View Count Prediction of a Video through Deep Neural Network based Analysis of Subjective Video Attributes

Spandan Basu

View Count Prediction of a Video through Deep Neural Network based Analysis of Subjective Video Attributes

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Spandan Basu
[Roll No: CS-1814]

under the guidance of

Dr. Dipti Prasad Mukherjee
Professor
Electronics and Communication Sciences Unit
Indian Statistical Institute

Indian Statistical Institute
Kolkata-700108, India

July 2020

To the entire deep learning community

CERTIFICATE

This is to certify that the dissertation entitled **"View Count Prediction of a Video through Deep Neural Network based Analysis of Subjective Video Attributes"** submitted by **Spandan Basu** to **Indian Statistical Institute, Kolkata**, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Dipti Prasad Mukherjee
Professor,
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to show my highest gratitude to my advisor, *Prof. Dipti Prasad Mukherjee*, Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, for his guidance and continuous support and encouragement. He has literally taught me how to do good research, and motivated me with great insights and innovative ideas.

I would also like to thank *Dr. Bhabatosh Chanda*, Professor, Indian Statistical Institute, Kolkata, for his valuable suggestions and discussions.

I would specially like to thank *Dr.Angshuman Paul* as the original idea of this research project emerged while discussing with him.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added an important dimension to my research work. I would like to acknowledge *Saikat Sarkar*, *Nasib Ullah*, *Sankarsan Seal*, *Bikash Santra*, *Avishek Shaw* and all other seniors at the lab for their constant guidance. I would specially like to thank *Sankarsan Seal* to make the GPU resources available for us to work during the crisis period of COVID-19 pandemic.

Finally, I am very much thankful to my parents and family for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Spandan Basu
Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

This research work is aimed to design a method for predicting the view count of a video using deep neural network based analysis of subjective video attributes. With more and more companies turning to online video content influencers to capture the millennial audience, getting people to watch your videos on online platforms is becoming increasingly lucrative. So we provide a solution to the problem by building a model of our own. Our model takes four subjective video attributes as input and predicts the probable view of the video as output. The attributes are the thumbnail image, the title caption, the audio associated with the video and the video itself. We preprocess each of the attributes seperately to obtain the feature vectors. Our model contains four branches to deal with these attributes. We pass the feature vectors of each of the component to the respective branches of the model to capture the salient features with regards to the thumbnail image using a pre-trained CNN architecture, AlexNet; the sentimental feature with regards to the title caption using Sentiment Intensity Analyzer; the temporal feature with regards to the audio waveform using LSTM and both the temporal and salient features with regards to the video using Convolutional LSTM. Since a user, clicks a video based on the title and the thumbnail associated with the video on most online platforms, the model tries to generate a click affinity feature depicting the affinity of the user to click the video. After the user clicked the video, the user decides to view the video based on the audio and the video itself, so the view count of the video is predicted by taking into account the click affinity feature alongwith the temporal feature of the audio waveform and the spatial - temporal feature of the video using a regressor network called the viral-video-prediction network. A loss function designed from this regression values is used to train the last two stages of the pipeline. We obtain a test accuracy as high as 95.89%.

Keywords: *AlexNet, deep neural network, Sentiment Intensity Analyzer, LSTM, Convolutional LSTM*

Contents

Chapter 1

Introduction

1.1 Introduction

A viral video is a video that becomes popular through a viral process of Internet sharing, typically through video sharing websites such as YouTube as well as social media and email [1]. It's usually a quick few days of exposure through which a viral video can potentially gain the attention of millions of internet users. That is why viral videos have a profound impact on many aspects of society, such as politics and online marketing. Therefore the task of predicting the popularity as well as number of views of an internet video is gaining attention to the researchers.

The factors which account for a video to be viral should be subjective as it should reflect a personal appeal of an internet user towards the video. On the other hand, the factors which contribute for a video to appeal to the majority of internet users needs to be captured, implying that the factors be generically relevant to a large section of these internet users as well. Therefore, to design a model that can infer whether a video is going to be viral or not is a difficult problem.

The count of views is an important measure of the popularity of an online video. The views of a video reflect how many times the video has been watched. A viral video has typically a large number of views in a short period of time. But why do an online video has millions of views while other videos do not?

Several studies show that viral videos typically have some common properties such as short title, short duration, the element of surprise in the video content and high quality of music [2, 3]. Some recent studies show that there are strong correlation between the popularity of a video and sharing platform like blog, hosting channel [4, 5]. But the task of predicting number of views of a video is still an unsolved problem.

1.2 What makes a video viral?

To sum up, we can discuss on the factors which decides whether a video is going to be viral or not. The answer to this question depends on the subjective analysis of different facts. A video is said to be viral when the number of its views crosses a threshold within a small amount of time after the release of the video. Therefore, in order to know if a video is going to be viral, we need to predict its views. The number of views in YouTube is calculated by finding the number of YouTube users (from now on, *user* would mean YouTube user) who clicked the video and watched it for at least 30 seconds or the full video duration whichever is smaller. Another factor that affects the number of views is the number of shares. When a video is shared in large numbers, it tends to have more views [6]. It is also evident that a majority of the users who shares a video typically likes the video as well. Hence from a user's perspective, the number of views depends on the answer of two questions: (a) why would a user click on a video? and (b) why would a user like the video and share it. As these factors are crucial, we have tried to model these in our research work which is discussed in details in chapter 4.

1.3 Related Work

Analysis of content popularity has attracted huge interest in recent years. One of the first works that analyze the factors affecting the popularity of internet videos can be found in [2]. However, the analysis has been performed on only a few videos making it difficult to generalize. Deza *et al.* [7] have identified several visual attributes that affect the virality of an image. Several characteristics of viral videos have been identified in [3]. The viewing pattern of popular videos in YouTube has been analyzed by Broxton *et al.* [8] where it has been shown that the popular videos have a sudden rise and fall in the views compared to the less popular videos. More advanced dependence between cross-platform contents have been explored in [9]. In this work, the authors analyze the properties that helps to popularize a Tweet and a YouTube video mentioned in the Tweet.

While researchers have tried to identify the factors influencing the popularity of a content, several methods have been developed to predict the popularity of different online contents. In [10], the authors have shown that the future view of a YouTube video can be predicted using the early viewing pattern of the video. In this work, the authors have trained a regression model to predicted the total number of views that a video can have in future based on the past viewing pattern of the video. Collection of the view count of a video at different time instants is major challenge in the training of the above method. Furthermore, this method does not take into account the different attributes of a video. Therefore it is impossible to relate the viewership with video quality using the above method. Utilizing the visual cues, a spatial transformer model has been proposed in [11] to predict whether an image is going to be viral. The popularity of an online content has been predicted in [12] using only the title. A multimodal popularity prediction method for micro-videos can be found in [13]. This method makes use of social, visual, acoustic and textual modalities related with a video to find whether the video is going to be popular. However, the definition of the popularity in the above model is ad-hoc and a relationship between the different modalities and the popularity can not be defined. Further, this method cannot predict the number of views of a video. See [14] for a multimodal popularity prediction approach for images. Zhang *et al.* [15] propose fusion of visual, textual and user information in an attention model for popularity prediction of Flickr images. In [16], the authors propose a support vector regression model to predict the popularity of online videos. A multimodal popularity prediction model for videos using self attention has been proposed in [17]. However, the above popularity prediction models lack a mathematical analysis of how the different controlling factors of popularity relate to the number of views of the content.

1.3.1 Existing Methods

According to our literature survey, we have selected one of the existing methods that target to predict the number of views of a video. This work is from a github project [18]. The model discussed here takes into account the following attributes of the data as follows:

- Thumbnail: The thumbnail image of the video in grayscale color space.
- Duration: The duration of the video.
- Dislike Count: The number of dislikes the video received.
- Comment Count: The number of comments the video received.
- Like Count: The number of likes the video received.
- Channel View Count: The number of view counts the channel publishing the video received.
- Channel Subscriber Count: The number of subscribers of the channel.
- Channel Video Count: The number of videos published under the channel.
- Number of Tags: The number of tags the video received.

Based on these attributes the model tries to predict the number of views the video would receive. The view counts that have been taken as ground truth is scaled to an integer number from 0 to 9 and treated as labels. The model contains two stages.

In the first stage, the thumbnail is fed into a convolutional neural network i.e. CNN (described in details in section 2.2) comprising of two convolution layers followed by two fully connected layers. The last layer of the CNN is the output layer comprising of 10 neurons, essentially treating the problem as a 10 class classification problem i.e.(class 0 to class 9) and assigning the class corresponding to the neuron with the highest value among the 10 neurons as the value of the prediction from the thumbnail which we denote here as thumbnail predictions.The CNN is trained against the labels and the training loss function used in this case is Cross Entropy Loss. The Cross Entropy Loss is explained in section 2.7.2. So, the thumbnail predictions contains integral values ranging from 0 to 9. In the second phase, the thumbnail predictions is considered as an attribute alongwith the other attributes mentioned above to form a new dataset. This dataset is then trained on Support Vector Regressor against the labels to output a fractional value from 0 to 9. This value when scaled to the original scale is the predicted number of views. We have tried to compare the existing method with our proposed model by training the existing model on our dataset and the results are reported in Chapter 5.

1.4 The Problem Statement

Now let us provide a formal definition of the problem at hand and provides a basic overview of the method we adopted to solve it.

The problem is formally defined as follows: Given a set of videos from YouTube and their respective view counts which we consider here as ground truth, we need to design a machine learning model that when trained on this set of videos alongwith its subjective attributes is capable of predicting accurately the number of views a video can get over a certain period of time. The model should have good generalization capabilities and should not overfit the training set. Provided the desired levels of accuracy achieved, the model could be deployed to predict how many views a new video gathers under a certain period of time.

1.5 Overview of Our Solution

Figure 1.1 displays our solution strategy in the form of a block diagram. Our strategy can be described as a two step process. In the first step, we extract the subjective video attributes of the query video including its associated thumbnail, the title of the video, the audio associated with the video and the video itself and featurize each of them by employing appropriate feature extractors. In the second step, after featurizing these four subjective attributes of the video, we get $\mathcal{F}_{th}$ representing the feature vector for thumbnail, $\mathcal{F}_{ti}$ representing the feature vector for title, $\mathcal{F}_{A}$ representing the feature vector for audio and $\mathcal{F}_{v}$ representing the feature vector for video as shown in

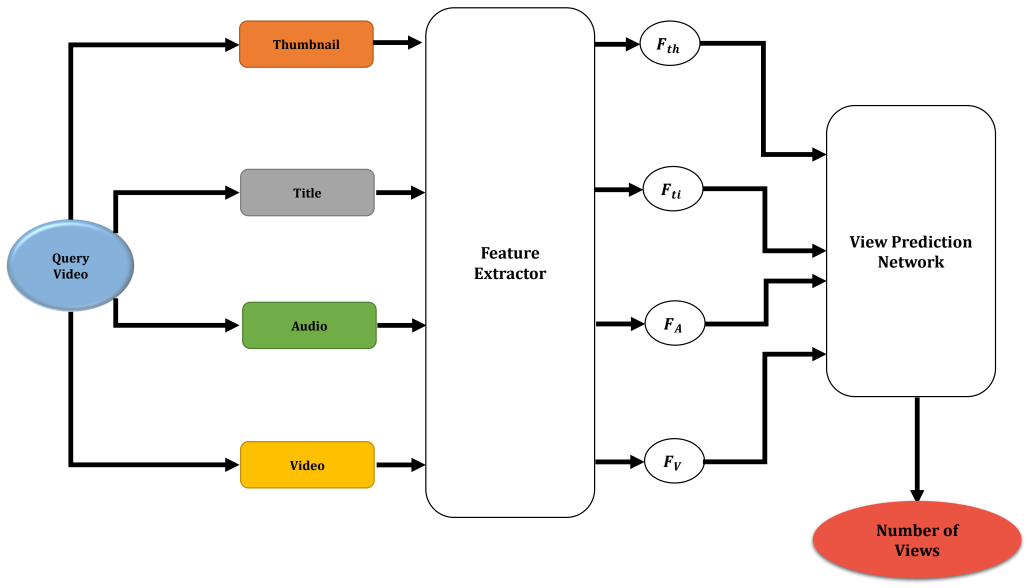


Figure 1.1: Block Diagram of our Deep Learning based solution

the figure 1.1. These four feature vectors are taken as input into our deep learning module which is the view prediction network, that generates the number of views the video can get over a certain duration of time. Both the featurizing and the deep learning module have been explained in detail in chapter 4.

1.6 Contribution

We have proposed a first-of-its-kind mathematical model that relates the different subjective attributes of video with its popularity in terms of the number of views. Our model can potentially predict the view of a video at any time instant after the release. Therefore, the proposed method, implemented using a complex deep neural network architecture, can be used to identify the potentially viral videos as well. Unlike the existing method mentioned in section 1.3.1, our model is quite integrated or comprehensive taking into account all the different aspects of the video involving thumbnail, title, audio and the video itself compared to considering only the thumbnail along with the video and channel statistics in the existing method. Also, in case of predicting the number of views of a new video which is yet to release, our model can be a more feasible option compared to the one mentioned in the existing method which requires the video statistics as input to the model including the number of likes, dislikes, comments and tags the video receives. So, we have tried to design a model that takes four attributes of a video namely, the thumbnail image, the title of the video, the audio associated with the video and the video itself to predict its popularity.

1.7 Organization

The dissertation report is organised as follows:

- Chapter 2 explains the mathematical operators and models required to understand our deep learning architecture. The chapter also explains the losses and the training procedure employed to train our architecture.
- Chapter 3 explains the dataset and describes the various subjective video attributes that are extracted from the dataset.
- Chapter 4 explains our proposed solution in details.
- Chapter 5 explains the results we obtained from our experiment and also the performance compared between our model and the competing one.
- Chapter 6 involves the conclusion of our experiment.
- Chapter 7 explains the shortcomings of our experiment which can be directed as future works.

Chapter 2

Prerequisites

In this chapter we provide a very brief description of concepts necessary to explain our approach and experiments for easier understanding of the reader.

2.1 Preliminaries

We need to understand both Convolution and Maxpooling in order to understand the Convolutional Neural Networks, which act as spatial feature extractors for images. So, a brief understanding of these mathematical operations and how they are performed on images is crucial and hence they are discussed here.

2.1.1 Convolution

The convolution operation involves sliding a matrix of weights known as the Kernel over an image and extract meaningful features from it [19]. This is similar to applying filters in image processing to extract edges, corners etc.

To understand the operation of convolution, let us consider an image I and a filter (kernel) K of dimensions $k_1 \times k_2$, then the convolution operation at (i, j) pixel of the convolved image is given by:

$$(I * k)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i-m, j-n).K(m, n), \tag{2.1}$$

where $0 \leq i \leq h$; $0 \leq j \leq w$; h and w be the size of the convolved image; $(I * K)$ is the convolved image, $I(i-m, j-n)$ is the pixel value at the $(i-m, j-n)$ position of the image I and $K(m, n)$ is the weight in the (m, n) position of the matrix of weights of kernel K. It is to be noted that when we are running the sum m, n across the kernel dimensions we are essentially placing an inverted kernel over a particular region of the image I of the size same as that of kernel and calculate element wise multiplication to find the convolution output at the (i, j) position of the convolved image. Now on further observation we can visualize that in order to calculate the entire convolved image, we essentially need to slide the inverted kernel over the entire image I to get $(I * K)$.

In order to draw a relationship between the size of the convolved image and the original image we need to understand two additional terms associated with convolution operation which are stride and padding.

Stride

We observed that in order to find the convolved image $(I * K)$, we slide the inverted kernel K over the entire image I. So the term stride refers to the the amount of slide the kernel K is shifted on the image I to carry out the next convolution operation along any dimension of the image I. Let us denoted stride as s.

Padding

Padding involves bordering the image with a layer of zeros, often used to handle odd dimensions. Let us denote padding as p.

Relation between the size of convolved image and original image

For simplicity, let us consider a square input image I of size $n \times n$, a square kernel K of size $k \times k$, stride s, padding p and the square convolved image of size $q \times q$. So, the relation between q and n is as follows:

$$q = \left(\left\lfloor \frac{n - k + 2 \times p}{s} \right\rfloor + 1 \right). \tag{2.2}$$

2.1.2 Maxpooling

This operation simply chooses the pixel value that is maximum among a group of pixels that occur within the pooling filter as the maxpooling kernel slides over the input image. It is mainly used to reduce dimensions of extracted feature maps [19].

2.2 Convolutional Neural Network

Convolution Neural Networks or CNN was developed for image classification [19]. We are focusing our point of discussion on the topic CNN with the sole objective to extract salient spatial features from thumbnail images associated with the video. CNN is made up of successive convolution layers having varying kernel sizes. They are followed by conventional hidden (also known as dense) layers that are present in traditional neural networks. Successive convolution layers act as automatic feature extractors while the dense layers serve as conventional neural network classifiers. Typically as we progress deeper into the network, the size of the extracted feature maps from the previous layer goes on decreasing. To ensure that an optimal number of features gets extracted the number of feature maps extracted in is gradually increased. So we have discussed here the first variant of CNN and that is AlexNet.

2.2.1 Why AlexNet?

Krizhevsky et al. [19] describes the Deep Convolutional Neural Network for Image Classification. The architecture contains eight learned layers - five convolutional layers and three fully connected layers. We have chosen AlexNet because it is light in the terms of parameters involving only the convolutional layers and utilizes properties like ReLU as activation function, local response normalization and overlapping pooling. We discuss here the justification of utilizing these features as follows:

Why is ReLU as an activation function?

The model that we are trying to develop involves a lot of layers rendering our model to be deep enough. So, inspite of modelling a neuron's output using tanh or sigmoid activation function which suffers from saturating nonlinearities, the network uses Rectified Linear units as an activation function which performs faster in training times. ReLU provides non-saturating nonlinearity $f(x) = max(0, x)$ as described by Nair and Hinton [20]. Consequently, the derivation of the ReLU activation function being non-saturating does not suffer from vanishing gradient problem as that in case of tanh or sigmoid activation.

Why local response normalization?

Local response normalization (LRN) implements a form of lateral inhibition inspired by the type found in real neurons, creating competition for big activities among neuron outputs computed using different kernels. So, LRN allows to diminish responses that are uniformly large for the neighborhood and make large activation more pronounced within a neighborhood i.e. create higher contrast in activation map. This is why LRN is particularly useful with unbounded activation functions as ReLU. However, it is found that the following local normalization scheme aids generalization. Denoting by $a^i_{x,y}$ the activity of a neuron computed by applying kernel i at position (x, y) and then applying the ReLU nonlinearity, the response-normalized activity $b^i_{x,y}$ is given by the expression:

$$b^i_{x,y} = a^i_{x,y}/(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2)^\beta, \tag{2.3}$$

where the sum runs over n “adjacent” kernel maps at the same spatial position, and N is the total number of kernels in the layer. The ordering of the kernel maps is of course arbitrary and determined before training begins. This sort of response normalization implements a form of lateral inhibition inspired by the type found in real neurons, creating competition for big activities amongst neuron outputs computed using different kernels. The constants k,n,α and β are hyper-parameters whose values are determined using a validation set. This normalization is applied after ReLU nonlinearity in certain layers of the AlexNet architecture.

Why overlapping pooling?

Pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map. Now, if the pooling layers do not overlap, the pooling regions are disjointed and a lot of information is lost in each pooling layer. But if some overlap is allowed, the pooling regions overlap with some degree and less spatial information is lost in each layer.Loss of spatial information by pooling even if is thought to give some degree of spatial invariance to CNNs can be detrimental if abused because it can lead to overfitting as the network will focus only on some dominant features; but because the pooling regions are disjointed, it looses quickly any information (in higher layers) of where the feature is located in the image. This obviously can happens for high capacity, deep, models. Letting the pooling regions overlap allows to mitigate for this effect.

2.3 Sentiment intensity analyzer

Sentiment analysis, or Opinion mining, is a sub-field of Natural Language Processing (NLP) that tries to identify and extract opinions within a given text. The aim of sentiment analysis is to gauge the attitude, sentiments, evaluations, attitudes and emotions of a speaker/writer based on the computational treatment of subjectivity in a text. With respect to our project, we need Sentiment intensity analyzer to extract salient sentimental features of the title caption associated with the video, giving a polarity score of positive, neutral and negative sentiment of the title respectively.

2.3.1 Why sentiment analysis is so important?

Businesses today are heavily dependent on data. Majority of this data however, is unstructured text coming from sources like emails, chats, social media, surveys, articles, and documents. The micro-blogging content coming from Twitter and Facebook poses serious challenges, not only because of the amount of data involved, but also because of the kind of language used in them to express sentiments, i.e., short forms, memes and emoticons. Sentiment Analysis is also useful for practitioners and researchers, especially in fields like sociology, marketing, advertising, psychology, economics, and political science, which rely a lot on human-computer interaction data. Sentiment Analysis enables companies to make sense out of data by being able to automate this entire process! Thus they are able to elicit vital insights from a vast unstructured dataset without having to manually indulge with it.

2.3.2 Why do we use VADER for sentiment analysis?

We chose VADER [21] because it has a lot of advantages over traditional methods of Sentimental analysis, including:

- It works exceedingly well on social media type text, yet readily generalizes to multiple domains
- It is fast enough to be used online with streaming data, and
- It does not severely suffer from a speed-performance trade-off.

2.3.3 Sentiment analysis using VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative. VADER has been found to be quite successful when dealing with social media texts, NY Times editorials, movie reviews, and product reviews. This is because VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

So the Sentiment intensity analyzer used in this research work is a model implemented in NLTK package using VADER motivated from the research paper [21].

2.3.4 How VADER works?

VADER analyzes sentiments based on certain key points. They are:

- Punctuation: The use of an exclamation mark(!), increases the magnitude of the intensity without modifying the semantic orientation.
- Capitalization: Using upper case letters to emphasize a sentiment-relevant word in the presence of other non-capitalized words, increases the magnitude of the sentiment intensity.
- Degree modifiers: Also called intensifiers, they impact the sentiment intensity by either increasing or decreasing the intensity.
- Conjunctions: Use of conjunctions like "but" signals a shift in sentiment polarity, with the sentiment of the text following the conjunction being dominant.
- Preceding Tri-gram: By examining the tri-gram preceding a sentiment-laden lexical feature, we can catch nearly 90% of cases where negation flips the polarity of the text.

2.4 Long Short Term Memory

One of the appeals of Recurrent Neural Networks (RNNs) is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. Sometimes, we only need to look at recent information to perform the present task. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information. But there are also cases where we need more context. Unfortunately, as that gap grows, RNNs become unable to learn to connect the information. On the other hand Long Short Term Memory (LSTMs) [22] are explicitly designed to avoid the long-term dependency problem thereby remembering information for long periods of time. With respect to our research project, we would necessarily require LSTMs in order to learn long-term dependencies to gain an information of the overall waveform of the audio signal and also to extract salient temporal features of the audio waveform associated with the video.

The LSTMs achieve these objectives by maintaining two internal states along the entire input sequence known as the hidden state denoted by h and the cell state denoted by c. The hidden state usually contains the output of the LSTM module at every instant of the input sequence and the cell state usually contains the information that needs to be passed across all the instants of the input sequences. Let the input sequence of audio waveform be denoted by X_A, and let the sequence comprise of T time steps, then at time step t, the LSTM module receives the following inputs:

- Previous cell state denoted as c_{t-1}
- Previous hidden state denoted as h_{t-1}

- Present preprocessed audio input denoted as $X_A^{(t)}$, as discussed in detail in section 3.6.3.

At any instant t, the LSTM module outputs the following:

- Present cell state denoted as c_t to pass the information to LSTM module at time step t+1.
- Present hidden state denoted as h_t which is the output of the LSTM module at time step t and is also passed to the LSTM module at time step t+1.

LSTMs achieve this ability to learn long term dependencies using four gates which are as follows:

2.4.1 The Forget Gate

The forget gate consists of a fully connected layer with sigmoid activation. At time step t, the gate takes the concatenated vector of h_{t-1} and $X_A^{(t)}$ as input denoted by $[h_{t-1}, X_A^{(t)}]$ and outputs a vector f_t. This gate decides the degree to which each dimension of the vector c_{t-1} is to be forgotten based on the contents of the concatenated vector of $[h_{t-1}, X_A^{(t)}]$ using a fully connected layer with sigmoid activation. If the i^{th} dimension of f_t is 0 (which we state here as degree), then we completely forget the i^{th} dimension of c_{t-1} and if the i^{th} dimension of f_t is 1, then we completely retain the i^{th} dimension of c_{t-1}. So we define f_t as follows:

$$f_t = \sigma(W_f.[h_{t-1}, X_A^{(t)}] + b_f). \tag{2.4}$$

The weights W_f and b_f are the parameters of the gate.

2.4.2 The Candidate Gate

The candidate gate consists of a fully connected layer with tanh activation. At time step t, the gate takes the concatenated vector of h_{t-1} and $X_A^{(t)}$ as input denoted by $[h_{t-1}, X_A^{(t)}]$ and outputs a vector $\tilde{c_t}$. This gate extracts the new information at the time step t, which we denote here as $\tilde{c_t}$ based on the present input $[h_{t-1}, X_A^{(t)}]$ that can "potentially" be incorporated into the new cell state c_t. The degree of inclusion of $\tilde{c_t}$ in c_t depends on the output of the input gate discussed in the section 2.4.3. So we define $\tilde{c_t}$ as follows:

$$\tilde{c_t} = \sigma(W_{\tilde{c}}.[h_{t-1}, X_A^{(t)}] + b_{\tilde{c}}). \tag{2.5}$$

The weights $W_{\tilde{c}}$ and $b_{\tilde{c}}$ are the parameters of the gate.

2.4.3 The Input Gate

The input gate consists of a fully connected layer with sigmoid activation. At time step t, the gate takes the concatenated vector of h_{t-1} and $X_A^{(t)}$ as input denoted by $[h_{t-1},X_A^{(t)}]$ and outputs a vector i_t. This gate decides the degree to which each dimension of the vector $\tilde{c_t}$ is incorporated into the new cell state c_t based on the contents of the present input $[h_{t-1},X_A^{(t)}]$ using a fully connected layer with sigmoid activation. If the i^{th} dimension of i_t is 0 (which we state here as degree), then we completely ignore the i^{th} dimension of $\tilde{c_t}$ in the i^{th} dimension of c_t and if the i^{th} dimension of i_t is 1, then we completely include the i^{th} dimension of $\tilde{c_t}$ at the i^{th} dimension of c_t. So we define i_t as follows:

$$i_t = \sigma(W_i.[h_{t-1}, X_A^{(t)}] + b_i). \tag{2.6}$$

The weights W_i and b_i are the parameters of the gate.

2.4.4 The Output Gate

The output gate consists of a fully connected layer with sigmoid activation. At time step t, the gate takes the concatenated vector of h_{t-1} and $X_A^{(t)}$ as input denoted by $[h_{t-1},X_A^{(t)}]$ and outputs a vector z_t. This gate decides the degree to which each dimension of the scaled version of the new cell state c_t (where the scaling factor is discussed in detail in section 2.4.5) is to be revealed as output of the LSTM module at the time instant t i.e h_t (discussed in section 2.4.5), based on the present input $[h_{t-1},X_A^{(t)}]$. If the i^{th} dimension of z_t is 0 (which we state here as degree), then we completely surpress the i^{th} dimension of the scaled c_t in the i^{th} dimension of output vector i.e. h_t (discussed in section 2.4.5) and if the i^{th} dimension of z_t is 1, then we completely reveal the i^{th} dimension of the scaled c_t in the i^{th} dimension of output vector i.e. h_t (discussed in section 2.4.5). So we define z_t as follows:

$$z_t = \sigma(W_z.[h_{t-1}, X^{(t)}] + b_z). \tag{2.7}$$

The weights W_z and b_z are the parameters of the gate.

2.4.5 The Cell State and Hidden State

As discussed earlier, the cell state is a passage containing the necessary information that needs to be carried across all the time steps of the input sequence to capture the long-term dependencies. The content of the new cell state at time instant t is obtained by necessary deleting the contents needed to be forgotten controlled by the forget gate based on the input at time step t and the hidden state obtained until time step t-1 i.e.$[h_{t-1},X_A^{(t)}]$ and adding the contents from the candidate gate controlled by the input gate based on the input at time step t and the hidden state obtained until time step t-1 i.e.$[h_{t-1}, X_A^{(t)}]$. The new cell state c_t can be defined as follows:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t, \tag{2.8}$$

where * stands for element-wise multiplication.

The hidden state captures the output of the LSTM by observing the input sequence upto a certain time instant. So if we want to observe an audio waveform with T time steps as input, we would definitely obtain its salient temporal features from observing the hidden state of the LSTM at the end of the sequence i.e. h_T. The contents of the new hidden state at any time instant t is obtained after appropriately scaling the the contents of the new cell state through a tanh function, thereby rescaling it between -1 to 1 and then surpressing or revealing the scaled new cell state by the output gate in the output vector i.e. h_t. The new hidden state h_t can be defined as follows:

$$h_t = z_t * tanh(c_t), \tag{2.9}$$

where * stands for element-wise multiplication.

2.5 Convolutional Long Short Term Memory

Convolutional LSTM is a different variant of LSTM, where it is used to handle sequence of images as input such as video. Let the input sequence of video be denoted by X_V, and let the sequence comprise of T time steps, then at time step t, the LSTM module receives the following inputs:

- Previous cell state denoted as c_{t-1}
- Previous hidden state denoted as h_{t-1}
- Present preprocessed video input denoted as $X_V^{(t)}$, as discussed in detail in section 3.6.4

The LSTM module outputs the following:

- Present cell state denoted as c_t to pass the information to LSTM module at time step t+1.
- Present hidden state denoted as h_t which is the output of the LSTM module at time step t and is also passed to the LSTM module at time step t+1.

Convolutional LSTMs achieve this ability to learn long term dependencies using four gates which are as follows:

2.5.1 The Forget Gate

The forget gate consists of a 2D convolutional layer alongwith sigmoid activation. At time step t, the gate takes the concatenated tensor of h_{t-1} and $X_V^{(t)}$ along the channel axis as input denoted by $[h_{t-1}$,$X_V^{(t)}]$ which is a 3-dimensional tensor and outputs a vector f_t which is also a 3-dimensional tensor. This gate decides the degree to which each dimension of the tensor c_{t-1} is to be forgotten based on the contents of the input tensor. If the i^{th} dimension of f_t is 0 (which we state here as degree), then we completely forget the i^{th} dimension of c_{t-1} and if the i^{th} dimension of f_t is 1, then we completely retain the i^{th} dimension of c_{t-1}. So we define f_t as follows:

$$f_t = \sigma(W_f * [h_{t-1}, X_V^{(t)}]). \tag{2.10}$$

The weights W_f is the parameter of the gate. Here * denotes convolution operation.

2.5.2 The Candidate Gate

The candidate gate consists of a 2D convolutional layer with tanh activation. At time step t, the gate takes the concatenated vector of h_{t-1} and $X_V^{(t)}$ as input denoted by $[h_{t-1}$,$X_V^{(t)}]$ and outputs a vector $\tilde{c_t}$ which is a 3-dimensional tensor. This gate extracts the new information from the frame at time step t, which we denote here as $\tilde{c_t}$, based on the input tensor that can "potentially" be incorporated into the new cell state c_t. The degree of inclusion of $\tilde{c_t}$ in c_t depends on the output of the input gate discussed in the section 2.5.3. So we define $\tilde{c_t}$ as follows:

$$\tilde{c_t} = \sigma(W_{\tilde{c}} * [h_{t-1}, X_V^{(t)}]). \tag{2.11}$$

The weights $W_{\tilde{c}}$ is the parameter of the gate. Here * denotes convolution operation.

2.5.3 The Input Gate

The input gate consists of a 2D convolutional layer alongwith sigmoid activation. At time step t, the gate takes the concatenated vector of h_{t-1} and $X_V^{(t)}$ as input denoted by $[h_{t-1}$,$X_V^{(t)}]$ and outputs a vector i_t.This gate decides the degree to which each dimension of the tensor $\tilde{c_t}$ is incorporated into the new cell state c_t based on the contents of the input tensor. If the i^{th} dimension of i_t is 0 (which we state here as degree), then we completely ignore the i^{th} dimension of $\tilde{c_t}$ in the i^{th} dimension of c_t and if the i^{th} dimension of f_t is 1, then we completely include the i^{th} dimension of $\tilde{c_t}$ at the i^{th} dimension of c_t. So we define i_t as follows:

$$i_t = \sigma(W_i * [h_{t-1}, X_V^{(t)}]). \tag{2.12}$$

The weights W_i is the parameter of the gate. Here * denotes convolution operation.

2.5.4 The Output Gate

The output gate consists of a 2D convolutional layer with sigmoid activation. At time step t, the gate takes the concatenated vector of h_{t-1} and $X_V^{(t)}$ as input denoted by $[h_{t-1}, X_V^{(t)}]$ and outputs a vector z_t. This gate decides the degree to which each dimension of the scaled version of the new cell state c_t (where the scaling factor is discussed in detail in section 2.5.5) is to be revealed as output of the Convolutional LSTM module at the time instant t i.e. h_t (discussed in section 2.5.5), based on the present input tensor. If the i^{th} dimension of z_t is 0 (which we state here as degree), then we completely surpress the i^{th} dimension of the scaled c_t tensor in the i^{th} dimension of output vector i.e. h_t (discussed in section 2.5.5) and if the i^{th} dimension of z_t is 1, then we completely reveal the i^{th} dimension of the scaled c_t tensor in the i^{th} dimension of output vector i.e. h_t (discussed in section 2.5.5). So we define z_t as follows:

$$z_t = \sigma(W_z * [h_{t-1}, X_V^{(t)}]). \tag{2.13}$$

The weights W_z is the parameter of the gate. Here * denotes convolution operation.

2.5.5 The Cell State and Hidden State

As discussed earlier, the cell state is a passage containing the necessary information that needs to be carried across all the time steps of the input sequence to capture the long-term dependencies. The content of the new cell state at time instant t is obtained by forgetting certain features from the past frames upto time instant t-1 and adding the new features or information obtained from the present frame at time instant t of the video sequence. The contents needed to be forgotten are controlled by the forget gate based on the input at time step t and the hidden state obtained until time step t-1 i.e. $[h_{t-1}, X_V^{(t)}]$ and adding the contents from the candidate gate controlled by the input gate based on the input at time step t and the hidden state obtained until time step t-1 i.e. $[h_{t-1}, X_V^{(t)}]$. The new cell state c_t can be defined as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{2.14}$$

where $\odot$ denotes the Hadamard operator.

The new hidden state at time instant t i.e h_t defines the salient information obtained by processing the frames of the video upto a time instant t. The contents of the new hidden state are the contents from the new cell state but are controlled by the output gate. The new hidden state h_t can be defined as follows:

$$h_t = z_t \odot tanh(c_t), \tag{2.15}$$

where $\odot$ denotes the Hadamard operator.

2.6 Support Vector Regressor

Support Vector Machines (SVM) [23] are well known in classification problems.The use of SVMs in regression finds applications typically through models such as Support Vector Regressors (SVR) [24]. We are discussing SVR in this section in order to explain the method which is implement in the model that we discussed in section 1.3.1. SVR gives us the flexibility to define how much error is acceptable in our model and find an appropriate line (or hyperplane in higher dimensions) to fit the data. The objective function of SVR is to minimize l2-norm of the coefficient vector. The error term is instead handled in the constraints, where we set the absolute error less than or equal to a specified margin, called the maximum error, ϵ (epsilon). We can tune epsilon to gain the desired accuracy of our model. The new objective function and the constraints are as follows:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & |y_i - w_i.x_i| \le \epsilon, \forall i \\ & . \end{aligned} \tag{2.16}$$

This algorithm doesn't work for all data points. The algorithm solved the objective function as best as possible but some of the points still fall outside the margins. As such, we need to account for the possibility of errors that are larger than ϵ. We can handle such situations with the help of slack variables which denoted for any value that falls outside of ϵ by its deviation from the margin denoting as ξ as shown in the figure 2.1. So we add these deviations to the objective function and get the new objective function as follows:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} |\xi_i| \\ \text{s.t.} \quad & |y_i - w_i.x_i| \le \epsilon + |\xi_i|, \forall i \\ & . \end{aligned} \tag{2.17}$$

2.7 Losses and Metrics

Neural networks fall in the domain of supervised learning. Training a neural network involves presenting it with a set of data for which the correct output, named as the ground truth is already known to us. The network provides us a predicted output. The difference between the ground truth and the predicted output is known as the loss. Metrics are methods of evaluating the output of the network. While loss is used to train the network, metric is used only for evaluation. A brief description of the loss functions and metrics used is provided in this section.

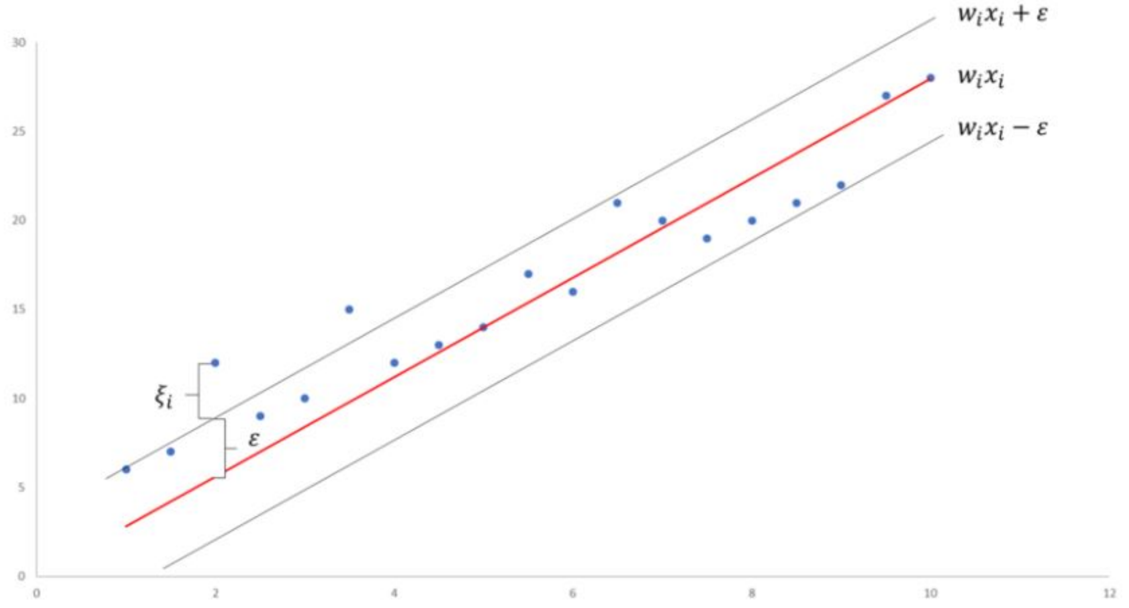


Figure 2.1: Illustrative Example of SVR with Slack Variables

2.7.1 Mean Squared Error

The loss function that we used to train our model is Mean Squared Error. The Mean Squared Error function used is as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}(V_i^{predicted} - V_i^{actual})^2, \tag{2.18}$$

where V_i^{actual} is the actual number of views i^{th} video gets, $V_i^{predicted}$ is the number of views i^{th} video can get predicted by the model and N is the number of training observations.

2.7.2 Binary Cross Entropy

Binary Cross Entropy (BCE) is a loss function derived from information theory [25]. Lets say the ground truth data comes from a distribution known as the true distribution $q(y)$ while the neural network predicts a result that comes from a distribution $p(y)$. Entropy is a measure of uncertainity of a distribution. Thus cross entropy becomes a measure of estimating how far away $p(y)$ is from $q(y)$. The purpose of training is to make $p(y)$ as close to $q(y)$ as possible. This loss is used to train the first stage of the model discussed in the section 1.3.1. Mathematically:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot (1 - \log(p(y_i))), \tag{2.19}$$

where N is the total number of datapoints. y_i is the ground truth value of the i_{th} datapoint and $p(y_i)$ is the predicted value for the current datapoint.

2.7.3 Mean Absolute Percentage Error

It is a measure of prediction accuracy of forecasting in statistics [26]. It is usually represented by the formula:

$$M = \frac{1}{N}\sum_{i=1}^{N} \frac{|V_i^{actual} - V_i^{predicted}|}{V_i^{actual}} \times 100, \tag{2.20}$$

where V_i^{actual} is the actual number of views i^{th} video gets, $V_i^{predicted}$ is the number of views i^{th} video can get predicted by the model and N is the total number of instances or datapoints. We have used this metric to evaluate our model in testing phase as well as to compare our model with the existing model discussed in section 5.5.

2.8 Training Neural Networks

As we will extensively use neural network layers in our model to generate a view count, so that involves a lot of weights or parameters to learn. This section describes in brief how the training of neural networks takes place in order to learn these weights to achieve a trained model desirable for our purpose.

2.8.1 The Gradient Descent Algorithm

Originally developed by Cauchy [27] this algorithm is used extensively for training neural networks. Training data is presented to the network and corresponding predictions are obtained. The value of the loss is calculated by comparing predictions with ground truth values. The objective of gradient descent is to minimize the loss by updating the weights of the network in the direction of steepest descent i.e the gradient. The algorithm converges when we have reached a local minima and no further reduction of the loss value is possible. Mathematically:

$$w_i^{'} = w_i - \eta\nabla(J), \tag{2.21}$$

where η is known as the learning rate which controls how fast the network reaches the local minima, w_i is the weight value, $w_i^{'}$ is the updated value of the weight w_i

and J is the calculated loss. Presenting the input to the network and obtaining the prediction is known as forward propagation while calculating loss, and updating weights using equation (2.21) is known as backpropagation. If backpropagation occurs after every forward propagation then it is called Stochastic Gradient Descent. When backpropagation occurs after forward propagating a batch of data and accumulation the net loss then it is known as Mini-batch Gradient Descent. We have used Mini-batch Gradient Descent as our training algorithm.

2.8.2 Regularization

This is a method used to fine tune predictions made by the network. It also helps in better training by reducing overfitting.

Dropout

Dropout regularization [28] is a technique that distributes the decision making process over the entire network. During training it may so happen that the weight value associated to a particular hidden neuron may become very high and it may start behaving as the deciding neuron for a particular class. Drop out randomly selects neurons and sets their activations to zero. This ensures the weight of those neuron don't get updated for the current pass. This maintains uniformity within the network.

2.8.3 ADAM Optimization

This optimization technique speeds up the training process. Instead of using a fixed learning rate η, ADAM [29] uses an adaptive learning rates for different parameters. Adaptability is achieved by using exponentially moving averages computed on gradients of the current mini-batch. We have utilized ADAM optimization to speed up our training process.

Chapter 3

The Dataset and its Description

This chapter describes the dataset that we used in this experiment. Several methods of preprocessing were needed to be carried out to featurize the different subjective attributes of the video dataset that we are trying to handle in this experiment. The dataset has been downloaded from kaggle [30].

3.1 General Description

The dataset contains the most daily trending videos from the countries like US, Great Britain, Germany, Canada and France, with the following attributes:

- Video ID: The unique ID assigned to the video in YouTube based on which you can access the video.
- Publishing Date: The date the video was released in YouTube.
- Trending Date: The date in which the video became trending after the date of it being published in YouTube.
- Thumbnail URL: The URL to download the thumbnail image corresponding to the video in YouTube.
- Title: The caption or title of the video in YouTube.
- Likes: The number of likes received by the video.
- Share: The number of people shared this video.

This dataset was collected using the YouTube API. We are concentrating on the trending videos from US only.

The four subjective attributes of video that we have focused on this research project are namely, the thumbnail image, the title of the video, the audio of the video and the video itself. The dataset that we considered here are of different genres involving Film and animation, Autos and vehicles, music, pets and animals, sports, short movies,travels and events,gaming, video blogging, people and blogs, comedy, entertainment,news and politics,education, science and technology,classics, drama, family and many more.

The dataset of trending videos from US contains on an overall of 6351 videos out of which we have focused on 800 videos of duration ranging between 30 seconds to 40 minutes. The video file size varies from 50 Mb to 500 Mb.

3.2 Thumbnail attribute

The thumbnail image of a video plays an important role to attract the user to click the video to view it in YouTube. So to capture the attractiveness, we have focused on this attribute.

3.3 Title attribute

The title of the video captures the sentimental context based on which the video might appeal to the user to click the video to view it. So this is the reason why we selected title as an attribute.

3.4 Audio attribute

The audio plays an important role in determining how pleasant the user responds to the video. In a study [31], it has been found that the audio in form of music is more appealing and has a greater effect than visuals implying the importance of audio. So due to this feature, we have focused on this subjective attribute as well.

3.5 Video attribute

The video is the most important aspect for determining whether the user will view it or not, because it has both spatial and temporal crucial features to decide this important factor. Due to this property, we have focused to include this subjective attribute as well.

Figure 3.1: Example of a thumbnail image [33]

3.6 Data Preprocessing

In this step, the data for the four subjective attributes namely, the thumbnail, the title, the audio and the video are pre-processed differently to featurize the data suitable enough to be used as input to the deep learning model.

3.6.1 Thumbnail

The thumbnail images downloaded from the YouTube, resized to a specific size depending on the pretrained model being used at its input. The resized thumbnail image is changed to RGB scale using OpenCV library package [32]. The image is then standardized locally by finding out the mean and standard deviation of the images per channel. An example of a thumbnail image is shown in the figure 3.1.

3.6.2 Title

The title as a text is preprocessed as follows:

- Removing Stop words: Stop words are those which do not affect the meaning of the sentence, thus are better removed. The NLTK package [34] contains a set of stop words based on which we can remove stop words.
- Stemming: : Replacing words with their roots, reducing different types of words with similar meanings. This helps in reducing the dimensionality of the feature set. We try to achieve text stemming using the Porter Stemmer algorithm [35] using the NLTK package [34].
- Generating a dictionary of words that are important for emoticons.
- Part of Speech Tagging: It assigns a tag to each word in the text and classifies a word to a specific category like a noun, verb, adjective etc. We implement POS tagging using the NLTK library package [34]. POS taggers are efficient for explicit feature extraction.

The title associated with the example of the thumbnail shown in the figure 3.1 is "**Red Sparrow—Official Trailer [HD]—20th Century FOX**".

3.6.3 Audio

The audio signal associated with the video is first convert to .wav file and using the input-output wavfile function of the SciPy package [36] we read the signal at its designated sampling rate. We then proceed to preprocess each of the signal as directed in one of the article [37] as follows:

- Pre-Emphasis: The first step is to apply a pre-emphasis filter on the signal to amplify the high frequencies. A pre-emphasis filter is useful in several ways:

  1. Balance the frequency spectrum since high frequencies usually have smaller magnitudes compared to lower frequencies
  2. avoid numerical problems during the Fourier transform operation
  3. may also improve the Signal-to-Noise Ratio (SNR).

  The pre-emphasis filter can be applied to a signal x using the first order filter in the following equation:

$$y(t) = x(t) - \alpha * x(t-1). \tag{3.1}$$

- Framing: After pre-emphasis, we need to split the signal into short-time frames. The rationale behind this step is that frequencies in a signal change over time, so in most cases it doesn't make sense to do the Fourier transform across the entire signal in that we would lose the frequency contours of the signal over time. To avoid that, we can safely assume that frequencies in a signal are stationary over a very short period of time. Therefore, by doing a Fourier transform over this short-time frame, we can obtain a good approximation of the frequency contours of the signal by concatenating adjacent frames. Typical frame sizes in speech processing range from 20 ms to 40 ms with 50% (+/-10%) overlap between consecutive frames. Popular settings are 25 ms for the frame size,and a 10 ms stride (15 ms overlap).

- Window: After slicing the signal into frames, we apply a window function such as the Hamming window to each frame. A Hamming window has the following form:

$$w[n] = 0.54 - 0.46.\cos\frac{2.\pi.n}{N-1}, \tag{3.2}$$

  where $0 \leq n \leq N-1, N$ is the window length. We apply the window function to the frames in order to counteract the assumption made by the FFT that the data is infinite and to reduce spectral leakage.

- Filter Banks: We compute the filter banks by applying triangular filters nearly 40 filters,on a Mel-scale to the power spectrum to extract frequency bands. The Mel-scale aims to mimic the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. We can convert between Hertz (f) and Mel (m) using the following equations:

$$m = 2595.\log_{10}\left(1 + \frac{f}{700}\right). \tag{3.3}$$

  Each filter in the filter bank is triangular having a response of 1 at the center frequency and decrease linearly towards 0 till it reaches the center frequencies of the two adjacent filters where the response is 0 as shown in the figure 3.2. This can be modelled as follows:

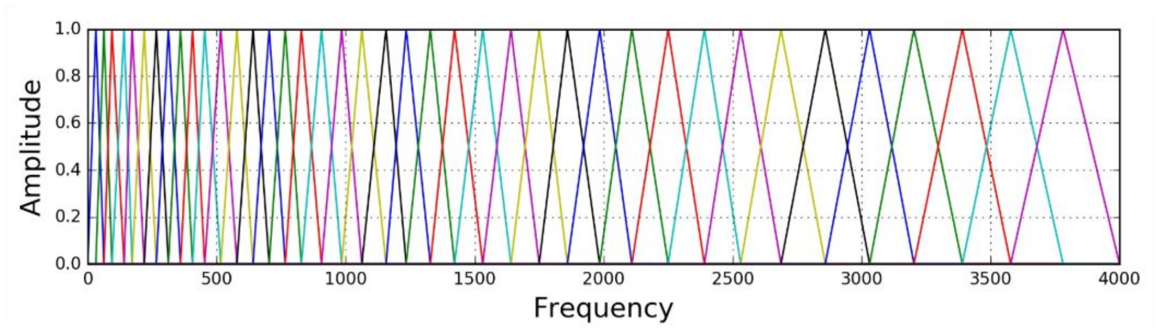


Figure 3.2: Filter Banks with triangular filters.

$$H_m(k) = \begin{cases} 0 & \text{if } k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & \text{if } f(m-1) \leq k < f(m) \\ 1 & \text{if } k = f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & \text{if } f(m) < k \leq f(m+1) \\ 0 & \text{if } k > f(m+1). \end{cases} \tag{3.4}$$

  After applying the filter bank we obtain the spectogram as shown in the figure 3.3.

- Mel Frequency Cepstral Coefficients: It turns out that filter bank coefficients computed in the previous step are highly correlated, which could be problematic in some machine learning algorithms. Therefore, we can apply Discrete Cosine Transform (DCT) to decorrelate the filter bank coefficients and yield a compressed representation of the filter banks. Typically, for Automatic Speech Recognition (ASR), the resulting cepstral coefficients 2-13 are retained and the rest are discarded. The mean normalized Mel Frequency Cepstral Coefficients is shown in the figure 3.4.

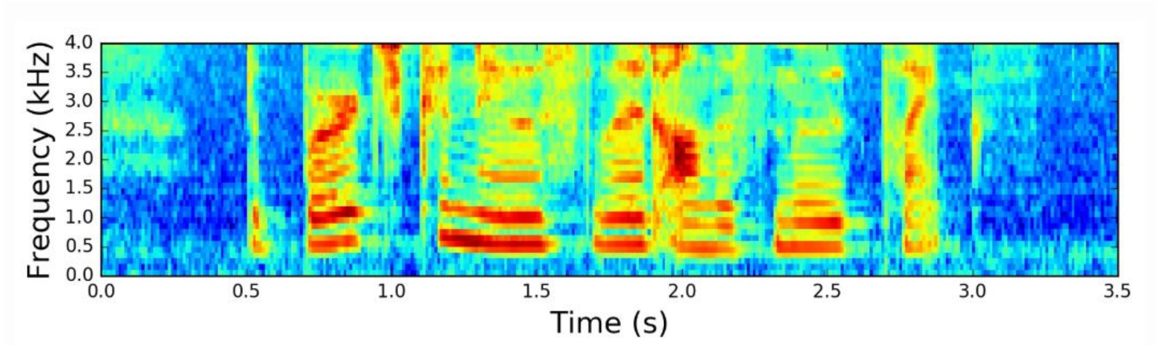


Figure 3.3: Spectogram obtained after applying filter banks.

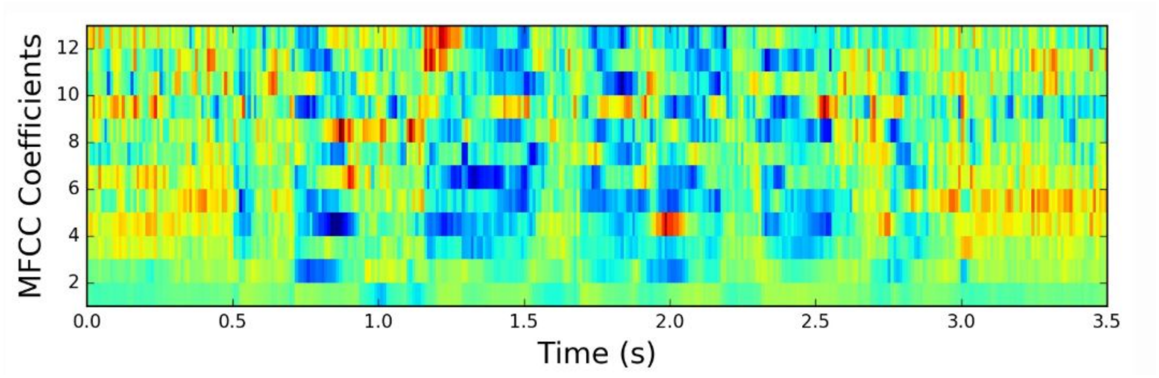


Figure 3.4: Mel Frequency Cepstral Coefficients.

- Delta MFCC: The Delta MFCC is obtained in order to understand the movement of formant frequencies. They can be understood using the equation:

$$\text{delta mfcc}[n] = \text{mfcc}[n+1] - \text{mfcc}[n]. \tag{3.5}$$

The MFCC and Delta MFCC are concatenated together to form the 24 dimensional feature vector for each frame, with 1024 frames in overall for a single audio file.

3.6.4 Video

The videos that we have in our database are of varying length ranging from 5 minutes to 35 minutes approximately. Considering an average frames per second to be around 25 fps, we would realize it would consist a lot of frames in the entire video duration. So in order to capture the essence of the entire video, we need to extract only essential frames which are informative enough thereby rejecting the redundant frames. These essential informative frames is what we term as I-frames also called Inter-coded frames or keyframes. These I-frames are detected observing the neighboring frames and thereby deducting to the fact whether the frame is informative or redundant, making

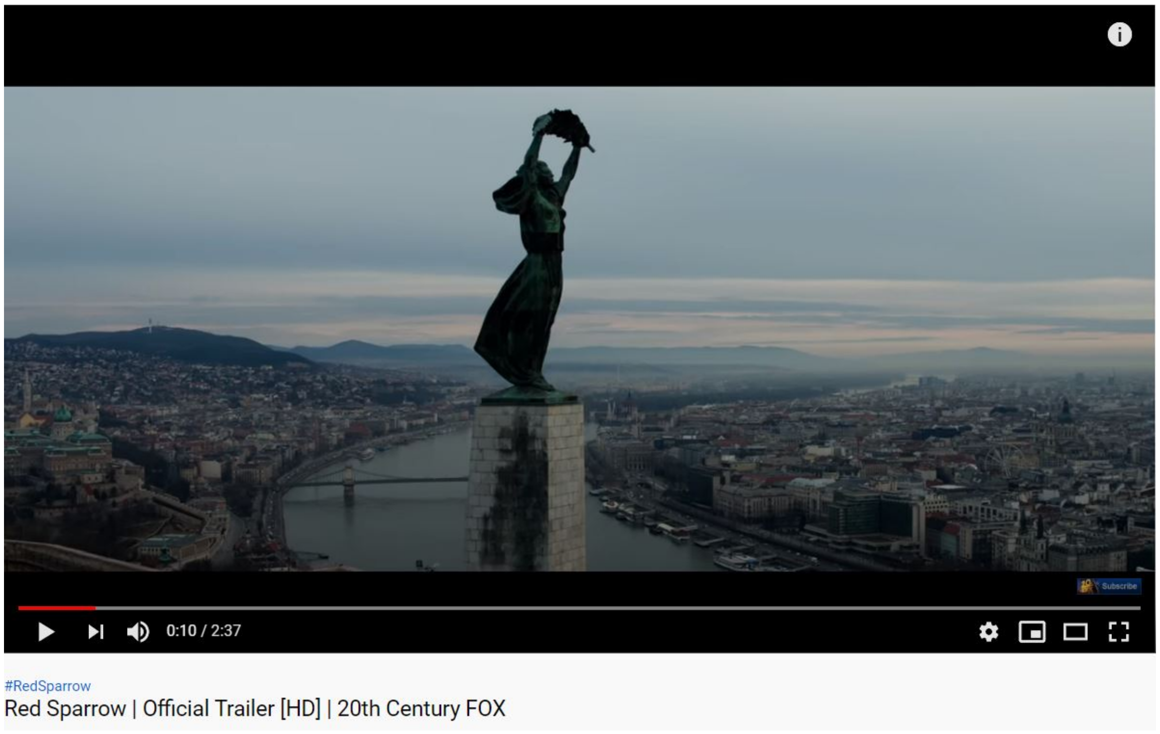


Figure 3.5: A snapshot of the video taken while playing in YouTube. [39]

it an I-frame or not. So this kind of video compression takes advantage from temporal redundancy between neighboring frames enabling higher compression rates. Since the videos are of unequal duration we have decided to sample equally spaced 25 I-frames from each video, such that these 25 I-frames best summarizes the entire video. These I-frames are extracted using ffmpeg software [38] and each frame is resized to a predetermined size using OpenCV library package [32]. The resized frames are then normalized using the local standardization where the mean and the standard deviation is obtained for each frame per channel. The snapshot of the video associated with the thumbnail shown in the figure 3.1 playing in YouTube is shown in the figure 3.5.

Chapter 4

The Proposed Solution

4.1 Proposed Solution

This section provides an elaborate description of our contribution to the research problem at hand. We firstly try to explain what makes a video viral, why would a video be clicked, liked and shared by the user and therefore try to mathematically model the number of views the video could get. We then try to explain the function our architecture tries to learn alongwith the implementation of the network architecture itself followed by the training details.

4.2 Factors that decide virality of a video from a user's perspective

As discussed earlier in section 1.2 we had come across some of the factors from a user's perspective regarding the number of views a video can get. They are: (a) why would a user click on a video? and (b) why would a user like the video and share it. In the next few paragraphs, we will aim to get the answer to these two questions.

4.2.1 Why would a User Click on a Video?

When a video appears in the search result of YouTube, we find the following mandatory items corresponding to each video: a thumbnail, a title, and the number of views (the number of YouTube users who have already viewed the video). An user typically tends to click on a video it if has

- an attractive thumbnail
- an attractive, relevant, informative title

- a large number of views

Therefore, the probability of clicking on a video can be modeled in terms of the attractiveness of the thumbnail, the attractiveness, the relevance, and the information contained in the title and the number of views the video already have.

Let $\mathcal{F}_{\text{th}}$ and $\mathcal{F}_{\text{ti}}$ be feature vectors corresponding to the thumbnail and the title. Assume that $\mathcal{F}_{\text{th}}$ captures the attractiveness of the thumbnail. Also assume that $\mathcal{F}_{\text{ti}}$ captures the attractiveness, the relevance and the information contained in the title. Let $V(t)$ be the total number of views of the video until time t and $P_{\text{C}}(t)$ be the probability that a new user (an user who have not already viewed the video) clicks the video at time t. Then based on the above discussion, we can model $P_{\text{C}}(t)$ as a function of $\mathcal{F}_{\text{th}}$, $\mathcal{F}_{\text{ti}}$, and $V(t)$:

$$P_{\text{C}}(t) = g_1(\mathcal{F}_{\text{th}}, \mathcal{F}_{\text{ti}}, V(t)), \tag{4.1}$$

where the function $g_1(\cdot)$ relates the probability of clicking the video at time t with $\mathcal{F}_{\text{th}}$, $\mathcal{F}_{\text{ti}}$, and $V(t)$.

4.2.2 Why would a User Like a Video and Share it?

Whether one would like a video or not strongly depends on primarily the quality of the video and the quality of the audio. As discussed earlier, the number of shares of video is dependent on how many users like the video. Another subtle factor that affects the number of sharing is the number of views a video already have. A highly viewed video tends to get shared more frequently. Therefore, we can model the probability of liking and sharing a video in terms of audio and video quality and the number of views it already has.

Let $\mathcal{F}_{\text{A}}$ and $\mathcal{F}_{\text{V}}$ be feature vectors corresponding to the audio and the video. Assume that $\mathcal{F}_{\text{A}}$ and $\mathcal{F}_{\text{V}}$ capture the quality of the audio and the video respectively. Let $P_{\text{LS}}(t)$ be the probability that a user likes and shares the video at time t. We can model $P_{\text{LS}}(t)$ in terms of $\mathcal{F}_{\text{A}}$, $\mathcal{F}_{\text{V}}$, and $V(t)$:

$$P_{\text{LS}}(t) = g_2(\mathcal{F}_{\text{A}}, \mathcal{F}_{\text{V}}, V(t)), \tag{4.2}$$

with $g_2(\cdot)$, being the function that relates $P_{\text{LS}}(t)$ with $\mathcal{F}_{\text{A}}$, $\mathcal{F}_{\text{V}}$, and $V(t)$.

4.3 Modeling the Number of Views

As discussed earlier, the number of views depends on the probability of clicking a video $P_{\text{C}}(\cdot)$ and the probability of liking and sharing the video $P_{\text{LS}}(\cdot)$. Let $V(t)$ be the total number of views of the video until time t and $\frac{\partial V(t)}{\partial t}$ be the time rate of change of views at time t. Hence, using (4.1) and (4.2), we write

$$\begin{aligned}\frac{\partial V(t)}{\partial t} &= G^*(P_{\mathrm{C}}(t), P_{\mathrm{LS}}(t)) \\ &= G^*(g_1(\mathcal{F}_{\mathrm{th}}, \mathcal{F}_{\mathrm{ti}}, V(t)), g_2(\mathcal{F}_{\mathrm{A}}, \mathcal{F}_{\mathrm{V}}, V(t))).\end{aligned} \tag{4.3}$$

The function $G^*(\cdot)$ is unknown and we need to learn it from the training data. Notice that the right hand side of (4.3) has time dependent component $V(t)$ and also time independent components $\mathcal{F}_{\mathrm{th}}$, $\mathcal{F}_{\mathrm{ti}}$, $\mathcal{F}_{\mathrm{A}}$, and $\mathcal{F}_{\mathrm{V}}$. We rearrange the time dependent and the time independent components of $G^*(\cdot)$ in terms of a composite function $G_r(\cdot)$ as follows:

$$\begin{aligned}\frac{\partial V(t)}{\partial t} &= G^*(g_1(\mathcal{F}_{\mathrm{th}}, \mathcal{F}_{\mathrm{ti}}, V(t)), g_2(\mathcal{F}_{\mathrm{A}}, \mathcal{F}_{\mathrm{V}}, V(t))) \\ &= G_r(V(t), \mathcal{F}_{\mathrm{th}}, \mathcal{F}_{\mathrm{ti}}, \mathcal{F}_{\mathrm{A}}, \mathcal{F}_{\mathrm{V}}).\end{aligned} \tag{4.4}$$

It is challenging to find an exact expression for the function $G_r(\cdot)$. However, we can have some idea about the nature of the function $G_r(\cdot)$. Let the maximum possible number of YouTube users be N which is a constant. Therefore as $V(t)$ increases, the maximum possible value $\frac{\partial V(t)}{\partial t}$ has to decrease. Consider the following expression:

$$\begin{aligned}& G_r(V(t), \mathcal{F}_{\mathrm{th}}, \mathcal{F}_{\mathrm{ti}}, \mathcal{F}_{\mathrm{A}}, \mathcal{F}_{\mathrm{V}}) \\ =\ & (N - V(t))\, G(\mathcal{F}_{\mathrm{th}}, \mathcal{F}_{\mathrm{ti}}, \mathcal{F}_{\mathrm{A}}, \mathcal{F}_{\mathrm{V}})\end{aligned} \tag{4.5}$$

This form of $G_r(\cdot)$ satisfies our requirement. Putting (4.5) in (4.4), we get:

$$\frac{\partial V(t)}{\partial t} = (N - V(t))\, G(\mathcal{F}_{\mathrm{th}}, \mathcal{F}_{\mathrm{ti}}, \mathcal{F}_{\mathrm{A}}, \mathcal{F}_{\mathrm{V}}). \tag{4.6}$$

The solution of (4.6) leads us to the number of views at time instant t:

$$V(t) = N - \exp\left\{-G(\mathcal{F}_{\mathrm{th}}, \mathcal{F}_{\mathrm{ti}}, \mathcal{F}_{\mathrm{A}}, \mathcal{F}_{\mathrm{V}})t + C\right\}, \tag{4.7}$$

where C is the integration constant. The value of C can be found considering zero views at the beginning ($t = 0$).

Based on the definition of a viral video, let us call a video to be a viral video if the view crosses a threshold N_{th} within a time t_v after the release of the video. From (4.7), we can find the number of views till time t_v:

$$N_v = N - \exp\left\{-G(\mathcal{F}_{\mathrm{th}}, \mathcal{F}_{\mathrm{ti}}, \mathcal{F}_{\mathrm{A}}, \mathcal{F}_{\mathrm{V}})t_v + C\right\}. \tag{4.8}$$

A video would be called viral if $N_v > N_{\mathrm{th}}$. Therefore, from (4.8), in order for a video to be viral, we need

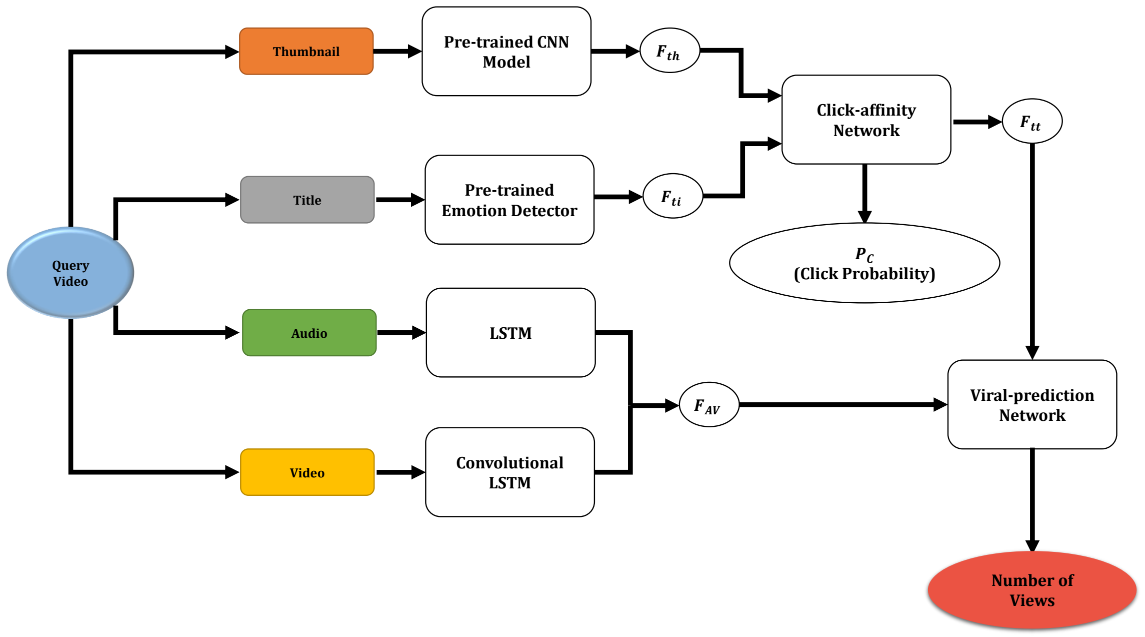


Figure 4.1: The block diagram of the proposed model

$$\begin{aligned} & N - \exp\left\{-G(\mathcal{F}_{\text{th}}, \mathcal{F}_{\text{ti}}, \mathcal{F}_{\text{A}}, \mathcal{F}_{\text{V}})t_v + C\right\} > N_{\text{th}} \\ \Rightarrow\; & G(\mathcal{F}_{\text{th}}, \mathcal{F}_{\text{ti}}, \mathcal{F}_{\text{A}}, \mathcal{F}_{\text{V}}) > \frac{1}{t_v}\left(C - \ln\left(N - N_{\text{th}}\right)\right). \end{aligned} \tag{4.9}$$

Notice that we know all the quantities in the right hand side of (4.9). The only unknown quantity in the above equation is the function $G(\cdot)$. Therefore, in order to predict whether a video would be viral, we need to learn $G(\cdot)$. We propose a neural network model to learn $G(\cdot)$.

4.4 The Network Model

In order to learn $G(\cdot)$, we need the thumbnail feature vector $\mathcal{F}_{\text{th}}$, title feature vector $\mathcal{F}_{\text{ti}}$, audio feature vector $\mathcal{F}_{\text{A}}$, and video feature vector $\mathcal{F}_{\text{V}}$. Next we discuss how to find these feature vectors.

4.4.1 Thumbnail Feature Vector $\mathcal{F}_{\text{th}}$

The feature vector corresponding to the thumbnail is generated using a AlexNet network, pre-trained on ImageNet.

4.4.2 Title Feature Vector $\mathcal{F}_{\text{ti}}$

We use an emotion detector to find the title feature vector $\mathcal{F}_{\text{ti}}$. We take the pre-trained model of Sentiment intensity analyzer [40] on VADER that generates a feature vector corresponding to the title of the video. In particular, we take the output from the network as $\mathcal{F}_{\text{ti}}$ expressing the polarity scores of positivity, negativity and neutral sentiments associated with the title.

4.4.3 Click-affinity Network

We have already argued that the probability of clicking a video is dependent on $\mathcal{F}_{\text{th}}$ and $\mathcal{F}_{\text{ti}}$. Hence we use $\mathcal{F}_{\text{th}}$ and $\mathcal{F}_{\text{ti}}$ to predict the probability of clicking on video through a neural network regressor. The regressor is trained with the view counts corresponding to the training videos. We can find the click probability corresponding to the training videos but YouTube does not provide any click probability corresponding to a video. Therefore, we define the click probability of a video. Consider a video released by some YouTube channel with N_{ch} number of subscribers. When the video is released, all the subscribers of the channel receives a notification. Therefore, N_{ch} is the minimum number of users who are aware of the release of the video. Let the total number of views of the video be N_v at $t = t_v$. If a video is viral, N_v usually becomes much greater than N_{ch} at $t = t_v$. In that case, we assign a click-probability value $P_{\text{C}}(t_v) = 1$ to the video; indicating that the number of clicks on the video has crossed the number of subscribers. Otherwise, we define $P_{\text{C}}(t_v)$ as a ratio of N_v and N_{ch}:

$$P_{\text{C}}(t_v) = \begin{cases} \frac{N_v}{N_{\text{ch}}}, \text{ if } N_v < N_{\text{ch}} \\ 1, \text{ otherwise.} \end{cases} \tag{4.10}$$

The input to the regressor is the the concatenation of the thumbnail feature vector $\mathcal{F}_{\text{th}}$ and the title feature vector $\mathcal{F}_{\text{ti}}$. Once trained, the click-prediction network can predict the probability that a user clicks the video. The output from the penultimate layer of the click-prediction network indicates the quality of the thumbnail and the title in making a user click a video. Let this vector be $\mathcal{F}_{\text{tt}}$. We use $\mathcal{F}_{\text{tt}}$ to predict the view of the video.

4.4.4 Audio and Video Feature Vectors ($\mathcal{F}_{\mathbf{A}}$, $\mathcal{F}_{\mathbf{V}}$)

We have already explained that the number of likes and shares are strongly dependent on the qualities of the audio and the video. Therefore, we extract the temporal feature vectors corresponding to the audio using a LSTM network and the spatial temporal feature vectors corresponding to the video using a Convolutional LSTM

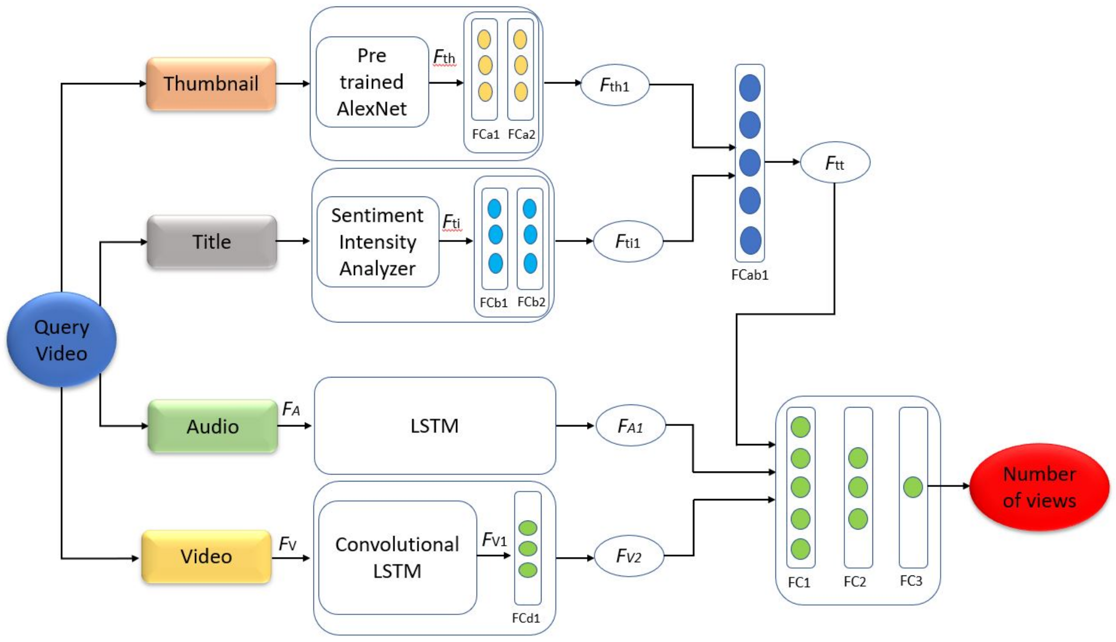


Figure 4.2: The proposed model

network. We find a composite audio-video feature vector $\mathcal{F}_{\text{AV}}$ by concatenating the two feature vectors. The concatenated feature vector $\mathcal{F}_{AV}$ represents the features which are relevant for a user to decide to like or share a video.

4.4.5 Viral-video-prediction Network

Finally, we design the viral video prediction network that takes the outputs from the click-affinity network and the concatenated feature vector $\mathcal{F}_{AV}$. $\mathcal{F}_{tt}$ from the click-prediction network and $\mathcal{F}_{AV}$ are concatenated and fed to the viral-video-prediction network which is a regressor designed to predict N_v, the number of views at $t = t_v$. If $N_v > N_{\text{th}}$, the video is said to be viral. Thus, from the viral-prediction network, we find the number of views and can predict if the video is going to be viral. A block diagram of the proposed model is presented in Fig 4.1.

4.5 Understanding the function $G(\cdot)$

There are many components in the network which are crucial for understanding the function $G(\cdot)$ that the network learns.

4.5.1 Thumbnail network

The feature vector corresponding to the thumbnail is generated using a Alexnet network, pre-trained on ImageNet, keeping initial five convolution layers fixed, followed by two fully connected layers denoted by FCa1 and FCa2, as shown in the figure 4.2. The pretrained Alexnet provides a mapping: $g_{th1}(\cdot) : Th \longrightarrow \mathcal{F}_{th}$, where $Th \in \mathbb{R}^{227*227*3}$ represents the input image and $\mathcal{F}_{th} \in \mathbb{R}^{43264*1}$.

$$\mathcal{F}_{th} = g_{th1}(Th). \tag{4.11}$$

The two fully connected layers denoted by FCa1 and FCa2 as shown in figure 4.2, learns a mapping: $g_{th2}(\cdot) : \mathcal{F}_{th} \longrightarrow \mathcal{F}_{th1}$, where $\mathcal{F}_{th1} \in \mathbb{R}^{1024*1}$.

$$\begin{aligned} \mathcal{F}_{th1} &= g_{th2}(\mathcal{F}_{th}), \\ g_{th2}(\mathcal{F}_{th}) &= W_{th}.\mathcal{F}_{th}, \end{aligned} \tag{4.12}$$

where $W_{th} \in \mathbb{R}^{1024*43264}$ represents the mapping learnt by the two fully connected layers FCa1 and FCa2 as shown in figure 4.2. So the thumbnail network represents a mapping: $g_{th}(\cdot) : Th \longrightarrow \mathcal{F}_{th1}$ where:

$$g_{th}(\cdot) = g_{th2} \circ g_{th1}(\cdot),$$

where $f \circ g$ represents a composite function i.e $f(g(\cdot))$. So we get:

$$\begin{aligned} \mathcal{F}_{th1} &= g_{th2} \circ g_{th1}(Th), \\ &= g_{th2}(g_{th1}(Th)). \end{aligned}$$

From equation (4.11), we get:

$$\mathcal{F}_{th1} = g_{th2}(\mathcal{F}_{th}).$$

From equation (4.12), we get:

$$\mathcal{F}_{th1} = W_{th}.\mathcal{F}_{th}. \tag{4.13}$$

The partial derivative of $\mathcal{F}_{th1}$ with respect to $\mathcal{F}_{th}$ is:

$$\frac{\partial \mathcal{F}_{th1}}{\partial W_{th}} = \mathcal{F}_{th}. \tag{4.14}$$

4.5.2 Title Network

The feature vector corresponding to the title is generated using a Sentiment intensity analyzer [40], pretrained on irony and sarcasm based tweets alongwith punctuations and special symbols, to capture the strong emotion of the Twitter users for sentiment analysis task, followed by two fully connected layers denoted by FCb1 and FCb2, as shown in the figure. In our case we utilize the model to extract sentiments from the title of the video. The input to the Sentiment intensity analyzer model is the title of the video in the form of one hot encoded vector and the output is the vector of length three depicting the probability score of a title to be positive, negative and neutral sentiment, denoted by $\mathcal{F}_{ti}$ as shown in the figure 4.2. We had taken these sentiments into account as they reflect the affinity of a viewer to click the video. The pretrained Sentiment Intensity Analyzer provides a mapping: $g_{ti1}(\cdot) : Ti \longrightarrow \mathcal{F}_{ti}$, where $Ti \in \mathbb{R}^{|V|}$ represents the one hot vector of the title, $|V|$: represents the vocabulary size of the corpus on which the Sentiment instensity analyzer is pretrained and $\mathcal{F}_{ti} \in \mathbb{R}^{3\times 1}$ represents the three polarity scores.

$$\mathcal{F}_{ti} = g_{ti1}(Ti). \tag{4.15}$$

The two fully connected layers denoted by FCb1 and FCb2 as shown in figure, learns a mapping: $g_{ti2}(\cdot) : \mathcal{F}_{ti} \longrightarrow \mathcal{F}_{ti1}$, where $\mathcal{F}_{ti1} \in \mathbb{R}^{1024\times 1}$.

$$\begin{aligned} \mathcal{F}_{ti1} &= g_{ti2}(\mathcal{F}_{ti}), \\ g_{ti2}(\mathcal{F}_{ti}) &= W_{ti}.\mathcal{F}_{ti}, \end{aligned} \tag{4.16}$$

where $W_{ti} \in \mathbb{R}^{1024\times 3}$ represents the mapping learnt by the two fully connected layers FCb1 and FCb2 as shown in the figure. So the title network represents a mapping : $g_{ti}(\cdot) : Ti \longrightarrow \mathcal{F}_{ti1}$ where:

$$g_{th}(\cdot) = g_{th2} \circ g_{th1}(\cdot),$$

where $f \circ g$ represents a composite function i.e $f(g(\cdot))$. So we get:

$$\begin{aligned} \mathcal{F}_{ti1} &= g_{ti2} \circ g_{ti1}(Ti), \\ &= g_{ti2}(g_{ti1}(Ti)). \end{aligned}$$

From equation (4.15) we get:

$$\mathcal{F}_{ti1} = g_{ti2}(\mathcal{F}_{ti}).$$

From equation (4.16) we get:

$$\mathcal{F}_{ti1} = W_{ti}.\mathcal{F}_{ti}. \tag{4.17}$$

The partial derivative of $\mathcal{F}_{ti1}$ with respect to $\mathcal{F}_{ti}$ is:

$$\frac{\partial \mathcal{F}_{ti1}}{\partial W_{ti}} = \mathcal{F}_{ti}. \tag{4.18}$$

4.5.3 Click-affinity Network

We have already argued that the affinity of clicking a video is dependent on $\mathcal{F}_{th1}$ and $\mathcal{F}_{ti1}$. Hence we use $\mathcal{F}_{th1}$ and $\mathcal{F}_{ti1}$ to featurize the affinity of a user to click the video using a layer of neural network denoted by FCab1 as shown in the figure 4.2. The input to the network is the the concatenation of the thumbnail feature vector $\mathcal{F}_{th1}$ and the title feature vector $\mathcal{F}_{ti1}$ denoted by $\mathcal{F}_{thti}$. The output from the network indicates the quality of the thumbnail and the title in making a user click a video. Let this vector be $\mathcal{F}_{tt}$. We use $\mathcal{F}_{tt}$ to predict the view of the video. So the network learns a mapping: $g_{ca}(\cdot) : \mathcal{F}_{thti} \longrightarrow \mathcal{F}_{tt}$, where $\mathcal{F}_{thti} \in \mathrm{I\!R}^{2048\times1}$ represents the concatenation of $\mathcal{F}_{th1}$ and $\mathcal{F}_{ti1}$ and $\mathcal{F}_{tt} \in \mathrm{I\!R}^{1024\times1}$.

$$\begin{aligned}\mathcal{F}_{tt} &= g_{ca}(\mathcal{F}_{thti}),\\ g_{ca}(\mathcal{F}_{thti}) &= W_{ca}.\mathcal{F}_{thti},\\ \mathcal{F}_{tt} &= W_{ca}.\mathcal{F}_{thti},\end{aligned} \tag{4.19}$$

where $W_{ca} \in \mathrm{I\!R}^{1024\times2048}$ represents the mapping learnt by the fully connected layer FCab1 as shown in the figure. The partial derivative of $\mathcal{F}_{tt}$ with respect to W_{ca} is:

$$\frac{\partial \mathcal{F}_{tt}}{\partial W_{ca}} = \mathcal{F}_{thti}. \tag{4.20}$$

4.5.4 Audio Network

The audio network consists of Long Short Term Memory network which inputs the audio waveform in the form of frames where each frame is featured as a 24 dimensional vector. The input signal corresponding to one frame is denoted by $\mathcal{F}_A^{(t)} \in \mathrm{I\!R}^{24\times1}$. The hidden vector is denoted by $h_t \in \mathrm{I\!R}^{512\times1}$.The cell state is denoted by $c_t \in \mathrm{I\!R}^{512\times1}$. The input to the LSTM at time step t is the concatenation of h_{t-1} and $\mathcal{F}_A^{(t)}$ denoted by $[h_{t-1},\mathcal{F}_A^{(t)}]$ which we would like to represent here as $X_A^{(t)} \in \mathrm{I\!R}^{536\times1}$ The four gates of the LSTM include the forget gate, the input gate, the candidate gate and the output gate. Let the output of the forget gate be denoted by f_t then we can define f_t as :

$$f_t = \sigma(W_f.X_A^{(t)} + b_f), \tag{4.21}$$

where the weight $W_f \in \mathrm{I\!R}^{512\times536}$ and the bias $b_f \in \mathrm{I\!R}^{512\times1}$ denote the parameters of the fully connected layer with sigmoid activation present in the forget gate. Let the output of the input gate be denoted by i_t then we can define i_t as:

$$i_t = \sigma(W_i.X_A^{(t)} + b_i), \tag{4.22}$$

where the weight $W_i \in \mathrm{I\!R}^{512\times536}$ and the bias $b_i \in \mathrm{I\!R}^{512\times1}$ denote the parameters of the fully connected layer with sigmoid activation present in the input gate. Let the output gate be denoted by $\tilde{c_t}$ then we can define $\tilde{c_t}$ as:

$$\tilde{c_t} = \tanh\Big(W_{\tilde{c}}.X_A^{(t)} + b_{\tilde{c}}\Big), \tag{4.23}$$

where the weight $W_{\tilde{c}} \in \rm I\!R^{512\times536}$ and the bias $b_{\tilde{c}} \in \rm I\!R^{512\times1}$ denote the parameters of the fully connected layer with tanh activation present in the candidate gate. Let the output of the output gate be denoted by z_t then we can define z_t as:

$$z_t = \sigma(W_z.X_A^{(t)} + b_z), \tag{4.24}$$

where the weight $W_z \in \rm I\!R^{512\times536}$ and the bias $b_z \in \rm I\!R^{512\times1}$ denote the parameters of the fully connected layer with sigmoid activation present in the output gate. Let the cell state that we obtain at the time step t be denoted as c_t, then we define c_t as:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t. \tag{4.25}$$

Let the hidden state that we obtain at the time step t be denoted as h_t, then we define h_t as:

$$h_t = z_t * \tanh(c_t). \tag{4.26}$$

where * denotes element wise multiplication. The final output of the LSTM is given by:

$$\begin{aligned} \mathcal{F}_{A1} &= h_T = z_T * tanh(c_T), \\ &\Rightarrow \mathcal{F}_{A1} = g_{LSTM}(\mathcal{F}_A). \end{aligned} \tag{4.27}$$

We observe that $\mathcal{F}_{A1}$ depends on input $\mathcal{F}_A$, hidden states h and cell states c of all T time steps where $\mathcal{F}_{A1} \in \rm I\!R^{512*1}$.
So in general in order to observe the gradient equation for the weights that are being learned, we can concatenate W_f, W_i,$W_{\tilde{c}}$ and W_z to form W_{LSTM}. The partial derivative of the output $\mathcal{F}_{A1}$ with respect to W_{LSTM} is as follows:

$$\begin{aligned} \frac{\partial \mathcal{F}_{A1}}{\partial W_{LSTM}} &= \frac{\partial \mathcal{F}_{A1}}{\partial c_T}.\frac{\partial c_T}{\partial c_{T-1}}......\frac{\partial c_2}{\partial c_1} * \frac{\partial c_1}{\partial W_{LSTM}}, \\ \Rightarrow \frac{\partial \mathcal{F}_{A1}}{\partial W_{LSTM}} &= \frac{\partial \mathcal{F}_{A1}}{\partial c_T}.(\prod_{t=2}^{T} \frac{\partial c_t}{\partial c_{t-1}}).\frac{\partial c_1}{\partial W_{LSTM}}. \end{aligned} \tag{4.28}$$

Similarly we can concatenate b_f, b_i,$b_{\tilde{c}}$ and b_z to form B_{LSTM}. The partial derivative of the output $\mathcal{F}_{A1}$ with respect to B_{LSTM} is as follows:

$$\begin{aligned} \frac{\partial \mathcal{F}_{A1}}{\partial B_{LSTM}} &= \frac{\partial \mathcal{F}_{A1}}{\partial c_T}.\frac{\partial c_T}{\partial c_{T-1}}......\frac{\partial c_2}{\partial c_1}.\frac{\partial c_1}{\partial B_{LSTM}}, \\ \Rightarrow \frac{\partial \mathcal{F}_{A1}}{\partial B_{LSTM}} &= \frac{\partial \mathcal{F}_{A1}}{\partial c_T}.(\prod_{t=2}^{T} \frac{\partial c_t}{\partial c_{t-1}}).\frac{\partial c_1}{\partial B_{LSTM}}. \end{aligned} \tag{4.29}$$

In both the cases, the ratio $\frac{\partial c_t}{\partial c_{t-1}}$ can be expressed as follows:

$$\begin{aligned} \frac{\partial c_t}{\partial c_{t-1}} &= \sigma'(W_f.X_A^{(t)}) + f_t + \\ \sigma'(W_i.X_A^{(t)}).W_i.z_{t-1} &* \tanh'(c_{t-1}).\tilde{c}_t + \\ \sigma'(W_c.X_A^{(t)}).W_c.z_{t-1} &* \tanh'(c_{t-1}).i_t. \end{aligned} \tag{4.30}$$

4.5.5 Video Network

The video network consists of Convolutional LSTM which inputs video in the form of frames, where each frame is denoted by $\mathcal{F}_V^{(t)} \in \mathbb{R}^{64\times64\times3}$,hidden vector is denoted by $h_t \in \mathbb{R}^{64\times64\times10}$ and the cell state by $c_t \in \mathbb{R}^{64\times64\times10}$. At any instant t, the frame $\mathcal{F}_V^{(t)}$ concatenated with the hidden state at time instant t along the channel axis to form $[\mathcal{F}_V^{(t)}, h_{t-1}]$ which we can denote as $X_V^{(t)} \in \mathbb{R}^{64\times64\times13}$. $X_V^{(t)}$ is passed through a 2D Convolutional layer present in each of the four gates of the LSTM namely the forget gate, the input gate, the candidate gate and the output gate followed by their respective activations. Let the output of forget gate be denoted by f_t, then we can define f_t as:

$$f_t = \sigma(X_V^{(t)} * W_f), \tag{4.31}$$

where the convolution layer involves 10 filters each filter is of size 3×3, padding $= 1$ and stride $= 1$. So $W_f \in \mathbb{R}^{3\times3\times13\times10}$. Let the output of input gate be denoted by i_t, then we can define i_t as:

$$i_t = \sigma(X_V^{(t)} * W_i), \tag{4.32}$$

where the convolution layer involves 10 filters each filter is of size 3×3, padding $= 1$ and stride $= 1$. So $W_i \in \mathbb{R}^{3\times3\times13\times10}$. Let the output of input gate be denoted by $\tilde{c}_t$, then we can define $\tilde{c}_t$ as:

$$\tilde{c}_t = \tanh\Big(X_V^{(t)} * W_{\tilde{c}}\Big), \tag{4.33}$$

where the convolution layer involves 10 filters each filter is of size 3×3, padding $= 1$ and stride $= 1$. So $W_{\tilde{c}} \in \mathbb{R}^{3\times3\times13\times10}$. Let the output of input gate be denoted by z_t, then we can define z_t as:

$$z_t = \sigma(X_V^{(t)} * W_z), \tag{4.34}$$

where the convolution layer involves 10 filters each filter is of size 3×3, padding $= 1$ and stride $= 1$. So $W_z \in \mathbb{R}^{3\times3\times13\times10}$. Let the cell state that we obtain at the time step t be denoted as c_t, then we define c_t as:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t. \tag{4.35}$$

Let the hidden state that we obtain at the time step t be denoted as h_t, then we define h_t as:

$$h_t = z_t \odot \tanh(c_t), \tag{4.36}$$

where * denotes convolution and $\odot$ denotes hadamard operation. Let us define a function $fl(\cdot)$, which flattens any input tensor. The final output of the LSTM is given by:

$$\begin{aligned} \mathcal{F}_{V1} &= fl(h_T) = fl(z_T \odot tanh(c_T)), \\ &\Rightarrow \mathcal{F}_{V1} = fl(g_{ConvLSTM}(\mathcal{F}_V)). \end{aligned} \tag{4.37}$$

We observe that $\mathcal{F}_{V1}$ depends on input $\mathcal{F}_V$. So in general in order to observe the gradient equation for the weights that are being learned, we can concatenate W_f, W_i,W_c and W_o to form $W_{ConvLSTM}$. Consequently, we find the expression:

$$\begin{aligned}\frac{\partial \mathcal{F}_{V1}}{\partial W_{ConvLSTM}} &= \frac{\partial \mathcal{F}_{V1}}{\partial c_T}.\frac{\partial c_T}{\partial c_{T-1}}......\frac{\partial c_2}{\partial c_1}.\frac{\partial c_1}{\partial W_{ConvLSTM}},\\ \Rightarrow \frac{\partial \mathcal{F}_{V1}}{\partial W_{ConvLSTM}} &= \frac{\partial \mathcal{F}_{V1}}{\partial c_T}.(\prod_{t=2}^{T}\frac{\partial c_t}{\partial c_{t-1}}).\frac{\partial c_1}{\partial W_{ConvLSTM}}.\end{aligned} \tag{4.38}$$

where the ratio $\frac{\partial c_t}{\partial c_{t-1}}$ can be expressed as follows:

$$\begin{aligned}\frac{\partial c_t}{\partial c_{t-1}} = \sigma'(W_f * X_V^{(t)}) + f_t +\\ \sigma'(W_i * X_V^{(t)}).W_i.z_{t-1} * \tanh'(c_{t-1}) \odot \tilde{c}_t +\\ \sigma'(W_{\tilde{c}} * X_V^{(t)}).W_{\tilde{c}}.z_{t-1} * \tanh'(c_{t-1}) \odot i_t.\end{aligned} \tag{4.39}$$

The output of the convolutional LSTM denoted by $\mathcal{F}_{V1} \in \mathbb{R}^{64\times64\times10\times25}$, which represents both the temporal and salient features of all the frames of the video combined in a single tensor. The feature is then passed through 3D maxpooling layer in order to consider only the salient components of the feature from each channel and also to reduce the number of parameters. The size of the tensor is reduced such that $\mathcal{F}_{V1} \in \mathbb{R}^{31\times31\times4\times25}$. The resultant feature is then flattened and passed through a fully connected layer denoted by FCd1, which learns the mapping: $g_{vid}(\cdot) : \mathcal{F}_{V1} \longrightarrow \mathcal{F}_{V2}$, where $\mathcal{F}_{V1} \in \mathbb{R}^{96100*1}$ and $\mathcal{F}_{V2} \in \mathbb{R}^{1024*1}$

$$\begin{aligned}\mathcal{F}_{V2} &= g_{vid}(\mathcal{F}_{V1}),\\ g_{vid}(\mathcal{F}_{V1}) &= W_{vid}.\mathcal{F}_{V1},\\ \mathcal{F}_{V2} &= W_{vid}.\mathcal{F}_{V1}.\end{aligned} \tag{4.40}$$

where $W_{vid} \in \mathbb{R}^{1024*96100}$ represents the mapping learnt by the fully connected layer FCd1 as shown in the figure 4.2.

4.5.6 Viral-video-prediction Network

We concatenate $\mathcal{F}_{A1}$, $\mathcal{F}_{V2}$ and $\mathcal{F}_{tt}$ to form a feature vector, denoted by $\mathcal{F}$, of size 2560×1, as shown in the figure 4.2. This network consists of fully connected layers which acts as a regressor model to produce a real value which denotes the predicted value of number of views. This network plays a crucial role to generate the number of views from the audio, video, thumbnail and title attributes combined in the vector denoted by $\mathcal{F}$. The networks tries to learn a mapping: $g_{viral}(\cdot) : \mathcal{F} \longrightarrow \mathcal{V}$, where $\mathcal{F} \in \mathrm{I\!R}^{2560*1}$ and $V \in \mathrm{I\!R}$.

$$\begin{aligned} \mathcal{V} &= g_{viral}(\mathcal{F}), \\ g_{viral}(\mathcal{F}) &= W_{viral}.\mathcal{F}, \\ \mathcal{V} &= W_{viral}.\mathcal{F}, \end{aligned} \tag{4.41}$$

where $W_{viral} \in \mathrm{I\!R}^{1\times2560}$ represents the mapping learnt by the viral video prediction network to predict the final view count.

So the function $G(\cdot)$ that the neural network wants to learn is a function of $\mathcal{F}_{th}$, $\mathcal{F}_{ti}$, $\mathcal{F}_{A}$ and $\mathcal{F}_{V}$.

$$\mathcal{G}(\mathcal{F}_{th}, \mathcal{F}_{ti}, \mathcal{F}_{A}, \mathcal{F}_{V}) = g_{viral}(\mathcal{F}).$$

As $\mathcal{F}$ is a concatenation of $\mathcal{F}_{tt}$, $\mathcal{F}_{A1}$ and $\mathcal{F}_{V1}$, then we get:

$$\mathcal{G}(\mathcal{F}_{th}, \mathcal{F}_{ti}, \mathcal{F}_{A}, \mathcal{F}_{V}) = W_{viral}.\left\{ \left[\mathcal{F}_{A1}, \mathcal{F}_{V2}, \mathcal{F}_{tt} \right] \right\},$$

where if a vector $\mathcal{A}$ is a concatenation of vectors $\mathcal{B}$,$\mathcal{C}$ and $\mathcal{D}$ then we can denote $\mathcal{A}$ as $[\mathcal{B}, \mathcal{C}, \mathcal{D}]$.

As we know that $\mathcal{F}_{tt}$ is related to $\mathcal{F}_{thti}$ through equation (4.19), so we get:

$$\mathcal{G}(\mathcal{F}_{th}, \mathcal{F}_{ti}, \mathcal{F}_{A}, \mathcal{F}_{V}) = W_{viral}.\left\{ \left[\mathcal{F}_{A1}, \mathcal{F}_{V2}, W_{ca}.\mathcal{F}_{thti} \right] \right\}.$$

We know that $\mathcal{F}_{thti}$ is a concatenation of $\mathcal{F}_{th1}$ and $\mathcal{F}_{ti1}$, so we get:

$$\mathcal{G}(\mathcal{F}_{th}, \mathcal{F}_{ti}, \mathcal{F}_{A}, \mathcal{F}_{V}) = W_{viral}.\left\{ \left[\mathcal{F}_{A1}, \mathcal{F}_{V2}, W_{ca}.[\mathcal{F}_{th1}, \mathcal{F}_{ti1}] \right] \right\}.$$

We know that $\mathcal{F}_{th1}$ is related to $\mathcal{F}_{th}$ through equation (4.13) and $\mathcal{F}_{ti1}$ is related to $\mathcal{F}_{ti}$ through equation (4.17), then we get:

$$\mathcal{G}(\mathcal{F}_{th}, \mathcal{F}_{ti}, \mathcal{F}_{A}, \mathcal{F}_{V}) = W_{viral}.\left\{ \left[\mathcal{F}_{A1}, \mathcal{F}_{V2}, W_{ca}.[W_{th}.\mathcal{F}_{th}, W_{ti}.\mathcal{F}_{ti}] \right] \right\}.$$

We know that $\mathcal{F}_{A1}$ is related to $\mathcal{F}_{A}$ through equation (4.27) and $\mathcal{F}_{V2}$ is related to $\mathcal{F}_{V1}$ through equation (4.40), then we get:

$$\mathcal{G}(\mathcal{F}_{th}, \mathcal{F}_{ti}, \mathcal{F}_{A}, \mathcal{F}_{V}) = W_{viral}.\left\{ \left[g_{LSTM}(\mathcal{F}_{A}), W_{vid}.\mathcal{F}_{V1}, W_{ca}.[W_{th}.\mathcal{F}_{th}, W_{ti}.\mathcal{F}_{ti}] \right] \right\}.$$

We know that $\mathcal{F}_{V1}$ is related to $\mathcal{F}_V$ through equation (4.37), then we get:

$$\mathcal{G}(\mathcal{F}_{th}, \mathcal{F}_{ti}, \mathcal{F}_A, \mathcal{F}_V) = W_{viral}.\Bigg\{\Bigg[g_{LSTM}(\mathcal{F}_A), W_{vid}.(fl(g_{ConvLSTM}(\mathcal{F}_V))), W_{ca}.[W_{th}.\mathcal{F}_{th}, W_{ti}.\mathcal{F}_{ti}]\Bigg]\Bigg\}.$$

So the function $G(\cdot)$ learnt is given as follows:

$$\mathcal{G}(\mathcal{F}_{th}, \mathcal{F}_{ti}, \mathcal{F}_A, \mathcal{F}_V) = W_{viral}.\Bigg\{\Bigg[g_{LSTM}(\mathcal{F}_A), W_{vid}.(fl(g_{ConvLSTM}(\mathcal{F}_V))), W_{ca}.[W_{th}.\mathcal{F}_{th}, W_{ti}.\mathcal{F}_{ti}]\Bigg]\Bigg\}.$$

Since the function $G(\cdot)$ outputs the predicted number of views, so we get:

$$V_{pred} = W_{viral}.\Bigg\{\Bigg[g_{LSTM}(\mathcal{F}_A), W_{vid}.(g_{ConvLSTM}(\mathcal{F}_V)), W_{ca}.[W_{th}.\mathcal{F}_{th}, W_{ti}.\mathcal{F}_{ti}]\Bigg]\Bigg\}. \quad (4.42)$$

4.6 Loss Function

We have used Mean Squared Error as our loss function to train our model. It is as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}(V_i^{predicted} - V_i^{actual})^2, \quad (4.43)$$

where V_i^{actual} is the actual number of views i^{th} video gets, $V_i^{predicted}$ is the number of views i^{th} video can get predicted by the model and N is the number of training observations.

4.7 Implementation Details

In this section, we focus on the implementation details of different sub-networks of the network which extract different feature vectors representing different aspects of the given video.

4.7.1 Extracting Thumbnail Feature Vector $\mathcal{F}_{th}$

We have used the AlexNet [19] model, pretrained with ImageNet dataset [41] alongwith two fully connected layers denoted by FCa1 and FCa2 consisting of 10000 and 1024 neurons respectively, as shown in the figure 4.2. The thumbnail image is first resized to 227 x 227 x 3 before passing to the AlexNet. We have frozen and kept only the first five layers of the AlexNet network. The output of this pretrained model is flattened and passed through a dropout layer [28] with probability of 0.4, in order to prevent overfitting, and we obtain a feature vector of size 43264×1, denoted by $\mathcal{F}_{th}$ as shown in figure 4.2. The vector $\mathcal{F}_{th}$ is then passed through two fully connected layers denoted by FCa1 and FCa2 to get the feature vector $\mathcal{F}_{th1}$.

4.7.2 Extracting Title Feature Vector $\mathcal{F}_{ti}$

We implement the Sentiment Intensity Analyzer [40] model to obtain the title feature vector $\mathcal{F}_{ti}$ from title of the video. The model is pretrained on irony and sarcasm based tweets alongwith punctuations and special symbols, to capture the strong emotion of the Twitter users for sentiment analysis task. In our case we utilize the model to extract sentiments from the title of the video. The input to the Sentiment Intensity Analyzer model is the title of the video in the form of one hot encoded vector and the output is the vector of length three depicting the probability score of a title to be positive, negative and neutral sentiment, denoted by $\mathcal{F}_{ti}$ as shown in the figure 4.2. We had taken these sentiments into account as they reflect the affinity of a viewer to click the video. This vector $\mathcal{F}_{ti}$ is then passed through two consecutive fully connected layers denoted by FCb1 and FCb2, consisting of 512 and 1024 neurons respectively, to get the feature vector $\mathcal{F}_{ti1}$ of size 1024×1, as shown in the figure 4.2.

4.7.3 Extracting Audio Feature Vector $\mathcal{F}_A$

It is usually found that audio in form of music is more memorable to humans compared to human speech[]. In order to capitalize on the above fact, we have extracted the mel frequency cepstral coefficients (MFCC) and the delta mel frequency cepstral cofficients (D-MFCC) using the librosa library [42] from the audio. The audio waveform is sampled at 22000 Hz. We have tried to conduct our experiment with different number of frames and hop length and obtained an optimal performance by sampling 1024 frames with hop length of 512 samples from the above samples. We take the fast fourier transform for each frame to create a spectrum called the Short Time Fourier Transform (STFT) [43, 44, 45]. We pass each of these frame through mel-frequency filter banks and perform cepstral analysis to get a vector, corresponding to each frame, consisting of mel frequency cepstral coefficients. We have taken mel-frequency filter banks because we want our features match more closely with what humans hear. Usually for this experiment we have taken only the first 24 cepstral coefficients because these 24 coefficients are sufficient to capture the spectral structure of the waveform as perceived by the human ear. This results in a 24 dimensional feature vector corresponding to each frame denoted by $\mathcal{F}_A \in \mathbb{R}^{1024 \times 24}$ We feed each frame at a time to a Long Short Term Memory (LSTM) network [22]. The LSTM network accepts an input vector of size 24×1, the hidden vector of the LSTM is of size 512×1. The output vector from the LSTM is a 512 dimensional vector denoted by $\mathcal{F}_{A1}$ as shown in figure 4.2.

4.7.4 Extracting Video Feature Vector $\mathcal{F}_V$

To extract the video features, we have conducted our experiment varying the number of frames to be extracted from the video. With respect to optimal performance, we found that it would be sufficient to extract only equally spaced 25 i-frames from the video. In order to reflect the salient features present in each frame of the video, we found it sufficient to resize the frames to $64\times64\times3$ denoted by $\mathcal{F}_V$. In order to capture both the temporal and salient aspect of each frame of the video, we decided to use a Convolution-LSTM model [46]. The size of the convolution filters is experimentally set to 3×3 with suitable zero-padding and stride is set to 1. The output feature tensor, denoted by $\mathcal{F}_{V1}$, is obtained from the last time-step of the Convolution-LSTM model of size $64 \times 64 \times 10 \times 25$, which represents both the temporal and salient features of all the frames of the video combined in a single tensor. The feature tensor is flattened and then passed through a fully connected layer denoted by FCd1, consisting of 512 neurons, to give a feature vector denoted by $\mathcal{F}_{V2}$, of size 512×1, as shown in the figure 4.2.

4.7.5 Building the click affinity network

The purpose of this network is to get a combined representation of thumbnail and title associated with the video which reflects the affinity of the user to click that particular video. We concatenate the thumbnail feature vector $\mathcal{F}_{th1}$ and the title feature vector $\mathcal{F}_{ti1}$, as shown in the figure 4.2 to produce a feature vector of size 2048×1. The click affinity network transforms this vector to a lower dimensional space through a fully connected layer, denoted by $\mathcal{F}_{tt}$, as shown in the figure 4.2. In order to avoid overfitting, we applied dropout [28] at the output with a probability of 0.4.

4.7.6 Building the viral video prediction network

The concatenation of $\mathcal{F}_{A1}$ and $\mathcal{F}_{V2}$ gives a combined a representation of audio visual aspect of the video. On further concatenating the vector $\mathcal{F}_{tt}$ gives a combined representation of audio, visual, aesthetic and sentimental aspect of the video which are the key attributes for a user to view the video. The view prediction network tries to learn the hidden representation that maps these attributes of a video with the view the video receives. We concatenate $\mathcal{F}_{A1}$, $\mathcal{F}_{V2}$ and $\mathcal{F}_{tt}$ to form a feature vector, denoted by $\mathcal{F}$, of size 2048×1, as shown in the figure 4.2. $\mathcal{F}$ is transformed to a lower dimensional space through a series of fully connected layers, denoted by FC1 and FC2, consisting of 1024 and 512 neurons respectively, as shown in the figure 4.2. The final layer in the network outputs a regression value, which is the predicted number of views of the video. To avoid overfitting of the network, We add a dropout layer in between the fully connected layers FC1 and FC2 with a dropout probability of 0.4.

4.7.7 Training details

The part of the network comprising of the fully connected layers labelled FCa1, FCa2, FCb1, FCb2, FCab1, FCd1, FC1, FC2, FC3 and both the LSTM and Convolutional LSTM models, as shown in the figure 4.2, are trained with Adam optimizer [29] for 150 epochs on Nvidia TITAN RTX [47]. We have set a learning rate of 0.00001. We have taken a batch size of 16. We have used Mean Square Error (MSE) loss between the actual view and the predicted view of the videos, averaged over all batches for each epoch, while training the network.

4.8 Summary

In this chapter we came across the factors that makes a video viral and tried to model the factors accordingly. Based on these factors, we tried to mathematically model the number of views of a video. We then discussed the utility of the four subjective video attributes namely, the thumbnail, the title, the audio associated with the video and the video itself. Based on the features of these subjective attributes, we designed our proposed model, explaining the sub-networks of our model like the click-affinity network and the viral-video-prediction network. Based on the results of modelling the number of views of a video, we tried to give some mathematical exposition of the function $G(\cdot)$ that the network tries to learn. Finally, we discussed on the implementation details of the network.

Chapter 5

Results

This chapter elaborates the results obtained from this research work. The dataset contains a collection of daily trending YouTube videos with attributes like the video title, video, audio, thumbnail image and view count of the video over a selected interval. The results obtained on the dataset through the experiment are discussed in this section.

5.1 Training results

We have taken 800 videos for training and 200 videos for testing phase. The five-fold cross validation result on 800 videos with 600 videos as training set and 200 videos as validation set has the following results as shown in the figure 5.1. The interpretation of the results with respect to the original model is as follows:

- Training Loss: 5.23% which means the model on average predicted the number of views at a deviation of 5.23% from the ground truth number of views
- Validation Loss: 5.82% which means the model while training on average predicted the number of views at a deviation of 5.82% from the ground truth number of views of unseen videos.
- Testing Accuracy: 83.38% which is derived from Mean Absolute Percentage Error (MAPE) explained in details in section 5.2 involving equations (5.1) and (5.2).

5.2 Evaluation metrices

The loss used to train our model is Mean Squared Error. The evaluation metric on which we are evaluating our model is Mean Absolute Percentage Error (MAPE) [48]

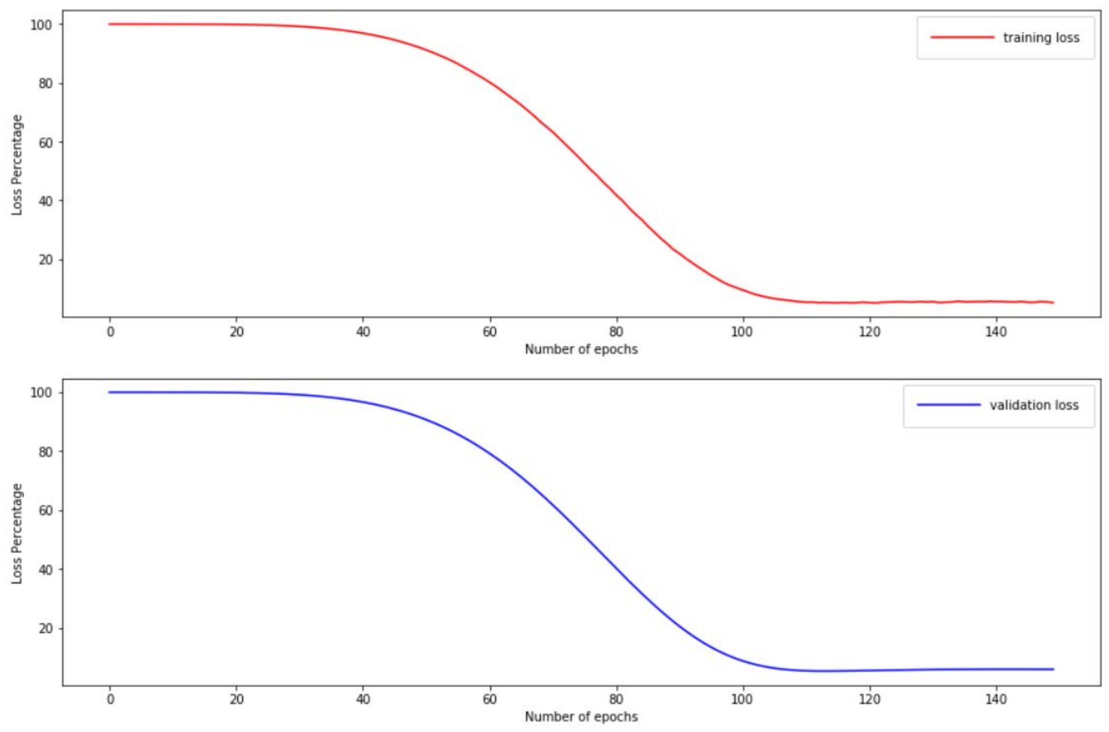


Figure 5.1: The training and validation loss for 150 epochs

Model	Training Loss	Validation Loss	Testing Accuracy
Original Model	5.23	5.82	83.38
Ablation study of thumbnail	90.25	89.83	9.95
Ablation study of title	10.35	9.74	74.17
Ablation study of Audio	27.85	26.86	49.98
Ablation study of video	29.66	28.63	48.18
Ablation study of thumbnail-title	90.26	89.82	10.04
Ablation study of audio-video	29.35	28.06	48.76

Table 5.1: Table showing the training, validation losses and testing accuracy of all the possible ablation studies conducted.

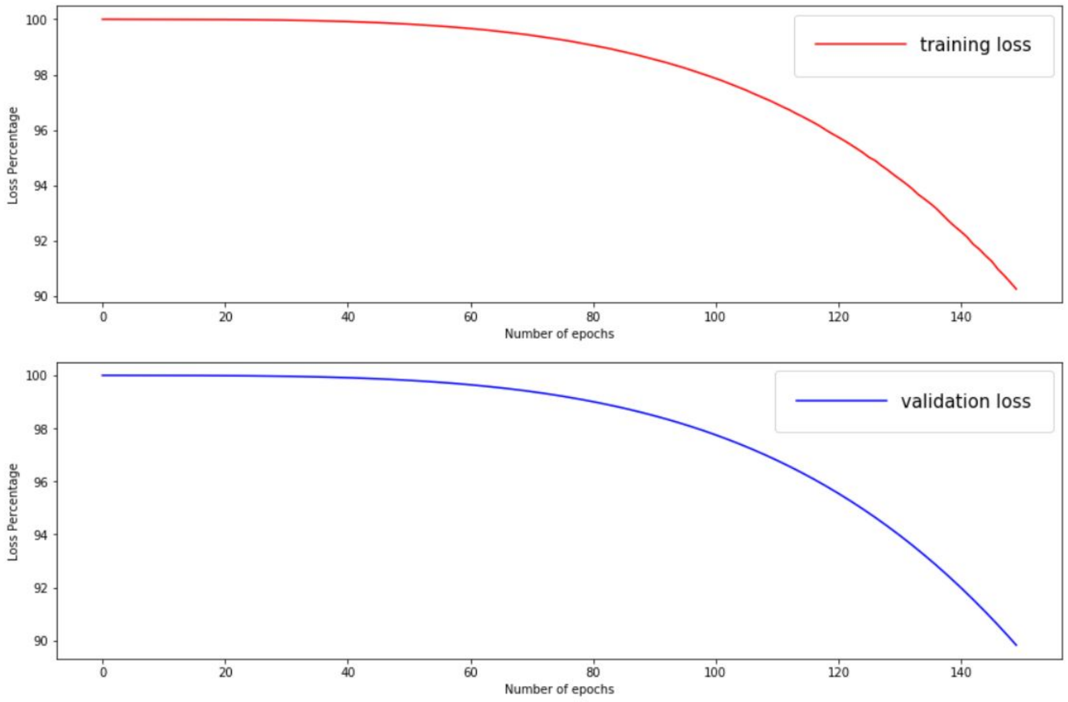


Figure 5.2: Plot of training and validation loss of ablation thumbnail model.

given by the equation:

$$M = \frac{1}{N}\sum_{i=1}^{N}\frac{|V_i^{actual} - V_i^{predicted}|}{V_i^{actual}} \times 100, \tag{5.1}$$

where V_i^{actual} is the actual number of views i^{th} video gets, $V_i^{predicted}$ is the number of views i^{th} video can get predicted by the model and N is the total number of test instances or datapoints. So, according to equation (5.1), the M for the test dataset is 16.61%. Now we define our test accuracy as:

$$\text{Test accuracy} = 100 - M, \tag{5.2}$$

then, according to equation (5.2), we can state our test accuracy to be 83.38%. So the model predicts an unseen video with a deviation of 16.61% on average from its actual number of views.

5.3 Ablation study

To show the effectiveness of all four components of model, we have performed the ablation study. In the ablation study, we have removed each particular component

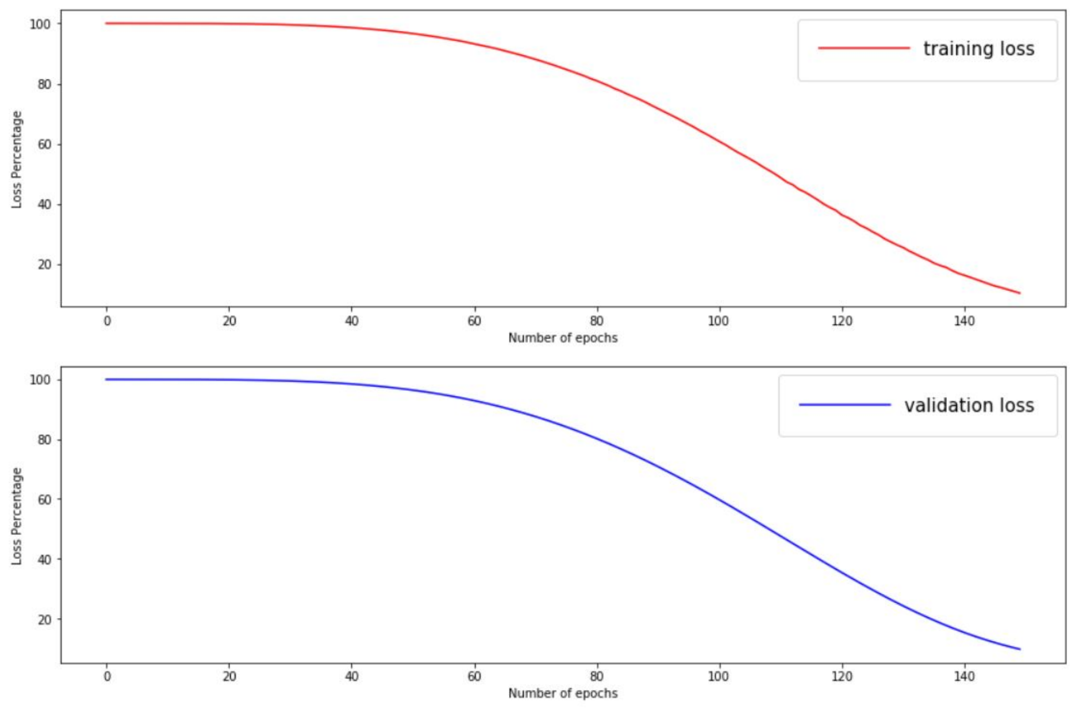


Figure 5.3: Plot of training and validation loss of ablation title model.

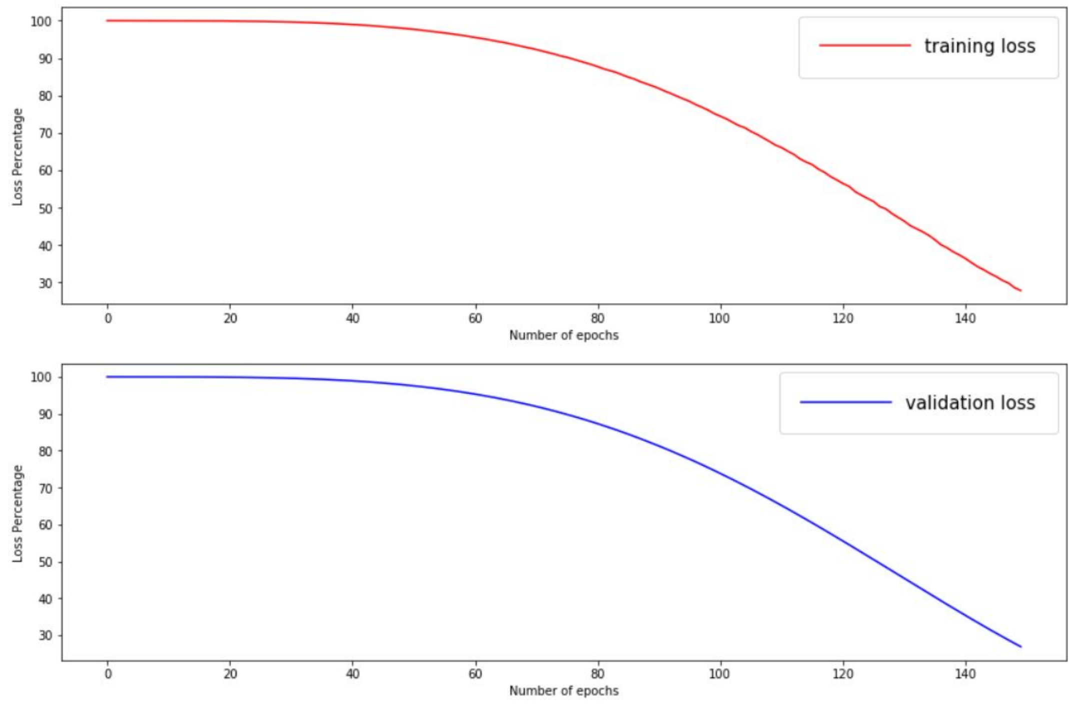
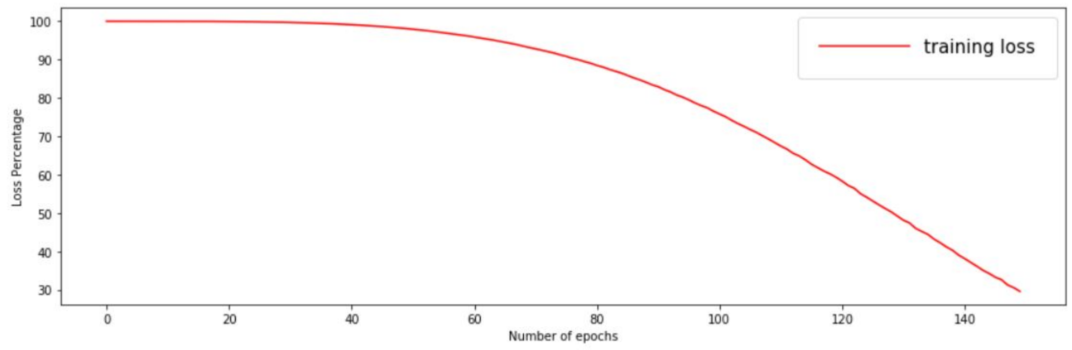


Figure 5.4: Plot of training and validation loss of ablation audio model.



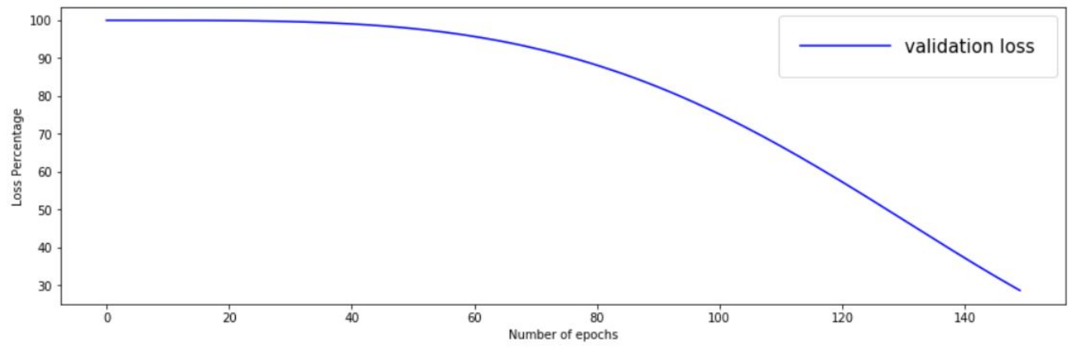
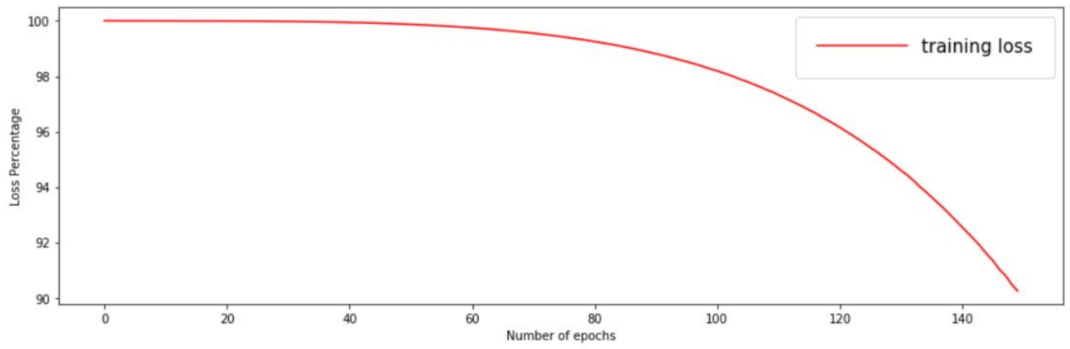


Figure 5.5: Plot of training and validation loss of ablation video model.



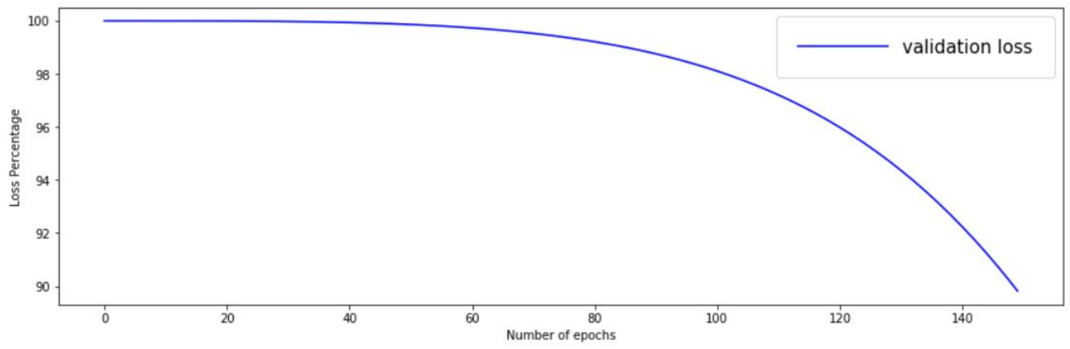


Figure 5.6: Plot of training and validation loss of ablation thumbnail-title model.

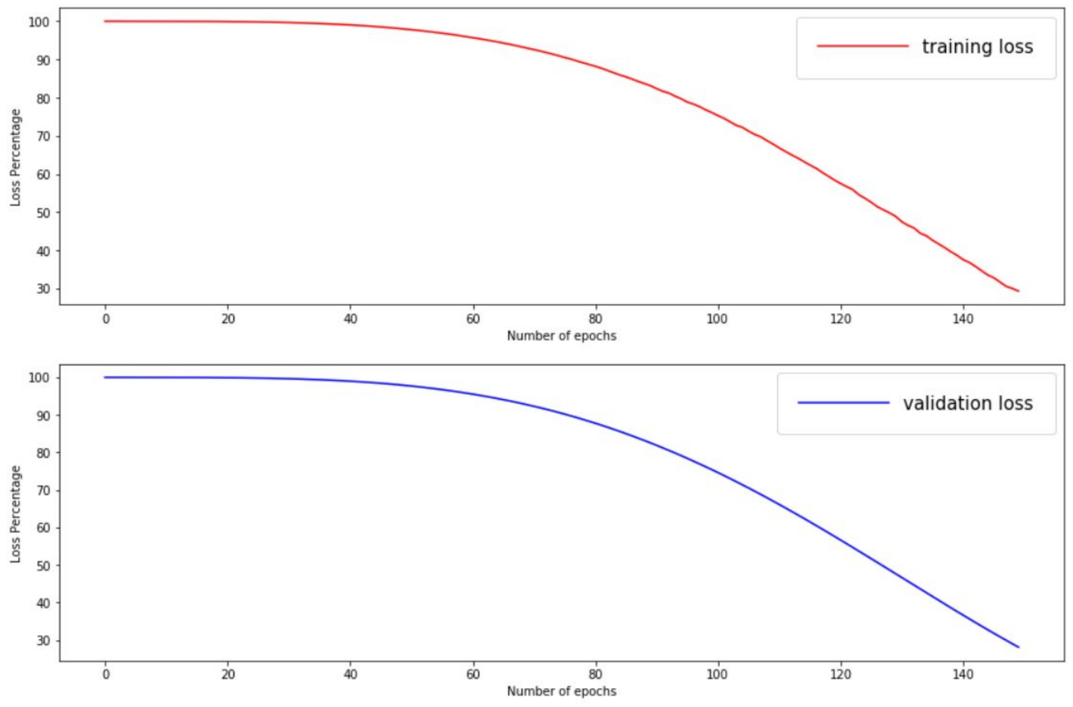


Figure 5.7: Plot of training and validation loss of ablation audio video model.

and their combinations to study their corresponding importance in the model. To compare their performance, we have trained the models, obtained after removing component(s), on the same dataset. In order to understand the importance of individual components, we conducted the following experiments:

- Ablation study of thumbnail: In this study, we have removed Pre-trained AlexNet, FCa1, FCa2 and FCab1 from the original network shown in figure 4.2. The F_{ti1} feature vector is fed directly to the viral video prediction network. The training of the model is shown in the figure 5.2.
- Ablation study of title: In this study, we have removed Sentiment Intensity Analyzer, FCb1, FCb2 and FCab1 from the original network shown in figure 4.2. The F_{th1} feature vector is fed directly to the viral video prediction network. The training of the model is shown in the figure 5.3.
- Ablation study of audio: In this study, we have removed Long Short Term Memory component from the original network shown in figure 4.2. So the final feature vector feeding the viral video prediction network is a concatenation of $\mathcal{F}_{tt}$ and $\mathcal{F}_{V2}$. The training of the model is shown in the figure 5.4.
- Ablation study of video: In this model, we have removed Convolutional LSTM and FCd1 from the original network shown in figure 4.2. So the final feature vector feeding the viral video prediction network is a concatenation of $\mathcal{F}_{tt}$ and $\mathcal{F}_{A1}$. The training of the model is shown in the figure 5.5.

In order to understand the importance of components in combination, we conducted the following experiments:

- Ablation study of thumbnail-title: In order to understand the importance of thumbnail image and title that attracts a user to click a particular video before being viewed, thereby affecting the view count of the video, we have removed thumbnail and title attribute branches from the model. In this model, we have removed Pre-trained AlexNet,Sentiment Intensity Analyzer, FCa1, FCa2, FCb1, FCb2 and FCab1 from the original network shown in figure 4.2. So the final feature vector feeding the viral video prediction network is a concatenation of $\mathcal{F}_{A1}$ and $\mathcal{F}_{V2}$. The training of the model is shown in the figure 5.6.
- Ablation study of audio-video: In order to understand the importance of the audio visual appeal that plays a role in the number of views a video receives, we have removed the audio and video attribute branches from the model. In this model, we have removed Long Short Term Memory component, Convolutional LSTM and FCd1 from the original network shown in figure 4.2. So the final feature vector feeding the viral video prediction network is a concatenation of $\mathcal{F}_{tt}$. The training of the model is shown in the figure 5.7.

The results of these ablation studies are shown in the Table 5.1.

5.4 Some of the sample examples predicted by our model

Since we are discussing some samples that have been predicted by our model in this section, we would illustrate some good predicted instances as well as some bad predicted ones.

5.4.1 Good Predicted Instances

We have shown in this section 4 samples which have been predicted by the model accurately. The details of these samples are summarised in the table on the following page.

5.4.2 Bad Predicted Instances

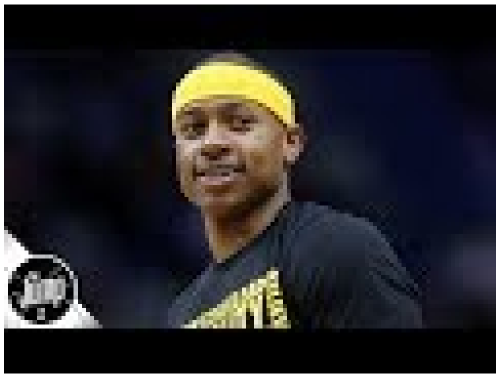
We have shown in this section 4 samples which have been predicted by the model not so accurately. The details of these samples are summarised in the table on page 61.

Instance	**Title**	**URL**	**Thumbnail Image**	**Actual Views**	**Predicted Views**
1	EVERY ELEVATOR EVER	https://www.youtube.com/watch?v=6Nr_xHolP1Q	Refer figure 5.8a	1339479	1340380
2	I Dribbled A Basketball For An Entire Day	https://www.youtube.com/watch?v=F8BdzL8OltM	Refer figure 5.8b	1162551	1163957
3	Tracy McGrady on Isaiah Thomas to the Lakers: He can't bring an ego with him — The Jump — ESPN	https://www.youtube.com/watch?v=FI3zMdqq4vE	Refer figure 5.8c	1051495	1054440
4	CELEBRATING 1 MILLION SUBSCRIBERS AT FALLON!!	https://www.youtube.com/watch?v=nsjPEcRq_0c	Refer figure 5.8d	1090412	1084535

Table 5.2: Details of Good Predicted Instances

(a) Thumbnail image of Instance 1 [49]

(b) Thumbnail image of Instance 2 [50]



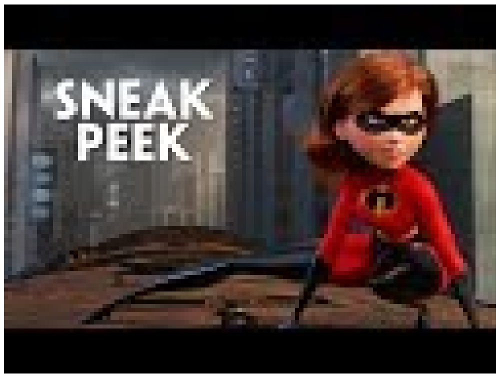
(c) Thumbnail image of Instance 3 [51]

(d) Thumbnail image of Instance 4 [52]

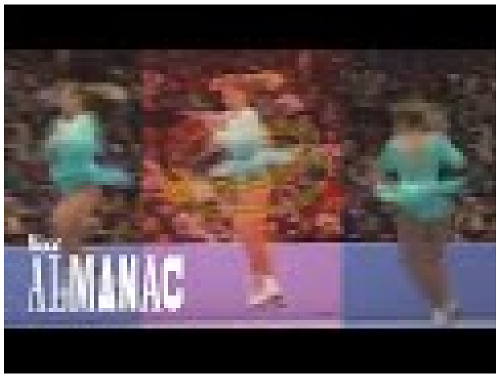
Figure 5.8: Thumbnail images of good predicted instances

(a) Thumbnail image of Instance 1 [53]

(b) Thumbnail image of Instance 2 [54]



(c) Thumbnail image of Instance 3 [55]



(d) Thumbnail image of Instance 4 [56]

Figure 5.9: Thumbnail images of bad predicted instances

Instance	**Title**	**URL**	**Thumbnail Image**	**Actual Views**	**Predicted Views**
1	NF - Let You Down	https://www.youtube.com/watch?v=fbHbTBP_u7U	Refer figure 5.9a on the facing page	1774018	914525
2	Kygo - Stranger Things ft. OneRepublic	https://www.youtube.com/watch?v=3ChgRbqGi-E	Refer figure 5.9b on the preceding page	1770318	886889
3	Incredibles 2 - Olympics Sneak Peek	https://www.youtube.com/watch?v=YBpdL9hSac4	Refer figure 5.9c on the facing page	1948452	1058699
4	Why the triple axel is such a big deal	https://www.youtube.com/watch?v=7rOQv_6L9fQ	Refer figure 5.9d on the preceding page	1960257	1009112

Table 5.3: Details of Bad Predicted Instances

5.5 Comparison with the Existing Method

The existing method as discussed in section 1.3.1, we have trained the model on our dataset and tried to compare the results between the two models. Based on the evaluation metric that we have discussed in section 5.2,in case of the model described in existing method, we received a $MAPE$ from the equation 5.1 as 20.11%. This implies that we obtain our test accuracy in case of this model according to equation 5.2 as 79.89%.

While in case of our method we obtain a $MAPE$ according to equation 5.1 as 16.61% and test accuracy based on the equation 5.2 as 83.38%.

The graph shown in the figure 5.10 displays a comparison between the two models with respect to the actual number of views and predicted ones from the respective models.

We obtain the graph as follows:

- We grouped the videos in groups of 10. (As there are 208 video instances, so we would get 20 groups of 10 video instances each and 21st group of 8 video instances). So there are 21 groups in all.
- We then take the average of actual views of the 10 videos in each group . Similarly we take average of "prediction from our model" of the 10 videos in

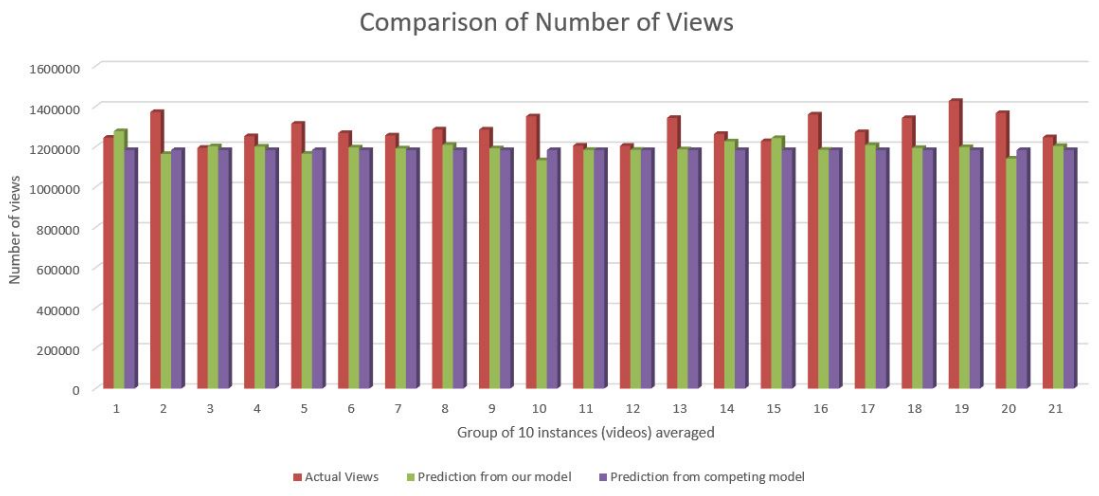


Figure 5.10: The comparison between the two models based on the number of views predicted by the models against the actual number of views

each group and also the same for ”prediction from the competing model”.

- Therefore, we get 21 values for the actual views representing each of the group. Similarly, we get 21 values for the ”prediction from our model” representing each of the group and also the same for ”prediction from the competing model”.
- From these values, we construct the bar graph shown in the figure 5.10.

In the graph 5.10, since there are 208 test instances (videos), we have grouped them in number of 10, so each bar represents a group of 10 instances (videos) averaged.

5.5.1 Time Complexity

Let us analyse the time complexity of our model and the competing model.

Our model

Let the input to the model be of size n. It is to be noted that the model architecture is fixed. So, if we are focusing on the time complexity, let us analyze the algorithm 1 the model follows.

So observing the algorithm 1, the training process can be divided into three steps:

- *single_pass_n_update* function call for each batch of input
- Repeat the above step for different batches of input

Algorithm 1 Model Training

```
Input: $Train\_dataset$ - Thumbnail, Title, Audio, Video, $model$, $batch\_size$, $epochs$
Output: $model$ - Trained model
1: $batches \leftarrow Train\_dataset/batch\_size$
2: $i \leftarrow 0$
3: while $i < epochs$ do
4:     for $batch$ in $batches$ do
5:         $model \leftarrow single\_pass\_n\_update(model, batch)$
6:     end for
7:     $i \leftarrow i + 1$
8: end while
9: return $model$
```

- Repeat the above step for a given number of epochs

The *single_pass_n_update* function which processes each batch of the input contains three important procedures as follows:

- Forward Pass
- Loss Calculation
- Backward pass and parameter update

Both the forward pass and the backward pass have a time complexity of $\mathcal{O}(n_{h1})$ [57] where n_{h1} is the number of hidden nodes in our model. The loss calculation is $\mathcal{O}(1)$ time complexity. Therefore, the function *single_pass_n_update* is $\mathcal{O}(n_{h1})$ time complexity.

Suppose there are N training examples and we enter the value *epochs* and *batch_size* as hyperparameters, considering the loop of *batches* i.e. line 4 of algorithm 1, we get that the time complexity as $\mathcal{O}(N/batch\_size)$. Considering the loop of epoch i.e. line 3 of algorithm 1, we get the time complexity as $\mathcal{O}(epochs)$. Therefore, the time complexity of the training algorithm is $\mathcal{O}(n_{h1} \times epochs \times N/batch\_size)$.

While testing since we use only the *single_pass_n_update* function, so we get the time complexity as $\mathcal{O}(n_{h1})$ [57], where n_{h1} is the number of hidden nodes in our model.

Competing Model

The model as discussed in section 1.3.1 consists of two stages. Let us consider that there are N training samples, so in the first stage as it involves training a CNN, so it follows the algorithm 1. Therefore, the time complexity of the first stage is

$\mathcal{O}(n_{h2} \times epochs \times N/batch\_size)$, where n_{h2} is the number of hidden nodes in the competing model.

In the second stage, the model involves training the Support Vector Regressor (SVR). Considering the fact that N is the number of training example, the time complexity of training SVR is $\mathcal{O}(N^3)$ [57].

So, the time complexity of the competing model is $\mathcal{O}(n_{h2} \times epochs \times N/batch\_size + N^3)$ during training.

In the testing phase, the time complexity of the first phase involving CNN is $\mathcal{O}(n_{h2})$ and that of SVR is $\mathcal{O}(n_{SV})$ [57], where n_{SV} is the number of support vectors for the SVR. So, the overall time complexity in testing for the competing model is $\mathcal{O}(n_{h2} + n_{SV})$.

Now, since n_{h1} is much greater than n_{h2}, so the time complexity of training phase for our model is much higher than the competing model. This agrees with the fact that the execution time for training is 60 minutes for 150 epochs in case of our model. While the execution time for training is only 10 mins for 150 epochs in case of competing model.

In case of testing, the execution time for our model is only 1.3 seconds while that of the competing model is 0.4 seconds.

These execution times are recorded while being executed on Nvidia TITAN RTX GPU [47].

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The model that we have designed in our experiment have been able to predict views of the video using a deep neural network based analysis of subjective video attributes namely the thumbnail image, the title caption, the audio waveform and the video itself. With an elaborate branching structure of the architecture, the model can retain the spatial features of the thumbnail using a pretrained CNN architecture, Alexnet; the sentimental features of the title caption using the pretrained Sentiment Intensity Analyzer; the temporal features of the audio waveform using LSTM and both the temporal and spatial features of the video using the Convolutional LSTM at each respective branches of the network. Based on the spatial features of the thumbnail and the sentimental features of the title the model generates a feature that depicts the affinity of the user to click a video and together with this feature alongwith the temporal feature of audio waveform and spatio-temporal features of the video, the model predicts the number of views of the video using a regressor network called the viral video prediction network. The videos prevailing in our dataset has come from a varied genres involving Film and animation, Autos and vehicles, music, pets and animals, sports, short movies,travels and events,gaming, video blogging, people and blogs, comedy, entertainment,news and politics,education, science and technology,classics, drama, family and many more. So our videos coming from such a varied genre alongwith a wide range of view count ranging from 10,00,000 to 20,00,000; our model is able to generalize the view count for these videos which is one of the main advantage of our model. The purpose of solving our problem is the one associated with online platforms so one of the most important aspect of online services is low latency. We have developed our model considering videos from different genres and a view count ranging from 10,00,000 to 20,00,000 implying it would require 150 epochs to converge. This means a lot of time consumption even operating under high end GPU such as Titan RTX [47]. We could try to solve the problem by employing a different approach towards optimizing our loss function. We had used Adam optimization which is a gradient based optimization technique, that involves a lot of time to reach the optimal values of the parameters. Instead we could use a gradient free approach to attack the same optimization with the sheer interest to observe for faster convergence to better optima. However since our model here tries to measure the response of the success of the video published in an online platform, it might be useful in the following aspects:

- Over the past 5 years YouTube has paid out more than $5 billion to YouTube content creators according to the medium article [58]. With a growing demand for online video content makers to capture the millennial audience, getting people to watch your videos on YouTube is becoming increasingly lucrative. So the model can help influencers predict the number of views for their videos.
- Since the model generates a feature vector revealing the attractiveness of the thumbnail, title, audio and video attributes, we can derive a score through a regressor model for each of these attributes to recommend about the quality of attributes to the maker of the video with an objective to increase the view count of the video.

6.2 Future Work

As discussed in section 6.1, the purpose of solving our problem is the one associated with online platforms so one of the most important aspect of online services is low latency. We have developed our model considering videos from different genres and a view count ranging from 10,00,000 to 20,00,000 implying it would require 150 epochs to converge. This means a lot of time consumption even operating under high end GPU such as Titan RTX. We could try to solve the problem by employing a different approach towards optimizing our loss function. We had used Adam optimization which is a gradient based optimization technique, that involves a lot of time to reach the optimal values of the parameters. Instead we could use a gradient free approach to attack the same optimization with the sheer interest to observe for faster convergence to better optima. So employing these needs a different kind of implementation where both the forward and back-propagation mathematics needs to be derived. All training functions has to be developed as the approach is a new one.

Bibliography

[1] Wikipedia, “Viral video,” https://en.wikipedia.org/wiki/Viral_video, [Online; accessed 13-March-2019].

[2] T. West, “Going viral: Factors that lead videos to become internet phenomena,” *The Elon Journal of Undergraduate Research in Communications*, vol. 2, no. 1, pp. 76–84, 2011.

[3] L. Jiang, Y. Miao, Y. Yang, Z. Lan, and A. G. Hauptmann, “Viral video style: A closer look at viral videos on youtube,” in *Proceedings of International Conference on Multimedia Retrieval*, 2014, pp. 193–200.

[4] D. A. Shamma, J. Yew, L. Kennedy, and E. F. Churchill, “Viral actions: Predicting video view counts using synchronous sharing behaviors,” in *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[5] Q. Kong, M.-A. Rizoiu, S. Wu, and L. Xie, “Will this video go viral: Explaining and predicting the popularity of youtube videos,” in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 175–178.

[6] J. Berger and K. L. Milkman, “Emotion and virality: what makes online content go viral?” *GfK Marketing Intelligence Review*, vol. 5, no. 1, pp. 18–23, 2013.

[7] A. Deza and D. Parikh, “Understanding image virality,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1818–1826.

[8] T. Broxton, Y. Interian, J. Vaver, and M. Wattenhofer, “Catching a viral video,” *Journal of Intelligent Information Systems*, vol. 40, no. 2, pp. 241–259, 2013.

[9] D. Vallet, S. Berkovsky, S. Ardon, A. Mahanti, and M. A. Kafaar, “Characterizing and predicting viral-and-popular video content,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1591–1600.

[10] H. Pinto, J. M. Almeida, and M. A. Gonçalves, "Using early view patterns to predict the popularity of youtube videos," in *Proceedings of the sixth ACM international conference on Web search and data mining*, 2013, pp. 365–374.

[11] A. Dubey and S. Agarwal, "Modeling image virality with pairwise spatial transformer networks," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 663–671.

[12] W. Stokowiec, T. Trzciński, K. Wołk, K. Marasek, and P. Rokita, "Shallow reading with deep learning: Predicting popularity of online content using only its title," in *International Symposium on Methodologies for Intelligent Systems*. Springer, 2017, pp. 136–145.

[13] J. Chen, X. Song, L. Nie, X. Wang, H. Zhang, and T.-S. Chua, "Micro tells macro: predicting the popularity of micro-videos via a transductive model," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 898–907.

[14] M. Meghawat, S. Yadav, D. Mahata, Y. Yin, R. R. Shah, and R. Zimmermann, "A multimodal approach to predict social media popularity," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2018, pp. 190–195.

[15] W. Zhang, W. Wang, J. Wang, and H. Zha, "User-guided hierarchical attention network for multi-modal social image popularity prediction," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1277–1286.

[16] T. Trzciński and P. Rokita, "Predicting popularity of online videos using support vector regression," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2561–2570, 2017.

[17] A. Bielski and T. Trzcinski, "Understanding multimodal popularity prediction of social media videos with self-attention," *IEEE Access*, vol. 6, pp. 74 277–74 287, 2018.

[18] A. Chhabra, "Github," https://github.com/aarish17212/YouTube-Views-Predictor/blob/master/Model_II.py, [Online; accessed 05-July-2020].

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[20] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[21] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth international AAAI conference on weblogs and social media*, 2014.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, 1997, pp. 155–161.

[25] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[26] Wikipedia, "Mean absolute percentage error," https://en.wikipedia.org/wiki/Mean_absolute_percentage_error, [Online; accessed 09-July-2020].

[27] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées."

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] "Trending youtube video statistics," https://www.kaggle.com/datasnaek/youtube-new, [Online; accessed 05-July-2020].

[31] "Music is more memorable than visuals in marketing," https://www.bandt.com.au/music-memorable-visuals-marketing-new-study-suggests/, [Online; accessed 02-July-2020].

[32] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[33] YouTube, "Red sparrow official trailer," https://i.ytimg.com/vi/PmUL6wMpMWw/default.jpgr, [Online; accessed 09-July-2020].

[34] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[35] M. F. Porter *et al.*, “An algorithm for suffix stripping.” *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[36] E. Jones, T. Oliphant, P. Peterson *et al.*, “SciPy: Open source scientific tools for Python,” 2001–. [Online]. Available: http://www.scipy.org/

[37] H. Fayek, “Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what’s in-between,” https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html, [Online; accessed 05-July-2020].

[38] “Ffmpeg,” http://ffmpeg.org/, [Online; accessed 03-July-2020].

[39] YouTube, “Red sparrow—official trailer—20th century fox,” https://www.youtube.com/watch?v=PmUL6wMpMWw, [Online; accessed 09-July-2020].

[40] P. Tungthamthiti, E. Santus, H. Xu, C.-R. Huang, and K. Shirai, “Sentiment analyzer with rich features for ironic and sarcastic tweets,” in *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, 2015, pp. 178–187.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition.* IEEE, 2009, pp. 248–255.

[42] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in python,” in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[43] J. Allen, “Applications of the short time fourier transform to speech processing and spectral analysis,” in *ICASSP’82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7. IEEE, 1982, pp. 1012–1015.

[44] ——, “Short term spectral analysis, synthesis, and modification by discrete fourier transform,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 3, pp. 235–238, 1977.

[45] J. B. Allen and L. R. Rabiner, “A unified approach to short-time fourier analysis and synthesis,” *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1558–1564, 1977.

[46] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2015, pp. 4580–4584.

[47] NVIDIA, “Nvidia titan rtx is here,” https://www.nvidia.com/en-in/deep-learning-ai/products/titan-rtx/, [Online; accessed 09-July-2020].

[48] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi, “Mean absolute percentage error for regression models,” *Neurocomputing*, vol. 192, pp. 38–48, 2016.

[49] YouTube, “Every elevator ever,” https://www.youtube.com/watch?v=6Nr_xHolP1Q, [Online; accessed 09-July-2020].

[50] ——, “I dribbled a basketball for an entire day,” https://www.youtube.com/watch?v=F8BdzL8OltM, [Online; accessed 09-July-2020].

[51] ——, “Tracy mcgradyon isaiah thomas to the lakers: He can’t bring an ego with him — the jump — espn,” https://www.youtube.com/watch?v=FI3zMdqq4vE, [Online; accessed 09-July-2020].

[52] ——, “Celebrating1 million subscribers at fallon !!” https://www.youtube.com/watch?v=nsjPEcRq_0c, [Online; accessed 09-July-2020].

[53] ——, “Nf - let you down,” https://www.youtube.com/watch?v=fbHbTBP_u7U, [Online; accessed 09-July-2020].

[54] ——, “Kygo - strangerthings ft.onerepublic,” https://www.youtube.com/watch?v=3ChgRbqGi-E, [Online; accessed 09-July-2020].

[55] ——, “Incredibles 2 -olympics sneak peek,” https://www.youtube.com/watch?v=YBpdL9hSac4, [Online; accessed 09-July-2020].

[56] ——, “Why the triple axel is such a big deal,” https://www.youtube.com/watch?v=7rOQv_6L9fQ, [Online; accessed 09-July-2020].

[57] S. Kang and S. Cho, “Approximating support vector machine with artificial neural network for fast prediction,” *Expert Systems with Applications*, vol. 41, no. 10, pp. 4989–4995, 2014.

[58] A. Srinivasan, “Medium,” https://towardsdatascience.com/youtube-views-predictor-9ec573090acb, [Online; accessed 08-July-2020].