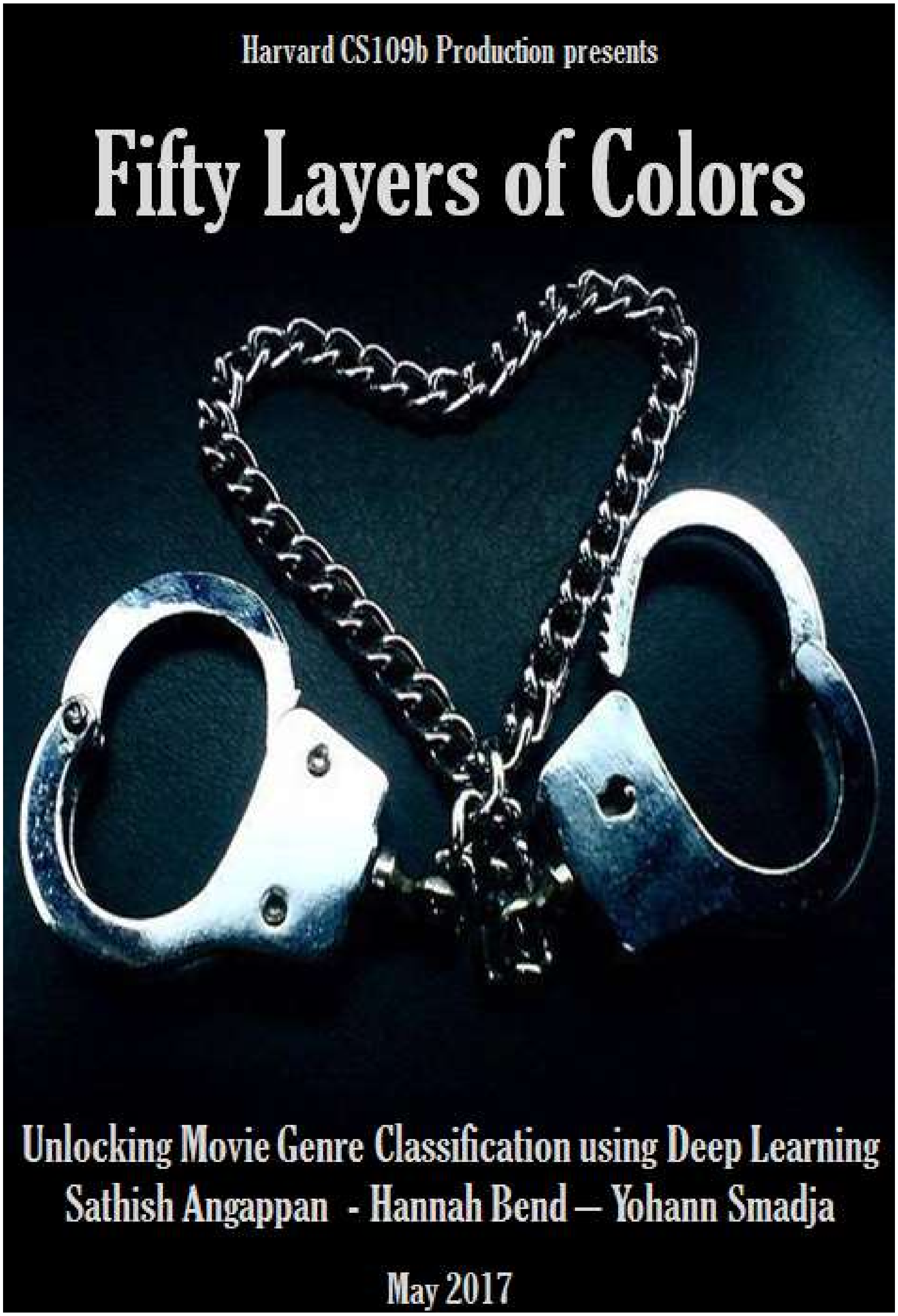
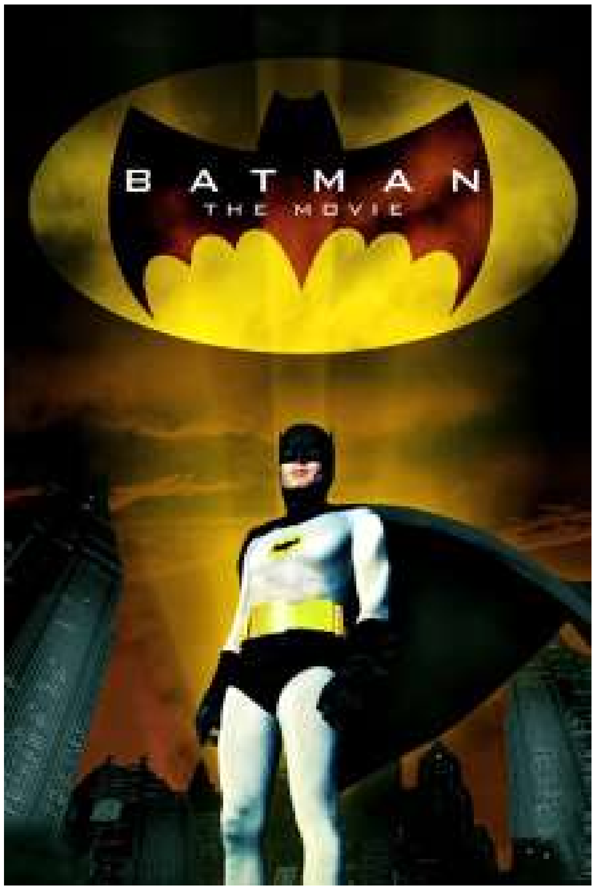
CS 109b: Final Project, Spring 2017
Sathish Angappan, Hannah Bend, Yohann Smadja


Harvard CS109b Production presents
Fifty Layers of Colors
Unlocking Movie Genre Classification using Deep Learning
Sathish Angappan - Hannah Bend – Yohann Smadja
May 2017

A Client-grade Product

Stanley is an online art dealer who recently, being a bit of a cinephile, has expanded his collections to include vintage movie posters, outsourcing much of the procurement to his many distributors. Thus, far, the expansion has gone well, having amassed a few hundred posters with more coming in daily and demand meeting the supply. The only problem is that his website is organized such that clients can browse posters by their movie genre, and oftentimes he only receives a jpeg file without any corresponding metadata. Rather than pay an intern to (subjectively) classify each image into a genre as it's received, Stanley asked us to build a model that will identify genre based on the image and auto-classify it for him. The only catch is, given the scarcity of vintage poster images (which is partly why his new business is so lucrative), would a model trained on movie posters from the past few decades work for vintage posters, given the stylistic differences and the evolution of computer graphics? What types of feature extraction would best help our model "see" that Adam West's Batman oozes just as much action as Christian Bale's?

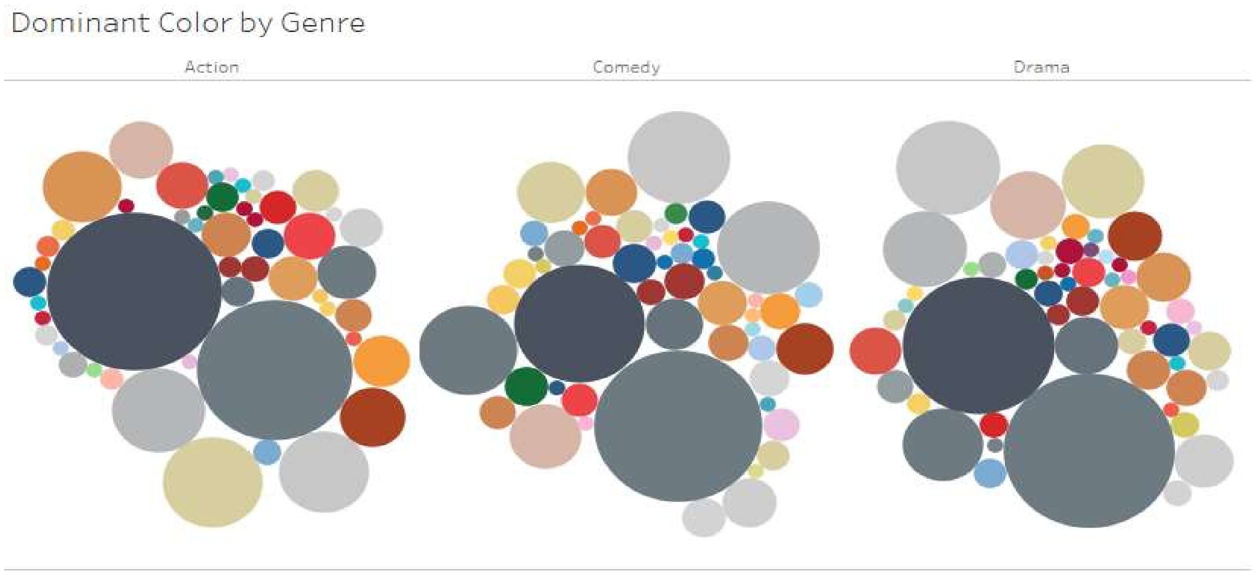


BATMAN BEGINS

Our first step was to identify the data available to us. We evaluated the APIs for The Movie Database (TMDb) and the Internet Movie Database (IMDb) and found TMDb's to be simpler, in addition to having the necessary image files. As we began our data exploration, two challenges quickly surfaced, both pertaining to imbalance. First, TMDb classifies movies by several genres simultaneously, though often with the most prominent listed first. Secondly, the emergence of several niche genres in recent years introduces a fair amount of sparsity in some categories.

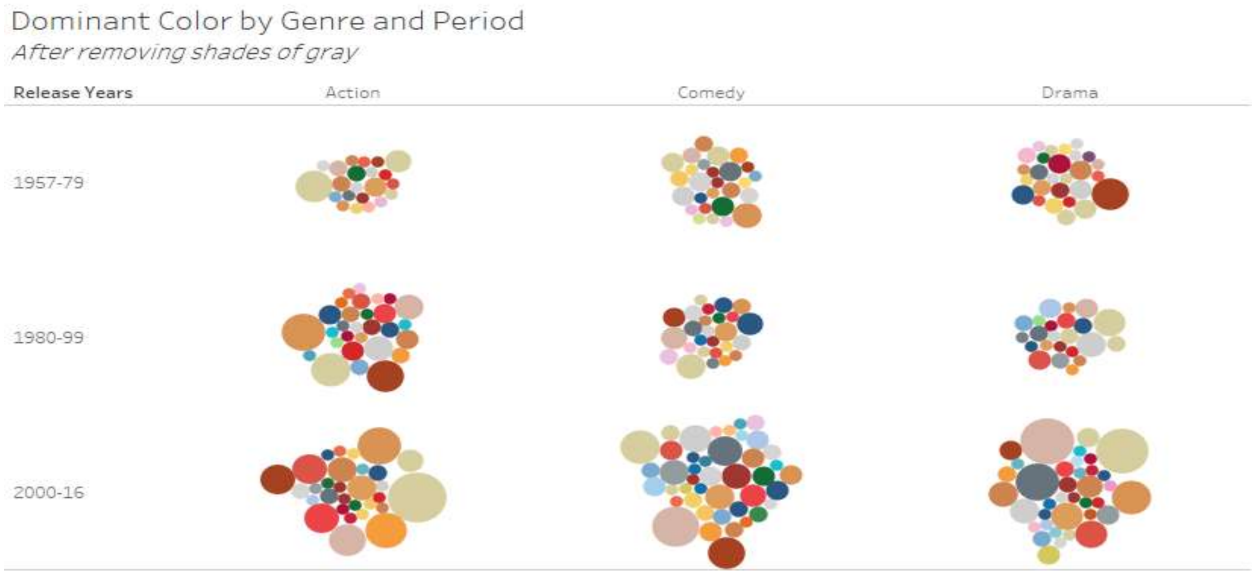
Genres and Colors

To get started, we limited our data extraction to three genres: action, comedy and drama, selecting movies whose first associated genre was one of the three. This limited extract will help us visualize the data while removing some noise like the niche genres. Taking a random sample of 500 posters per genre, we used the ColorThief library to identify the dominant color in each image and plotted that by genre to get an initial insight into any patterns.



What's immediately apparent is the prevalence of shades of gray (not the movie) across all three genres. While logical, this isn't exactly useful when considering whether certain colors are associated with a given genre. Further, this doesn't give any sense of a genre's stylistic change over time, so we pulled out the top five colors and grouped by periods.

Here we start to see some subtle patterns, like the emergence of blues within comedy that don't appear in action, but nothing leads us to believe individual colors will be a strong predictor between genres. On the upside, there is continuity across periods, which bolsters our hope that a model trained primarily on modern posters will be able to predict on vintage posters.



First Attempts at Classifying

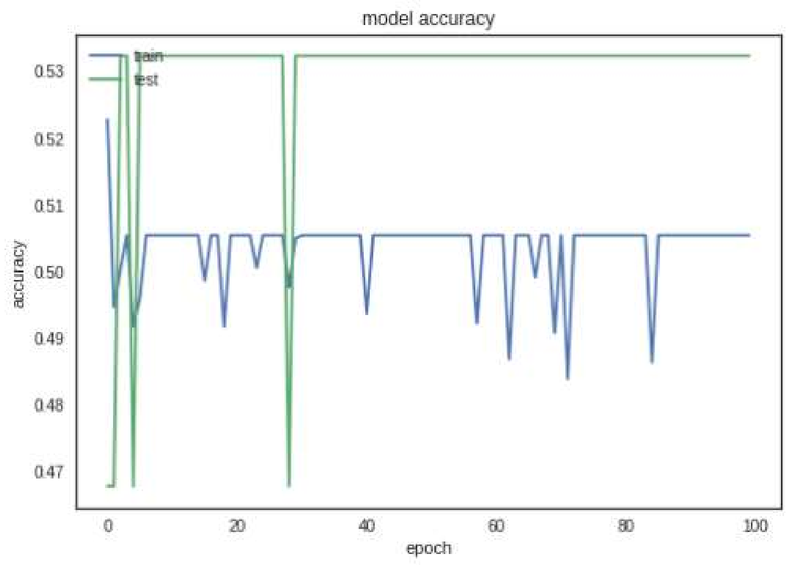
With this insight, we shifted our focus back to the data to start modeling. We decided to stay with our three genres as our focus was on predicting accurately from one period to another, not just predicting genre with precision. Since, at the time, we were still ironing out our cloud instance, we limited our next extract to 300 movies from the year 2016. In addition to the poster images, we also pulled some additional metadata around the poster's movie in case it was useful: title, main genre, popularity, release date and year, average vote and the IMDB's id.

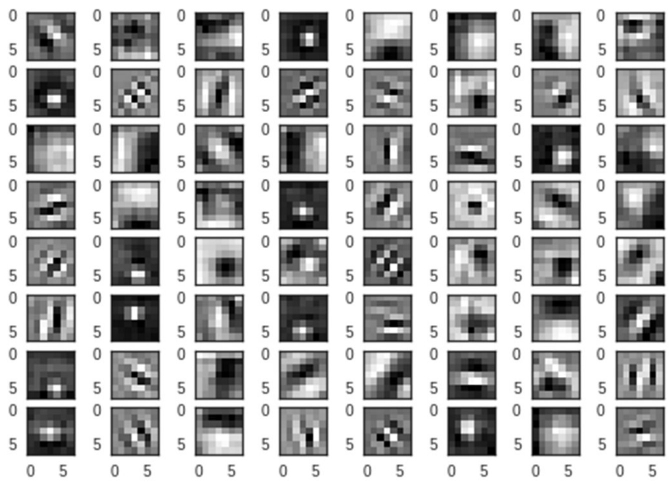
Once we had downloaded the data and images, we tackled the necessary data manipulations, such as resizing the images to a standard dimension (750X500) and vectorizing them. For our initial model attempts, we also converted the images to greyscale as some of the approaches we wanted to try only work on 2D data.

Before getting into machine learning, we tried our hand at a few of the more traditional models. We started with PCA to reduce our dimensions as our vector arrays were 375,000 wide; ultimately, we found that 20 principal components ~~were able to~~could account for 65% of the variance, which we were happy with. From there, we split our data in training and testing sets and began fitting models using our 20 principal components as predictors. We chose K-Nearest Neighbors, Logistic Regression and Random Forests as these are all standard classification models. Unfortunately, none of them performed very well. Our best accuracy was 37% with a logistic regression model, while KNN and Random Forests came in at 33% and 34% respectively. Now, while some degree of inaccuracy can be blamed on our small sample size and gray scaling, it would likely not be enough to make up for such a poor start. Thus, it was easy to conclude that traditional approaches were not going to cut it for our model.

Going Deeper with Convolutional Neural Networks

Our next step was to train a CNN model from scratch; we chose CNN because it is better suited to do feature extraction from an image than RNN. We started with 5,000 posters split across two genres, action and drama, again to start with something small which we (hopefully) build off. Using a GPU instead of a CPU was absolutely necessary at this point, as we witnessed response times improve from 50 seconds per epoch on a CPU to 2 seconds per epoch on the GPU. Even still, it was necessary to further reduce our image dimensions all the way down to 320X200 to prevent crashing. We used 3 convolution layers with kernels of size of 5 by 5, 4 by 4 and 3 by 3. We used the popular ReLU activation function and max pooling layers of size 2 by 2 after each convolution layer. From there, we played with epochs in the range of 20 to 100, used a batch size of 256 and variable learning rates from 0.1 to 0.01, but our best accuracy was 53%. The accuracy by epoch chart here shows that the model is not learning anything. The model was probably too "small" for our task and our training set too small as well. We needed help from Deep Learning professionals.

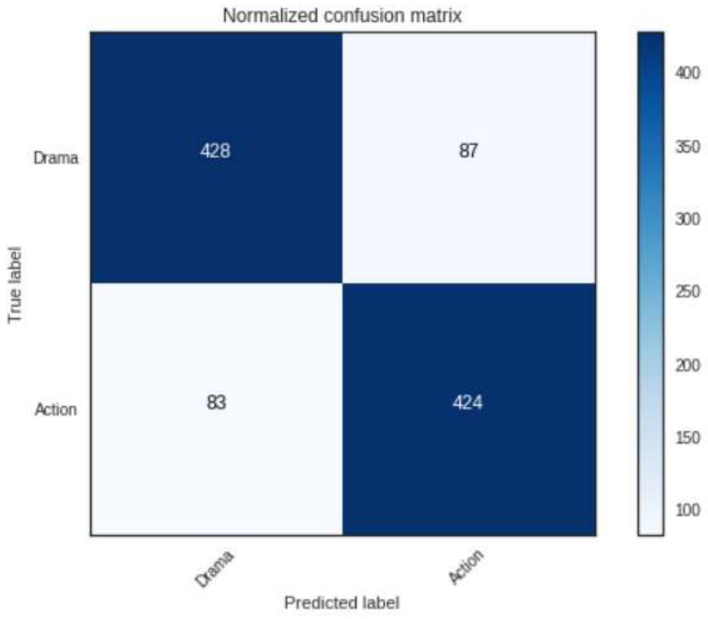




Looking at the pre-trained models available in Keras, we noticed that those more "mature" models had many more layers and neurons than what we began with. One in particular, ResNet50, focuses on feature extraction, categorizing the main object in the image per 1,000 predetermined classes. The network has 50 layers and more than 23.5 million parameters. Its convolution layers are already trained to detect shapes and contours; the first layer features are plotted here in grey scale.

There are three strategies when dealing with pre-trained models. In each case, we had to remove the last fully connected layer that is performing the classification task and replace it with another classifier designed to classify posters into genre. We could either use the first 49 already-trained layers of ResNet50 which return 2,048 features and then input those features into a classifier like a Support Vector Machine, or a second option would be to use the first 49 layers of ResNet50, add one or several fully connected layers and keep training either the full network or just the last layers. Trying to continue the training of the full model and its 23+ million parameters is out of the question even with a p2.xlarge instance with extra RAMs.

Having only 5,000 images to work with, we thought the SVM approach would be more time efficient. A simple 'RBF' SVM with default parameters returned a classification accuracy of 83.4% on our testing set! We were so proud of our results we ran to tell Stanley our results without spending too much time fine tuning the SVM. He almost fired us when we told him we could classify with a high success rate…on just two genres.



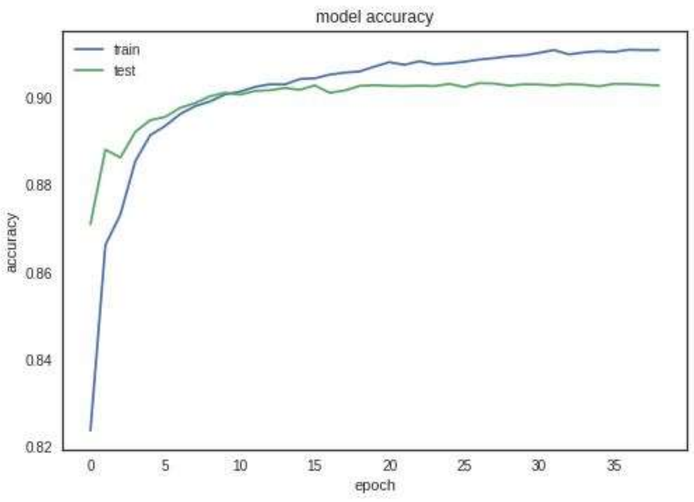
Multilabel Classification

We convinced Stanley to give us a second chance and let us see if we were able to handle multiple genres for the same movie. The TMDb database recognizes 19 genres and obviously, movies can belong to multiple genres. To complete the remaining work, we downloaded 5,000 posters from the 2010s and 4,800 from the 1970s. For each movie, we kept the 5 first genres listed by TMDb. We randomly split our dataset into training, validation and testing sets with 70%, 20% and 10% of the full dataset respectively.

Since our model could return a lot of “negatives” (a movie does not belong to any genre), a naïve model that forecasts that movies do not belong to any genre would do pretty well if we were to compute a simple accuracy ratio, hence we use the F1 score to compare the different models. This measure is the harmonic mean of precision (number of correct positive results divided by the number of all positive results) and recall (number of correct positive results divided by the number of positive results that should have been returned). The score reaches its best value at 1 and worst at 0.

We computed the 2,048 features output by the 49 first layers of the ResNet50 model, then trained an ‘RBF’ SVM on the training set from the 2010s using Sklearn’s OneVsRestClassifier function. It takes a few minutes for our GPU-boosted instance to fit those models. The model returned an average F1 score across classes of 0.51 (precision: 0.61, recall: 0.47) on the testing set from the 2010s and 0.37 (precision: 0.39, recall: 0.37) on the set from the 1970s. Training a similar SVM on the posters from the 1970s led to inversed scores. In both cases, there appears to be a real advantage in training separate models on separate decades.

We repeated the experiment with a neural network using the features extracted from ResNet50. We added 4 fully connected layers with 500, 250, 250 and 100 neurons respectively, using a ReLU activation function for each. We used the basic “deep learning tricks” to make sure we did not overfit or that the fitting was optimal. We used dropout filters set at 20%. Following recommendations of the original paper on dropout we also added a constraint on the weights for each hidden layer, ensuring that the maximum norm of the weights does not exceed a value of 3. We set up a learning rate schedule with a rate that gradually decrease every 10 epochs by 50%. We used a binary cross-entropy loss function along with a stochastic gradient approach as this is known to work well with multilabel classification. We also used an early stop strategy with a patience of 10 epochs: if we did not see an improvement in the validation accuracy after 10 epochs, the learning process would stop. The output layer used a sigmoid function since we are treating each class independently. If the output had a probability above 50%, we would classify a movie as belonging to a genre. This neural network returned slightly better results. Trained on posters from the 2010s, we got an F1 score of 0.57 (precision: 0.66, recall: 0.53) for the testing set from the same decade and 0.40 (precision: 0.46, recall: 0.40) for posters from the 1970s.

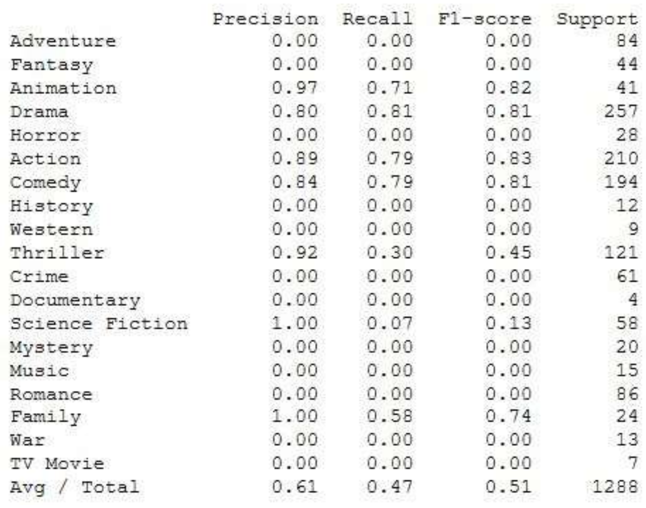


Correcting for Class Imbalances

One thing that caught our attention is that the accuracy rate starts above 80% and increases until it does not learn anything new. By forecasting by default that a movie does not belong to any class, the model does relatively well in terms of accuracy. This is because we are doing 19 binary classifications where, on average, the probability to belong to a genre is lower than 20%. It is very straightforward to correct for imbalances with Sklearn so we fit a new SVM and the scores improved dramatically with a F1 score of 0.74 (precision: 0.67, recall: 0.87). The main improvement comes from the recall variable which is higher as we are doing better at classifying less frequent genres.

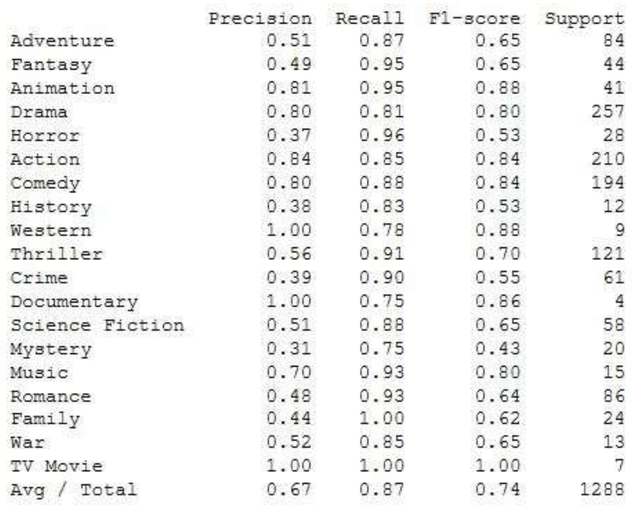
Classification report

Without correction for imbalance:



```
                 Precision  Recall  F1-score  Support
Adventure             0.00    0.00      0.00       84
Fantasy               0.00    0.00      0.00       44
Animation             0.97    0.71      0.82       41
Drama                 0.80    0.81      0.81      257
Horror                0.00    0.00      0.00       28
Action                0.89    0.79      0.83      210
Comedy                0.84    0.79      0.81      194
History               0.00    0.00      0.00       12
Western               0.00    0.00      0.00        9
Thriller              0.92    0.30      0.45      121
Crime                 0.00    0.00      0.00       61
Documentary           0.00    0.00      0.00        4
Science Fiction       1.00    0.07      0.13       58
Mystery               0.00    0.00      0.00       20
Music                 0.00    0.00      0.00       15
Romance               0.00    0.00      0.00       86
Family                1.00    0.58      0.74       24
War                   0.00    0.00      0.00       13
TV Movie              0.00    0.00      0.00        7
Avg / Total           0.61    0.47      0.51     1288
```

With correction for imbalance:



```
                 Precision  Recall  F1-score  Support
Adventure             0.51    0.87      0.65       84
Fantasy               0.49    0.95      0.65       44
Animation             0.81    0.95      0.88       41
Drama                 0.80    0.81      0.80      257
Horror                0.37    0.96      0.53       28
Action                0.84    0.85      0.84      210
Comedy                0.80    0.88      0.84      194
History               0.38    0.83      0.53       12
Western               1.00    0.78      0.88        9
Thriller              0.56    0.91      0.70      121
Crime                 0.39    0.90      0.55       61
Documentary           1.00    0.75      0.86        4
Science Fiction       0.51    0.88      0.65       58
Mystery               0.31    0.75      0.43       20
Music                 0.70    0.93      0.80       15
Romance               0.48    0.93      0.64       86
Family                0.44    1.00      0.62       24
War                   0.52    0.85      0.65       13
TV Movie              1.00    1.00      1.00        7
Avg / Total           0.67    0.87      0.74     1288
```

Happily Ever After

Ultimately, we were able to build a model that could accurately classify genre for two out of three posters while ensuring high recall levels, and Stanley was pretty jazzed about our final results, especially once we explained how the model could continue to improve as he fed it more vintage posters. He does have the release year in his database so we will be able to refine our classification by decade. He also liked that our model was able to detect rare genres as it's an important feature for his niche business. We were glad we knew to "under promise and over deliver" as an F1 score of 0.74 is a decent result for such a challenging task.

We, of course, told Stanley we would keep working on improving our results for a modest subscription to our services…or a few framed posters. With the deep learning our brains compiled on this project, we are thinking about applying the same class imbalance correction to our synthetic neural network, so stay tuned for the latest improvements in deep learning!

THE END