Predicting Genre from Movie Posters

Gabriel Barney (barneyga) and Kris Kaya (kkaya23)

***Abstract*— The film industry is incredibly reliant upon the use of posters to advertise movies in the hopes of increasing viewership and hence profits. A good poster is able to convey important qualities of a film such as theme and genre to make the movie seem as appealing to as wide of a viewership as possible. We wanted to see if we could train a model to learn features of a poster that could successfully predict the genre of the movie it represents. We found that ResNet34 and a custom architecture were reasonably successful at predicting a poster's genres. This implies that there is relative consistency in visual elements included in movie posters of a given genre.**

Introduction

Posters represent one way in which people use media to influence human behavior. One industry that is reliant upon the efficacy of posters is the film industry. It is advantageous for movies to have good posters, since they increase interest in watching the movie. Good posters are posters able to communicate important aspects of a film such as cast, theme, and elements of plot. Thus, designers have incentive to include salient features in their posters to make their movie attract more viewers. The topic of image recognition with respect to movie posters is interesting for a few reasons. First, it would have practical implications for advertisers of movies. Movies are commonly marketed using posters, which means that a model that could identify features that predispose them to success would be beneficial. Second, it would be an interesting experiment into what aspects of posters individuals find appealing and is useful for media creators generally. Finally, the topic of image recognition allows us to take explore a wide variety of algorithms and network architectures to create a successful model.

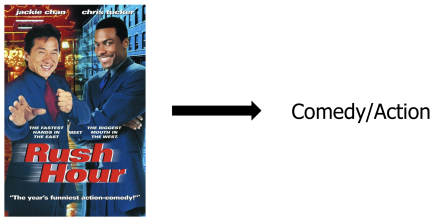

Fig. 1. A visualization of our problem.

Our project explores if there are certain elements of a poster that allow a model to predict the movie's genre. The input to our algorithm is a color image of a movie poster and our model outputs a list of genres that classify the movie. We apply various deep learning models and strategies to learn features of the poster to make our predictions.

Related Work

There were attempts in the literature at models that predict a movie's genre using non-visual promotional materials. A study conducted by Hoang [4] used various machine learning methods such as Naive Bayes and RNN's to predict a movie's genre using plot summaries. It found that a Gated Recurrent Units neural network was able to identify genre in 80.5% of cases. Another study conducted by Makita and Lenskiy [5] used a multi-variate Bernoulli event model to learn likelihood of genre based off of a movie's ratings. It had a success rate of 50%, which is a reasonably significant result.

Beyond the previous studies, there have also been attempts at genre prediction using images. Wehrmann and Barros [6] used Convolutional Neural Networks to perform multi-label genre classification from movie trailers. This shows efforts to identify genre of a movie on the basis of visual elements of its advertisements. The study found that the use of CNNs outperformed pre-existing methods of genre-classification such as the limiting of features. We plan on using CNN's for the purpose of our project and modifying the network architecture. Work done by Ivasic-Kos, Pobar, and Mikec [7] represented one of the earliest attempts at classifying movies into genres on the basis of their poster. Using a dataset of 1500 posters with 6 genre labels, they were able to predict a movies entire genre classification 14% of the time. Our project attempts to expand upon this groundwork by using a more robust dataset, more genre labels, and more modern techniques. A more recent attempt by Chu and Guo [8] attempts to apply more modern frameworks to the problem of poster classification by using the pre-trained YOLO version 2 network for object detection yielding promising results.

Dataset

Fig. 2. An example poster from our dataset

We used the Full MovieLens Dataset [3] from Kaggle, which consists of meta-data collected from TMDB and GroupLens. The dataset contains entries for 45,466 movies. Each entry for a given movie contains elements about the film such as genre, cast, and most importantly, 500x750

resolution poster images. We preprocessed the data to delete unnecessary information, standardize formatting, and remove entries with improper formatting, such as those containing unidentifiable characters or forward slashes. Then, we replaced all characters that cannot be represented in file names with parentheticals containing understandable terms such as (QM) for question marks. We deleted any movies that have the exact same title, except for the first movie of that title, and deleted all movies with no listed genres. This yielded a dataset of 35000 movies with the following genre distribution: **War**: 1080, **Fantasy**: 1760, **Mystery**: 2065, **TV Movie**: 655, **Science Fiction**: 2547, **Western**: 949, **Comedy**: 10961, **Documentary**: 3494, **Crime**: 3496, **Action**: 5164, **Music**: 1422, **Adventure**: 2831, **Family**: 2337, **Thriller**: 6353, **History**: 1072, **Horror**: 3968, **Foreign**: 1104, **Drama**: 15941, **Romance**: 5471, **Animation**: 1351

FEATURES

We formatted each individual image into a 224x224 resolution image. This was done automatically using a preprocessing script. Our raw input data is the color of each pixel in the image expressed in terms of RGB values - a 224x224x3 matrix. Therefore, the specific features that we analyze are the RGB values at each of the pixels. The genres associated with a given movie are expressed as a length 20 vector with one-hot encoding. If a given movie belongs to a genre, the value at the associated index is 1. Otherwise, the value at the index is zero.

METHODS

A. Multi-Label K-Nearest Neighbors

One algorithm we implemented is Multi-Label K-Nearest Neighbors, or ML KNN. It uses the k nearest neighbors for every unseen training instance and then uses maximum a posteriori to determine the label set for the unseen instance. We decided that this would be a good approach for a few reasons. First, it seems like an intuitive approach towards identifying genre from movie poster - a multi-label classification problem. Second, experiments show ML-KNN achieves superior performance to some well-established algorithms in the literature. [1] We implemented KNN using Hamming loss as the main metric to evaluate our algorithms performance, which evaluates how many times an instance-label pair is misclassified. Our algorithm attempts to minimize hamming loss. The formula for Hamming loss is included below:

$$hloss_s(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{Q}|h(x_i)\Delta Y_i|$$

Where Q: number of genres, Y_i: set of labels for movie i, $h(x)$: our multi-label classifier, Δ: the symmetric difference between two sets

B. ResNet Transfer Learning

Deeper neural networks are difficult to train since as network depth increases, accuracy can become overly saturated and suddenly degrade. Using a deep residual learning framework, the ResNet34 model [9] lets deeper layers fit a residual mapping, which is easier to optimize. In practice, this results in neural networks with shortcut connections that perform identity mapping. The result of these connections were then added to the outputs of the stacked layers. The ResNet34 solves a problem plaguing deep Neural Networks known as vanishing gradients. In deeper networks, the gradients that are calculated from the loss function approach zero. This prevents learning since the weights are never updated. Therefore, ResNet employs the usage of residual blocks, which are blocks of the neural network that contain shortcut connections. Due to the presence of shortcut connections, which is an identity connection from earlier parts of the network, the residual block tries to learn the residual of the true output Using these connections avoids the problem by preserving the gradient since we backpropagate through the identity function. This allows for the propagation of larger gradients and therefore, lets us train deeper networks.

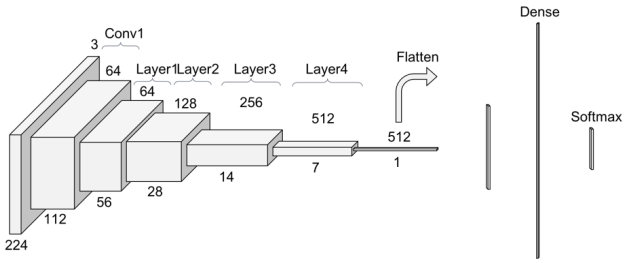

Fig. 3. The ResNet34 architecture

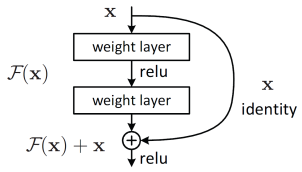

Fig. 4. An example residual block. Our block tries to learn the residual F(x). The true output would be modeled by F(x) + x.

Our experiments utilized the pretrained version of the model that was trained on the ImageNet Dataset. We thought that genre classification on posters was a problem where transfer learning could present a suitable solution, and using the pre-trained network allowed for our model to be more robust. We also slightly edited the architecture for our own purposes. We replaced the final softmax layer with a sigmoid layer and changed the loss function from cross entropy loss to binary cross entropy loss. Both are shown below:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i log(\hat{y_i}) + (1-y_i) log(1-\hat{y_i})$$

C. Custom Architecture

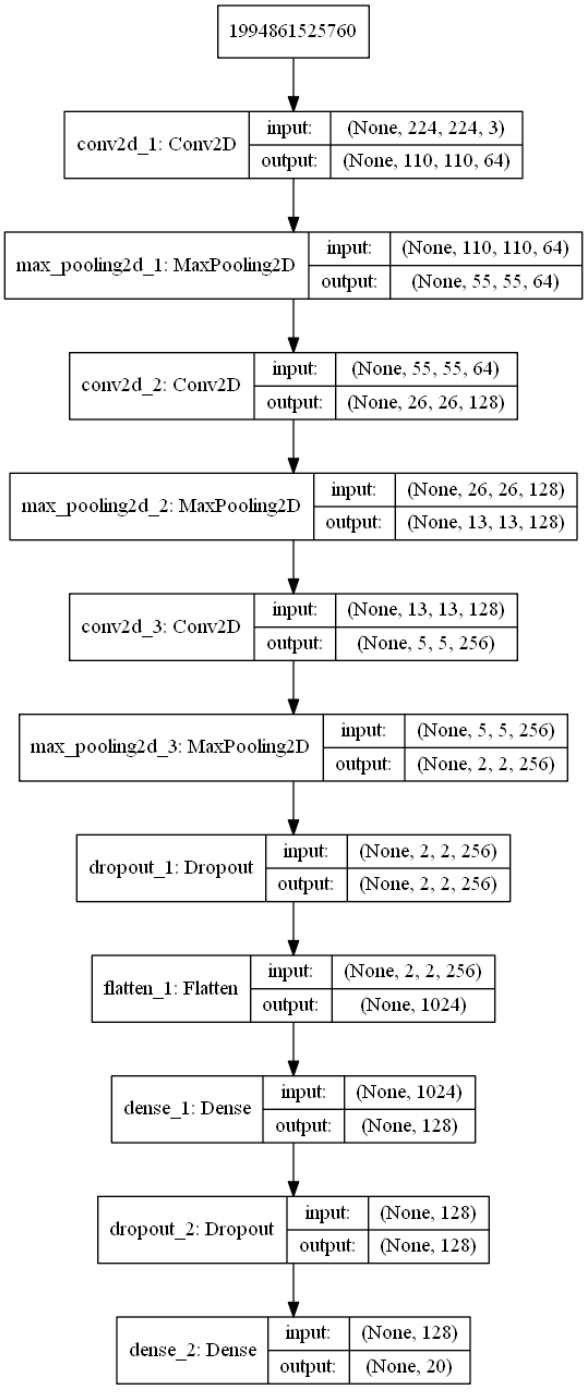

Fig. 5. The custom architecture

We implemented a custom architecture with infrastructure to help us visualize performance. It was created with common practices in multi-class image recognition in mind. First, it employed the usage of 2D Convolutional Layers. We increased the number of filters as the network got deeper to extract increasingly specific features. Moreover, we utilized MaxPool layers in between the convolutional layers since it is common to do so. This reduces the dimensions of our input and the number of parameters and preserves important features while preventing overfitting. We also included Dropout layers, which ignore inputs from specific neurons with a certain probability. This counteracts the co-dependency that neurons develop in deep networks and regularizes our model. Our model uses a Flatten layer, which flattens our feature matrix into a column vector. This allows us to use fully connected layers. Additionally, we use two activation functions. The first is the Rectified Linear Unit function (ReLU) defined as $y = max(0, x)$. We utilize ReLU after the flattening layer and after the convolutions since it is a common practice. The second activation function is the sigmoid (described in the prior section), which is the final output function. As with ResNet, we used binary cross entropy loss. Finally, our model uses an Adam Optimizer, which is the recommended in deep learning models. Adam combines two other techniques of stochastic gradient descent known as RMS Prop and AdaGrad to "compute individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients." [10] It calculates an exponential moving average of the gradient and the squared gradient and then multiplies the moving averages by a decay rate. This has the effect of making our algorithm efficiently reach convergence given lots of data.

Experiments and Results

A. Baseline Algorithm

We implemented a baseline to set a lower limit on the performance of our models. We decided to implement a random baseline, which randomly picks genres for a movie without looking at the poster. This allowed us to quickly create a model to compare our actual algorithms against and provided a good metric for determining if a strategy was worth pursuing - being better than random guessing. We expected this baseline to perform poorly, especially as the number of genres increase. We ran the baseline on a set of 2000 movies and got the following results:

Baseline Performance

One Genre Correct	12.34%

B. ML-KNN Algorithm

We then tested the performance of our ML-KNN model. KNN is computationally taxing and we were unable to run it on our computers with the full dataset. We were also unable to run on some large k-values, with the size of impossible to run k-values varying with dataset size. We maximized the amount of data that we could include in our analysis while also considering a reasonably large k value. But, we were limited by the memory capacity of our computers.

We first ran the algorithms on a subset of the data with 64x64 versions of the poster images in order to establish the viability of this method with a faster runtimes. This 64x64 dataset was composed of a train set of size 1600, a dev set of size 200, and a test set of size 200. We systematically tuned the hyperparameter k by testing values k = 1, 3, 5, 7, 10, 15, 20, 30, 40, 50, 100. Below are the three best performing and worst performing k values by hamming loss.

ML-KNN Performance

K Value	All Match	At least one match	Binary	Hamming Loss
40	7%	44%	0.8875	0.1125
30	8%	44.5%	0.885	0.115
50	6%	42.5%	0.887	0.113
1	5.5%	19.5%	0.852	0.148
3	3.5%	25%	0.8625	0.1375
5	6.5%	28.5%	0.8735	0.1265

Our results led us to believe that k = 40 was optimal for generalization. Using this value, we ran ML-KNN on a larger dataset with 2800 100x150 images in the train set, 350 in the dev set, and 350 in the test set. We chose the 100x150 size for the following KNN experiments in order to maintain the aspect ratio of the posters (2:3), and so we could have more training data. We believed that 100x150 was a suitable replacement for 500x750 because additional testing established that there was a negligible difference between the two resolutions.

ML-KNN Performance

K Value	All Match	At least one match	Binary	Hamming Loss
40	7.77%	34.28%	0.883	0.117

We also implemented the One vs. Many multi-label strategy (OVR), which creates separate binary classifiers for each class (genre). For each classifier, the given class is fitted against the other classes, and the class that gives the maximum score will be predicted. We chose to implement OVR because it is a common form multi-label classification. The dataset for this experiment is the same 100x150 dataset with the 2800/350/350 splits.

OVR Performance

K Value	All Match	At least one match	Binary	Hamming Loss
40	9.714%	35.428%	0.882	0.118

C. ResNet34

We trained the pretrained ResNet34 on a dataset with 28000 images in the training set, 3500 in the dev set, and 3500 in the test set for 50 epochs. Again, we evaluated our models using percentage of predicting at least one genre and completely predicting all genres, test loss, and Hamming loss.

ResNet34 Performance

Accuracy	All Match	At least one match	Test Loss	Hamming Loss
90.62%	12.49%	38.26%	0.2486	0.0938

We also analyzed the movie genres that our model performed particularly well on. The following table presents the 5 genres for which our model had the highest percentage of recall.

Top 5 Class Performances by ResNet34

Genre	Recall	Precision	F1	Count
Animation	0.44	0.84	0.58	135
Comedy	0.40	0.77	0.52	1135
Drama	0.39	0.68	0.50	1558
Horror	0.32	0.52	0.40	406
Family	0.23	0.75	0.35	233

D. Custom Architecture

Finally, we tested our custom architecture. We split up the dataset into the following size: 28000 in the training set, 3500 in the dev set, and 3500 in the test set and trained it for 50 epochs. We measured the following metrics provided by Keras: binary accuracy, categorical accuracy, top k categorical accuracy, and loss.

Custom Architecture Performance

Dataset	All Match	At least one match	Binary	Hamming Loss
Train	7%	44%	0.8875	0.1125
Dev	8%	44.5%	0.885	0.115

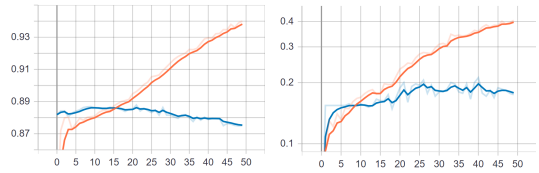

Fig. 6. (Left) Binary accuracy (Right) Categorical Accuracy

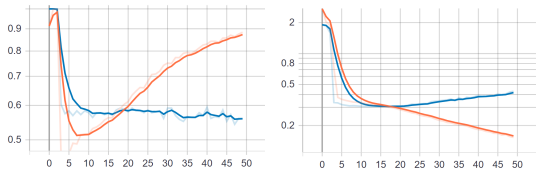

Fig. 7. (Left) Top K Categorical Accuracy (Right) Loss

As with the ResNet model, we found the genres that with which the model was most successful in terms of recall.

Top 5 Class Performances by Custom Model

Genre	Recall	Precision	F1	Count
Drama	0.46	0.48	0.47	1558
Comedy	0.38	0.46	0.41	1135
Thriller	0.17	0.35	0.23	632
Horror	0.10	0.27	0.15	406
Action	0.08	0.26	0.12	526

DISCUSSION

Our ResNet and custom models perform only slightly better than the kNN implementations in pure accuracy. Yet, the inclusion of important evaluation metrics such as recall and all-match allow for future refinement of these models and insight on our model's performance. We can also notice that ResNet outperforms our custom architecture, but not so much as to make the custom architecture insignificant.

To give a direct understanding of precision, recall, and F1 Score we present the following formulas, which will be a refresher for many:

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$F_1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

For us, we believe recall to be the most important of these three metrics for our classification task. This is because our personal goal was to correctly identify as many genres as possible.

Our custom model was reasonably successful at identifying the most likely category for a film. One set of examples were the movies Toy Story (Comedy, Family, Animated) and Jumanji (Fantasy, Adventure, and Family). Our model noted that for Toy Story, the genres 'Family' and 'Animation' had a value above 0.5 when outputted by the sigmoid function, while Jumanji had no such genres. Yet, the model was most certain that Toy Story was an 'Animation' and Jumanji was a 'Comedy'. While the classification of Jumanji was incorrect, it represents a close guess to a potential category as there are some elements of the poster that could suggest that it is a comedy. Moreover, the fact that it did not have a sigmoid value of above 0.5 indicated that the model was not quite sure that this was a correct evaluation. Yet, the classification of Toy Story was correct and had a sigmoid value of above 0.5, which indicated that the model was most certain that Toy Story was an animation. This corresponds with the intuition that the most percievable quality about the poster for the film is that it is animated.

One factor that could have potentially resulted in both the ResNet34 model and our custom model making incorrect prediction may have been due to the distribution of the dataset. The data overwhelmingly included dramas (15941) yet had very few TV movies (655). This could have potentially caused our model to default towards dramas in instances of uncertainty. The extreme imbalances in this dataset seem to have a much greater effect on our custom model, though, as we find that our evaluation metrics are much higher for the highest occurring classes in the test set, while for ResNet we can witness that Animation evaluates the best of any genre. We also believe that the large number of classes makes it very ambitious to expect highly effective results when paired with the imbalanced dataset.

There is room for improvement on overall evaluation of these models, as we do not have an established metric by which we would want our model to be the most successful at. And due to this indecision as to the best evaluation metric, we believe that the metrics that one would like in this specific multi-label classification task is based on preference. We believe that top k categorical accuracy is a very good general purpose descriptor of the effectiveness of the model, because it gives the ability to determine the top k related genres to a poster. There is much that could be improved upon in terms of evaluation clarity, as some models we present here do not have the metrics presented for the others.

We did not present averaged precision, recall, and F1 scores for the ResNet and custom models for the sake of space and demonstrating the potential effectiveness on a genre given proper sample size.

Future Work

One possible improvement is training our model using more balanced data. Since the dataset that was utilized is imbalanced, we could augment our dataset to expose our model to more examples to make the model more robust. We could augment the dataset to make it more balanced by oversampling minority classes, undersampling majority classes, or a mixture of the two. Also, due to limited memory we had to limit the number of datapoints we could consider for KNN as well as the number of possible neighbors to analyze. Therefore, with more computational resources, we could go more in depth in analyzing the performance of KNN, on top of being able to feasibly train much more extensive architectures. Finally, we used features on posters to make predictions about a movie's genre. However, movie posters have more potential information about a film than just genre. Given more time, we would attempt to predict other movie traits from posters such as ratings. Ideally, we would train a single model to output predictions for all the relevant features that we want to identify.

Conclusion

We found that the ResNet network and the Custom Architecture, despite performing only slightly better in pure accuracy when compared to ML-kNN, performed well in other important evaluation metrics such as F1 Score, Recall, and Top K Categorical Accuracy. In particular, our models did well on specific genres such as Comedy. This indicates the potential for improved performance given more computational resources and a more balanced dataset. Our model also shows that across posters, there seems to be a consistent set of features that occurs within those of a specific genre.

Acknowledgement

We would like to thank Chris Ré, Tengyu Ma, and the rest of the CS 229 staff for their support throughout the quarter.

Group Member Contribution

Gabe preprocessed the dataset and implemented the ResNet and KNN algorithms. Kris implemented the random baseline, the custom architecture, and formatted the write-up. Both partners worked on the poster together. A GitHub repository of our code can be found at this URL: https://github.com/barneyga/229_pythonfiles

REFERENCES

[1] Zhang, Min-Ling and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition 40 (2007): 2038-2048.

[2] Piotr Szymanski, Tomasz Kajdanowicz. A scikit-based Python environment for performing multi-label classification. arXiv:1702.01460.

[3] Kaggle. The Movies Dataset, 2017. https://www.kaggle.com/rounakbanik/the-movies-dataset/kernels

[4] Hoang, Q. (2018). Predicting movie genres based on plot summaries. arXiv preprint arXiv:1801.04813.

[5] Eric Makita and Artem Lenskiy. A multinomial probabilistic model for movie genre predictions. arXiv preprint arXiv:1603.07849, 2016.

[6] Jonatas Wehrmann and Rodrigo C. Barros. 2017. Convolutions through Time for Multi-label Movie Genre Classification. In Proceedings of the Symposium on Applied Computing. 114119.

[7] Marina Ivasic-Kos, Miran Pobar, and Luka Mikec. 2014. Movie Posters Classification into Genres based on Low-level Features. In Proceedings of International Convention on Information and Communication Technology, Electronics and Microelectronics.

[8] Wei-Ta Chu and Hung-Jui Guo. 2017. Movie Genre Classification based on Poster Images with Deep Neural Networks. In Proceedings of MUSA217, Mountain View, CA, USA, October 27, 2017, 7 pages. DOI: 10.1145/3132515.3132516

[9] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2016.90

[10] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2016.90