Hashtags Prediction for Image Post in Social Networks

A dissertation submitted for the Degree of Master of Computer Science

W.L.A.T.A.L.Weerasooriya
University of Colombo School of Computing
2019

Hashtags Prediction for Image Post in Social Networks

W.L.A.T.A.L.Weerasooriya
2019

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.
To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: W.L.A.T.A.L.Weerasooriya
Registration Number:2016/MCS/003
Index Number:16440033

_____
Signature: Date:

This is to certify that this thesis is based on the work of

Ms. W.L.A.T.A.L. Weerasooriya

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:
Supervisor Name: Dr. Ajantha Athukorale

_____
Signature: Date:

Abstract

Currently social media plays an important role in day to day life of people. Young generation as well as the middle-aged generation tend to share their experiences via social media. This could be a place the visit, could be a food they try out, could be a new dress or could be anything they wish to share with their network. Considering the engagement with social networks, posting images is a good way to share their experiences to a wider audience. Because of that the image posting is very popular in major social networks like Facebook, Instagram, Twitter, Tumbler and etc. Users tend to post some hashtags along with the images they are posting. This can be a small description to an image or a sentiment related with it.

This study is supposed to identify the images post in social networks and predict hashtags for the identified images while considering the users age, gender and geographic location. A CNN-LSTM model is proposed to accomplish the above purpose. CNN model is trained with the public Instagram posts collected and CNN model is able to classify images. For this study we have consider only four image classes which are belonged to two geographic locations. Relevant pre-processed hashtags were fed to the LSTM model along with the output of CNN model. Hashtags were tagged with the users age category, gender and geo location. Output of the LSTM model is the predicted hashtags.

The model was trained 5,10,15,20, 25 and 30 epochs to identify the accurate prediction. The model shows the best accuracy after 20th epoch. This study considered the single object images. As a future work we can consider the images contains more than one object. Also, in this study we consider the age ranges of users due to limited availability of data. So as a next phase we can predict more user specific hashtags. Also, in this study based on the availability of data we consider the base location as New York and London, as a future work we can improve the granularity of the location. Other than that, in this study we only consider the hashtags with English language, as a future work we can consider the prediction of hashtags with native languages.

Acknowledgement

I would like to express my deepest sincere gratitude towards my supervisor, Senior Lecture Dr. Ajantha Athukorale for his proper guidance, supervision, helpful advice and sparing his valuable time from the very early stage of this research as well as giving me extraordinary experiences throughout the work.

I offer my obeisance to the entire senior and junior lecturer panel and staff of UCSC for the help and inspiration they extended advice, supervision, and crucial contribution on this research.

It is pressure to convey my special thanks for my family and friends supported me to make this study success.

Finally, I would like to offer my heartiest gratitude to all the people whose names are not appeared but their untiring effort was very much crucial to make this study success throughout the work.

Table of Content

List of Figures

List of Tables

Abbreviations

CNN- Convolutional Neural Network

CPU- Central Processing Unit

DL- Deep Learning

GPU- Graphics Processing Unit

LRCN- Long-term Recurrent Convolutional Network

LSTM- Long Short Term Memory

LSTM- RNN-Long Short-Term Memory Recurrent Neural Network

ML- Machine Learning

RNN- Recurrent Neural Network

TPU- Tensor Processing Unit

Chapter 1
Introduction

1.1 Introduction

Social Networks play an interesting role in day to day life and now a days people tend to share most of the things in their lives among the network. There are many social networking platforms which are being used for content sharing in various Medias (text, image, audio & video). According to the Global social media research summary 2018; the daily usage of major social networks is as follows. 76% of Facebook users log in daily, whilst 51% do for Instagram. Twitter manages just 42% of users login in daily [1].

Considering the image posting context, according to [2] Facebook manages 300 million photo uploads per day while Instagram users share 95 million posts per day [3]. Most of the time Social Network users tend to post their images with a little description about the image. This could be the exact content of the image or could be a sentiment related with it. In that case users tend to use hashtags to express their opinions about the image and tag the image with similar content. These hashtags can be different according to the age, gender, geographic location and etc. of the users.

1.2 What is a hashtag?

Simply hashtag is a text preceding "#" sign (ex: #MCS) [4]. This can be defined as a label for content or a metadata tag to describe content [5]. Once content is tagged with a particular hashtag; it is bound to the similar content. Hashtags help to categorize topics and gather similar ideas together. This will help users to find related things with a particular hashtag if they have an interest. Hashtags are generated by users themselves.

Usage of hashtags is very popular in Social networks such as Twitter, Instagram, Facebook, YouTube, Google+, Pinterest, Tumblr and etc. [4], [6]. The usage of Hashtag was introduced by Chris Messina who is the designer of Twitter in 2007. He suggested users to use the hash sign or the pound sign to denote posts for groups. Twitter uses clickable hashtags which are directing users to similar tweets since 2009. Facebook added hashtag feature since 2013 [7]. Instagram introduced hashtags since 2011 January [8].

Hashtag may contain letters, underscores and numbers. Spaces are not allowed in hashtags. Special Characters (ex: "!, $, %, ^, &, *, +, .") are also not allowed in hashtags [4]. Hashtags are not case sensitive which means results provided for #mcs, #MCS and #Mcs would contain similar posts [7]. Different social networks have kind of slight different changes in their functionality; but more or less it's same.

Following sub sections explain the usage of hashtags on Instagram and Facebook which are two major and popular photo sharing social networking platforms.

1.2.1 Hashtags on Instagram

According to [9] Instagram posts with at least one hashtag get 12.6% more engagement. Instagram users can only post up to 30 hashtags per post. Numbers are allowed in hashtags on Instagram. However, spaces and special characters, like $ or %, won't work [10]. Instagram has two types of hashtags which are branded hashtags and community hashtags. Branded hashtags are associated with a brand or a company or a product and 70% of Instagram hashtags are branded hashtags. Community hashtags are generic and not associated with a brand. It is recommended to use hashtags which are matched to the photo since it improves the discoverability of the post [11].

1.2.2 Hashtags on Facebook

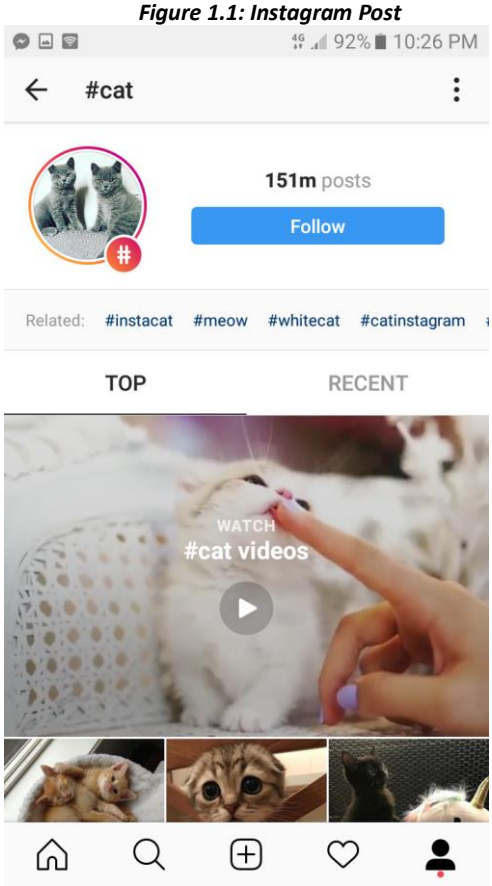
Facebook allows numbers and restrict the use of spaces, punctuation and special characters (like $ and %) in their hashtags. It is recommended to use simple and relevant hashtags for the posts [12], [13]. A Study by Social Bakers shows that usage of 1-2 hashtags have a significant engagement (593 interactions). According to their study usage of more than 2 hashtags would drop the engagement [14].

Figure 1.1 shows an Instagram post with hashtags while Figure 1.2 shows a Facebook post with hashtags.

Both platforms (Facebook, Instagram) allows users to find similar or related content by clicking on a particular hashtag. Figure 1.3 and Figure 1.4 shows how related posts are list down in Instagram and Facebook respectively in a mobile view.

Figure 1.1: Instagram Post

Figure 1.2: Facebook Post



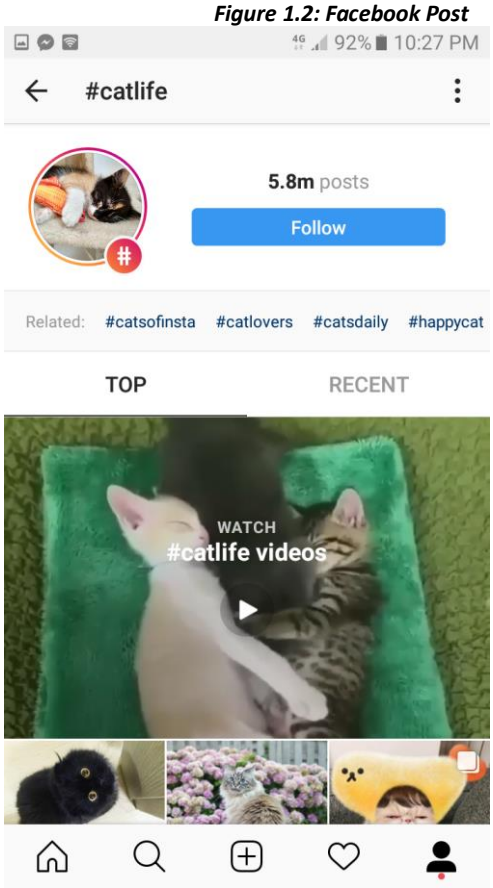


Figure1.3: Search similar posts by hashtag in Instagram

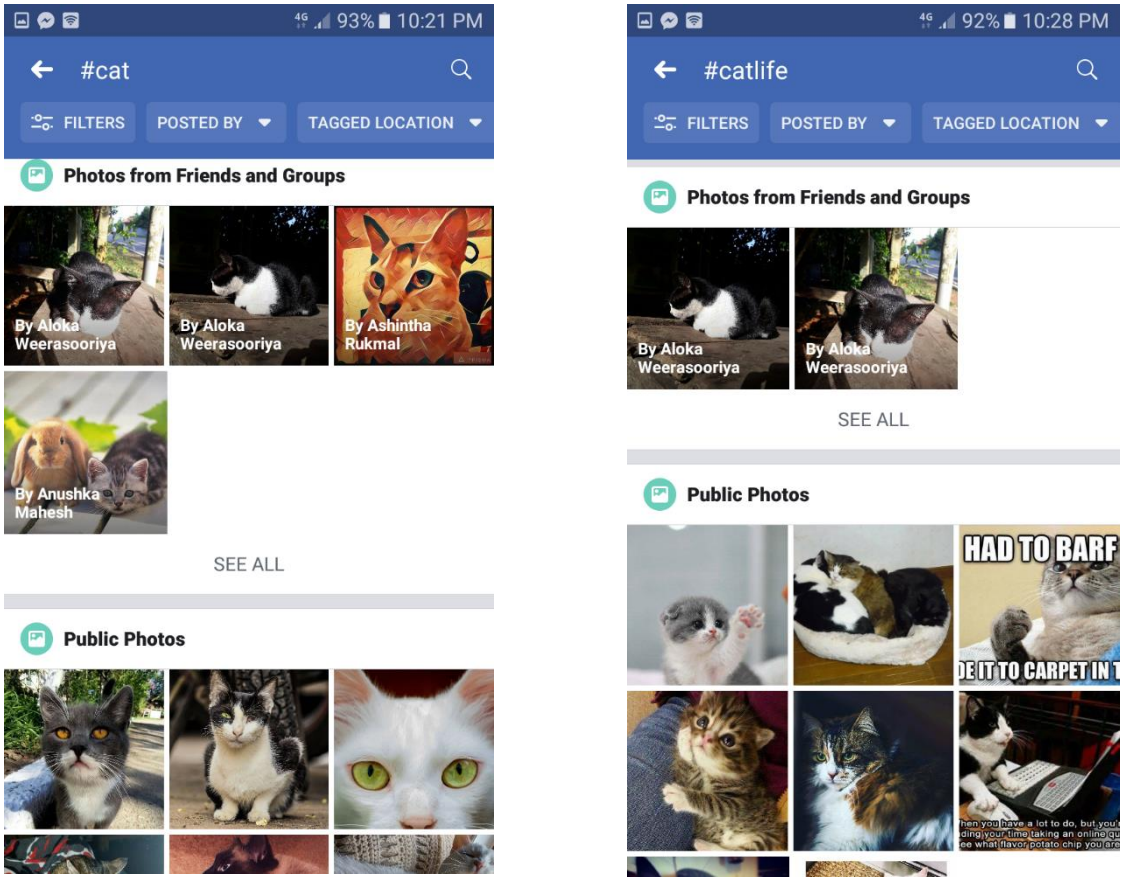


Figure 1.4: Search similar posts by hashtag in Facebook

1.3 Problem Definition

As we all know the world is moving very fast & people are continuously seeking easy ways for doing things. Considering the image posting context in social network, currently users have to manually enter hashtags while posting an image. In that case they have to think about the hashtags related with their image and type it. Sometimes this may be a cumbersome for them with the busy life styles of modern society.

At present we could not see a proper mechanism to detect the content of the image and predict relevant hashtags for the photos which are posting on Social Networks. It would be easy and more user friendly if we can have a proper mechanism to predict appropriate and suitable hashtags for image postings.

Under the domain of computer vision there are state of the art mechanisms designed for understanding images (image processing, deep learning). Also, there are mechanism designed for automatic text generation. Though we have these mechanisms separately, there is a lack of incorporating these two mechanisms together. Therefore, in this study we are going to find a proper

mechanism to detect the content of image and generate texts or hashtags relevant to that image with the aid of computer science.

1.4 Objectives

This study is supposed to find an appropriate mechanism to detect the content of image posts and predict most relevant hashtags for the detected image while taking into account user's contextual data such as age, gender, spatial temporal data, etc. and user inputs. Below are the identified objectives for this study,

1. Find a proper mechanism to detect the content of an image
2. Find a suitable mechanism to recommend hashtags based on the user inputs and contextual data (age, gender, spatial temporal data)
3. Incorporate above two mechanisms together in order to predict hashtags for the images upload in Social Networks
4. Ease Social Network users by predicting hashtags while uploading the images
5. Provide a more user friendly experience for users

1.5 Scope

As stated above this project aims to find a suitable solution to predict most appropriate hashtags by incorporating a proper image understanding mechanism and a hashtag recommendation mechanism. Image data, image description (texts and hashtags) and user contextual data (age, gender and spatial temporal data) will be used to formulate the model.

As the scope of this project images with single object would be considered. Below are the assumptions & limitations identified for the study.

1.5.1 Assumptions

Quality of the images use for the study is similar.

1.5.2 Limitations

Hashtags posted with the native languages other than English could not be predicted.
Numerical hashtags could not be predicted.
Hashtags with Emoji's (smileys) and special characters could not be predicted

1.6 Structure of the Thesis

Beyond this point of introduction, the thesis will be structured as follows.

Chapter 2 will provide a comprehensive review of the literature regarding previous research attempted related to this study.

Chapter 3 will describe the high-level system architecture and the detailed description of the research methodology followed in the study. Also, it describes the implementation details of the proposed solution.

Chapter 4 Contains the model training process and the evaluation of the obtained results and finally

Chapter 5 will describe the conclusion of the study and the prospective future work of the study

Chapter 2
Literature Review

2.1 Introduction

In this chapter, the background and previous work related to this project will be discussed. The literature on Machine Learning (ML), Deep Learning (DL), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) Networks and its applications are included.

2.2 Machine Learning

Machine Learning (ML) is a science that provides the learning ability to computer systems without being explicitly programmed [15]. To achieve that ML uses statistical techniques. Basically, ML algorithms are categorized into supervised learning and unsupervised learning. Supervised learning uses a set of inputs and the corresponding desired outputs to train the algorithm where unsupervised learning does not have any labeled data to train the algorithm. Here the model or the algorithm find the hidden patterns of data itself [16].

2.2.1 Supervised Learning

Supervised learning uses classification and regression techniques [16].

Classification models classify input data into categories. Image recognition, speech recognition, handwriting recognition and credit scoring are some application of classification. Classification can be used if the data can be tagged, categorized, or separated into specific groups or classes [16].

Classification algorithms:

Support Vector Machine (SVM), boosted and bagged decision trees, k-nearest neighbor, Naïve Bayes, discriminant analysis, logistic regression, and neural networks [16].

Regression techniques predict continuous responses. Typical applications include electricity load forecasting and algorithmic trading. It suites if you are working with a data range or if the nature of the response is a real number [16].

Regression algorithms:

Linear model, nonlinear model, regularization, stepwise regression, boosted and bagged decision trees, neural networks, and adaptive neuro-fuzzy learning [16].

2.2.2 Unsupervised Learning

Unsupervised learning often uses clustering for the development predictive models. Clustering is used to discover the hidden patterns through the exploratory learning. Gene sequence analysis, market research, and object recognition are some applications of clustering technique [16].

Clustering algorithms:

k-means and k-medoids, hierarchical clustering, Gaussian mixture models, hidden Markov models, self-organizing maps, fuzzy c-means clustering, and subtractive clustering [16].

2.3 Deep Learning

Deep Learning (DL) is a sub set of machine learning. Most DL methods use Neural Network architectures. Multi layered Artificial Neural Networks are used between input and output layers. Output of each of these hidden layers is fed as the input to the next layer. Learning can be supervised, semi supervised or unsupervised [17]. A large amount of data is required for deep learning [18].

Deep Neural Networks, Deep belief networks and Recurrent Neural Networks, Convolutional Neural Networks and feed forward neural networks are commonly used DL architectures [17]. These architectures have been used in the field of image recognition, speech recognition, handwriting recognition, Natural Language Processing and etc [17].

2.4 Convolutional Neural Network

Convolutional Neural Network (CNN) is a deep artificial neural network which is widely used in the field of computer vision [19], [20]. CNNs are inspired by human brain. CNNs are commonly used to find patterns in images to recognize objects, faces, and scenes. Since CNNs learn directly from image data using patterns to classify images, it eliminates the need for manual feature extraction. CNNs

provide optimal solution for image recognition combining with the advancement of Graphics Processing Unit (GPU) and parallel computing [20].

2.4.1 Architecture of CNN

Architecture of the CNN is little bit different from traditional neural networks. As depicted in Figure 2.1 the layers of the CNNs are organized in 3-dimentional way (height, width and depth) [21].

.

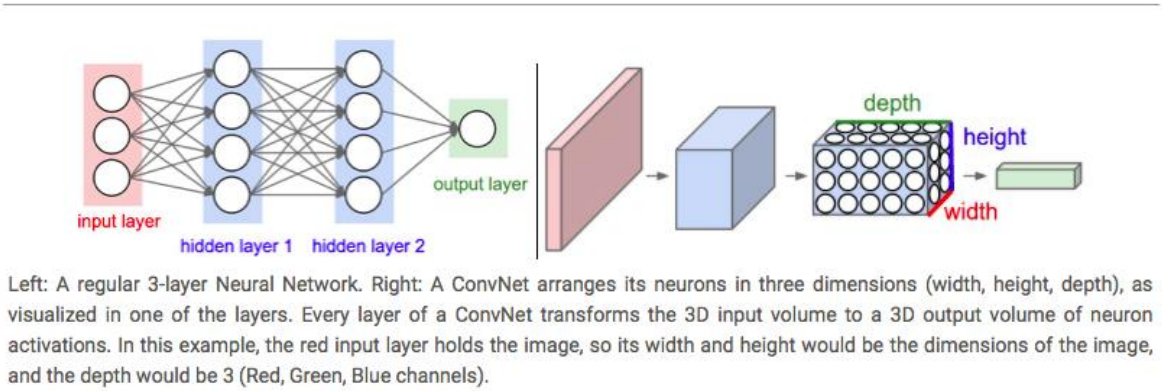


Figure 2.1: Architecture of the CNN *[21]*

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers [19].

Convolutional Layer apply the convolution to the input and pass it to the next layer. CNNs use set of filters in order to detect specific features of a given image. Convolutional layer of the CNN makes use of filters [22].

After convolutional layer, the size of the representation produced by the convolutional layer is reduced. Pooling layers perform the reduction of the size of the output by performing downsampling [23]. Downsampling reduce the number of parameters that the network needs to learn and it speed up the training process and reduce the amount of memory consumed by the network [23].

Most common way of downsampling is max pooling [23]. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer [19]. Max pooling significantly reduces the representation size. Also, it reduces the amount of memory required and the number of operations performed later in the network [23]. Basically, fully connected layers connect every neuron in one layer to every neuron in another layer [19].

2.5 Long Short-Term Memory Networks

A Long Short-Term Memory Network (LSTM) is a type of Recurrent Neural Network (RNN) which is widely used in sequence prediction problems [24], [25]. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate [26]. LSTMs overcomes the exploding and vanishing gradient problems that can be encountered when training traditional RNNs [25]. In LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies which can be generalized to any problem as [25]:

The previous cell state (i.e. the information that was present in the memory after the previous time step)

The previous hidden state (i.e. this is the same as the output of the previous cell)

The input at the current time step (i.e. the new information that is being fed in at that moment)

2.5.1 Architecture of LSTM

Figure 2.2 represented the architecture of the LSTM. As depicted in the diagram classical LSTM consist with different memory blocks which are called as **cells**. The **cell state** and the **hidden state** are two states that are being transferred to the next cell. The memory blocks are responsible for remembering things and manipulations to this memory and it is done through three major **gates** which are **input gate**, **output gate** and **forget gate**. [25]

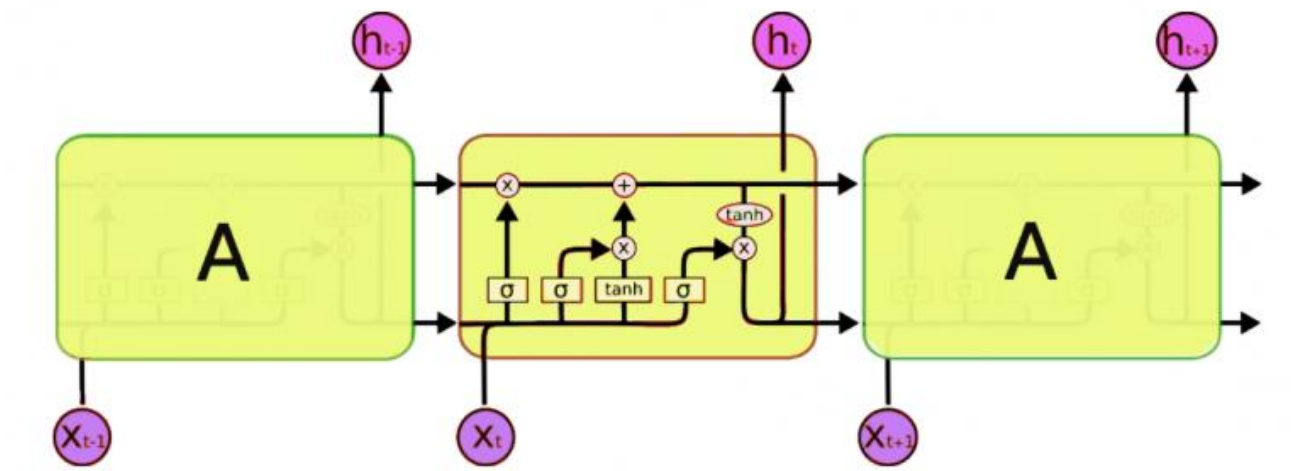


Figure 2.2: Architecture of LSTM *[25]*

The process of adding some new information to the cell state can be done via the input gate. Selecting useful information from the current cell state and showing it out as an output is done via the output gate. Forget gate is responsible for removing information from the cell state [25].

It removes the information that is no longer required for the LSTM to understand things or the information that is of less importance. The removal is done via multiplication of a filter. This is required for optimizing the performance of the LSTM network [25].

2.6 Similar Studies

This section contains the details of similar studies found in literature.

2.6.1 Visual Sentiment Prediction with Deep Convolutional Neural Networks

C. Xu et al [27] proposed a novel visual sentiment prediction framework that performs image understanding with Convolutional Neural Networks (CNN). The proposed framework transfers learning of a CNN on a large-scale dataset to the task of visual sentiment prediction.

According to their methodology they first trained the network on a large-scale image dataset for object classification, and then the learned parameters of the network are transferred to the task of sentiment prediction for generating image-level representations. Finally, classifiers are trained from the features extracted from the sentiment images.

The CNN is composed of seven internal layers and ultimately a soft-max layer. The hidden layers are five successive convolutional layers followed by two fully connected layers. The CNN takes a 224 by 224 pixel RGB image as input. Implementation and training of the CNN has been done using an open source implementation named Caffe [28]. Study has used Logistic Regression as the classifier on top of the generated features.

2.6.2 Separating Self-Expression and Visual Content in Hashtag Supervision

Veit et al [29] present an approach that extends upon modeling simple image-label pairs by modeling the joint distribution of images, hashtags, and users. A user-specific hashtag model was developed that takes the hashtag usage patterns of a user into account and it was trained on image-user-hashtag triplets.

The model is comprised of a convolutional network that feeds image features into a three-way tensor model, which is responsible for modeling the interaction between image features, hashtag embeddings, and an embedding that represents the user. They have investigated two approaches for training image recognition models using user-agnostic hashtag supervision those two approaches are softmax multi-class classification and hashtag embedding regression.

Experiments were conducted on the YFCC100M dataset which contained approximately 99.2 million photos. According the results of the experiment the classification approach performs consistently well across all tasks, whereas the embedding-regression approach mainly performs well for (user-specific) image tagging. In general, the ability to capture user-specific features in the predictions the user-specific approach is better than the user-agnostic model.

2.6.3 #TAGSPACE: Semantic Embeddings from Hashtags

#TAGSPACE [30] is a CNN that learns feature representations for short textual posts using hashtags as a supervised signal. It is used for large scale ranking tasks, and apply it to hashtag prediction. This model represents both words and the entire textual post as embeddings as intermediate steps. In order to investigate the generality of these learned representations the learning of CNN is transferred to the task of personalized document recommendation.

In the experiments two large corpora of posts containing hashtags from a popular social network have been used. The first corpus, consists of 201 million posts from individual user accounts, comprising 5.5 billion words. The second corpus, consists of 35.3 million page posts, comprising 1.6 billion words.

Models has been trained on both the people and page datasets, and collected precision at 1, recall at 10, and mean rank for 50,000 randomly selected posts withheld from training. A further 50,000 withheld posts are used for selecting hyper parameters.

#TAGSPACE is compared with respect to the Frequency and #words models. Frequency, ignores input text, always ranking hashtags by their frequency in the training data. #words, assigns each tag a static score based on its frequency plus a large bonus if it corresponds to a word in the input text. According to their results they have identified that #TAGSPACE performs well, beating several baselines and an industrial system engineered for hashtag prediction.

2.6.4 User Conditional Hashtag Prediction for Images

Denton et al [31] show how user metadata (age, gender, etc.) combined with image features derived from a CNN can be used to perform hashtag prediction. Firstly, user hashtags are employed to capture the description of image content; and secondly, user contextual information has been utilized. These varied features were combined into a learning framework with following two ways (i) simple concatenation; and (ii) a 3-way multiplicative gating, where the image model is conditioned on the user metadata.

This the study is wish to predict the hashtags for a given image uploaded by a given user. Therefore, the learning techniques they used contain feature representations modeling both the image via pixels and the user in the form of metadata.

Three different methods of embedding image descriptors have been proposed in this study. The simplest model which is bilinear model does not use any user information and embeds an image descriptor via a linear mapping. Two different methods have been proposed to incorporate user metadata into the embedding process. The user-biased model introduces a user-dependent additive bias into the image embedding function and the user-multiplicative model has a user descriptor gate, via a multiplication operation, the image embedding function.

The image embedding function takes an image descriptor as input and produces a d-dimensional embedding vector. A pre-trained convolutional neural network has been used as a feature extractor to obtain the image descriptor from the raw image post. The convolutional network was trained separately on approximately 1 million Facebook images. The network was trained to classify 1000 different concepts which span multiple categories such as Object, Scenes, Actions, Food, Animals, etc.

Models were trained by minimizing the weighted approximate rank pairwise (WARP) loss. The image embedding models were evaluated on a large Facebook dataset, consisting of 20 million public images uploaded by nearly 10.4 million de-identified users. The images were collected at random from uploads over a period of several days. The images were selected to have at least one hashtag per image, but often had multiple hashtags. In total, the dataset contained nearly 4.6 million distinct hashtags. The mean number of hashtags per image was 2.7, with a standard deviation of 3.1.

According to the experiments, the user-biased approach gives slight gains over bilinear. But the user-multiplicative model significantly outperforms both showing it is able to make effective use of the user metadata.

These models produce hashtags that capture many subtle social and sentiment of the images, and outperform many current recognition models. The resulting system is highly scalable and could be used in a number of applications such as automated hashtag suggestion, image search or for recommendation and ranking images based on content.

2.6.5 LRCNs for Visual Recognition and Description

Donahue et el [32] describe a class of recurrent convolutional architectures which is end-to-end trainable and suitable for large-scale visual understanding tasks, and demonstrate the value of these models for activity recognition, image captioning, and video description.

They have proposed a Long-term Recurrent Convolutional Network (LRCN) model combining a deep hierarchical visual feature extractor (such as a CNN) with a model that can learn to recognize and synthesize temporal dynamics for tasks involving sequential data (inputs or outputs), visual, linguistic, or otherwise. With this study the proposed models are used for activity recognition, image captioning and, video description.

Activity Recognition

For activity recognition, during the training, videos are resized to 240 × 320 and data were augmented by using 227 ×227 crops and mirroring. Other than that, the LRCN networks are trained with video clips of 16 frames.

LRCN is trained to predict the video's activity class at each time step. To produce a single label prediction for an entire video clip, they average the label probabilities which are the outputs of the network's softmax layer across all frames and choose the most probable label. Evaluation of the proposed architecture has been done with the UCF101 dataset which consists of over 12,000 videos categorized into 101 human action classes. According the results they have identified LRCN shows a significant improvement over the baseline single-frame system and is comparable to accuracy achieved by other deep learning models.

Image Captioning

Compared to activity recognition, the static image captioning task requires only a single invocation of a convolutional network since the input consists of a single image. At each time step, both the image features and the previous word are provided as inputs to the sequence model, in this case a stack of LSTMs (each with 1000 hidden units), which is used to learn the dynamics of the time-varying output sequence, natural language.

Image description model is evaluated in terms of caption retrieval and caption generation tasks Evaluation done with the Flickr30k and COCO 2014 data sets.

Video Description

For video description generation model, they relied on more traditional activity and video recognition processing for the input and use LSTMs for generating a sentence. Following three architectures have been proposed for video description generation, which are LSTM Encoder-Decoder, LSTM Decoder (CRF-max) and LSTM Decoder (CRF-prob).

Evaluation of above approaches has been carried out on the TACoS multilevel dataset, which has 44,762 video/sentence pairs (about 40,000 for training/validation). According to the results they have identified the LSTM outperforms an SMT-based approach to video description. Other than that the simpler decoder architectures which are LSTM Decoder (CRF-max) and LSTM Decoder (CRF-prob) achieve better performance than LSTM Encoder-Decoder.

2.6.6 Tweet Modeling with LSTM RNNs for Hashtag Recommendation

This study [33] have proposed a novel RNN model to learn vector-based tweet representations to recommend hashtags. They used a skip-gram model to generate distributed word representations and then apply a convolutional neural network to learn semantic sentence vectors. Training of the long short-term memory recurrent neural network (LSTM-RNN) have been done using the sentence vectors. LSTM have been used to encode the intrinsic (syntactic or semantic) relations between sentences into the tweet vectors.

They have used the public Twitter streaming API to collect 1,330,908 tweets for experiments. Approximately 20.6% of these tweets contain hashtags. After data preprocessing only 42,000

tweets left. Pre-processed tweets divided into a training dataset (80%), a validation dataset (10%), and a test dataset (10%).

The performance of different approaches were compared with two evaluation methods which are accuracy and hitrate. The proposed LSTM-RNN model is compared with several state-of the art hashtag classification approaches.

According to the results of the experiments (in both accuracy and hitrate comparisons) they have identified that the proposed LSTM-tweet achieves the highest accuracy based on distributed tweet representations and LSTMs. This demonstrates that the proposed method can suggest more accurate hashtags.

2.6.7 Hashtag recommendation with topical attention-based LSTM

Li et al [34] proposed a novel attention-based LSTM model that incorporates topics of microblog posts into the LSTM architecture through an attention mechanism. The model mainly consists of three parts, which are LSTM based sequence encoder, topic modeling, and topical attention. The model is trained in a supervised manner by minimizing the cross-entropy error of the hashtag classification. The dataset consist with 185,391,742 tweets. Among them, there are 16,744,189 tweets including hashtags annotated by users. Randomly selected 500,000 tweets as training set, 50,000 tweets as development and test set respectively.

First train the proposed model on training data, and save the model which has the best performance on the validate dataset. For the experiments comparison of the proposed method has been done with the several baseline methods. Results of experiments show that the proposed model significantly outperforms various competitive baseline methods.

2.7 Comparison of Similar Studies

Table 2.1: Comparison of similar studies

Study	2.6.1	2.6.2	2.6.3	2.6.4	2.6.5	2.6.6	2.6.7
Deep Learning Architecture used	CNN	CNN	CNN	CNN	LRCN	LSTM-RNN	LSTM
Input type	Images	Images	Texts	Images	Images & Video	Texts	Texts
User specific nature	No	Yes	No	No	No	No	No
Spatial Temporal data consideration (location, time)	No	No	No	Yes	No	No	No
User Contextual Data Consideration (age, gender)	No	No	No	Yes	No	No	No
Time series nature of prediction	No	No	No	No	Yes	Yes	Yes
Semantic nature of prediction	No	No	Yes	No	Yes	Yes	No

According to above comparison, a mechanism is needed to predict hashtags for a given user considering user contextual data, spatial temporal data, time series nature and semantic nature of the prediction. This study is supposed to fill the gap identified above.

2.8 Technologies Used in Deep Learning

2.8.1 Tensorflow

Tensorflow [35] is an open source software library developed by Google brain team for numerical computation using data-flow graphs. Its flexible architecture allows easy deployment of computation across a variety of platforms (Central Processing Units (CPU), GPUs, Tensor Processing Units (TPU)), and from desktops to clusters of servers to mobile and edge devices.

Tensorflow provides a strong support for deep learning and machine learning. TensorFlow Lite is a software stack specifically for Android development.

2.8.2 Keras

Keras is an open source neural network library written in Python which is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit or Theano [36]. Keras is designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras supports both convolutional networks and recurrent networks, as well as combinations of the two and it runs on both CPU and GPU [37].

2.8.3 Caffe (Convolution Architecture For Feature Extraction)

Caffe is a deep learning framework developed by the Berkeley Vision and Learning Center (BVLC). It is written in C++ and has Python and Matlab bindings. Caffe supports many different types of deep learning architectures in image classification and image segmentation. It supports CNN, RCNN, LSTM and fully connected neural network designs. Also it supports GPU- and CPU-based acceleration computational kernel libraries such as NVIDIA cuDNN and Intel MKL [38], [39].

2.8.4 PyTorch

PyTorch is an open source machine learning python library which is developed by Facebook's Artificial Intelligence group [40]. It provides two high-level features [41] which are Tensor computation (like numpy) with strong GPU acceleration and Deep Neural Networks built on a tape-based autodiff system.

PyTorch has a unique way of building neural networks: using and replaying a tape recorder [41]. Reverse-mode auto-differentiation of PyTorch allows you to change the way that the network behaves arbitrarily with zero lag or overhead [41].

Chapter 3
Research Methodology & Design

3.1 Introduction

This chapter includes the high-level architecture of the proposed system, its modules and their usage. The module naming and boundaries can be changed alongside with the development.

3.2 Research Methodology

This study is supposed to find an appropriate mechanism to detect the content of image posts and predict most relevant hashtags for the detected image while considering user's contextual data. In order to achieve this goal exploratory type of research is carried out. There are many researches which have studied about different approaches to predict hashtags for image posts in social networks. They have applied different machine learning algorithms to build a proper predictions models for different areas.

But there are a smaller number of researches carried out to predict hashtags which consider the semantic nature of the hashtag while taking into consideration the user contextual data. This study is supposed to design a prediction model to fulfill the above goal by using a proper methodology. From several paths of obtaining the goal, following design describes the most appropriate method selected by analyzing information of the literature survey.

3.3 System Design

In order to design the prediction model first, the collected data need to be carefully analyzed and need to preprocess the data set (images along with the hashtags). CNN and LSTM need to be trained accordingly with the pre-processed image data and hashtag data. Thereafter a model would be formulated by combining the output of the CNN which are classified images, the output of the LSTM which are hashtags or hashtag phrases along with the user contextual data. After that the output of the model will be analyzed to obtain the best outcome. High level architecture of the proposed model is illustrated in Figurer 3.1.

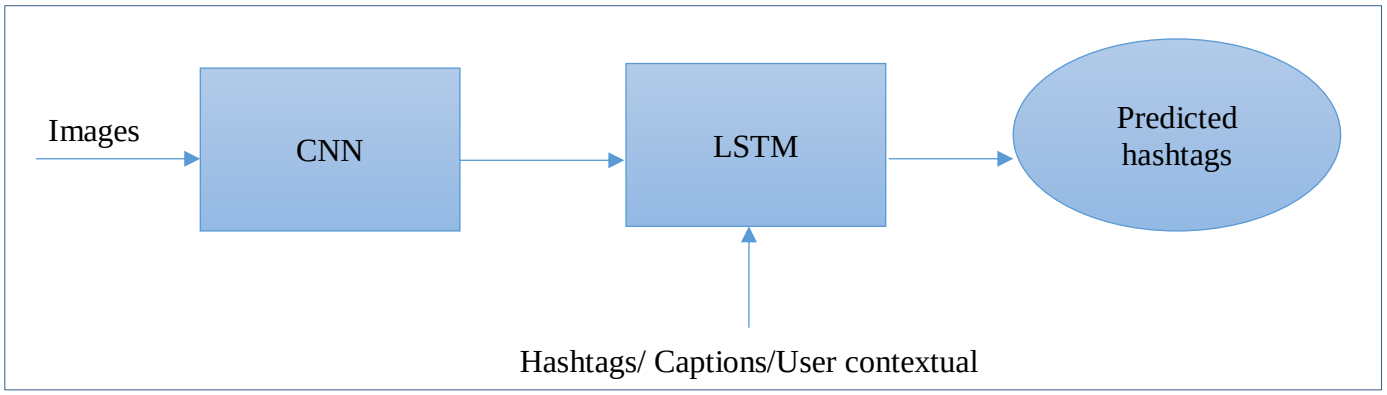


Figure 3.1: High-level Architecture of the proposed system

3.4 Methodology

3.4.1 Data Collection

In order to train the proposed model public Instagram image posts which contains image data along with the hashtags and user contextual data was collected via Netlytic (A system that uses Instagram API). These data were collected around October to December in 2018 for two geographical locations New York and London. Data was recorded in excel files and a selenium script was written to download the images from the given URLs.

3.4.2 Data Pre-processing

Collected data need to be pre-processed before feeding them directly to the CNN and LSTM. For this study we only consider the image posts with single objects along with the English hashtags. Data pre-processing techniques have been applied in order to cleanse the collected data in following ways. Data preprocessing scripts have been written in both java and python.

1. First images contain more than one object was removed and 4 image categories (cat, dog, girl, boy) selected based on the data availability and this was a manual task
2. The hashtags contain numerical characters, Emoji's (smileys) and special characters have been removed using a data cleansing script.
3. The hashtags contained native language words other than the English terms have been removed using the same data cleansing script.
4. Since large amount of image data is required to feed the CNN the collected images were preprocessed in following ways.

3.4.2.1 Augmentation

Augmentation techniques have been used for the collected image data of each category since the number of collected data per each class is limited. Below are the augmentation techniques used in this study.

1. Rotation - apply random rotations, given in degrees in the range from 0 to 180

2. Skewing - skew the given images in a random direction, either left to right, top to bottom, or one of 8 corner directions

3. Shearing - shear intensity, used for linear mapping that displaces each point in a fixed direction

4. Zooming - use for random zooming

5. Flipping - used for randomly flipping inputs horizontally

3.4.2.2 Standardization

Standardization was applied on top of the augmented data. Collected images were in various formats (eg: JPEG, PNG) and the size of the images also varied. While standardizing these images with different formats were converted to a single format which is jpg and the size of the converted images are 300x300 pixels.

Figure 3.2 shows the example images after applying above mentioned preprocessing techniques.

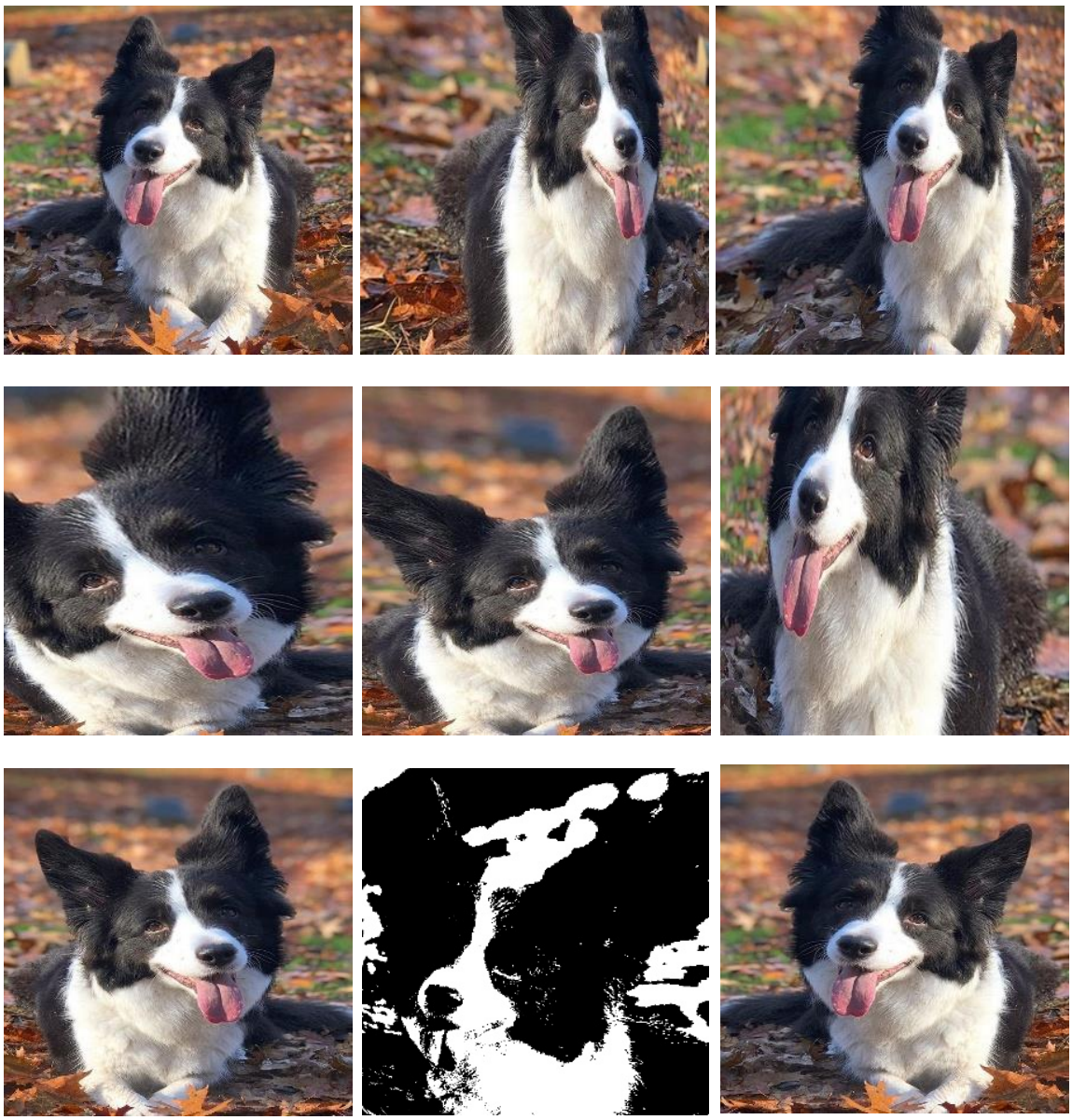


Figure 3.2: Preprocessed Images

4 image classes have been selected based on the image availability and after pre-processing, images of each class are divided into two sets such that test set and training set and feed to CNN.

3.4.3 CNN Model

CNN model creation needs to go through following phases.

Model construction -> Model Training -> Model Testing -> Model Evaluation

CNN model implemented in this study consists with three convolutional layers and three max-pooling. Convolutional layers and max-pooling layers are acting as feature extractors. Input for the max-pooling layer is the output of convolutional layer which is the resultant feature map after

applying the convolution operation on a given image. The aim of max-pooling layer is to reduce the size of the images so that it would help to improve the computational efficiency. Pooled images then converted into a 1-D feature vector through flattening. Output of the flattening is the input to the fully connected layer. Fully connected layer which performs non-linear transformations of the extracted features and acts as the classifier. A dropout is added to avoid overfitting. Since this study is focus on multi class classification softmax is used as the activation function [42]. Figure 3.3 is a schema for the implemented CNN.

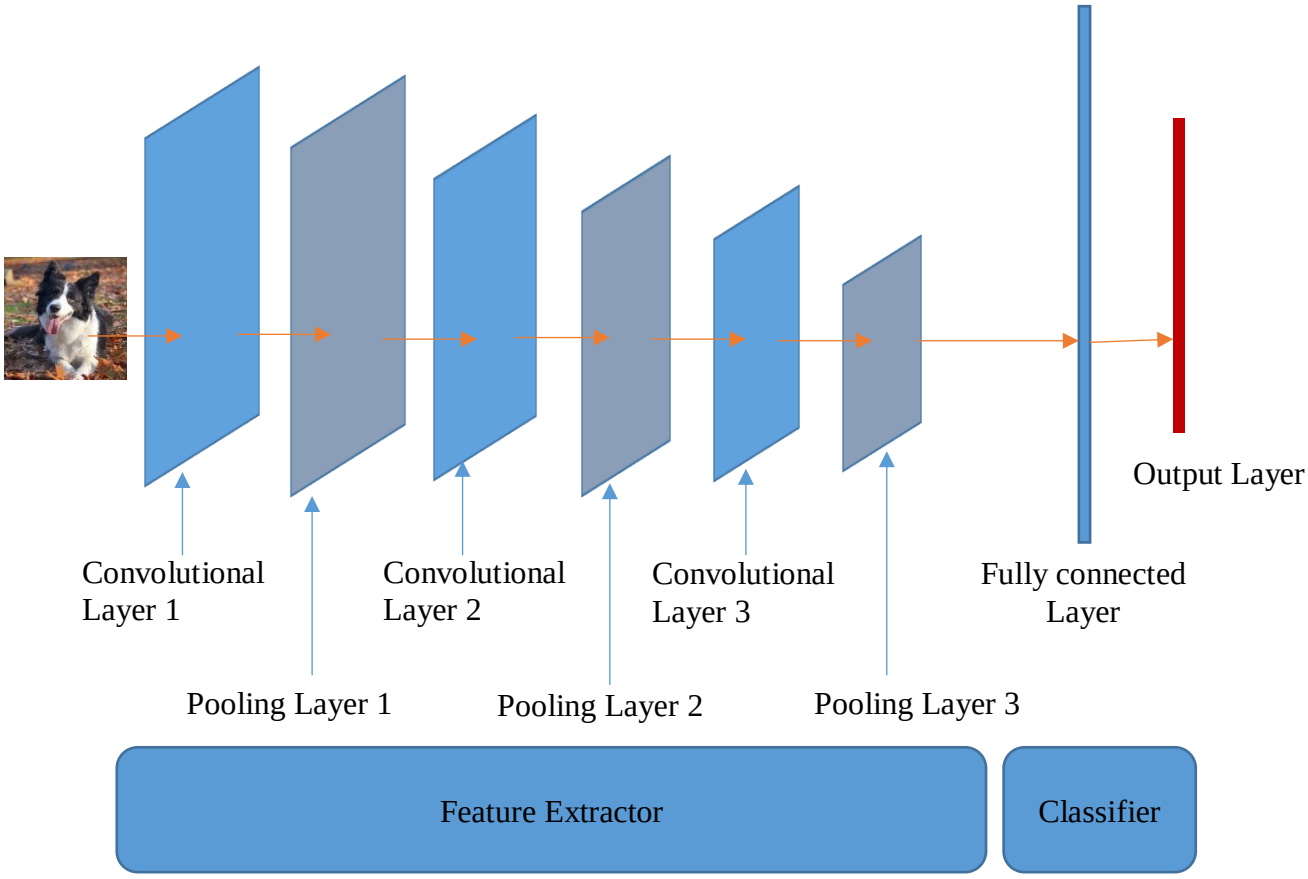


Figure 3.3: Schema for the implemented CNN

Following Figure 3.4 describes the model summary of the implemented CNN model.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 298, 298, 32)	896
activation_1 (Activation)	(None, 298, 298, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 149, 149, 32)	0
conv2d_2 (Conv2D)	(None, 147, 147, 32)	9248
activation_2 (Activation)	(None, 147, 147, 32)	0
max_pooling2d_2 (MaxPooling2	(None, 73, 73, 32)	0
conv2d_3 (Conv2D)	(None, 71, 71, 64)	18496
activation_3 (Activation)	(None, 71, 71, 64)	0
max_pooling2d_3 (MaxPooling2	(None, 35, 35, 64)	0
flatten_1 (Flatten)	(None, 78400)	0
dense_1 (Dense)	(None, 64)	5017664
activation_4 (Activation)	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 2)	130
activation_5 (Activation)	(None, 2)	0

Total params: 5,046,434
Trainable params: 5,046,434
Non-trainable params: 0

Figure 3.4: CNN Model Summary

3.4.4 LSTM Model

As depicted in Figure 3.5 hashtags were preprocessed before feeding them to LSTM model. Each of the steps in the diagram will describe in this section.

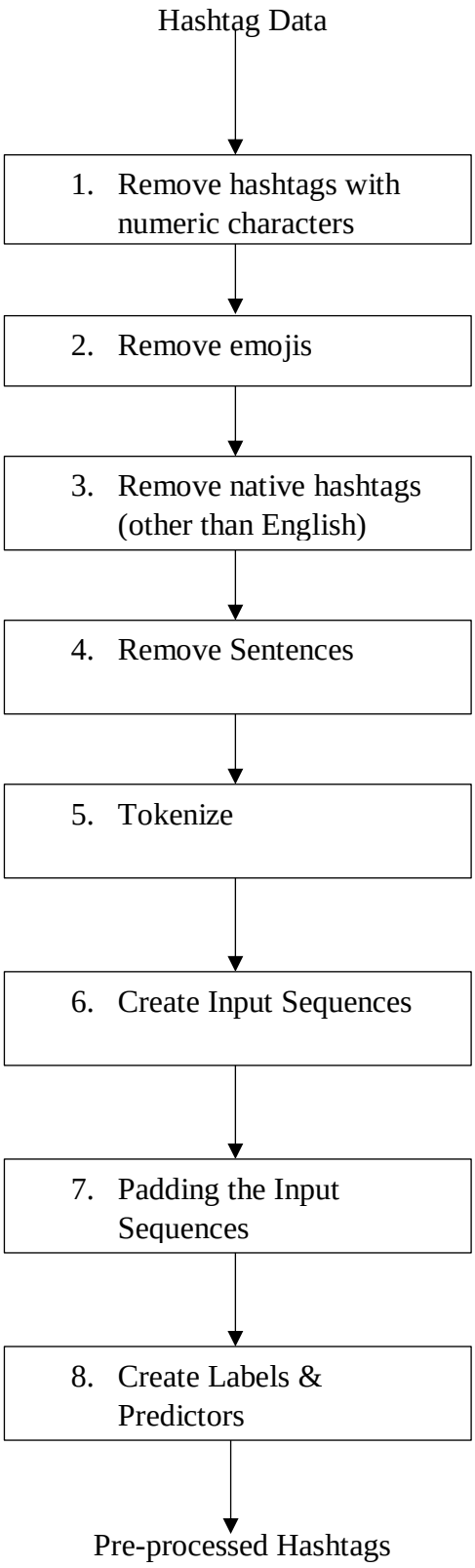


Figure 3.5: Hashtags Preprocessing Steps

Step 1- Numerical hashtags (Example: #21, #30) were removed.

Step 2- Emojis which are typed along with hashtags were removed.

Consider the following example hashtags.

Example:

😍💎💝 #love #my #girls #thanksgiving #thankful #blessed #family #fun

After the second step it removed the 😍💎💝. The output is "#love #my #girls #thanksgiving #thankful #blessed #family #fun".

Step 3 - hashtags with native language have been removed.

Example:

#NY#バーグドルフグッドマン いるだけでテンションあがるね。 #blackfriday 結局何も買えないよね #毎年恒例 #今まで1番の旅行 #グランドセントラル駅#街中クリスマス #今からみんなの土産のチョコかってくるかなー。 #キーホルダーも探さねば。#あいらぶNY でいいよね。 #和食食べたい。 #あったかい味噌汁#寿司とか。うなぎとか。 日本が1番美味しくて安い。 お風呂という文化も素晴らしい。日本は本当に豊かな国だな。大切にしたいね。 とにかくね、あたたかくなりたいね。

After applying the third step. Output is #NY #blackfriday.

Step 4 - The sentences without hashtags removed. Consider the following example.

Example:

The best of NYC #friends #holiday #vacation #barcade #love #toomuchfun 💝💝

After applying the fourth step output is "#friends #holiday #vacation #barcade #love #toomuchfun"

Step 4

As the final step, previously preprocessed hashtags were tokenized. Tokenization is applying on these hashtags because raw data cannot be feed to the LSTM model. With the tokenization raw data was converted to a set of integers. This was done with the help of Keras. Below is an example tokenized output.

Example:

Hashtags:

#girl #newyork #family #newhaircut #pretty #beauty #centralpark #amazing #goodtimes #nikon #dog #goodfriend #mylove #GS #doglovers #girl #family

Tokenized output:

[1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1, 2]

According to the above example when it is identified for the first time it gives an integer token if it repeats the same hashtag then it gives the previously tokenized integer value.

Step 6 - create input sequences from tokenized hashtags. This has helped to create multiple learning from a given hashtag. Below is the example input sequences generated for the above set of hashtags.

Example input sequences:

[[1, 3], [1, 3, 2], [1, 3, 2, 4], [1, 3, 2, 4, 5], [1, 3, 2, 4, 5, 6], [1, 3, 2, 4, 5, 6, 7], [1, 3, 2, 4, 5, 6, 7, 8], [1, 3, 2, 4, 5, 6, 7, 8, 9], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1, 2]]
[[1, 3], [1, 3, 2], [1, 3, 2, 4], [1, 3, 2, 4, 5], [1, 3, 2, 4, 5, 6], [1, 3, 2, 4, 5, 6, 7], [1, 3, 2, 4, 5, 6, 7, 8], [1, 3, 2, 4, 5, 6, 7, 8, 9], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1], [1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1, 2]]

Step 7 - Since there is a possibility to have different lengths for the input sequences generated in the above step, need to pad them in order to make their length equal before feeding them to the model. This was done by adding 0 to the generated input sequences. Take the lengthiest sequence

and padded the other sequences by adding 0 to the length of longest sequence. Below is an example padded sequences for the above generated input sequences.

Example Padded Sequences:

```
[[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  3]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  3  2]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  1  3  2  4]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  1  3  2  4  5]
 [ 0  0  0  0  0  0  0  0  0  0  0  1  3  2  4  5  6]
 [ 0  0  0  0  0  0  0  0  0  0  1  3  2  4  5  6  7]
 [ 0  0  0  0  0  0  0  0  0  1  3  2  4  5  6  7  8]
 [ 0  0  0  0  0  0  0  0  1  3  2  4  5  6  7  8  9]
 [ 0  0  0  0  0  0  0  1  3  2  4  5  6  7  8  9 10]
 [ 0  0  0  0  0  0  1  3  2  4  5  6  7  8  9 10 11]
 [ 0  0  0  0  0  1  3  2  4  5  6  7  8  9 10 11 12]
 [ 0  0  0  0  1  3  2  4  5  6  7  8  9 10 11 12 13]
 [ 0  0  0  1  3  2  4  5  6  7  8  9 10 11 12 13 14]
 [ 0  0  1  3  2  4  5  6  7  8  9 10 11 12 13 14 15]
 [ 0  1  3  2  4  5  6  7  8  9 10 11 12 13 14 15  1]
 [ 1  3  2  4  5  6  7  8  9 10 11 12 13 14 15  1  2]]
```

Step 8 - as the final step we need to create predictors and labels from the given hashtags. In this study N-grams sequence is taken as predictors and the next word of the N-gram is taken as label. Consider the following example.

#girl #newhaircut #pretty #beauty #centralpark #newyork

Below table Table 3.1 shows the example predictors and labels for the above given hashtags.

Table 3.1 Example Predictors & Labels

PREDICTORS	LABEL
#girl	#newhaircut
#girl #newhaircut	#pretty
#girl #newhaircut #pretty	#beauty
girl #newhaircut #pretty #beauty	#centralpark
#girl #newhaircut #pretty #beauty #centralpark	#newyork

Since we have tokenized the hashtags output vectors of the predictors and labels as follow.

Predictors:

[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 3]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 3 2]

[0 0 0 0 0 0 0 0 0 0 0 0 0 1 3 2 4]

[0 0 0 0 0 0 0 0 0 0 0 0 1 3 2 4 5]

[0 0 0 0 0 0 0 0 0 0 0 1 3 2 4 5 6]

[0 0 0 0 0 0 0 0 0 0 1 3 2 4 5 6 7]

[0 0 0 0 0 0 0 0 0 1 3 2 4 5 6 7 8]

[0 0 0 0 0 0 0 0 1 3 2 4 5 6 7 8 9]

[0 0 0 0 0 0 0 1 3 2 4 5 6 7 8 9 10]

[0 0 0 0 0 0 1 3 2 4 5 6 7 8 9 10 11]

[0 0 0 0 1 3 2 4 5 6 7 8 9 10 11 12]

[0 0 0 1 3 2 4 5 6 7 8 9 10 11 12 13]

[0 0 1 3 2 4 5 6 7 8 9 10 11 12 13 14]

[0 1 3 2 4 5 6 7 8 9 10 11 12 13 14 15]

[1 3 2 4 5 6 7 8 9 10 11 12 13 14 15 1]]

Labels:

[0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

[0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

Now the hashtags are ready to feed to the LSTM models. Figure 3.6 is the schematic representation of the LSTM model implemented in this study. Preprocessed hashtags provided as the input. Three LSTM layers with 100 units per each layer is added to compute the output. Dropout layer is added to avoid the overfitting. Softmax is used as the activation function which is good at categorical classifications [42]. Therefor the output computes the probability of the best possible next hashtag as output.

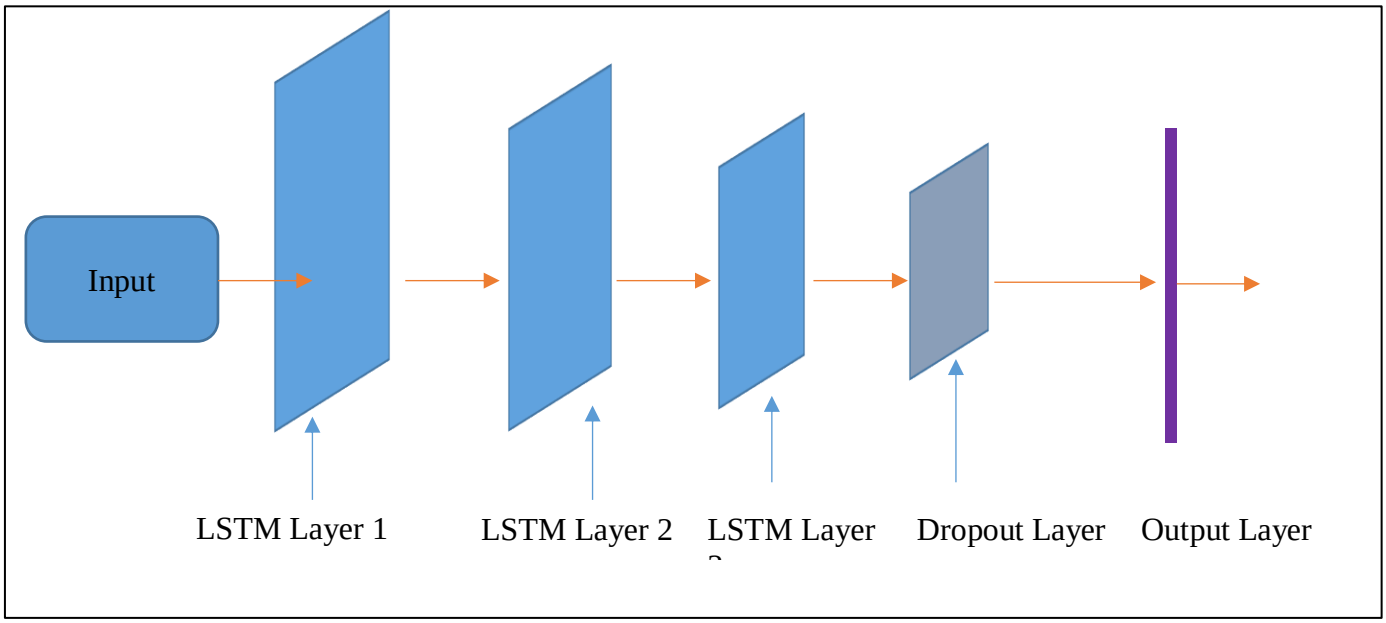


Figure 3.6: Schema for implemented LSTM

Figure 3.7 shows the model summary of the implemented LSTM model.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 8, 10)	100
lstm_1 (LSTM)	(None, 8, 100)	44400
lstm_2 (LSTM)	(None, 8, 100)	80400
lstm_3 (LSTM)	(None, 100)	80400
dropout_1 (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 10)	1010

Total params: 206,310
Trainable params: 206,310
Non-trainable params: 0

Figure 3.7: LSTM Model Summary

3.4.5 Tools used for model implementation

Model is implemented in python programming language using TensorFlow [35] which is an open source software library developed by Google brain team. TensorFlow GPU version is used and Keras Library is used on top of Tensorflow. Keras is also an open source neural network library written in Python [36]. Pandas [43] and scikit-learn [44] libraries are used for data mining and data analysis. Both of these libraries are open source, BSD-licensed [43], [44]. Implementation is done with PyCharm IDE developed by JetBrains [45]. Below is the summary of the implementation environment.

Python	3.5.3
Tensorflow-gpu	1.10.0
Keras	2.1.6
pandas	0.23.4
scikit-learn	0.20.2
PyCharm	2018.3

Chapter 4
Evaluation & Results

4.1 Introduction

This chapter describes how the training of the CNN-LSTM model is carried out with the collected preprocessed data set and the evaluation of the results of CNN-LSTM model. It shows how accurate the model is and how the accuracy changes based on the number of epochs it used to train the CNN-LSTM model.

4.2 Model Training Process

As explained before collected images were categorized into 4 classes as girl, boy, cat and dog. Images of each class were divided into two sets.

Training set – for training the model
Test set - for validating the model

After augmenting the images training set contains 10000 images per each class and test set contains 2000 images per each class. Respective preprocessed hash tags also divided as training and test sets. Image data were fed into CNN model. Classes were labeled as girl, dog, cat, boy. That means every image inside the girl folder is identified as a girl by the network. Below is the folder structures for training and test sets.

training_set
- boy
- cat
- dog
- girl

test_set
- boy
- cat
- dog
- girl

Output of the CNN model which is classified images were feed into the LSTM model. Preprocessed hashtag data along with the user contextual data directly feed to the LSTM. Only two geographic locations have selected for this study which are New York and London. Geo coordinates of New York (42.9254°, 78.8696°) and London (51.5211°, 0.1166°) feed to the LSTM model along with the hashtags of corresponding location. Since the limited availability of data users are grouped into three age categories which are 18-25, 25-30 and 30-35. Relevant Age category of the user is also labeled with the hashtags. Similarly gender of the user is also feed along with the hashtags. Hashtags also divided into training and validation sets accordingly. Output of the LSTM is the predicted hashtags. For the scope of this project only three hashtags will be predicted.

4.3 Model Training Results

In order to identify the best prediction, the model was trained on different number of epochs. For this study the model was trained on 10, 25, 30 & 50 epochs respectively. Following sub sections will analyze the training accuracy, validation accuracy, training loss and validation loss of each training cycle.

4.3.1 Model Training Results for 10 epochs

After executing the model training for 10 epochs validation accuracy of the model is 0.5. Figure 4.1 and Figure 4.2 depicted the loss and accuracy curves respectively. These graphs were plotted while training the model using matplotlib python library [46]. As depicted in Figure 4.1 training loss seems nearly constant (after 10 epochs training loss is 0.4701) where validation loss is very high. After 10 epochs validation loss is 1.6765.

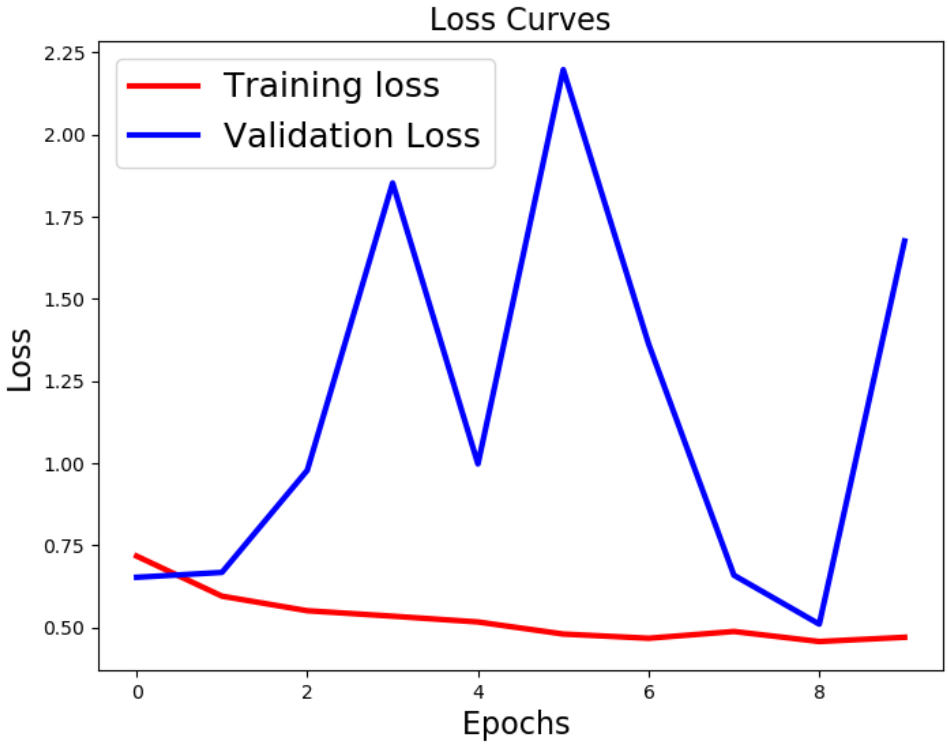


Figure 4.1: Training & Validation Loss Curves for 10 epochs

Figure 4.2 depicted the training and validations accuracy curves. As depicted in the figure after 10 epochs training accuracy is 0.7995. But the validation accuracy is fluctuation and after 4th and 6th epochs highest validation accuracy for the cycle which is 0.75 was recorded. But after the 10th epoch validation accuracy is recorded as 0.50.

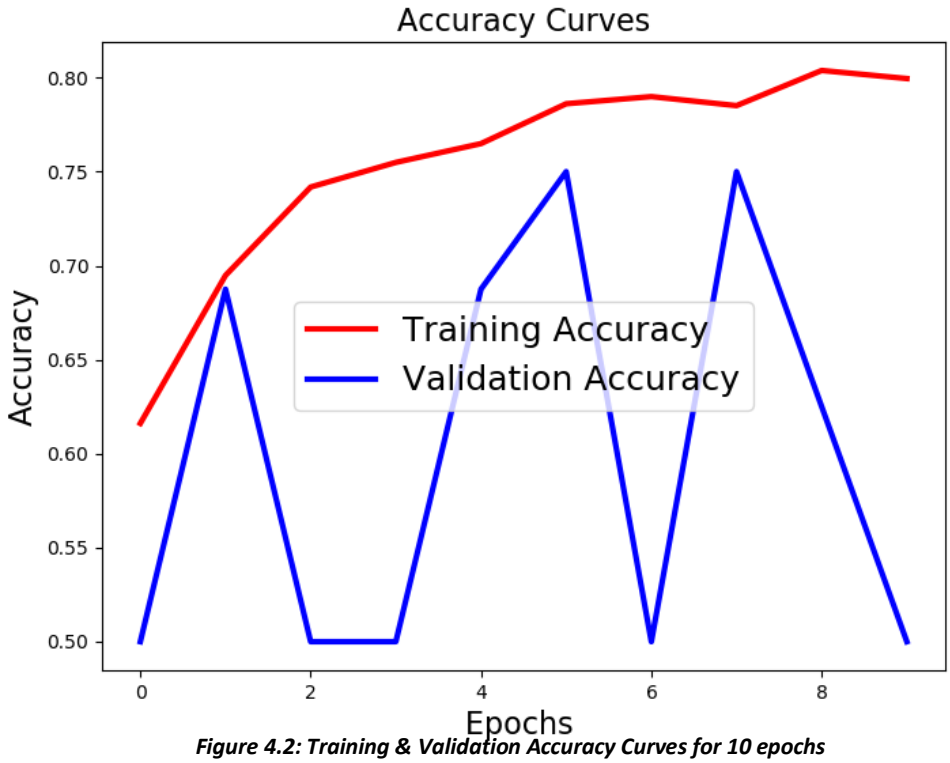


Figure 4.2: Training & Validation Accuracy Curves for 10 epochs

4.3.2 Model Training Results for 15 epochs

Figure 4.3 and Figure 4.4 displays the loss and accuracy curves for model training for 15 epochs. As the training loss and validation loss curves depicted in figure 4.3 after 15 epochs training loss is 0.4469 and validation loss is 2.8461.

According to Figure 4.4 after 15 epochs, training accuracy is 0.8125 and validation accuracy is 0.6250.

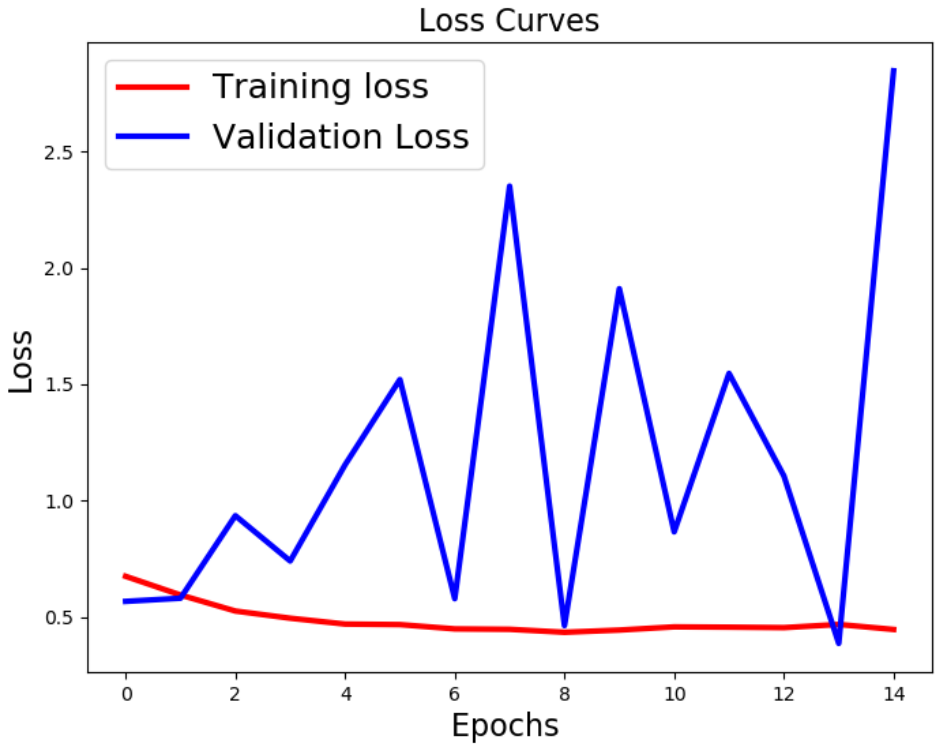


Figure 4.3: Training & Validation Loss Curves for 15 epochs

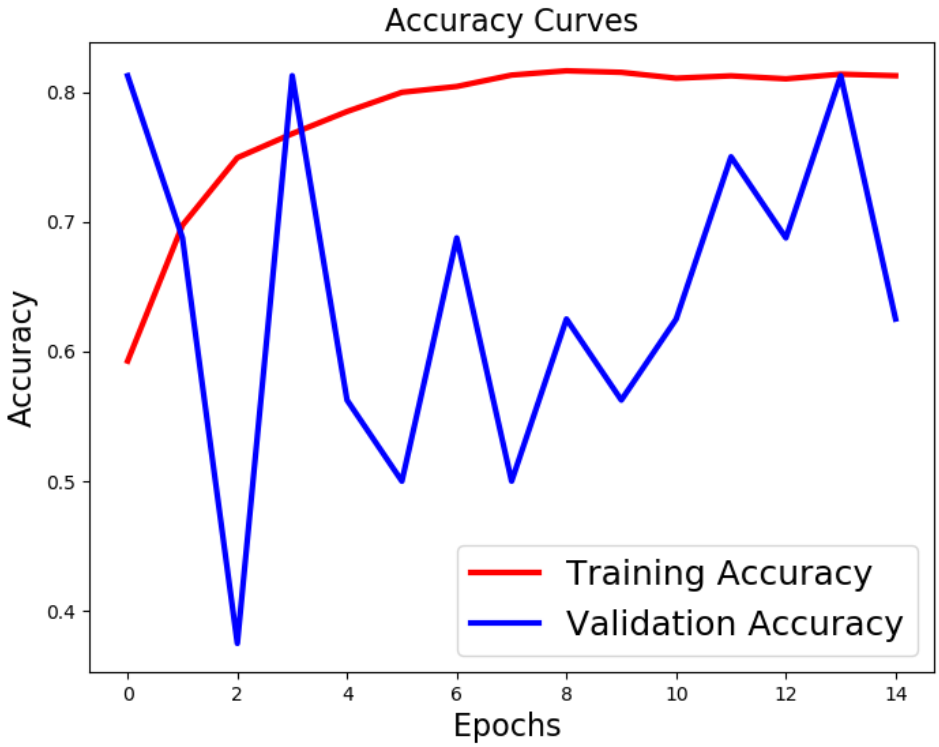


Figure 4.4: Training & Validation Accuracy Curves for 15 epochs

4.3.3 Model Training Results for 20 epochs

Figure 4.5 and Figure 4.6 are the loss and accuracy curves for model training for 20 epochs. As the training loss and validation loss curves depicted in figure 4.6 after 20 epochs training loss is 0.5048 and validation loss is 0.2975.

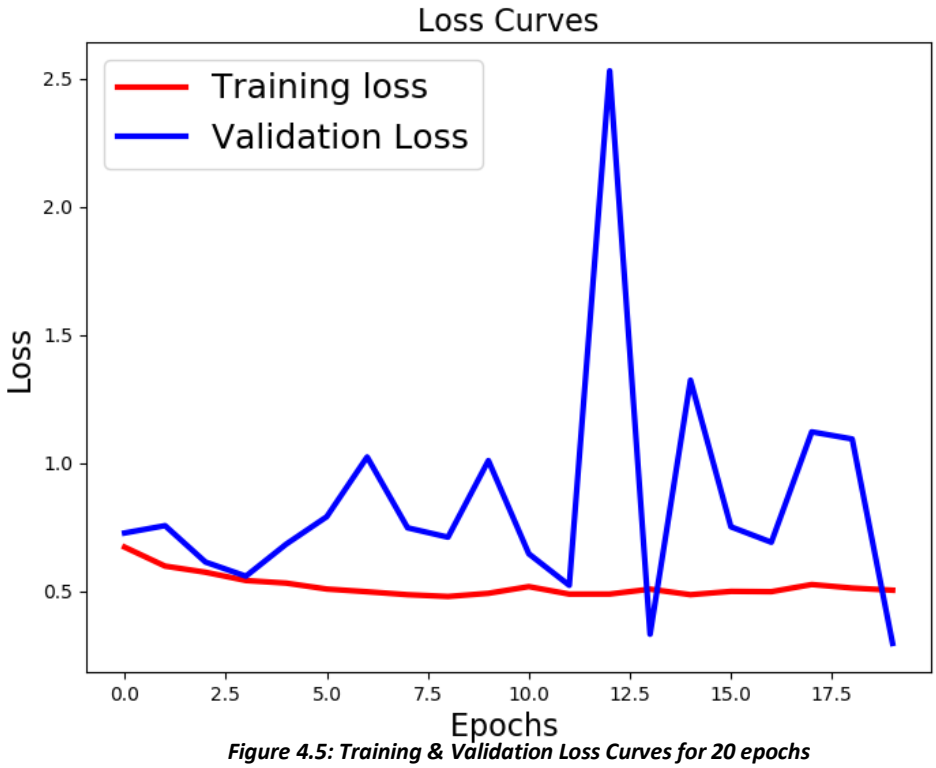


Figure 4.5: Training & Validation Loss Curves for 20 epochs

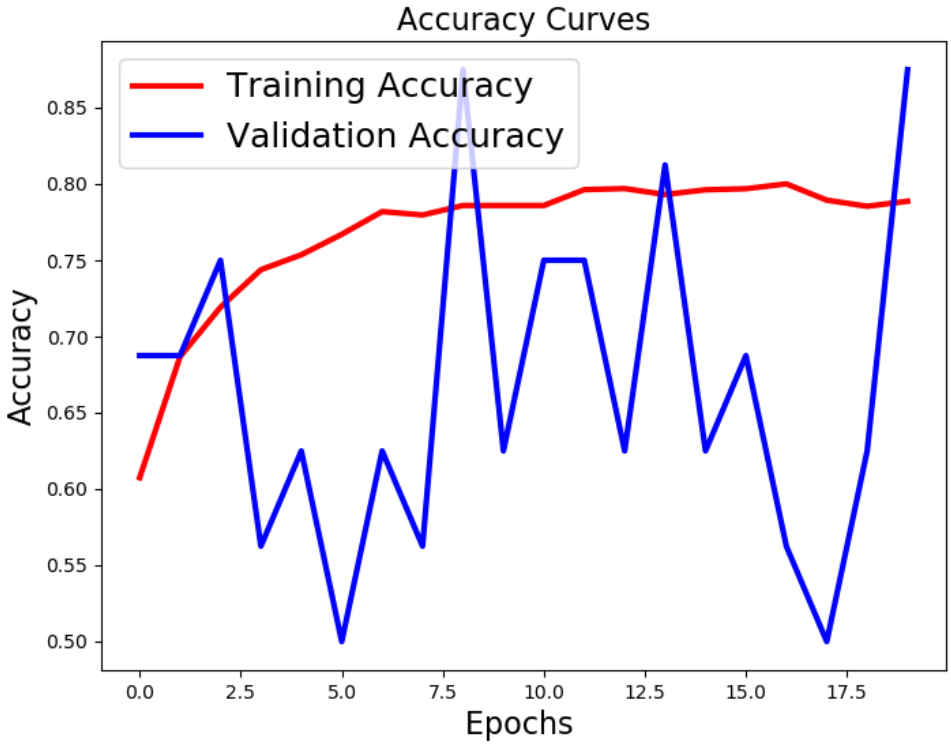


Figure 4.6: Training & Validation Accuracy Curves for 20 epochs

Figure 4.6 displays the training and validation accuracy curves for 20 epochs. After 20 epochs training accuracy is 0.7887 and validation accuracy is 0.8750.

4.3.4 Model Training Results for 25 epochs

Figure 4.7 and Figure 4.8 displays the loss and accuracy curves for model training for 25 epochs. As the training loss and validation loss curves depicted in figure 4.7 after 25 epochs training loss is 0.6728 and validation loss is 0.5694.

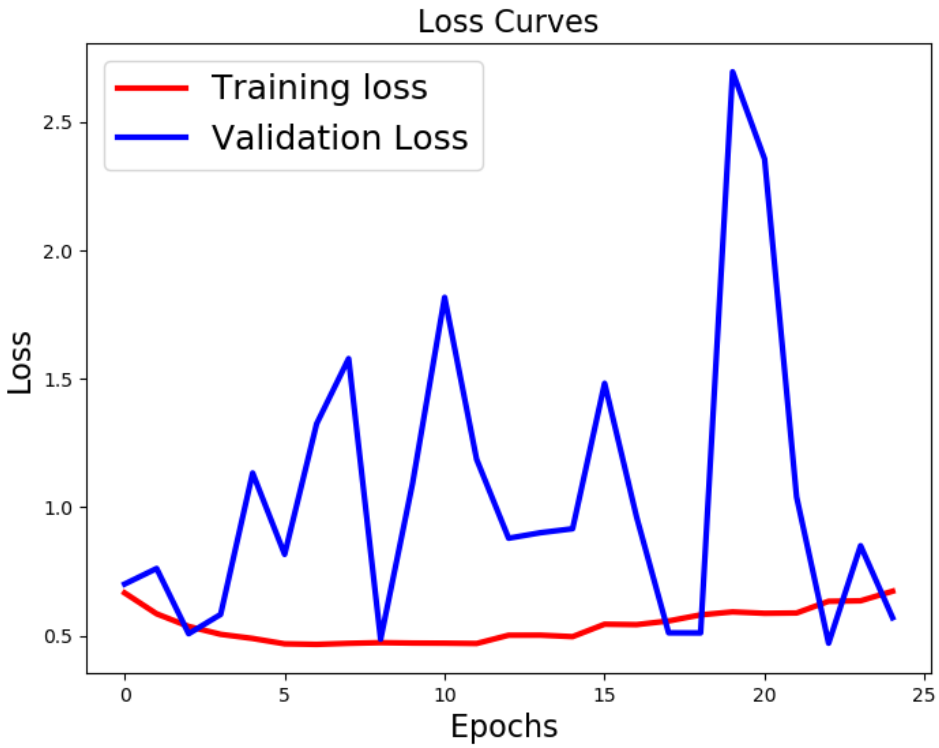


Figure 4.7: Training & Validation Loss Curves for 25 epochs

Figure 4.8 displays the training and validation accuracy curves for 25 epochs. As per the curves, after 25 epochs training accuracy is 0.7244 and validation accuracy is 0.6875.

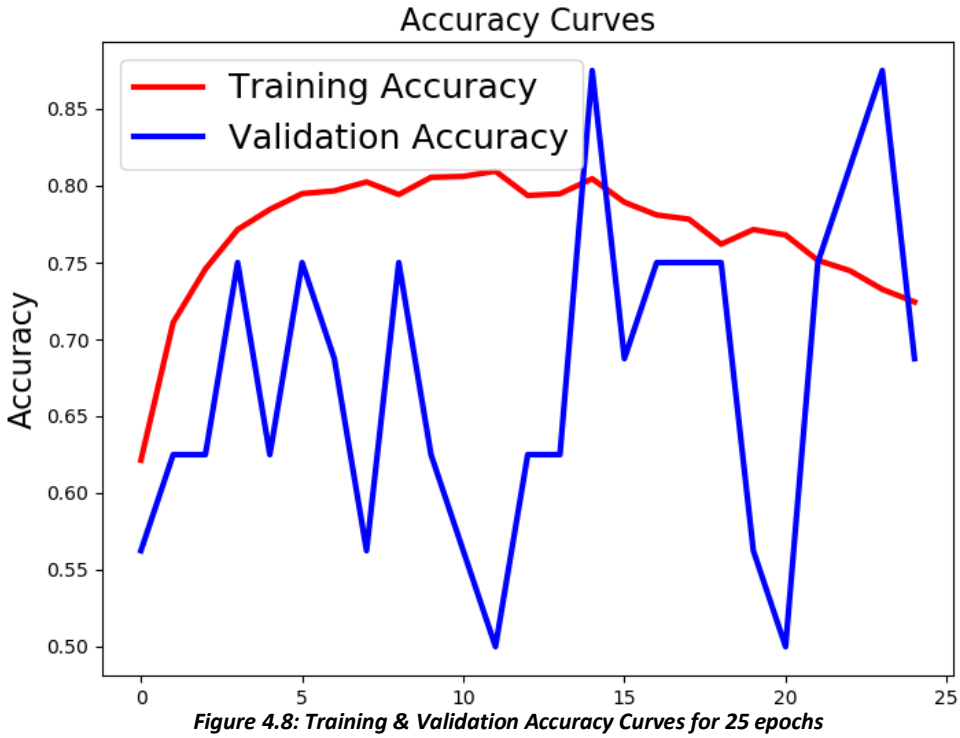


Figure 4.8: Training & Validation Accuracy Curves for 25 epochs

4.3.5 Model Training Results for 30 epochs

Figure 4.9 and Figure 4.10 displays the loss and accuracy curves for model training for 30 epochs.

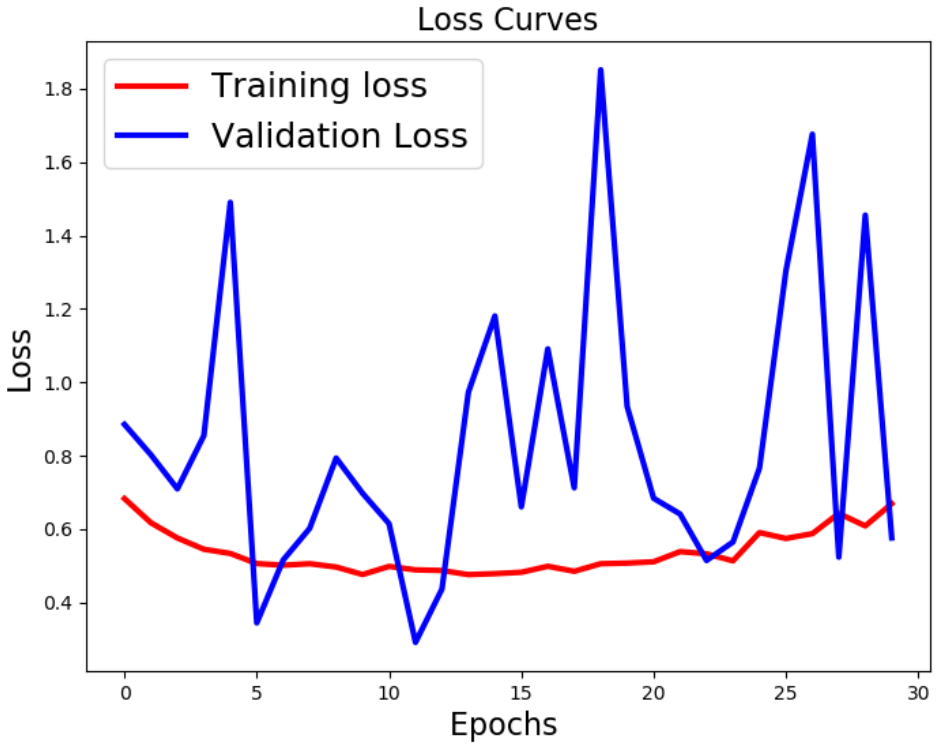


Figure 4.9: Training & Validation Loss Curves for 30 epochs

According to figure 4.9 training loss is 0.6692 after 30 epochs where validation loss 0.5760. Figure 4.10 shows the training and validation accuracy curves after training the model training for 30 epochs. As per the figure 4.10 training accuracy is 0.6860 and validation accuracy is 0.6250.

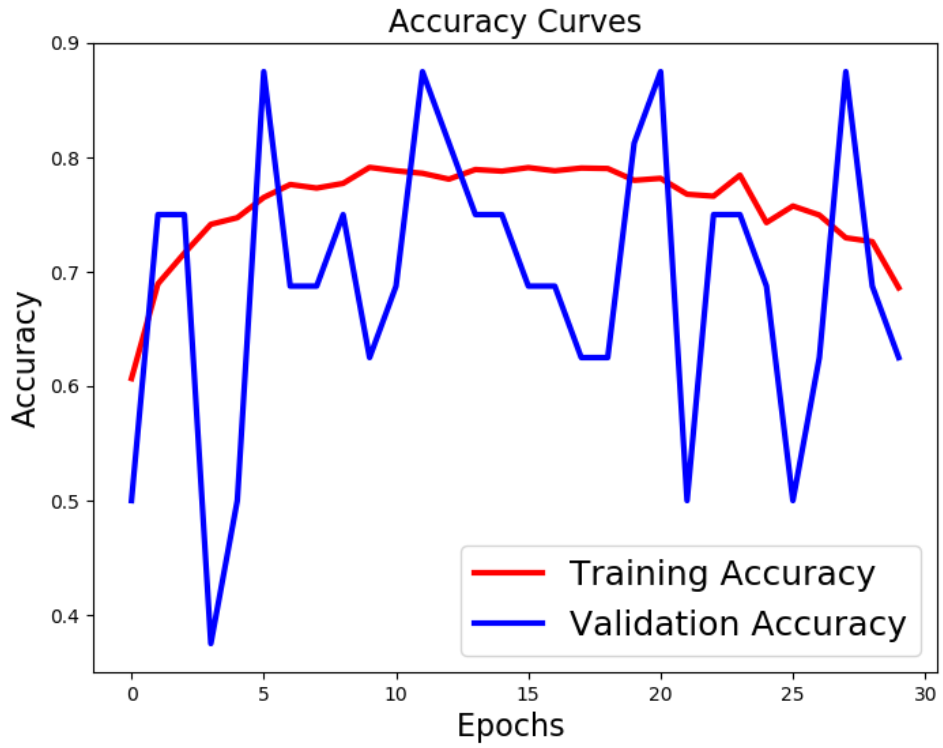


Figure 4.10: Training & Validation Accuracy Curves for 30 epochs

Summarizing the results of the above model trainings into one Figure 4.11 we can see that the model shows the highest accuracy after 20 epochs. After that it decreases the accuracy.

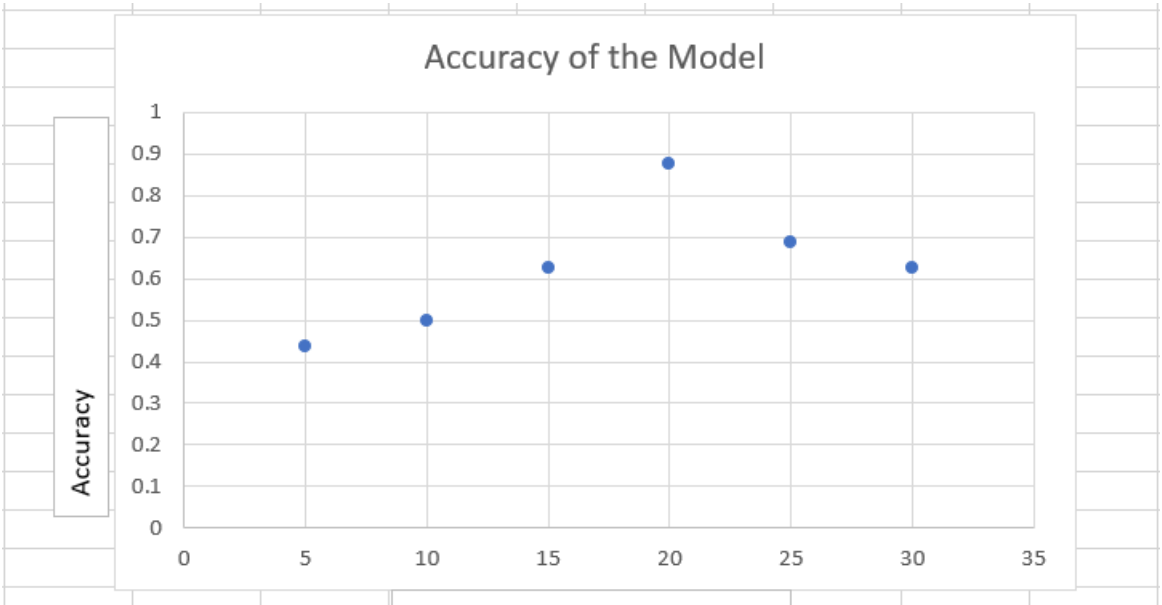


Figure 4.11: Summary of accuracy of training epochs

After every training cycle the results of the model is saved in .h5 format file. Those files are used to make new prediction for a given image. This module is capable of predicting three hashtags for a user given image.

Chapter 5
Conclusion & Future Work

5.1 Introduction

This chapter summarizes the whole study and discusses the conclusions made through the study and finally shows possible directions the study can take in the future.

5.2 Conclusion

This study is supposed to predict hashtags for a user given image considering the user contextual data such as age, gender and geographic location. In order to achieve the mentioned objectives a prediction model was designed. As per the design public Instagram posts along with the user data was collected via Netlytic system. Data was collected around 2018 October- December and two geo graphic locations New York and London was selected for data collection purpose due to the peak availability of data.

The collected data was carefully analyzed and manually selected for single object images and based on the data availability girl, boy, dog, cat categories were selected for this study. After that the selected data was preprocessed. In the preprocessing step Image data was augmented and standardized using data preprocessing scripts. Hashtag data also preprocessed by removing emojis, numeric characters, special characters, sentences and non-English tags.

After that the image data is feed to the CNN model and hashtag data along with the user contextual data was feed into the LSTM model. Output of the CNN model which is classified images were also an input to the LSTM model. CNN-LSTM model was trained for different number of epochs in order to identify the accurate prediction. Below Table 5.1 summarizes the model accuracy gained against different number of epochs.

Table 5.1: Accuracy of the model

Number of epochs	Accuracy of the Model
5	0.4375
10	0.5
15	0.625
20	0.875
25	0.6875
30	0.625

According to the results gain the best accuracy is recorded against 20 epochs which is 0.875. The training of this run is used to make new predictions.

5.3 Future Work

In this study we have implemented a model to predict hashtags for a given image according to user's age category, gender and geo location. Single object images are considered for the prediction. But users are posting images with more than one objects therefore as a future work we can consider the images contains more than one object.

Also, in this study we consider the age ranges of users due to limited availability of data. So as a next phase we can predict more user specific hashtags by taking the exact age as the input. Also, in this study we consider the base location as New York and London, as a future work we can improve the granularity of the location by taking exact locations of the image. Other than that, considering the timeseries nature of the data system can be extended to predict the seasonal tags for example Christmas, New Year.

Other than that, the study only considers the hashtags with English language, as a future work we can consider the prediction of hashtags with native languages. Because users tend to post tags with their native languages.

Proposed solution of this study can be further extended to predict hashtags for the video content posted in Instagram. Though we have taken the Instagram as the test bed, this solution can be extended to other major social networking platforms as well.

References

[1] D. Chaffey, "new-global-social-media-research," Smart Insights (Marketing Intelligence) Ltd, 28 Mar 2018. [Online]. Available: https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/. [Accessed 21 May 2018].

[2] "Top 20 Valuable Facebook Statistics," Zephoria Inc, July 2018. [Online]. Available: https://zephoria.com/top-15-valuable-facebook-statistics/. [Accessed 8 August 2018].

[3] R. Mathison, "22+ Useful Instagram Statistics for Social Media Marketers," Hootsuite, 24 January 2018. [Online]. Available: https://blog.hootsuite.com/instagram-statistics/. [Accessed 8 August 2018].

[4] V. Doctor, "What Characters Can A Hashtag Include?," LOGIKA Copearion, 12 June 2012. [Online]. Available: https://www.hashtags.org/featured/what-characters-can-a-hashtag-include/. [Accessed 14 August 2018].

[5] "Hashtag," MediaWiki, 08 August 2018. [Online]. Available: https://en.wikipedia.org/wiki/Hashtag. [Accessed 14 August 2018].

[6] "What is hashtag," mashable, [Online]. Available: https://mashable.com/2013/10/08/what-is-hashtag/.

[7] B. Barbee, "So What is a hashtag," Designtechnica Corporation, 18 February 2018. [Online]. Available: https://www.digitaltrends.com/social-media/what-is-a-hashtag/. [Accessed 14 August 2018].

[8] J. Van Grove, "Instagram Introduces Hashtags for Users & Brands," Mashable, 27 January 2011. [Online]. Available: https://mashable.com/2011/01/27/instagram-hashtags/#HyF_0ORDekqk. [Accessed 15 August 2018].

[9] K. Boogaard, "Instagram Hashtags: What You Need to Know," 12 January 2018. [Online]. Available: https://schedugr.am/blog/everything-to-know-instagram-hashtags/. [Accessed 15 August 2018].

[10] "How do I use hashtags?," Instagram Inc, 2018. [Online]. Available: https://help.instagram.com/351460621611097. [Accessed 15 August 2018].

[11] A. York, "How to Not Suck at Instagram Hashtags in 2018," Sprout Social, Inc., 15 May 2018. [Online]. Available: https://sproutsocial.com/insights/hashtags-for-instagram/. [Accessed 15 August 2018].

[12] "Use #Hashtags on Facebook," Facebook, 21 March 2016. [Online]. Available: https://www.facebook.com/facebookmedia/blog/using-hashtags-on-facebook. [Accessed 15 August 2018].

[13] "How do I use hashtags?," Facebook, 2018. [Online]. Available: https://www.facebook.com/help/587836257914341?helpref=faq_content. [Accessed 15 August 2018].

[14] S. Ayers, "Tips from 13 Experts on How to Use Hashtags on Facebook," Post Planner,Inc , [Online]. Available: https://www.postplanner.com/how-to-use-hashtags-on-facebook/. [Accessed 15 August 2018].

[15] "Machine Learning," MediaWiki, [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning. [Accessed 15 Spetember 2018].

[16] "Machine Learning," The MathWorks, Inc., [Online]. Available: https://www.mathworks.com/discovery/machine-learning.html. [Accessed 15 September 2018].

[17] "Deep Learning," MediaWiki, [Online]. Available: https://en.wikipedia.org/wiki/Deep_learning. [Accessed 12 September 2018].

[18] "Deep Learning," The MathWorks, Inc., [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html. [Accessed 14 September 2018].

[19] "Convolutional Neural Network," MediaWiki, [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network. [Accessed 15 September 2018].

[20] "Convolutional Neural Network," The MathWorks, Inc., [Online]. Available: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html. [Accessed 15 September 2018].

[21] "Convolutional Networks," [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed 15 September 2018].

[22] "Convolutional Neural Netwworks," The MathWorks, Inc, [Online]. Available: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html. [Accessed 15 September 2018].

[23] A. Escontrela, "Convolutional Neural Networks from the ground up," Towards Data Science, 17 June 2018. [Online]. Available: https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1. [Accessed 15 September 2018].

[24] P. SRIVASTAVA, "Essentials of Deep Learning : Introduction to Long Short Term Memory," Analytics Vidhya, 10 December 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/. [Accessed 15 September 2018].

[25] "Long Short Term Memory (LSTM)," The MathWorks, Inc., [Online]. Available: https://www.mathworks.com/discovery/lstm.html?s_tid=srchtitle. [Accessed 15 September 2018].

[26] "Long Short Term Memory," MediaWiki, [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory. [Accessed 15 September 2018].

[27] Xu, Can;Cetintas Suleyman; Lee, Kuang-Chih; Li,Li-Jia, "Visual Sentiment Prediction with Deep Convolutional Neural Networks".

[28] Y. Jia, " Caffe: An open source convolutional architecture," 2014.

[29] Veit, Andreas;Nickel, Maximilian; Belongie,Serge; Maaten,Laurens van der, "Separating Self-Expression and Visual Content in Hashtag Supervision".

[30] Weston,Jason;Chopra,Sumit; Adams,Keith, "#TAGSPACE: Semantic Embeddings from Hashtags," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014.

[31] Denton,Emily;Weston,Jason;Paluri,Manohar;Bourdev,Lubomir;Fergus,Rob, "User Conditional Hashtag Prediction for Images," ACM, Sydney, NSW, Australia, 2015.

[32] Donahue,Jeff; Hendricks,Lisa Anne;Rohrbach,Marcus;Venugopalan, Subhashini;Guadarrama,Sergio;Saenko,Kate;Darrell,Trevor , "Long-term Recurrent Convolutional Networks for Visual Recognition and Description," 2016.

[33] Li,Jia;Xu,Hua;He,Xingwei;Deng,Junhui;Sun,Xiaomin, "Tweet Modeling with LSTM Recurrent Neural Networks for Hashtag Recommendation," in *International Joint Conference on Neural Networks (IJCNN)*, 2016.

[34] Li,Yang;Liu,Ting; Jiang,Jing;Zhang,Liang, "Hashtag Recommendation with Topical Attention-Based LSTM," in *COLING 2016, the 26th International Conference on Computational Linguistics*, Osaka, 2016.

[35] "Tensorflow," Tensorflow, [Online]. Available: https://www.tensorflow.org/. [Accessed 15 September 2018].

[36] "Keras," MediaWiki, [Online]. Available: https://en.wikipedia.org/wiki/Keras. [Accessed 15 September 2018].

[37] "Keras: The Python Deep Learning library," [Online]. Available: https://keras.io/. [Accessed 15 September 2018].

[38] "Caffe(Software)," MediaWiki, [Online]. Available: https://en.wikipedia.org/wiki/Caffe_(software). [Accessed 15 September 2018].

[39] "Caffe," [Online]. Available: http://caffe.berkeleyvision.org/. [Accessed 15 September 2018].

[40] "Pytorch," MediaWiki, [Online]. Available: https://en.wikipedia.org/wiki/PyTorch. [Accessed 15 September 2018].

[41] "About," [Online]. Available: https://pytorch.org/about/. [Accessed 15 September 2018].

[42] S. Polamuri, "DIFFERENCE BETWEEN SOFTMAX FUNCTION AND SIGMOID FUNCTION," dataaspirant.com, 7 March 2017. [Online]. Available: http://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/. [Accessed 24 Feb 2019].

[43] "pandas," [Online]. Available: https://pandas.pydata.org/. [Accessed 28 Feb 2019].

[44] "scikit-learn," INRIA, [Online]. Available: https://scikit-learn.org/stable/. [Accessed 28 Feb 2019].

[45] "PyCharm," JetBrains, [Online]. Available: https://www.jetbrains.com/pycharm/. [Accessed 28 Feb 2019].

[46] J. Hunter, D. Dale, E. Firing, M. Droettboom and M. d. team, "matplotlib," [Online]. Available: https://matplotlib.org/. [Accessed 20 Jan 2019].

[47] S. Ray, "6 Easy Steps to Learn Naive Bayes Algorithm (with codes in Python and R)," Analytics Vidhya., 11 September 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. [Accessed 15 November 2018].

Appendix A

Code examples

CNN Model

```
# MODEL
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(300, 300, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# the model so far outputs 3D feature maps (height, width, features)

model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(2))
model.add(Activation('softmax'))
# COMPILE
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
model.summary()
batch_size = 16

# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1./255)

# this is a generator that will read pictures found in subfolers of 'data/train', and indefinitely generate
# batches of augmented image data
train_generator = train_datagen.flow_from_directory(
        'training_set',  # this is the target directory
        target_size=(300, 300),  # all images will be resized to 300x300
        batch_size=batch_size,
        class_mode='categorical')  # since we use binary_crossentropy loss, we need binary labels

# this is a similar generator, for validation data
validation_generator = test_datagen.flow_from_directory(
        'test_set',
        target_size=(300, 300),
        batch_size=batch_size,
        class_mode='categorical')

# TRAINING
history=model.fit_generator(
        train_generator,
        steps_per_epoch=10000 // batch_size,
        epochs=20,
        validation_data=validation_generator,
        validation_steps=2000 // batch_size)

model.save('n20_epochs.h5')  # always save your weights after training or during training
```

LSTM Model

```
# Split the data to train and test sets:

categories = ['girl', 'dog', 'boy', 'cat',  'newyork', 'london', '18-25','25-30','30-35']
# categories = ['girl', 'dog']
train, test = train_test_split(df, random_state=42, test_size=0.33)
X_train = train.stars
X_test = test.stars
print(X_train.shape)
# print(X_test.shape)
#
print(X_train)
# print('============================================================')
# print(X_test)

# Create sequence
vocabulary_size = 20000
tokenizer = Tokenizer(num_words= vocabulary_size)
tokenizer.fit_on_texts(X_train)
sequences = tokenizer.texts_to_sequences(X_train)
train_data = pad_sequences(sequences)

print(train_data)

tokenizer.fit_on_texts(X_test)
sequences = tokenizer.texts_to_sequences(X_test)
test_data = pad_sequences(sequences)

# print(test_data)

## Network architecture
model = Sequential()
model.add(Embedding(20000, 100, input_length=22))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

word_embds = model.layers[0].get_weights()
# print(word_embds)

for category in categories:
    print('... Processing {}'.format(category))
    # print(train_data)
    # train the model using X_dtm & y
    model.fit(train_data, train[category], validation_split=0.4, epochs=50)
    # compute the testing accuracy
    prediction = model.predict(test_data)
    print(prediction)
    fileName=category+'.h5'
    model.save(fileName)
model = Sequential()
model.add(Embedding(total_words, 10, input_length=max_sequence_len-1))
model.add(LSTM(100, return_sequences = True))
model.add(LSTM(100, return_sequences = True))
model.add(LSTM(100))
model.add(Dropout(0.4))
model.add(Dense(total_words, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
earlystop = EarlyStopping(monitor='val_loss', min_delta=0, patience=5, verbose=0, mode='auto')
model.fit(predictors, label, epochs=100, verbose=1, callbacks=[earlystop])
```