CSE 546 Final Paper
Predicting Musical Genre from Album Cover Art

Nathan Lee & Robert Baraldi
Department of Applied Mathematics
University of Washington
Seattle, WA 98195
nlee@uw.edu & rbaraldi@uw.edu

Abstract

In this paper, we have attempt to develop an algorithm that will classify an album's musical genre/style based on the cover art. We pursue a color analysis where we create color histograms and then compare genres against each other. We also implement several neural networks, both completely trained on our dataset and pretrained on Imagenet data and implemented via transfer learning. Overall, our results were mixed, as some genres are classified accurately and others are not.

1 Introduction

Sonic media is interesting in that the product is sound, yet the physical package marketed to the customer has to be visual. This visual style of the album cover art usually reflects the overall tone of the album or perhaps portrays a visual representation of the album's topic. Hence, understanding the style of an album cover is important since it gives one an idea of the type of music as well as how potentially interesting that music will be. Certain musical genres are known to be associated with a particular cover art style. For example, ambient albums tend to have more abstract, computer-generated designs, and metal albums tend to use darker colors with unusual fonts for the album title. These styles are recognizable to most human observers, and we want to test whether or not machines can also detect them.

In this project, we attempt to classify the genre of an album based on its cover art. While an important visual component, it is difficult to enunciate how style varies from image to image as well as quantify said difference. Distinct visual styles are readily apparent in various media, and entire fields are built on distinguishing between different periods of art. Hence, it would be interesting to see if we can train models to identify major components of certain styles of cover art and use these models to identify the corresponding musical genre.

We define 10 different genres of music and classify each with a digit between 0 and 9. These are: ambient (0), dubstep (1), folk (2), hiphop/rap (3), jazz (4), metal (5), pop (6), punk (7), rock (8), and soul (9). The genres listed represent the 10 most common genres on Bandcamp. Next, we scraped aforementioned media platform for albums/artists using similar methodology as in [1]. In this phase, the training dataset has been selected to have 8800 album covers and the testing dataset has 1000 randomly selected album covers. We note here that the number of albums per genre in the training set vary between 990 and 970. The testing set has an equal number of album covers per genre. A sample of these is displayed in Figure 1.

2 Preprocessing

In this project we use `opencv` and `pytorch` data loading for accurate downsizing of our pixel space into a more compact grid of a smaller size (32x32 unless otherwise specified), using the built-in

(a) Ambient (b) Metal (c) Folk (d) Pop

(e) Punk (f) Dubstep (g) Rap (h) Jazz

(i) Rock (j) Soul

Figure 1: Image dataset

opencv interpolation functions. This is necessary as we found that using the full dataset/image size quickly drains the memory requirements of our computers. Each picture comes in 350x350 pixel rgb images; in all we have 5 Gb of data when using the full colorized pictures. In this project, we do not use grayscale as we can observe that color is a big indication of some genres, such as metal. We thus scale the images down to 32x32 rgb images with two approaches: the first is simply randomly sampling a 32x32 grid from the full 350x350 grid, and the second is smoothing the picture as we shrink the pixels size down to 32x32 from 350x350 (i.e. interpolation). We found that randomly downsampling the pixel space produces very poor results for neural net implementation and about the same results for the other methods (yet still very poor).

The rest of the paper is structured as follows: first, we perform a color analysis to see if certain genres can indeed be distinct. Next, we perform a variety of basic machine learning algorithms to see if these basic techniques can perform well enough for our goal. This is meant to primarily further our intuition that a larger neural network implementation will need to be used. Finally, we examine several neural network architectures, both pretrained and not, and discuss their success.

3 Color Analysis

In this section, we use `opencv`'s cubic interpolation to scale down all the images to the 32x32 size. Then we calculate each image's rgb histogram, with 8 bins in each channel. These three-dimensional histograms are then normalized and flattened. Next, we use these to find the similarity scores between a set of 50 randomly selected albums in each genre with a different set of 50 randomly selected albums in each genre. This requires computing $50^2 \cdot 55$ similarity scores. We display the results in a 10x10 symmetric matrix, where each element $(i.j)$ represents the average similarity score of an album in genre i to an album in genre j (Figure 2). When comparing the same genre, none of the same albums are used. A larger score indicates that the color histograms are more similar to each other. In comparing these similarity scores, we use employ four different distance metrics: correlation, χ^2, intersection, and hellinger metrics for each album. Here we only report intersection test scores as

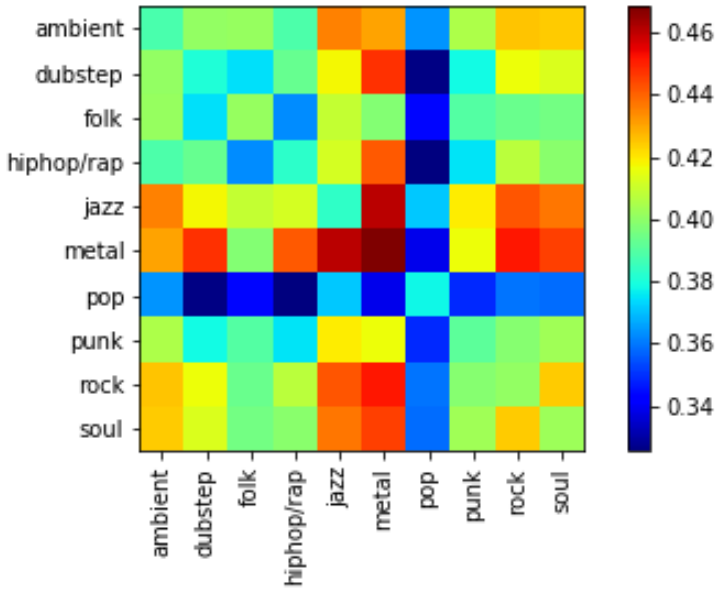

Figure 2: Sample heat map of averaged similarity scores between albums. Note that red (higher) values mean more related, while blue (lower) values are less related.

they are the most telling. From this, we can see that metal albums selected were more similar to itself, jazz, and rock, while pop albums selected were not that similar to anything (including itself), indicating that it is not necessarily true that color profile is consistent across genre. This intuitively makes sense, as pop isn't a precise genre descriptor, with more variability from album to album.

For each genre i, we also order and rank the average similarity scores of the ten different genres. In Table 1 we report this rank of how similar each genre is to other albums in the same genre. In other words, a 1 means that a genre was the most similar to itself relative to the other genres, where a 10 would mean that it was the least similar. This shows that metal matched with itself the best, while ambient actually matched with itself almost last. Repeating this test many times indicates that some genres like metal, pop, and folk are slightly consistent, but most are not. The average rank for all genres except metal was between 5 and 6, which is an indication of huge color variability in most album art. Looking at a random sample of images in the genre and taking the average of those images is presented in Figure 3. We can see some differences, but one can tell that running this many times will simply degenerate into a brownish mess. Thus, while color plays a significant role in some genres, it is not necessarily true in others. Indeed, running an SVM on the histogram data essentially optimizes to putting every test case into one (or a few categories), while running a random forest on the histogram data amounts to guessing. From this, we can gather that color is a useful indicator for some genres, but likely some higher level features exist that are not quite being captured by this analysis.

4 Naive ML Technique Implementations

We used the basic pre-processing lessons learned above and used this data on a variety of `scikit-learn` algorithms to test what style of machine learning we will have to use for this problem. We believe that some sort of deep neural nets and/or kernel methods will be required, but wanted to explore our options with some naive implementations before we fully committed to one particular direction. As expected, Ridge and LASSO did not fair very well in genre classification, so we changed the number of pixels we downsampled as well as our $\lambda = 10^{-4}$ value for both algorithms, and found that it was possible to overfit the data as a function of the image size approximation. For instance, we found that when approximating each image by a 32x32 grid, the training data comes back with an accuracy of 43% and the testing accuracy is 12%, whereas if you used a 70x70 grid, we get that Ridge gets 89% accuracy with the training data and a 10% accuracy with the testing

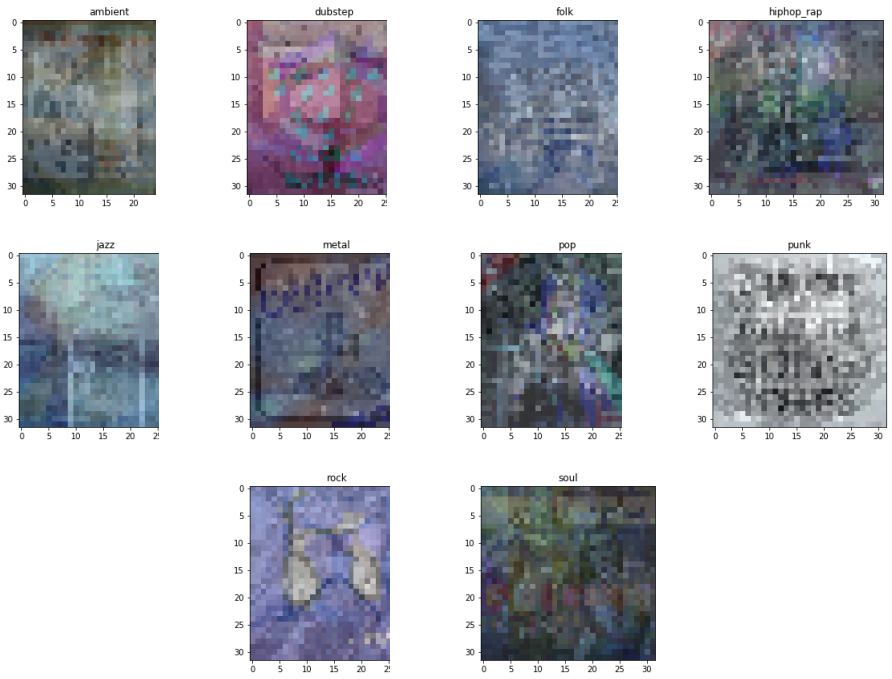

Figure 3: Sampled genre averages

Table 1: Ranked Distances in randomly chosen album covers.

Genre	Rank to itself
ambient	9
dubstep	7
folk	2
hiphop/rap	7
jazz	9
metal	1
pop	1
punk	6
rock	7
soul	7

data. With a 100x100 grid and using `opencv`, Ridge gives us a 99% training accuracy yet a 9.3% test accuracy. For a grid of 35x35, we found that LASSO classified 11.5% of the training data correctly and 12.5% of the testing data correctly. Hence, increasing pixel space only really means one overtrains the model.

With regards to the other Python-based methods found in `scikit-learn`, we have our results in Table 2. From this, we can see that SVM using a RGB kernel and random forests (max depth of 2) do marginally the best, with a testing accuracy of about 15% for both types of preprocessing. Most of the other methods tend to overfit the data, such as decision trees (which were discussed in class to be prone to overfitting). Again, this concludes that we will probably need more involved methods. In this, we've established that traditional ML routes are likely not the answer, and decided to pursue other options.

5 Neural Net Implementation

Having seen all of the previous classification techniques fail, we turn to more traditional image classification in the form of convolutional neural networks. In this, we consider two approaches:

Table 2: Training and Test accuracy of different methods in `scikit-learn` with random downsampling and interpolation/resizing.

Method	Random Training	Random Testing	Resize Training	Resize Testing
Ridge	33%	9%	33.6%	10.5%
LASSO	10.2%	9.2%	10.6%	9.3%
SVM	19.2%	14.8%	18.3%	14.8%
Adaboost	16.5%	13.2%	16.8%	12.7%
5-nearest neighbors	34.2%	11.3%	32.9%	10.2%
Random Forest (max depth= 2)	14.3%	14.8%	14.9%	15.5%
Decision Tree (max depth=2)	13.7%	12.7%	13.8%	12.9%
Linear Discriminant Analysis	33.9%	12.9%	34.9%	11.3%
Quadratic Discriminant Analysis	70.2 %	10.3%	46.1%	10.4 %
Gaussian Naive-Bayes	14.7%	13.2%	14.5%	12.2%

1) training models on our training data and then using them to predict the test data, and also 2) implementing models that have previously done well in classifying CIFAR-10 or Imagenet data and retraining the last layer on our data. This will allow us to determine if our dataset is rich enough to fully train a neural network as well as use some of the richness of the imagenet dataset in our favor. This latter approach could also suggest an inherent difference in our data from more traditional deep learning datasets, such as CIFAR-10.

Our code was written in `pytorch`, and this allows us to implement pre-trained models relatively easily. While playing around with parameters shifted some of the results around, we saw no significant increase in classification accuracy overall as hyperparameters were tuned. Again, images were cropped and interpolated to 32x32x3 unless otherwise specified. Our batch size is 10 and our learning rate is 0.01. We optimize using SGD and a momentum factor of 0.9. For all the networks used, we implement the softmax classifier with cross-entropy loss:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

where L_i is the loss for example i in the training mini-batch, f is the score for a particular class calculated by the network, j is one of the possible 10 classes, and y_i is the correct class for example i. This allows the network to classify genre by constantly trying to maximize the score of the current genre of its training examples relative to other genres during training.

5.1 Fully Trained Performance

The first model we used was a modification of one of the examples on pytorch, which acts as a stand-in for a generic baseline model that will serve as a comparison for other, more complex models tested. It can be written as (from inside to outside)

$$\begin{aligned}
x &= \text{Conv2d}\left(x^{input}, W_{1c}\right) + b_{1c} \\
x &= \text{MaxPool}\left(\text{relu}\left(x\right)\right) \\
x &= \text{Conv2d}(x, W_{2c}) + b_{2c} \\
x &= \text{MaxPool}(\text{relu}(x)) \\
x &= \text{relu}\left(W_1 \text{vec}(x) + b_1\right) \\
x &= \text{relu}\left(W_2 \text{vec}(x) + b_2\right) \\
x^{out} &= W_3 \text{vec}(x) + b_3
\end{aligned}$$

ie a convolution with $M = 50, p = 5$, another convolution with $M = 10, p = 3$, and a maxpool with $N = 2$. The linear maps are from $W_1 : \mathbb{R}^{32\times(3\cdot2)^2} \to \mathbb{R}^{120}$, $W_2 : \mathbb{R}^{120} \to \mathbb{R}^{84}$, $W_3 : \mathbb{R}^{84} \to \mathbb{R}^{10}$. This algorithm is then trained on the full 8800 training dataset. While this algorithm does not do that well overall (see Table 3), it does surprisingly well relative to other models we tried. The resulting model had an overall error of 17.6%.

The "famous" model we fully trained was Alexnet [6]. The best model on the trained data had a total error of 15%. We chose it specifically because it performed well in the transfer learning portion

Table 3: Fully Trained Models and Errors per class.

Genre	Custom % Correct	Alexnet % Correct
ambient	24	49
dubstep	17	0
folk	24	40
hiphop/rap	6	3
jazz	10	0
metal	44	41
pop	14	0
punk	21	0
rock	3	1
soul	13	0

of testing (discussed below), and exhibited the characteristics of getting certain classes correct (as well as being computationally tractable). We can see the same here, as some of the classes are much more accurate than others. However, we note that the algorithm is more than likely settling at a local minimum where it simply guesses between 3 classes. Hence, it does only about 10% better than guessing.

5.2 Pretrained Models

Next, implement transfer learning techniques on models that were initially trained on ImageNet data and were available in `pytorch`. In it, we actually have to use the cropping size of 224x224 (although some can run with 32x32 by modifying a few more lines; these were the Resnet models). ResNets use residual blocks to ensure that up-stream gradients are propagated to lower network layers, thereby aiding in optimization convergence[2]. Here, we replace the final fully connected layer with a new one to calculate the score for each genre in our dataset instead of a score for Imagenet classification. This allows us to test whether or not a feature representation from Imagenet is a valuable starting point for genre classification. All the other parameters are kept the same as well. Our goal in this is to enunciate the difference between the structure of our dataset and the structure of other, larger datasets, such as Imagenet.

Here, we test Alexnet, Resnet18, Resnet34, and Resnet152. Note that we chose these based on modification to a 32x32 rgb picture (all the involved modification on the last layer) as well as accuracy on ImageNet testing. For Alexnet, we stripped the last layer and trained the model on just this last part, hoping to preserve the architecture established by a much larger database. The results are shown in Table 4. The best model for all architectures selected had an accuracy of 18% on the testing data. What was interesting in particular about Alexnet in constrast to the Resnets was that the training accuracy was usually about 49% as opposed to 20-25% on training data. Resnet34 had an accuracy of 19% overall, while Resnet18 had an overall accuracy of 17.5%. Resnet 152 (which had the highest accuracy on Imagenet) only had an overall accuracy of 16%.

Between these, it is interesting to see which genres are consistently higher. For instance, metal is consistently at least 30%, where as rock and jazz are typically low. In this sense, it matches the color data to a degree. However, note that in some cases the data is no better than guessing, which we will talk more about in the discussion. Surprisingly, Resnet152 did worse than the other pretrained models; we believe that this could be due to the deeper architecture that is particularly attuned to Imagenet.

6 Discussion

In this project, we have performed a histogram color analysis, several out of the box ML techniques, and convolutional neural nets. None of these performed in a satisfactory way, though our analyses suggest some interesting directions that could be pursued for better accuracy, as well as some fundamental differences our dataset has from other more common learning datasets. The color histograms showed that the color content alone of the album art was generally not useful for most genres, and many of the standard ML techniques from `sk-learn` more or less resulted in classification probabilities equal to guessing. Some overfit the data while others did not. While image classification

Table 4: Pre-Trained Models and Errors per class.

Genre	Alexnet % Correct	Resnet34 % Correct	Resnet18 % Correct	Resnet152 % Correct
ambient	19	19	22	19
dubstep	17	15	13	13
folk	12	24	19	11
hiphop/rap	19	11	6	7
jazz	14	11	7	11
metal	35	38	47	39
pop	11	16	9	13
punk	15	19	15	16
rock	12	7	6	9
soul	17	13	12	10

with neural nets had some success in some genres, the overall result was poor. Averaging around 18% correctness is only slightly better than guessing for the 10 named genres.

The two main antagonists in this case are related, and seem to be the type of object we are trying to classify and the labeling of the data itself. The first issue is that we are trying to classify *style* of an image, which is more subtle and complex than its physical structure or shape. It's interesting to note that the models pre-trained to perform very well (>90%) on CIFAR-10 perform poorly on our dataset, suggesting some fundamental difference in what these two image datasets represent. In more standard learning problems like CIFAR-10 or MNIST (or even Imagenet) image classification, the problem is to extract some type of visible pattern or shape to detect an object, which has some type of consistent structure in the different images. In this project however, there isn't always a consistent structure that corresponds to each genre. It would not be enough to classify simply what the image contains in this instance, as we are actually trying to relate how the objects in an image fit together in an aesthetic that is representative of a musical genre. Karayev et al[4] attempted something similar with paintings, but even those are an easier task in that certain styles are known to have certain color palettes and brush strokes. Indeed, even the humans we tested this on did not perform well.

The other, possibly more serious, issue is that our dataset is not robust enough. Karayev et al[4] had hundreds of thousands of paintings to choose from; we only have 9800 total to train our networks. If our dataset was similar to some of the others we have seen throughout the class, then one would believe the pretrained models would have classified the genres better. This, however, could also be related to the fact that we are not classifying images in the conventional sense. Another issue we believe is confounding our results is that genre designation is naturally fluid. For instance, of these 10 genres, several of them can be considered subgenres of another (punk and metal are subgenres of rock). We think that having more distinct genres to classify or even allowing albums to be classified as more than one genre will increase the accuracy of our results. Notice that neural networks performed the best (close to 50%) at classifying metal, which is a very specific genre. Whereas, they performed poorly on rock, which is an ambiguous, very broad designation that could include many different styles of music, ranging from indie rock to heavy metal. Instead of obtaining data that labels each album with a single genre, one could label each album with multiple sub-genre classifications. For example, rather than labeling a certain album rock, it might be post-punk and alternative. A modification to our analysis to obtain better prediction could involve changing our algorithms to output the percentages of the top three possible genre designations. This would actually be better at matching the bandcamp descriptions, as albums are typically described by several genres or keywords. Other features to be extracted from the images could also be considered. For example, the image gradients could be extracted in a similar way as the color histograms we extracted.

In conclusion, we chose a very difficult problem. It is difficult for people to classify genres in this fashion (only one genre per album), and our algorithms do not perform much better. Certain genres with more defined, consistent characteristics typically yield better results. It's possible that the genres labels are too general and so include too many diverse artists, yielding varied cover art. Future work could include allowing multiple labels, collecting better-labeled data, and considering other possible features to extract from the images.

References

[1] Yanir Serouss. Learning about deep learning through album cover classification. 2015.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[3] Nitin Viswanathan. Artist identification with convolutional neural networks. 2017.

[4] Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller. Recognizing image style. In *Proceedings of the British Machine Vision Conference*. BMVA Press, arXive preprint: arXiv:1311.3715v3, 2014.

[5] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.