# Predicting Movie Ratings with Movie Posters

Christopher Pyles

May 5, 2020

## Abstract

# Contents

# 1 Introduction

The ability to predict the rating of a movie before its release would be a great asset to production companies. Ratings are generally correlated with the revenue made by a movie, as people tend to base their decisions of whether or not to see a movie on the reviews they read and the general attitude toward the movie. This analysis considers the use of movie posters as predictors of ratings, comparing the efficacy of this to classifiers built on other data about the movie.

Various techniques have been applied to the question of predicting movie ratings based on different types of data. Oghina et. al. [1] applied prediction algorithms to data from the Internet Movie Database (IMDb) and attempted to predict ratings based on social media. Their analysis is based on textual data from sources like YouTube and Twitter, which they used to extract other textual features that were used in the final predictive model. Navarathna et. al. [2] consiconsider a different model of movie rating prediction based on observations of moviegoers' faces and body language during a viewing experience. Other implementations of this problem include the problem of automatically predicting specific users' ratings, as discussed in Marović et. al. [3], who use machine learning methods to predict these ratings in service of a recommendation algorithm.

The paper by Kudagamage et. al. [4] discussed an approach to the prediction of movies' success based in data mining techniques. This analysis looks at various geometric properties of its data, including spatial clustering, and how this relates to the success of a movie. Instead of predicting movies directly, movies are partitioned into four clusters representing most successful, successful, unsuccessful, and least successful movies.

A natural extension of this research is demonstrated in Diao et. al. [5], which describes how ratings and review sentiments can be used to model movie recommendations. This model is again trained on IMDb data and thinks about how data scraped from the site on ratings and reviews can be used to build a recommendation algorithm. The ability to predict the ratings for a movie is a precursor to analyzing the market for the movie using such recommendation models, for example to predict how many streaming service users might watch this movie based on its predicted rating.

In this analysis, the use of posters as predictors of ratings is examined, using both fully-connected and convolutional neural networks as models trained on the posters. As a control, models trained on other data about the movies, e.g. the synopsis and genre, are also trained and tested as a point of comparison against the poster-based models. Different types of classifiers are used for the control, and the best of them is used as the point of comparison. Exploratory data analysis and A/B testing are also used in service of building a set of features for these control models.

## 2 Data

The data used in this project is from The Movie Database's (TMDb) API. The data queried covers movies released in 2018 in English, spanning many genres:

- Action ($n = 242$)
- Adventure ($n = 134$)

- Animation ($n = 116$)
- Comedy ($n = 566$)
- Crime ($n = 154$)
- Documentary ($n = 432$)
- Drama ($n = 746$)
- Family ($n = 147$)
- Fantasy ($n = 143$)
- History ($n = 66$)
- Horror ($n = 438$)
- Music ($n = 119$)
- Mystery ($n = 143$)
- Romance ($n = 222$)
- Science Fiction ($n = 188$)
- TV Movie ($n = 194$)
- Thriller ($n = 506$)
- War ($n = 29$)
- Western ($n = 31$)

| Column | Description |
|---|---|
| `id` | **primary key**, a unique ID number for each movie |
| `adult` | whether or not the movie is R+ rated |
| `genre_ids` | list of genre ID numbers corresponding to `genres` table |
| `original_language` | the language that the movie was originally released in |
| `original_title` | the title of the movie |
| `overview` | movie synopsis |
| `release_date` | the date of first release |
| `vote_average` | average of rating votes on a 10-point scale |
| `vote_count` | the number of votes |
| `poster_path` | URL path to movie poster |

Table 1: Column descriptions for TMDb API data (the `movies` table).

The data, after being queried, were joined into a single table and written to a CSV file. The columns of interest are described in Table 1. After dropping rows with missing values in the columns of interest, the data contained $n = 2700$ rows.

## 2.1 Data Cleaning

Cleaning the data for this project, after dropping missing values, involved rounding the `vote_average` column, cleaning the synopsis strings, joining the `genres` and `movies` tables, and converting the movie posters into $140 \times 92 \times 3$ arrays of RGB values.

The `vote_average` column ($\mu = 6.628$, $\sigma = 1.939$) describes the variable of interest, representing the average rating across votes for a single movie. In this analysis, this column will be transformed from a continuous variable into an ordinal variable with possible values 1 to 10 by rounding the vote average to a whole number. After rounding, the distribution of ratings is provided in Figure 1.

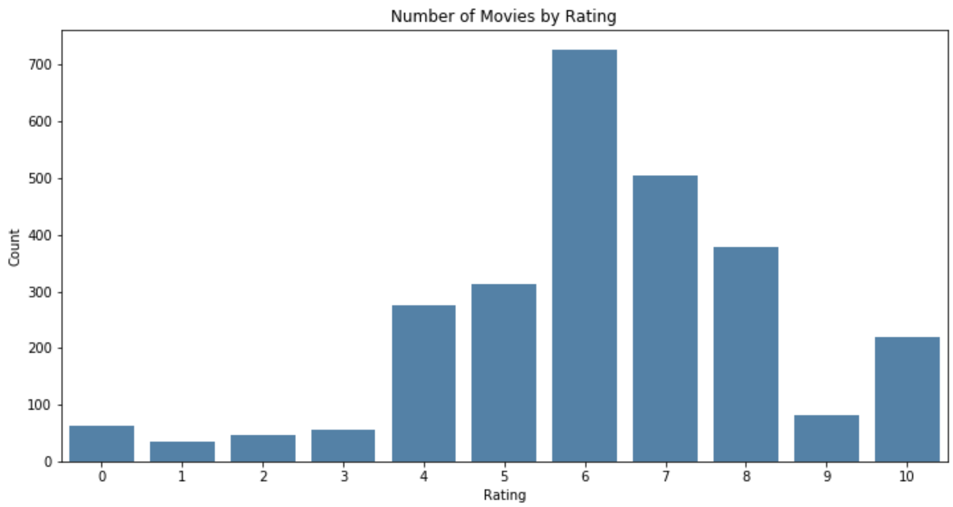


Figure 1: Number of movies for each value of ratings, rounded from `vote_average`.

To clean the movie synopses, all letters were transformed to lowercas, and any characters not matching the regex `[A-Za-z0-9 ]` were replaced with spaces.

The `genre_ids` column is a list of genre IDs encoded as a string, so to start any empty lists, `"[]"`, were replaced with `NaN`. Then, the string were split into Python lists of IDs, and were joined with the `genres` table, so that `movies` had a column `genres` where each value is a list of genres for that movie. Figure 2 shows the breakdown of movies by genre.

Lastly, each movie poster was downloaded from TMDb and stored as a 92 $\times$ 140 $\times$ 3 array in NumPy, stored as a .mat file. The structure of this file is analagous to a dictionary, where each key is a movie ID (as a string) and each value is the 3-D array of RGB values describing the poster.

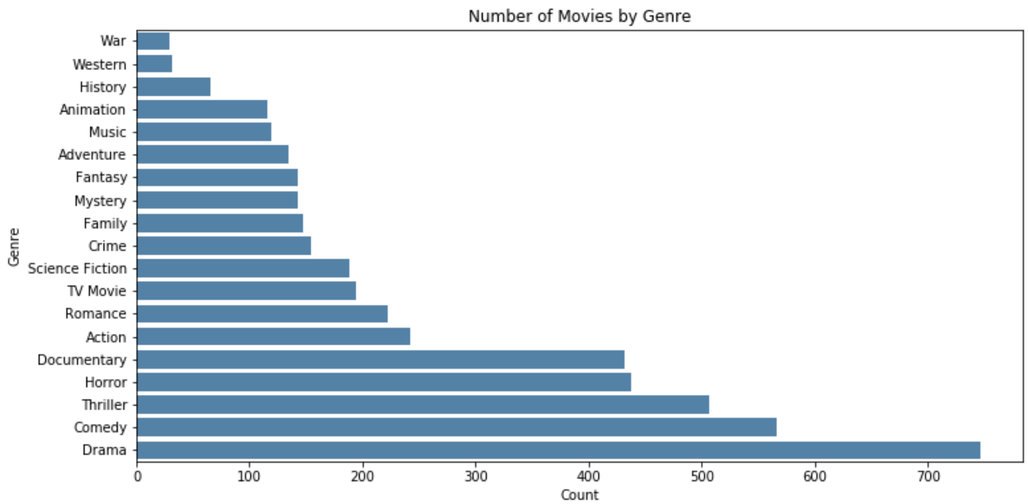


Figure 2: Number of movies in each genre.

## 2.2 Exploratory Data Analysis

Before modeling, a cursory analysis of the data yielded the interesting relationships:

- Figure 3 shows that vote count tends to increase logarithmically with rating until 8, after which it drops significantly. This demonstrates that having more votes tends to "bring down the curve."

- There are significantly more non-adult movies than adult movies, as demonstrated in Figure 4. It is interesting to note that adult movies have a double-peak distribution with far less spread than do the non-adult movies.

- Figure 5 shows the distribution of ratings for each genre.

- No discernible relationship is seen between the length of the overview and the

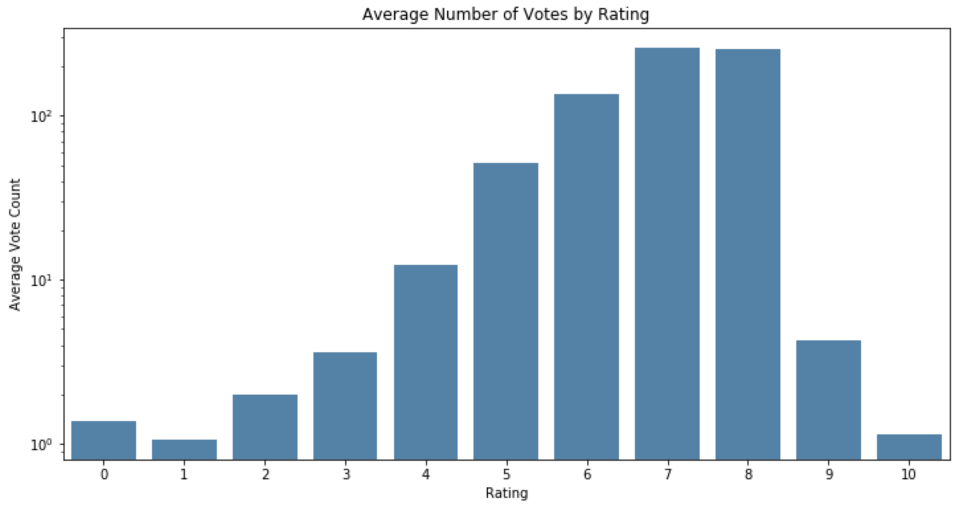


Figure 3: Average vote count by rating.

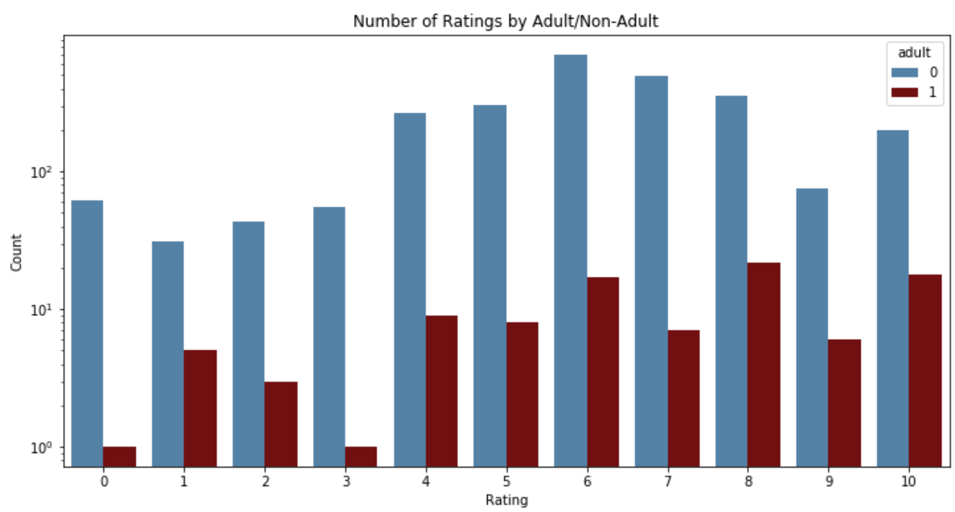


Figure 4: Number of ratings by adult/non-adult.

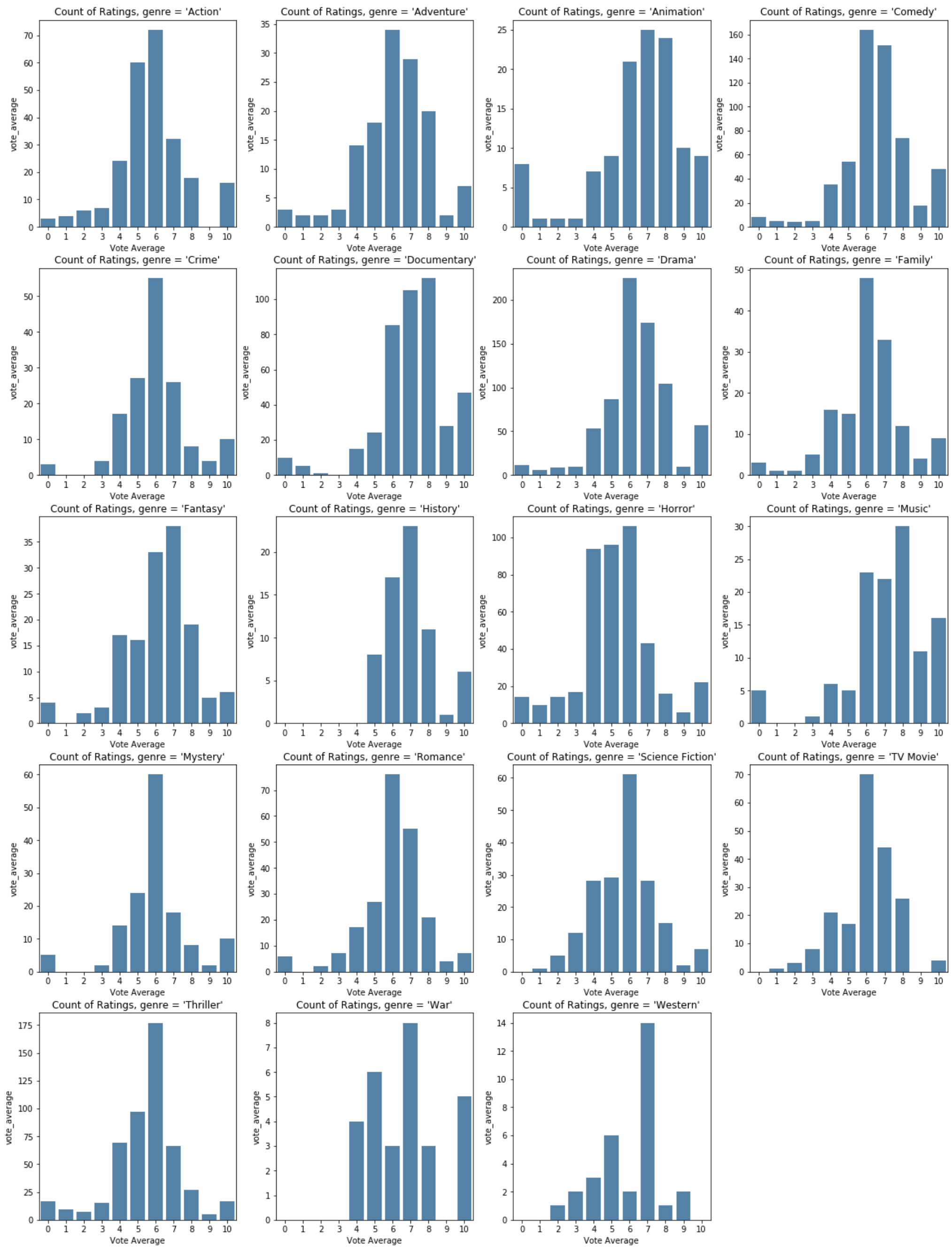


Figure 5: Distribution of ratings by genre.

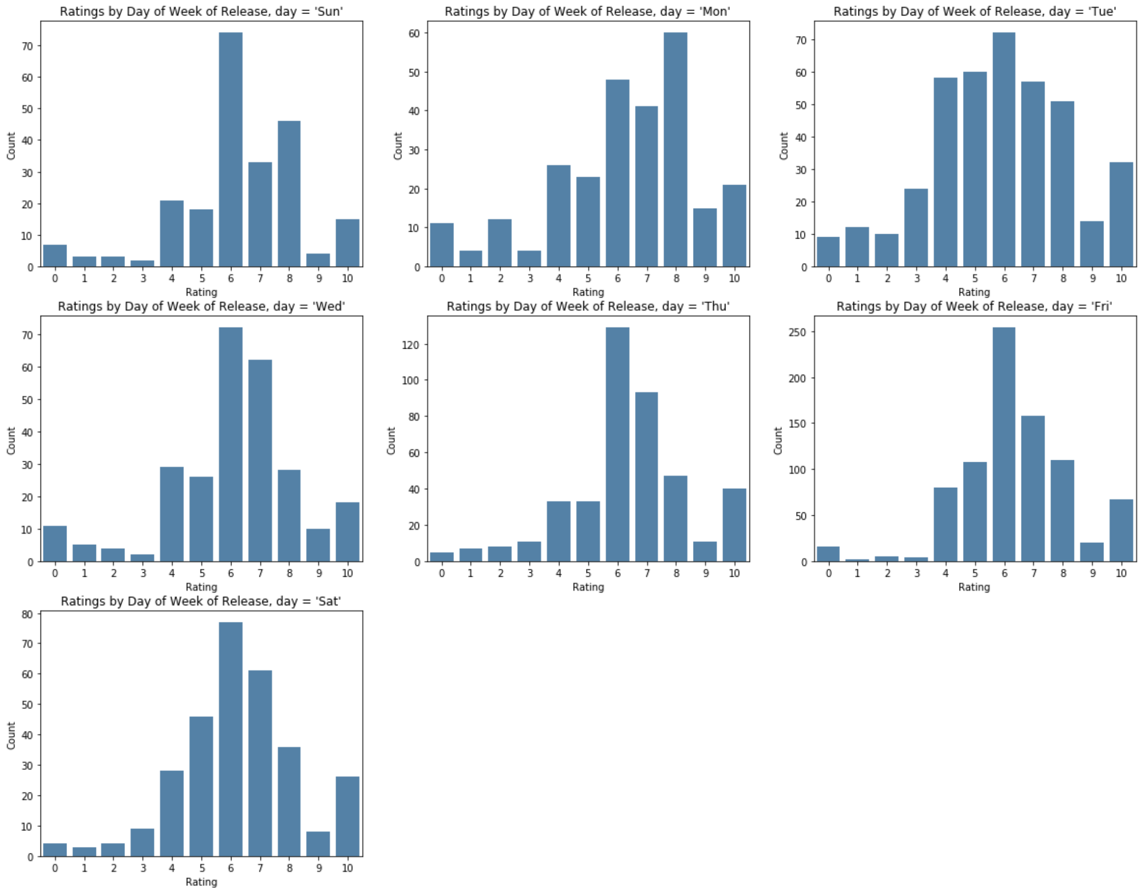


Figure 6: Distribution of ratings by day of week of release.

movie's rating.

- The distributions of ratings by release date day of week appear to be different in skew (Figure 6), indicating that this may be an important feature. No discernible relationship is seen between rating and day of month or month of release.

# 3 Analysis

## 3.1 Words in Synopses

As the first part of this analysis, the statistical significance of the presence of different words in synopses is studied using hypothesis testing. The null hypothesis for this test is that the presence of a word does not affect the average rating, and the alternative is that the presence of the word results in an increase in rating. The test statistic used here is the aboslute difference between the mean rating of movies where the word is present in the synopsis and the movies where it is not. Under the null hypothesis, the truth value of a word being present in a synopsis is shuffled for each observation, so that same number of "present"s is obtained. Then, the test statistic is computed and the p-value is calculated as the proportion of the test statistic values that are greater than or equal to the observed value for that word. Finally, those words whose p-values indicate statistical significance are added to the movies data as one-hot features, with a 1 in their column if the word is present in the synposis and a 0 otherwise.

## 3.2 Principal Components Analysis

After analyzing the presence of words in synopses, the dimensionality of the data is considered for reduction using principal components analysis (PCA). In this portion of the analysis, all of the features now in `movies` are analyzes via PCA in order to determine what proportion of the variance in the data can be explained using only a few principal components as a basis. This is accomplished with scikit-learn's `sklearn.decomposition.PCA` class.

## 3.3 Classification without Posters

The next step in the analysis is to attempt classification using the features in `movies` alone (i.e. without the posters). In this section, several different classification methods are tested, including a ridge classifier, a decision tree classifier, a support vector classifier (SVC), and a $k$-nearest neighbors (kNN) classifier. Each of these models is trained on training data from `movies` and evaluated using different error metrics on testing data from `movies`. Three different error metrics are used to evaluate each classifier: classification accuracy, root-mean-squared prediction error (RMSE), and mean absolute error (MAE). The last two, normally regression metrics, are used because the ratings are derived from continuous variables, and a classifier that classifies a movie closer to its rating than another is more accurate, and hit-miss accuracy does not account for this.

In order to select features, multiple methods are used and compared. The $k$ best features based on chi-squared statistics, ANOVA F-value, and mutual information are each selected and compared using tuned hyperparameters for each model. $k$ is also tuned using hyperparameter selection.

5-fold cross-validation (CV) is used to tune hyperparameters. The hyperparameters being tuned for each model are:

- ridge classifier: $\alpha$, the regularization strength
- SVC: $C$, the regularization parameter (analagous to $\alpha^{-1}$ from the ridge classi-

fier)

- kNN classifier: the number of neighbors

Hyperparameters are tuned individually for sets of features chosen by the different methods of feature selection described above. The model with the best CV error is chosen and trained on the training set using the corresponding features and evaluated using testing error.

### 3.4 Classifying Movie Posters

In this portion, movie posters, stored as 3-D arrays, are used as inputs to conlutional neural networks (CNNs) in order to classify the movie by rating. This section does not involve any features in `movies`. CNNs are trained on the 4-D array of all images and the classes encoded as dummy variables. The networks are compiled using categorical cross-entropy loss and the Adam optimization algorithm. They are evaluated as in §3.3, using classification accuracy, RMSE, and MAE with 5-fold CV to determine the best network structure. The model with the best metrics is trained on the entire training set and evaluated on the test set.

## 4 Results

### 4.1 Words in Synopses

This analysis found that there were 110 words that were statistically significant, summarized in Table 5 (cf. Appendix A) with their p-values. Each of these words

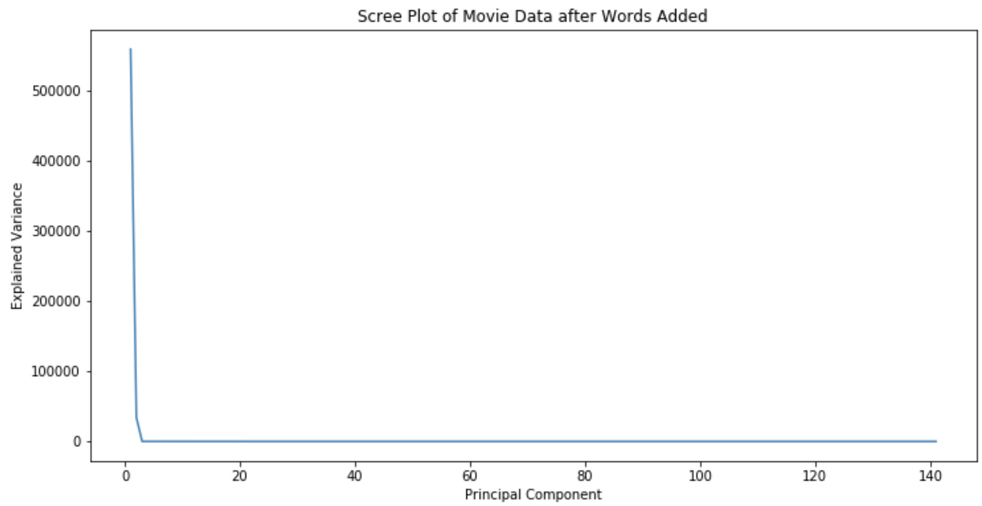


Figure 7: Scree plot of principal components.

was added as a feature to `movies` as dummy variables with 0-1 values indicating the presence of that word in the synopsis, resulting in the data points in `movies` becoming 141-dimensional.

## 4.2 Principal Components Analysis

The PCA on `movies` shows that only a few principal componetns (two, in fact) are required to account for almost 100% of the variance in the data set, as demonstrated in Figures 7 and 8.

As Figure 9 shows, the first two princpal components account for almost 100% of the variance in the dataset. The rotated data points are plotted along Princpal Components 1 and 2 in Figure 10.

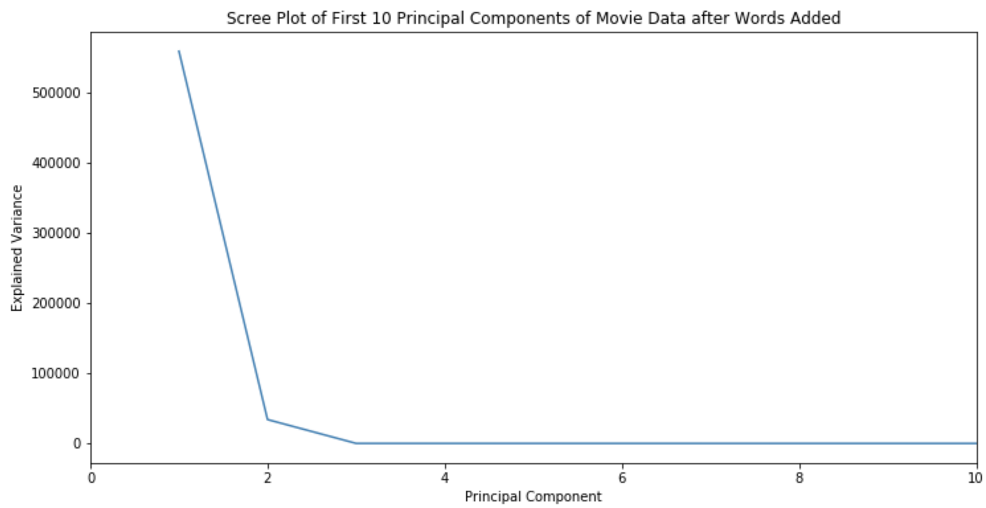


Figure 8: Scree plot of first 10 principal components.

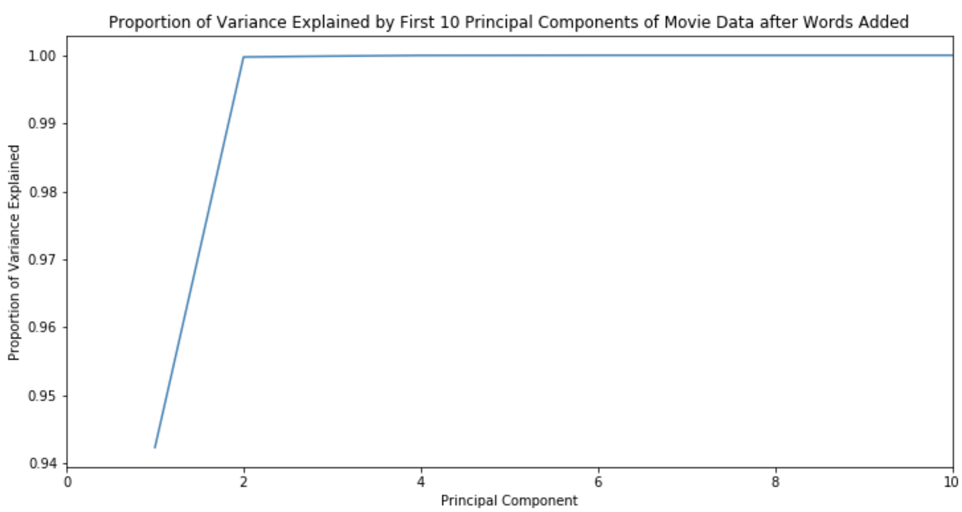


Figure 9: Cumulative proportion of variance explained by first 10 principal components.

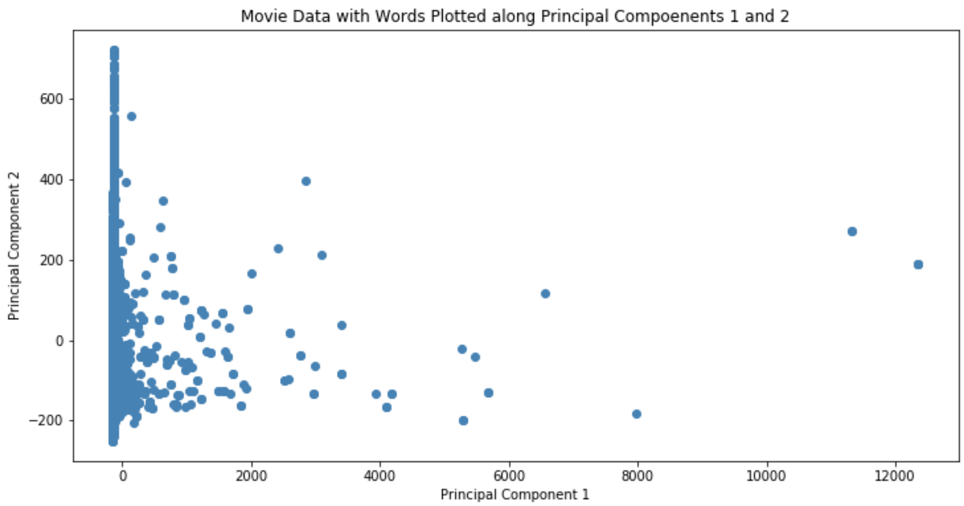


Figure 10: Rotated data plotted along Princpal Components 1 and 2.

### 4.3 Classification without Posters

After the first round of hyperparameter search, the ridge classifier performed best with $\alpha = 0$, the support vector classifier with $C = 100$, and the kNN classifier with 1 neighbor. The decision tree classifier had no hyperparameters of interest. The results of feature selection and tuning are provided in Table 2. Table 3 details the testing MAE and RMSE for each model and feature selection scheme. The hyperparameters used in these models are:

- ridge: $\alpha = 0$
- SVC: $C = 100$
- kNN: 1 neighbor

| Classifier → | Ridge | | SVC | | Decision Tree | kNN | |
|---|---|---|---|---|---|---|---|
| **Feature Selection Scheme ↓** | $\alpha$ | **MAE** | $C$ | **MAE** | **MAE** | **neighbors** | **MAE** |
| **No Selection** | 0 | 1.19740 | 100 | 0 | 0 | 1 | 0 |
| **Chi-Squared Statistics** | 0.1 | 1.40022 | 10 | 0 | 0 | 1 | 0 |
| **ANOVA F-value** | 0 | 1.40130 | 100 | 1.15998 | 0.27983 | 1 | 0.32484 |
| **Mutual Information** | 0 | 1.41811 | 10 | 0 | 0 | 1 | 0 |

Table 2: Results of feature selection and hyperparameter tuning without posters.

| Classifier → | Ridge | | SVC | | Decision Tree | | kNN | |
|---|---|---|---|---|---|---|---|---|
| **Feature Selection Scheme ↓** | **MAE** | **RMSE** | **MAE** | **RMSE** | **MAE** | **RMSE** | **MAE** | **RMSE** |
| **No Selection** | 1.46016 | 2.14438 | 0.95772 | 1.80289 | 0.95610 | 1.94309 | 1.07805 | 1.94518 |
| **Chi-Squared Statistics** | 1.36260 | 1.98449 | 0.90569 | 1.76460 | 1.01789 | 2.01377 | 1.07967 | 1.96638 |
| **ANOVA F-value** | 1.38537 | 2.03386 | 1.43577 | 2.15836 | 1.14472 | 2.15271 | 1.18699 | 2.20421 |
| **Mutual Information** | 1.46992 | 2.04661 | 0.87154 | 1.72500 | 1.03089 | 1.96886 | 1.08618 | 1.96804 |

Table 3: Classifier testing errors on classification without posters.

The results show that a support vector classifier with $C = 100$ and feature selection using mutual information is the best classifier, with an MAE of 0.87154 and an RMSE of 1.72500. The predictions of this classifier are depicted in Figure 11. This feature selection scheme chose 10 features: `popularity`, `vote_count`, `Horror`, `Music`, `Thriller`, `overview_len`, `day`, `take`, `direct`, and `evil`. The last three features correspond to the presence of words as described in §4.1.

## 4.4 Classifying Movie Posters

This section created multiple neural networks, both convolutional and otherwise. The non-convolutional neural networks were trained on the flattened image arrays and contained different numbers dense hidden layers. Both softmax and sigmoid activation functions for the final layer were considered. The convolutional NNs were trained on the 3-dimensional image arrays, which were then flattened and run through non-convolutional dense layers. All NNs in this section use the Adam optimizer and

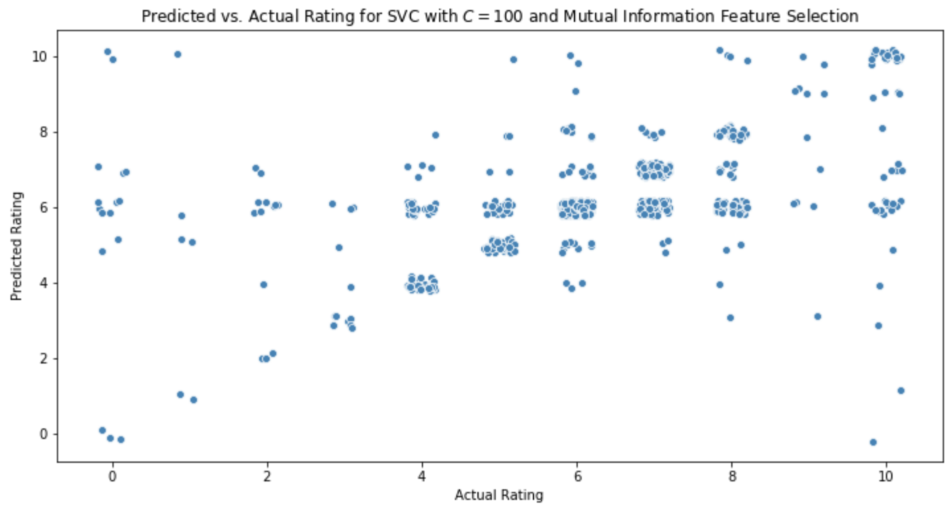


Figure 11: Results of classification without posters with $U(-0.2, 0.2)$ jittering on both axes.

categorical cross-entropy loss.

Some neural network models analyzed in this section predicted all fours or all zeros, including one of the convolutional NNs. Figure 13 shows the features picked out by the first convolutional layer of this network when predicting on the poster in Figure 12.

A better CNN trained in this section, summarized in Figure 14, predictions with an MAE of 1.96941 and an RMSE of 2.62073. The features learned by the input layer of this model are shown in Figure 15.

What is interesting to note is that the all-fours CNN learns a more diffuse representation and a wider variety of features in the model. Figure 13 highlights grays of different hues and some of the neurons highlight textual features in the image. Con-

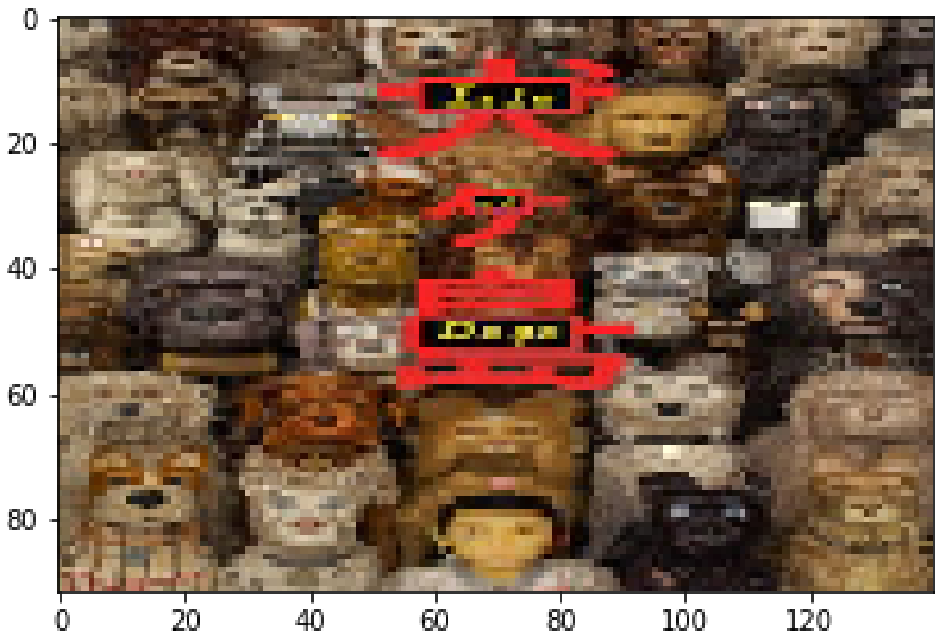


Figure 12: Original poster for picturing CNN layer features.

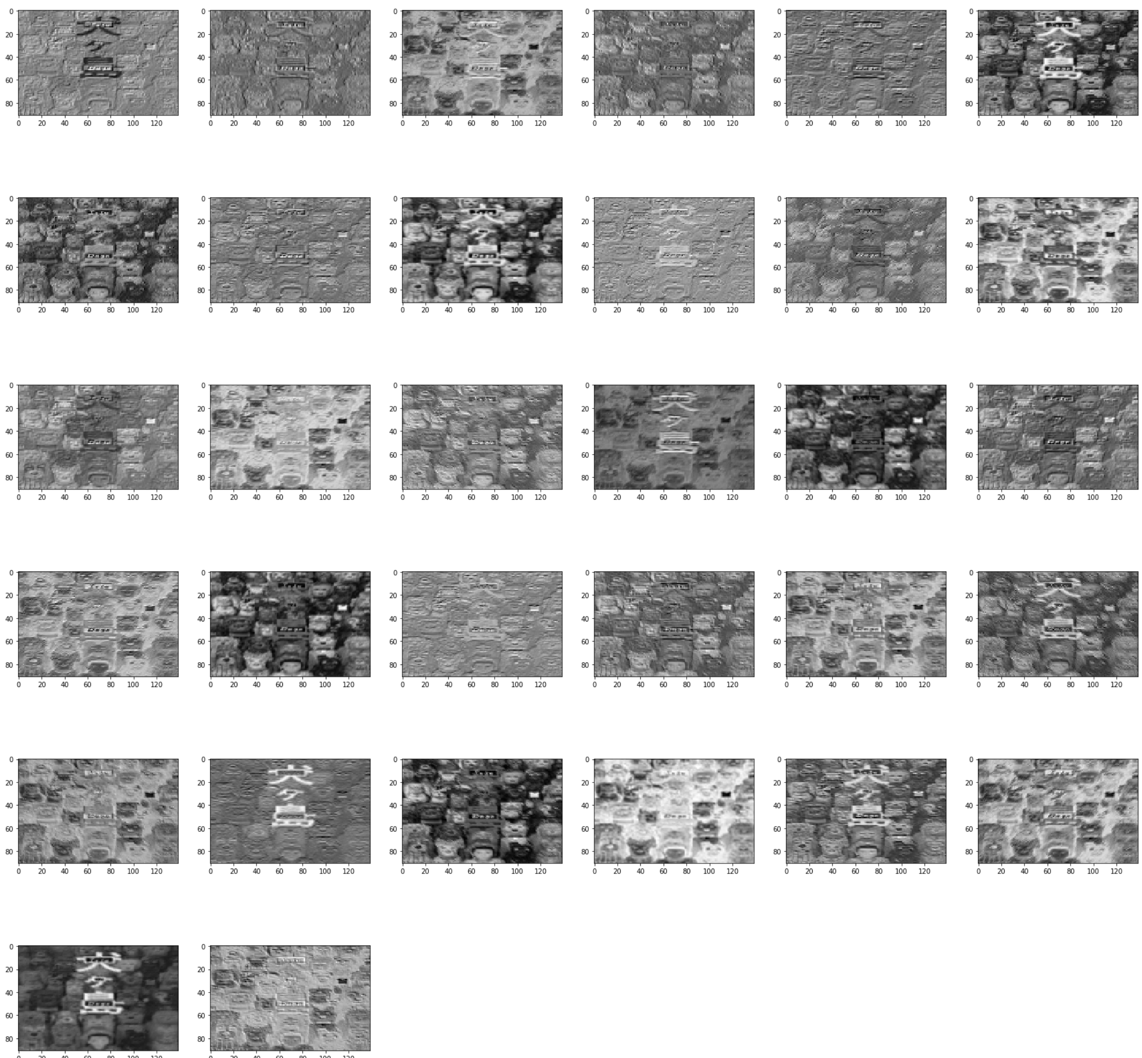

Figure 13: All-fours CNN first layer features.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 89, 137, 64)       3136
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 86, 134, 32)       32800
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 83, 131, 32)       16416
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 80, 128, 16)       8208
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 77, 125, 16)       4112
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 74, 122, 16)       4112
_________________________________________________________________
flatten_3 (Flatten)          (None, 144448)            0
_________________________________________________________________
dense_6 (Dense)              (None, 11)                1588939
=================================================================
Total params: 1,657,723
Trainable params: 1,657,723
Non-trainable params: 0
_________________________________________________________________
```

Figure 14: Better CNN model summary.

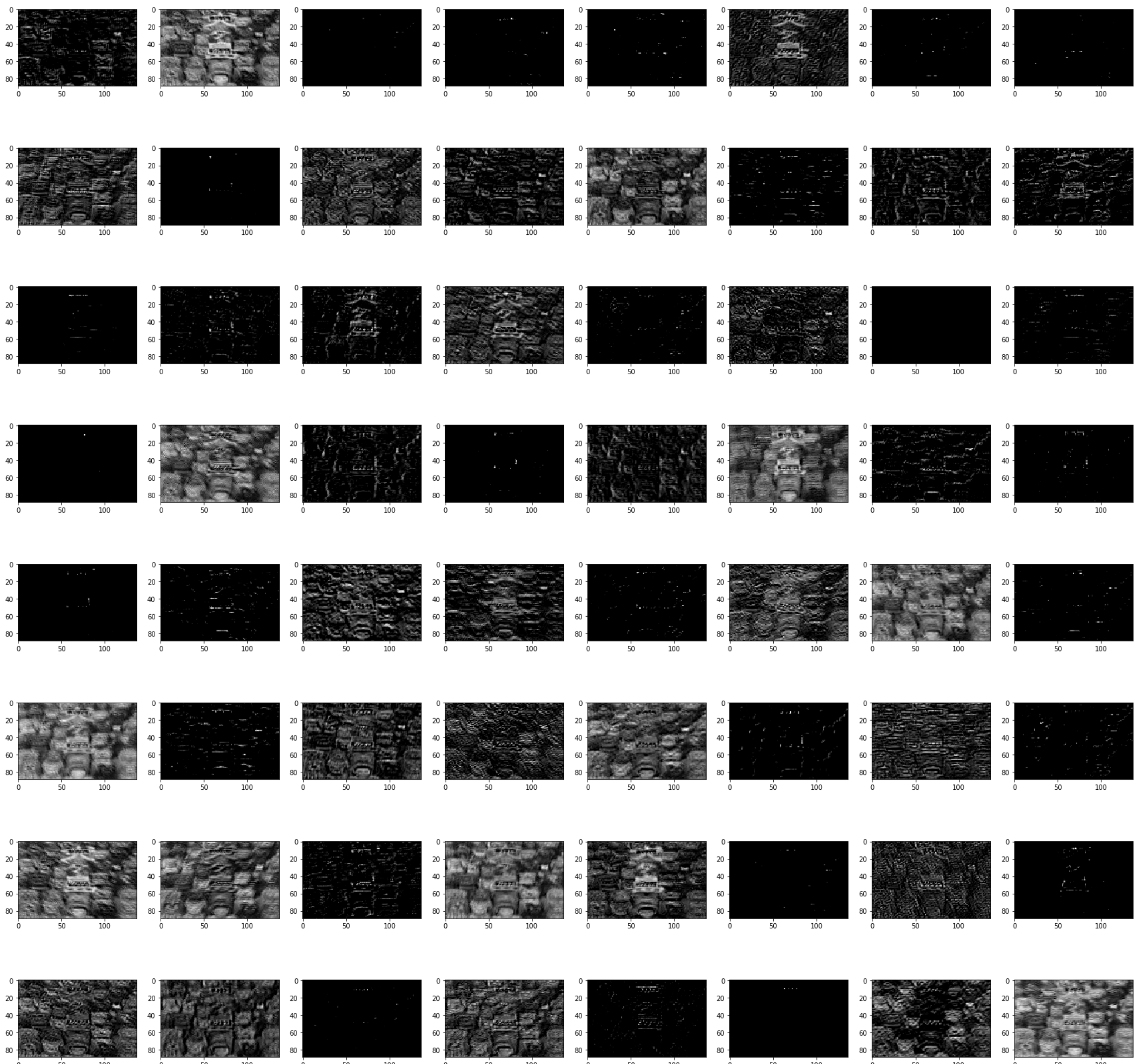

Figure 15: Better CNN input layer features.

| **Number of Hidden Units** | **MAE** |
|---|---|
| 1 | 1.989720549637178 |
| 2 | 1.9078307858576502 |
| 4 | 1.6101929905820598 |
| 8 | 2.181723019916628 |
| 16 | 1.712555195306469 |
| 32 | 1.5834213370387524 |
| 64 | 1.492082754361587 |
| 128 | 1.492082754361587 |
| 256 | 2.388825073336421 |
| 512 | 2.0121877412382276 |
| 1024 | 1.9250671607225567 |

Table 4: MAEs for different numbers of hidden units in non-convolutional FCNNs with 1 hidden layer.

trast this to Figure 15, which shows several edge detectors (the mostly-black frames) and only a few neurons that learn more textual features. None of the neurons in the better CNN appear to recognize different shades of gray. This indicates that perhaps edge detection is a better method of learning features for predicting movies. Lower layers of this CNN learned to detect different textures in addition to edges, as shown in the Figure 16.

Several non-convolution neural networks were tested, including 11 non-convolutional fully-connected networks (FCNNs) with 1 hidden layer trained over 100 epochs with batch size 16. The MAEs for these networks for different numbers of hidden units are provided in Table 4.

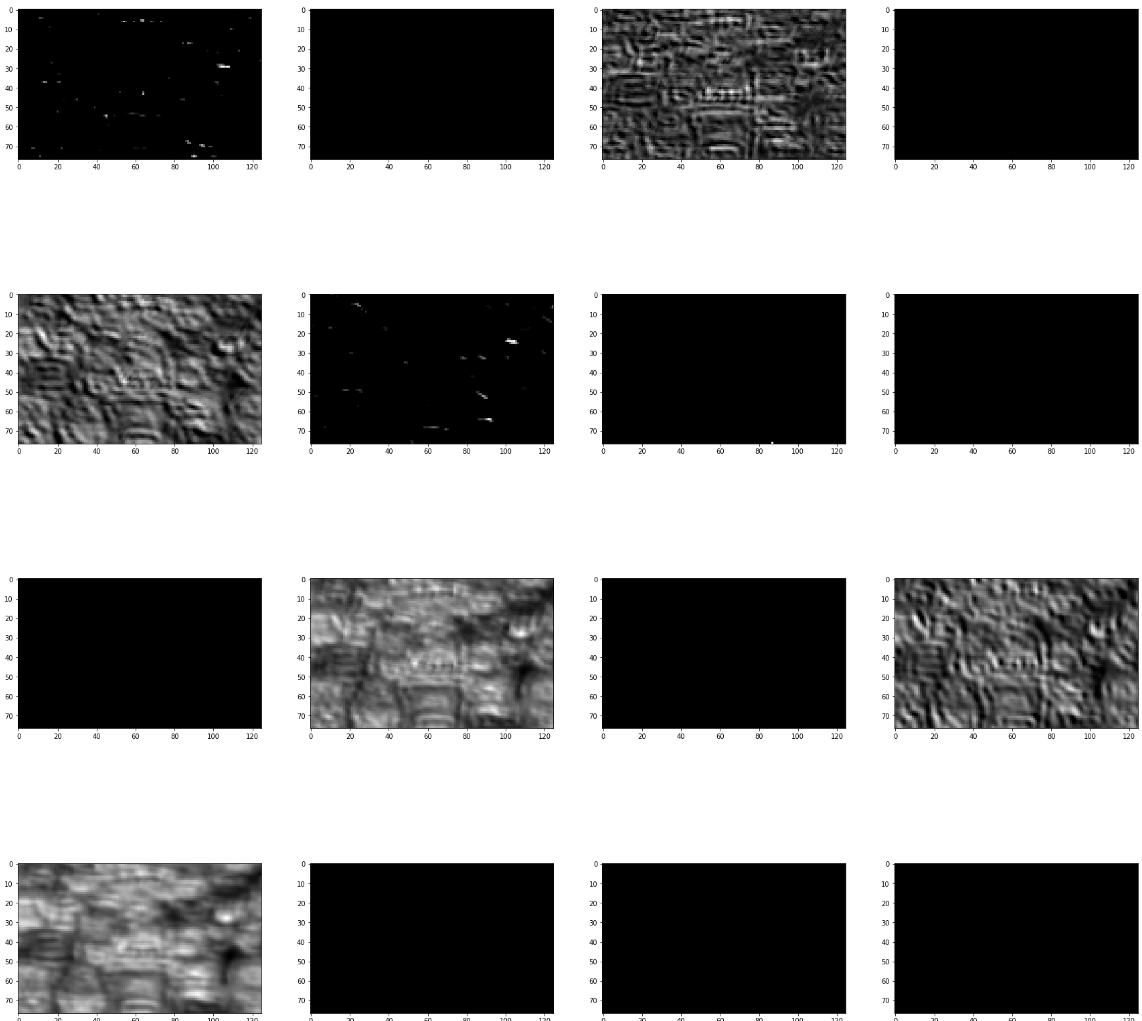

Figure 16: Better CNN fifth layer features.

# 5 Conclusions

Several interesting conclusions can be drawn from the analysis of this project, with two salient points:

1. fully-connected neural networks trained on flattened images appear to be better predictors than do convolutional neural networks, and

2. movie posters are not the best predictors of ratings

To the first, compare the errors noted in Table 4 to those noted for the CNNs. The different is clearly in favor of the FCNNs on this score. One possible reason for this, however, may be the lack of depth of the CNNs trained in this analysis. Traditionally, deep learning performed with CNNs is done with networks of 15+ layers, a level clearly not broached in this analysis due to computation power limitations. So while, at a shallow level, it appears that the FCNNs have the upper hand, the convolutional networks may be able to outperform them at the level of deep learning.

To the second, we observed that when comparing even the best networks trained in this analysis that they do not outperform the classifiers trained on other pieces of movie data from the `movies` dataframe. This is perhaps a reasonable conclusion as posters are more determinant of a person's desire to see a movie in the first place, and so likely have a lesser effect on their *rating* of that movie after having seen it.

The principal components analysis reveals that while the data are largely well-represented by two principal components, which together explain almost 100% of the variance in the data, there is quite a lot of variance among the sample points themselves. This is observed in the scale of the axes of Figures 7 and 10. Figure

10 shows in particular that there is a more uniform density of points along different values of Principal Component 2 compared to a more right-skewed distribution along Principal Component 1.

Some suggestions for further analysis include training deeper networks on the 3-tensor posters in order to see if these can outperform the FCNNs trained herein, and using posters perhaps as a predictor of box office-success. As noted above, posters are usually more involved in a person's decision to see a movie, and so the profits of a movie are likely better predicted by posters than are the ratings.

# References


[1] Andrei Oghina, Mathias Breuss, Manos Tsagkias, and Maarten de Rijke. Predicting IMDB Movie Ratings Using Social Media. In Ricardo Baeza-Yates, Arjen P. de Vries, Hugo Zaragoza, B. Barla Cambazoglu, Vanessa Murdock, Ronny Lempel, and Fabrizio Silvestri, editors, *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 503–507, Berlin, Heidelberg, 2012. Springer.

[2] Rajitha Navarathna, Patrick Lucey, Peter Carr, Elizabeth Carter, Sridha Sridharan, and Iain Matthews. Predicting movie ratings from audience behaviors. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1058–1065, March 2014. ISSN: 1550-5790.

[3] Mladen Marović, Marko Mihoković, Mladen Mikša, Siniša Pribil, and Alan Tus. Automatic movie ratings prediction using machine learning. In *2011 Proceedings of the 34th International Convention MIPRO*, pages 1640–1645, May 2011.

[4] Upeksha Kudagamage, Banage Kumara, and Chaminda Baduraliya. Data Mining Approach to Analysis and Prediction of Movie Success. August 2018.

[5] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14, pages 193–202, New York, New York, USA, August 2014. Association for Computing Machinery.

# A Appendix: Word Signficiance Table

| Word | p-value | Word | p-value | Word | p-value |
|---|---|---|---|---|---|
| this | 0.001 | through | 0.005 | rough | 0.004 |
| after | 0.006 | take | 0.012 | cross | 0.002 |
| known | 0.003 | film | 0.004 | hard | 0.001 |
| killer | 0.001 | gain | 0.041 | brother | 0.037 |
| business | 0.019 | trip | 0.02 | childhood | 0.001 |
| seem | 0.009 | care | 0.009 | small | 0.018 |
| follows | 0.028 | couple | 0.049 | director | 0.02 |
| become | 0.03 | even | 0.042 | danger | 0.005 |
| house | 0.003 | party | 0.027 | dang | 0.002 |
| aunt | 0.017 | cove | 0.003 | between | 0.003 |
| each | 0.012 | down | 0.004 | mean | 0.003 |
| making | 0.001 | story | 0.01 | christmas | 0.016 |
| place | 0.002 | father | 0.04 | year | 0.018 |
| foot | 0.009 | disc | 0.009 | press | 0.046 |
| lore | 0.036 | under | 0.047 | ross | 0.005 |
| give | 0.038 | ller | 0.003 | dark | 0.001 |
| found | 0.031 | busi | 0.008 | document | 0.002 |
| front | 0.047 | direct | 0.007 | brea | 0.009 |
| women | 0.024 | light | 0.002 | anger | 0.001 |
| ours | 0.035 | does | 0.018 | strange | 0.005 |
| behind | 0.001 | dangerous | 0.002 | hunt | 0.002 |
| kill | 0.004 | break | 0.007 | takes | 0.004 |
| attempt | 0.01 | evil | 0.005 | fall | 0.014 |
| disco | 0.003 | host | 0.002 | which | 0.029 |
| mysterious | 0.004 | survive | 0.001 | soon | 0.002 |
| king | 0.015 | meet | 0.037 | know | 0.017 |
| less | 0.013 | realize | 0.03 | wing | 0.032 |
| turn | 0.049 | college | 0.014 | form | 0.001 |
| begin | 0.001 | documentary | 0.005 | test | 0.02 |
| years | 0.002 | self | 0.02 | cover | 0.026 |
| range | 0.001 | super | 0.042 | mall | 0.04 |
| discover | 0.011 | across | 0.006 | | |

Table 5: Words with statistically significant presences and corresponding p-values.