Music Genre Classification

Derek A. Huang
huangda@stanford.edu

Arianna A. Serafini
aserafini@stanford.edu

Eli J. Pugh
epugh@stanford.edu

https://github.com/derekahuang/Music-Classification

1 Introduction

Genre classification is an important task with many real world applications. As the quantity of music being released on a daily basis continues to sky-rocket, especially on internet platforms such as Soundcloud and Spotify – a 2016 number suggests that tens of thousands of songs were released every month on Spotify – the need for accurate meta-data required for database management and search/storage purposes climbs in proportion. Being able to instantly classify songs in any given playlist or library by genre is an important functionality for any music streaming/purchasing service, and the capacity for statistical analysis that correct and complete labeling of music and audio provides is essentially limitless.

We implemented a variety of classification algorithms admitting two different types of input. We experimented with a RBF kernel support vector machine, k-nearest neighbors, a basic feed-forward network, and finally an advanced convolutional neural network. For the input to our algorithms, we experimented with both raw amplitude data as well as transformed mel-spectrograms of that raw amplitude data. We then output a predicted genre out of 10 common music genres. We found that converting our raw audio into mel-spectrograms produced better results on all our models, with our convolutional neural network surpassing human accuracy.

2 Related Work

Machine learning techniques have been used for music genre classification for decides now. In 2002, G. Tzanetakis and P. Cook [8] used both the mixture of Gaussians model and k-nearest neighbors along with three sets of carefully hand-extracted features representing timbral texture, rhythmic content and pitch content. They achieved 61% accuracy. As a benchmark, human accuracy averages around 70% for this kind of genre classification work [4]. Tzanetakis and Cook used MFCCs, a close cousin of mel-spectrograms, and essentially all work has followed in their footsteps in transforming their data in this manner. In the following years, methods such as support vector machines were also applied to this task, such as in 2003 when C. Xu et al. [9] used multiple layers of SVMs to achieve over 90% accuracy on a dataset containing only four genres.

In the past 5-10 years, however, convolutional neural networks have shown to be incredibly accurate music genre classifiers [8] [2] [6], with excellent results reflecting both the complexity provided by having multiple layers and the ability of convolutional layers to effectively identify patterns within images (which is essentially what mel-spectrograms and MFCCs are). These results have far exceeded human capacity for genre classification, with our research finding that current state-of-the-art models perform with an accuracy of around 91% [6] when using the full 30s track length. Many of the papers which implemented CNNs compared their models to other ML techniques, including k-NN, mixture of Gaussians, and SVMs, and CNNs performed favorably in all cases. Therefore we decided to focus our efforts on implementing a high-accuracy CNN, with other models used as a baseline.

3 Dataset and Features

We obtained all of our musical data from the public GTZAN dataset. This dataset provides us with 1000 30-second audio clips, all labeled as one out of 10 possible genres and presented as .au files. From each clip, we sampled a contiguous 2-second window at four random locations, thus augmenting our data to 8000 clips of two seconds each. Since this data was sampled at 22050HZ, this leaves us with 44100 features for the raw audio input. We restricted our windows to two seconds to limit the number of features. We found that 44100 features was the perfect balance between length of audio sample and dimension of feature space. Thus after pre-processing our input is of shape (8000, 44100), where each feature denotes the amplitude at a certain timestep out of the 44100. We also used 100 samples of un-augmented data each of our cross validation and test sets.

We also experimented with pre-processing our data by converting the raw audio into mel-spectrograms. In doing this, we experienced significant performance increases across all models. Mel-spectograms are a commonly used method of featurizing audio because they closely represent how humans perceive audio (i.e. in log frequency). In order to convert raw audio to mel-spectogram, one must apply short-time Fourier transforms across sliding windows of audio, most commonly around 20ms wide. With signal $x[n]$, window $w[n]$, frequency axis ω, and shift m, this is computed as

$$\textbf{STFT}\{x[n]\}(m,\omega) = \sum_{n} x[n]w[n-m]e^{-j\omega n}.$$

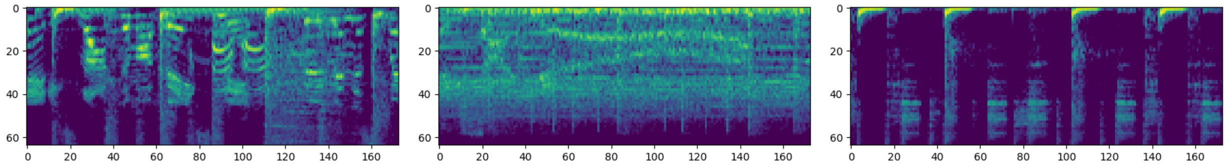
These are computed more quickly in practice using sliding DFT algorithms. These are then mapped to the mel scale by transforming the frequencies f by

$$m = 2595\log_{10}(1+\frac{f}{700}).$$

Then we take the discrete cosine transform of the result (common in signal processing) in order to get our final output mel-spectogram.

In our case, we used the Librosa library [5] and chose to use 64 mel-bins and a window length of 512 samples with an overlap of 50% between windows. Based on previous academic success with such transformations, we then move to log-scaling using the formula $\log(X^2)$.

The resulting data can be visualized below:



Hip-Hop

Metal

Reggae

Figure 1: Examples of log-scaled mel-spectrograms for three different genres.

3.1 Principal Component Analysis

In addition to pre-processing our data as described above, we also used Principal Component Analysis to reduce dimension for two of our models (k-NN and SVM). PCA is a form of factor analysis that tries to approximate the data by projecting it onto a subspace of lower dimension. In order to do this, we first transformed the mel-spectrograms by normalizing their mean and variance. In order to preserve as much variance as possible with m examples $x^{(i)}$, unit length u, PCA maximizes

$$\frac{1}{m}\sum_{i=1}^{m}(x^{(i)^T}u)^2 \;=\; \frac{1}{m}\sum_{i=1}^{m}x^{(i)}x^{(i)^T}.$$

Since our data has mean 0, this is equivalent to taking the principal eigenvector of Σ, the covariance matrix of the data. Empirically, we had best results reducing to 15 dimensions, analogous to taking the top 15 eigenvectors of Σ and projecting our data onto the subspace spanned by them. We implemented this using scikit-learn [7].

4 Methods

We decided to first implement two simpler models as baseline measures of performance, then progress to more complicated models in order to increase accuracy. We implemented variants of k-nearest neighbors, support vector machine, a fully connected neural network, and a convolutional neural network. In addition to mel-spectogram features, we used principal component analysis to reduce dimensionality for the input to k-NN and SVM.

4.1 K-Nearest Neighbors

After reducing the dimensionality to 15 features using PCA (see above), we applied the k-nearest neighbors algorithm. Predictions are made on a per-case basis by finding the closest training examples to our test or cross-validation example that we wish to classify and predicting the label that appeared with greatest frequency among their ground-truth labels. Through trial and error, we found that best accuracy resulted from setting $k = 10$ and weighting the label of each neighbor by distance.

Explicitly, denoting our data point as x, let $x^{(1)}, \dots, x^{(10)}$ be the 10 closest neighbors to x, those which return the largest value on $||x - x^{(i)}||_2$, where $|| \cdot ||_2$ denotes the Euclidean distance between points. Then, we choose weights w_i for each i such that

$$w_i \propto ||x - x^{(i)}||_2, \quad \sum_{i=1}^{1} 0w_i = 1.$$

Finally, we return

$$\arg\max_y \sum w_i \mathbb{1}(y = y^{(i)}),$$

the label which is most prevalent among x's 10 nearest neighbors when weighted by distance. This was implemented with scikit-learn [7].

4.2 Support Vector Machine

After reducing dimensionality using PCA, we trained an SVM classifier as well. SVMs are optimal margin classifiers that can use kernels to find more complex relations in the input data. Since our data is not linearly separable, this equates to finding

$$\min_{\gamma, w, b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \xi_i \quad \text{s.t. } y^{(i)}(\sum_j \alpha_j K(x^{(j)}, x^{(i)}) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

where $x^{(i)}$ are examples, α_j, b are weights and biases, C is a penalty parameter, and $1 - \xi_i$ is the functional margin for example i. $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function. In a traditional SVM this function corresponds to inner product between two vectors $x^{(j)}, x^{(i)}$, but in our case we are using an RBF (radial basis function) kernel:

$$K(x^{(j)}, x^{(i)}) = \exp\left(-\frac{||x^{(j)} - x^{(i)}||_2^2}{2\sigma^2}\right).$$

This kernel, also sometimes called the Gaussian kernel, corresponds to an infinite dimensional feature space is related to Euclidean distance. This function as a whole is often minimized using sequential minimization optimization. This was implemented with scikit-learn [7].

4.3 Feed-Forward Neural Network

We used a fully connected neural network as well, with ReLU activation and 6 layers, with cross-entropy loss. As the input to our model was 1D, when using mel-spectrograms, we flattened the data. Our model is fully connected, which means each node is connected to every other node in the next layer. At each layer, we applied a ReLU activation function to the output of each node, following the formula:

$$\text{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0. \end{cases}$$

At the end, we construct a probability distribution of the 10 genres by running the outputs through a softmax function:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

To optimize our model, we minimized cross entropy loss:

$$CE(\theta) = -\sum_{x \in X} y(x) \log \hat{y}(x)$$

We experimented with various regularization techniques, such as dropout layers and L2 regularization. Dropout randomly selects features to drop based off a specified constant, and L2 regularzation adds a penalty term to the loss function in the form of $\lambda \sum_i \theta_i^2$. This was implemented with TensorFlow [1].

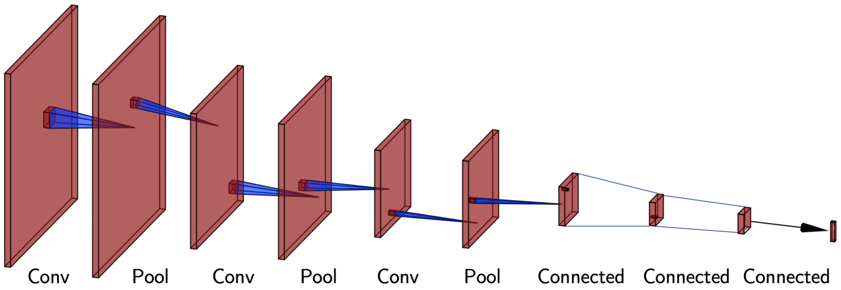


Figure 2: CNN architecture.

4.4 Convolutional Neural Network

This was our most advanced model, using 3 convolution layers, each with its own max pool and regularization, feeding into 3 fully connected layers with ReLU activation, softmax output, and cross entropy loss. Most of the equations can be found above, and our architecture is visually presented below:

This approach involves convolution windows that scan over the input data and output the sum of the elements within the window. This then gets fed into a max pool layer that selects the maximum element from another window. Afterwards, the output is fed through a model described in section 4.1. This was implemented with TensorFlow and Keras [1] [3].

5 Results and Discussion

The main quantitative metric which we used to judge our models is accuracy (that is, percentage of predicted labels which matched their true labels), and our main way of visualizing the performance of our best model is through the confusion matrices as seen in Figure 3. Because the labeling was uniformly distributed on our data, cross-validation, and test sets, these confusion matrices offer not only a way to visualize our data, but more specific information than precision and recall values offer.

We selected our hyperparameters based of empirical results and industry standards. For instance, choosing 4 by 4 window for our first convolution window was a result of seeing a similar window size work in other academic results, and then fine tuning to meet our data-specific needs. We chose to use Adam optimization for a few reasons. Working with audio time-series data over 2 dimensions cause sparse gradient problems, similar to those often encountered in natural language or computer vision problems. We also felt our data was somewhat noisy and messy. Adam mitigates the sparse gradient problem by maintaining a per-parameter learning rate, and mitigates the noise problem by basing updates on a weighted average of recent updates (momentum). With Adam our models trained more quickly and didn't plateau as early.

Consult Table 1 for model accuracy both without data processing – i.e. with raw audio as input – and with data processing – i.e. using the scaled mel-spectrograms as input.

Table 1: Accuracy of predictions by model used.

	With data processing			Without data processing		
	Train	CV	Test	Train	CV	Test
Support Vector Machine	.97	.60	.60	.75	.32	.28
K-Nearest Neighbors	1.00	.52	.54	1.00	.21	.21
Feed-forward Neural Network	.96	.55	.54	.64	.26	.25
Convolution Neural Network	.95	.84	**.82**	.85	.59	.53

The clearest trend identifiable here is the dramatic jump in performance after we moved from using raw audio to mel-spectrograms. We saw a substantial improvement in accuracy on all four of our models after converting our data to this form, which suggests that there is essentially no benefit to looking directly at raw amplitude data. While it is true that converting to mel-spectrograms takes a little bit of time, especially with our high number of training examples, this pre-processing step can be pre-computed and stored in a file format such as `.npz` for quick access across all models. Additionally, mel-spectrograms are essentially images, as in Figure 1, which provides an human-accessible way to

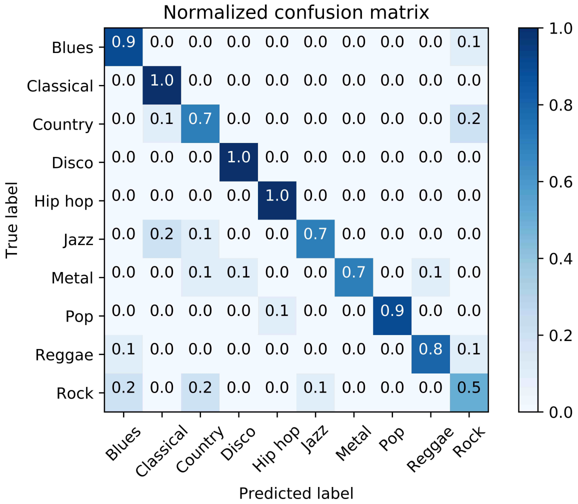


Figure 3: Confusion matrix for CNN predictions.

visualize our data and to think about what our neural networks may be classifying on. In other words, there is essentially no downside to switching to mel-spectrograms.

We also see that all four of our models struggle with over-fitting. We spent the most time trying to mitigate this issue on the CNN. To do so, we introduced three methods. We played around with a combination of batch normalization, dropout layers, and L2 regularization. We found some difficulty in using this, as allowing the model to over-fit actually increased our accuracy. While we could bring the training accuracy and the test accuracy to within .05 of each other, this would result in poor model performance. Thus we accepted our over-fitting issue.

Looking more closely at our confusion matrix, we see that our CNN struggled most with the rock genre. It only managed to correctly classify 50% of rock audio as rock, labeling the others as mainly country or blues. Additionally, it incorrectly classified some country, as well as a small fraction of blues and reggae, as rock music. While it's not all that surprising that rock was a challenging genre – a qualitative inspection of rock mel-spectrograms implies that many rock music excerpts lack the easily visible beats that other genres such as hip-hop and disco possess, while our personal experience with rock music vis a vis the other genres tells us that rock is also missing distinctive traits such as high-register vocals (pop) or easily audible piano (classical or jazz). Additionally, rock is a genre that both encapsulates many different styles (light rock, hard rock, progressive rock, indie rock, new wave, etc.) and heavily influences many other derivative genres. However, we were surprised that rock and country were so easily confused, as opposed to rock and metal, which would seem to rely on more similar instrumentation and tempo.

Additionally, we note that correct classification of jazz was less accurate than most other categories. Our algorithm falsely classified some jazz music as classical, although never did the reverse of this. We hypothesize that the more piano-heavy jazz tracks may have been too close to classical music in terms of both tempo and instrumentation, and may have been missing the saxophone or trumpet sounds and timbres associated with many other samples of jazz audio.

6 Conclusion and Future Work

Across all models, using frequency based mel-spectrograms produced higher accuracy results. Whereas amplitude only provides information on intensity, or how “loud" a sound is, the frequency distribution over time provides information on the content of the sound. Additionally, mel-spectrograms are visual, and CNNs work better with pictures. The CNN performed the best, as we expected. It took the longest time to train as well, but the increase in accuracy justifies the extra computation cost. However, we were surprised to see the similarity in accuracy between the KNN, SVM, and feed-forward neural network.

In the future, we hope to experiment with other types of deep learning methods, given they performed the best. Given that this is time series data, some sort of RNN model may work well (GRU, LSTM, for example). We are also curious about generative aspects of this project, including some sort of genre conversion (in the same vein as generative adversarial networks which repaint photos in the style of Van Gogh, but for specifically for music). Additionally, we suspect that we may have opportunities for transfer learning, for example in classifying music by artist or by decade.

Contributions

Our group has collaborated closely on the majority of this project. We worked together to find a project idea, lay out the direction of the project, and determine matters of data processing as well as implementation details. The writing of the project proposal, milestone report, and this final report have all been a team effort – we found that we never felt the need to explicitly delegate tasks. For the poster Arianna drove most of the design process, but all three group members helped to provide the content.

However, when it came to coding, we realized that our team worked better when we could mostly code by ourselves. Derek was instrumental in setting up the environments and compute resources, and together with Eli took on most of the implementation of the neural networks. Arianna implemented the non-deep learning models, as well as spearheaded the data processing and feature selection aspects of our project.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Hareesh Bahuleyan. Music genre classification using machine learning techniques. *CoRR*, abs/1804.01149, 2018.

[3] François Chollet et al. Keras. `https://keras.io`, 2015.

[4] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. *CoRR*, abs/1802.09697, 2018.

[5] Brian McFee, Matt McVicar, Stefan Balke, Carl Thomé, Vincent Lostanlen, Colin Raffel, Dana Lee, Oriol Nieto, Eric Battenberg, Dan Ellis, Ryuichi Yamamoto, Josh Moore, WZY, Rachel Bittner, Keunwoo Choi, Pius Friesch, Fabian-Robert Stöter, Matt Vollrath, Siddhartha Kumar, nehz, Simon Waloschek, Seth, Rimvydas Naktinis, Douglas Repetto, Curtis "Fjord" Hawthorne, CJ Carr, João Felipe Santos, JackieWu, Erik, and Adrian Holovaty. librosa/librosa: 0.6.2, August 2018.

[6] Y. Panagakis, C. Kotropoulos, and G. R. Arce. Music genre classification via sparse representations of auditory temporal modulations. In *2009 17th European Signal Processing Conference*, pages 1–5, Aug 2009.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002.

[9] Changsheng Xu, N. C. Maddage, Xi Shao, Fang Cao, and Qi Tian. Musical genre classification using support vector machines. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, volume 5, pages V–429, April 2003.