Information Systems
and Machine Learning Lab
Stiftung Universität Hildesheim
Universitätsplatz
31141 Hildesheim
Prof. Dr. Dr. Lars Schmidt-Thieme
Mofassir ul Islam Arif

Box Office Return Prediction (BORP) using Movie Trailers

Student Research Project 2019-2020

Muhammad Mehmood Ali
Raaghav Radhakrishnan
Diana Artiom
Kalaiselvan Panneerselvam
Muhammad Inzimam Saghir

Abstract

Predicting movie success in early stages is a topic of great concern for production companies. A reliable box office return prediction before the theatrical release can significantly reduce financial risks. Existing approaches do not leverage the available information to full extent, being only based on information from movie metadata, postal marketing material, and/or user interaction in the form of comments and feedback. In this work, we predict box office return using both the movie trailers and the movie metadata information. To balance the information extracted from trailer video frames and computational feasibility different frame sampling techniques have been considered. Additionally, we apply different augmentation techniques to compensate the limited size of the dataset. To preserve the temporal information in the trailers, we consider two different neural network models: Long-term Temporal Convolutions (LTC) and Long Short-Term Memory(LSTM). Being compared against existing approaches such as Average Model, Linear Regression and Multi Layer Perceptron (MLP), both proposed architectures (LTC and LSTM) outperform the classical approaches and prove that leveraging information from video trailers help improve the prediction.

Keywords: Data augmentation, LSTM, LTC, Metadata, Movie box office revenue, Movie trailers.

Contents

1 Introduction

As a medium of entertainment, movies have become an integral part of human lives. Movies are a multibillion-dollar industry. Thus, the movie industry has become a promising sector for big investments and has made a huge market profit. It is always challenging for investors to decide whether to invest in a movie or not. Box office returns deal with this complex problem of forecasting the financial success of movies before the movie gets screened. The ability to accurately predict the box-office returns will help the producers reduce the financial risk and maximize their profit.

Many factors influence the success or failure of a movie in terms of the box office. Movies' success depends on different factors, including storyline, script, screenplay, cinematography, cast & crew, people critics and reviews to name but a few [2, 3, 4]. Although movie success has been considered an unpredictable problem [5], many researchers have attempted different approaches for movie box-office returns prediction [5, 6, 7]. In the similar and previous researches and works, in order to predict the box office revenue, only the movies' metadata were used to train the deep neural networks. These metadata are publicly available in IMDB, TMDB and Box Office Mojo. In our study, we proceeded a step further in predicting the box office revenue by employing more complex data, such as movie trailers. By using the movie trailers, we combine the temporal information from videos with the metadata information to predict the box office revenue.

Movie trailers represent data that could influence profits from film box office. Being an advertising medium, movie trailers are intended to deliberately express film material and capture prospective audience interest. As a consequence, spectators often recognize movie trailers soon before the theatrical launch of the film. Movies are an interesting market, since more and more of them are produced every year. In Figure 1 it can be seen the amount of movies released each year since 2000. We see an increasing trend in the number of movies released every year. This makes the movie area a domain

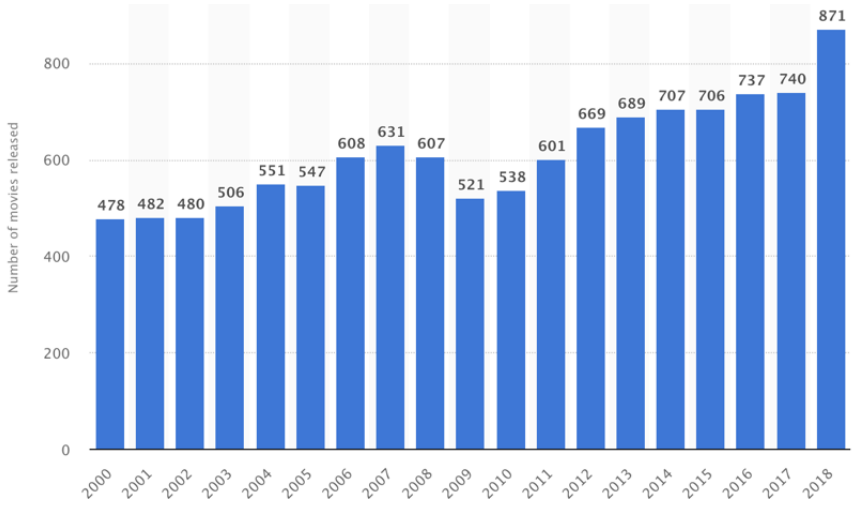


Figure 1: Number of movies released in the U.S and Canada between 2000 and 2019 [8]

of increased financial concern. Many of the trailers are of these movies are openly accessible in the internet on websites like YouTube. The quality information hidden in the movie trailers and their impact on the audience are the obvious reasons for our investigation on the impacts of trailers on movie box office returns.

Neural networks are the state of the art methods for many applications including information extraction, image classification, object detection and text classification to name but a few. Convolutional neural networks (CNN) and Recurrent neural networks (RNN) are widely used for different applications. CNNs excel at capturing still features, but recent study has shown that Long term Temporal convolutions (LTC) [9] perform better for action recognition tasks. In that paper, the LTC model extracts temporal information between the frames to predict the action performed in the video. On the other hand, Long Short-Term Memory (LSTM) [10] approaches are used to retain the history of information from the sequence of data. We aim to achieve our goal, the box-office return prediction, by extracting quality and useful information from the movie trailers using both the LTC and LSTM network architectures. Further on, with both the models, we include the

metadata with documented movie revenues to train the regression model and predict a better box office revenue. The main contributions of our paper are as follows:

1. Analysed and cleaned the dataset to get a better correlation between the target and the predictor variables.

2. Used a frame selection approach to filter the required number of frames and augmented the trailers with different frame filtering.

3. Implemented the baseline architectures of LTC and LSTM and conducted experiments with the same.

4. Merged the features extracted from the trailers using LTC and the metadata features and were further regressed to predict the box office revenue.

5. Extracted the temporal information from the frames using LSTM that required the embeddings from a pretrained feature extractor - a ResNet [11] network. These features are merged with the metadata features and were further regressed to predict the box office revenue.

6. Tuned the hypermaters to get an optimal combination of the parameters and conducted the experiments in different environments for different models.

7. Implemented different regression and boosting methods to compare the results and performance of the model.

8. Tabulated and analysed the experimental results and loss graphs are plotted and discussed to get a better understanding of the models.

The rest of the paper is organized as follows: Section II discusses the related works; section III describes the proposed methodologies; section IV discusses the dataset collection and description with insights; the experiment and evaluation are discussed in section V and section VI discusses and concludes the research paper and provides future research directions.

2 Literature review

In this section we provide an outline of current research done in the area of box office return prediction or tasks related to it.

Predictive models for estimating the movies' success have been of high interest ever since movies industry has started. Research has been done in different directions. One of the most popular databases for predictive models over movies is the Internet Movie Database (IMDB) [12]. In [13], Kim et al. propose model which uses features such as MPAA ratings and budget, as well as different representations of star power in actresses and actors for box office prediction. Similar approaches are considered in [14, 15]. Movie performance is also considered against critical reviews of the movies [16, 17, 18]. Other approaches are using linear models to predict the box office return based on social media interaction [19]. A similar approach is presented in [20], where the authors focused on the hype of twitter for prediction and found that the opening weekend income and the hype the movie received from audience mainly influenced the success. In [21], the authors used seven different classifiers to predict an approximate net profit value of a movie by analyzing historical data from different sources like IMDB, Rotten Tomatoes, Box Office Mojo and Meta Critic. A different approach based on social media interaction was introduced by Apala et al. in [22]. In this work the authors used unsupervised learning, with K-means clustering. They predict the box office of movies from different social media such as Twitter and YouTube comments and sentiments.

Some approaches considered predicting box office return using movie posters. In [23], a multimodal deep neural network is proposed. The authors used convolutional neural networks to extract the features from movie posters and predicted the box office returns. The same goal is achieved in [24], where the authors use used an Inception-V3 for feature extraction from posters and prediction of box office return.

Other approaches transform the box office return regression problem into a classification problem. A similar approach is taken in [5], where a neural network was implemented for the movie box office gross prediction. They categorized the net profit into 9 different bins and made it a classification problem.

Important work is done in the area of predicting movie attendance. In [25], the authors use the online reviews and rating, and present the Hierarchical Bayes model, which consists of three models which are regression model, standard logit model and nested logit model. In [26], the authors present a system that uses historical data to forecast new movie attendance. To achieve this they use two models: The Bass, which explains adoption through innovation and imitation effectsy The Sawhney and Eliashberg, which characterizes the adoption process through time-to-decide and time-to-act effects. An approach for predicting movie attendance model that considers attendance seasonality (weekends and holidays) is presented in [27]. The authors demonstrate that the intertemporal demand shift results in weekend fluctuations, while the extra demand causes the seasonal holiday effect.

One of the most remarkable works in this area of predicting attendance with movie trailers is done by a joint research group from Google and 20th Century fox. In [28], the authors make an analysis on how movie trailers help to predict the movie attendance. In [29], the authors use a hybrid Collaborative Filtering model based on Deep Neural Networks (DNN), that combines video features with historical movie attendance records. The use of DNN based models extends the capability of such hybrid models and allows them to learn deeper insights from the content [30, 31].

3 Background

3.1 Neural Network

The idea of neural networks is based on how the mind analyses and perceives information in our day-to-day life. Explanation of brain processing by Warren et.al in 1943 using linear threshold gates became the base of Artificial Neural Networks (ANN) and after extensive efforts in 1959, Bernard created the first Neural Network for real-world problems. The whole idea became redundant in the '60s and '70s. But in the '90s, neural networks became a hot topic which lead to the deep learning approaches in today's world. Neural Networks are widely used for modeling and predicting complex linear models because of their self-learning and adaptive capabilities. Unlike statistical models, NN relies on observed data rather than an analytical solution. The parameters of NN are solely determined by the dataset. There are different types of NN's, like Convolutional Neural Networks (CNN) often used for vectorial data; Recurrent Neural Networks (RNN) used for capturing connections in sequenced data; and many others. For extracting video information, a standard 2D CNN will not be able to capture long term dependencies. They can only capture a fixed size of window sequence and are unsuitable for historical dependencies. Hence, 3D CNNs can be used instead of 2D CNNs. On the contrary, Recurrent Neural Networks are more promising in handling temporal dependencies. This section briefly discusses the neural network architectures used in this project work.

3.2 Convolutional Neural Network

A Convolutional neural network is a feed forward neural network that is primarliy designed to solve problems related to image recognition, image classification and object detection. Unlike general matrix multiplication in feed forward neural network, these models use convolutions inplace. Usually, for these tasks, 2D convolutions are used as they are invariant to translation and has an advantage of weight sharing. A 2D CNN is useful when a spatial relationship between the objects' components is considered. For the temporal

or spatial relationship in a 3D space, 3D CNNs play a major role. The 3D filter moves in 2-direction (height and width of the image). With videos, these 3D CNNs tend to have different applications including action detection, 3D segmentations or reconstructions of biomedical images to name but a few.

3.2.1 Long term Temporal Convolution (LTC)

The current CNN methods for action recognition learn representations for short intervals at a maximum of 16 frames. But the typical human action lasts for several seconds and contains characteristic patterns with specific long term temporal structure. The video representations are learned using 3D CNNs with Long term temporal convolutions (LTC).

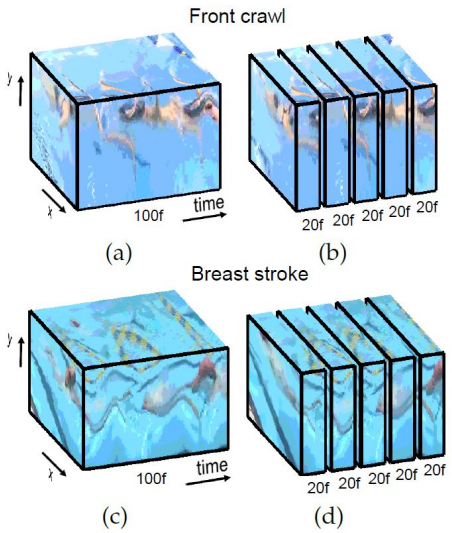


Figure 2: Video patches for two classes of swimming actions[9]

In figure 2, (a) & (c) represents actions that contain characteristic of front crawl and breast stroke respecitvely, patterns that last for several seconds. (b) & (d) shows that splitting videos into short intervals is likely to destroy such patterns making recognition more difficult.

LTC performs action classification based on short videos. It splits video into multiple channels of smaller size (smaller number of frames). LTC explores the impact of the length of the input video on the 3D convolutional network on the recognition effect, and proposes the 3D network structure, as shown in the figure 3.

The proposed network has 5 space-time convolutional layers with 64, 128, 256, 256 and 256 filter response maps, followed by 3 fully connected layers

of sizes 2048, 2048 and number of classes. The convolution kernel is 3x3x3, each convolution layer uses the ReLU activation function, and max-pooling layers inbetween the convolutions(except the first layer is 2x2x1, the others are 2x2x2). All the three dimensions are padded with one pixel, and the con-

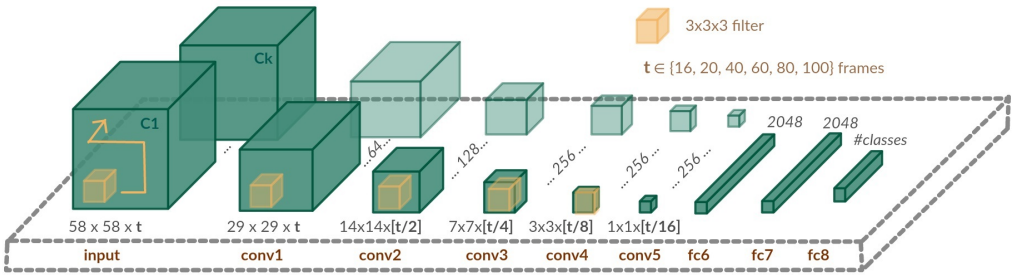


Figure 3: LTC network architecture[9]

volutions with 1 stride and 2 strides for the pooling. The pooling layer is activated by the ReLU layer. By the end of the fifth convolutional layer, the features are made to a dense fully connected layer. The first two fc layers use dropout with a ratio of 0.5. The final fc layer is activated by softmax to predict the action. This network provides an increased temporal extent by the cost of decreased spatial resolution.

3.2.2 ResNet

In the ILSVRC 2015[32] and COCO 2015[33] competitions, ResNet is one of the deep neural networks that showed outstanding performance results. Based on the number of layers, the ResNet is classified to different variants. The basic concept of the deep residual network and the architecture of the ResNet-152 used for the project are discussed below.

The primary contribution of deep residual network is the residual block but it is similar to the other networks with basic convolution, pooling, activation and fc layers. Residual blocks introduces skip connections with identity mapping to the end of the blocks. Skip connections with identity mapping

is denoted as X as shown in figure 4. The introduction of residual blocks addresses one of the main problems of deep neural network architectures, the vanishing gradients. In a deep neural network, vanishing gradient is common that during back propagation, the gradient gets smaller and smaller that it vanishes at the input layer. This problem of vanishing gradient is solved by back propagating through the identity mapping thus skipping the residual blocks and help them learn the correct weights. The output of the residual block is as shown in the equation.

$$H(x) = relu(F(x) + x)$$

The common idea behind every ResNet architecture is that they have 4 stages in common. Similar to other variants, the ResNet-152 inputs a 224x224x3 image and performs a convolution and pooling of kernel sizes 7x7 and 3x3 respectively. This is followed by 3 residual blocks and with ResNet-152, more 3 layers are stacked one over the other to get a bottleneck design. Finally, the last stage has an average pooling followed by 1000 fully conected neurons that are activated by a softmax function to ge the class output. The problem of covariate shift and the performance of the network is improved as it uses Batch Normalization as a regularizer and a bootle neck design in the deep network architectures.

3.3 Recurrent Neural Network

A Recurrent Neural Network (RNN)[34] is a type of neural network that model sequence of input data. In a traditional neural network or a CNN, the output is independent of the input, but in a RNN, the output of a previous step is fed as input to the current step. The ability of the RNNs to remember the history of information with the hidden state plays a major role in modelling the sequences. According to the original

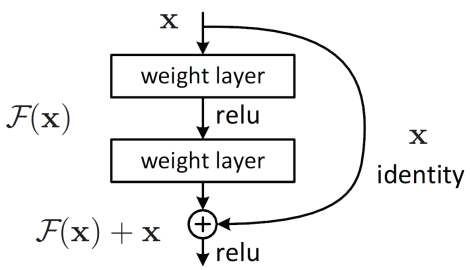


Figure 4: A Residual block of the Deep Residual Network[11]

paper[35], the formula for the current state is defined by

$$h_t = f(h_{\text{t-1}}, x_t)$$

Applying the tangent activation, the above equation becomes

$$h_t = tanh(w_{\text{hh}} h_{\text{t-1}} + w_{\text{xh}} x_t)$$

where h is the hidden state vector, x is the input at time t, W_{hh} is the weight at previous hidden state, W_{hx} is the weight at current input state. Then, the output is given by

$$y_t = w_{\text{hy}} h_t$$

where y_t is the output state. w_{hy} is the weight at the output state. However, training RNN with back-propagation through time (BPTT) technique is difficult due to vanishing gradient problem. Due to this problem, RNN's are unable to capture long term dependencies. The Long Short Term Memory (LSTM) models proposed solve this problem by introducing a new cell state and having a constant error carousel, which allows the error to backpropagate without vanishing.

3.3.1 Long Short Term Memory (LSTM)

In 1996, Schmidhuber et al. in [10] proposed LSTM as an alternative method of recurrent neural networks in order to combat the vanishing gradient problem. It solves the problem by utilizing the gating mechanism over an internal memory cell. The gates enable the LSTM to determine which data to be stored or discarded in the next cell state.

LSTMs, unlike other networks, preserve and maintain the errors so that they can be easily backpropagated through layers. This allows the RNN to continue to learn and train the model over a vast number of time steps. LSTM uses analog gates to store the information. They contain information besides the regular flow of the RNN. The sigmoid function implemented with element-wise multiplication is used in the range of 0-1 in the analog gates.

The implementation of analog over digital function makes them suitable for backpropagation since they are easily differentiated. Every individual gate acts on the signals once they receive them, then this information is blocked or passed on based on its strength. These information signals are also filtered with the weight of each gate. These weights act similarly to the input and hidden states of the neural network as they are adjusted throughout the RNN training process. Through the learning process, the cells learn when to allow data to enter, be deleted or leave through the iterative processes.

LSTM networks comprise of memory blocks called cells (shown in Figure 5). Each cell has two different states: the cell state and the hidden state, which are transferred to the next cell. The three major gates responsible for the history of information and manipulations to the memory are described below.

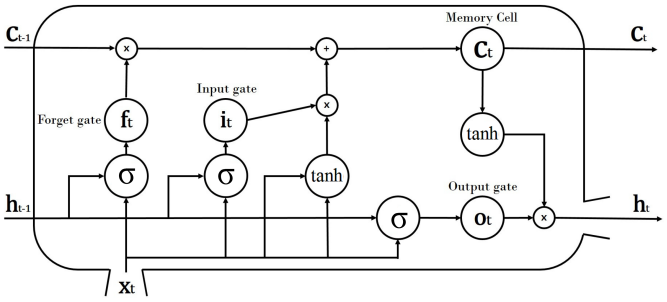


Figure 5: LSTM network architecture

Input Gate: The input gate is used to update all the important and non-redundant information to the cell state. The process of input gate is similar to the forget gate. First, the previous hidden state and current input are passed into a sigmoid function. The results are then passed through a tanh activation function which creates another output vector with values from -1 to +1. Then the output vector is multiplied with the sigmoid output. Like before, the sigmoid output will decide information to be retained. The equation for the input gate is as follows:

$$i_t = \sigma(w_i[h_{t\text{-}1}, x_t] + b_i)$$

where i_t represents input gate, σ is the sigmoid activation function, w is the weight vector, h_{t-1} is the output of previous LSTM block, x_t is input at time t , and b_i is the bias.

Forget Gate: This gate is designed for removing information from the current state of the cell. In simple, it decides what information should be thrown away and what should be kept. The signal from the last hidden state and information from the current step is passed through the sigmoid function which results in a value between 0 and 1. Subsequently, information with the sigmoid output closer to 0 gets discarded and on the other hand, closer to 1 is fed to the next step. The equation for the forget gate represented by f_t is as follows:

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

Output Gate: The output gate decides the selection of useful information from the current cell state and the next hidden state. First, the previous hidden state and the current input is passed into a sigmoid function. Then the newly modified cell state is passed through the tanh activation function. The resultant tanh output is multiplied with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state and the new cell state which is then carried over to the next time step. The equation for the output gate represented by o_t is as follows:

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

4 Dataset

Since a compound dataset, including both videos and metadata is not publically available, the first step in providing a methodological solution for our objective is collecting and forming the dataset. In this section we introduce our strategies to dataset collection and processing for both metadata and video trailers, and provide an exhaustive exploration and interpretation of the data features.

4.1 Dataset collection

Initially, we crawled data from different sources such as The Movie DB, IMDB, Metacritic via Open Movie DataBase(OMDB) API. The architecture of our data extraction model is presented in Figure 6. We used python scripts with beautifulSoup framework for scraping and aggregated movie statistics for the top 100 Movies in the year 2000-2019 with features such as opening gross($), theaters, etc. We choose movies with genre types such as Action, Adventure, Crime, Comedy, Drama, Family, Horror, Romance, Science Fiction, Thriller. Our dataset included top 500 actors & directors in terms of total gross & number of movies directed or featured. But there were a lot of missing values in the dataset. So we decided to use the open dataset obtained from the Kaggle for our experiment. Automated scripts were implemented to fetch and download the movie trailers. From the total metadata, based on the filtered trailers, the data points are further processed and used for training the model.

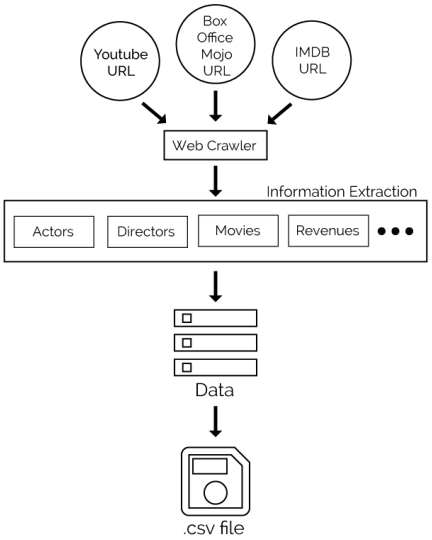


Figure 6: Data collection framework

4.2 Metadata

The film metadata collection is created by aggregating information from various sources. The dataset contains all related information for most of the movies that are released world wide between 1920 and 2018. A total of 3000 movies with complete information from the Kaggle dataset is considered.We have used the following data sources in order to obtain movie metadata:

4.2.1 The Internet Movie Database (IMDB)

This is one of the most famous resources on all kind of movies related data. It has information for about 360 759 feature movies and approximately to over 8 million people are involved in this industry.There is unfortunately no official API for systematically accessing the information. Therefore we mainly use IMDb as an additional source to check our data. Also we use IMDb's box office and budget information.

4.2.2 The Open Movie Database (OMDb)

It is a free database in order to access movie related data. In contrast to IMDB, it is a non-profit organization with a complete and well documented with free API to get the required information for training the model. It is the primary source that we have used for our movie metadata.

4.2.3 Box Office Mojo

Box Office Mojo is a website for tracking box office revenue that is used in film industry worldwide. It records the box office information on weekly basis for US and Canada. Additionally,it occupies information for approximately 50 other territories for most famous movies.

After collecting the data from all the mentioned sources, the final dataset is processed through several steps such as removing data for which movie trailers are not available, irrelevant videos, missing values e.t.c as shown in Figure 7. Movies are labeled with id. Data points include cast, crew, plot

keywords, budget, posters, release dates, languages, production companies, and countries.

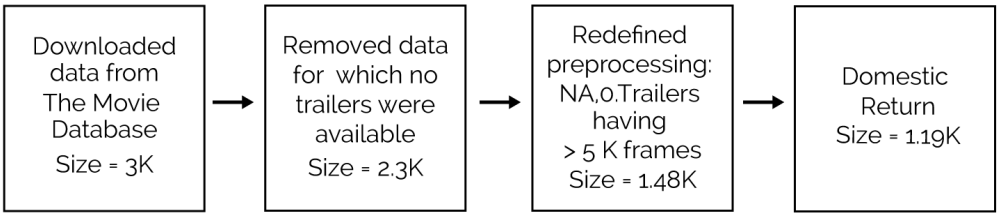


Figure 7: Metadata collection process

4.2.4 Dataset Insights

Understanding of the data an important part in our research. Therefore, by applying the feature engineering techniques, we are exploring some compelling insights. These insights helped us better understand the domain and also refine the data.Most prominent insights, among the other features, are the following:

1. The distribution of revenue and the budget does not follow the normal distribution. To resolve this issue, we removed the anomalies in the dataset. We discovered that few movies had zero revenue, and in some cases, there were no budget values.

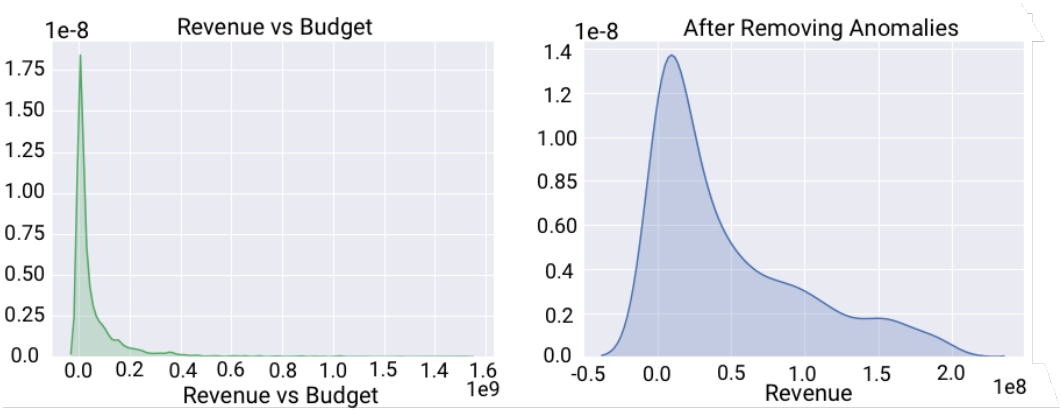


Figure 8: Revenue distribution

2. We studied the correlation between revenue and other features such as budget, popularity, and runtime. There is a strong correlation between budget and revenue. But there is a moderate correlation between budget and popularity, as shown in Figure 9. The relation between budget and revenue are more or less linear with some outliers.

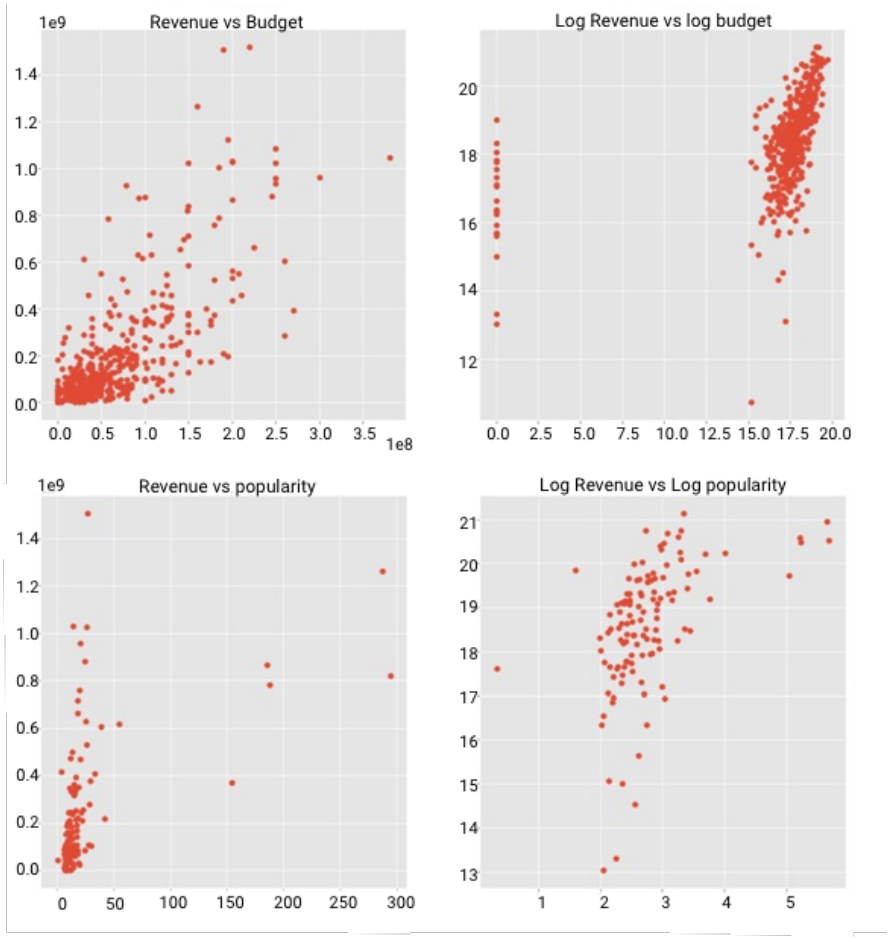


Figure 9: Revenue vs Budget and Popularity correlation

3. We have observed the pattern of having a website tend to generate more revenue comparative to others, as shown in Figure 10 (a). Similarly, if the movie is part of some collection series, then they will have a more extensive fan base and will tend to have more revenue. Furthermore, if the movie is released on Wednesday, will have chances to earn

more compare to other days, as shown in Figure 10 (c). Figure 10 (d) indicates revenue on movies based on the number of featured genres. We can observe that movies with more than three genres tend to have higher revenues.

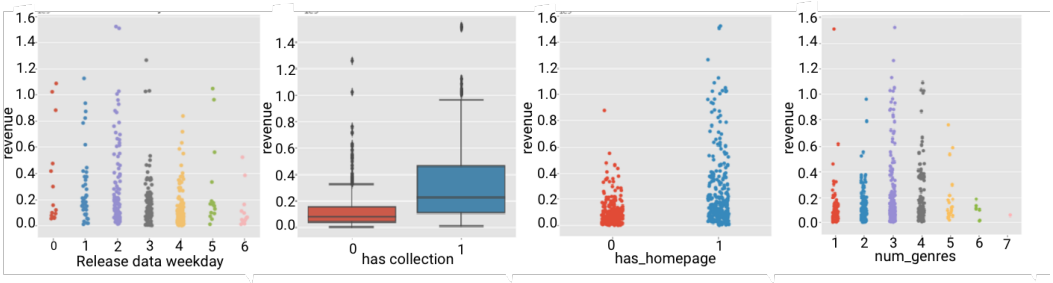


Figure 10: Impact of other features on revenues

4. The top genres are Drama, Comedy, Action, Horror & Adventure in terms of revenue. The revenue based on different genres is shown in Figure 11.

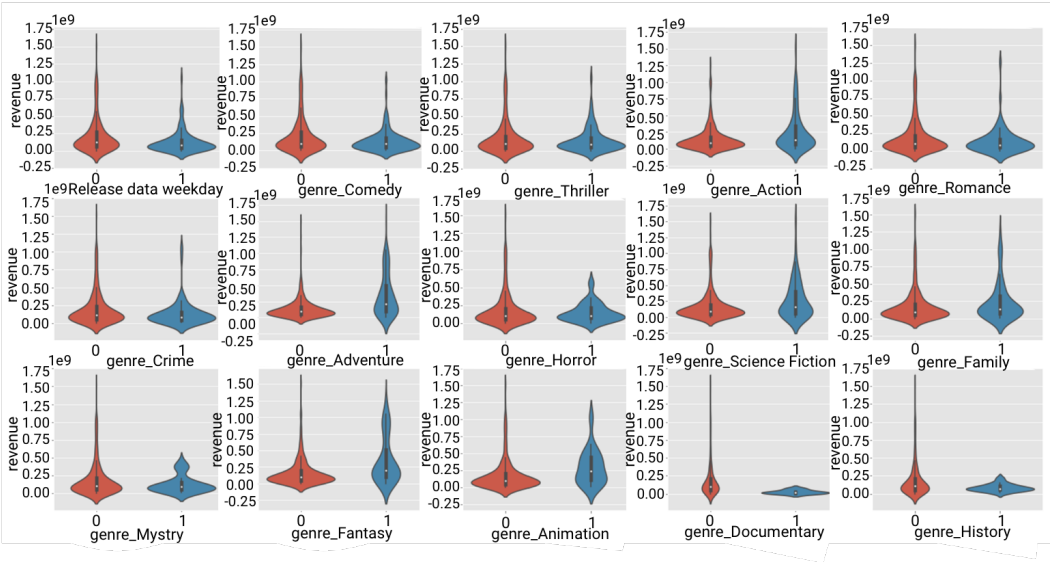


Figure 11: Genre vs Revenues

After exploring the data more, we discovered that the revenue is higher when more than 2 and less than 5 production companies are involved in the project.

Another insight we noticed that maximum revenue is generated when the runtime is around 150mins.

4.2.5 Features Extrapolation

Some features alone don't give us a lot of information —especially the features which are dependent on time and currencies. Currencies are very directly dependent on factors, such as economic situations at that particular time and on depreciation factor. For example, if some movies in the past and now had the same x amount of budget should not be considered the same because of the difference in the economic situation, we can't directly evaluate the value of the amount in the past with current. To accommodate this factor, we gather information from the existing features. In the existing dataset, we have the budget and release year so that we can estimate the average budget cost of the movies for each year, which will give as the average spending rate for each year. By considering this factor, we are calculating the mean year budget ratio and budget year ratio, which will provide us with a relative ratio according to the budget of the movies released in that specific year.

```
# computing budget/year ratio
budget_year_ratio = budget/(year * year)
mean_year_budget_ratio = budget/groupby("year", "budget").
    mean()
```

Listing 1: Computing budget/year ratio

Listing 1 shows an implementation of the aforementioned. We have observe in the below Figure 12 the gradual increase of budget spending each year.

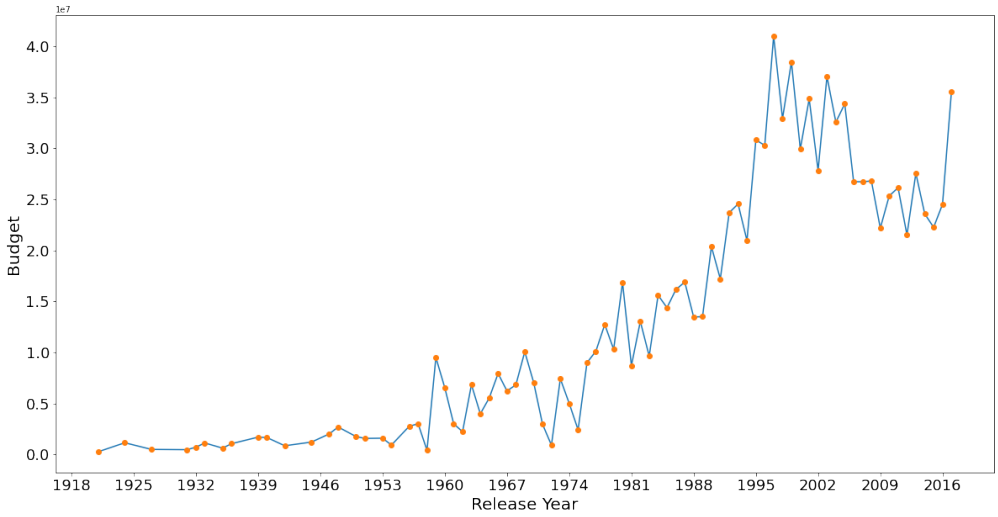


Figure 12: Mean budget per each year

Similarly, the same technique is applied on the popularity index. By considering the average popularity of each year, we are calculating the mean year popularity ratio and also creating a budget to popularity ratio, as shown in Listing 2. The yearly mean popularity spread is shown in Figure 13

```
# computing popularity/year ratio
budget_popularity_ratio = budget / popularity
mean_year_popularity_ratio = population / groupby("year","
    popululation").mean()
```

Listing 2: Computing popularity/year and popularity/year ratios

We can see the loss comparison between the features with and without feature extrapolation in Figure: 14. There is a noticeable decrease in the loss by creating new features from the existing features.

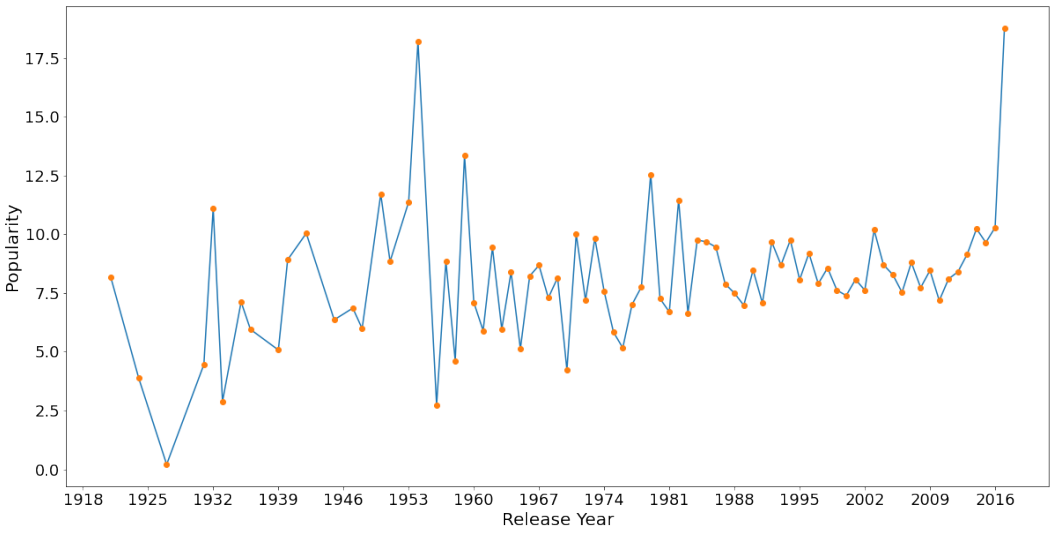


Figure 13: Mean popularity per each year

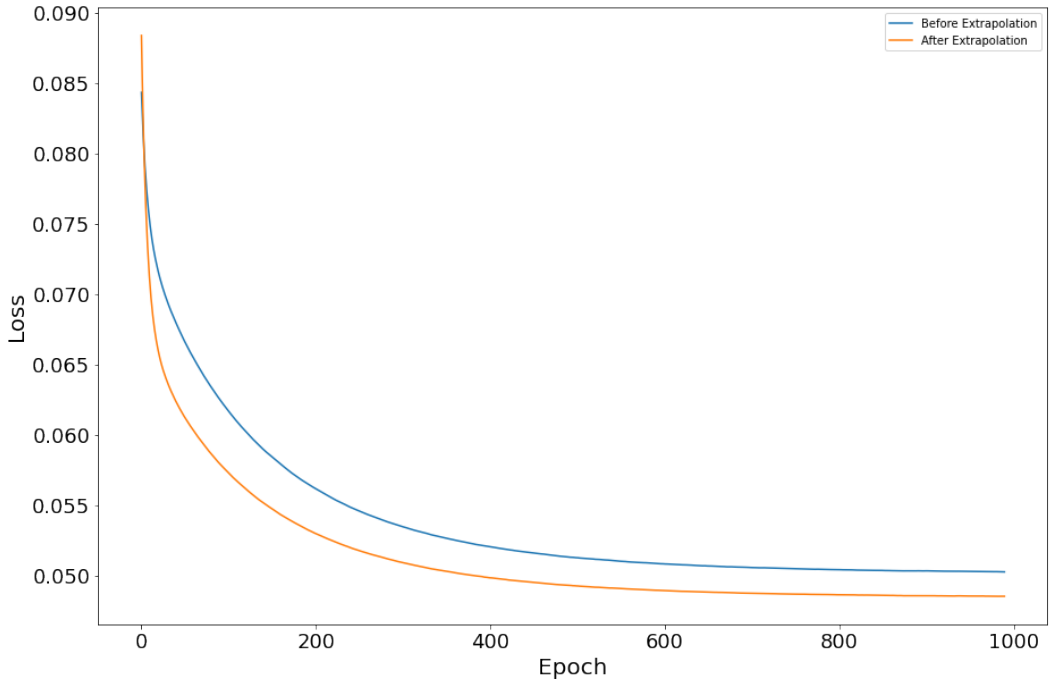


Figure 14: Linear regression with and without feature extrapolation

4.3 Movie Trailers

Movie trailers provide main contribution to the proposed methodology. They movie trailer collection process pipeline is represented in Figure 15. Given the already aggregated metadata, we start fetching the movie trailers by first considering the movie names. Then, an automated selenium script fetches the links from YouTube. These URLs are further used by a javascript to download the trailers. The downloaded trailers are preprocessed by removing the videos that are irrelevant or have a number of frames greater than a 5000, since trailers of such length do not correspond to the international standards of movie trailers, as discussed in [36].

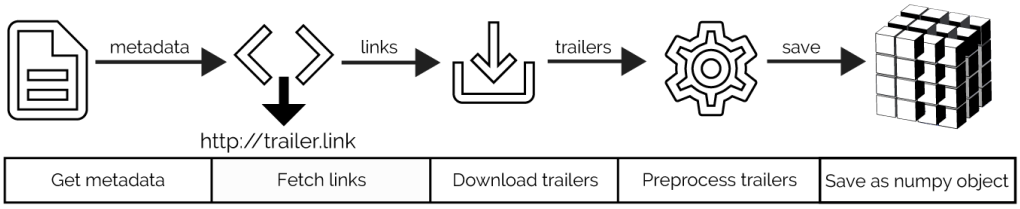


Figure 15: Trailer collection process

The filtering of trailers is followed by resizing, sampling the frames, augmenting using different techniques and are prepared separately for both the LTC and LSTM methodologies. For each trailer, the sampled frames are ordered to 16 time depths with 6 channels each (6 frames/time depth), appended to a tensor and saved for easy retrieval.

4.3.1 Frames Sampling

Movie trailers have an average of 150 seconds and 24 frames per second. Processing all the frames in a trailer is a tedious process. One of the ways of reducing the large number of frames is by removing frames with no change in intensity, i.e., transition of frames. Another way is to remove very similar frames. Firstly, training a deep neural network requires the input data to be same dimension and thus all the frames are resized to 58×58. Secondly, we implemented different techniqus and one of which is the Histogramic selection of frames. For a single frame, histograms are computed and compared with

the other frames. The histogram of current frame is compared with the histograms of the consecutive frames. If the correlation between the histograms are higher than a threshold, they are removed. The next frame is considered considered only when the correleation is lower. Once a uncorrelated frame appears, its histogram is used to compare with the consecutive frames. An example is illustrated below. By sampling the frames using histogram, only very similar frames are discarded and change in motion between the frames still have high correlation are are considered.

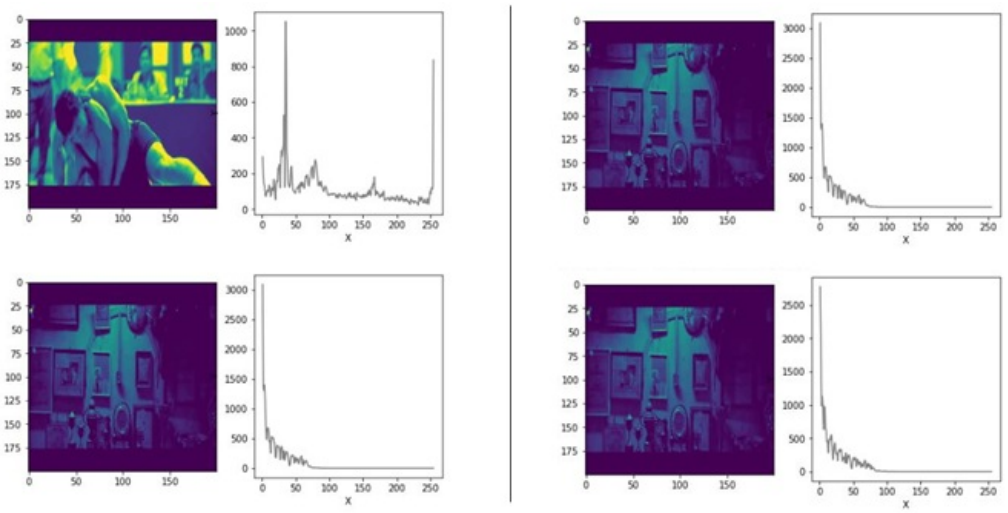


Figure 16: Frames sampling with histograms

In Figure 16, the frames on the left shows different histograms that are not correlated. Hence the next frame is considered. On the right side, both the histograms are correlated and hence, the next frame is discarded. Finally, following the frame sampling on all movie trailers, the final number of frames per trailer are tabulated and analysed to prepare a suitable dataset for training the model.

4.3.2 Data Preparation - LTC

The preprocessed movie trailers were prepared such that they are suitable for the LTC environment. Analysing the histogramically sampled frames of all the trailers, the number of frames to be used per trailer is set to 96 with

a time depth of 16 and 6 channels or frames per time depth. In order to sample the required number of frames in a trailer, a frame selection script selects the required number of frames dynamically irrespective of the total number of frames. An illustration of frame selection is shown below. Table 1 outlines the frame selection setup. Since the frame selection frequency is a not an integer scheme, we propose the following frame selection scheme: for the first 171 frames in a video, we select 1 frame for each other 3 frames. This results in a total of 57 frames. For the rest 156 frames we select one frames out of four, and obtain a sequence of 49 frames.

Hyperparameter	**Value**
Number of frames in movie trailer	327
Required number of frames	96
Frame selection frequency	327/96 = 3.4

Table 1: Frame selection setup

A tensor with 5 dimensions of form $n * d * c * w * h$ is initialized and the frames are appended in the order of 6 - 58×58 consecutive frames with each of the 16 timedepths. Finally a tensor with information of all the trailers is saved and retrieved for training the LTC model.

4.3.3 Data Augmentation

The final data after preprocessing the metadata and trailers were not sufficient to learn a better model. Hence, data augmentation plays a role in increasing the number of data points. Data augmentation in trailers is carried out in the similar way as shown above but that the frames are selected in the other way around. Similar to what is shown above, with data augmentation, 1 in 4 frames are selected for the first 156 frames and 1 in 3 frames are selected from the next or last 171 frames which provides a possiblity of choosing frames that are not selected previously. This allows the model to learn new information and hence improving the performance of the model.

4.3.4 Data Preparation - LSTM

The frames sampled using histograms are further filtered to suit the LSTM environment. The frames are resized to 150×150 and all the 3 channels (r, g, b) are considered. Similarly, after analysing the frames, the number of frames per video for the LSTM environment is set to 100. Of the total frames, 100 frames are sampled and saved as tensor objects. The tensors are parsed and the features are extracted using a ResNet model which are further embedded as input to the LSTM network to predict the box office revenue.

5 Methodology

In this section we present our methodology on predicting box office using movie trailers. We first give a brief problem formulation, and then proceed to details regarding the proposed methodologies. We extract the spatio-temporal features of videos using the Long Temporal Convolutions architecture. We consider a second model architecture to compare against - Long Short Term Memory. We then concatenate each temporal feature extractor with other information available (metadata) to test our hypothesis that trailer information would help leveraging more insights than the metadata alone.

5.1 Problem formulation

Given the metadata information $X_D \in \mathbb{R}^M$, and Movie trailer $X_V \in \mathbb{R}^{d*c*w*h}$ with M metadata features, d frames per channel, c number of channels, w, h is the frame width and height respectively, predict the domestic box office return $y \in \mathbb{R}$, for the opening week, by minimizing the Mean Square Error loss function:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2$$

$$\text{where}, y \rightarrow \text{Ground truth, and } \hat{y} \rightarrow \text{Prediction}$$

5.2 The LTC model

In this subsection we describe LTC model, with a focus on the architecture, input and training.

5.2.1 Network architecture

The proposed architecture consists of a Long-term Temporal Convolutions (LTC) network to model the temporal video representations and a Deep Neural Network (DNN) to extract the features from metadata information.

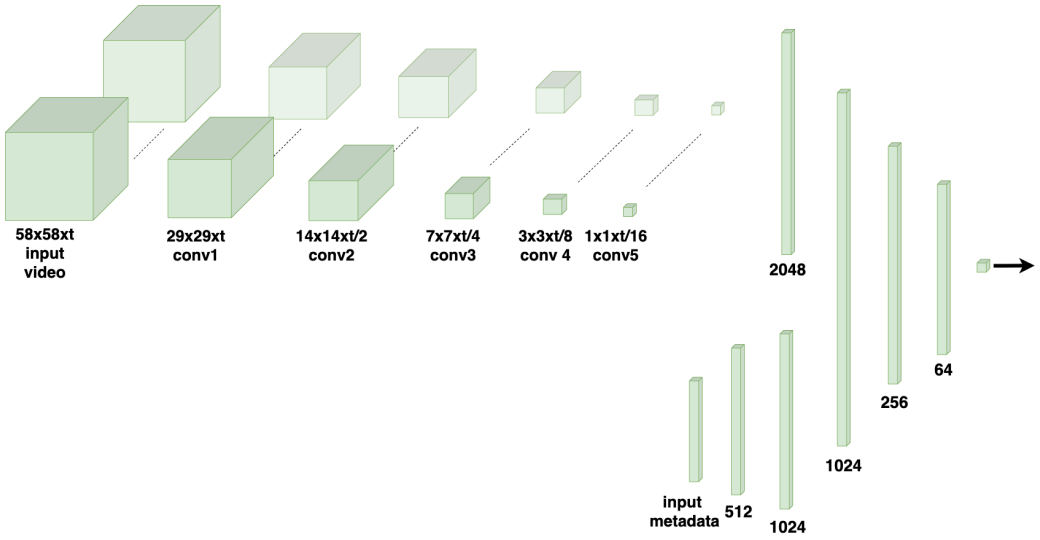


Figure 17: The proposed model with LTC + Metadata Architecure

The network architecture with long-term temporal convolutions is illustrated in Figure 17. The network has 5 space-time convolutional layers with 64, 128, 256, 256 and 256 filter response maps, followed by a fully connected layer, and concatenated to a fully connected network which embeds the metadata. Following [37] we use $3 \times 3 \times 3$ space-time filters for all convolutional layers. Each convolutional layer is followed by a rectified linear unit (ReLU) and a space-time max pooling layer. Similarly, for max pooling filters we set the size $2 \times 2 \times 2$ in all the layers but first, where we set the size $2 \times 2 \times 1$. We keep the convolution output size constant by padding 1 pixel in all three dimensions. Filter stride for all dimensions is 1 for convolution and 2 for pooling operations. We use a dropout ratio of 50% for the CNN part of the model; 40% on the fully connected layers to regularize and prevent overfitting. Fully connected layers are followed by ReLU layers. On the metadata part of the network we use a Fully Connected Network (FCN), with layers of size 512, 3072 (2048 from the CNN on videos and 1024 from the metadata), 1024, 256 and 64, and 1 for the output - a real number, indicating the predicted box-office return value.

5.2.2 Network input

To determine the impact of long-term temporal convolutions, we adopted a greedy approach, and tried various setups of frames per channel and number of channels in general. While choosing a more intelligent hyperparameter optimization approach could have yielded improved results, the research in this direction is beyond the scope of this paper. Initially, we have considered all the frames, which became computationally infeasible for our available resources. After an ample chain of trial and error, we proceeded with a setup of 16 time steps with 6 frames or channels per steps, and a total of 96 frames per video trailer.

The second part of the network embeds the metadata, to provide richer information. We use a 257 sized input layer for the metadata, which consists of numerical such as rating, popularity, runtime, budget, etc; and one-hot-encoded features, including cast, director, genre, and others.

5.2.3 Learning

We train the network on the training set, which contains 2.4K videos, including augumented samples. We use stochastic adam optimizer[38] applied to mini-batches with mean squared error criterion. We use a batch size of 30, learning rate of 10^{-5}, betas for the adam optimizer 0.9 and 0.999, and the min-max scaler. The network converges after 2000 iterations on Google Colab GPU, and takes 72 hours. Since Google Colab does not allow a continous run of more than 12 hours, we save the instance of the trained model and continue to train it by loading it into a new instance of a program.

5.3 The LSTM model

In this subsection we describe LSTM model, with a focus on the architecture, input and training.

5.3.1 Network architecture

Our second approach to capturing the spatio-temporal features is using an LSTM model which trains on the image embeddings. We extract the video frames embeddings using a pretrained feature extractor, which we chose to be the ResNet152, previously trained on the ImageNet Dataset[39]. We first load the pretrained model, change it's structure by removing the output layer and extract the feature maps for the frames. For each video we store a sequence of 100 feature maps. These feature maps are then loaded in a LSTM instance. The input layer of the LSTM is a 100 sized vector. This is followed by a hidden unit with a dimensionality of 512. The size of the output layer is 30. This is passed to a DNN along with the metadata information, similarly to what we use for the LTC architecture, with layer dimensions of 257, 512 + 30 (from the LSTM output), 1024, and the output layer with a dimensionality of 1 for predicting the box office return. A high level overview of the network architecture can be seen in Figure 18.

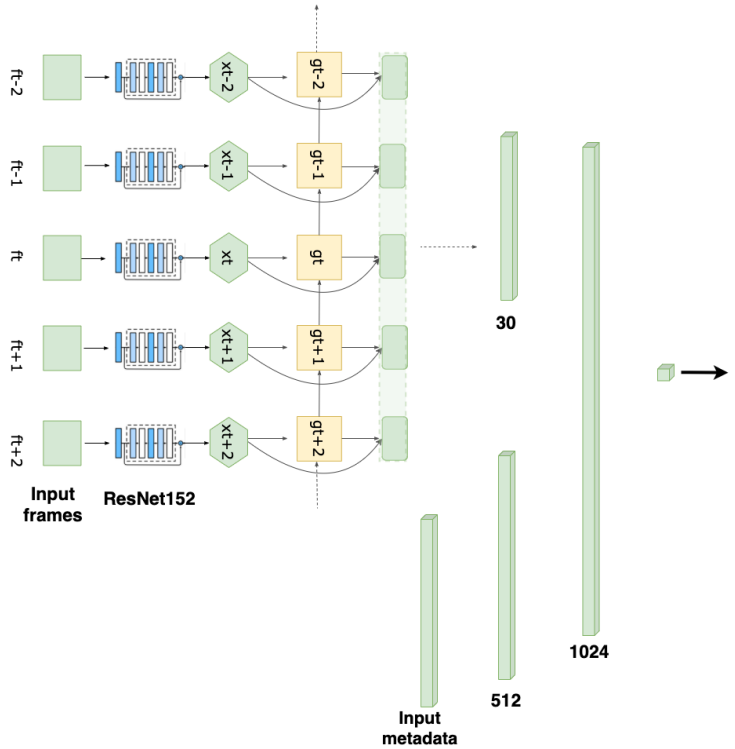


Figure 18: The proposed model with LSTM + Metadata Architecure

5.3.2 Network input

The input consists of the two parts: the video part and the metadata part. For the ResNet152, the frame dimensionality is $150 \times 150 \times 3$ for image width, hight, and three RGB channels respectively. Similarly to the LTC Network, the metadata is employed, and we do so by plugging 257 features, both numeric and one-hot-encoded.

5.3.3 Training

For training the LSTM network we use a set of 1.5K samples. We use stochastic adam optimizer[38] applied to mini-batches with mean squared error criterion. We use a batch size of 30, learning rate of 10^{-7}, and the min-max scaler. To prevent overfitting we implement a dropout of 0.8 for the LSTM part of the architecture and 0.2 on the metadata fully connected layers. We train the network for around 3 hours, and the register a convergence after 2700 iterations on the Google Colab GPU.

6 Key Performance Indicators (KPIs)

Box office revenue prediction, being a regression problem, is a task of predicting continuous values that is dependent on various predictors variables. The KPIs are different for different tasks and are mainly defined based on the problem. For the task of predicting the box office revenue, we defined the following KPIs to evaluate the performance of our model.

1. Mean Squared Error (MSE)
2. Root Mean Squared Error (RMSE)
3. Coefficient of Determination (R^2)

Mean Squared Error (MSE): MSE is one of the most commonly used evaluation metric for the regression problems. It is the average of the squared difference between the actual and the predicted values. For our problem, it is preferred because it is prone to outliers that a small error will be squared and makes a huge impact on the performance of the model.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where, $y_1, y_2, ..., y_n \rightarrow$ Actual value or Ground truth

$\hat{y}_1, \hat{y}_2, ..., \hat{y}_n \rightarrow$ Predicted value

Root Mean Squared Error (RMSE): RMSE is the most commonly used metric for the quantitative analysis of the regression tasks. It is the square root of the average squared difference between the actual and the predicted values. RMSE is preferred to evaluate our model because that the large errors in the revenue prediction make impact on the problem.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where, $y_1, y_2, ..., y_n \rightarrow$ Actual value or Ground truth

$\hat{y}_1, \hat{y}_2, ..., \hat{y}_n \rightarrow$ Predicted value

Coefficient of Determination ($\mathbf{R^2}$): The coefficient of determination is another way of evaluating the model's performance. It determines the quality of the model by estimating how good it is with respect to its baseline model. In other words, it is the measure of how much a model is better than the baseline model. The baseline model in our case is a simple average or a mean model.

$$R^2 = 1 \; - \; \frac{MSE(model)}{MSE(baseline)}$$

The MSE of the model is calculated as above the MSE of the baseline model is defined as follows:

$$MSE\ (baseline) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

where, $y_1, y_2, ..., y_n \rightarrow$ Actual value or Ground truth

$\bar{y} \rightarrow$ Mean of the observed actual data

7 Experiments and Results

This section breifly describes the experimental setup for each model, results of the proposed LTC and LSTM methods and comparision of the same with the baseline methodologies.

7.1 Experimental Setup

The experiments for different models were carried out in different environments. The models were trained on GPU support with the Google Colab's free cloud service. The scripts for training and testing were implemented with Python and PyTorch of versions 3.6.9 and 1.4.0 respectively. Table 2 shows the experimental setup including the hyperparameters for different models.

Hyperparameter	**Metadata**	**Metadata+LTC**	**Metadata+LSTM**
Learning Rate	$1e^{-4}$	$1e^{-7}$	$1e^{-7}$
Epochs	2600	2600	2600
Batch Size	30	30	30
Train/Test Ratio	80/20	80/20	80/20
Dropout	0.2	0.5	0.8
Number of Channels		6	3
Time Depth		16	100

Table 2: Experimental setup

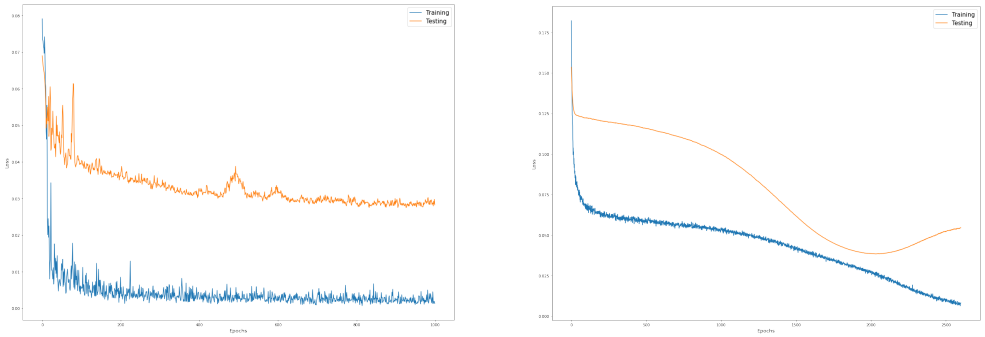
7.2 Results

In this section, two different versions of the LTC and LSTM, along with the metadata, models were trained and the results are compared. Also, the results of the model trained with the metadata alone is discussed.

7.2.1 LTC+Metadata

In order to get a better understanding of the performance of the model, two different versions of the model were trained. The earlier version and the

current version is shown in Figure 19. The earlier version was trained with a limited dataset of about 1400 data points, a learning rate of $1e^{-5}$ and not regularized. Whereas the current version was trained on augmented data,a learning rate of $1e^{-7}$ and highly regularized using dropouts. As the number of parameters in the earlier version of the model is large, the model converges in the initial stage of the training. The earlier version overfits, later 200 epochs, due to the less number of data and model's complexity that the model learns the pattern in the training data but performs poor on the testing data. The training loss is as low as 0.003 and that of testing is 0.038.



(a) Earlier version (b) Current version

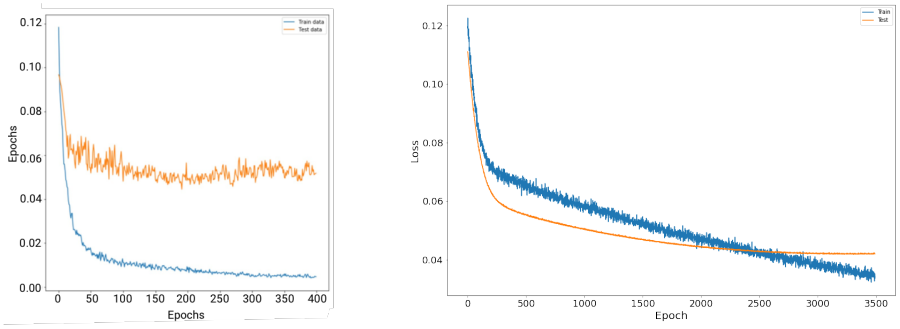
Figure 19: Training vs Testing loss - LTC + Metadata

The loss curves of the current version in figure 19(b) shows smooth convergence and does not overfit in the initial stages. The testing curve shows the early stopping epoch where the model converges and the testing loss starts to diverge. The training loss is similar to the earlier version whereas the testing loss (0.035) shows a slight improvement in performance.

7.2.2 LSTM+Metadata

For the LSTM model, we tried multiple approaches. Earlier, we had bi-directional LSTM and a lot of learning parameters, which was performing very poorly and was overfitting, although this overfitting is due to video learning. As in high dimensional data, where the number of features are greater than the observation generally leads towards the overfitting, especially in the video regression case. In our initial bi-directional LSTM, we

had more than 16 M parameters. As we can see in Figure 20 (a) the convergence was very fast. After that, we decided to change the overall structure of the architecture, and we used simple single directional LSTM with only one hidden layer and with a large amount of dropout rate.



(a) Earlier version (b) Current version

Figure 20: Training vs Testing loss - LSTM + Metadata

In the above figures, we can comprehend the loss behavior of both models. In the current model, we can see that the model with the learning rate of $1e^{-7}$ is learning rapidly till the 250 epochs, then the loss is decreasing smoothly till 3000 epochs. Whereas in the earlier model, with the learning rate of $1e^{-4}$ the learning was too fast till 40 epochs and model converges in 150 epochs and overfits. Current LSTM has a lower loss of 0.041 MSE comparative to the previous version of LSTM. In Figure 20 (b), we can see that validation is smaller than training loss initial because we are using a large amount of dropout on the training process. So basically, dropout is activated in the training process but deactivated when evaluating on the validation set. So, it seems to be normal behavior

7.2.3 Metadata

Finally, we implemented a model that is trained only on the metadata. This is to get an idea if the inclusion of trailer information improves the performance. For this, the input of 257 predictors are fed into a multi-layer perceptron with hidden layers of 512, 1024, 256, 64 neurons to learn the features and further regressed to predict the box office revenue. We can observe

in Figure 21 that during training, the model overfits in 500 epochs and later which, there is no noticeable change in the model's performance.

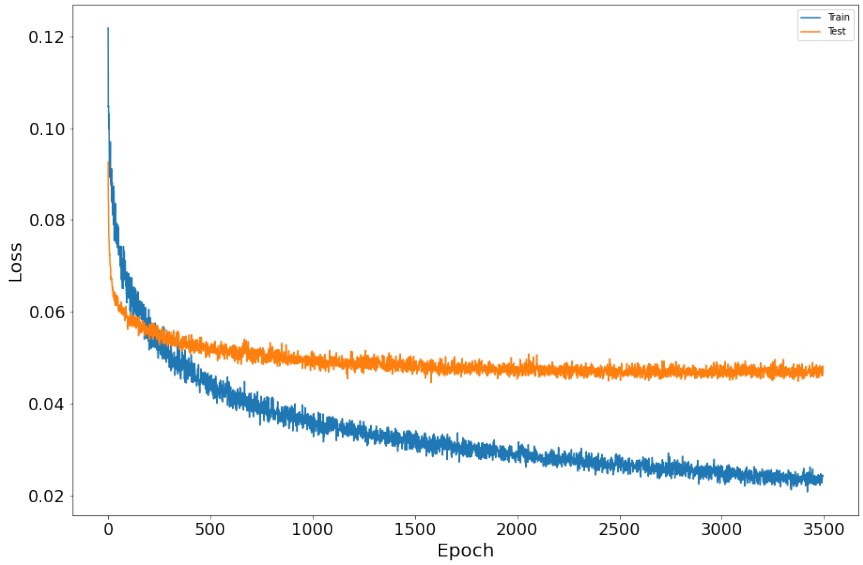


Figure 21: Training vs Testing loss - Metadata only

7.3 Comparison

While training separately provides detailed information about how each model is performing separately, a general view on how each model performs along with other models is helpful for reasoning about which model performs best.

Table 2 shows the setup for each model we consider, including simple baselines such as Average, Linear Regression or MLP, along with the methods proposed by us: LTC+Metadata and LSTM+Metadata. We train all the methods for 2.5K epochs. Figure 22 shows the behaviour of the MSE loss function for each of the models in the training process. The simplest curve out of all is the brown one, also corresponding to the simplest model - Average. This model simply takes the target variable - box office return, sums up all the values and divides by the number of simplest. While being the simplest, it also performes the worst. The green line correspondes to the LSTM+Metadata model. We see that it learns vastly in the first 500 epochs

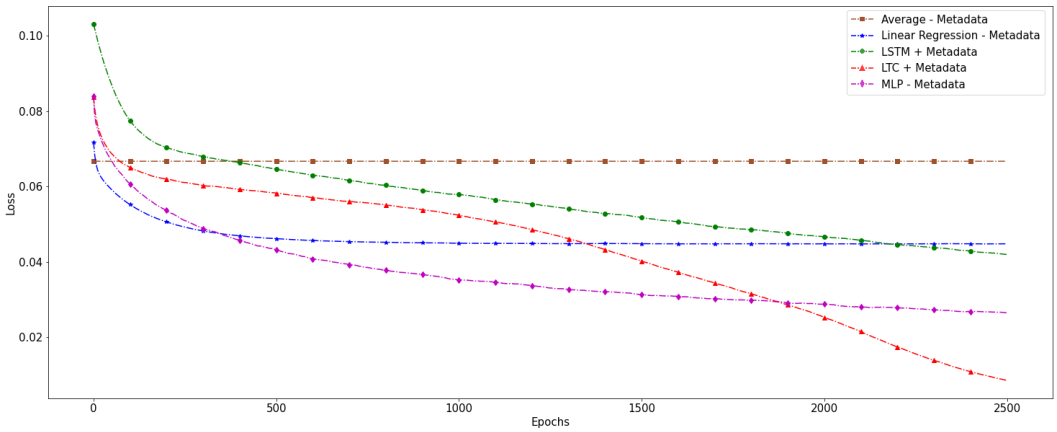


Figure 22: Training Loss - Comparison

and after that decrease is slow gradually thoughout. A decrease in the error is still registered till the 2500th epoch, but it is only by a thing margin. The blue curve corresponds to the Linear Regression model. This model is a very simplistic one and it registers progress in the first 600 iterations. Afterwards the loss curve flattens and never decreases. We assume that this is due to the fact that since the model is very simplistic, the parameters do not get a big change after the 600th iteration. The MLP model trained only on the metadata registers a well-behaving decreasing curve. The model learns patterns in the metadata and converges around 1500th epoch, with a slight improvement afterwards as well. Last but not least, the red curve corresponds to the LTC+Metadata curve. This curve behaves different than others. It learns slower in the beginning but seems to improve after the 1300 epoch, where the loss decreases in a more aggressive fashion.

The most interesting part is the model comparison on the unseen data. Figure 22 shows the behaviour of the MSE loss function for each of the models on the test set. The color scheme in this figure corresponds to the one in the Figure 22. The Average model, represented by the brown curve does, as expected, not improve during the time axis, and performes the worst out of all the models. Two other models - Linear Regression and MLP, both of which

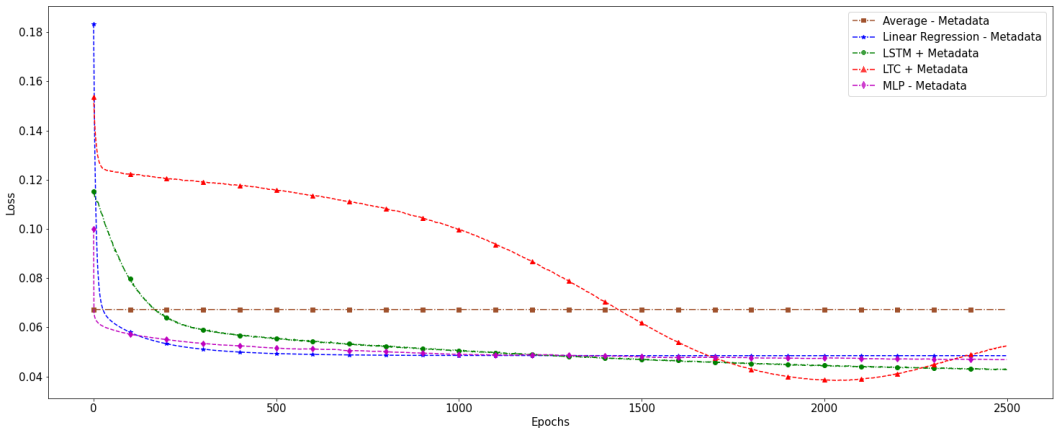


Figure 23: Testing Loss - Comparison

have only been trained on metadata and used no video trailer information perform similarly, by first decreasing their loss in the first 700-1000 epochs, and flattening afterwards. In contrast, the green curve, corresponding to the LSTM+Metadata models shows a more stable behaviour. First the loss corresponding to this method decreases speedily till the 1000th epoch, but it keeps decreasing in a well-behaving fashion afterwards as well, showing further improvements even after the 2500th epoch.

The red curve has likely the most curious behaviour in this graph. In the beginning the method has a very slow learning. Subsequently, the decrease in the loss curve is also shallow. However, after the 500th epoch, the learning improves and the loss registers a substantial decrease, with the lowest point being reached at the 2000th epoch. We believe that this behaviour is due to the complexity of the network - we learn this network from scratch, and therefore it takes a while for the method to converge, with no prior "experience". This is why it behaves differently to other curves: while LR and MLP are also trained from scratch, using video information complicates the network setup, inducing more parameters to be learnt. An LTC-like model would need more information and time to train and learn. One could argue that we also use videos in case of the LSTM+Metadata network. However,

this is not fully true. For the LSTM network we first pass the video frames to a feature extractor. This drastically reduces the amount of model parameters to be learnt. After the 2000th epoch, the curve starts to increase as well, meaning that the model is overfitting. This is likely our most powerful model, but arguably also the least stable. Given the limited dataset we cannot test this hypothesis, but we believe research in this direction is the right way to go. Nevertheless, we see a clear pattern: embedding movie trailer information helps improving the prediction of the box office return, but it does so by a thin margin.

Figure 24 shows the distribution of actual and predicted revenue values. In this graph, it can observed that the LSTM model is performing better in the higher confidence interval of the distribution. The LSTM model is not able to predict the long tail of the distribution correctly. Whereas on the other side, LTC model is performing much better than LSTM and prediction values on the long tail as well.

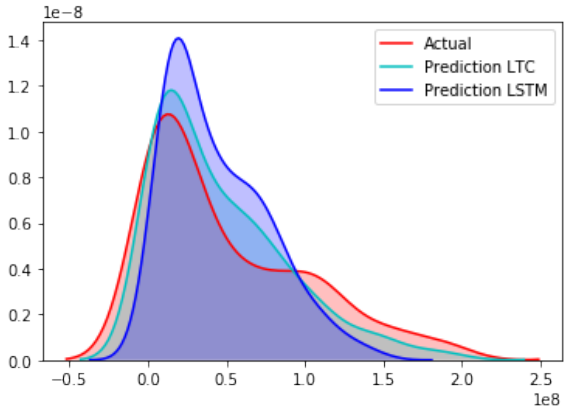


Figure 24: Prediction vs Actual Distributions

7.4 Evaluation

As discussed in the previous section, MSE, RMSE and R^2 are the KPIs used to evaluate the performance of different models for a better understanding.

The results are tabulated and shown in Table 3.

Methods / Evaluation Metric (KPI)	MSE		RMSE		R^2	
	Train (80/100)	Test (20/100)	Train (80/100)	Test (20/100)	Train (80/100)	Test (20/100)
Metadata	0.006	0.047	0.077	0.21	0.93	0.35
LSTM + Metadata	0.005	0.041	0.070	0.20	0.90	0.33
LTC + Metadata	**0.003**	**0.035**	**0.055**	**0.19**	**0.95**	**0.43**
Linear Regression	0.045	0.049	0.21	0.22	0.33	0.27
Averaging / Mean	0.067	0.068	0.26	0.26	0	0

Table 3: Comparison of the proposed methods with other Machine Learning methods

It can be seen from Table 3 that the proposed methods perform better than the baseline methods - Linear Regression and Average model. The coefficient of determination shows that the models LTC and LSTM are 43% and 33% better than the baseline average model and 28% and 10% better than the linear regression model. The Mean squared error for the LTC and LSTM models are 0.003 and 0.005 which are very much less compared to the linear regression (0.045) and the average (0.067) models. Similarly, the root mean squared error of the proposed methods are less and comparable with the base machine learning methods. Finally, as discussed in the previous section, it can be seen from the tabulated results that the LTC model which uses the trailers and metadata information performs better than the other machine learning and deep neural networks.

8 Discussion

This work report outlines the most important contribution of our project. Besides the main proposed methodologies which have already been outlined above, different other approaches have been tried, and are discussed in this section. While some of them have led to dead ends, they still helped discover directions for the methodologies presented above.

One of our first approaches was using a pretrained 3D-ResNet, initially trained for the classification task. We modified the head to be a regreassion head, and used it on 30%, 50% and then 90% of the training dataset. We have also applied 3D-CNN as in [43, 44]. With different setups and hyperparameter settings the results proved suboptimal, and due to the time constraints we aborted this idea.

Research has been done in the direction of emotion analysis based on movie frames. We have passed the video frames to a emotion detection pretrained network, and then forwarded the input to a fully connected network. This setup yielded poor performance, and we believe that there is something deeper when it comes to emotions: the emotion present in the movie trailer is not necessary the emotion a person feels when watching it. While this is a valid hypothesis, it is hard to prove it, since it is extremely challenging to find datasets on what real people experience when watching a movie trailer.

Working with the LSTM, we first implemented bidirectional LSTM, since it seemed to perform better in literature. However, in our case, the bidirectional LSTM performed worse. We assume that this is due to the fact that with bidirectional LSTM the number of parameters in the network is almost double. This lead to immediate overfitting, and the results were unsatisfactory. However, for a larger dataset, chances are that this implementation would work better than the vanilla LSTM. Similarly, we tried implementing an attention mechanism in the LSTM network. Due to the same challenge of too many parameters, the model was again underperforming.

Another approach to improving the prediction was changing the regression head from predicting box office return to predicting a highly correlated variable. We chose the budget feature, and retrained our network, but since the predictions did not improve by much we switch back to predicting the box office return value.

Worth mentioning, we have also tried predicting the box office return by only using the movie trailers. We set up an LTC network with only the part responsible for videos, and removed the layers corresponding to the metadata. This was followed by a series of hyperparameters grid search for identifying the right network architecture. Any setup tried proved to preform poorly. We have worked in parallen on the LSTM network, with a similar setup - removing the layers coming from metadata - but the second method did also perform poorly, and therefore we aborted research in this direction.

From the domain research, we discovered that there is a whole supply chain process in the movie industry. One of the critical factors in this process is the distributor. If we have a distributor of the movie, we will have an understanding of how many theaters it will be released. We tried scraping the IMDB website for this information, but in most cases, there was no data. We found data for a very few hundred movies. So, with a smaller dataset, training videos was again the issue of overfitting, so we decided not to decrease data anymore.

In the high dimensional data, autoencoder is very common. So, we gave try to this as well and we created an encoder which as two outputs decoders, one for revenue prediction and other as for reconstruction loss. These kinds of autoencoders are called composite autoencoders, which is a kind of supervised model. But we couldn't train this model because it was consuming to much ram that we couldn't run this model even for one epoch in google colab.

9 Conclusion

Box office return likely a key factor in judging the success of a movie. Having an upfront prediction of this variable before releasing the movie or even it's trailer could help production houses take the right actions, so that they could increase the box office return. In this work we analyze the problem of box office return prediction, and propose two methodologies, that make use of movie trailers to improve it.

An important contribution of our project consists of data aggregation and processing. We first collect movie metadata, by using various web resources and automated tools to aggregate a more complete dataset, which is in itself a challenge: many movie websites or web resources provide incomplete data. We solve this challenge by aggregating the information from multiple sources. The second part in the dataset collection is aggregating the movie trailer. While it is a simple task, it is not easy. First, it is because open source tools solving this task are scarce. Second, some movies for which metadata was available did not have an open source trailer, publicly available on YouTube. Third, since the tools which we have adopted were using the movie name for searching the trailer, sometimes our code was downloading the movie, instead of the trailer. This task it also not easy because it takes a while for hours of video to be downloaded. Last but not least, even after getting the trailers on our disk, the video dataset was not fully ready to be used. We employed a frame-sampling technique for selecting the best frames, concatenated them and saved as numpy object for further use.

Our main contributions in this project are the two methodologies proposed: the LTC and the LSTM models, both of which combine and use metadata information along to movie trailers. The LTC method makes use of long-temporal features of movie trailers, by applying a sequence of convolutional and pooling layers, followed by a deep neural network. This model proves better than simple naive approaches (eg. Linear Regression, MLP), which use only metadata. This model also achieves the lowest MSE Loss out

of all the models. The second method employing video trailers is the LSTM architecture. To train it, we chose a slightly different approach. First we used a feature extractor, previously trained on images, passed selected movie frames through this pretrained model, and then stored the activations in the last hidden layer in sequence of 100 vectors for each video trailers. Accordingly, we passed these sequences to the LSTM model, the output of which is forwarded to a DNN, which does in parallel make use of metadata. This architecture enables us achieve the second smallest loss, but behaves likely the most stable out of all models, except Average. Both of our proposed methods preform best and register the lowest loss value. Nevertheless, the difference between simple models and our models is rather small. This could be due to many factors. First, we use networks with many parameters. This means that in order to get more stable behaviour, we need a bigger dataset, which in our case is hard to achieve. On the other hand, an increase in the dataset automatically means a demand in hardware capacity, which likely was one of the major impediments during the implementation of this project.

Multiple other directions have been considered, including different network configurations; distinctive types of pretrained networks for 3D-CNN (compatible with videos, and not images as we do in the current work); applying sentiment analysis or convolutions over sound. While many of these approaches took great effort, their results have been rather suboptimal and research in thos directions has been discontinued.

Finally, we can conclude that movie trailers do contribute to box-office return prediction, but are not a central variable. Oftentimes, variables such as budget, cast, production company, and distributor are crucial to making accurate predictions. Movies take a long path till they are shown in theaters. Almost always, negotiations with the movie theaters is started well before the creative part, and trailers alone serve just an advertisement material for the public. Movie trailers likely contribute by a very thin margin in the decision process of movie theaters to choose a movie to be displayed on the screens, and this is reflected in our experiment results as well.

References

[1] M. Saraee, S. White & J. Eccleston, "A data mining approach to analysis and prediction of movie ratings", University of Salford, England, The Fifth International Conference on Data Mining, Text Mining and their Business Applications, 15-17 September 2004, pp 344- 352.

[2] Bhave, A., Kulkarni, H., Biramane, V., & Kosamkar, P. (2015, January). Role of different factors in predicting movie success. In 2015 International Conference on Pervasive Computing (ICPC) (pp. 1-4). IEEE.

[3] Wallace, W. T., Seigerman, A., & Holbrook, M. B. (1993). The role of actors and actresses in the success of films: How much is a movie star worth?. Journal of cultural economics, 1-27.

[4] De Vany, A., & Walls, W. D. (1999). Uncertainty in the movie industry: Does star power reduce the terror of the box office?. Journal of cultural economics, 23(4), 285-318.

[5] Sharda, R., & Delen, D. (2006). Predicting box-office success of motion pictures with neural networks. Expert Systems with Applications, 30(2), 243-254.

[6] Ghiassi, M., Lio, D., & Moon, B. (2015). Pre-production forecasting of movie revenues with a dynamic artificial neural network. Expert Systems with Applications, 42(6), 3176-3193.

[7] Zhang, L., Luo, J., & Yang, S. (2009). Forecasting box office revenueof movies with BP neural network. Expert Systems with Applications,36(3), 6580-6587.

[8] Statistics of movies released since 2001, https://www.statista.com/statistics/187122/movie-releases-in-north-america-since-2001/, visited online on January 25th, 2020

[9] Varol, G., Laptev, I., & Schmid, C. (2017). Long-term temporal convolutions for action recognition. IEEE transactions on pattern analysis and machine intelligence, 40(6), 1510-1517.

[10] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[11] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778.

[12] Internet Movie Database, https://www.imdb.com/interfaces/, accessed online March 17th

[13] T. Kim, J. Hong, and P. Kang. (2015). Box office forecasting using machine learning algorithms based on SNS data.

[14] Travis Ginmu Rhee and Farhana Zulkernine. 2016. Predicting movie box office profitability: a neural network approach. In 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pages 665–670. IEEE.

[15] Sameer Ranjan Jaiswal and Divyansh Sharma. 2017. Predicting success of bollywood movies using machine learning techniques. In Proceedings of the 10th Annual ACM India Compute Conference on ZZZ, pages 121–124. ACM.

[16] A. Chen, "Forecasting gross revenues at the movie box office," Working paper, University of Washington Seattle, WA, June 2002.

[17] Simonoff, J. S., & Sparrow, I. R. (2000). Predicting movie grosses: Winners and losers, blockbusters and sleepers. Chance, 13(3), 15-24.

[18] M. S. Sawhney and J. Eliashberg, "A parsimonious model for forecasting gross box-office revenues of motion pictures," Marketing Science, vol. Vol. 15, No. 2, pp. 113–131, 1996.

[19] Sitaram Asur and Bernardo A. Huberman, "Predicting the Future with Social Media," http://arxiv.org/abs/1003.5699, March 2010.

[20] Reddy, A. S. S., Kasat, P., & Jain, A. (2012). Box-office opening prediction of movies based on hype analysis through data mining. International Journal of Computer Applications, 56(1), 1-5.

[21] Quader, N., Gani, M. O., & Chaki, D. (2017, December). Performance evaluation of seven machine learning classification techniques for movie box office success prediction. In 2017 3rd International Conference on Electrical Information and Communication Technology (EICT) (pp. 1-6). IEEE.

[22] Apala, K. R., Jose, M., Motnam, S., Chan, C. C., Liszka, K. J., & de Gregorio, F. (2013, August). Prediction of movies box office performance using social media. In 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013) (pp. 1209-1214). IEEE.

[23] Zhou, Y., Zhang, L., & Yi, Z. (2019). Predicting movie box-office revenues using deep neural networks. Neural Computing and Applications, 31(6), 1855-1865.

[24] K. Ozkan, O. N. Atak, and S. Işik. "Using movie posters for prediction of box-office revenue with deep learning approach." 2018 26th Signal Processing and Communications Applications Conference (SIU). IEEE, 2018.

[25] Gevaria, K., Wagh, R., & D'mello, L.R. (2015). Movie Attendance Prediction.

[26] Marshall, P., Dockendorff, M., & Ibanez, S. (2013). A forecasting system for movie attendance.

[27] Zhang, Xu., Hou, Guangming., Dong, Weijia., Modelling movie attendance with seasonality: evidence from China, Applied Economics Letters. Nov2017, Vol. 24 Issue 19, p1351-1357. 7p.

[28] Hsieh, C., Campo, M., Taliyan, A., Nickens, M., Pandya, M., & Espinoza, J.J. (2018). Convolutional Collaborative Filter Network for Video Based Recommendation Systems. ArXiv, abs/1810.08189.

[29] Miguel Campo, Cheng-Kang Hsieh, Matt Nickens, J. J. Espinoza, Abhinav Taliyan, Julie Rieger, Jean Ho, Bettina Sherick: Competitive Analysis System for Theatrical Movie Releases Based on Movie Trailer Deep Video Representation. CoRR abs/1807.04465 (2018)

[30] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1235–1244. ACM, 2015

[31] X. Li and J. She. Collaborative variational autoencoder for recommender systems. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 305–314. ACM, 2017

[32] The ILSVRC 2015 ImageNet Competition, http://image-net.org/challenges/LSVRC/2015/, accessed online March 3rd, 2020

[33] Coco competition, http://cocodataset.org/home, accessed online March 3rd, 2020

[34] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (1986-10-09). "Learning representations by back-propagating errors". Nature. 323 (6088): 533–536. doi:10.1038/323533a0. ISSN 1476-4687.

[35] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, Yoshua Bengio. "Gated Feedback Recurrent Neural Networks". Submitted on 9 Feb 2015.

[36] Finsterwalder, J., Kuppelwieser, V.G., de Villiers, M. (2012) The effects of film trailers on shaping consumer expectations in the entertainment industry—A qualitative analysis. Journal of Retailing and Consumer Services, 19(6), pp. 589-595.

[37] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in ICCV, 2015.

[38] Adam: A Method for Stochastic Optimization D. Kingma, and J. Ba.(2014) Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[39] ImageNet dataset link, http://www.image-net.org/, visited on February 2nd, 2020

[40] Chen, A. (2002). Forecasting gross revenues at the movie box office. University of Washington, Seattle.

[41] Lash, M. T., & Zhao, K. (2016). Early predictions of movie success: The who, what, and when of profitability. Journal of Management Information Systems, 33(3), 874-903.

[42] Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In Advances in neural information processing systems (pp. 568-576).

[43] Ji, S., Xu, W., Yang, M., & Yu, K. (2012). 3D convolutional neural networks for human action recognition. IEEE transactions on pattern analysis and machine intelligence, 35(1), 221-231.

[44] Taylor, G. W., Fergus, R., LeCun, Y., & Bregler, C. (2010, September). Convolutional learning of spatio-temporal features. In European conference on computer vision (pp. 140-153). Springer, Berlin, Heidelberg.