Algorithms for Massive Dataset Project Report

Simone Laudani

January 29, 2025

1 About the dataset

The dataset used is the **Letterboxd (Movies Dataset)**, available from Kaggle at: Kaggle link. The dataset state is of 16 January 2025. It contains more than 900k movies data, about their title, actors, genres, releases, countries, .. and circa 760k image posters of the movies.

The parts taken from the database are:

- genres.csv

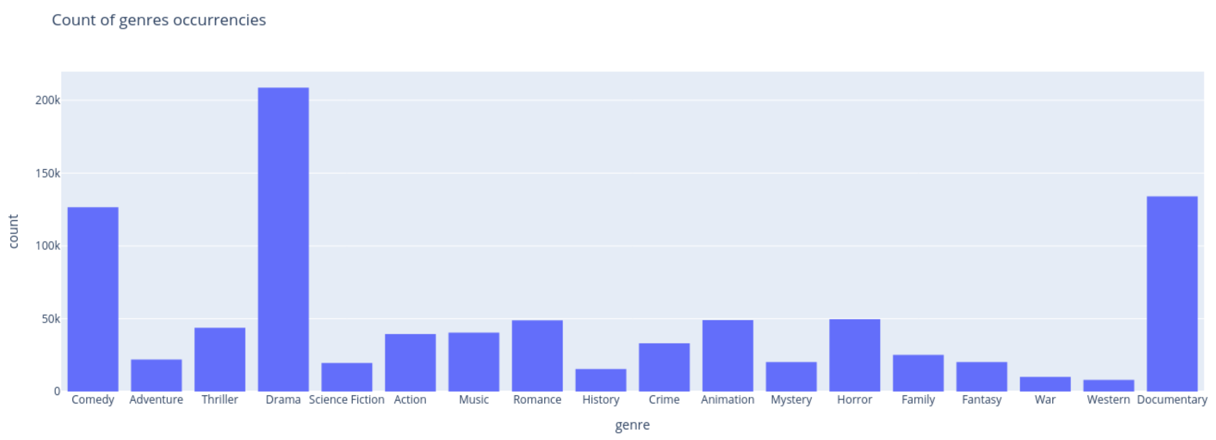
  With columns: *id* (integer uniquely identifying the movie) and *genre* (string identifying one genre).

  Note that a movie can have multiple genres, so there will be multiple rows with same id and different genre.

- posters

  Is a folder containing *jpg* image files in the form of: *movie_id.jpg*. For example: *1000001.jpg* would be the poster associated to the movie having id = 1000001.

2 Data organization



Distribution of genres in the full original dataset (minus *TV Movies* genre).

As we can see the original dataset is very unbalanced: the 3 most common genres appear more than double of any other.

So I made a smaller subset of 10k movies and organized it in two parts:

1. 70% is an equal distribution of movies according to their genres. Meaning that, having 18 unique total genres, there are $(10000 * 0.7) \div 18$ movies per genre, picked only from the movies having a single genre. Otherwise the model would be too much weighted towards more common genres.
2. 30% is random picked movies from the remaining ones, to simulate the real world distribution of genres.

This subset is available from Hugging Face at: `https://huggingface.co/datasets/sisimone/movie_subset`.

If instead we choose to use the full dataset, to balance it more, I implemented an undersampling of the most common genres by discarding half of the mono-genre movies among the most common ones.

For both cases, subset and full dataset:

- The movies having genre: *TV Movies* are discarded, since it doesn't have distinguishable features in the movie posters.
- The entries from *genres.csv* of the movies that don't have a corresponding poster image available in the dataset are also dropped.

Then I created a python *dict* to associate each movie id to its list of genres.

3 Data preprocessing

The tensorflow dataset is created starting from a list of all the poster images file paths. For each image file are mapped two elements of the tuple feature-label:

- **decoded image**: the image file is decoded, resized to 128x128 pixel to be more space efficient while maintaining enough visual features, and normalized with pixel values from 0-255 to 0-1. The resizing and normalization are implemented as network's layers for portability.
- **encoded label**: using the *movie_id-genres* dict we can associate the image to the list of its genres. This list is encoded into a **multi-hot** vector of length equal to the number of equal genres, and having 1 for the indexes of the genres of the considered movie, 0 otherwise. That is because I allowed movies to have more than one genre. Otherwise a one-hot implementation would be needed.

Other functions are then applied to the dataset; explained in the *"Scaling up with data size"* paragraph.

4 Algorithms and implementation

After the decoding of the images and encoding of the labels we need to **shuffle** and **split** the dataset into the training, validation and test sets. They will then be **cached** in case we are working with the sampled dataset (otherwise the size would be too large to fit in memory), **batched** and a **prefetch** function applied.
The last is useful to parallelize the reading and training of the data: while a data batch is being trained the next batch is being read instead of doing it at separate times sequentially.
Then we can proceed to feed the data to the model.

The model is made up of a **resizing** and **rescaling** layers, 3 **convolutional** layers with a **max pooling** layer for each.
After a **flattening** layer there are two final **dense** layers.
Every convolutional and dense layer have a **ReLu** activation function, while the last layer has a **Sigmoid** activation function to allow for a multi label classification.

The training is made with the *accuracy* metric for 30 epochs, with a *early stopping* callback function to monitor the validation accuracy and prevent over fitting of the model.
At the end an evaluation of the model is performed against the test set to give the accuracy of the model.

5 Scaling up with data size

To optimize the algorithm with larger data size, the solutions are:

- **Avoiding caching**: the data will not be cached to central memory avoiding disk accesses. But this will prevent the fast filling of the memory space.
- **Prefetching**: as explained before the prefetching allows to have a faster data loading and processing by parallelization.
- **Checkpointing**: implementation of model saving to disk in between epochs to be able to recover the training in case of crashing, or to pause and resume the now extended training time.

6 Experiments

The experimentation was firstly about the sampling method of the original dataset. Choosing a subset size of 10 thousand movies, how to pick them?

Chronologically ordered experiments tried:

1. **Random** movies picked by *id*. It creates the most unbalanced dataset with a strong predominance of the *drama* and *comedy* genres.
2. **Totally equal distribution** of the genres. Meaning that the number of picked movies would be equally divided among the total number of unique genres, in order to have an equal number of movies per genre.
3. **Mix between equal and random** picking of genres. To have both the advantages of: the model being able to generalize better thanks to the equal distribution and still reflecting the real world prevalence of drama and comedy movies.

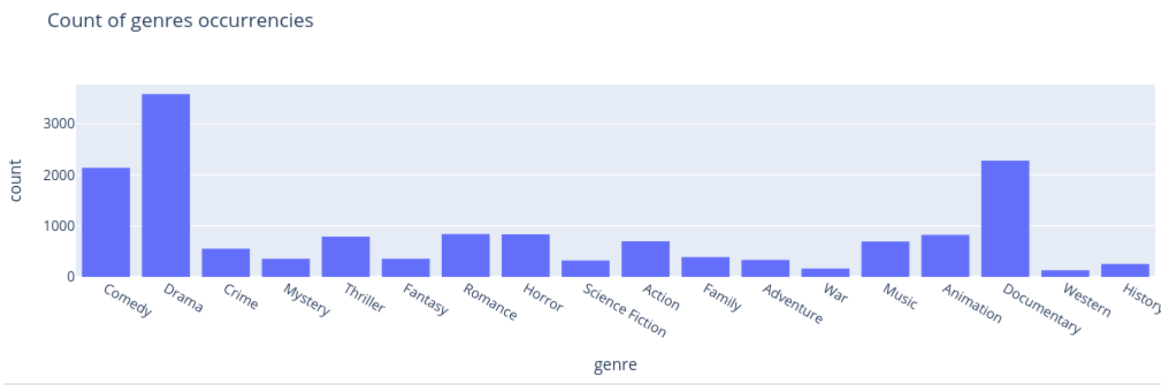
   So I decided to have a 70% of the total subset size to be equally distributed genres, plus the remaining 30% of random picked.

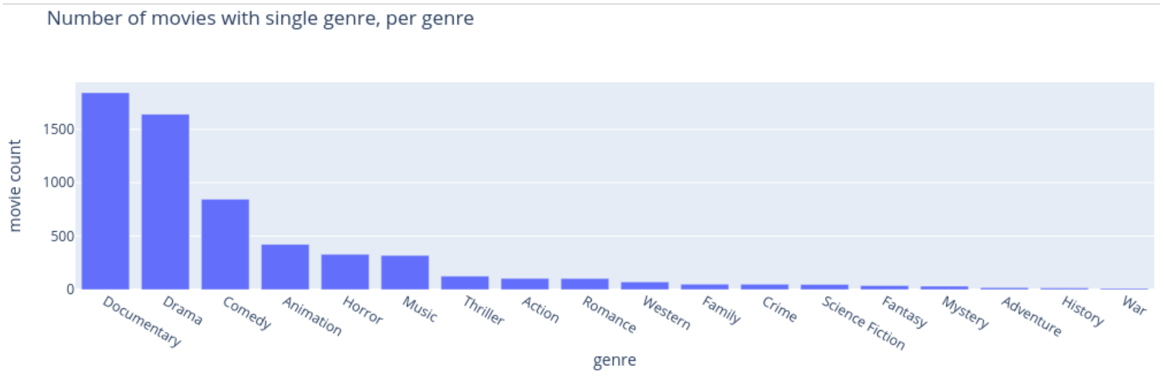
6.1 Experiments graphs

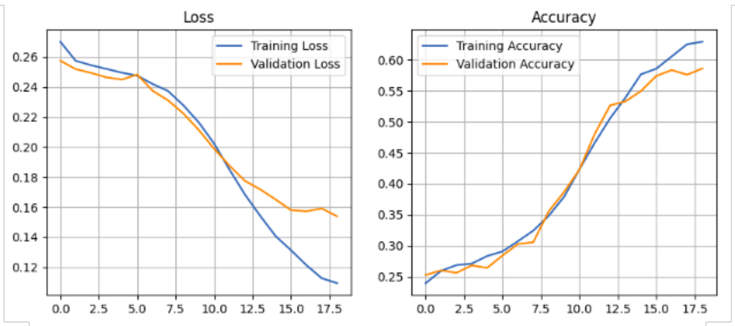
The following graphs are the results of the 3 experiments aforementioned. One per page.

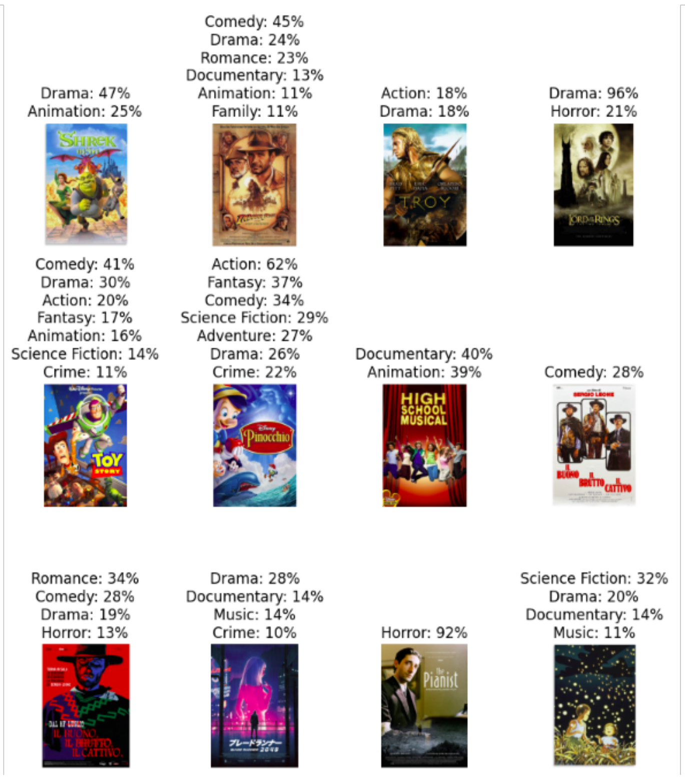
Each of them is divided into 4 parts:

- The amount of genres entries by genre. Note that a movie can have multiple genres, resulting in increasing the count of each of them.
- The distribution of genres among the movies with just one genre.
- The plotting of the loss and accuracy curves for each epoch while training.
- Predictions for random movie posters with the degree of confidence (only above 10% is showed).

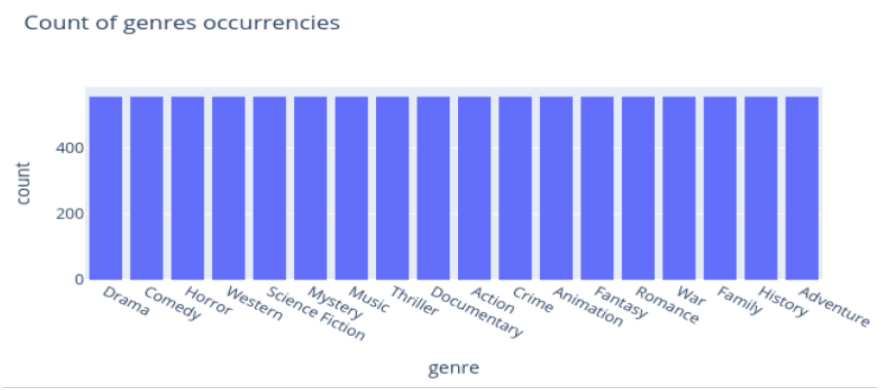


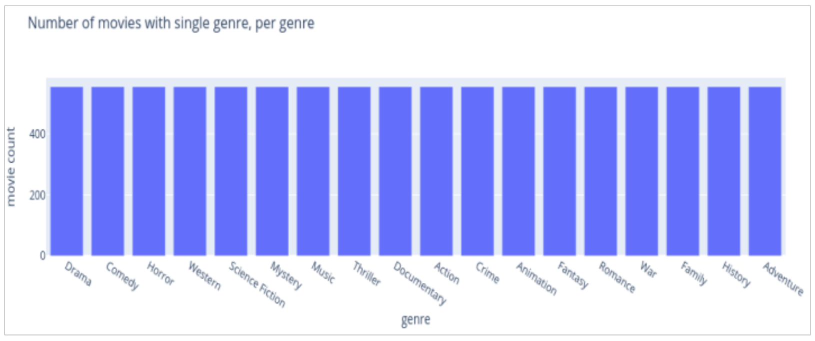


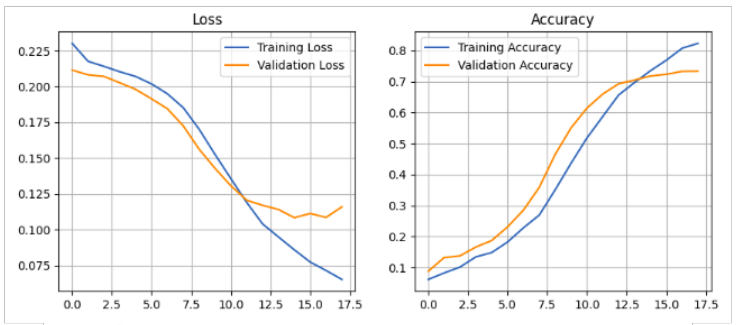


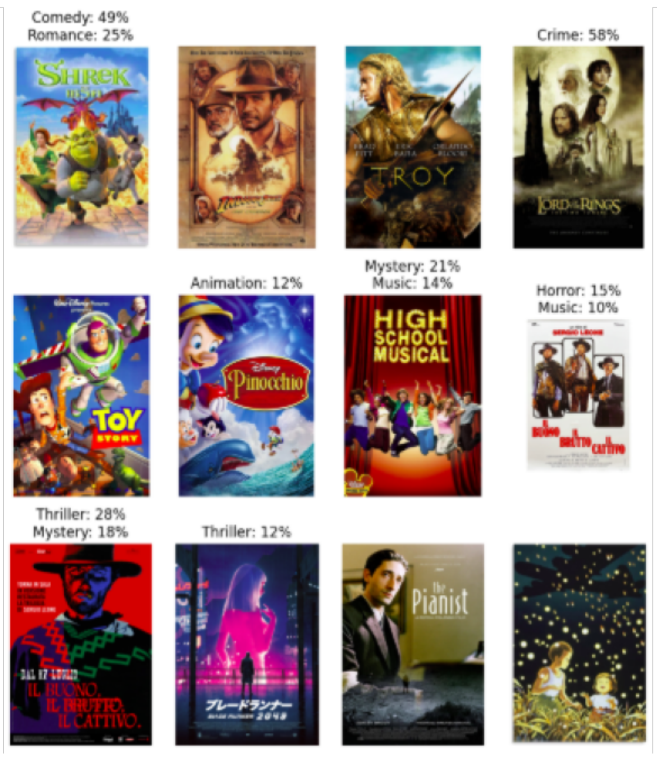


The random sampled.

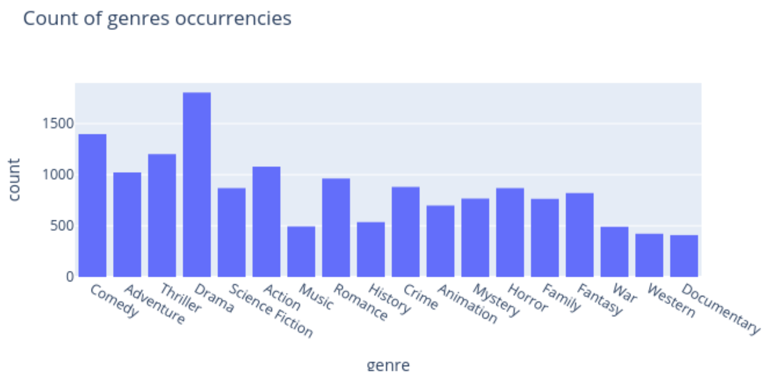


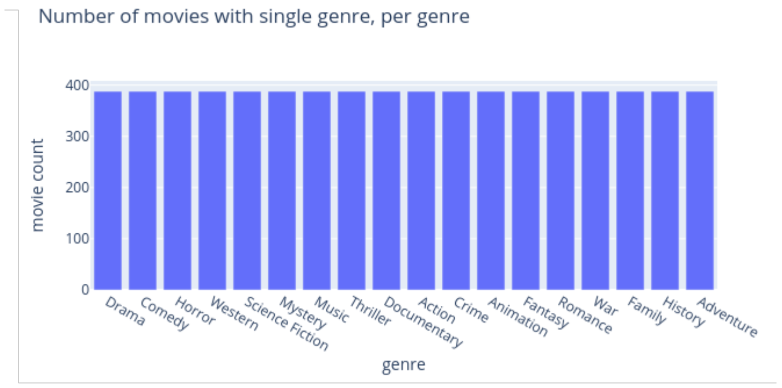


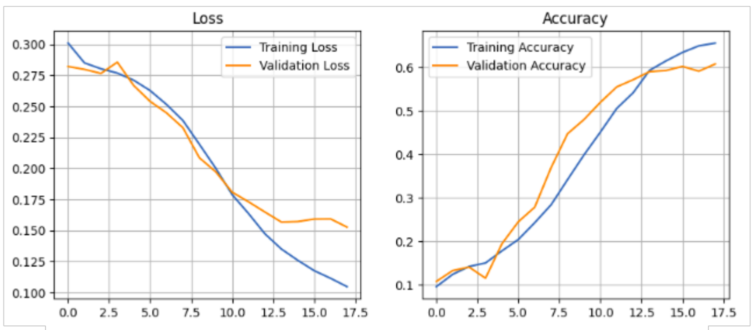


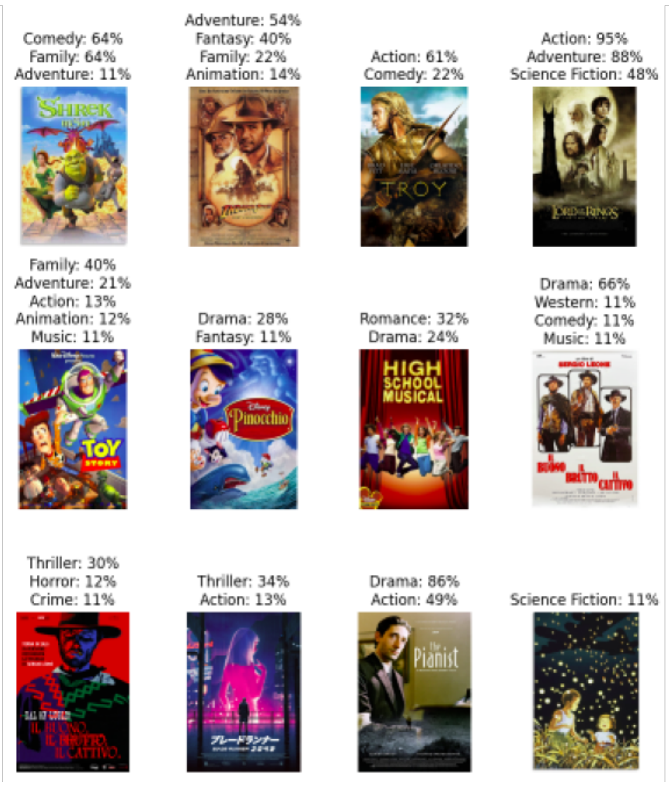


The equally sampled.









The mixed sampled.

6.2 Results analysis

From the accuracy plots we can see that the models start overfitting after the 13th epoch.

Let's focus the attention to how the distribution of the genres affects the final prediction tests:

In the first case, for the random sampling, the most common genres: *Drama*, and *Comedy* appear in almost every movie genres array.

In the second case, for the equally sampled, note that the first and second graphs are the same. The model struggles to decisively assign a genre to a movie, resulting in very few movies with a strong confidence of the prediction. Instead we have the majority of the movies with prediction confidences below 15% (under 10% is not showed).

The last example, the mixed sampling, clearly shows the best results among the three.

The accuracy of the model with the third approach is around 60%. I don't think it can be improved by much more than that, since even for humans is hard to *judge a movie by its cover.*

7 Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.