Final Report: Project CS 337

Vatsal Goyal, Kanishka Mittal, Khyati Patel, Sanyam Saxena

Autumn 2022

Contents

1 Introduction

A movie poster image is one of the important media in the filmmaking process, providing valuable information about the movie, such as movie titles, characters, and genres. Identifying a movie genre from a poster can be a daunting task, as it can relate to multiple genres. In the film-making process, a movie poster is quite important. It's more than a mere visual; it's a well-thought-out distribution and promotional tool intended to convey the film's overall message. Furthermore, the poster provides vital information about the film by utilizing rich visual aspects such as the title, lead characters (actor/actress), plot, genre, and others. The posters are created using a set of predetermined guidelines for layout, color, typeface, and composition. The guidelines may vary depending on the genre and audience. A movie title, for example, might convey genre information based on the color it is composed of; romantic movie titles use light blue and pink colors.

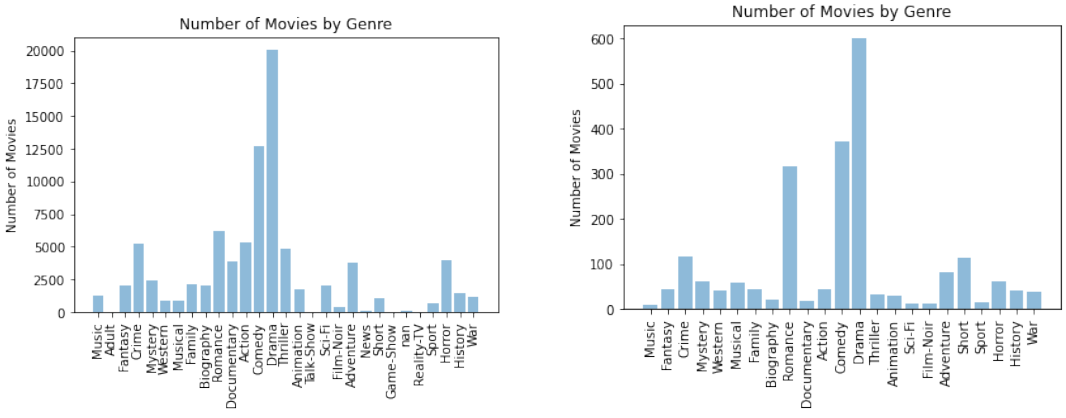
2 Dataset Analysis

Following is the link to the raw dataset Kaggle Movie Poster Dataset
No. of data-points - 41979
Labels - 28 genres
(Action, Adult, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, Film-Noir, Game-Show, History, Horror, Music, Musical, Mystery, News, Reality-TV, Romance, Sci-Fi, Short, Sport, Talk-Show, Thriller, War, Western)

2.1 Smaller Dataset from humongous Data

We had a dataset of 41K poster images of size near about 200*100*3. However handling such large data wouldn't be possible for us. Hence, we used a smaller extracted version of this dataset with 1k points with properties similar to the superset.
We also curated an experimental dataset of 7k points again having a similar distribution as that of the superset. Results were similar between the 1k dataet and the 7k dataset however time taken naturally increased.
We have included the results of only the 1k dataset.

2.2 Similarity between the super Dataset and the used Dataset



(a) 41K Dataset

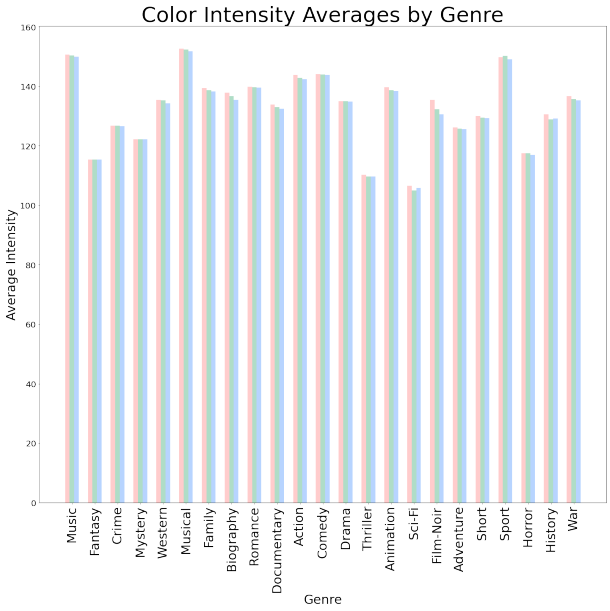
(b) 1K Dataset

- Drama comes out to be most frequent followed by comedy in the dataset with 41k points

- The distribution of the smaller dataset resembles the one of the larger dataset
- The most frequent genre is Drama followed by Comedy and Romance.

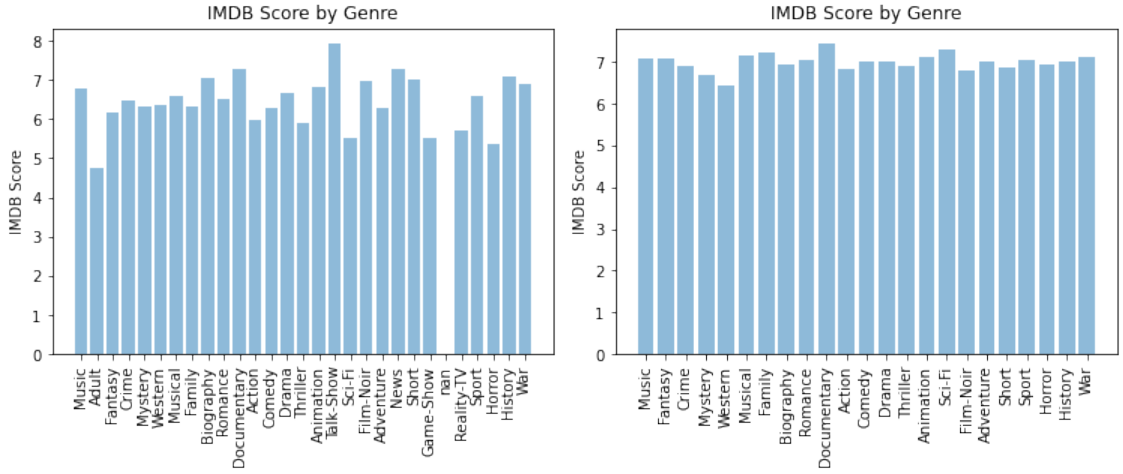
2.3 Relation between Genre and Poster

- One way to see the relationship between genre and poster is to see the variation in color schemes of the posters with genres
- Genres like Sci-Fi, Thriller, Horror and Crime have low averages of rgb values indicating posters inclined towards darker shades.
- On the other hand genres like Musical, Sport, Animation, Comedy have high rgb averages indicating light shaded or bright posters



2.4 Indifference in IMDb rating with genre variation

We can see that movie rating are somewhat indifferent to genre as there is very little variation in rating averages across genres



(a) 41K Dataset (b) 1K Dataset

3 Preprocessing

3.1 Cleaning of Data

As mentioned earlier we had 2 datasets: 1k images(already extracted) and 7k images dataset which we extracted form the 41k dataset.

The CSV file used contained close to 41 thousand movie data. It had discrepancies like empty rows, nan values, duplicate titles, and corrupted links. In both datasets, a few files were corrupt and could not open. Thus it was necessary to clean the data before performing any experiments on it.

Steps performed to do so:

- **Cleaning CSV**: We dropped the nan values and empty rows from the CSV file and removed duplicate values.
- **Removal of corrupt or bad images**: We opened each image to check if it was opening and loadable or not. We made a list of IDs and paths of images that were loadable.
- **Modified CSV**: Finally we trimmed CSV to keep only the rows of movies whose poster was available and reduced the number of columns to: 'imdbId', 'Genre', 'Title', 'Image_Paths'. We saved this file as MovieGenre_final.csv.

3.2 Multi-Hot Encoding

The class labels (i.e the genres) are categorical in nature and have to be converted into numerical form before classification is performed. One-hot encoding is adopted, which converts categorical labels into a vector of binary values. 28 unique genres are found and each genre is represented as a one-hot encoded column. If a movie belongs to a genre, the value is 1("hot"), else 0. As an image can belong to multiple genres, here it is a case of multiple-hot encoding (as multiple genre values can be "hot"). After transformation, the encoded labels look like this:

	Img-paths	Action	Adult	Adventure	Animation	Biography	Comedy	Crime	Documentary	Drama	...	Reality-TV	Romance	Sci-Fi	Short	Sport
0	Posters/27428.jpg	0	0	0	0	0	0	1	0	1	...	0	0	0	0	0
1	Posters/972555.jpg	0	0	0	0	0	0	1	0	1	...	0	0	0	0	0
2	Posters/1424062.jpg	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0
3	Posters/1185371.jpg	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0
4	Posters/113369.jpg	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0

4 Models

4.1 Optimizer, Loss Function and Epochs

1. **Optimizer**:
   We have used Adam optimiser. The results of the Adam optimizer are generally better than every other optimization algorithms, have faster computation time, and require fewer parameters for tuning. Because of all these qualities we used Adam.

2. **Loss Function**:

   Categorical cross-entropy is based on the assumption that only 1 class is correct out of all possible ones (the target should be [0,0,0,0,1,0] if the 5 class) while binary-cross-entropy works on each individual output separately implying that each case can belong to multiple classes(Multi-label). Output like [0,1,0,1,0,1] is a valid one if you are using binary-cross-entropy but not in categorical cross entropy.

   Since the task is of genre prediction (a classification task), then the categorical cross entropy loss function appears to be the best loss function but our task is a bit different. **In the training data each movie can belong to multiple genres, thus categorical cross-entropy loss failed to work here. That is when using one-hot encoding, per row more than 1 entry had a 1.**

   Instead, we used the **binary cross entropy loss**.

3. **Early Stopping**: We have kept the epochs in the models 100 but have implemented early stopping criteria with patience 10.

4.2 CNN Model 1:- Custom Architecture

4.2.1 HyperParameters

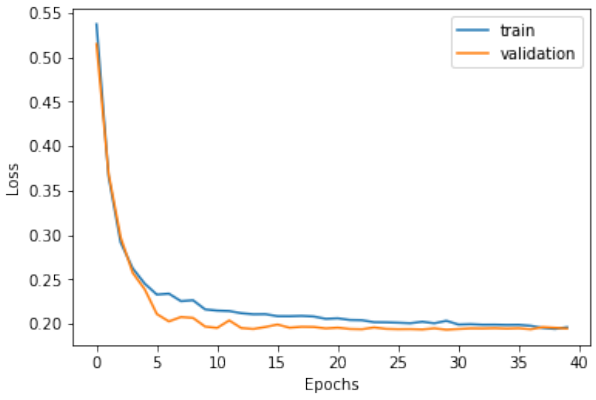
Epochs - 100
Batch Size - 32

4.2.2 Model

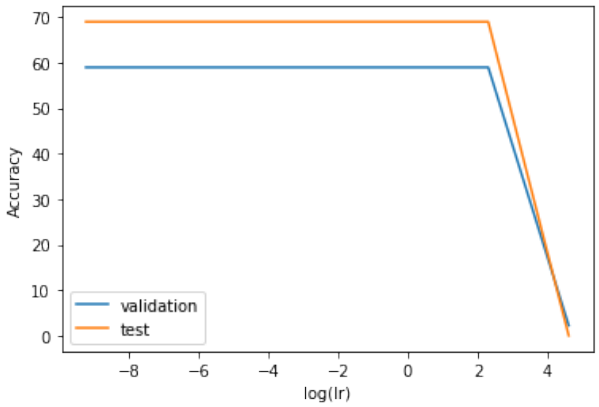
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 196, 146, 16)      1216

 max_pooling2d (MaxPooling2D  (None, 98, 73, 16)       0
 )

 dropout (Dropout)           (None, 98, 73, 16)        0

 conv2d_1 (Conv2D)           (None, 94, 69, 32)        12832
```

```
 max_pooling2d_1 (MaxPooling  (None, 47, 34, 32)       0
 2D)

 dropout_1 (Dropout)         (None, 47, 34, 32)        0

 conv2d_2 (Conv2D)           (None, 43, 30, 64)        51264

 max_pooling2d_2 (MaxPooling  (None, 21, 15, 64)       0
 2D)

 dropout_2 (Dropout)         (None, 21, 15, 64)        0

 conv2d_3 (Conv2D)           (None, 17, 11, 64)        102464

 max_pooling2d_3 (MaxPooling  (None, 8, 5, 64)         0
 2D)

 dropout_3 (Dropout)         (None, 8, 5, 64)          0

 flatten (Flatten)           (None, 2560)              0

 dense (Dense)               (None, 128)               327808

 dropout_4 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_5 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 28)                1820

=================================================================
Total params: 505,660
Trainable params: 505,660
Non-trainable params: 0
_________________________________________________________________
```

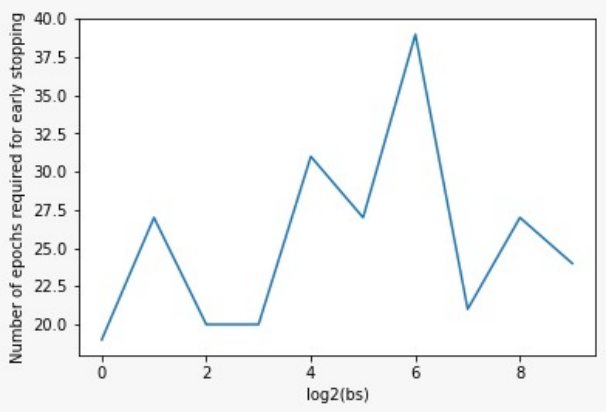
4.2.3 Learning Curve



4.2.4 Variation in accuracy with learning rate



4.2.5 Variation in number of epochs required vs batch size



4.2.6 Results

- **Split - 0.7-0.27-0.03**(1k images)
  Validation Accuracy- 58.95522388059702 %
  Test Accuracy- 68.96551724137932 %
- **Split - 0.7-0.15-0.15**(1k images)
  Validation Accuracy- 62.16216216216216 %
  Test Accuracy- 57.71812080536913 %
- **Split - 0.8-0.1-0.1**(1k images)
  Validation Accuracy- 61.224489795918366 %
  Test Accuracy- 66.66666666666666 %
- **Split - 0.7-0.15-0.15**(7k images)
  Validation Accuracy- 61.224489795918366 %
  Test Accuracy- 66.66666666666666 %

4.3 CNN Model 2:- Fine-tuning pre-trained VGG16

In this model, we tried **Feature extraction** using a pre-trained VGG16 model. We just added a flatten layer, and 2 dense layers which have a dropout in between. The weights learnt by the VGG parameters are not changed during further training. Only the last 2 layers are trained.

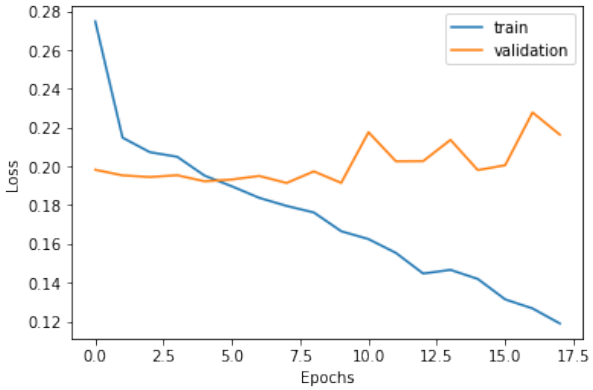
4.3.1 HyperParameters

Epochs - 50(early stopping)
Batch Size - 32

4.3.2 Model

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 6, 4, 512)         14714688

 flatten_3 (Flatten)         (None, 12288)             0

 dense_7 (Dense)             (None, 1024)              12583936

 dropout_6 (Dropout)         (None, 1024)              0

 dense_8 (Dense)             (None, 28)                28700

=================================================================
Total params: 27,327,324
Trainable params: 19,692,060
Non-trainable params: 7,635,264
_________________________________________________________________
```

4.3.3 Learning Curve



4.3.4 Results

Split 0.7-0.27-0.03
Validation Accuracy- 54.850746268656714 %
Test Accuracy- 62.06896551724138 %

Split 0.7-0.15-0.15
Validation Accuracy- 50.0 %
Test Accuracy- 51.67785234899329 %

Split 0.8-0.1-0.1
Validation Accuracy- 54.08163265306123 %
Test Accuracy- 55.55555555555556 %

4.4 CNN Model 3:- Fine-tuning pre-trained ResNet50

In this model, we tried Feature extraction using a pre-trained ResNet50 model. We just added a flatten layer, and 2 dense layers which have a Relu activation in between. Resnet50 parameters are not changed during further training. Only the last 2 layers are trained.

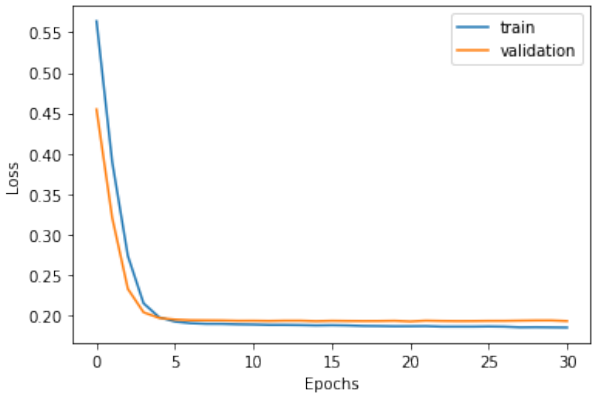
4.4.1 HyperParameters

Epochs - 100(early stopping)
Batch Size - 32

4.4.2 Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 32)                65568

 dense_1 (Dense)             (None, 28)                924

=================================================================
Total params: 23,654,204
Trainable params: 66,492
Non-trainable params: 23,587,712
_________________________________________________________________
```

4.4.3 Learning Curve



4.4.4 Results

- **Split 0.7-0.27-0.03**
  Validation Accuracy- 57.46268656716418 %
  Test Accuracy- 65.51724137931035 %
- **Split 0.7-0.15-0.15**
  Validation Accuracy- 62.16216216216216 %
  Test Accuracy- 55.033557046979865 %
- **Split 0.8-0.1-0.1**
  Validation Accuracy- 60.204081632653065 %
  Test Accuracy- 67.67676767676768 %

4.5 CNN Model 4:- Fine-tuning pre-trained ResNet2

In this model, we tried Feature extraction using a pre-trained ResNet50 model. We just added a flatten layer, and 2 dense layers which have a Relu activation in between. Resnet50 parameters can also be changed during further training along with last 2 layers.

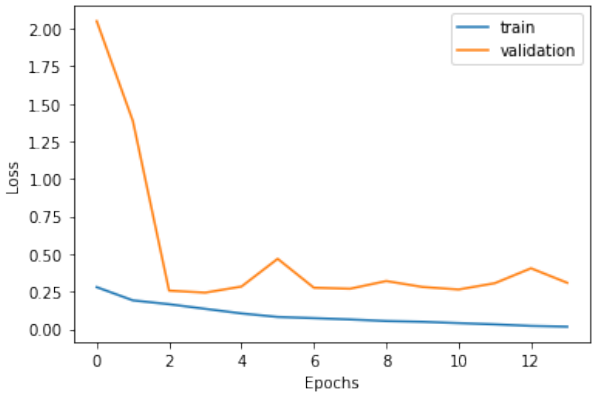
4.5.1 HyperParameters

Epochs - 100(early stopping)
Batch Size - 32

4.5.2 Model

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten_2 (Flatten)         (None, 2048)              0

 dense_5 (Dense)             (None, 32)                65568

 dense_6 (Dense)             (None, 28)                924

=================================================================
Total params: 23,654,204
Trainable params: 23,601,084
Non-trainable params: 53,120
_________________________________________________________________
```

4.5.3 Learning Curve

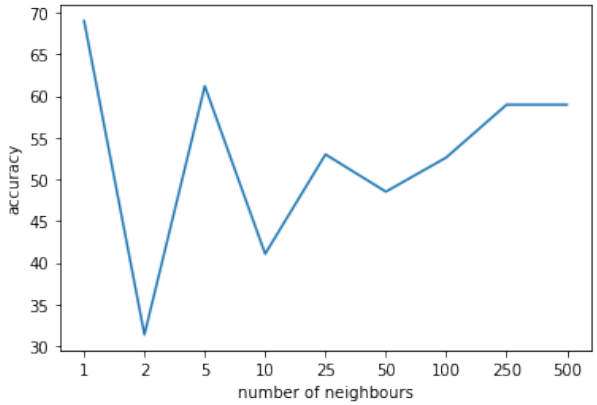


4.5.4 Results

- **Split 0.7-0.27-0.03**
  Validation Accuracy- 48.88059701492538 %
  Test Accuracy- 48.275862068965516 %
- **Split 0.7-0.27-0.03**
  Validation Accuracy- 10.135135135135135 %
  Test Accuracy- 12.751677852348994 %
- **Split 0.7-0.27-0.03**
  Validation Accuracy- 17.346938775510203 %
  Test Accuracy- 14.14141414141414 %

4.6 KNN Model

4.6.1 Variation in Accuracy with varying number of neighbours



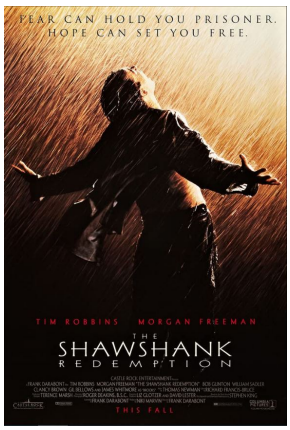
5 Can a movie be judged by its Poster?

As a further extension of this project we tried to predict the imDb rating of the movie using its poster.

5.1 Examples:

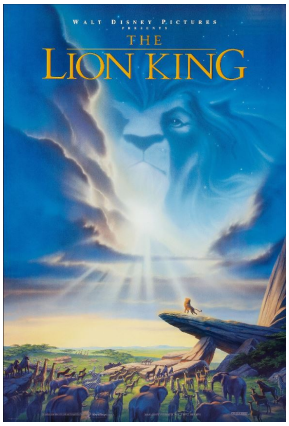
Movie posters and its reviews/ratings seem relatively unrelated as "you can't judge a book by its cover". Below we see two of the movie with highest IMDb ratings, The Shawshank Redemption (9.2) and 12 Angry Men (9). However, one can easily observe high difference between their posters.

(a) 12 Angry Men (9)

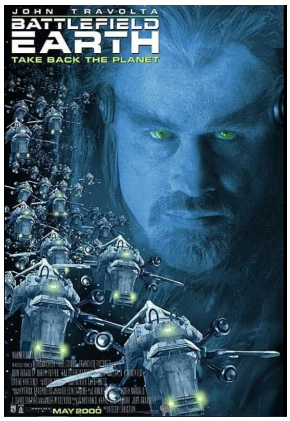


(b) The Shawshank Redemption (9.2)

Now, below we see other two movies, The Lion King (8.5) and Battlefield Earth (2.6). One can see the similarity between the posters again implying no relation between poster and rating.



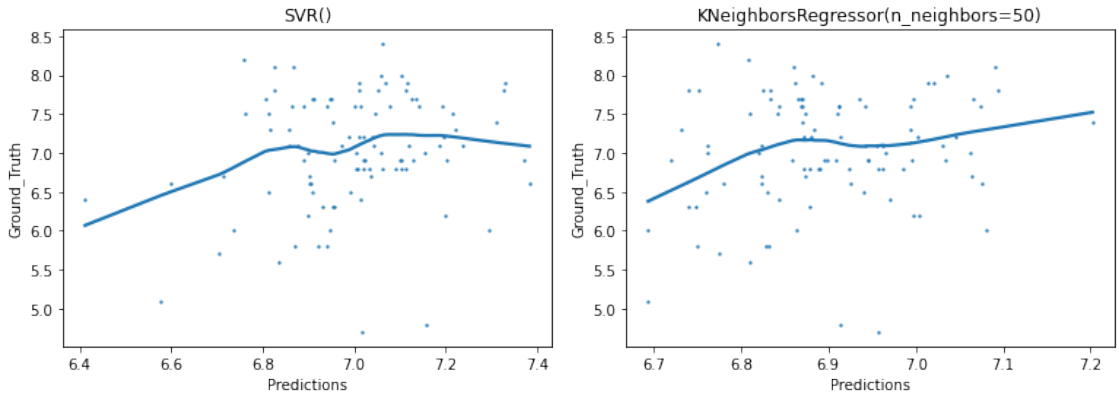
(a) The Lion King (8.5)



(b) Battlefield Earth (2.6)

5.2 Attempts

We attempted to use Regression Models for predicting movie rating by its poster. We used SVM and KNNeighbours Regression and both seemed to perform similar. We got an MSE of about 0.5 which is decent. However plotting graphs for prediction vs ground truth seem to have significant issues. We would have liked a line similar to y=x however the regression fit curve of our results look horizontal.



6 Can we predict if a movie will be successful by its poster?

Posters are a great marketing tool for a movie, people most often just look at a movie's poster and trailer before going. Here we try to predict if we can guess whether a movie will be successful or not judging by its poster. Can a model identify if a particular type of poster tends to do better in the box office? Success can be seen as whether the movie is profitable or not, if the revenue is higher than the budget, then we can say the movie is successful, else it is a failure at the box office

In our experiment, we define how well a movie is doing by its revenue to budget ratio. A higher budget movie would require higher revenue to break even. The higher the ratio, the better the movie is doing.

6.1 Dataset

1. The dataset that has been used has been curated by merging two datasets(IMDB and TMDB) of 45000 points each, one containing information about movie's revenue and budget, and another containing link to movie posters.
2. After cleaning the dataset(removing null entries), It contains 5000 data points. The points are split into two classes, depending on whether the revenue to budget ratio is greater or smaller than 1.
3. PyTorch DataLoader is used to load the images and split into train and val sets

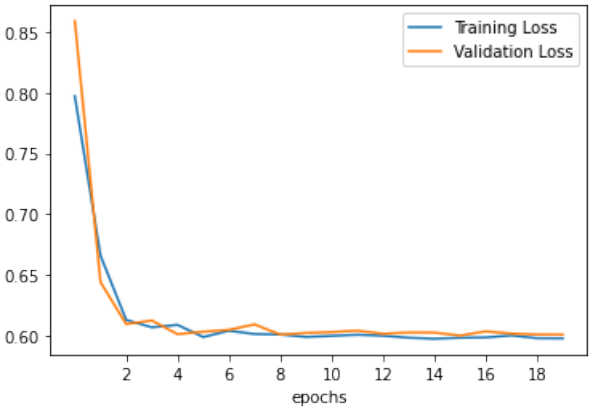
6.2 Model

1. We are fine-tuning a pretrained Resnet18 model available from PyTorch with Cross Entropy Loss
2. Added a linear layer at the top of Resnet model to get the final output of required size, then tuned all layers. There are 62 layers with parameters to learn

6.3 Result

Validation Accuracy : 0.711

6.4 Learning Graph



7 Difficulties faced

- We first tried to train our model on the super dataset with 40,000 data-points, but were unable to train the model on these many points (after train-val-test split) due to limitation of GPU and RAM on google colab, hence we had to stick to the dataset with 1000 data-points.
- Generally classification tasks are single labelled while this was a multi-labelled classification problem. Earlier we had used categorical cross-entropy loss function but we didn't get the expected results hence we had to switch to binary cross-entropy loss function.

8 Conclusion

The best possible accuracy which could be obtained was 68%. Which leads us to think, what could be causing all the misclassification? How could we further improve its accuracy? What we realised is that the model is having difficulty differentiating between horror and thriller posters. If you think about it, it is true even for us humans, where we might not be able to tell the difference between horror and thriller posters.

The same result is observed for comedy and romance, as both genres' posters are in the lighter mood, and contains human and smiling faces.

9 References

https://drive.google.com/drive/folders/1cXPMqGUR9GrVAwS6pGZY1oiucAhrsndw

https://www.kaggle.com/datasets

https://www.kaggle.com/code/jmurthy/simple-eda/notebook

https://github.com/d-misra/Multi-label-movie-poster-genre-classification

https://towardsdatascience.com/predict-movie-earnings-with-posters-786e9fd82bdc

https://gist.github.com/jinglescode/60b496abfd87b2093da9cf55bde2e944

https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/: :text=The%20

https://androidkt.com/choose-cross-entropy-loss-function-in-keras/: :text=Categorical%20cross%2Dentropy%20is%20