TÜRK-ALMAN ÜNİVERSİTESİ
TÜRKISCH-DEUTSCHE UNIVERSITÄT

Movie Genre Classification Based on Poster Images for INF003 - Spring 2020

Deniz Mert Tecimer
Department of Computer Engineering
Turkish - German University
e160503131@stud.tau.edu.tr
github.com/deniz997/MoviePosterClassification

Abstract

We will classify movies by genres, based on their posters. Beside that, it is also aimed in this project to examine the effect of style transfer on this multilabel-classification.

1 Introduction

Genre classification for movies, a type of multilabel-classification, is well used in industry. Such companies like Netflix, Amazon etc. use this functionality to deliver their contents, to whom it may attract. Beyond static genres, they even produce dynamic(combined) genres. Netflix also generates dynamic posters for their contents, accordingly to consumers interests. Based on this feature, we can say that posters are very important for attracting consumers and also they inform the consumer about the content of the product.[1] Some posters might lead its consumer wrong, and this might cause time waste and disappointment. Dynamic posters of Netflix are mostly based on the objects, and do not represent the style of the genre, movie belongs. This project might lead a future work to see the effect of style transfer on poster based movie genre classifier and by comparing this effect with a survey, with some audience and maybe lead a future work of making this dynamic posters better.

The input to our algorithm is a database of movie poster images and their genres. We then use a neural network to predict movie genres.

2 Database and Features

The database "Movie Genre from its Poster" from Kaggle is used in this project.[2] It contains information about movies from 1995 to 2018, such as IMDB id, IMDB link, title, IMDB score and the link of the poster of the movie. Data is pre-processed similarly as Rahul et al.(2020)[3]'s work. After pre-processing the database and the retrieval of the movie posters, a movie poster image dataset, consists of 36898 268X182 resolution RGB posters, is obtained. Distribution of examples is 80/10/10 as 31704 train set, 3522 val set, 3914 test set respectively. First movies with missing genres or image link to the posters are removed. Then posters are transferred in to the resolution of 150X150. Dataset contains 28 different genres, resulting a high imbalance on the dataset. To overcome this problem, we have determined genres with a threshold of 4500 examples per genre. 28 different genres are then reduced to top 6 most popular genres. For the movies, belong to a genre that is not in our list, we have created an extra column called 'Others' and set it to 1.

INF003: Neural Networks, Spring 2020, Turkish - German University, Istanbul. (LateX template borrowed from NIPS 2017.)

Genre	Count(Train)	Count(Test)	Total
Action	4824	469	5293
Comedy	11413	1105	12518
Drama	17968	1820	19788
Romance	5723	445	6168
Crime	4862	372	5234
Thriller	4352	435	4787
Other	19663	2177	21840

Table 1: Number of posters per genre

At the end we have the pre-processed data that the algorithm requires. Yet, the data is not equally distributed, and this limits the expected accuracy.

Three examples from the database are as follows:

Action | Adventure | Crime
Figure 1. Example 1

Action | Adventure | Comedy
Figure 2. Example 2

Action | Adventure | Comedy
Figure 3. Example 3

3 Multi-label Classification

The database contains movies, that has multiple genres. These genres are independent from each other. Thus we need to run N-classifiers on the data to determine whether the movie belong to a poster, where N is the number of genres to be predicted. In the multi-class classification problem, it must be determined, which genre movie belongs. In contrast to that in this project, we aim to classify movies for multiple genres, so that the output of the proposed architecture will be n X N, where n is the number of examples and N is the number of genres. As a result of that, we have to come up with an algorithm that calculates the probability of each genre.

4 Deep Neural Network

I have implemented a standard ResNet-50 architecture with some modifications, similar to the modifications proposed by Rahul Chokshi and Samuel Sung[3]. Because the model is based on the ResNet-50, I used it via Keras Applications API.The final fully connected layer of 1000 units of this base model was replaced by 3 sequential fully connected layers of 1024, 128 and 7 units. These layers use ReLu, ReLu and sigmoid activations.

I used sigmoid activation instead of softmax activation, because softmax divides the probability among the classes, which is helpful, when the input belongs to only one class, in contrast to multilabel-classification. As another result of that I used Binary-Crossentropy as a loss function.Binary-Crossentropy measures how far the predictions from the true value for each class and averages the class-wise error to calculate the final loss of a prediction.In order to deal with the imbalance of the dataset, class weights were implemented by using a built-in Keras function. During the training class weights are set as balanced. This built-in functionality calculates the weight of each class, according to its occurrence in the dataset. The loss function is described by the equation below:

$$\sum_{n=1}^{7} W_n(y_n \times log(y'_n) + (1 - y_n) \times log(1 - y'_n))$$

W_n is the weight of the class n. y_n is the ground truth value, and y'_n is the predicted value of genre n. The average value of the loss function over all the examples in a mini-batch was used as the cost function for training, as Rahul Chokshi and Samuel Sung[3] stated in their work.

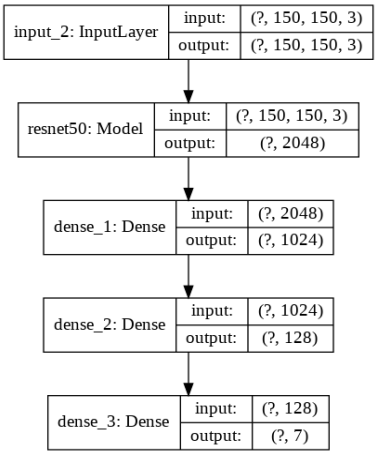

Figure 4. Model Graph

As an optimizer we used Adam. Adam is a gradient-based optimization algorithm, which performs very well on finding individual learning rates for each parameter. To generalize, an Adam optimizer adapts the learning rate to the parameters based on the change of parameters. It is often used for Deep Neural Networks. We used Adam with its default parameters, provided by Keras API.

Model has an input size of $150 \times 150 \times 3$ as image width, image height and color channels respectively. I chose mini-batch size of 128 and with that such relatively big mini-batch size I aimed to improve the generalization of the network. Decision threshold was 0,3 and all of these parameters were found through a heuristic approach and gave the best results within an acceptable computation time.

5 Experiments and Results

Total number of parameters for the experiments, using no transfer learning, are 25,817,991 and 25,764,871 of them were trainable.For the experiments, using transfer learning, 18,062,087 of them were trainable.The results are shown below as tables.

5.1 Random Initialization

First the model, described in Section 4, has been trained for 20 epochs in total. No transfer learning method was used, therefore all layers were trainable and weights were randomly initialized.This experiment is mentioned as ResNet-R in the tables.

5.2 Initialization with ImageNet Weights

Secondly the model has been trained for 20 epochs in total by using the weights of the pre-trained ResNet-50 model. This model was trained on ImageNet Dataset and was successful on detecting many object with a high probability. No transfer learning method was used, therefore all layers were trainable. This experiment is mentioned as ResNet-Img in the tables.

5.2.1 Transfer Learning

Thirdly the model has been trained for 20 epochs in total by using the weights of the pre-trained ResNet-50 model. Only last 40 layers of the model were trainable.This experiment is mentioned as ResNet-TL in the tables.

	ResNet-R			**ResNet-Img**			**ResNet-TL**		
Accuracy	**Train**	**Validation**	**Test**	**Train**	**Validation**	**Test**	**Train**	**Validation**	**Test**
Epoch 10	0,77	0,72	0,76	0,81	0,71	0,72	0,95	0,76	0,75
Epoch 20	0,85	0,73	0,74	0,94	0,74	0,75	0,98	0,78	0,77

Table 2: Epoch - Accuracy Table

	ResNet-R		ResNet-Img		ResNet-TL		Random
F1-Score	**Epoch 10**	**Epoch 20**	**Epoch 10**	**Epoch 20**	**Epoch 10**	**Epoch 20**	**Guess**
Action	0,13	0,29	0,16	0,31	0,30	0,22	0,21
Comedy	0,50	0,43	0,45	0,49	0,51	0,47	0,39
Drama	0,63	0,60	0,59	0,61	0,51	0,63	0,50
Romance	0,25	0,23	0,27	0,19	0,19	0,17	0,23
Crime	0,15	0,20	0,22	0,19	0,11	0,16	0,21
Thriller	0,20	0,25	0,09	0,21	0,03	0,17	0,19
Others	0,69	0,69	0,68	0,69	0,61	0,63	0,52
Weighted Average	0,52	0,52	0,50	0,52	0,45	0,49	

Table 3: F1-Score per genre

5.3 Results

In Table 3, blue cells denote that the result is better than random, grey cells denote that the result is worse than random and yellow cells denotes the best result. In Table 2, I have used built-in accuracy property provided by Keras, which chooses the best accuracy metric according to the output format of the model. In this case, model uses Binary Accuracy as the accuracy metric.Nevertheless, because the dataset contains a lot of negative values(0), it is not so meaningful to evaluate the model just by this accuracy metric. In the dataset, an example has maximum 3 positive values(1).And even if the model would predict everything as negative, it would deliver more than %70 accuracy.

Even it is not very determining for the evaluation of the model. Still it provides how fast accuracy increases, when the model uses pre-trained weights or transfer learning methods. Although reaching higher accuracy fast is a leverage for transfer learning methods, it did not improved model's generalization ability, which is discussed in Section 6.

According to the results denoted in Table 3, ResNet-R has the best weighted average F1-Score, followed by ResNet-Img. However, usually using transfer learning methods delivers better results, these results were unexpected. In other experiments without class weights, model was prone to overfit on the classes that cover the big portions of the dataset, which results some classes to have almost 0,0 F1-Score. ResNet-R has the best generalization performance among its competitors. After the first 10 epochs, ResNet-R has the best score on genres Drama and Others, ResNet-Img has the best score on distinguishing genres Romance and Crime, which are also more than a random guess and ResNet-TL has the best performance on genre Comedy. After 20 epochs, ResNet-R has the best score on genres Thriller and Others, on the other hand ResNet-Img has the best performance on genres Romance and Crime and finally, ResNet-TL has the best score on genre Drama, same as ResNet-R.

	Action	**Comedy**	**Drama**	**Romance**	**Crime**	**Thriller**	**Others**	**Average**
Epoch 20	0,83	0,71	0,59	0,82	0,87	0,85	0,54	0,74

Table 4: F1-Score of ResNet-50 from Rahul Chokshi and Samuel Sung's work[3]

In Rahul Chokshi and Samuel Sung's work[3], they denoted an F1-Score for their random initialized ResNet-50 model, trained for 20 epochs with implemented class weights and a decision threshold of 0.3.Results can be examined in Table 4.

6 Discussion

At first I wish to discuss about the performance of the models, based on the F1-Score they deliver,secondly the imbalance of the dataset,and then I wish to point to the difference between my results and the result of Rahul Chokshi and Samuel Sung's work[3].

People, their needs, cultural motives and perspective of the day changes constantly through time, which triggers the change on the style of banners and posters. While there are posters from 1995 to 2018, which also causes examples to come from different distributions, and makes the problem harder.

In contrast to artworks, posters should make it possible easy to people to have an intuition about the movie. People do not examine the posters to understand the context, they just look at the overall style of the poster. Yet, there are many movie posters, which lead its audience wrong. For example, a clown can be an object on a movie poster, which might belong to either the genre thriller or comedy.Thus, using transfer learning or initializing the model with pre-trained weights might be leading the model to associate genres with the images on the poster and because objects might lead the audience wrong, as explained, therefore these model might not deliver the best results. On the contrary, when the model is with random weights initialized, it can learn the other features, rather than looking for the objects for the prediction. It might be overcome, when the model is trained for more epochs.

Even I used balanced class weights, it causes model to overfit on the examples of the classes, which have relatively less examples.Therefore, model can not gain a high ability of generalization for these classes and results worse on these classes in the aspect of F1-Score.

The difference between my results and the result of Rahul Chokshi and Samuel Sung's work[3], might be a result of the input shape, they used in their work.They used images as 256 X 256 in RGB format. Shrinking the image size to 150 X 150 might cause some information loss on the data, even it causes relatively high computational cost.

7 Future Works

- Detecting objects on the posters through a YOLO model and feeding this deep neural network with its output will be tried.
- More data will be collected to overcome the overfit on the examples of the classes, which have relatively less examples.
- A custom and relatively shallower network will be implemented and results will be compared. This network also can be fed with the output of the YOLO model.
- More models will be tried, such as Xception, VGG-16, MobileNet. Their performance will be evaluated with and without transfer learning.
- Problems, denoted in Section 6, will be examined.

References

[1] Amat, F., Chandrashekar, A., Jebara, T., & Basilico, J. (2018). Artwork personalization at Netflix. *Proceedings of the 12th ACM Conference on Recommender Systems.* doi: 10.1145/3240323.3241729

[2] Neha. (2020, April 10). Movie Genre from its Poster. Kaggle. https://www.kaggle.com/neha1703/movie-genre-from-its-poster

[3]Rahul Chokshi, Samuel Sung. (2020) Classification of Movie Posters to Movie Genres. CS230 - Stanford University.