Movie Genre Classification From Poster Images

Team Members: Gerardo Zavala-gzavala, Dileep Badveli, Alex McKinnon

I. Methodology

A. DataSet and Preprocessing

Current dataset contains 39,515 JPEG images, each having 182x268 dimension. Going through all the posters we found that there was a great imbalance between different genres, "Fig1" shows the distribution of movies per genre for top 15 genres.

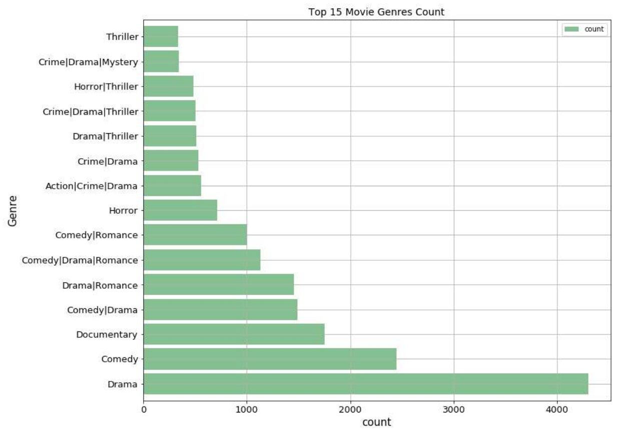


Figure 1: Posters available for the top 15 popular genres

The next step was to split movies genres with more than 1 genre, and select top 6 genres counts, to speed up training and being able to have at least 1000 classes per genre, " Figure 2" shows the distribution of genres it can be seen the imbalance loaded for Drama and Comedy, due to this we use a random sampling approach to try to make a more distributed population, " Figure 3."

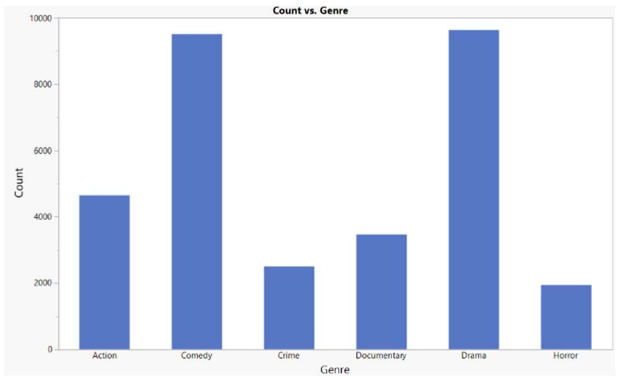


Figure 2: Top 6 genres selected for classification

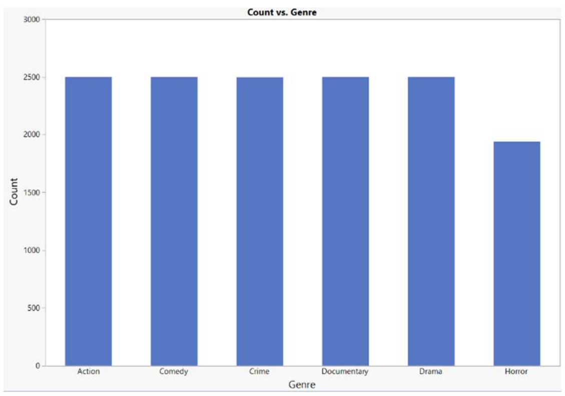


Figure 3:Top 6 genres selected for classification after balancing the samples of each genre

These images are fed into the models using 3 channels, so the color of the image is a variable in defining the genre of the poster. The images were first normalized so that their pixel values lie in the range of 0 to 1 and then used in 2 different models, for the purpose of finding which one is better at classifying for our application.

For training the models we created 3 sets, training = 60%, validation = 20% and testing = 20%, with stratify distribution and the genres encoded as can be seen in "Figure 4".

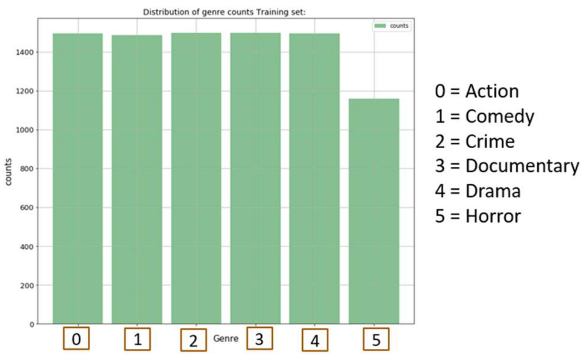


Figure 4: Encoding of each genre as integer. This allows the expected values be fed into model

B. Transfer Learning with MobileNetV2

MobileNetV2 is a neural network architecture primarily developed to run efficiently on mobile devices. One of the most expensive operations to perform is a traditional convolution in neural networks, so it was replaced with a Depth wise Separable Convolution. This sort of convolution increases speed of computation as well and decreases the model size by trying to approximate the standard convolution. We used this model for the purpose of transfer learning.

Transfer Learning is the process of using already trained models, such as Mobile NetV2, to train a specific application by adding extra layers on top of it. This process speeds up training as well helps the newer model to predict more accurately. We have added a global max pooling layer and a dense layer with a SoftMax activation function to categorize the images into 6 different genres.

C. Convolutional Neural Network (CNN)

Convolutional neural networks are commonly used in deep learning to train Networks for image classification, and object detection.

In this type of networks the convolution layers act like feature extractors, they can detect shapes, figures etc.. They can learn representations of the images used for training the network. (Waseem Rawat, Zenghui Wang, Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review).

We will make use of random search and a sequential CNN, for hyperparameter tuning, the hyperparameters that we tried to optimize were, learning rate, dropout and convolution layer filter size.

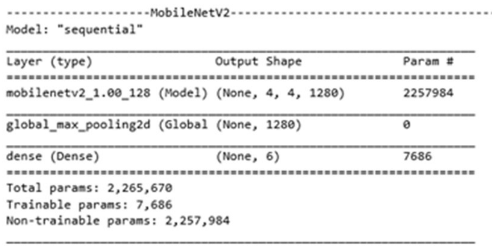
After these runs, we will select the best parameters and try to optimize the models. If we are not able to beat the transfer learning, we will look for another approach.

II. Model Training and Hyperparameter Selection

A. Transfer Learning Hyperparameter Tuning

Mobile NetV2 model was used as a baseline and then tried to improve the accuracy with a CNN. For the first 3 trials of transfer learning, all the weights will be frozen, last layer was changed to 6 nodes with activation softmax, and will be tested 3 different learning rates [0.0001, 0.001, 0.01], model summary for first 3 runs is shown in “Figure5”.

In the last try we will unfreeze the weights so it will be able to train more parameters and increase the epochs for fine tuning reducing best previous learning rate trying to achieve maximum performance of network.



```
-------------------MobileNetV2----------------------------------
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_128 (Model) (None, 4, 4, 1280)        2257984
_________________________________________________________________
global_max_pooling2d (Global (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 6)                 7686
=================================================================
Total params: 2,265,670
Trainable params: 7,686
Non-trainable params: 2,257,984
_________________________________________________________________
```

Figure 5:Transfer learning first 3 runs

Table 1: Transfer learning hyperparameter tuning. Table is shows the 3 runs for the 3 different learning rates for transfer learning.

Run	Learning Rate	Batch size	Steps per epoch	Validation Steps
1	0.0001	32	449	20
2	0.001	32	449	20
2	0.01	32	449	20

After the first 3 runs the best performance was found to be for run 1, with learning rate 0.0001, the plot for the results for 20 epochs can be seen in “Figure 6”.

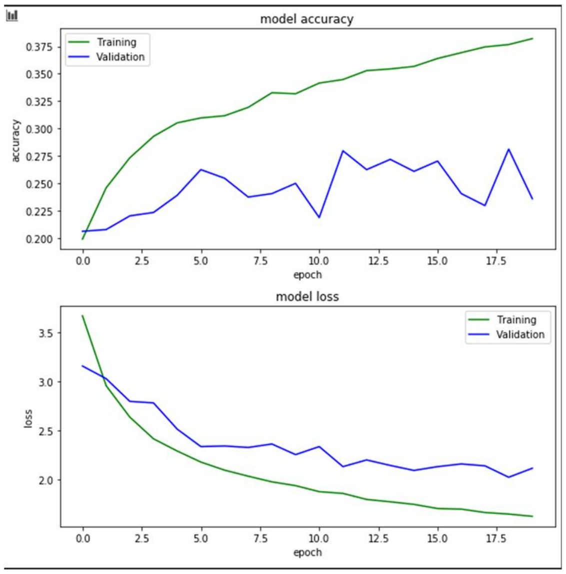


Figure 6:Transfer learning best model

Learning rates above >0.001 has been tested, and we decided they were not good enough to compete, since accuracy was displayed inconsistently. After looking at these graphs we decided to go with the learning rate of 0.0001. Even though it is slower to increment than the higher learning rate, it will smoothly raise to a reasonable accuracy, instead of having a wide range of tolerances for the accuracy.

We will use run 1, lr = 0.0001 for final transfer learning model, results will be shown next section “Evaluation”.

B. CNN Hyperparameter Tuning

The learning rate was a random uniform value 10^x where x is a random uniform value from [-4,-2), in the case of dropout it was a random 2 decimal value between .2 and .5, for the first case we used an architecture shown in “Table 2”, for the next run we used the same architecture with batch normalization after each convolutional layer and dense layer.

We run 10 random searches for both cases and evaluate the performance for a 4-fold cross validation for 15 epochs of training on the CNN.

Table 2:CNN Architectural Model

Image input	Input shape (150, 101, 3)

2D Convolutional layer 1	Filter = X
	Activation = Relu
	Kernel = (3 x 3)

2D Max Pooling	Kernel size = (2 x 2)

2D Convolutional layer 2	Filter = X
	Activation = Relu
	Kernel = (3 x 3)

2D Max Pooling	Kernel size = (2 x 2)
Flatten	
Dense	Nodes =X
Dropout	Rate = X

Dense	Activation = Softmax
	Output = 6

III. EVALUATION

A. Transfer Learning Final Results

In this final run we increase the number for 20 more epochs, figure X shows that once we unfreeze weights there were approximately 2 million parameters to train, and we used a smaller learning rate = 1x10(-5).

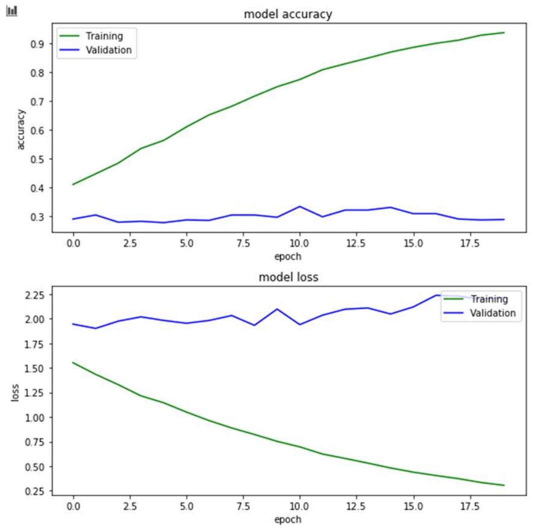


Figure 7: Transfer learning fine tuning

The results show a great improvement in training accuracy, but validation was not behaving as expected since it seems that validation was not increasing, instead only remained constant, “Figure7” show the final transfer learning results.

Since fine tuning for transfer learning shows an expected accuracy in validation of ~30%, we generated the predictions for the testing set to see how it behaves.

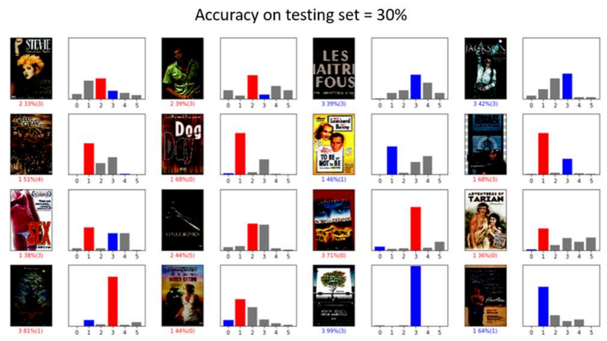


Figure 8: Predictions made with posters in 1 batch fine tuning

As expected, the model accuracy on the testing set is of 30%, this will be our baseline that we will try to improve for CNN.

B. CNN Random Searches Final Results

A validation of k fold for 15 epochs have been performed using the first architecture. The results for the different hyperparameter set are shown in Figure 9. The results have lower accuracy than transfer learning model. From the results learning rate was what increased the accuracy from set1 to the other remaining sets.

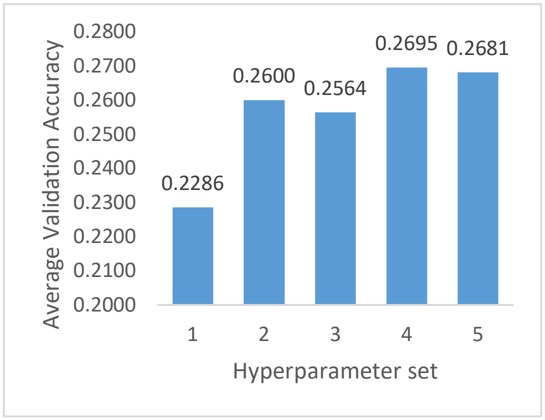


Figure 9: Accuracy of validation random search 1

Table 3: hyper paramters found in random search 1

Set	Learning Rate	Dropout Rate	Width 1	Width 2	Dwidth
1	0.00180133	0.41	10	30	106
2	0.00037912	0.34	8	16	92
3	0.00019251	0.24	4	24	84
4	0.00013504	0.31	9	31	115
5	0.00035341	0.39	12	31	81

The results for random search with batch normalization show similar results as without batch normalization.

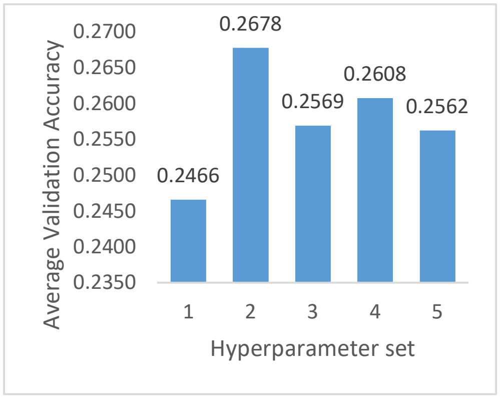


Figure 10: Accuracy of validation random search 2

Table 4: Hyperparameters found in random search 2

Set	Learning Rate	Dropout Rate	Width 1	Width 2	Dwidth
1	0.00088016	0.37	12	17	115
2	0.00021302	0.3	10	24	80
3	0.00113867	0.38	8	18	84
4	0.00306337	0.42	9	25	85
5	0.00024632	0.33	9	18	89

C. New Approach for labels

Due to the low accuracy of model one, a final trial was performed. We changed the approach and used a one hot encoding as shown in "Table 5", everything else remained the same. We took only top 6 genres and a distributed sample.

Table 5: Example of One hot encoding based on genres

Movie	Action	Crime	Documentary	Drama	Horror	Comedy
Movie 1	0	1	0	0	0	0
Movie 2	0	0	0	0	1	0
Movie 3	0	0	1	0	0	0

After changing from Integer Encoding to One hot encoding, we developed a CNN model to test the change on. "Table 6" shows the structure of this model.

Table 6: CNN Model for one hot encoding

Image input	Input shape (150, 101, 3)

2D Convolutional layer 1	Filter = 32
	Activation = Relu
	Kernel = (3 x 3)

2D Convolutional layer 2	Filter = 64
	Activation = Relu
	Kernel = (3 x 3)

2D Max Pooling	Kernel size = (2 x 2)

2D Convolutional layer 3	Filter = 128
	Activation = Relu
	Kernel = (3 x 3)

2D Convolutional layer 4	Filter = 64
	Activation = Relu
	Kernel = (3 x 3)

2D Max Pooling	Kernel size = (2 x 2)
Dropout	Rate = 0.25
Flatten	
Dense	Nodes =128
Dropout	Rate = 0.5

Dense	Activation = Sigmoid
	Output = 6

The first runs of the model show validation accuracy higher than training, this could be happening due to heavy dropout. We used heavy dropout, which is a regularization technique. Dropout disables neurons to learn during some iterations, some information is lost, and subsequent layers are trying to predict and construct on these incomplete representations. Since in the training we removed some neurons, in validation all neurons are available and that might be why now it has more information to predict, so it is behaving better than training.

Another possible situation is that the validation sample was not representative, or it could have been an easier sample to predict.

The following results show first trials learning rate =0.0001

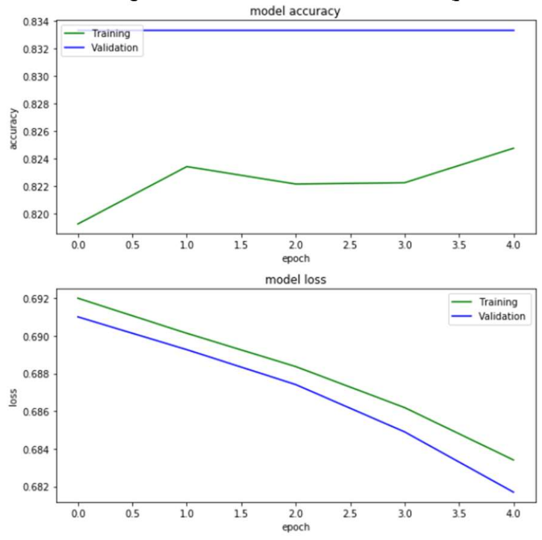


Figure 11: Results from running 4 epochs of Table 6 CNN Model

Another run was performed with the same architecture but removing first layer of dropout and setting learning rate = 0.0001

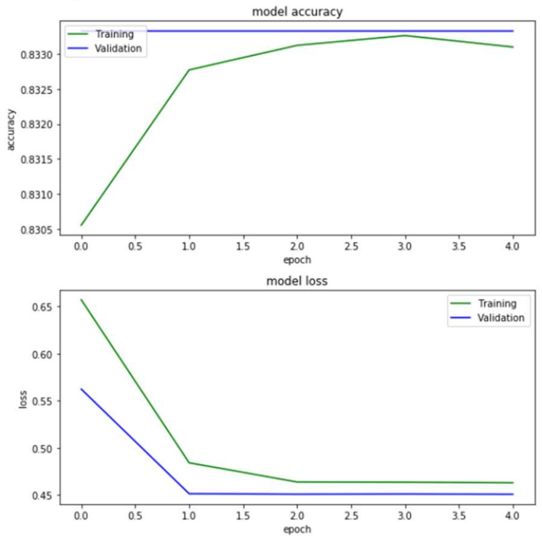
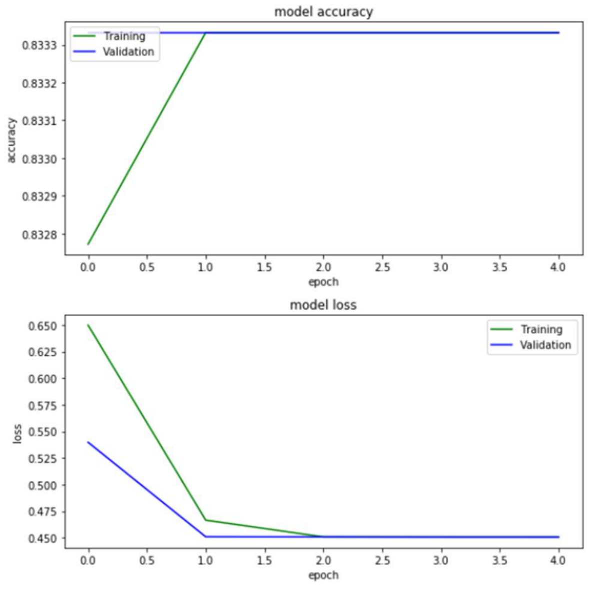


Figure 12:Removed Dropout=0.25 and set learning rate=0.0001

Another run was performed but this time all dropout layers were removed and changed learning rate to "0.001". The results look better but still it looks like training accuracy is stuck, that could be due to low epochs or maybe it was stuck in a local minima, due to time complexity it was not tested for more epochs.



The results in the testing set shown an accuracy of 72.24%, which is better that all previous models.

Conclusion

The preprocess step we have been using Integer encoding, to convert the categorical data to a numerical format for the model to understand. The accuracy has increased tremendously for the validation sets after moving from Integer encoding to One hot encoding. One of the possible reasons for this circumstance is that Integer encoding is enough when dealing with ordinal variables. Our variables were not ordinal, so the model trained incorrectly.

Dropout as shown not to be helpful in constructing the correct behavior for the correct model. With the Data collected we can say that CNN has given us a **72.24%** accuracy rate. It is possible to predict movie genre from posters.

Some improvements to this project would be to add more epochs with a more powerful computer and test grid search for new learning rates. As well as, add more images to the data, also create more robust data, by using data augmentation.

[1] https://www.kaggle.com/neha1703/movie-genre-from-its-poster/download

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon,