VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum-590018, Karnataka

A project phase-I report on

"AUTOMATED IMAGE CAPTIONING USING MACHINE LEARNING"

Submitted in fulfilment for the requirements of VII semester degree of

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

by

ANOOPKUMAR KULKARNI (1DB17IS008)
SANJAY Y V (1DB16IS043)

Under the Guidance of

Mr. PUNEETH KUMAR P
ASSISTANT PROFESSOR
DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

Department of Information Science and Engineering

DON BOSCO INSTITUE OF TECHNOLOGY

Kumbalagodu, Mysore road, Bangalore-560074

2020-2021

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Bengaluru – 560 074.

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

<u>CERTIFICATE</u>

This is to certify that the project report entitled **"AUTOMATED IMAGE CAPTIONING USING MACHINE LEARNING"** is a bonafide work carried out by **ANOOPKUMAR KULKARNI(1DB17IS008) and SANJAY Y V(1DB16IS043),** in partial fulfilment of award of Degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi, during the academic year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The project has been approved as it satisfies the academic requirements associated with the degree mentioned.

Signature of Guide	**Signature of HOD**
___________________	___________________________
Mr. Puneeth Kumar P	**Mrs. Gowramma G. S**
Assistant Professor,	**Assoc Prof, Head of Dept,**
Dept. of ISE,	**Dept. of ISE,**
DBIT, Bengaluru.	**DBIT, Bengaluru.**

External Viva

Name of the Examiners	**Signature with Date**
1. _________________	_________________
2. _________________	_________________

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Bengaluru – 560 074.

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

DECLARATION

We **ANOOPKUMAR KULKARNI(1DB17IS008)** and **SANJAY Y V(1DB16IS043)**, students of seventh semester B.E, Department of Information Science and Engineering, Don Bosco Institute of Technology, Kumbalagodu, Bangalore, declare that the project work entitled **"AUTOMATED IMAGE CAPTIONING USING MACHINE LEARNING"** has been carried out by us and submitted in partial fulfilment of the course requirements for the award of degree in **Bachelor of Engineering** in **Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during the academic year **2020-2021**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bangalore **ANOOPKUMAR KULKARNI(1DB17IS008)**
Date: **SANJAY Y V(1DB16IS043)**

ACKNOWLEDGEMENT

Here by we are submitting the project phase-1 report on **"AUTOMATED IMAGE CAPTIONING USING MACHINE LEARNING"**, as per the scheme of Visvesvaraya Technological University, Belagavi.

In this connection, we would like to express our deep sense of gratitude to our beloved institution Don Bosco Institute of Technology and also, we like to express our sincere gratitude and indebtedness to **Dr. Hemadri Naidu T,** Principal, DBIT, Bengaluru.

We would like to express our sincere gratitude to **Mrs. Gowramma G. S,** Assoc Prof, Head of Dept, Department of Information Science and Engineering, DBIT, for providing a congenial environment to work in and carry out our seminar.

We would like to express the deepest sense of gratitude to thank our Project Guide **Mr. Puneeth Kumar P,** Assistant Professor, Department of Information Science and Engineering, DBIT, Bengaluru for his constant help and support extended towards us during the project.

Finally, we are very much thankful to all the teaching and non-teaching members of the Department of Information Science and Engineering for their constant encouragement, support and help throughout the completion of report.

ANOOPKUMAR KULKARNI (1DB17IS017)

SANJAY Y V(1DB16IS043)

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Captioning images automatically is one of the heart of the human visual system. There are various advantages if there is an application which automatically caption the scenes surrounded by them and revert back the caption as a plain message. In this paper, we present a model based on CNN-LSTM neural networks which automatically detects the objects in the images and generates descriptions for the images. It uses various pre-trained models to perform the task of detecting objects and uses CNN and LSTM to generate the captions. It uses Transfer Learning based pre-trained models for the task of object Detection. This model can perform two operations. The first one is to detect objects in the image using Convolutional Neural Networks and the other is to caption the images using RNN based LSTM(Long Short Term Memory). Interface of the model is developed using flask rest API, which is a web development framework of python. The main use case of this project is to help visually impaired to understand the surrounding environment and act according to that.

Image caption generation has emerged as a challenging and important research area generation of captions from images has various practical benefits, ranging from aiding the visually impaired, to enabling the automatic and cost-saving labelling of the millions of images uploaded to the Internet every day. The field also brings together state-of-the-art models in Natural Language Processing and Computer Vision, two of the major fields in Artificial Intelligence.

Caption generation is one of the interesting and focussed areas of Artificial Intelligence which has many challenges to pass on. Caption generation involves various complex scenarios starting from picking the dataset, training the model, validating the model, creating pre-trained models to test the images, detecting the images and finally generating the captions.

Chapter 2

LITERATURE SURVEY

This is the systematic analysis work published and unpublished on data from secondary sources in the field of special interest to the investigator. It is important for gathering the secondary data for the research which might be proved very helpful in the research. There are many explanations for conducting the literature survey. The analysis of the literature may be in any field of the market. The few cited articles are below.

2.1 Data-Driven Image Captioning via Salient Region Discovery

Given an image, these models obtain a variety of appropriate images and their annotations from a big array of pictures where the explanation is generated. This paper a defines a two-step process that includes an early extraction of global level relevant pictures and a successive picture analysis to choose the much more appropriate picture. This last step is based on running detectors; these detectors' replies are used to re-rank the pictures so that the description of the more similar picture is chosen as the description of the input photo. This is a new method for selecting more appropriate picture from a big collection by integrating feature vectors with a portrayal of an object-specific picture. In particular, we initially try comparing an image data to a huge sequence of pictures, by using state-of-the-art feature vectors that encrypt the qualities of the picture and the thing. We further enhance this early restoration by re-ranking the pictures in consideration of a new symbolic image, called Bag of-Object-Proposals, that also encrypts attribute pictures. This enhances the effectiveness of the relevance of the extracted features, so we can create more precise definitions from here.

2.2 Multimodal Neural Language Models

It is possible for an image text multimodal neural language system to retrieve pictures from the provided compound sentence questions. To extract image query phrase descriptions and produce image-conditioned text. This indicates that in the area of image text modelling, it can study word representation and image features collectively by training the system together with a convolutional network.

While focusing methodologies can easily be extended to other methods such as voice on image text modelling. In this section they introduced integrative models of human language; natural language models can be influenced on certain methods. They introduced two techniques on the log bilinear model: the bilinear log model that is based on modalities and the three-way database bilinear model based on the components. Then showed the way for studying word depiction and picture characteristics along with training the language systems with a convolution network. Trial is conducted on three picture-text descriptive datasets: IAPR TC-12, Discovery attributes, and SBU datasets. Even more demonstrate models of abilities through quantitative assessment of retrievals and outcome visualization tools.

2.3 Every Picture Tells A Story: Generating Sentences from Images

This paper investigates tools for creating small description for the pictures which are uploaded to the system. To research this question, it is put into a dataset. They added an intermediate narrative depiction between the pictures and the sentences. It is an exclusionary approach which yields very positive results at the classification of sentences.

The model makes the assumption that there is a space of sense between the sentence fragments space and the picture set. This defines a sentence's resemblance to a picture by (a) projecting to the context area, then (b) analysing the performance. We must study the projection from photos to image features meaning from pairs of images and allocated interpretations of context. The current representation of meaning is a triplet of (object, action, and scene). This triplet provides a holistic idea about what the image is about and what is most important. For the picture, this is the part of which people will first speak, for the sentence; this is the structure that should be maintained in the narrowest description. Mapping takes place between picture area to interpretation area, description area to interpretation area. Retrieve sentences for images via meaning space. There will be more information about the image. With the help of mapping meaning and image are evaluated result is stored as triplets of (image, action, object) and score is evaluated by predicting image and sentence.

2.4 Convolutional Image Captioning

This section discusses about the system that depicts the human understandable language description of pictures and their different parts. Here they have used CNN as the input signal and recurrent neural network (RNN) as the decoder. Convolutional Neural Network, which extracts patterns and characteristics from the photo that is given to the input. These features are then fed to Recurrent Neural Network, which generates text from these features. We train our model using flickr30k and MSCOCO training datasets, which contains image and their corresponding text to study the correspondence between the data which can be seen and its natural language description.

2.5 Image captioning based on deep neural networks

In this study the author describes the three-picture description technique by the use of deep neural network. They are CNN-RNN Cantered System, CNN focused CNN platform and Reinforcement dependent structure. Instead show the collective job of these three best solutions, define the assessment criteria, outline the advantage, Large obstacles. Combining human speech and machine vision method has gained considerable interest over the past few years with the advancement of deep learning. Picture description reflect this field, which makes machine to study using one or more phrases to explain the picture's visuals. The significant method of producing high level image semi conception includes object and scene recognition, ability to evaluate position, components, and how the objects are related to each other. Although image captioning is a complex and important job, major improvements have been made by a group of researchers.

Chapter 3

SYSTEM ANALYSIS

3.1. System Study and Environment

In order to tackle the image captioning task, recent work shows it is in one's interest to utilize neural networks [7]. This frequently used term dates back to 1950s when notions such as the Perceptron Learning Algorithm were introduced [8]. Modern neural networks draw on notions discovered in the era of a Perceptron. In this section, we rst de ne a neuron as a fundamental part of modern neural networks. Then we elaborate on Convolutional Networks and Recurrent Networks.

3.1.1.Perceptron

For the purposes of this work, a perceptron is de ned generally as it became a funda-mental part of modern neural networks and the notation is utilized further on. Thus, a perceptron is compounded of one neuron. The neuron's output, known as the activation a, is mapped by :

R^N ! R as follows:

$$a = (x) = (w^T x + b) \qquad (1)$$

where x 2 R^N is a feature vector, w 2 R^N and b 2 R are weights and ( ) is a non-linear function. In case of the Perceptron, ( ) stands for

$$(z) = \begin{cases} 1 \text{ if } z > 0 \\ 0 \text{ otherwise} \end{cases} \qquad (2)$$

In other words, a perceptron is a non-linear function separating data into two classes each associated with either 1 or 0. A perceptron is parametrized by weights w and b. By setting proper weights, one e ects the output and the perceptron's behaviour for a given feature vector. Therefore, such weights trimming is essential, yet non-trivial task. In order to nd the weights, a learning algorithm was introduced, named the Perceptron Learning Algorithm [8]. This algorithm has a limiting property such that successful learning is achieved if and only if the data are linearly separable which is a major drawback pointed out in by Minsky and Paper in 1969 [9]. For example, there is no vector w and bias b that would make a perceptron mimic the XOR function

3.1.2. Multi-Layer Neural Network

Taking the perceptron as inspiration, the XOR problem can be overcome by aligning neurons into layers and interconnecting those layers. This function is called a Feedfor-ward Neural Network, an Arti cial Neural Network or simply a Neural Network.

In a neural network, each layer comprises N neurons processing inputs coming from the previous layer and producing activations used later in the following layer. For the sake of simplicity, it is now assumed that the number of neurons N is same for all layers. However, this varies very often, for example, usually the output layer consists of fewer neurons corresponding to the nature of the problem being solved. To conclude, the activations of the k-th layer $(a_1^{(k)}; a_2^{(k)}; : : : ; a_N^{(k)}) = a^{(k)}\ 2\ R^N$ are each a function of the activations of the previous layer k=1, noted as $a^{(k\ \ 1)}\ 2\ R^N$:

$$\begin{array}{c} a_1^{(k)} = {}_1^{(k)}(a^{(k\ \ 1)}) \\ a_2^{(k)} = {}_2^{(k)}(a^{(k\ \ 1)}) \\ . \\ . \quad\quad . \end{array} \tag{3}$$

where ${}_i^{(k)}(\ )$ is a function de ning the properties of the i-th neuron in the layer k, often having the form similar to Eq. (1). However, there are exceptions that proved to be essential to modern neural networks designs [2, 7, 10{12]. We discuss these in the later sections.

In practise, to lower the complexity of a network, in the given layer k all the functions ${}_i^{(k)}(\ )$ are always of the same form, only distinct in weights. Therefore, it is convenient to use vector notation. This is done by simplifying Eq. (3) into the following form:

$$a^{(k)} = {}^{(k)}(a^{(k\ \ 1)}) \tag{4}$$

As an example, we show a neural network with one hidden layer. The network takes in a vector that is propagated forward into the hidden layer. The processes in the hidden layer are noted here as ${}^{(1)}$.

Further, the activations of the hidden layer are again propagated, analogically, into the second layer $^{(2)}$ whose activations are the output of the network. The network's design is shown in Fig. 3. Formally, the network is fed with a feature vector x 2 R^{N} producing a vector y 2 R^{M} :

$$y = \ (x) = \ ^{(2)}(\ ^{(1)}(x)) \qquad (5)$$

where ^{1} : R^{N} ! R^{H} is called the hidden layer and ^{2} : R^{H} ! R^{M} is called the output layer. Note that the number of neurons in the hidden layer H is a hyper-parameter.

Although the structure of the network in the example has been de ned, there are still other hyper-parameters to be determined. For example, the form of the layer mappings $^{(1)}$ and $^{(2)}$ needs to be speci ed. A layer with the most simple form of its mapping is called a Fully Connected Layer and is discussed in the following subsection.

3.1.3.Fully Connected Layer

In the most basic neural network | a feedforward neural network comprising fully-connected layers only { each neuron processes activations of all neurons in the previous layer and is activated using ( ) de ned in Eq. (1). Thus, based on vector notation in Eq. (4), the activations of the k-th fully-connected layer are de ned as

$$a(k) = (W(k \ \ 1;k)a(k \ \ 1) + b(k \ \ 1;k)) \qquad (6)$$

where W$^{(k\ 1;k)}$ 2 R$^{N\,N}$ is a weights matrix with weights vectors aligned in rows, b$^{(k\ 1;k)}$ 2 R^{N} is a vector of biases and : R^{N} ! R^{N} is a non-linear function. Note that, since in practise the elements $_{i}$( ) are identical single-variable functions, we refer to them as simply ( ).

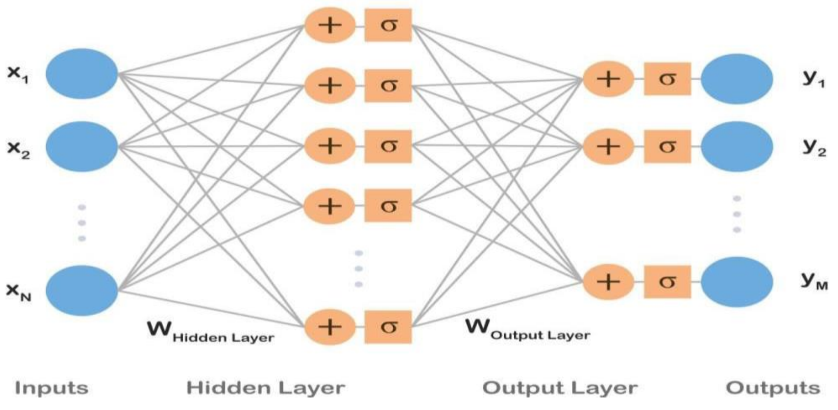


Figure 3.1.3: Fully Connected Layer

Inputs $x_1; : : : ; x_N$ are processed by a hidden layer and, consecutively, by an output layer producing outputs $y_1; : : : ; y_M$. Biases b were omitted for the sake of simplicity.

In contrast to a perceptron, ( ) is generally required to be di erentiable due to the nature of learning algorithms used in the eld. For example, ( ) used to be set to a sigmoid curve as similar to the perceptron's activation fuction shown in Eq. (2). Most commonly, tanh() or the logistic function (Eq. (7)) were used.

$$(z) = \underline{\hspace{3em}}\ 1 + e^{z} \qquad (7)$$

Nevertheless, when used in deep learning sigmoids su er from problems such as vanishing or exploding gradients, therefore those were replaced with a Recti ed Linear Unit (ReLU) [13]:

$$(z) = \max(0; z) \qquad (8)$$

In modern networks, it is recommended to use ReLUs as they proved to provide better results and are thus the most common activation function used nowadays [7].

Drawing on the example presented above, we now assume that both layers are fully connected, meaning that layer mappings have a form of Eq. (6). Then Eq. (5) can be rewritten as follows:

$$y = (W(1;2)\ \ (W(0;1)x + b(0;1)) + b(1;2)) \qquad (9)$$

3.1.4.Number of Parameters

Let us now assume x 2 R^N , the activations of the hidden layer a 2 R^H and y 2 R^M . Then we can calculate the number of parameters as N +N H+H+H M. Considering a small gray-scale image, 28 28, of a hand-written number taken from the MNIST dataset , that is classi ed as 0-9 digit, N = 784 and M = 10. Then the number of parameters, needed to be learned, is 795 H + 784 where H, the number of hidden layers, is a hyper-parameter. For a hidden layer having the same width as the input vector, i.e. H = 28 in this example, the number of parameters reaches 23; 044.

Truly, this is a large number for such a shallow network suggesting that fully connected layers extensively increase the number of parameters.

3.1.5.Hornik's Theorem

The network mentioned above is a common design that in past was believed to yield promising results. It was shown by Hornik [15], a multilayer feedforward neural network is able to approximate any continuous function that is bounded. Yet, a possibly great number of hidden neurons might be needed in order to do so because the theorem does not quantify this important hyper-parameter.

3.1.6.Name Origin

As shown in Fig. 3, the structure is called a network since it can be drawn as a directed acyclic graph. Wondering about the name's background, one may notice the word neural and mistakenly assume a relation to biological neurons. However, as stated for example in [7], it is a common misconception as the name, neural networks, was derived in 1950s from former biological models serving as motivation. Those models are now, however, considered outdated, and conversely, modern neural networks are not designed to be realistic models, rather going beyond neuroscience perspective. Since that, endeavours to infer an algorithm from the brain's functioning have not faded. For example, Je Hawkins et al. developed Hierarchical Temporal Memory (HTM) [16] as a strict mathematization of human neocortex based on current neuroscience. Internally, HTM as a biologically inspired algorithm is distinct from deep learning and, admittedly, its results have not been as fruitful as those obtained by deep nets [17], which are discussed int the following section.

3.2.Deep Learning

In spite of former beliefs, it was found that [7] it is more e cient to insert several hidden layers one by one and propagate information sequentially creating a deep structure, instead of utilizing a shallow network given in the example. This concept is called deep learning and, surprisingly, has its roots already in the pioneer 1960s as it was assumed that an intelligent algorithm solving complex problems shall work with hierarchy of concepts [7] that was rather deep. This is why we get the name deep learning.

The notion was later found in the idea of modern neural networks which, as stated above, consist of numerous nested layers each extracting more abstract and complex features as information is propagated forward the network. Therefore, the modern neural networks and techniques used for learning them are usually nowadays referred to as deep learning.

Deep neural networks were introduced already in 1998 and the optimization algorithm (back-propagation) was known by then and used frequently [3]. Yet, the deep nets were found too complex to be trained. In their books, Ian Goodfellow etal. list those reason that enabled the boom of neural networks in 2012: rstly, more data were available as well, therefore, the deep nets have started to outperform other models. Secondly, deeper models require decent architectures both in software and hardware and those had become available. Then on, promising results enabled advent of neural networks, especially in their deep form. Models such as convolutional neural networks or recurrent nets are considered state-of-the art building blocks nowadays. Their detailed design is discussed in the following subsections.

3.3.Convolutional Neural Networks

In image analysis, many of recent advances in deep learning are built on the work of LeCun et al. who introduced a Convolutional Neural Network (CNN) which had a large impact on the eld. A CNN is a type of a neural network that is designed to process an image and represent it with a vector code. The architecture of CNN a draws on fully-connected neural networks. Similarly, a convolutional neural network is a compounded structure of several layers processing signals and propagating them forward.

However, in contrast to a vector activation in a fully-connected layer, activations in CNNs have a shape of three-dimensional tensors. Commonly, this output tensor is called a feature map. For instance, an input image of shape 3 W H is transformed by the rst convolutional layer into a feature map of shape C W ^{0} H^0, where C is the number of features. In other words, a convolutional layer transforms a volume into a volume.

A typical CNN consists of several convolutional layers and, at the top, fully connected layers that attention convolutional volumes into a vector output. In the eld's terminology, this vector code of an image is often called fc7 features as it used to be extracted from the seventh fully connected layer of AlexNet [2]. Even though AlexNet has already been outperformed by many and the state-of-the-art designs are different from AlexNet, the term maintained its popularity. Additionally, depending on a problem the network is supposed to solve, an additional layer, such as soft-max, can be added on top of fc7 features. A common design of a CNN is depicted.

3.3.1.Receptive Field

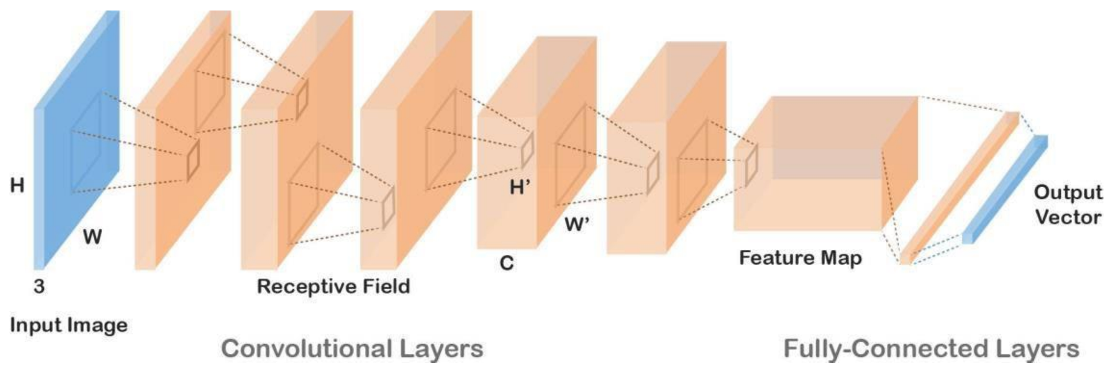


Figure 3.3.1: Fully Connected Layer

As mentioned above, a convolutional layer takes a tensor on input and produces a tensor, too. Note that these tensors have two spatial dimensions W and H, and one feature dimension C as they copy the form images are stored in. The context conveyed by the spatial dimensions is utilized in the CNN design which takes into account correlations in small areas of the input tensor called receptive elds. Concretely, in contrast to a neuron in a fully connected layer that processes all activations of the previous layer, a neuron in a convolutional layer "sees" only activations in its receptive eld. Instead of transforming layer's activations it restraints to a specific small rectangularly shaped subset of the activations. When mentioning a receptive eld, it is often expected only spatial dimensions of the input volume are referred to, i.e. a receptive elddenes an area in the W H grid. The shape of the receptive eld is a hyperparameter and varies across the models.

A convolutional neural network takes an image on input (in blue) and transforms it into a vector code (in blue). Convolutional Neural Networks are characteristic for processing volumes. An output of each layer is illustrated as an orange volume. Each neuron process only activations in the previous layer that belong to its receptive eld. The same set of weights is used for neurons across the whole grid. On top of convolutional layers, fully-connected layers are commonly connected.

3.3.2.Convolution in CNNs

A neuron's receptive eld is processed similarly to fully connected layer neurons. The values below the receptive eld along the input tensor's full depth are transformed by a non-linear function, typically ReLU (Eq. (8)).

However, in contrast to fully connected layer neurons, the same set of weights (referred to as a kernel) is used for all receptive elds in the input volume resulting into a transformation that has a form of convolution across the input. A kernel is convolved across W and H spatial dimensions. Then, a different kernel is again convolved across the input volume producing another 2D tensor. Aligning up the output tensors into a C W 0 H0 volume assembles the layer's output feature map.

This is an important property of convolutional neural networks because each kernel detects a specific feature in the input. For example, in the rst layer, the rst kernel would detect presence of horizontal lines in the receptive elds, the second kernel would look for vertical lines, and similarly further on. In fact, learning such types of detectors in the bottom layers is typical for CNNs.

The design of CNNs has an immensely practical implication { since a kernel is con-volved across the input utilizing the same set of weights and it covers only the receptive eld, the number of parameters is significantly reduced. Therefore, convolutional layers are less costly in terms of memory usage and the training time is shorter.

3.3.3.Pooling Layer

Convolutional layers are designed in such a way the spatial dimensions are preserved and the depth is increased along the network ow. However, it is practical to reduce spatial dimensions, especially in higher layers. Dimension’s reduction can be obtained by using stride when convolving, leading to dilution of receptive elds overlap. Nevertheless, a more straightforward technique was developed called a pooling layer. An input is partitioned into non-overlapping rectangles and the layer simply outputs a grid of maximum values of each rectangle. In practise, pooling layers are inserted often in between convolutional layers to reduce dimensionality.

3.4.Recurrent Neural Networks

Convolutional and fully connected layers are designed to process input in one time step without temporal context. Nonetheless, some tasks require concerning sequences where data are temporally interdependent. For that, a Recurrent Neural Network (RNN) { an extension of fully connected layers } has been introduced. RNNs are neural networks concerning information from previous time steps.

RNNs are used in a variety of tasks: transforming a static input into a sequence (e.g. image captioning); processing sequences into a static output (e.g. video labelling); or transforming sequences into sequences (e.g. automatic translation).

A simple recurrent network is typically designed by taking the layer's output from the previous step and concatenating it with the current step input:

$$y_t = f(x_t; y_{t\ 1}) \tag{10}$$

The function f is a standard fully-connected layer that processes both inputs indis-tinctly as one vector. Due to its simplicity, this approach is rather not su cient and does not yield promising results [19]. Thus, in past years, a great number of meaningful designs have been tested. The notion was advanced and designs have become more complex. For example, an inner state vector was introduced to convey information between times steps:

$$h_t; y_t = f(x_t; h_{t\ 1}) \tag{11}$$

The most popular architecture nowadays is a Long Short-Term Memory (LSTM) { a rather complex design, yet outperforming others.

3.4.1.Long Short-Term Memory

A standard LSTM layer is given as follows:

ft = g(Wf xt + Uf ht 1 + bf) (12)

it = g(Wixt + Uiht 1 + bi) (13)

ot = g(Woxt + Uoht 1 + bo) (14)

ct = ft ct 1 + it c(Wcxt + Ucht 1 + bc) (15)

ht = ot h(ct) (16)

where x_t is an input vector and $h_{t\ 1}$ is an output vector. All matrices W and U and biases b are weights that together, with $_g$ which is a logistic function Eq. (7), represent a standard neural network layer.

Thus, the forget gate vector f_t, the input gate vector i_t and output gate vector o_t are outputs of three distinct one-layer neural nets each having its output between 1 and 1. c_t is a cell state vector that, as a hidden output, is propagated to the next time.

step. ht is an output of the LSTM cell. stands for element-wise multiplication, c and h are usually set to tanh.

Note that ct is a combination of the previous time step ct 1, element-wisely adjusted by the forget gate ft, and the output of a neural network, gated similarly by the input gate it.

The output of LSTM ht is a function of the cell state vector, rst squashed between 0 and 1, and then adjusted by the output gate ot.

Connected to a network, LSTM consists typically of one layer only. LSTMs are known to preserve long-term dependencies, as shown for example by Karpathy.

3.5.Existing System

(RNN) in order to generate captions. In the last 5 years, a large number of articles have been published on image captioning with deep machine learning being popularly used. Deep learning algorithms can handle complexities and challenges of image captioning quite well. So far, only two survey papers [8, 13,] have been published on this research topic. Although the papers have presented a good literature survey of image captioning, they could only cover a few papers on deep learning because the bulk of them was published after the survey papers. These survey papers mainly discussed template based, retrieval based, and a very few deep learning-based novel image caption generating models. However, a large number of works have been done on deep learning-based image captioning. Moreover, the availability of large and new datasets has made the learning-based image captioning an interesting research area. To provide an abridged version of the literature, we present a survey mainly focusing on the deep learning-based papers on image captioning.

3.5.1.Disadvantages

The problem of image captioning is a complex and widely interested research topic since the evolution of deep learning. There are many proposed solutions for this problem which are replacing the previous solutions every single day. In [1] Karpathy proposed a system which uses multimodel neural networks to generate novel descriptions of the image by providing suitable descriptions for the image. In [2], Deng proposed a model which uses a database called ImageNet which is build using the core called WordNet. This model uses ImageNet to generate sentence descriptions from the image. Kelvin atel [3] proposed an attention based model, which generate captions of the images based on the region of interest. It generates the captions based on the region the image is surrounded. In [4] , Yang proposed a multimodal recurrent neural network based model, which generates the descriptions of the image by detecting the objects and converting them to sentences, . which is almost similar to human visual system In [5], Aneja proposed a convolutional neural network based modal to generate descriptions from the image after the rigorous training given to the model. In [6], Pan proposed a multiple neural network model, which is experimented with large sets of datasets to generate the accurate sentence descriptions from the image. In [7], Vinyals proposed a model that uses Natural Language Processing and Computer Vision to detect the objects in the image and generate captions based on word processing and keyword retrieval techniques.

3.6.Proposed System

Our model uses two different neural networks to generate the captions. The first neural network is Convolutional Neural Network(CNN), which is used to train the images as well as to detect the objects in the image with the help of various pre-trained models

like VGG, Inception or YOLO. The second neural network used is Recurrent Neural Network (RNN) based Long Short Term Memory(LSTM), which is used to generate captions from the generated object keywords.

As, there is lot of data involved to train and validate the model, generalized machine learning algorithms will not work. Deep Learning has been evolved from the recent times to solve the data constraints on Machine Learning algorithms. GPU based computing is required to perform the Deep Learning tasks more effectively.

3.6.1.Advantages

There are various advantages of Image captioning in multiple disciplines.

- It can be used for visually impaired people to understand the environment.
- It can be used in areas where text is more used and it can be used to infer text from images.
- Image captioning can also be used in self driving cars.
- It can be used by social networks to describe the image being uploaded by the user.
- It can be used in various NLP applications, where insights and summary is needed from the images.

3.7 System Requirements

The following are the software and hardware requirements:

3.7.1 Software Requirements

Language	:	Python 3.x
Database	:	Flickr8k
Operating System	:	OS Independent
IDE	:	Visual Studio Code
Front End	:	BOOTSTRAP, HTML and CSS

3.7.2 Hardware Requirements

Processor	:	Intel I3
Speed	:	1.6 Ghz
RAM	:	4GB (min)
Hard Disk	:	500 GB

3.7.3 Deployment Tools

Flask Rest API

Flask-Python

Chapter 4

SYSTEM DESIGN

4.1.System Architecture

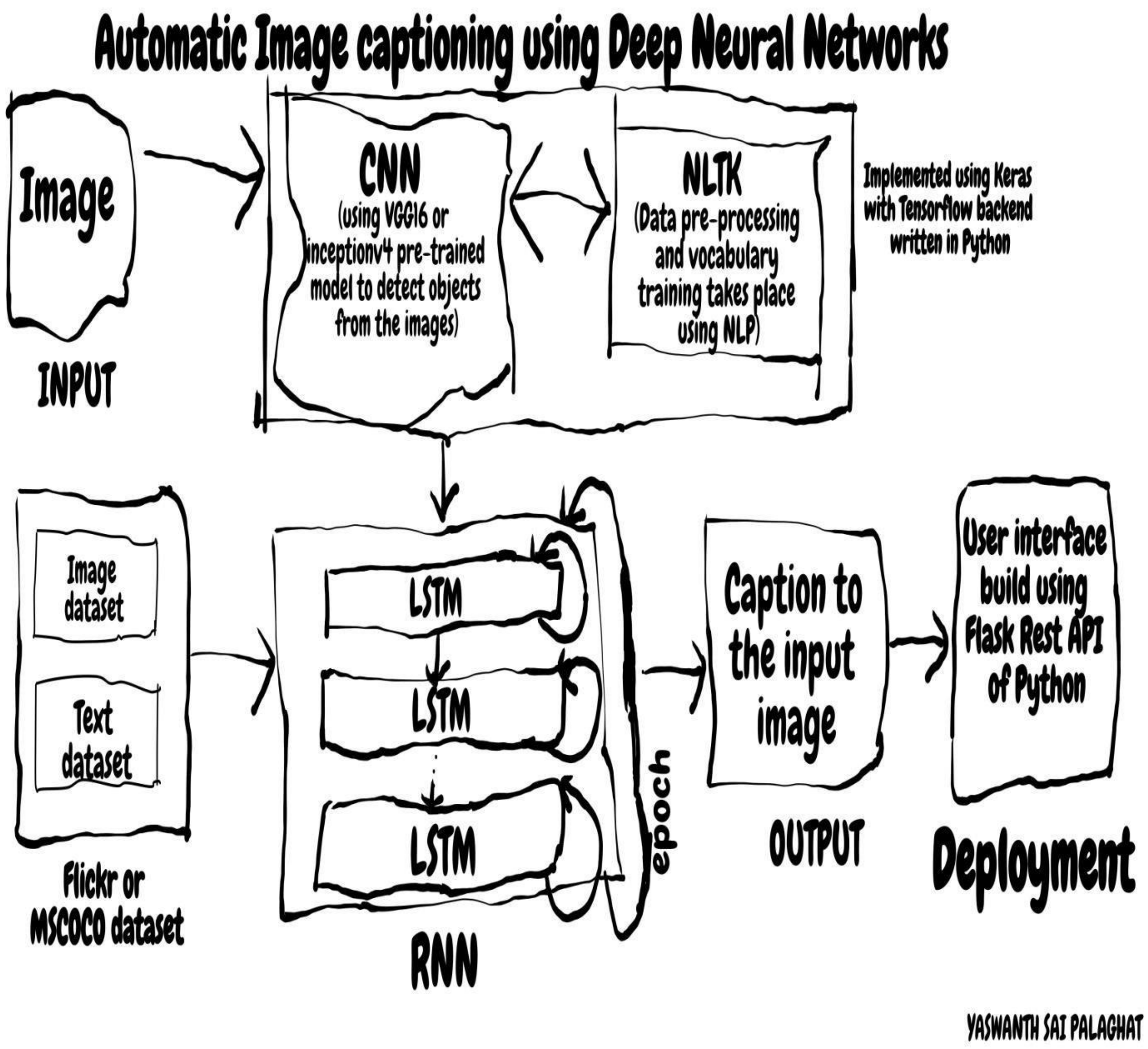

Automatic Image captioning using Deep Neural Networks
Image
INPUT
CNN
(using VGG16 or inceptionv4 pre-trained model to detect objects from the images)
NLTK
(Data pre-processing and vocabulary training takes place using NLP)
Implemented using Keras with Tensorflow backend written in Python
Image dataset
Text dataset
Flickr or MSCOCO dataset
LSTM
LSTM
LSTM
RNN
epoch
Caption to the input image
OUTPUT
User interface build using Flask Rest API of Python
Deployment
YASWANTH SAI PALAGHAT

Fig 5.1: System Architecture

4.2 Component Diagram:

An element diagram describes the organization of the physical elements in a very system. An element could be a physical building block of the system. it's pictured as a parallelogram with tabs.

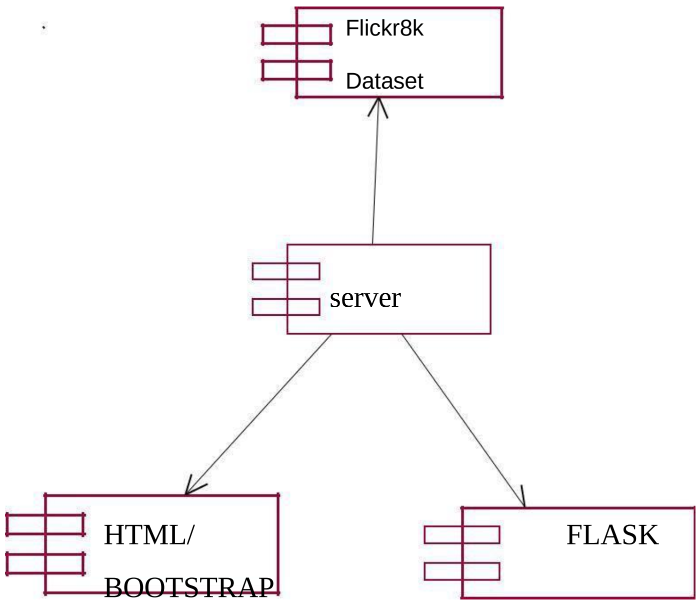


Fig 5.2.: Component Diagram

4.3 Deployment Diagram:

Deployment diagrams depict the physical resources in an exceedingly system as well as nodes, components, and connections. A node may be a physical resource that executes code parts.

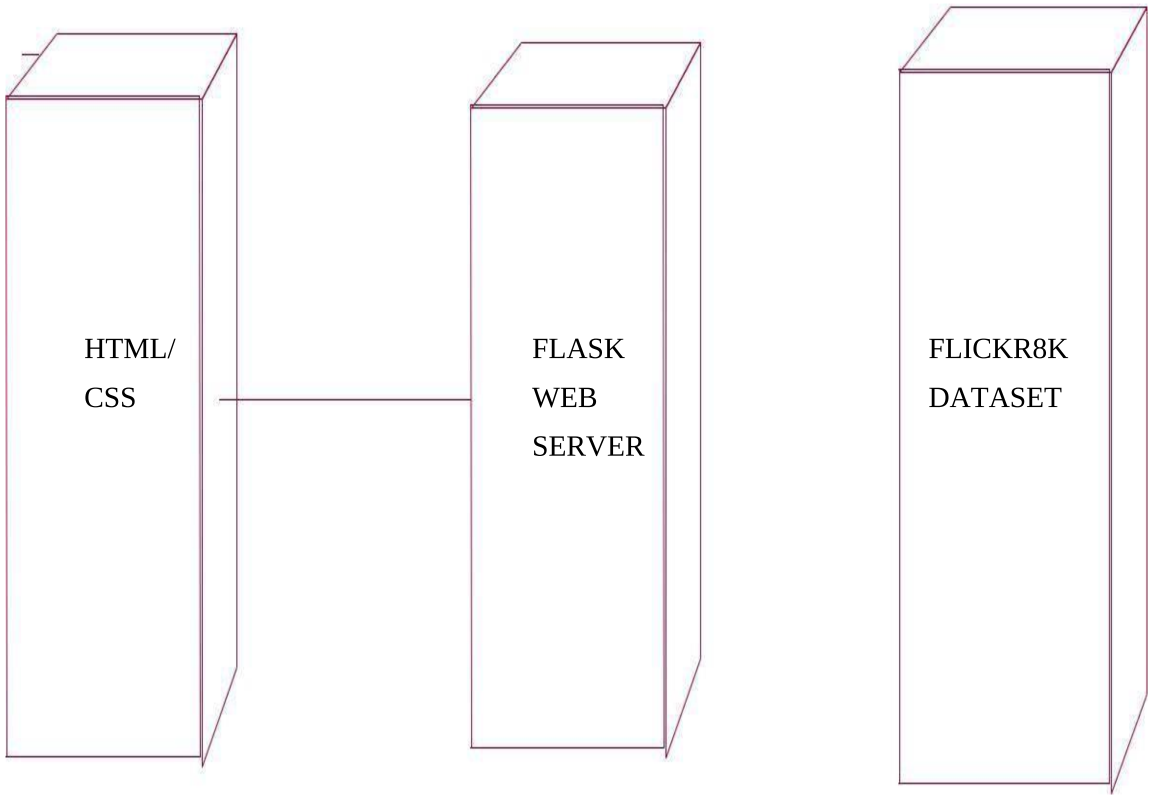


Fig 2.4: Deployment Diagram

Chapter 5

IMPLEMENTATION

Implementation of the system is the stage where the theoretical concept is transformed into functioning system. The new system could be completely unique, updating a current manual system or computer process, or it could be a hybrid one. significant improvements of an existing one. The system is implemented using the following tools.

5.1 Anaconda

It is a fair and democratic source supply of Python and R programming languages for Science computing aimed at simplifying the management and delivery of products. Product variants are maintained by the conda process management. The anaconda provides software for Windows, Linux, and macOS suitable for data-sciences. With 1500 modules picked from PyPI, as well as the conda bundle and virtual world manager, Anaconda distribution arrives. This also offers a GUI, Anaconda Navigator, as a visual replacement to the command line interface (CLI). Anaconda navigator is a desktop graphical user interface (GUI) included within the anaconda distribution that enables customers to run apps and control conda packages, setups, and platforms without ever using command line instructions. Navigator may scan anaconda cloud or local anaconda database for packages, mount them in a setting, run the modules, and upgrade them.

5.2 Jupyter Notebook

Jupyter Notebook was designed for showing one's development work simpler, and letting others enter in. jupyter notebook allows code, notes, graphics, and infographics to be incorporated in an immersive document called a notebook, which can obviously be exchanged replicated, and retooled. And since Jupyter Notebook runs through a search engine, it could launch the notebook itself on the local computer, or on a web server. Jupyter Notebook, initially intended in Python, R, and Juila for data analytics apps, is suitable for all types of installations in every way.

5.3 Python

Python is a Vocabulary of explanation, which is of high standard, and can be used for general activities. Python 's design philosophy, constructed by Guido van Rossum and first published in 1991, underlines code usability with its notable use of huge white space. The vocabulary constructs and object-oriented methods aim to encourage users to create simple, functional code for major and minor tasks. It is flexibly compiled, and is complicated. It allows various programming languages. Because of its enormous source file, occasionally it is described as the encourage users to create simple, functional code for major and minor tasks. it is flexibly compiled, paradigms, including object oriented structured (particularly procedural), and programming languages. Because of its enormous source file, occasionally it is described as the language "including batteries." In the late 1980s Python was created of as an ancestor of the ABC language. Authored in 2000, Python 2.0 introduced features such as list comprehensions and a garbage collection method worthy of gathering correct technique.

5.4 LSTM

Long Short-Term Memory Networks commonly referred to as LSTMs are special types of R NN, capable of knowing long-term dependence. LSTM networks are well appropriate for the classification, storage and estimation of data sets; as there may be legs of uncertain lengths in a time series between important events. The LSTM are specifically designed to resist the risk of long-term dependence. Their defect behaviour is basically to recall details for more amount of time. it has better accuracy than other types of RNN, for example vanilla RNN.

5.5 Pickle module

The pickle module introduces binary rules to serialize and de serialize a structure of the Python objects. “pickling” that was the method by which a Python object ideology is transformed as a byte stream and again transformed into an object ideology. Pickling which was defined as "serialization," "marshalling", "flattening;" but the words used here are “pickling” and "unpickling" to avoid ambiguity.

5.6 Adam

The Adam optimization algorithm is an extension of stochastic gradient descent, which has recently seen broader use in computer vision and natural language processing for deep learning applications. There are list of advantage using a dam when applying the algorithm on non-convex optimization problems as follows:

- Implementation is straight forward.
- Effective in computers.
- Diagonal rescale symmetric of the gradients.

- Small amount of storage is required.
- It suits for tasks that are big in terms of components and parameters.
- Suitable for dynamic targets.
- Suits for tasks having very disturbing curves.
- User friendly view point of hyper parameter required generally small adjustments.

5.7 Libraries Used

5.7.1 Numpy

Numpy works as an array processing program. It provides some of the services such as a multidimensional array object with high performance, tools for working with those arrays and an essential package which contain Python for scientific computing. It contains several features including these:

• Sophisticated functions that are broadcasted.
•There are tools for C / C++ and FORTRAN code integration.
•Linear algebra, Fourier transforms and capabilities of random numbers can be used.
• It provides an effective object for N-Dimensional Array.

Numpy is also used as an efficient multidimensional repository of generic information, in addition to its obvious scientific rules. Numpy contain arbitrary data types which allows Numpy to integrate with a wide variety of databases seamlessly and quickly.

5.7.2 Pandas

Pandas is one of the Python packages that gives simple, versatile and expressive data which is created to make it easy and intuitive to work with ‘relational’ or ‘labelled’ information. Pandas is the fundamental high-level building block in Python to do realistic and can-do real-world data analysis. In addition, it has the wider goal of being the most effective and versatile open-source data analysis/ manipulation tool in any language available. It’s already well on the way to the target.

Pandas works well on the various data types:

- For data in the SQL table or Excel spreadsheet which is in the Tabular format.
- Ordered and unordered time series data (not necessarily fixed frequency).

- Arbitrary matrix data with marks for rows and columns (heterogeneous or homogeneously typed). Any other form of observational / statistical data sets.
  In addition, the data do not need to be labelled to be put in a data structure for pandas.

5.7.3 Matplotlib

Matplotlib is most common Data Visualization Python packages used. It is used in making of 2D plots from data in arrays, it is a cross platform library. This provides an object-oriented API which uses Python GUI toolkits such as WxPythonotTkinter, PyQt which help to integrate plots in applications. It can be used in shells from Python, IPython, Jupyter notebook and servers for web applications. Matplotlib is created in Python language and uses NumPy, Python 's numerical math extension. Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Together with NumPy, Matplotlib can be the equivalent of open-source MATLAB.

5.7.4 OpenCV

OpenCV is a Python library which is designed to solve problems with computer vision.
OpenCV was initially developed by Intel in 1999 but later it was backed by Willow Garage.
OpenCV supports various programming languages including C++, Python, and Java etc.

- It works on various platforms including Windows, Linux, and MacOS.
- OpenCV Python is a wrapper package used with Python on the original C++ library.

- All structures of the OpenCV array are translated from / to NumPy arrays.
- Which make this easier for other libraries that use NumPy to incorporate.

5.7.5 Keras

Keras is an opensource library written in Python and it is a neural-network. It was created so that it can be user friendly, easy to extend, modular, and to work easily with Python. The main reasons to use Keras stem because of its guiding principles, which help to create user friendly interface. Keras has advantages of broad adoption, integration with at least five backend engines, support for a wide range of production deployment options, and strong support for multiple GPUs and distributed training.

5.7.6 TQDM

TQDM is a progress bar library with good nested loop support and notebook Jupyter/IPython. Nested progress bars are sponsored by TQDM. The nested loops should show properly. Besides this it has low overhead, TQDM uses intelligent algorithms to estimate the remaining time and skip the unnecessary iteration displays. It just needs Python and an environment supporting carriage return \r and line feed \n control characters. This in most cases allows for marginal overheads. TQDM runs on all the systems (Windows, Mac, Linux, FreeBSD, Solaris/SunOS, NetBSD), any console or GUI, and IPython / Jupyter notebooks are also nice.

5.7.7 TensorFlow

TensorFlow is a machine learning platform which is an end-to-end open source. It has a fully flexible ecosystem of tools, libraries and community resources which allows researchers to create new thing in ML and developers to build and deploy ML powerful applications easily. Simple model building: quickly create and train ML models using intuitive high-level APIs such as Keras with quick execution, allowing experimentation and quick debugging for the immediate model.

- Efficient development of ML anywhere: Easy to train and utilize models in the cloud, browser or app irrespective of the language you use.
- Strong research experimentation: a clear and scalable framework for bringing new concepts related to concept to code, state-of-the-art models and quicker release.

5.7.8 Resnet50

Resnet50 is a 50layer deep convolution neural network. A pre trained version of the trained network for more than one million images from the ImageNet database can be loaded. The pre-trained network classifies images into 1000 categories of objects such as keyboard, mouse, pencil and so on. Therefore, the network has learned rich representation of features for a large variety of images. The network has an input size 224-by-224.

5.8 Algorithm

5.8.1 Algorithm for CNN Model

Step 1: Entering a picture into the system.

Step2: Obtain region proposals (picture areas possibly containing objects) using a method like selective search.

Step3: Use transfer learning especially extraction function, to calculate characteristics using the pre trained model CNN for each proposal.

Step 4: Classification.

5.8.2 Algorithm for RNN Model

Step 1: Convert abstracts from list of strings into list of lists of integers.

Step 2: Create feature and labels from sequence

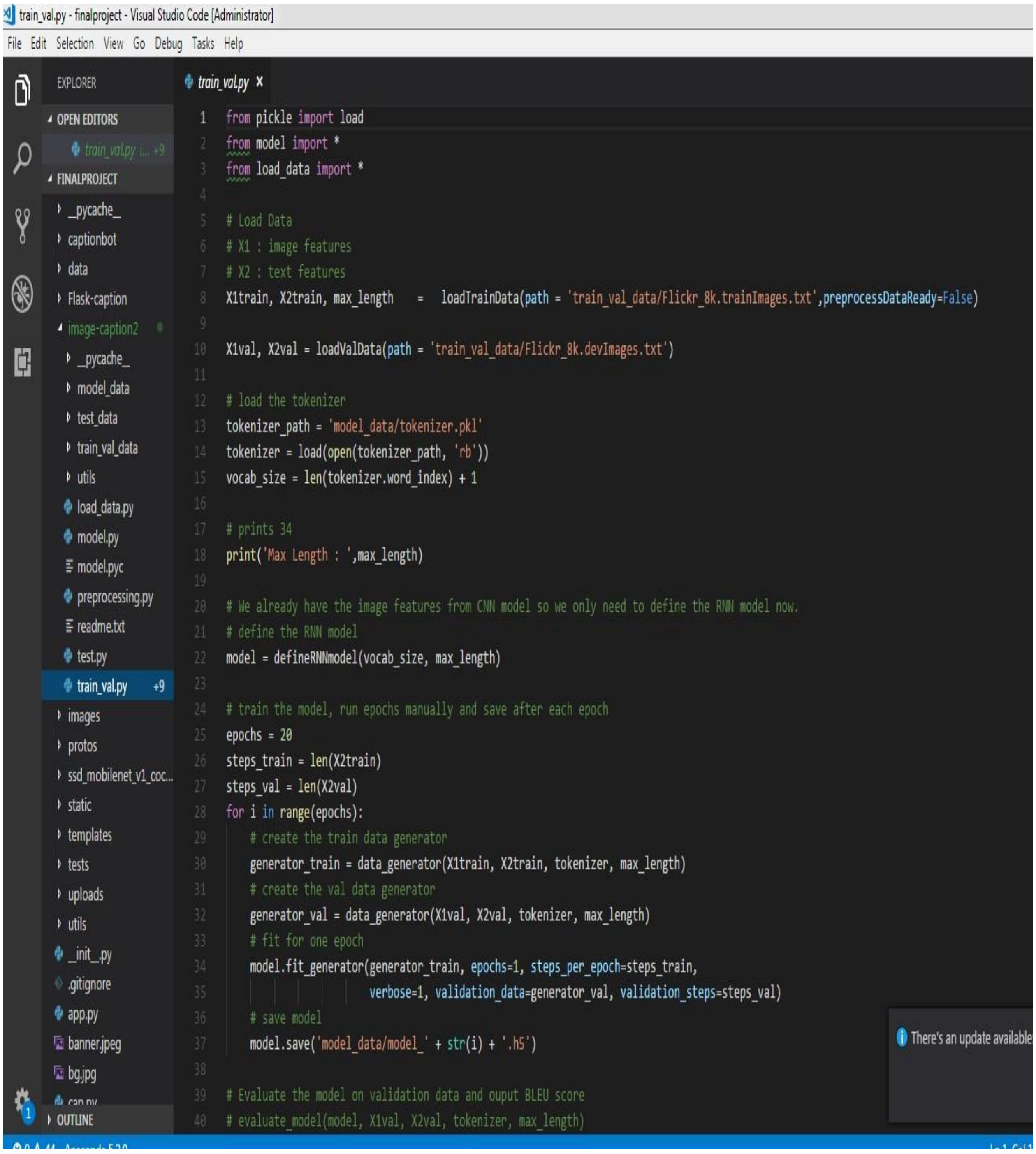
Step 3: Build LSTM model with embedding, LSTM and Dense Layers.

Step 4: Load in pre-trained embeddings.

Step 5: Train model to predict the next work in sequence.

Step 6: Make prediction by passing in starting sequence.

5.9 Snapshots:



```python
from pickle import load
from model import *
from load_data import *

# Load Data
# X1 : image features
# X2 : text features
X1train, X2train, max_length   =   loadTrainData(path = 'train_val_data/Flickr_8k.trainImages.txt',preprocessDataReady=False)

X1val, X2val = loadValData(path = 'train_val_data/Flickr_8k.devImages.txt')

# load the tokenizer
tokenizer_path = 'model_data/tokenizer.pkl'
tokenizer = load(open(tokenizer_path, 'rb'))
vocab_size = len(tokenizer.word_index) + 1

# prints 34
print('Max Length : ',max_length)

# We already have the image features from CNN model so we only need to define the RNN model now.
# define the RNN model
model = defineRNNmodel(vocab_size, max_length)

# train the model, run epochs manually and save after each epoch
epochs = 20
steps_train = len(X2train)
steps_val = len(X2val)
for i in range(epochs):
    # create the train data generator
    generator_train = data_generator(X1train, X2train, tokenizer, max_length)
    # create the val data generator
    generator_val = data_generator(X1val, X2val, tokenizer, max_length)
    # fit for one epoch
    model.fit_generator(generator_train, epochs=1, steps_per_epoch=steps_train,
                        verbose=1, validation_data=generator_val, validation_steps=steps_val)
    # save model
    model.save('model_data/model_' + str(i) + '.h5')

# Evaluate the model on validation data and ouput BLEU score
# evaluate_model(model, X1val, X2val, tokenizer, max_length)
```

Fig 5.9.1: Visual Studio Code Interface

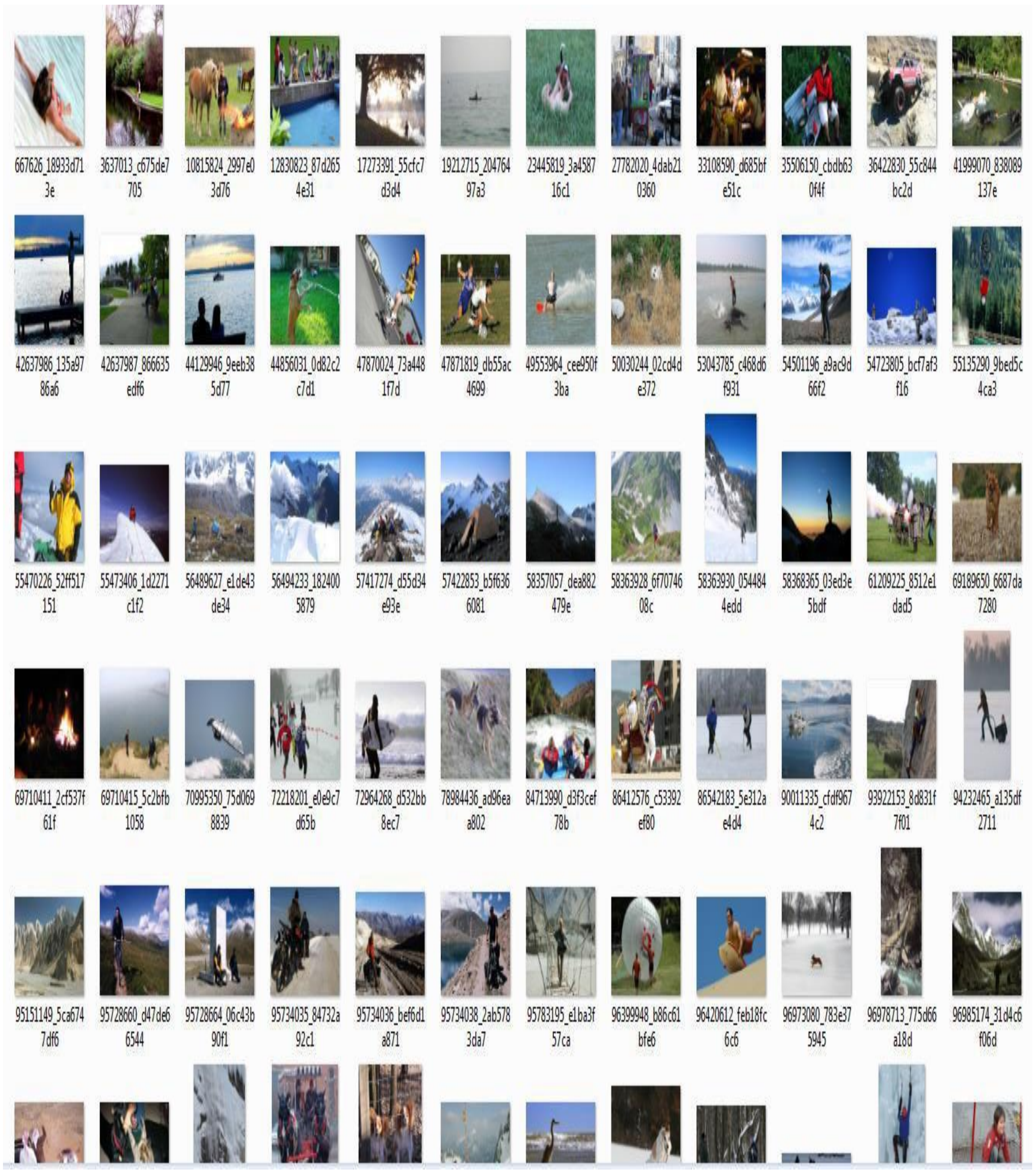
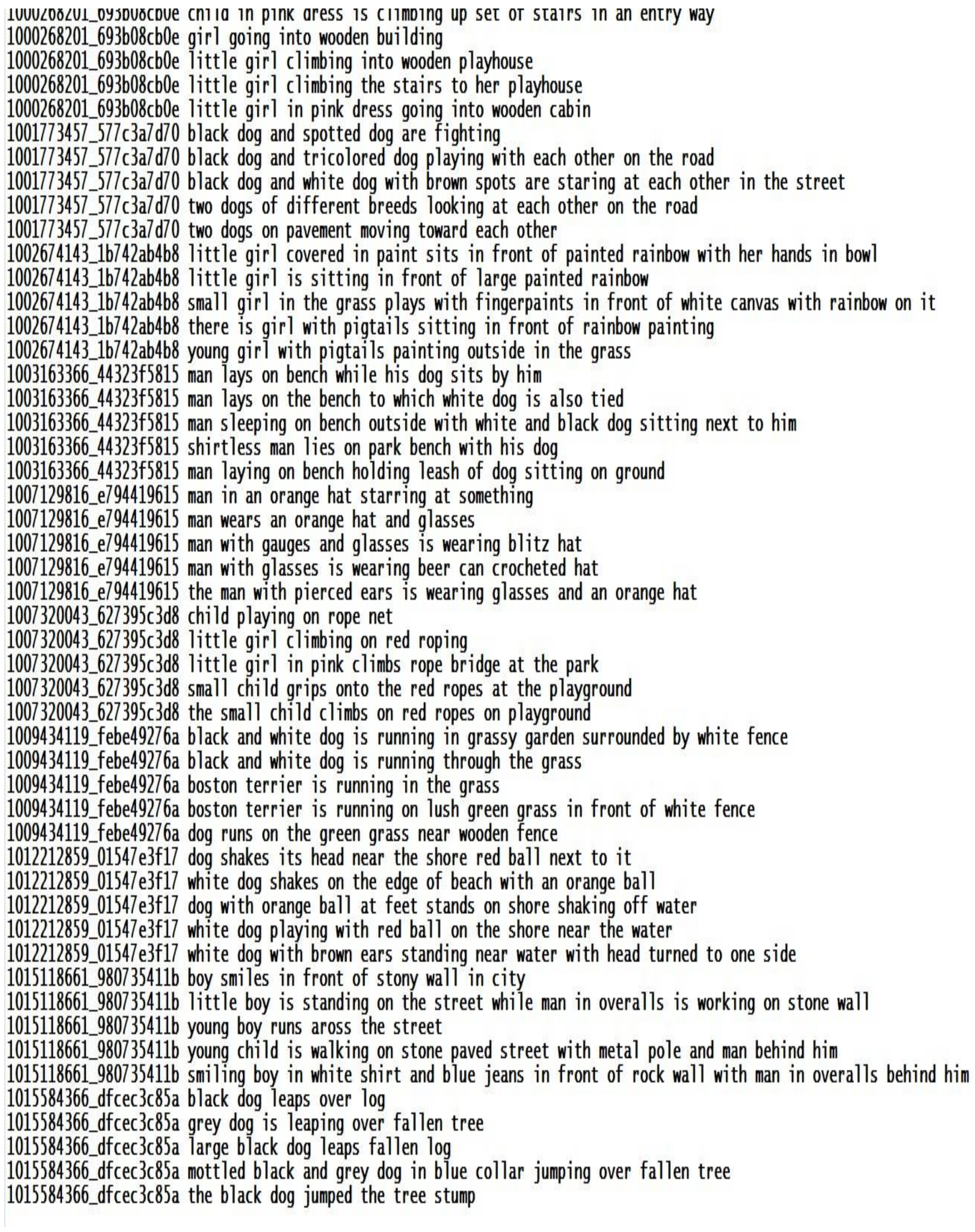


Fig 5.9.2: Flickr8k Dataset



```
1000268201_693b08cb0e child in pink dress is climbing up set of stairs in an entry way
1000268201_693b08cb0e girl going into wooden building
1000268201_693b08cb0e little girl climbing into wooden playhouse
1000268201_693b08cb0e little girl climbing the stairs to her playhouse
1000268201_693b08cb0e little girl in pink dress going into wooden cabin
1001773457_577c3a7d70 black dog and spotted dog are fighting
1001773457_577c3a7d70 black dog and tricolored dog playing with each other on the road
1001773457_577c3a7d70 black dog and white dog with brown spots are staring at each other in the street
1001773457_577c3a7d70 two dogs of different breeds looking at each other on the road
1001773457_577c3a7d70 two dogs on pavement moving toward each other
1002674143_1b742ab4b8 little girl covered in paint sits in front of painted rainbow with her hands in bowl
1002674143_1b742ab4b8 little girl is sitting in front of large painted rainbow
1002674143_1b742ab4b8 small girl in the grass plays with fingerpaints in front of white canvas with rainbow on it
1002674143_1b742ab4b8 there is girl with pigtails sitting in front of rainbow painting
1002674143_1b742ab4b8 young girl with pigtails painting outside in the grass
1003163366_44323f5815 man lays on bench while his dog sits by him
1003163366_44323f5815 man lays on the bench to which white dog is also tied
1003163366_44323f5815 man sleeping on bench outside with white and black dog sitting next to him
1003163366_44323f5815 shirtless man lies on park bench with his dog
1003163366_44323f5815 man laying on bench holding leash of dog sitting on ground
1007129816_e794419615 man in an orange hat starring at something
1007129816_e794419615 man wears an orange hat and glasses
1007129816_e794419615 man with gauges and glasses is wearing blitz hat
1007129816_e794419615 man with glasses is wearing beer can crocheted hat
1007129816_e794419615 the man with pierced ears is wearing glasses and an orange hat
1007320043_627395c3d8 child playing on rope net
1007320043_627395c3d8 little girl climbing on red roping
1007320043_627395c3d8 little girl in pink climbs rope bridge at the park
1007320043_627395c3d8 small child grips onto the red ropes at the playground
1007320043_627395c3d8 the small child climbs on red ropes on playground
1009434119_febe49276a black and white dog is running in grassy garden surrounded by white fence
1009434119_febe49276a black and white dog is running through the grass
1009434119_febe49276a boston terrier is running in the grass
1009434119_febe49276a boston terrier is running on lush green grass in front of white fence
1009434119_febe49276a dog runs on the green grass near wooden fence
1012212859_01547e3f17 dog shakes its head near the shore red ball next to it
1012212859_01547e3f17 white dog shakes on the edge of beach with an orange ball
1012212859_01547e3f17 dog with orange ball at feet stands on shore shaking off water
1012212859_01547e3f17 white dog playing with red ball on the shore near the water
1012212859_01547e3f17 white dog with brown ears standing near water with head turned to one side
1015118661_980735411b boy smiles in front of stony wall in city
1015118661_980735411b little boy is standing on the street while man in overalls is working on stone wall
1015118661_980735411b young boy runs aross the street
1015118661_980735411b young child is walking on stone paved street with metal pole and man behind him
1015118661_980735411b smiling boy in white shirt and blue jeans in front of rock wall with man in overalls behind him
1015584366_dfcec3c85a black dog leaps over log
1015584366_dfcec3c85a grey dog is leaping over fallen tree
1015584366_dfcec3c85a large black dog leaps fallen log
1015584366_dfcec3c85a mottled black and grey dog in blue collar jumping over fallen tree
1015584366_dfcec3c85a the black dog jumped the tree stump
```

Fig 5.9.3: Descriptions

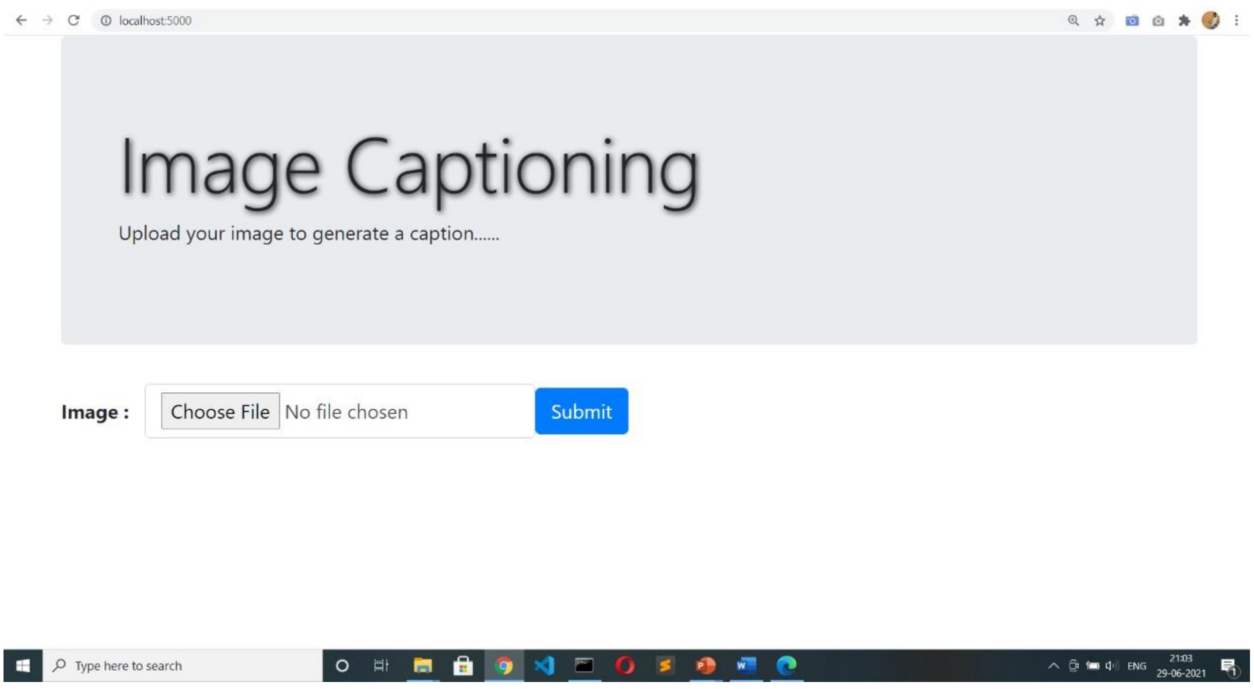
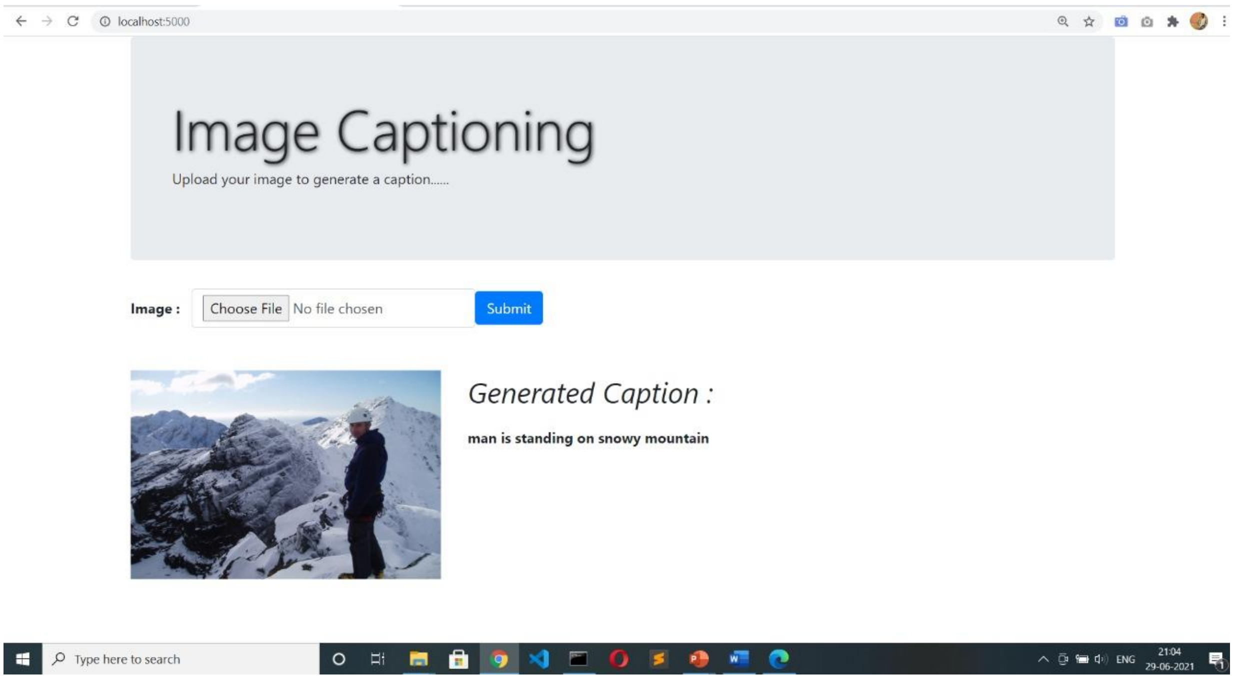


Fig 5.9.3: User Image Upload



Generated Caption :

man is standing on snowy mountain

Fig 5.9.4: Caption Generation Example

Chapter 6

CONCLUSION & FUTURE SCOPE

Image captioning has many advantages in almost every complex area of Artificial Intelligence. The main use case of our model is to help visually impaired to understand the environment and made them easy to act according to the environment. As, this is a complex task to do, with the help of pre trained models and powerful deep learning frameworks like Tensorflow and Keras, we made it possible. This is completely a Deep Learning project, which makes use of multiple Neural Networks like Convolutional Neural Network and Long Short Term Memory to detect objects and captioning the images. To deploy our model as a web application, we have used Flask, which is a powerful Python's web framework.

We are going to extend our work in the next higher level by enhancing our model to generate captions even for the live video frame. Our present model generates captions only for the image, which itself a complex task and captioning live video frames is much complex to create. This is completely GPU based and captioning live video frames cannot be possible with the general CPUs. Video captioning is a popular research area in which it is going to change the lifestyle of the people with the use cases being widely usable in almost every domain. It automates the major tasks like video surveillance and other security tasks.

REFERENCES

[1] Abhaya Agarwal and Alon Lavie. 2008. Meteor, m-bleu and m-ter: Evaluation metrics for high-correlation with human rankings of machine translation output. In Proceedings of the Third Workshop on Statistical Machine Translation. Association for Computational Linguistics, 115–118.

[2] Ahmet Aker and Robert Gaizauskas. 2010. Generating image descriptions using dependency relational patterns. In Proceedings of the 48th annual meeting of the association for computational linguistics. Association for Computational Linguistics, 1250– 1258.

[3] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In European Conference on Computer Vision. Springer, 382–398.

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2017. Bottom-up and top-down attention for image captioning and vqa. arXiv preprint arXiv:1707.07998 (2017).

[5] Jyoti Aneja, Aditya Deshpande, and Alexander G Schwing. 2018. Convolutional image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 5561–5570.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations (ICLR).

[7] Shuang Bai and Shan An. 2018. A Survey on Automatic Image Caption Generation. Neurocomputing.

[8] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, Vol. 29. 65–72.

[9] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In Advances in Neural Information Processing Systems. 1171–1179.

[10] Cristian Bodnar. 2018. Text to Image Synthesis Using Generative Adversarial Networks. arXiv preprint arXiv:1805.00676.