songs of approximately 30 seconds each in the training set and 1200 songs of approximately the same length in the testing set. The songs belong to five genres: Rock, Pop, Folk, Instrumental, Electronic, or Hip-Hop. I used time series, Mel frequency spectrograms, and wavelets scattergrams to process the data and extract the features. Time series and Mel spectograms were used to train a 1D convolutional neural network and wavelets scattergrams were used to train a support vector machine. The convolutional neural network model gave the highest accuracy on predictions of the testing set submitted to Kaggle with 76% accuracy. The support vector machine model gave an accuracy of 50.8% on predictions on the testing set submitted to Kaggle.**

I. Introduction

Machine Learning (ML) algorithms apply mathematical and statistical models to data in order to extract useful information. This trained model can then be used to make predictions about new and incoming data. Classification as presented here is a supervised learning approach in ML that learns the model parameters based on the input observations and can then classify new observation. There are various classification methods that nowadays are widely used in different areas of science such as economics, genetics, topic categorization etc.

The problem of music genre classification consists assigning correct class (genre) labels to songs. A suitable algorithm is trained to associate a given song with its corresponding genre as agreed upon by musicians, producers and other music experts. A set of new or previously unknown songs is put into a classification function that uses the trained algorithm in the previous step, to give a label to each of these new tracks. By classifying music, the algorithm aims to assign a label to a song, making it easier to sort for databases. This is especially useful for publishers, critics, online streaming and archiving platforms, to name a few.

Music classification, in one sense, is a well-specified task since its measure of performance is the accuracy of its predictions. On the other hand, for decades composers, writers, musicians, and artists have mixed and melded what had previously been considered different genres to create new ones. Often these new styles are named in a manner descriptive of their parts, and automated classification can easily be confused by the use of unifying themes in what are now considered different genres.

A dataset of 2,400 songs is available for training, 1,920 (80%) of which are used for feature extraction for the actual learning with supervised learning algorithms and the remaining 480 (20%) are used for validation. The testing dataset contains 1,200 songs and testing occurs on the Kaggle website. A ML approach is a viable option for the task of classifying songs into their corresponding categories. In particular, I present time series (untransformed sound file), Mel frequency spectrograms, and wavelets scattergrams classifiers as examples of conceptually simple feature representations as input for ML algorithms in multi-genre classification. There are several classification algorithms that are commonly used for music classification, including but not limited to decision trees, random forests, and Naïve Bayes (NB), logistic regression (LR). Specifically, the classification algorithms I present here are: 1) a convolutional neural network for spectrogram classification as an image classifier; 2) a support vector machine as a classifier on time series and wavelet scattergrams. Rather than reuse NB and LR, I thought it would be good to explore more classifiers.

II. Methodology

A. Data Description

Classification of music is necessary to build searchable catalogues and can yield insights into the underlying structure of melody, beats, and instrumentation, for example, of a given genre as can be gleaned from a music repository. This project works with a 2,400 song dataset of 6 equally represented genres; that is, 400 songs of each genre. The musical genres represented in the dataset are presented in Table I. The songs are taken from the Free Music Archive (FMA) dataset and come with their corresponding genre classification label. Anpther set of 1,200 songs is made available as testing data which does not contain the classification labels. The accuracy on testing data is calculated by the Kaggle data science competition website.

The entirety of the training data is is prepared for feature extraction by downsampling from the original 44,100Hz sampling rate, which presented small variability across songs, to 16,000Hz uniformly for all training data. Once this is done for all 2,400 songs, they are ready for feature extraction.

B. Time Series Analysis

Time series analysis refers to the statistical and structural analysis of a sequence or stream of data that follow time ordering. As such, time series can be discrete or continuous, with the former being more common in applications due to

Music Genre Category Labels
0 Rock
1 Pop
2 Folk
3 Instrumental
4 Electronic
5 Hip-Hop

TABLE I: List of the 6 music genre that correspond to labels/classes for our dataset. The number to the left of the name is used in the last column of the training data.

discrete measurements or digital quantization of a continuous series. For the purpose of music classification, time series are discrete data that encode the recorded amplitude of performed music. The recording process and subsequent digitization take what is a continuous waveform produced by musical instruments and record a digital representation of sound. For accurate reproduction of a song, the necessary sampling rate of the recording is given by the Shannon-Nyquist frequency, which is twice the highest frequency present in the original signal [1]. For the training set used, almost all songs were sampled at 44,100Hz with little variation around this value.

C. Mel Frequency Sprectrogram

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency [3]. The cepstrum of a signal is the power of the inverse Fourier transform of a signal's logarithmic power spectrum.

Mel-frequency cepstral coefficients (MFCCs) are derived from a type of cepstral representation of the audio clip. The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, which makes it an intuitive feature to use in music classification by genres.

MFCCs may be derived as follows

1) Take the Fourier transform of the signal.
2) Map the powers of the spectrum obtained above onto the mel scale with, for example, the formula $m = 2595 \log_{10}(1 + f/700)$.
3) Take log of the powers at each of the mel frequencies.
4) Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5) The MFCCs are the amplitudes of the resulting spectrum.

This procedure is implemented by Python's `librosa` library in practice.

D. Wavelets Scattergram

Wavelets are a family of kernels of linear transformations, as such, the coefficients have the form

$$w_{k\ell} = \int x(t) \frac{1}{\sqrt{2^k}} \psi \left(\frac{t - 2^k \ell}{2^k} \right) dt, \tag{1}$$

where ψ is a wavelet mother funciton [4]. Wavelets also enjoy the following properties which make them a go-to for compression [5]:

1) Filterbank implementations have $O(N)$ complexity [4].
2) They are parameterized by scale and location so that fine and coarse features may be captured throughout the signal's support.
3) Wavelets promote compression sparsity in the sense that they require a small subset of largest-magnitude coefficients for high-fidelity signal reconstruction [6].

Figure (1) shows the frequency response of the wavelet filterbanks used in the project.

E. Convolutional Neural Networks

Neural networks (NNs) are universal *function approximators*. That is to say, their purpose is to compute high-dimensional, nonlinaer functions by successively improving on a function approximation by adding to it the output of a simple funciton of features. This might take the form of a linear combination of features, or an activation such as a hyperbolic tangent or softmax. As the name suggests, NNs are a network, i.e. interconnected set of basic building blocks, of neurons. Neurons are computational units that carry out a part of the global approximation by computing some of the above mentioned functions. There are connections between neurons of successive layers that carry a weight depending on how important an input is to a given neuron in the next layer. In this setting, the learning refers to estimations of the weights between neuronal layers and the optimal parameters of the function evaluated in each neuron.

The building blocks of a convolutional neural network (CNN) are layers of NNs, where each layer computes the convolution of its input with a kernel that needs to be learned during training [7], [8]. The usefulness of CNNs comes from their versatility in learning important features of complicated objects like time series (1D) or images (2D). There is also the added advantage that convolution can be performed efficiently through matrix multiplication by putting the convolution filter into a Toeplitz matrix and left-multiplying the input by the convolution matrix (reshaped into a 1D array) [9]. When the filter is the same size as the input (or can be recast into this shape), the convolution can be effected by the convolution theorem of the Fourier transform whose [5]. Lastly, deep NNs, whether convolutional or otherwise, allow for parameter regularization in the cost function.

F. Support Vector Machines

Support vector machines (SVMs) are supervised learning models for classification and regression. In essence, SVMs are linear maximum margin classifiers, in the sense that they

find a separating plane that will be maximally distant to any data class, making for maximal separation of different classes represented in data [10].

For data that cannot be linearly classified, SVMs can be transformed from the data space into feature space by the so-called *kernel trick*. The intuition behind it is to find a suitable nonlinear transform of the data so that the transformed data is linearly separable in the higher (possibly infinite) dimensional space. What looks like a plane in the new space looks like an arbitrary curve in Euclidean space [10]. Hence, SVMs are capable of very general classification tasks by taking the data space into account. Lastly, SVMs allow for parameter regularization in the cost function.

III. Implementation

All songs were preprocessed from their time series by downsampling for faster implementation.

A. Time Series and Mel Frequency Spectogram

As noted before, by time series is meant use of song amplitudes directly. No further evaluation is needed to use this as a feature, which is why this has been chosen as a data representation. Frequencies occurring in songs are found through the fast Fourier transform (FFT) [5], which is another reason to use time series in this analysis. As has been noted before, the time series is used first to extract features such as frequency spectrum with the FFT.

Frequency and amplitude distributions are used as features of a genre in classification. The frequency time series is referred to as the *spectrum* is the starting point for the cepstrum analysis that was also performed.

My implementation of this is in Python. The preprocessed data is transformed by the FFT and from there any number of features can be calculated with the `librosa` music analysis library. Figure 4 shows the real and imaginary part of a wavelet function and its scaling function. The latter can take account of important features that occur on fast and slow time scales by raising it to higher or lower powers, respectively.

The function `mfcc` from the Python Speech Features library is used to obtain the Mel frequency cepstrum and spectrogram. function shown in Fig. 5. The Mel spectrogram was chosen since they are built specifically for representing frequencies in a way that is closer to how humans hear melody as compared to regular frequency.

B. Wavelets Scattergram

This framework uses wavelets and a low-pass scaling function to generate low-variance representations of a song's time series. Wavelet time scattering yields representations insensitive to translations in the input signal without sacrificing class discriminability [4], [11]. This is one important reason for my chosing to work with them. The scattergram is a 2D function of time series frequency and time, with the value returned being the amplitude of a wavelet scattering coefficient. This was implemented in MATLAB. Lastly, I also chose this representation for the efficiency and speed of implementation.

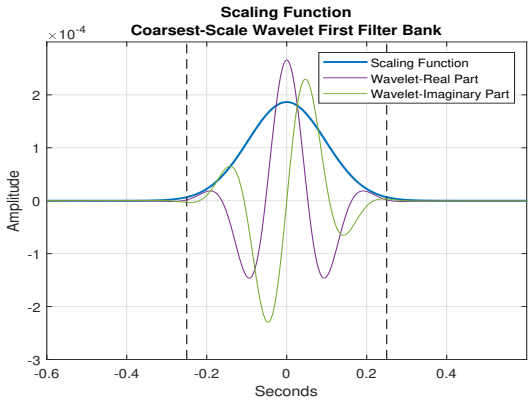

Fig. 1: The scaling filter (blue) in time along with the real (purple) and imaginary (green) parts of the coarsest-scale wavelet from the first filter bank.

C. Convolutional Neural Networks

I use a CNN programmed with the `keras` library in Python to carry out feature selection of the mfcc. I start with a hyperbolic tangent activation. Then, following standard structure, I add convolutional layers of increasing size to refine the scale of the learned features. This is followed by pooling of current features, flattening of the network, activation through dense layers of decreasing size to integrate the current function approximation and finally put it through a softmax classification layer. See Fig. 2 for a code snippet of the CNN implementation. The rationale for using a CNN is that they are very effective at finding complicated relationships in image and this is exploited for excellent image classification [7]. Thus, this property of CNNs can be exploited to classify songs from their 2D features such as mfccs.

```
### CNN Model ###
def get_conv_model():
    model = Sequential()
    model.add(Conv2D(8, (7,7), activation='tanh', strides=(1,1),
                     padding='same', input_shape=input_shape))

    model.add(Conv2D(16, (5,5), activation='relu', strides=(1,1),
                     padding='same', input_shape=input_shape))
    model.add(Conv2D(32, (5,5), activation='relu', strides=(1,1),
                     padding='same'))
    model.add(Conv2D(64, (5,5), activation='relu', strides=(1,1),
                     padding='same'))
    model.add(Conv2D(128, (5,5), activation='relu', strides=(1,1),
                      padding='same'))
    model.add(MaxPool2D((2,2)))
    model.add(Dropout(0.1))
    model.add(Flatten())
    model.add(Dense(128,activation='relu'))
    model.add(Dense(64,activation='relu'))
    model.add(Dense(6,activation='softmax'))
    model.summary()
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam', metrics=['acc'])
    return model
```

Fig. 2: Code snippet of the Convolutional Neural Network.

D. Support Vector Machines

The computational implementation of the SVM classifier was done in MATLAB. See Fig. 3 for a code snippet of the SVM classifier implementation.

```
%% SVM Model

template = templateSVM(...
    'KernelFunction', 'polynomial', ...
    'PolynomialOrder', 3, ...
    'KernelScale', 'auto', ...
    'BoxConstraint', 1, ...
    'Standardize', true);
Classes = {'0','1','2','3','4','5'};
classificationSVM = fitcecoc(...
    TrainFeatures, ...
    trainLabels, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
    'ClassNames', categorical(Classes));
```

Fig. 3: Code snippet of the Support Vector Machine classifier.

The input to the SVM was the wavelet scattergram. The kernel in this instance of SVM classificiation with wavelet scattergram features is a 3rd order polynomial with standardized data to fit the 6 classes (genres) present in the data. The wavelet representation was chosen as a fast and compressive implementation of a feature. One reason for choosing wavelet transforms as features is that they can be computed in $O(N)$ time, making it a desirable feature for learning. Another reason for choosing them, is the compressive property of wavelets is that there are half as many coefficients in for a time series than there are points in the series. Furthermore, a property can may often be exploited for very high compression of the source is that most of the power of the wavelet coefficients is found in a small percentage of the coefficients ($<$10% and as few as 4% or less) [4], [6]. Finally, I chose SVMs as they are a fundamental algorithm of ML that has a long successful history in classification [10] and there are ready-made implementations [11].

IV. Results

A. Convolutional Neural Networks

Figure 4 shows the time series representation of one song for each of the six genres. We can see that at the time series level, the amplitudes have visibly different envelop functions for each of the genre. It is interesting that the time series for Pop, Hip-Hop and electronic agrees with intuition, as these are related genre and their time series have similar behaviour.

However, these similarities between the time series representation of different genre would make classification based only on these features equivocal. For this reason the FFT is applied to the signal and the Mel spectrogram computed since the former will separate frequencies, and the latter nonlinear transformations lead to further distinction of features from the original time series. In particular the logarithm in the cepstrum will have the effect of separating out very low frequencies that otherwise may be indistinguishable at the time series level.

Furthermore, adding the Mel spectrogram and FFT adds many more features to reduce equivocation. The Mel spectrogram shown in Fig. 5 shows the resulting iamge for each genre. A close look reveals, even to the human eye, that there is now distinguishability between since Pop has more power at

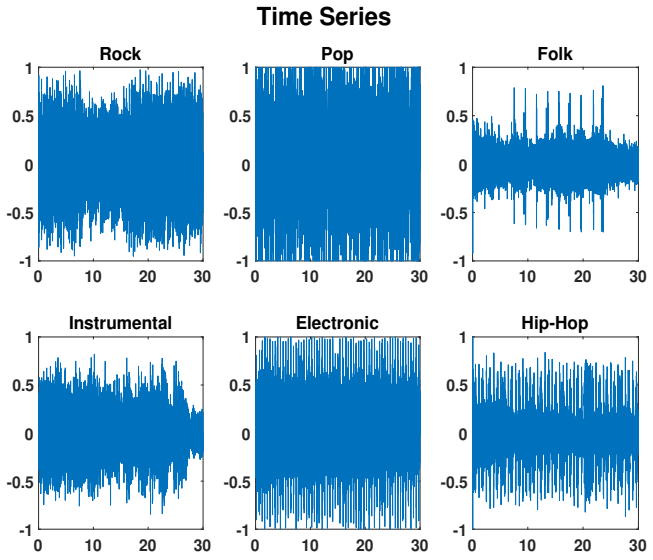

Fig. 4: Time Series representation of the data. For each genre class, one song corresponding to that genre is shown in its Time Series representation.

low frequencies than Electronic, and Hip-Hop shows a fainter spectrogram, indicating less power over all, and less density of high frequencies over time.

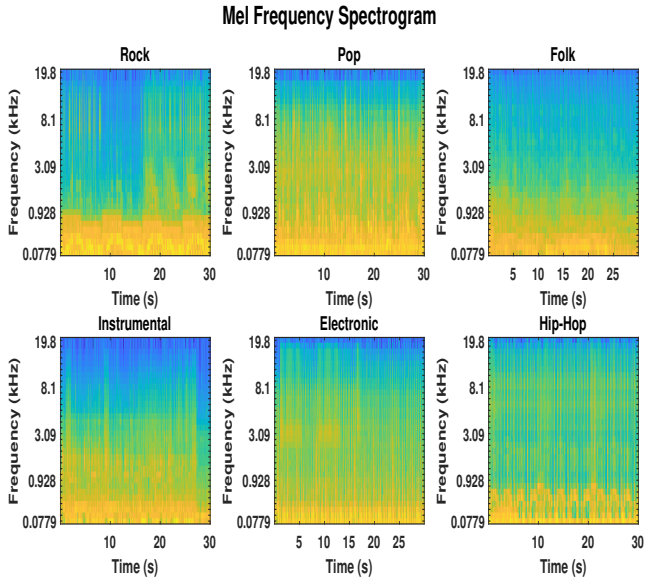

Fig. 5: Mel Frequency Spectrogram representation of the data. For each genre class, one song corresponding to that genre is shown in its Mel Frequency Spectrogram representation.

The CNN yielded higher classification accuracy than the SVM. My CNN, which is based on the code found at `https://github.com/seth814/Audio-Classification`, originally gave classification rates of 40%. I tuned my CNN by adding the initial

hyperbolic tangent layer as a means to start separating the different genres before learning specific features. I increased the size of the kernels to have more learnable parameters and reduced the dropout rate to 0.1. This tuning resulted in a 76% accuracy on Kaggle and 88.3% accuracy with 80-20 testing data split. I retained usage of the Adam optimizer for stochastic gradient descent. Exploration of batch size and number of epochs revealed that 128 and 20, respectively, yielded the best results.

B. Support Vector Machines

The wavelet scattergram filter banks shown in Figs. 6 and 7. They are a visual representation of the wavelet scattering coefficients corresponding to a song as a function of sound frequency and time. There are two, one for each filter in the filterbank. These scatter grams are the features that the SVM with 3rd degree polynomial kernel will use for genre classification. The accuracy obtained was 50.8% on Kaggle and 50.9% accuracy with 80-20 testing data split. This may be improved by further exploration of the parameter space. I tried different low degree polynomial kernels and found degree 3 to perform better than degree 1 or 2. The effect of the regularization parameter was not too noticeable as compared to the degree of the polynomial kernel in my experiments. MATLAB's *waveletScattering* function was given the inputs of 2^{19} for signal length, a sampling frequency of 22,050Hz, and 0.5 for the invariant scale of the filterbank functions (blue curve in Fig. 1).

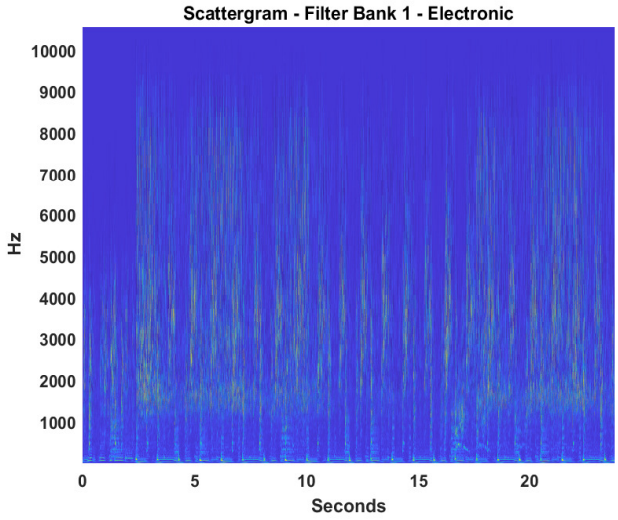

Fig. 6: Scattergram representation for the first filter bank applied to wavelets. A random song corresponding to the genre of Electronic music was chosen for this figure.

The SVM training, however, was about 5 times faster (16min vs. 77min) than the CNN training, even though the CNN was run on an NVIDIA GTX 1080 Ti with 11Gb of dedicated DDR5 RAM and the SVM was run on an Intel i7 8700K 6-core processor with 64Gb of DDR4 system RAM on a Windows 10 machine.

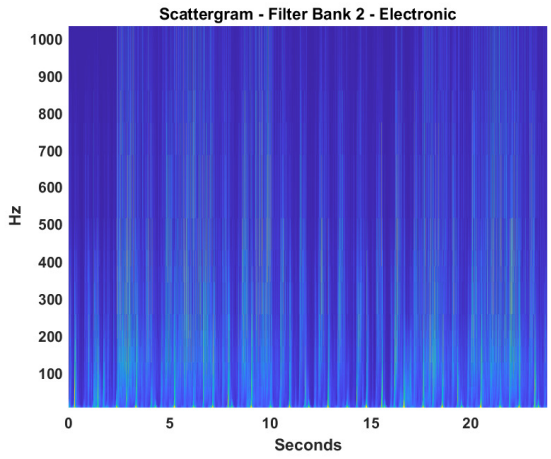

Fig. 7: Scattergram representation for the second filter bank applied to wavelets. A random song corresponding to the genre of Electronic music was chosen for this figure.

C. Overall Testing Accuracy, Confusion Matrix and Confidence Interval

The CNN outperformed the SVM in classification, 76% to 50.9% on Kaggle. While the selected features for each classifier may have some bearing on this result, it is much more likely, based on extensive numerical experimentation, that the classifier algorithms themselves account for most of the discrepancy observed. The CNN has many more learnable parameters than the SVM stemming from the convolution filters and number and size of layers.

The *confusion matrix* is handy visualization of the classification performance. The diagonal elements of this matrix are the number of correctly classified documents, while the off-diagonal elements, C_{ij} are the times a document of class j was erroneously classified as i. The mean over all confusion matrix elements is the classification performance.

Genre	**Training Split** (80%)	**Testing Split** (20%)
0 Rock	320 Songs	80 Songs
1 Pop	320 Songs	80 Songs
2 Folk	320 Songs	80 Songs
3 Instrumental	320 Songs	80 Songs
4 Electronic	320 Songs	80 Songs
5 Hip-Hop	320 Songs	80 Songs

TABLE II: Number of songs of each genre on the 80-20 split of the training data for testing accuracy

In Table II, I report the training data split used. Note that there are exactly the same number of songs per genre with an $80-20$ split, yielding 320 songs for training and 80 for testing for each genre.

Figures 8 and 9 show the confusion matrices for the CNN and SVM classifiers, respectively. Interestingly, the CNN confusion matrix is upper diagonal. This is likely do to the order in which data were presented and the learned features for

the first few songs from a new category were not good enough for classification. This may be partially addressed by further reshuffling of data. The confusion matrix for the SVM shows high values off diagonal, consistent with the 50% classification rate.

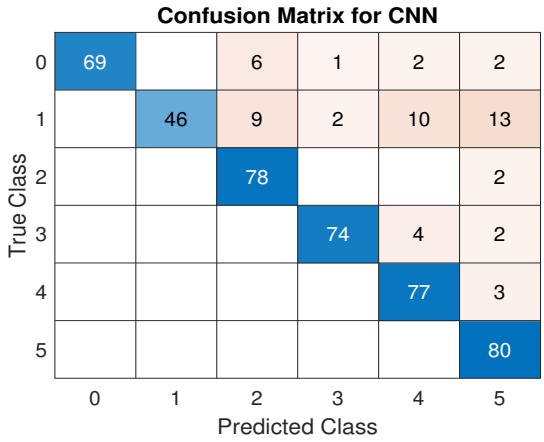

Fig. 8: Confusion matrix for the Convolutional Neural Network classifier on an 80-20 split of the training data. This parameters correspond to the Convolutional Neural Network classifier with 5 activation layers, one $\tanh$ and four $ReLu$ layers.

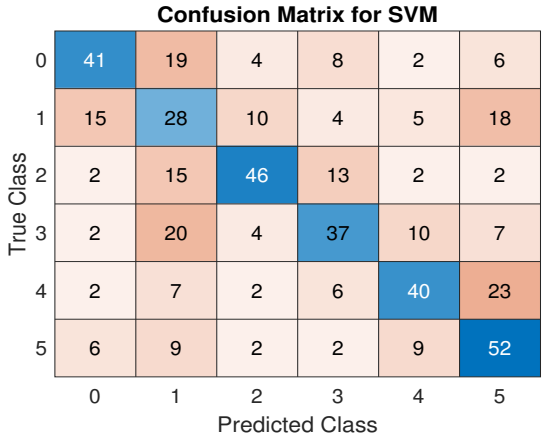

Fig. 9: Confusion matrix for the Support Vector Machine classifier on an 80-20 split of the training data. This parameters correspond to the Support Vector Machine classifier with a polynomial of third order as the kernel function.

Interestingly, in general, C is not symmetric. This, of course, is not surprising, since feature patters from songs that are prominent in two genres may lead to incorrect classification, and correcting this would perhaps require a nonlinear score function to be passed into the softmax function in the final CNN layer, or an altogether different and much more nonlinear kernel for the SVM. Thus, for example, as Pop, Electronic, and Hip-Hop often use the same or similar beat patters and tempo they may be confused with each other. Similarly, and quite prominently, rock has had much historic influence from folk music, often using similar tempo and some melodies leading to confusion between them. Instrumental music can vary greatly as the only requirement is that there be no vocals, but otherwise anything can be instrumental. Some of the training data had prolonged instrumental periods even if the song itself was not labeled as instrumental. This has lead to further misclassification.

The 95% confidence intervals for the testing accuracies on 20% of the training data are

$$\mathrm{CI}_{\mathrm{radius}} = 1.96\sqrt{\frac{0.883(1-0.883)}{480}} = 0.029 \qquad (2)$$

for the convolutional neural network and

$$\mathrm{CI}_{\mathrm{radius}} = 1.96\sqrt{\frac{0.5(1-0.5)}{480}} = 0.045 \qquad (3)$$

for the support vector machine. It is expected that there is higher confidence in the performance of the CNN as it yielded higher classification, thus the uncertainty band around CNN classification is smaller than the SVM, which is reflected by a smaller interval, i.e. tighter around the predictions.

D. Bias Analysis

I see indications of training set bias as evidenced by the confusion matrices. The indication comes from higher confusion between some genres than others and the upper triangular nature of the CNN confusion matrix. For an unbiased set, the confusion is expected to have a large diagonal and the rest of its elements should be uniformly distributed, not preferentially as in the CNN confusion matrix. This is likely due to the size of the data set. Music has much nuance and this requires very large samples to unequivocally classify genres, especially when different genres may be similar to each other. The SVM confusion matrix does not show such a strong indication of bias at first glance. However, the poor classification of SVMs in this case leads me to believe that the songs contained in the training data set may not be quite representative of their genre, leading to high confusion in classification.

V. Discussion

I found that my implementation of an SVM classifier, using wavelets coefficients as features, learns much faster than my implementation of a CNN classifier, using Mel frequency spectograms as features. However, my CNN outperforms the SVM by more than 25%. This may be due to further need for optimization of the SVM. The wavelets coefficients used for the SVM classifier can carry the same or even more information than the Mel frequency spectogram, so the choice of data representation and feature extraction is likely not causing problems. It is more likely that improvement in classifier performance may lie in making use of simpler features in case overfitting has been an issue with my implementations. For future work on the improvement of the model accuracy, I suggest using a 1D CNN with less activation and dense layers, trying different kernel functions for the SVM model, and combining both models in a random forest that can also use a logistic regression model.

VI. Conclusion

In this paper, I implemented a 2D Convolutional Neural Network classifier. I also implemented a Support Vector Machine classifier on wavelet coefficients. Both algorithms were used for for music genre classification of nearly 1,200 songs over six different genre classes. My results show that the best accuracy obtained form Kaggle, 76%, was obtained by both the CNN classifier trained on 20 epochs with a 0.1 dropout. The SVM classifier approach got a lower accuracy on Kaggle of 50.8%. On our testing scenario of an 80-20 split on the training data, the CNN approach achieved 88.3% accuracy while the SVM approach achieved a 50.9% accuracy. From the confusion matrices we see that the Pop genre often gets confused with any of the other genres, while the Instrumental and Electronic genre often get confused with each other, as can be expected by the similarity on features. Furthermore, I noted that the training data showed some level of bias, as determined by analysing the confusion matrices and comparing the accuracy on the 80-20 split of the training set with the accuracy given by Kaggle on the testing set.

References

[1] T. M. Cover, J. A. Thomas, *Elements of Information Theory*, 2nd Ed. Wiley, 2006.

[2] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, 2009.

[3] Meinard Müller (2007). Information Retrieval for Music and Motion. Springer..

[4] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed. San Diego, CA: Academic, 1999

[5] S. A. Broughton, K. Bryan, *Discrete Fourier Analysis and Wavelets: Applications to Signal and Image Processing*, Wiley, 2009.

[6] D.L. Donoho, IEEE Transactions on Information Theory (Volume: 52, Issue: 4 , April 2006)

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[8] C. C. Aggarwal, *Neural Networks and Deep Learning. A Textbook*, Springer, 2018.

[9] A. Böttcher, S. M. Grudsky, *Toeplitz Matrices, Asymptotic Linear Algebra, and Functional Analysis*, Birkhäuser, 2012.

[10] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007.

[11] MATLAB documentation.