Collin Allen
Justin Berry
Isabella Garza
Aravindakumar Sundaramraj
Note: Code used is attached as a pdf at the bottom of the report

IMDB Genre Classification

ELEC 301 Final Project

Introduction

Movie posters are designed to convey information to consumers both concisely and quickly. A person simply walking down the street should be able to glance at a movie poster and intuitively understand what this film is about, and at the very least which genre it belongs to. These posters themselves follow a structure, with a title and scene from the movie meant to convey the general premise, and movies from the same genre tend to share many characteristics in common. This common structure and shared qualities led us to explore how machine learning tactics can be used to categorize unlabeled movie posters into a simple spread of categories.

Methods

When brainstorming how to approach this problem, we decided to break it down into a simple workflow structure: preprocessing, feature extraction, model choice, and model parameter optimization,

Preprocessing/Grayscale Image

When importing the movie posters into the code, we divided them into a 64x64 image in order to speed up the connected neural net training later. However, to explore the effect of resolution on performance we tried modifying the resolution of the images in the preprocessing step. We attempted both 128x128 and 224x224 images yet saw inconsistent results in performance. At times the performance seemed to increase, though it was difficult to discern whether this was due to an actual increase in performance as a result of increasing the resolution or simply randomness in the initialization of the filters in the neural nets. In contrast, increasing the resolution noticeably led to an increase in computation time when training neural nets. This is an intuitive relationship considering how significantly the amount of information processing in training a neural net scales with resolution. Since increasing the resolution led more to an increase in computation time with little to no gain in performance, we stuck to 64x64 pixel images in most of our analysis.

Feature Extraction

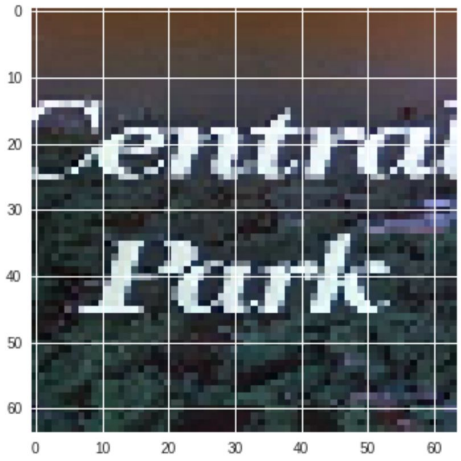
Due to our limited capabilities with neural nets, we decided to rely heavily on extracting features from the images and metadata as a preprocessing step to then feed these extracted features into neural networks for categorization. Our first step in this project was to look through the movie posters and consider subjectively what aspects of the posters themselves seem to

indicate genre. The characteristics that we deemed most significant and which we actually attempted to use in our models are colors, edges, and faces. On top of our analysis of the posters themselves, we also considered what values in the metadata would be most strongly correlated with genre. However, over the course of our design phase, we explored all of the values given to us in the metadata.

In this section, we will dig deeper into each of these features and discuss our results.

Edge Detection

When we first began looking at the movie posters ourselves, we felt that the general level of activity seemed indicative of genre. Dramas tended to be simpler, with less distinct objects, whereas action movies tended to be filled with complicated designs and shapes. In order to explore this quality we developed an edge detection model to feed into a convolutional neural net. For preprocessing, we first turned the posters into size 64x64, and then used a 3x3 kernel to smooth the image. We then fed the smooth image into the Canny edge detection function from the cv2 library. The Canny function runs a 5x5 Gaussian filter over the image to smooth it further, followed by a Sobel kernel to find the gradient intensity for each pixel. The algorithm then finds local maximum values based on the gradient at each pixel, and uses Hysteresis thresholding to find the strongest edges. Hysteresis thresholding involves deciding maximum and minimum intensity gradient values, which are used to select the strongest edges. Pixels are included in an edge if they are above the maximum value, or if they are within an edge that has pixels above that maximum value. Pixels below the minimum are discarded, as well as pixels that are in an edge only between the minimum and maximum. This algorithm runs very well, and did a good job of extracting edges from our training and test images. The Canny function returns a 2D image containing the strongest edges for each of the posters, which is what we fed into a neural network for categorization. An example of the original poster for the movie Central Park and its corresponding edge detected version are below.



Original poster

Edge Detected Preprocessed image

Face Detection

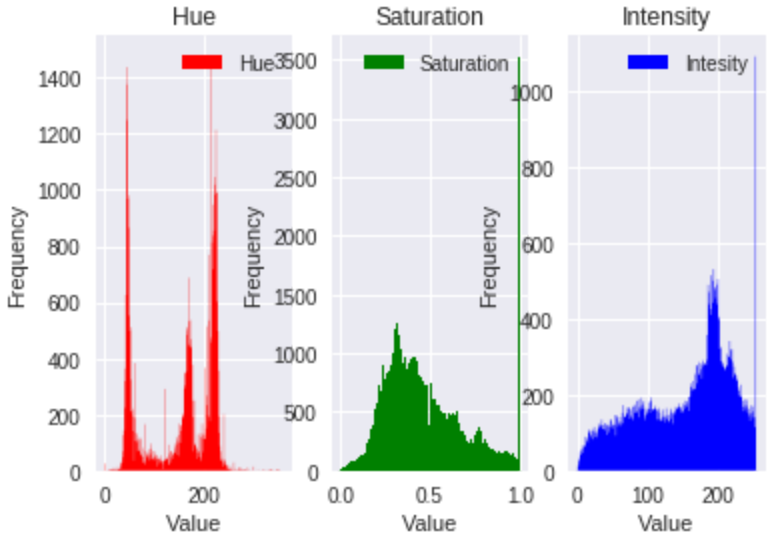
Another hypothesis we had was that movie genres focused on people, such as comedy or drama, would feature more faces on the poster. In order to find the number of faces on a movie poster, we used the Haar Feature-based cascade classifier found in cv2. This classifier was already trained to detect faces (found in the haarcascade_frontalface_default.xml file). The gray scale, 64x64 movie posters were fed into this algorithm to give us the number of detected faces in a movie poster. Initially when we used `face_cascade.detectMultiScale(gray, 1.3, 5)`, the algorithm was conservative in its estimates; some movie posters with faces were spitting out zeros. To fix this, we tried `face_cascade.detectMultiScale(gray, 1.3, 3)` to make the algorithm more lenient. The number of faces was fed into a CNN that used 1D convolution. However, when inspecting the output predictions, only dramas and biographies were being predicted. Since we didn't pursue using faces since we didn't want to use a feature that was incapable of predicting action or romance movies.

Color histogram

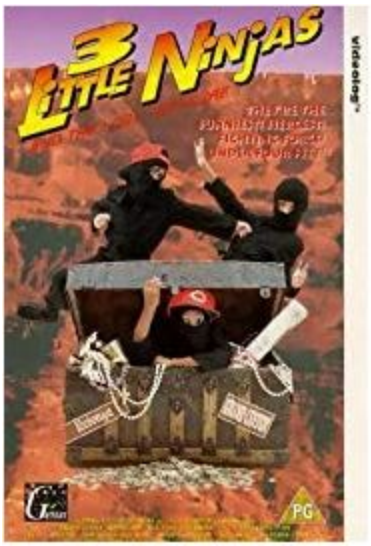
One of the most clear aspects of the poster that correlated to genre was the spread of colors across the image. Comedies and romances tended to be be filled with many different bright colors, whereas more serious films such as dramas tended to be darker and grittier. Our first step to incorporate color information into our model was to change from reading the images in as grayscale to reading them in with their RGB values. This was a simple change in our image reading function of passing in cv2.IMREAD_COLOR instead of cv2.IMREAD_GRAYSCALE. The color images did have a different structure which we had to consider when passing them in as inputs to neural networks, since each image was now a 64x64x3 structure, with the third dimension being the image RGB values. Using the RGB colorspace did lead to our convolutional neural network that was running on the raw images to have an increase in performance.

While switching to RGB values did lead to an increase in performance, we were still not getting the accuracy that we wanted. The field of color spaces and the information they contain in an image is very broad and complicated, but from doing research and exploring the research paper "Automatic Movie Posters Classification into Genres" (Ivasic-Kos) we decided to explore the HSV (hue, saturation, value) colorspace for representing the posters.

From each of the posters, we created a normalized histogram using the HSV color space to mimic the way human vision perceives color.



HSV Histogram for “3 Little Ninjas”

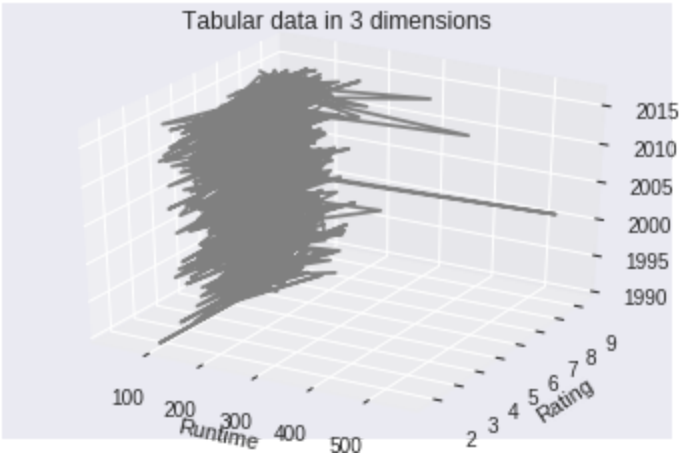


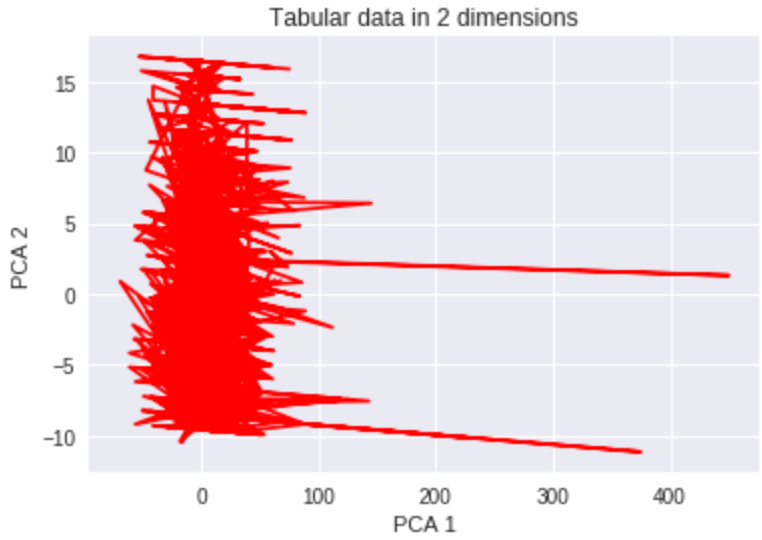
The histograms were then fed into a Neural Net that had 3D Convolution (n=384), Batch Normalization, and Max Pool to extract features from the histogram. Dense layers were used to predict based off of these features while Dropout(r = 0.1) layers were placed to prevent overfitting. The model was trained with 5 epochs and batch size of 5.

Tabular Data and PCA

Once the project was assigned, we began thinking of what information would be valuable in classifying movies. As a person who loves movies, one of the first things that came to mind where the runtime and rating. We thought of horror movies, for example, having lower ratings on average, and action movies being shorter on average. The first approach that we tried was to make a CNN using only one of the tabular data entries (rating, runtime, etc.). We altered the model training code a little to accommodate the tabular values by removing the convolutional

layer. After seeing how useful this data could be, it became crucial to try training on three of the categories: runtime, rating, and year. However, we wanted to also try speeding up the neural net training; PCA seemed to be a good idea since there is probably a correlation between rating, runtime, and year. We took the top 2 PCA components of this vector and fed it into a neural net.





Choosing Neural Nets

During our initial exploration, we had to choose between K Nearest Neighbors or Convolutional Neural Nets. To make this choice, we built the two models using colored histograms as the training data. After submitting both, they scored 41 and 48 percent respectively, making us go forward with Neural Nets for each of the features. Also, we chose the number of epochs such that the accuracy on the last epoch is around 0.5 - 0.6 in order to prevent overfitting.

Combining the models and results

Originally as we explored these different methods of feature extraction models we kept them seperate, and tested on our split training data to get a sense of accuracy. Each of these models alone would get into an accuracy range of 30% to 60%, and in seeking better performance we sought to combine the models in a meaningful way. Running an analysis on the RGB image model, edge detecting model, and tabular data model we discovered 75% of the images were being categorized correctly by at least one of the models. This seemed so promising that we shifted the focus of our efforts into the best way to combine these models.

When given an unlabeled test data point each model would return a 1x4 vector with the probabilities that the movie belonged to each of the four categories. Working with these three models, our first attempt was to take the highest probability across all three. This was based on our intuition that the model that would correctly categorize the image would have the highest prediction value. This intuition was apparently wrong, as the combined model performed worse than the best individual model (the tabular data net which consistently gave around 55% accuracy on our split training data) at around 30-35%.

Our next thought was to primarily use the best model, and only use the predictions from the other two models when the max probability from the best model was only below some threshold. This method produced about the same performance as the max prediction model discussed previously.

With these hardcoded simple models of combining predictions proving to be ineffective, we next moved to incorporate another neural network. We took the three prediction vectors and concatenated them into a 1x12 vector and fed this into a simple neural network (only dense layers and no convolutions). This method sadly only performed at about the same level as the previous models when given unlabeled test data. Our first version of the model likely had this poor performance due to overfitting. The last few epochs had an accuracy on the training data in the upper 90 percentile. We reduced the number of output filters in the densely connected layers to just two or four in consideration to how small the input is to this net, and while this increased the performance the accuracy still never broke 50% Though we had put much work into this method, we eventually abandoned it to do its poor performance.

The prediction combination model that proved to be most effective was to simply take a weighted average of the three prediction models. The weights were skewed heavily to the best performing model of the three, and only led to an increase in about two percent from the accuracy of that best model. Since the best model was our tabular data network, it achieved around 58% at best. Each of the individual models did not perform very well, so we decided that a “weighted average” of each of the predictions would produce better predictions. First, we focused on utilizing the RGB images, color histogram, and tabular data/PCA predictions in order to give us a wide variety of information. Next, we trained the individual models on 2500 training points and got 3 different predictions. With the remaining 593 training points, we gave weights based on which prediction had the highest probability. These weights were then normalized to give a probability distribution for how often to use each prediction. The probability distribution turned out to be [0.386, 0.246, 0.3676] for tabular data, color histogram, and gray images respectively. Finally, the individual models were trained on the full 3094 data points and made predictions on the actual test data. However even after combining the models using a stochastic approach, the accuracy was only 39%, which is worse than our submission with the color

histogram model of 48%.Unfortunately this success was only achieved locally on our split training data, and we never managed a submission that achieved such levels of accuracy.

Discussion

None of the models that we submitted performed very well on the unlabeled test data. This was often surprising for us, as we would get accuracy percentages in the 55-60% range when testing locally on our split test data, and then submit our solutions and get accuracies only in the 20-30% range. We are unsure of why this discrepancy was occurring, but exploring this is a task we would want to pursue deeper if given more time. One such method that we were unable to implement was stacking. A type of ensemble learning, stacking would take the predictions of each model on the training data, and build a logistic regression model with the predictions as features. It is a commonly used technique to combine different models that we would expect to work well with our problem.

We attempted many different models and methods for prediction (many different feature extractions, neural networks, K-NN, etc.) and then tried to combine them together through our own original ideas. While there was much value in such an exercise and attempting new ideas, we might have performed better if we had done more extensive research into other methods for similar tasks that have already been demonstrated to perform well.

If given more time, we would want to explore a better way to combine our models during the training phase itself, rather than combining our models after forming predictions. While this was an interesting idea that we worked hard on and learned a lot from, it did not perform very well. While we think a combined model using the different features that we extracted might be a better method for classifying accurately. Ultimately, we learned some very interesting machine learning methods and gained valuable experience in applying our ideas on a real dataset.

Works Cited

Ivasic-Kos, Marina, and Ivo Ipsic. “Automatic Movie Posters Classification into Genres.” *Researchgate.net*, Advances in Intelligent Systems and Computing, Jan. 20125, www.researchgate.net/publication/282196711_Automatic_Movie_Posters_Classification_into_Genres.