Kaggle Compeition : TMDB Box Office Prediction with Deep Neural Network

Cheng Chi Fung (12219691)
cfchengac@connect.ust.hk

Abstract

In this project, we try to use deep neural network to tackle the TMDB Box Office Prediction Compeition (https://www.kaggle.com/c/tmdb-box-office-prediction/data), a box office revenue prediction competition based on the metadata of past films from The Movie Database (TMDB). The architecture of our deep neural network is mainly composed by attension-based bi-LSTM, Resnet and feedforward network with attention and it is trained by the movie data after feature enginerring that achieves a test set minimum MST Loss of 0.5991 and obtain the score 2.66035 on kaggle platform

1 Introduction

1.1 Box Office Prediction

Film industry is booming, the revunues are growing, so we have a lot of data about films. What we want to see is if we can use the latest advances in artificial intelligence to accurately predict film revenues in order to make some changes in movies to increase their revenues even further.

1.2 Box Office with Deep Neural Network

Moreover, we find that most of the participants in this kaggle compeition using some popular machine learning model like xgboost or lightgbm to solve the problem and achieve a very good results (from kaggle discussion and kernel) and none of the participants using deep neural network. Therefore, in the project, we also want to see the performance of applying deep neural learning in this problem.

1.3 Problem Statement

In this probject, we are going to implement a deep neural network model by using the only data source, the dataset given in the competition to predict the box office revenue. The implemented model would be allowed to takes moive's features, included its categorical, text, image and numerical features. And the model would then outputs the predicted box office revenue.

2 Dataset

The dataset will be using in this project is the movie dataset given in this kaggle compeition, which is a dataset consisting of 7,398 past films metadata collected from The Movie Database (TMDB). Moreover, this dataset has been further seperated into training set and test set which consists of 3,000 films and 4,398 films metadata respectively. The data point in the dataset includes the films's cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries etc. The full list of the features in the original dataset are as follow.

Table 1: dataset columns

Columns
belongs_to_collection
budget
genres
homepage
imdb_id
original_language
original_title
overview
popularity
poster_path
popularity
production_companies
production_countries
release_date
runtime
spoken_languages
status
tagline
title
Keywords
cast
crew
revenue **(Only in training set)**

2.1 Exploratory Data Analysis

Before the data preprocessing, some analysis has been performed on the training sets to summarize their main characteristics

2.1.1 Number of Missing Values

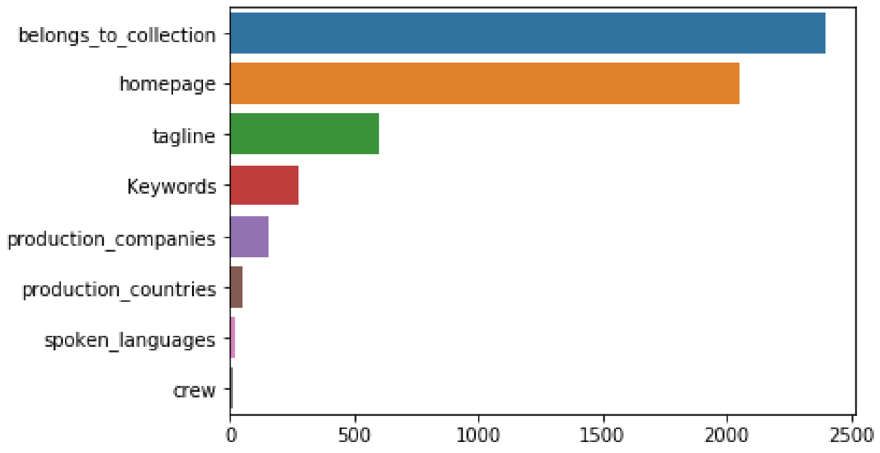

Figure 1: Number of missing values in each columns

belongs_to_collections and home_page contain so many missing value, it is better to drop it or extract some new feature from them.

2.1.2 Tag lines

Most of the films in the training set are related to love, story, world and life.

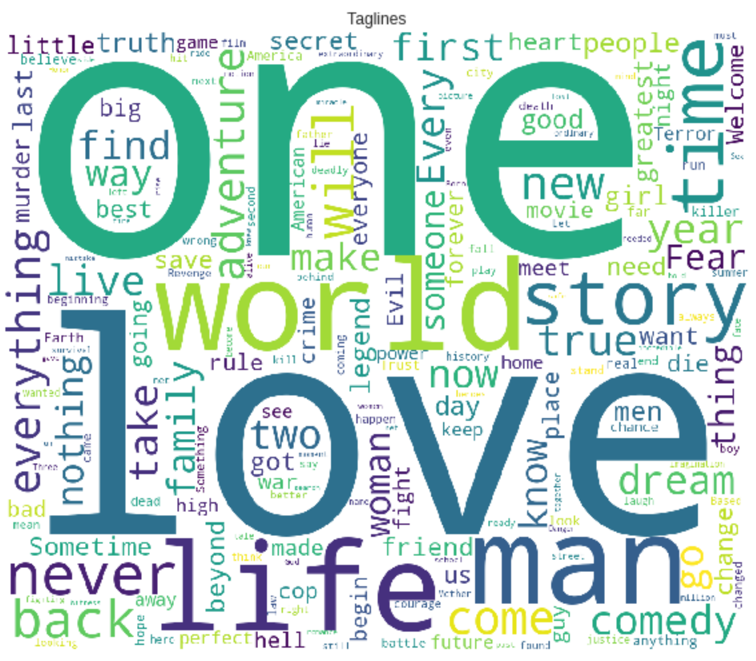

Figure 2: taglines word cloud

2.1.3 Production Countries

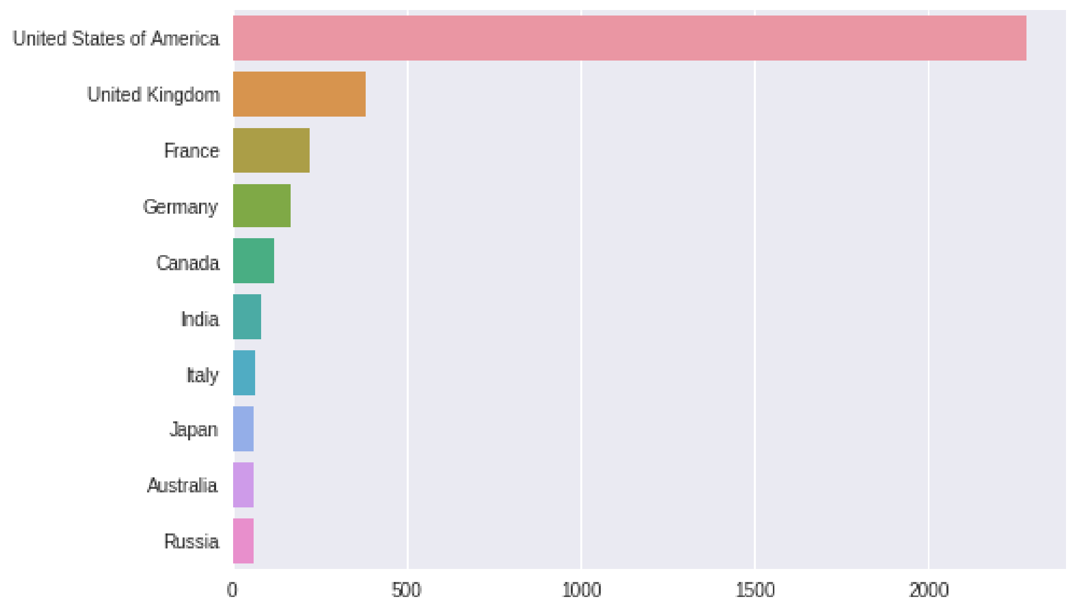

Figure 3: top production countries

Most of the films are produced in USA, UK and France. For the film of that come from the countries which only produce small number of films such as Russia. It may be difficult to predict their revenues using our model.

2.1.4 Spoken Languages

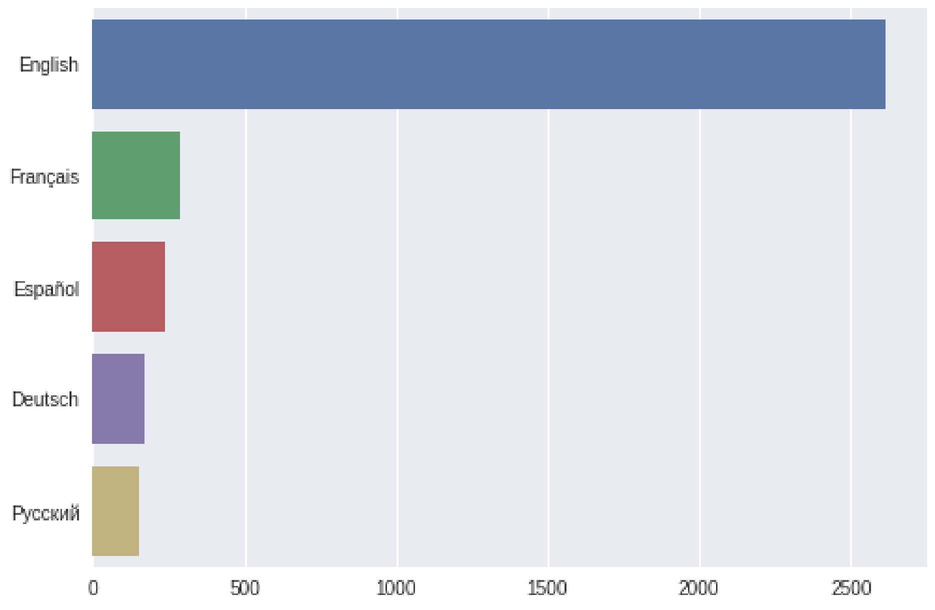

Figure 4: top production countries

Most of the films has the spoken languages of English, France, Espafnol. It is interested that Russian get the top five in spoken language but not appear in the top five of production countries.

2.1.5 Genres

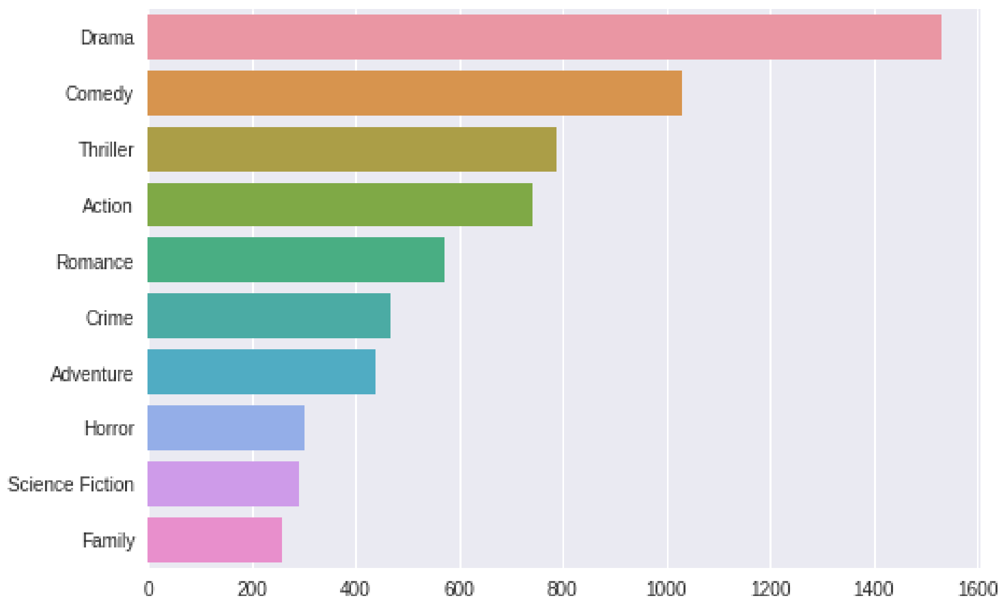

Figure 5: top genres

Drama is the most common. Comedy and Drama film conquer nearly half of the training set.

2.1.6 Revenues

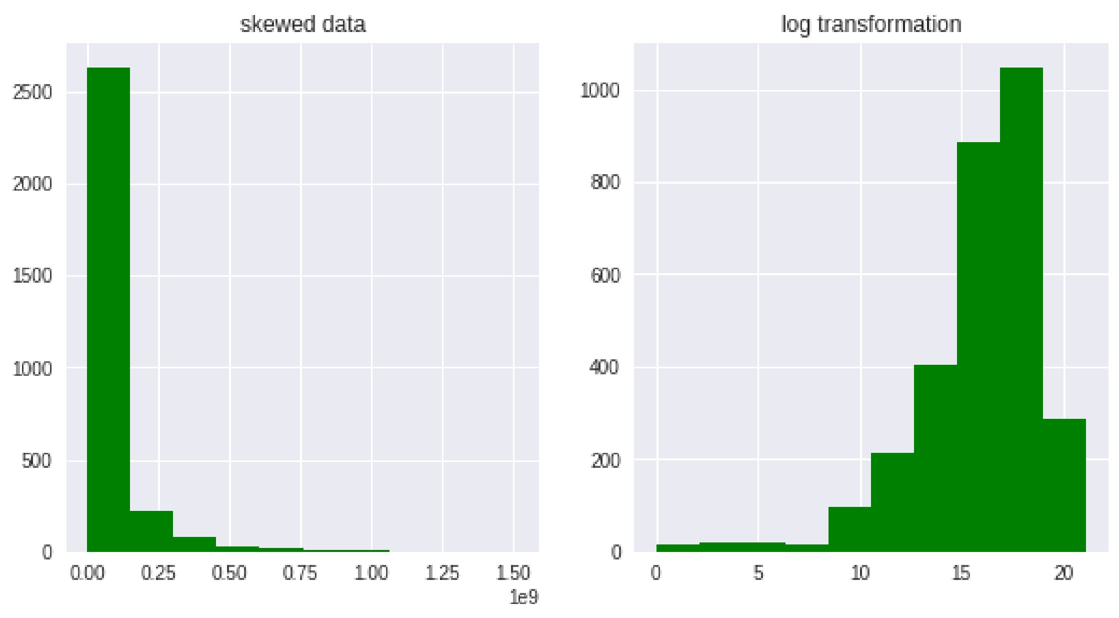

Figure 6: Revenue

The revenue is left skewed. Log tranform may need to apply to make it normal disturbuted.

2.1.7 Budgets

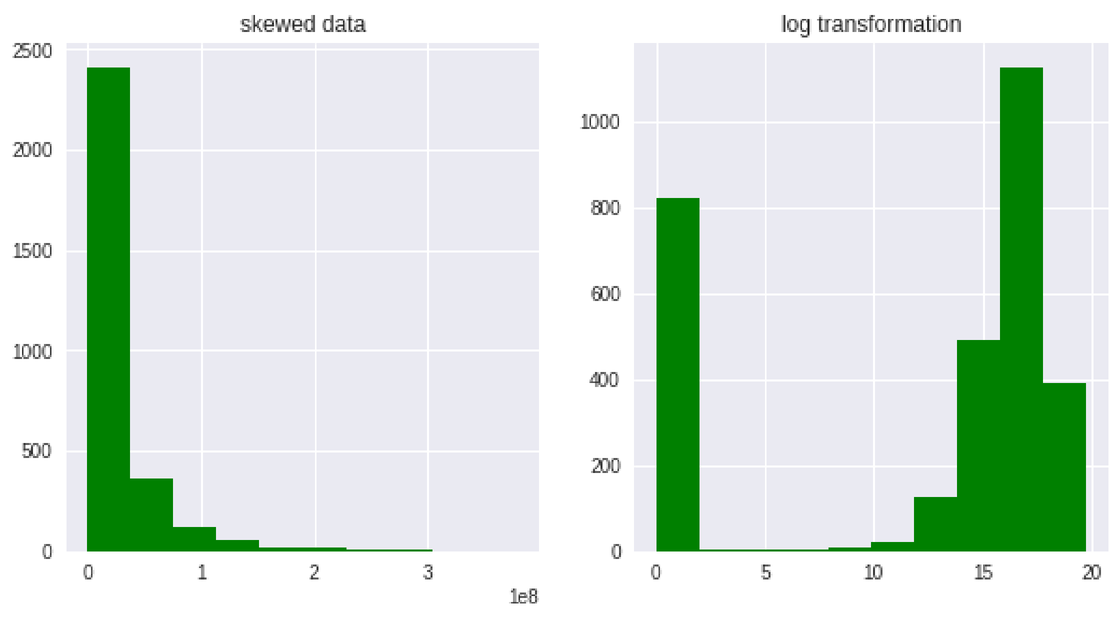

Figure 7: Budgets

A interested effect in budgets is that a larget number of films has zero budgets. It may be a problem if we use budget as the feature input. In addition, the budget is left skewed. Log tranform may need to apply to make it normal disturbuted.

2.1.8 Popularity

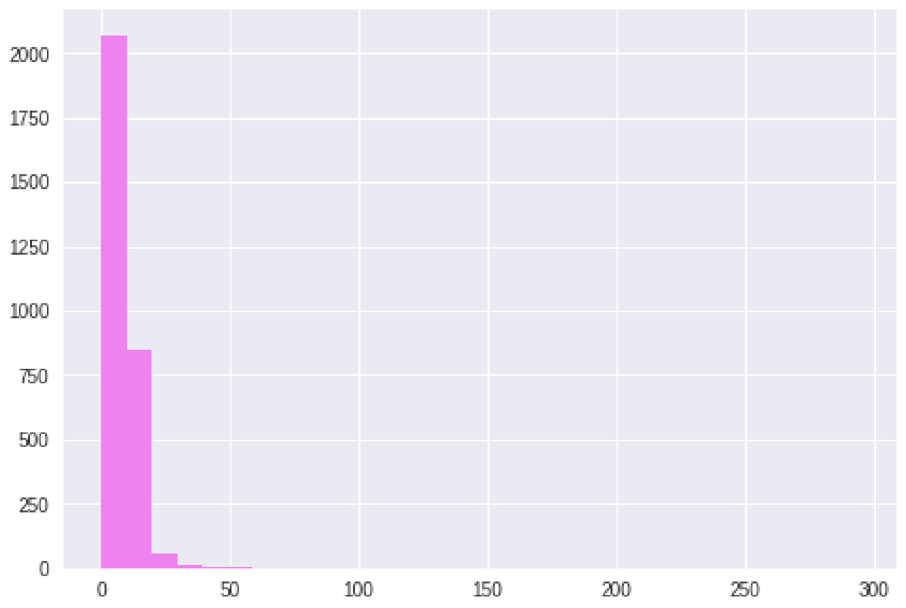

Figure 8: Popularity

The popularity is left skewed. Log tranform may need to apply to make it normal disturbuted.

2.1.9 Budgets Mean by Year

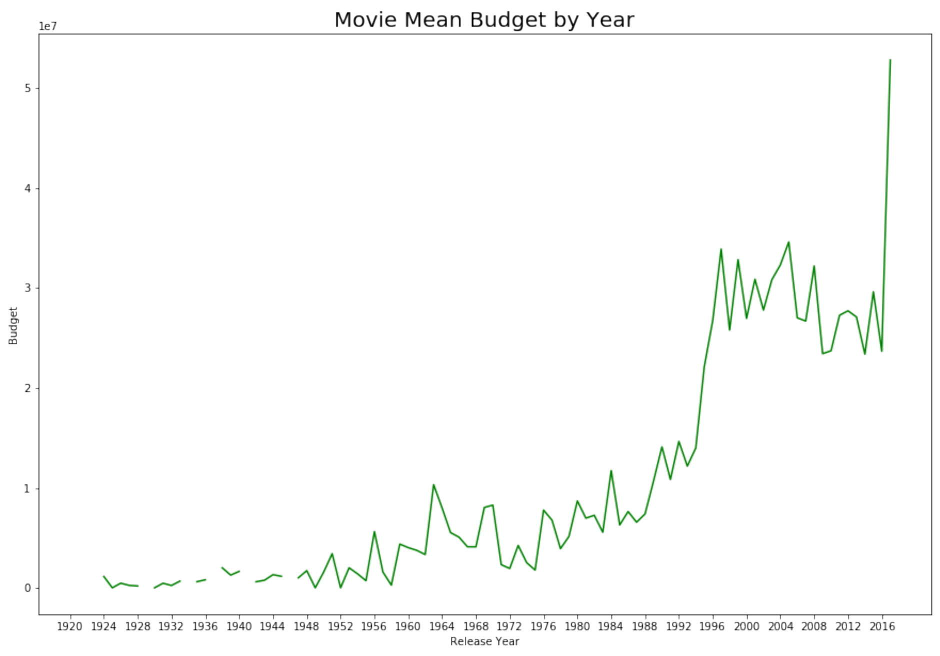

Figure 9: Popularity

We can see the budget is increasing each year. This may be because of inflation.

2.2 Data Preprocessing and Feature Enginerring

Before the data are feeded into neural network, some feature enginerring and data preprocessing are performed to enhance the training performance. And it is mainly composed by four parts (Feature Selection, Data Cleaning, Encoding and Feature transformation)

2.2.1 Feature Selection

In feature selection part, some new features are extracted from the original features in the metadata and some unused features are dropped.

The following are the new features we extracted.

Table 2: new features extracted

New Features	Extracted from Feature
release_day	release_date
release_month	release_date
release_year	release_date
production_countries_count	production_countries
production_companies_count	production_companies
cast_count	cast
crew_count	crew
num_Keywords	Keywords
inflationBudget	budgets

For the release_date, since the original format of release_date is difficult to be input into the neural network. Therefore, three new features, release_year, release_month and release_day will be extracted from release_date.

For the production_countries_count, production_companies_count, cast_count, crew_count and num_Keywords, some experiements from (`https://www.kaggle.com/zero92/tmdb-prediction`) has found out that these new features can boost the performance. And in fact, these extra features can help us to ease down the difficulty of training the neural network for prediction.

For inflationBudget, since the original budgets has not take consideration of the effect of inflation. Therefore, we base on the original budget with Inflation simple formula to caculate the budget of the film after inflation.

The following are the features we dropped.

Table 3: dropped features

Dropped Features	Drop Reason
homepage	url is difficult to use
status	constant feature
original_language	almost same as the spoken_language column
original_title	almost almost same as the title column
imdb_id	unique value

After performaning the feature selection, the following features are used for the prediction and we can categories them into four types, namely catatogical, numerical, text and image.

Table 4: selected features after feature selection

Features	Types
belongs_to_collection	categorical
budget	numerical
genres	categorical
overview	text
popularity	numerical
poster	image
popularity	numerical
production_companies	categorical
production_countries	categorical
release_day	numerical
release_month	numerical
release_year	numerical
runtime	numerical
spoken_languages	categorical
tagline	text
title	text
Keywords	text
cast	categorical
crew	categorical
revenue	numerical
production_countries_count	numerical
production_companies_count	numerical
cast_count	numerical
crew_count	numerical
num_Keywords	numerical
inflationBudget	numerical

2.2.2 Data Cleaning

In data cleaning part, the features with missing value are filled and transfer into the easy processable format. In addition, the text features will be cleaned with serveral procedures.

For all the columns which are in the dictionary format in the metadata (belongs_to_collection, genres, production_companies, production_coutries, spoken_languages, Keywords, cast, crew), we extract the value with the key "name" and give up all other things such as id in the dict . For the other columns, we keep them as the original forms as in the meta data.

For all the catagorical features, we fill all the missing value by "!". Moreover, for the scalar columns, we fill the missing value by their median since it is more resistant to outlier.

Moreover, for the data cleaning of all the text features, we first turn all the unicode string into **ASCII** and **lower case**. Then, we **expand the contradiction** and **remove all the characters except (A-Za-z0-9,.!?;)**.

2.2.3 Data Encoding

In encoding part, all the catagorical feature are encoded into integer representation.

2.2.4 Feature Transformation

In feature transformation part, the scalar feature are transformed into the disturbution that are easily trained by neural network. The transofmations of the scalar columns are shown below.

For log transform, it is refered to using applying log to feature. the skew data into normal like disbutrubtion. For cyclic transform, we take a reference from (`https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/`). It used is to encode the cyclical continuous features into encoding with cyclical nature. For standardize, it transforms the feature into mean = 0 and variance = 1.

For image features, they are transformed into grey scale with normalization.

Table 5: feature transformation

Features	Transformations
revenues	log transform, standardize
budget	log transform, standardize
popularity	log transform, standardize
runtime	log transform, standardize
release_day	cyclic transform, standardize
release_month	cyclic transform, standardize
release_year	standardize

3 Deep Neural Network

3.1 Architecture

In the project, we try to apply deep neural network to predict the box office revenues. The archiecture of the deep neural network is composed by four different encoders to encode different types of input features and the main net.

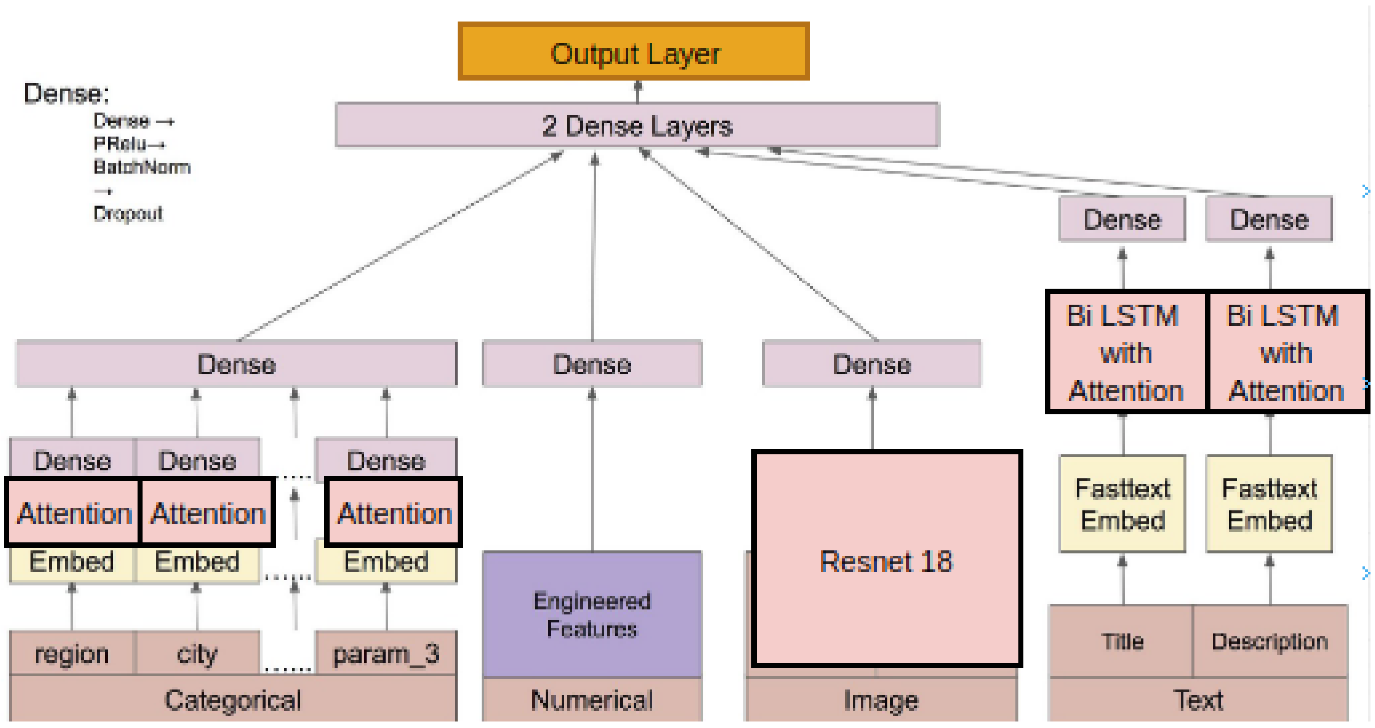

Figure 10: The archiecture of the our deep neural network

3.1.1 Categorical Encoder

The categorical encoder in our network is used to encode all catagorical features. It is composed by an embedding layer to embbed the feature inputs, an additive attention layer to handle the features that can have **multiple values** such as genres. (A film can have multiple genres), and two dense layers to embed all categorical features into a feature vector.

3.1.2 Numerical Encoder

The numerical encoder in our network is used to encode the numerical features. It is composed by a dense layer to embed all the numerical feature in to a feature vector.

3.1.3 Text Encoder

The text encoder in our network is used to encode the text features. It is composed by a word2Vector embedding layer, an Attention Based Bi-LSTM layer and a dense layer to embbed the text feature into a context vector.

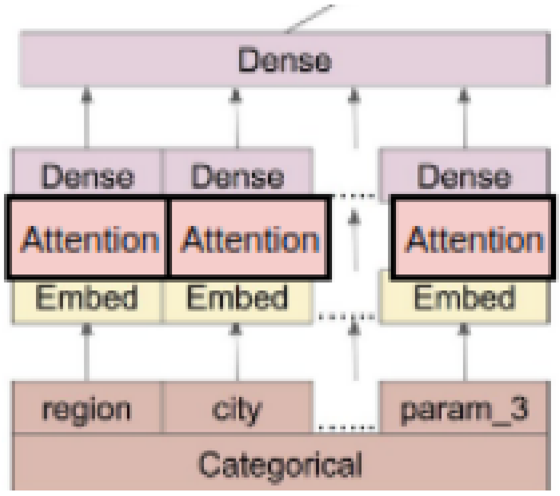

Figure 11: Categorical Encoder

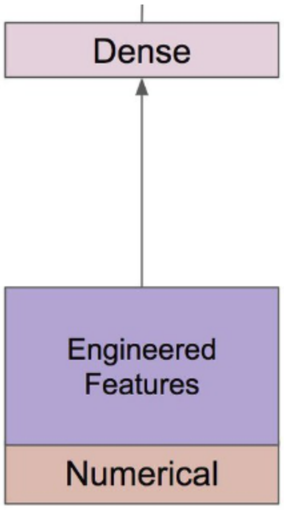

Figure 12: Numerical Encoder

For the word embedding layer, it has been pretrained with **Google News corpus** (3 billion running words) word vector model. (Google News Corpus: `https://github.com/mmihaltz/word2vec-GoogleNews-vectors`). And since the dimension of the embedding matrix is enormously big which cause some memory error during training, we have limited to only use ten thousands of vocabs. All above process can be easily done through a python libarary named **gensim**.

For Attention Based Bi-LSTM, the archiecture of is same as the paper **Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification** (`https://www.aclweb.org/anthology/P16-2034`) which use two LSTM in reverse direction and using addition attention for caculating the final context vector from each of the hidden states in LSTMs.

3.1.4 Image Encoder

The image encoder in our network is used to encode the image feature. It is composed by a pretrained **18 layers ResNet** and a dense layer to embed image feature in to a feature vector.

The pretained resnet can be easily done through a python libarary named **torchvision.models**

3.1.5 Main Net

The main net is used to concatenate all the feature vectors from different encoders and predict the final results. It is compose by two dense layers and a output layer with a hidden unit.

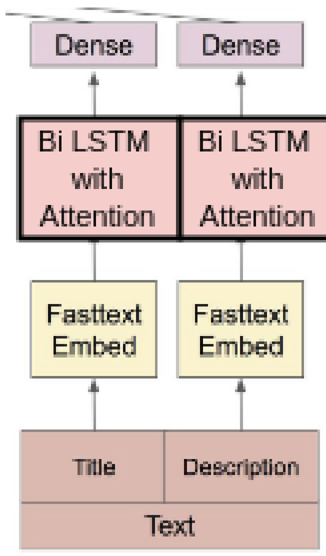

Figure 13: Text Encoder

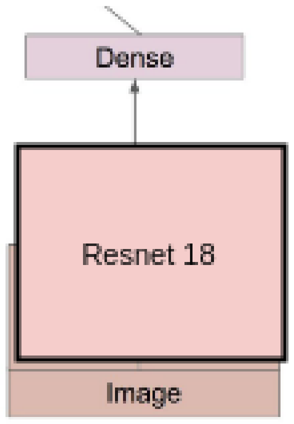

Figure 14: Image Encoder

3.1.6 Dense Layer settings

The dense layer setting in our network are as follow. Except the last dense layer and the output layer, all dense layers in the network is followed by a **Randomized Leaky ReLU (RReLU)** , a batch noralization layer and a dropout layer

Table 6: Dense Layer Setting

Setting	Other dense layers	Last dense Layer in Main Net	Output Layer
Dense	*	*	*
RReLu	*	*	
BatchNorm	*		
Dropout	*		

4 Experimental Setting

For the experiment, we have further splitted the training set into training set and validation set with the ratio 8:2. We use the train using our training set and perform the hyperparameters tuning using the validation set. After performing the tuning, we train the model with the best hyperparameter obtained and test the model on the test set. Moreover, the training will be early stopped when the validation MSE Loss has decraseed for specified number of epochs.

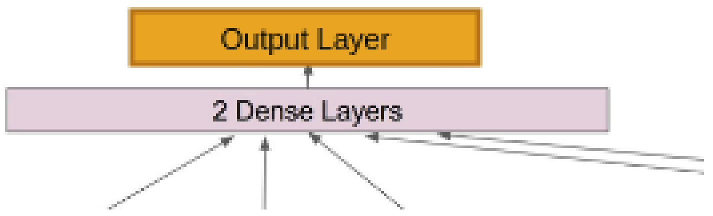

Figure 15: Main Net

4.1 Computational Environment

All the experiment conducted in this project using the following computational environment.

Table 7: Computational Environment

Computational Settings	–
CPU	AMD Ryzen 2700x
GPU	RTX2080Ti
RAM	32GB
OS	Ubututu 16.04
Python Version	3.5
Deep Learning Libarary	Pytorch

4.2 Hyperparameters Tuning

For the hyperparameters tuning, we perform using grid search to find the best hyparameters of drop out rate, learning rate and the size of the number of hidden units in the last dense layer of main net.

4.2.1 Tuning Environments

For the hyperparameters tuning, the following environment are used.

Table 8: Tuning Environment

Tune Settings	–
Epoch	100
Optimizer	Adam
Batch Size	32
Early Stopping	30

4.2.2 Tuning Results

Table 9: Hyperparameter tuning results

Learning Rate	Dropout rate	Number of hidden units in Last Layer	Best Validation MSE Loss
0.1	0.25	200	0.7231
0.05	0.25	200	0.7532
0.01	0.25	200	0.6672
0.005	0.25	200	0.634
0.001	0.25	200	0.6011
0.0005	0.25	200	0.6343
0.0001	0.25	200	0.6314
0.1	0.25	200	0.7356
0.1	0.30	200	0.7422
0.01	0.35	200	0.7245
0.01	0.40	200	0.6035
0.01	0.45	200	0.6422
0.01	0.50	200	0.6135
0.01	0.30	200	0.6314
0.01	0.30	300	0.6456
0.01	0.30	500	0.6042
0.01	0.30	1000	0.7324

Best Parameters Hyperparameter obtained learning rate : 0.001, Dropout: 0.4, Number of Hidden Unit: 500

4.3 Final Training Parameters Setting and Environment

We use the best hyperparameters obtained from tuning and train the model same as the archiecture above with the following parameters settings.

Table 10: Training Parameters Settings

Parameters	–
Epoch	1200
Optimizer	Adam
Batch Size	64
Early Stopping	100
Learning Rate	0.001
Dropout	0.4

5 Results and Analysis

5.1 Performance

The following are the results of the final training with the best hyperparameters. You can find me on the leaderboard with the name fingnn.

Table 11: Score in Kaggle

Dataset	Best Score
Test Set	2.66035

Table 12: Best MSE Loss in Validation Set

Dataset	Best MSE Loss
Validation Set	0.5991

Table 13: Time Used for the experiments

Module	Wall time
Parameter Tuning	7.12 hours
Final Training	1.1 hours

5.2 Learning Curve

The below are the loss curve trained with best hyper-params. The blue curve is the validation MSE Loss. The orange curve are the training MSE Loss Curve.

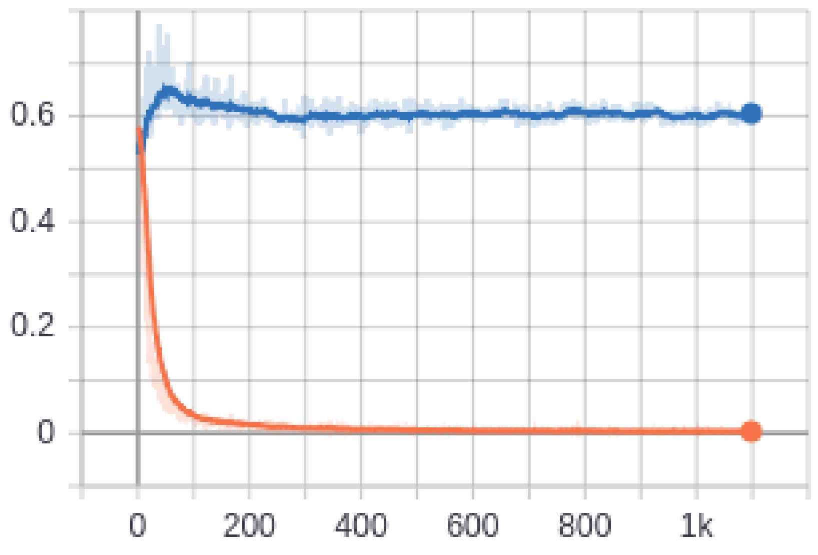

Figure 16: MSE Loss Curve

From the results, we can see although we have apply dropout layer in the neural network but, it is still fail to lower the validation MSE Loss too much. On the other hand, for the training MSE loss, it is able to converge to minma. Therefore, although we got a very low training MSE Loss but still got a normal kaggle score.

5.3 Comparing the results to others kaggle participants

Comparing the results to the model of other participants such as `https://www.kaggle.com/akumaldo/tmdb-prediction-eda-lgb-xgb-cat-plus-keras-nn`. We can see, in fact our model is not very bad.

Table 14: Results of other participants

Model	Best Validation MST Loss
xgBoost	2.0656
keras NN	5.4218

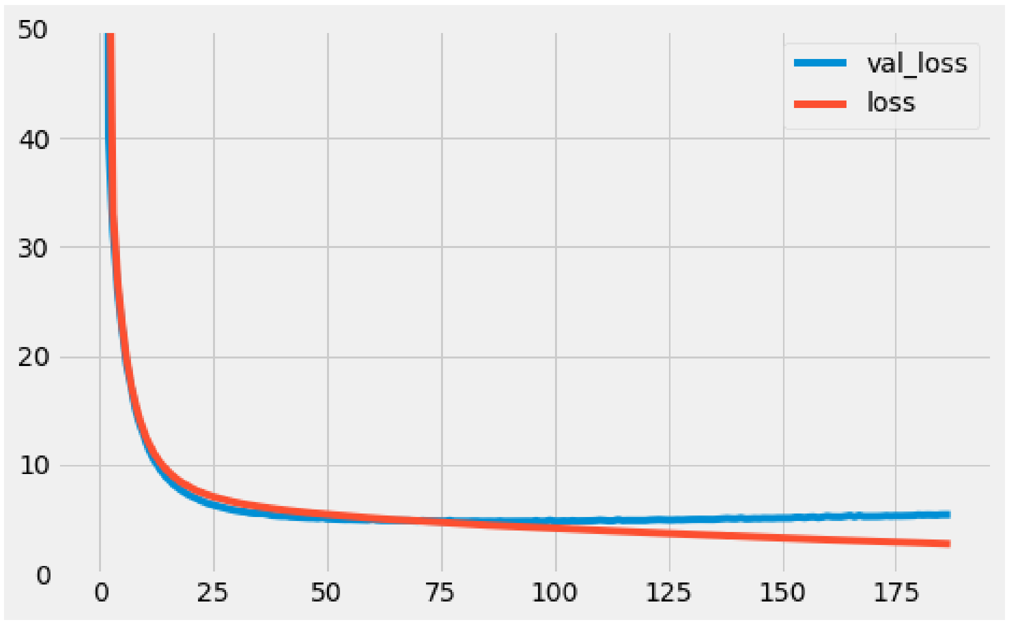

Figure 17: MSE Loss Curve of the Keras NN implemented by other participants

For xgBoost, we can see it done very well in regression problem. Its best validation MST loss is much lower than keras NN and our model. However, the performance for its keras NN is quite similiar to our deep neural network. Moreover, we can also see that others also face the similiar overfitting problem in training NN for solving the problem.

5.4 Visualization of some Gradient Histogram

In the gradient histogram, we can see the even we trained the network for 1100 epoches. The network does not have any gradient vanishing or exploding problem. All Layers contains non-zero gradients. (More Gradient Histogram can be found on running the tensorboard in logs folders.

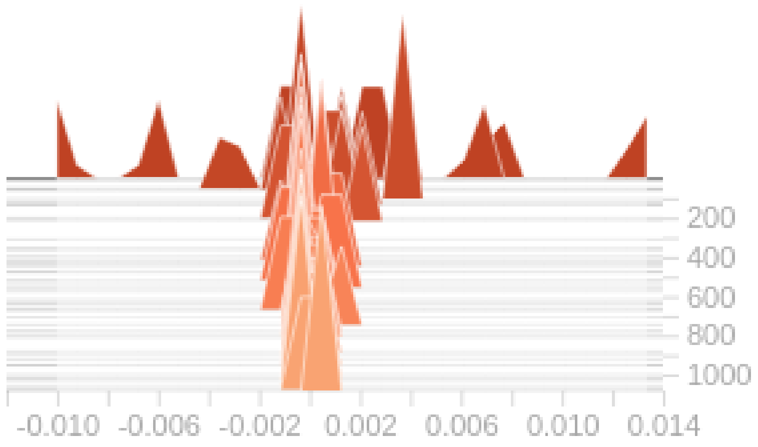

Figure 18: Gradient Histogram of Genres Layer

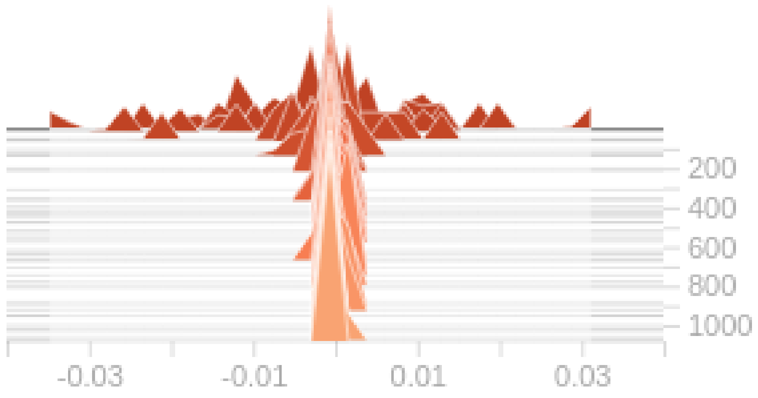

Figure 19: Gradient Histogram of Production Countries Layer

5.5 Some samples from prediction

5.6 Problems

First, it is difficult and time consuming to tune the parameters when we apply deep neural network in the regression problems. Since, unlike classification only having limited numbers of labels, for regression, the neural network need to fit so many values. Therefore, it is difficult for the neural network to converge. That is why, xgbost and lightgbm are more common and popular in kaggle competition.

Moreover, training dataset provided by the kaggle is too small and even smaller than the test set. Therefore, it is difficult to achieve a good results without using any external data sources.

Last but not last, some data in test set having the labels that do not appear in the training set which make our model difficult to predict the data with a lot of unseen labels.

5.7 Future Development

To enhance the model perforance, we can replace Attension Based Bi LSTM to more advanace model like transformer.

Also, we can apply some clustering techique such as k-mean clustering to cluster the similiar data in order to fill in the missing value.

Finally, more feature enginerring can be applied to the model in order to get better platformance. We can see from the competition discussion, most of the participant perform a lot of feature enginerring works other than improving the structure of the model.

5.8 Conclusion

In this project, we have explored to using deep learning to solve the regression problem in kaggle. Alrough this is difficult, it is funny.