Genre Prediction for TV Series

Zübeyir Bodur
Computer Engineering
Bilkent University
Ankara, Turkey
zubeyir.bodur@ug.bilkent.edu.tr

Alperen Öziş
Computer Engineering
Bilkent University
Ankara, Turkey
alperen.ozis@ug.bilkent.edu.tr

***Abstract*—We aim to examine the performance of neural networks designed for multi-label classification task, by training them with a dataset of different context, tuning the hyperparameters accordingly. In our case, the multi-label classification task is predicting the genres of TV series, by using a neural network designed for predicting the genres of movies. The dataset consists of 6803 posters of TV series, which classifies those posters in different genres in multi-hot encoded format. The model consists of 7 layers, where first 4 of them are convolutional neural networks, followed by fully connected neural networks. It uses Adam optimizer, categorical crossentropy loss and inverted dropout as well. By tuning the learning rate and shrink ratios of the posters, we will train the model and gather our results.**

***Index Terms*—neural networks, classification, computer vision, genre prediction, model**

I. Introduction

Our mind can work both non-deterministic and deterministic, therefore we can easily make predictions with or without having enough prior knowledge on what we are predicting. Computers, on the other hand, can not make such decisions that easily. Depending on the application, they will need a set of data, large or small, to make such predictions. Some of the applications have scarce data, therefore methods that require less data were developed, such as conjoint data analysis. For example, using this method, we can predict the rank of another computer by using the rankings of computers ranked by a customer, who ranked as less as 16 computers [1]. This is because we have several features about those computers, such as the processing power, memory size, other than their name or brand.

Multi-label classification, however, is a task where large amount of data can be found. For example, in order to classify set of images of animals, we can easily find hundreds of thousands of images of cats, dogs, or any specific animal image. However, only data we have in this problem is the image of an animal itself. Therefore, in addition to ease of finding data on this task, the models being developed also require large data, a model that was trained with only hundreds of movie posters would be unsuccessful, as there would be a lot of bias and error due to data scarcity, and trained model would be less random. For solving multi-label classification and other problems that require data, there have been various methods developed. Using Neural Networks (NN) have become a common practice as of 2022.

Genre prediction is also a multi-label classification task. So far, There have been various methods developed, in which using NNs become common. However, it is still questionable that a genre predictor that is developed for one domain can be applied to another domain. In this paper, we aim to answer this question by using a NN that was developed for genre prediction of movies, and by training and testing this model with a dataset of TV series, and discuss the results.

II. Related Literature

One of the earliest works for classifying a dataset for genre prediction was developed by Z. Rasheed and M. Shaah in 2002 [2]. The data to be classified was trailers of movies. Using features such as average shot length and visual disturbence in the scenes, they were able to classify movies into action and non-action. Then, they did further classification into non-action movies, and were able to subclass those non-action movies into horror, comedy and drama or other as well, by looking at the distribution of light in those trailers [2].

Later, in 2014, M. Ivasic-Kos, M. Pobar and L. Mikec tried to classify images of movie posters into genres [3]. They examined the low-level features of those posters, such as edges and colors to classify posters into 6 different genres, using a dataset of 1,500 movie posters. Later, in 2015, M. Ivasic-Kos, M. Pobar, and I. Ipsic tried this on 6,000 posters, but they used algorithms such as K-Nearest Neighbours (KNN), Random K-Labelsets (RAKEL), and Naïve Bayes. This way, they were able to transform multi-label classes into multiple single-label classifications [4].

First approach that used neural networks (NN) was introduced by S. Kjartansson and A. Ashavsky [5]. In their paper, they discuss genre classification of books. They used a dataset of 19,000 books which were classified into 10 genres. In addition to the covers of the books, they also had another dataset for titles of the images, since they also used these to predict the book's genre. As their approach, they used numerous methods and tested each of those methods, namely Fully Connected Networks (FCN), Convolutional Pool Networks, SqueezeNet, VGG-16 and 5 more methods.

In 2017, an interesting paper, written by W. Chu and H, Guo, objects detected in a movie poster were also considered by using YOLO object detection in the NN, which was trained with IMDB movie dataset [6]. However, the accuracy of the model were as low as 19 %.

Currently, there are plenty of open source repositories in GitHub and datasets in Kaggle, where genre classification problem is attempted to be solved [7]. One of them was Nirman Dave, who tried to mimic a mini-VGG styled network using 7 layers. The results were promising, however, within their hyperparameters, the model achieved 51 % accuracy in their predictions [8].

III. Approach

The approach we will be using for predicting the genres of TV series from their posters is to use Nirman Dave's model [8], which was previously designed for movies.

A. Architecture

The model consists of 4 convolution layers, followed by 3 fully connected layers. The layers were sequentially placed as follows:

$$ConvLayer1(size = 3x3, filters = 32) \rightarrow$$
$$Max - Pool(size = 2x2) \rightarrow$$

$$ConvLayer2(size = 3x3, filters = 32) \rightarrow$$
$$Max - Pool(size = 2x2) \rightarrow$$

$$ConvLayer3(size = 3x3, filters = 32) \rightarrow$$
$$Max - Pool(size = 2x2) \rightarrow$$

$$ConvLayer4(size = 3x3, filters = 64) \rightarrow$$
$$Max - Pool(size = 2x2) \rightarrow$$

$$FullyConnectedLayer(size = 64) \rightarrow$$
$$Dropout(probability = 0.5) \rightarrow$$

$$FullyConnectedLayer(size = 32) \rightarrow$$
$$Dropout(probability = 0.5) \rightarrow$$

$$FullyConnectedLayer(num\_classes = 4)$$

All layers except the last one has a ReLU activation to introduce non-linearity in the network, where the ReLU function is the following:

$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{1}$$

The last layer uses a Softmax activation, so that output vector is converted into $num\_classes$ amount of numbers, $P_i(I)$, in the closed interval [0,1], which denotes the probability if a TV series whose poster $I_{m,n}$ is given belongs to $genre_i$. The number of classes, and what those classess are selected in consideration of the exploratory analysis of the dataset in section IV-A.

B. Optimization

Optimizers in neural networks are used to help the model learn by tweaking weights in the network. For this project, we will use the same optimizer as Dave uses, which is a well-known stochastic optimization called Adam [9].

The reason for not changing the optimizer is that Dave uses this optimizer as it works well on data with noisy gradients, like poster image data. In our case, TV series posters also share this problem, therefore Adam optimizer should still work in theory.

Adam optimizer is defined as follows, where the moving averages of the gradient (dx) and gradient squared (dx^2) are m_t and v_t respectively, and model weights in each iteration t is w_t; if learning rate is η, and ϵ is a small number to prevent division by zero, $\hat{m_t}$ and $\hat{v_t}$ are biased estimators of m_t and v_t [10], [11]:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)dx \tag{2}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)dx^2 \tag{3}$$

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{4}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{5}$$

$$w_t = w_{t-1} - \eta \frac{\hat{m_t}}{\sqrt{\hat{v_t} + \epsilon}} \tag{6}$$

C. Loss Function

Loss is a summation of errors for each data point made by training and validation set. The idea is to tune the model to decrease the loss as much as possible. For this project, we have decided to use the same loss function as Dave uses, namely Categorical Crossentropy Loss (CC Loss), since genres in TV series are also multi-hot encoded, that is a TV series can have more than one genre [9].

CC Loss measures the average number of bits needed to identify an event drawn from a set [9]. In this case, this event is the correct set of genres for a given TV series poster, and the bits are genres for which a given poster is predicted. CC Loss is defined by:

$$CCLoss = - \sum_{i=1}^{num\_classes} x_i \cdot log_2(\hat{x_i}) \tag{7}$$

where $\hat{x_i}$ is the model output, which is an array of size $num\_classes$, corresponding the prediction percentage for each genre, and x is the target values for x, where $\forall x_i \in \{0, 1\}$ [12].

D. Regularization

Regularization is a process of reducing overfitting in the neural network. In this project, we will keep using Inverted Dropout method from the model we are using.

Inverted Dropout method drops-out randomly chosen units in a neural network at a probability $p : dropout\_rate$. By dropping randomly selected units, interdependent learning among the neurons and overfitting can be reduced [13]. For this project, we will keep using the same dropout rate $p = 0.5$ and will not use Inverted Dropout on the test set.

E. Hyperparameters

The model obviously has several hyperparameters. However, in this project, we will fix the following:

- Dropout rate, $p = 0.5$
- Number of layers, 7.
- Activation functions. ReLU is needed for CC Loss, as the probabilities can not be less than zero. For the last layer, Softmax can not be changed as well.
- Batch size, fixed at 32.
- Number of epochs, fixed at 50.
- Decay rates for exponential moving averages of m and v, namely β_1 and β_2. Fixed at $\beta_1 = 0.9$, $\beta_2 = 0.999$
- Epsilon, fixed at $\epsilon = 10^{-8}$.

The following hyperparameters will be tuned:

a) Learning Rate (η): Learning rate is the η value in the Adam optimizer discussed in III-B. The original paper which introduces Adam optimizer recommends the default value of learning rate to be $\eta = 10^{-3}$ [9]. We will tune the learning rate for the values in the set [10^{-2}, $5 \cdot 10^{-3}$, 10^{-3}, $5 \cdot 10^{-4}$, 10^{-4}], while Dave's model tunes for the values in the set [10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}] [9].

b) Input image size: Originally, the model aims to see if the CNN can pick up on granular details on images that are less shrunk, meaning the model will always be trained on shrunken images [9]. In this project, we will also include the original image size (182x268) in the training process. Therefore the shrink ratios will be 30, 44, 58, 72, 86 and 100. Below is a table for shrink ratios and the size of the image after shrinking:

TABLE I
SHRUNKEN IMAGE SIZES

Shrink Ratio (%)	Size (pixels)
30	55 x 80
44	80 x 118
58	106 x 155
72	127 x 193
86	157 x 230
100	182 x 268

IV. EXPERIMENT

As the purpose of the project suggests, we are using a dataset of TV series, which includes their genres and posters.

A. Dataset

We were not able to find a ready-to-use dataset that contained TV series posters together with their genres. So we decided to create our own dataset using IMDB database [14] and a Python library which allows us to fetch data from IMDB. We joined some of the tables in the database and selected unique ids of the tv series released after 1950 with at least 1000 vote to eliminate some of the possible samples. Then using these ids and a library called imdbpy [15] we retrieved genres and poster image URLs of the TV series. and then finally we have downloaded the poster images. The dataset with which we are going to train the model consists of 6803 images of posters of TV series. We have a csv file which has a row for every movie containing the $uniqueId$ and the $genres$ related to that series. And then in a folder we have our poster images who are named with with their ids. Though these images are not all in the same size, we will resize them while preprocessing.

V. FUTURE WORK

Currently, in the project, we have created our dataset, chosen the model we will use, and decided how the hyperparameters will be tuned. In the next month, we will be conducting exploratory data analysis to decide which genres should be categorized, and perform our experiments on training data and test data, and will conclude our work.

REFERENCES

[1] S. Dayanık, ”Traditional Conjoint (or Trade-off) Analysis and Market Simulation”, pp. 1-10, Mar. 15, 2022. [Online]. [Accessed Mar. 23, 2022].

[2] Rasheed, Zeeshan & Shah, Mubarak. (2002). Movie Genre Classification By Exploiting Audio-Visual Features Of Previews. 2. 10.1109/ICPR.2002.1048494.

[3] Ivašić-Kos, Marina & Pobar, Miran & Mikec, Luka. (2014). Movie posters classification into genres based on low-level features. 1198-1203. 10.1109/MIPRO.2014.6859750.

[4] Ivašić-Kos, Marina & Pobar, Miran & Ipsic, Ivo. (2015). Automatic Movie Posters Classification into Genres. Advances in Intelligent Systems and Computing. 311. 319-328. 10.1007/978-3-319-09879-1_32.

[5] Kjartansson, Sigtryggur & Ashavsky, Alexander. (2017). Can you Judge a Book by its Cover? Retrieved from http://cs231n.stanford.edu/reports/2017/pdfs/814.pdf

[6] Chu, Wei-Ta & Guo, Hung-Jui. (2017). Movie Genre Classification based on Poster Images with Deep Neural Networks. 39-45. 10.1145/3132515.3132516.

[7] J. Zhou, “Genre classification on movie posters,” www.linkedin.com. https://www.linkedin.com/pulse/genre-classification-movie-posters-jianda-zhou/ [Accessed 27-Mar-2022]

[8] N. Dave, “Movie Genre Prediction,” GitHub. [Online]. Available: https://github.com/nddave/Movie-Genre-Prediction [Accessed 27-Mar-2022]

[9] N. Dave, “Movie Genre Prediction,” GitHub. [Online]. Available: https://github.com/nddave/Movie-Genre-Prediction/blob/master/paper/predicting-movie-genres-paper.pdf [Accessed 27-Mar-2022]

[10] D. Kingma and J. Lei Ba, “Adam: A Method For Stochastic Optimization,” ICLR 2015. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf. [Accessed: 27-Mar-2022].

[11] Vitaly Bushaev. “Adam, Latest Trends in Deep Learning Optimization.” Towards Data Science, Towards Data Science, 22 Oct. 2018, [Online]. Available: towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.[Accessed: 27-Mar-2022].

[12] “Categorical Crossentropy Loss Function.” [Online]. Available: peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy. [Accessed: 27-Mar-2022].

[13] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014, Accessed: Aug. 11, 2020. [Online]. Available: https://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf/ [Accessed: 27-Mar-2022].

[14] ”Tables From IMDB Database”. [Online]. Available: https://datasets.imdbws.com/ [Accessed: 27-Mar-2022].

[15] “Cinemagoer (previously known as imdbpy).” [Online]. Available: https://cinemagoer.github.io/. [Accessed: 27-Mar-2022].