# Predicting Genre from Movie Posters

*Duaa Alsaaideh*

*Boeing's Data Science and Career Readiness Program*

***Abstract*—Posters represent one way in which people use media to influence human behavior. Like when we come across a movie poster, we try to quickly figure what kind of movie is, just by trying to look at the colors, expressions, and objects in the movie poster. In this paper I aim to build a 7-layer convolutional neural network that can explain the visual appearance of a movie poster to classify it into one or more genres. Each movie can be classified into multiple genres, so this is a multi-label classification problem. I choose a dataset collected from the IMDB website which contains a 7254-movie poster from 1980 to 2015 across 24 different genres.**

## INTRODUCTION

Movie posters embrace the whole message and feel of the film, and from the impact of the movie posters on the audience's curiosity, we can state that movie poster is more than just promotional material, but rather the primary design element that captures a viewer's attention. For humans, it takes less than a jiffy to determine the genre of a movie, though it might be simple, it involves millions of simultaneous processes starting from understanding the cues like color, ending with analyzing the expressions on the faces of actors. So, if humans can predict the genre of a movie from its poster by a glance, can't we expect the same from a machine?

The topic of multilabel classification with respect to movie posters allows us to take explore a wide variety of algorithms and network architectures to create a successful model. This project explores if there are certain elements of a poster that allow a model to predict the movie's genre. The input to my algorithm is a color image of a movie poster and the output model is a list of genres that classify the movie. Transfer learning and various deep learning models will be applied to make predictions.

## RELATED WORK

There have been various studies in genre classification using deep learning. The majority of genre classification models have focused on predicting the movie genre based on its poster. A study conducted by Nirman Dave [1] used a 7-layer CNN model trained on 4 genres which are Romance, Action, Horror, and Documentary. It found that the built model proposed in that paper does well on certain movie posters where the elements in the poster are clearly visible and large enough. And the highest accuracy was 50.5% when using a learning rate of le-4. Another study conducted by Kaushil Kundalia, Yash Patel, and Manan Shah [2] used transfer learning on 12 genres, the pre-trained models were the InceptinV3 on the weights of ImageNet. The achieved accuracy was 84.82% with a loss of 0.4892. Beyond the previous studies, there have also been efforts at genre prediction using images. Samuel Sung and Rahul Chokshi [3] customized versions of standard deep learning architectures were implemented: ResNet-50, VGG-16, and DenseNet-169. The best result was achieved by the DenseNet-169. I plan on using CNN's and transfer learning for the purpose of my project and modifying the network architecture.

## DATASET

The dataset [4] was collected from the IMDB website. One poster image was collected from one (mostly) Hollywood movie released from 1980 to 2015. Each poster image is associated with a movie as well as some metadata like ID, genres, and box office. The ID of each image is set as its file name. The dataset contains entries for 7,254 movies, with 25 genres, which are Action, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, History, Horror, Music, Musical, Mystery, N/A (other), News, Reality-TV, Romance, Sci-Fic, Short, Sports, Thriller, War, and Western.

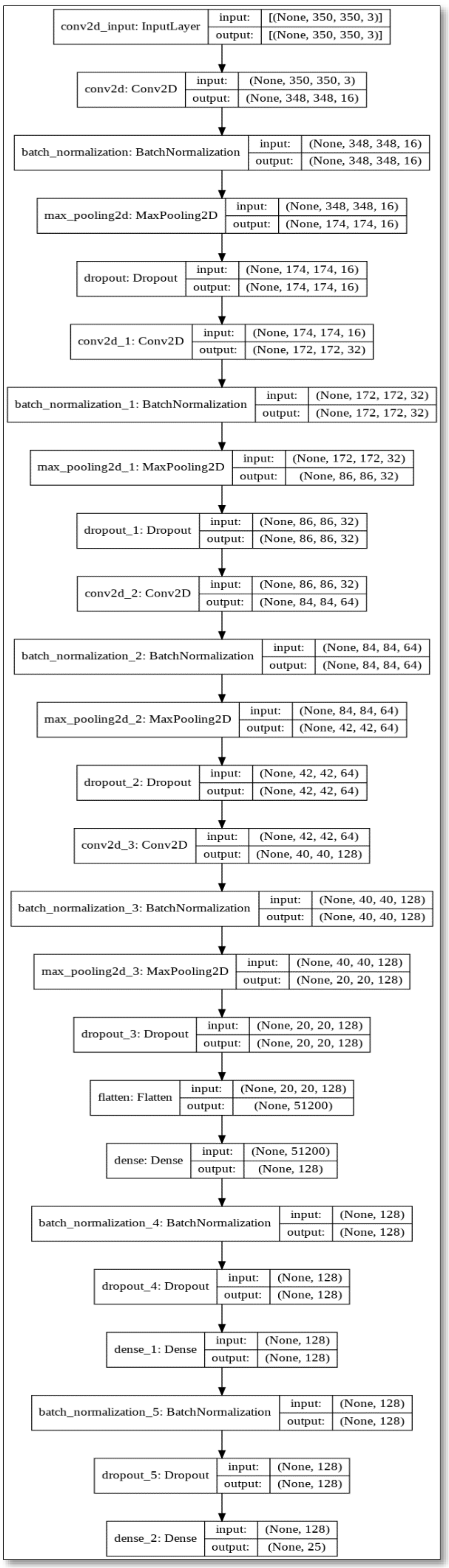
## FEATURES

The genres associated with a given movie are expressed 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. The Images are with different dimensions, but for the processing, we need the same dimensions for all the images, so I formatted each individual image into a 320x320 resolution image. And My raw input data is the color of each pixel in the image expressed in terms of RGB values - a 320x320x3 matrix. Then I loaded all the images that are in a separate file into the same file by their id, after that, I scaled the images by dividing each of them on 255.0. I split the data into train test data with a test size of 0.15.

## METHODS

### *A.* Convolutional Neural Network

One algorithm I implemented is Convolutional Neural Network or CNN. It is a class of deep neural networks that are used to analyze visual imagery. This type of architecture is dominant to recognize objects from a picture or video [5]. I used a sequential model since it is used for a plain stack where each layer can have exactly one input tensor and one output tensor. Then I added to it a 2D Convolution Layer, which helps produce a tensor of outputs. I determined the size of the convolutional window to be as 3x3 with 16 filters and the input shape as 350x350x3 and with using the relu activation function. After that, I added a Batch Normalization which allows each layer of a network to learn by itself a little bit more independently of other layers. And to increase the stability of a neural network, it normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. It also applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. I also added a MaxPool2D with a pool size 2×2. Finally, I added a dropout layer that is used to randomly set the outgoing edges of hidden units to 0 at each update of the training phase. For each layer in my model, I repeat these steps with different filters and learning rates, adding after the fourth layer a Flatten layer that is used to convert the data into a 1-dimensional array for inputting it to the next layer. The full CNN architecture.is shown in figure 1. Although the model is large and perhaps unreadable, I provided it for more understanding.



conv2d_input: InputLayer | input: [(None, 350, 350, 3)] | output: [(None, 350, 350, 3)]
conv2d: Conv2D | input: (None, 350, 350, 3) | output: (None, 348, 348, 16)
batch_normalization: BatchNormalization | input: (None, 348, 348, 16) | output: (None, 348, 348, 16)
max_pooling2d: MaxPooling2D | input: (None, 348, 348, 16) | output: (None, 174, 174, 16)
dropout: Dropout | input: (None, 174, 174, 16) | output: (None, 174, 174, 16)
conv2d_1: Conv2D | input: (None, 174, 174, 16) | output: (None, 172, 172, 32)
batch_normalization_1: BatchNormalization | input: (None, 172, 172, 32) | output: (None, 172, 172, 32)
max_pooling2d_1: MaxPooling2D | input: (None, 172, 172, 32) | output: (None, 86, 86, 32)
dropout_1: Dropout | input: (None, 86, 86, 32) | output: (None, 86, 86, 32)
conv2d_2: Conv2D | input: (None, 86, 86, 32) | output: (None, 84, 84, 64)
batch_normalization_2: BatchNormalization | input: (None, 84, 84, 64) | output: (None, 84, 84, 64)
max_pooling2d_2: MaxPooling2D | input: (None, 84, 84, 64) | output: (None, 42, 42, 64)
dropout_2: Dropout | input: (None, 42, 42, 64) | output: (None, 42, 42, 64)
conv2d_3: Conv2D | input: (None, 42, 42, 64) | output: (None, 40, 40, 128)
batch_normalization_3: BatchNormalization | input: (None, 40, 40, 128) | output: (None, 40, 40, 128)
max_pooling2d_3: MaxPooling2D | input: (None, 40, 40, 128) | output: (None, 20, 20, 128)
dropout_3: Dropout | input: (None, 20, 20, 128) | output: (None, 20, 20, 128)
flatten: Flatten | input: (None, 20, 20, 128) | output: (None, 51200)
dense: Dense | input: (None, 51200) | output: (None, 128)
batch_normalization_4: BatchNormalization | input: (None, 128) | output: (None, 128)
dropout_4: Dropout | input: (None, 128) | output: (None, 128)
dense_1: Dense | input: (None, 128) | output: (None, 128)
batch_normalization_5: BatchNormalization | input: (None, 128) | output: (None, 128)
dropout_5: Dropout | input: (None, 128) | output: (None, 128)
dense_2: Dense | input: (None, 128) | output: (None, 25)

Fig. 1: CNN Architecture

### *B. VGG-16*

VGG-16 is a convolutional neural network that is 16 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [6]. Using the VGG-16 model, I test a one picture. I loaded the model then prepared the image to test it on the model by loading it and converting it to a NumPy array, then reshape it and preparing it for the VGG model, and lastly, I retrieved the most likely results.

## Experiments and Results

### A. RAM getting crashed

Since I am using google colab through this project, I had an issue while compiling some line of code, the google colab RAM has 12 GB, and when I was trying to load the images and converting them to a NumPy array, a message appears telling me that my RAM has crashed, and there was no option to increase my RAM, so I tried a lot to find a way or to change my code, and then I found a google colab notebook with a 35 GB RAM, so I used it just by copying the notebook and then paste my code in it. Unfortunately, it took me time to figure a solution.

### B. ResNet-50

ResNet-50 is a convolutional neural network that is 50 layers deep. which you can load a pre-trained version of the network trained on more than a million images from the ImageNet database [6]. During implementing the ResNet-50 to train the model on a big set and have high accuracy. An error arose showing that Keras requires TensorFlow 2.2 or higher, and when I tried to download it the CNN model showed a very low accuracy, I tried to find many solutions but none of them worked so as a last try I opened another notebook and implement it without the CNN model, but the RAM crashed, after that, I knew it will not work, and it will take from me a lot of time to make it working.

### C. TensorFlow Versions

After building my model, another error raised which is the validation accuracy, for some reason it was so low, after doing some research I found that I have to download TensorFlow-GPU = 2.0.0-rc0, after downloading it I got a high accuracy and the model worked.

As a result, I got a 90% accuracy from the CNN model, as shown in figure 2 below the accuracy over 5 epochs, and in figure 3 the loss over 5 epochs and 68.61% from the VGG-16 model.

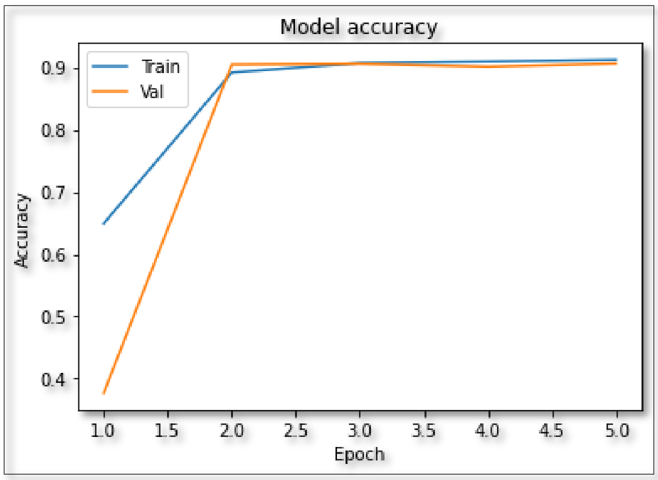


Figure 2: Model accuracy over 5 epochs

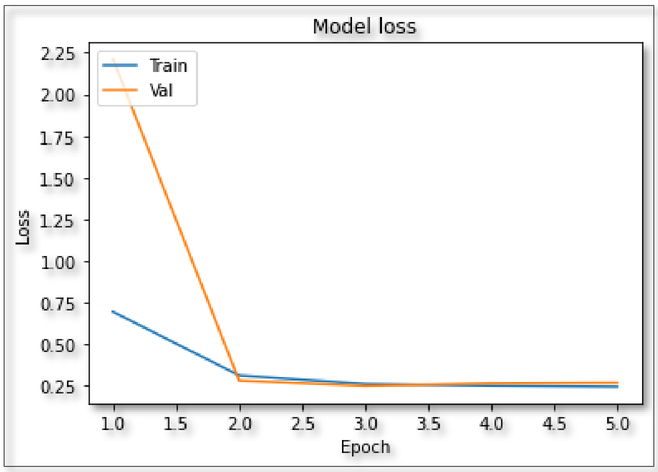


Figure 3: Model loss over 5 epochs

## FUTURE WORK

This work can be extended further by training the model on more than 24 genres and building more models like dens-169 or ResNet-50 also training the model on movie posters across different cultures, like Spanish, Arabic, and Italian movies. Also, we can improve our model by using the movie trailer. This project is like the first step to help a machine understand description through identifying movie genres.

## CONCLUSION

I noted that the CNN model proposed in this paper worked well where the elements in the poster are clearly visible. During working on the research many issues arose, that needed a lot of time to figure a solution, but after finishing this project I know that I improved my skills for sure.

## REFERENCES

[1] Dave, N. (n.d.). Predicting movie genres from movie posters. Retrieved March 17, 2021, from https://github.com/nddave/Movie-Genre-Prediction/blob/master/paper/predicting-movie-genres-paper.pdf

[2] Kundalia K, Patel Y, Shah M (2021) Multi-label movie genre detection from a movie poster using knowledge transfer learning. Augment Hum Res. https ://doi.org/10.1007/s4113 3-019-0029-y

[3] Sung, S., & Chokshi, R. (n.d.). Classification of Movie Posters to Movie Genres (Rep.)

[4] Movie genre classification based on Poster images with deep neural networks. (n.d.). Retrieved February 2021, from https://www.cs.ccu.edu.tw/~wtchu/projects/MoviePoster/index.html

[5] Tensorflow cnn image classification with steps &amp; examples. (n.d.). Retrieved March 2021, from https://www.guru99.com/convnet-tensorflow-image-classification.html

[6] ImageNet. (n.d.). Retrieved March 2021, from http://www.image-net.org/

## Appendix

1. Code on google colab
   https://drive.google.com/file/d/1b4h5A-ihI_tFfrZOw6etp6x7aXtM_1VJ/view?usp=sharing

2. Code in pdf file
   https://drive.google.com/file/d/1mBT-_y-sMjZoYfzcDCdikdT2Af1veGEX/view?usp=sharing