A Project Report
on

DETECTING AND CAPTIONING IMAGES USING CNN-LSTM DEEP NEURAL NETWORKS AND FLASK

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, ANANTHAPURAMU

In partial fulfillment of requirement for the award of the degree of

Bachelor of Technology

In

Computer Science & Engineering

By

P.YASWANTH SAI	**15F11A0563**
SK.ANEEF	**16F15A0501**
T.SAI HARISH	**15F11A0585**

Under the esteemed Guidance of

Dr.P.VENKATESWARA RAO
PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARAYANA ENGINEERING COLLEGE :: GUDUR

Recognized by UGC 2(f) and 12(B),An ISO 9001:2015 Certified Institution, Approved by AICTE New Delhi &Permanently Affiliated to JNTUA, Ananthapuramu

NARAYANA ENGINEERING COLLEGE :: GUDUR

Recognized by UGC 2(f) and 12(B),An ISO 9001:2015 Certified Institution, Approved by AICTE New Delhi &Permanently Affiliated to JNTUA, Ananthapuramu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project entitled **DETECTING AND CAPTIONING IMAGES USING CNN-LSTM DEEP NEURAL NETWORKS AND FLASK** that is being submitted by **P.YASWANTH SAI, SK.ANEEF, T.SAI HARISH** in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** to JNTUA Ananthapuramu is recorded to be the bonafide work carried out by them under my guidance and supervision.

PROJECT GUIDE

Dr. P Venkateswara Rao
Professor

HEAD OF THE DEPARTMENT

Dr. P Venkateswara Rao
Professor.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely thankful to **Dr. P. Narayana**, the **Founder Chairman** of **Narayana Group** for his good initiation starting technical institution in Gudur like rural area for helping economically poor students. We are also thankful to **Mr. K. Puneeth**, the **Chairman** of **Narayana Group** for providing the infrastructural facilities to work in, without this our work would not have been possible.

We would like to express our deep sense of gratitude to **Dr. Ch.V.S.Parameswara Rao, Principal, Narayana Engineering College, Gudur** for his continuous effort in creating a competitive environment in our college and encouraging throughout this course.

We would like to convey our heartfelt thanks to **Dr. P.Venkateswara Rao, Professor & HOD** of Computer Science and Engineering for giving the opportunity to embark up on this topic and for his continues encouragement throughout the preparation of the project.

We would like to thank our guide **Dr. P. Venkateswara Rao, Professor** for his invaluable guidance, constant assistance, support, endurance and constructive suggestions for the betterment of the project.

We also wish to thank all the **staff members** of the Department of Computer Science & Engineering for helping us directly or indirectly in completing this project successfully.

Finally we are thankful to our **parents and friends** for their continued moral and material support throughout the course and in helping us to finalize the report.

P.YASWANTH SAI (15F11A0563)
SK.ANEEF (16F15A0501)
T.SAI HARISH (15F11A0585)

DECLARATION

We hereby declare that the project entitled **DETECTING AND CAPTIONING IMAGES USING CNN-LSTM DEEP NEURAL NETWORKS AND FLASK** has been done by us under the guidance of **Dr. P. Venkateswara Rao, Professor** Department of Computer Science & Engineering.This project work has been submitted to **NARAYANA ENGINEERING COLLEGE, GUDUR** as a part of partial fulfillment of the requirements for the award of degree of **Bachelor of Technology**.

We also declare that this project report has not been submitted at any time to another institute or University for the award of any degree.

P.YASWANTH SAI (15F11A0563)
SK.ANEEF (16F15A0501)
T.SAI HARISH (15F11A0585)

Place : Gudur

Date:

INDEX

ABSTRACT

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.

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.

LIST OF FIGURES

LIST OF TABLES

ABBREVATIONS

CNN	-	Convolutional Neural Network.
RNN	-	Recurrent Neural Network.
LSTM	-	Long Short Term Memory.
OPENCV	-	Open Computer Vision.
NLTK	-	Natural Language Tool Kit
NLP	-	Natural Language Processing.
PYPI	-	Python Package Installer
TF	-	Tensorflow

1.Introduction

Image caption generation has emerged as a challenging and important research area following ad-vances in statistical language modelling and image recognition. The 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.

There are two main approaches to Image Captioning: bottom-up and top-down. Bottom-up ap-proaches, such as those by [1] [2] [3], generate items observed in an image, and then attempt to combine the items identified into a caption. Top-down approaches, such as those by [4] [5] [6], attempt to generate a semantic representation of an image that is then decoded into a caption using various architectures, such as recurrent neural networks. The latter approach follows in the footsteps of recent advances in statistical machine translation, and the state-of-the-art models mostly adopt the top-down approach.

Our approach draws on the success of the top-down image generation models listed above. We use a deep convolutional neural network to generate a vectorized representation of an image that we then feed into a Long-Short-Term Memory (LSTM) network, which then generates captions. Figure 1 provides the broad framework for our approach.

One of the main challenges in the field of Image Captioning is overfitting the training data. This is because the largest datasets, such as the Microsoft Common Objects in Context (MSCOCO) dataset, only have 160000 labelled examples, from which any top-down architecture must learn (a) a robust image representation, (b) a robust hidden-state LSTM based representation to capture image semantics and (c) language modelling for syntactically-sound oriented design for the unique purpose of caption generation.

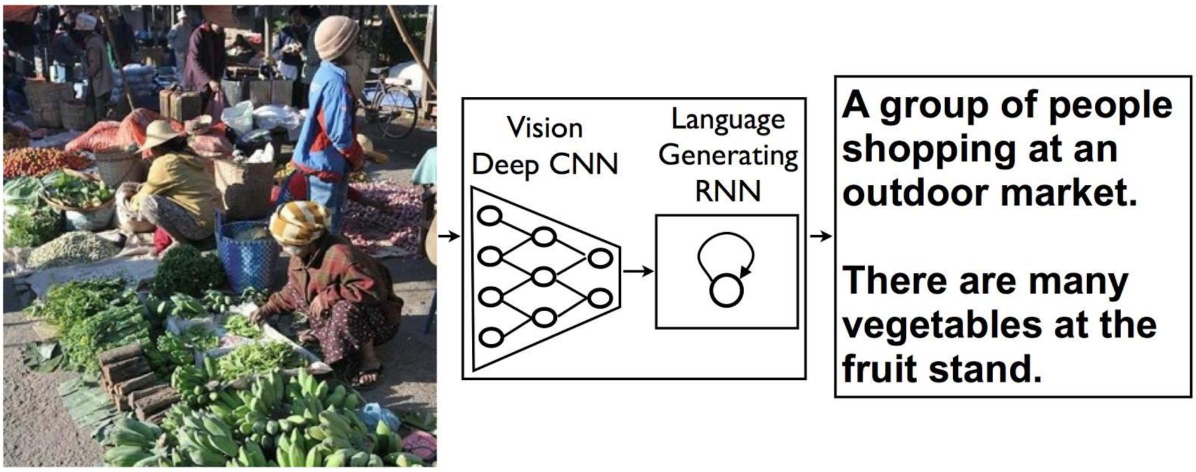


Figure 1: (Left) Our CNN-LSTM architecture, modelled after the NIC architecture described in [6]. We use a deep convolutional neural network to create a semantic representation of an image, which we then decode using a LSTM network. (Right) A unrolled LSTM network for our CNN-LSTM model. All LSTMs share the same parameters. The vectorized image representation is fed into the network, followed by a special start of sentence token. The hidden state produced is then used by the LSTM predict/generate the caption for the given image. Figures taken from [6] manifests itself in the memorization of inputs and the use of similar sounding captions for images which differ in their specific details. For example, an image of a man on a skateboard on a ramp may receive the same caption has an image of a man on a skateboard on a table.

To cope with this, recent advances in the field of Image Captioning have innovated at the architecture-level, with the most successful model to date on the Microsoft Common Objects in Context competition using the basic architecture in Figure 1 augmented with an attention mecha-nism [7]. This allows it to deal with the main challenge of top-down approaches, i.e. the inability to focus the caption on small and specific details in the image. In this paper, we approach the problem via thorough hyper-parameter experimentation on the basic architecture in Figure 1.For most computer vision researchers the classification task has always been dominant in the field. Either it was a scene understanding in the pioneer 1960s or a traffic sign detection in the modern days, the task has been rooted in the soil of computer vision. It is not surprising that one of the most significant competition in the field comprises the image classification task among others. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) awards annually the algorithm which is most successful at predicting the class of an image in its five estimates (known as top-5 error). For the record, the lowest top-5 classification error reached 28.2% at the ILSVRC2010 and 25.8% a year later, respectively [1]. Nonetheless, an unxpected breakthrough came in the year 2012 when Krizhevsky et al. [2] presented decades old algorithms [3, 4] enhanced by novel training techniques achieving so-far-not-seen results. In particular, the top-5 classification error was pushed to 16.4%. At the latest contest in 2015, the lowest top-5 error was brought to 3.5%, drawing on the work of Krizhevsky et al. After this success, neural networks has revolutionised the field and brought in new challenges that had not been merely considerable before. One of those newly feasible techniques – image captioning – is discussed in this thesis. In fact, as an arising discipline with promising potential, image captioning still is an active area of research nowadays, striving to answer unsolved questions. Consecutively, since the field has not been entirely established yet, one must rely mainly on recently published papers and on-line lectures only. Considering recent work, we define image captioning as a task in which an algorithm describes a particular image with a statement. However, it is expected that the statement is meaningful, self-contained and grammatically and semantically correct. In other words, the caption shall describe the image concretely, shall not require or rely on additional information and, last but not least, be consisted of a grammatically correct sentence that semantically corresponds to the image.

2. FEASIBILITY STUDY

Preliminary investigation examine project practicability, the chance the system are helpful to the organization. The most objective of the practicability study is to check the Technical, Operational and Economical practicability for adding new modules and debugging previous running system. All system is possible if they're unlimited resources and infinite time. There are unit aspects within the practicability study portion of the preliminary investigation

- Technical Feasibility
- Economical Feasibility
- Social Feasibility

2.1 Technical Feasibility

The technical issue typically raised throughout the practicableness stage of the investigation includes the following:

- Does the mandatory technology exist to try to what's suggested?
- Do the planned equipments have the technical capability to carry the info needed to use the new system?
- Will the planned system offer adequate response to inquiries, despite the amount or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, responsibleness, simple access and information security?

Earlier no system existed to cater to the requirements of 'Secure Infrastructure Implementation System'. this system developed is technically possible. it's an internet primarily based interface for audit work flow at NIC-CSD. therefore it provides a simple access to the users.

The database's purpose is to make, establish and maintain a work flow among numerous entities so as to facilitate all involved users in their numerous capacities or roles. Permission to the users would be granted supported the roles nominative. Therefore, it provides the technical guarantee of accuracy, responsibleness and security. The package and laborious needs for the event of this project aren't several and area unit already out there in-house at NIC or area unit out there as free as open supply.

The work for the project is finished with this instrumentality and existing package technology.

Necessary information measure exists for providing a quick feedback to the users no matter the amount of user's victimization the system.

2.2 Economical Feasibility

A system is developed technically which are used if put in should still be an honest investment for the organization. within the economical practicableness, the event price in making the system is evaluated against the last word profit derived from the new systems. money advantages should equal or exceed the prices.

The system is economically possible. It doesn't need any addition hardware or code. Since the interface for this technique is developed mistreatment the prevailing resources and technologies out there at NIC, there's nominal expenditure and economical practicableness sure.

2.3 Social Feasibility

Proposed comes square measure useful given that they will be clad into data system. That may meet the organizations in operation needs. Operational feasibleness aspects of the project square measure to be taken as a vital a part of the project implementation. a number of the vital problems raised square measure to check the operational feasibleness of a project includes the following: -

- Is there spare support for the management from the users?
- Will the system be used and work properly if it's being developed and implemented?
- Will there be any resistance from the user that may undermine the potential application benefits?

This system is targeted to be in accordance with the above-named problems. Beforehand, the management problems and user needs are taken into thought. Therefore there's absolute confidence of resistance from the users that may undermine the potential application edges.

The well-planned style would make sure the optimum utilization of the pc resources and would facilitate within the improvement of performance standing.

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:

$$a = (x) = (w^T x + b) \tag{1}$$

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

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

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 Papert 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 :

$$a_1^{(k)} = {}_1^{(k)}(a^{(k\ \ 1)})$$

$$a_2^{(k)} = {}_2^{(k)}(a^{(k\ \ 1)})$$

$$. \qquad (3)$$

$$.$$

$$.$$

$$a^{(k)}_{N} = {}^{(k)}_{N} \qquad (a^{(k\ \ 1)})$$

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:

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

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 ( ).

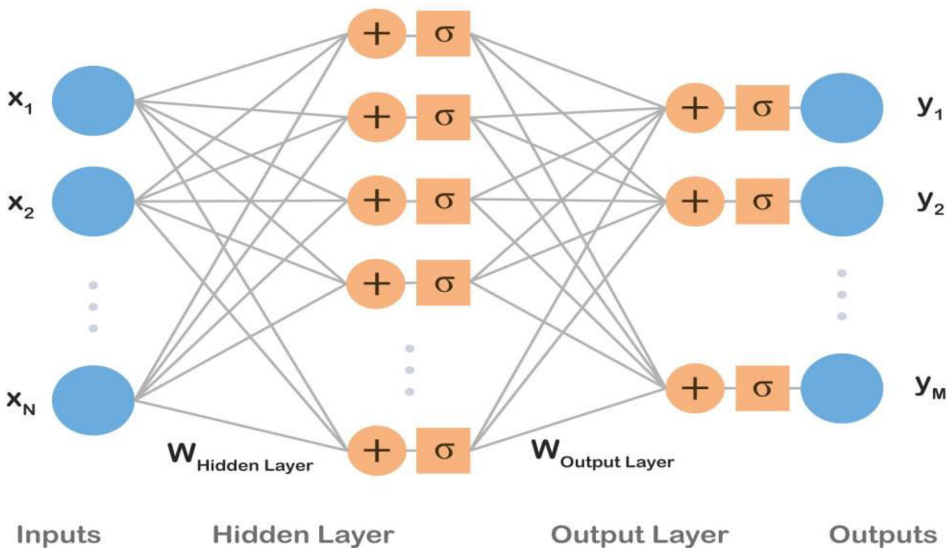


Figure 3: Inputs $x_1; : : : ; x_N$ are processed by a hidden layer and, consecutively, by an output layer producing outputs $y_1; : : : ; y_M$. Biases b were omitted for the sake of simplicity.

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.

$$(z) = \frac{1}{1 + e^{z}} \tag{7}$$

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]:

$$(z) = \max(0; z) \tag{8}$$

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:

$$y = (w(1;2) \ (w(0;1)_x + b(0;1)) + b(1;2)) \tag{9}$$

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 [14], 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 con-nected 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 [18] 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.

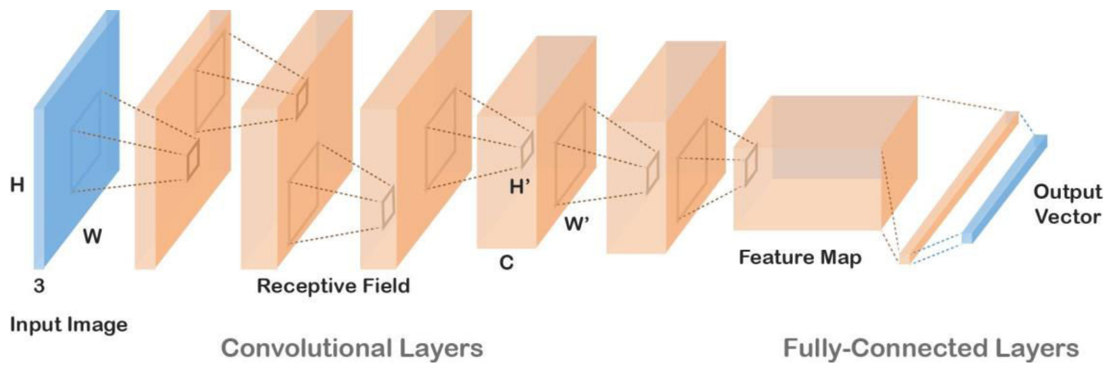
3.3.Convolutional Neural Networks

In image analysis, many of recent advances in deep learning are built on the work of LeCun et al. [18] 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 CNNa 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 con-nected layers that atten 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 di erent from AlexNet, the term maintained its popularity. Additionally, depending on a prob-lem 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 in Fig. 4

3.3.1.Receptive Field



As mentioned above, a convolutional layer takes a tensor on input and produces a ten-sor, 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 speci c 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 eld de nes an area in the W H grid. The shape of the receptive eld is a hyper-parameter and varies across the models.

Figure 4: 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 di erent 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 speci c 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 signi cantly 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. Dimensions reduction can be obtained by using stride when convolving, leading to dilution of receptive elds overlap. Nev-ertheless, 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 [20]. 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) [21] { a rather complex design, yet outperforming others [20].

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 [22].

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 three survey papers [8, 13, 75] 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 at el [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 expermimented 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

4.SYSTEM DESIGN

4.1.System Architecture

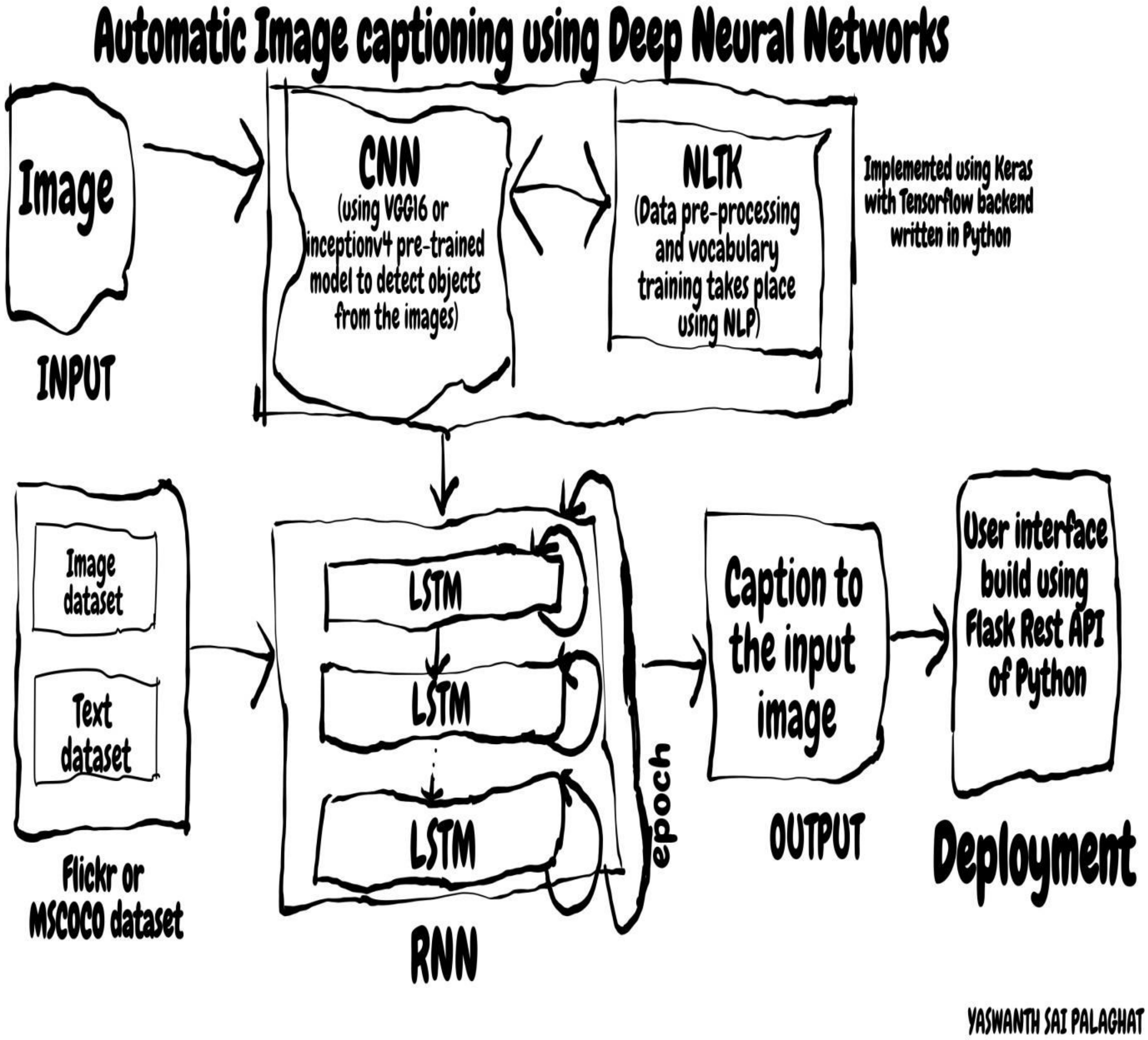


Fig 1: System Architecture

4.2 UML Diagrams

Unified Modeling Language:

The Unified Modeling Language permits the technologist to specific AN analysis model mistreatment the modeling notation that's ruled by a group of grammar linguistics and pragmatic rules.

A UML system is diagrammatical mistreatment 5 completely different views that describe the system from clearly different perspective. Every read is outlined by a group of diagram, that is as follows.

It represents the dynamic of behavioral as elements of the system, portrayal the interactions of assortment between varied structural components delineated within the user model and structural model read.

Use case Diagrams represent the practicality of the system from a user’s purpose of read. Use cases are used throughout needs induction and analysis to represent the practicality of the system. Use cases specialize in the behavior of the system from external purpose of read.

Actors are external entities that move with the system. Samples of actors embody users like administrator, bank client …etc., or another system like central info.

4.2.1 Use Case Diagrams:

Use case diagrams model the practicality of system treatment actors and use cases.

Use cases are services or functions provided by the system to its users.

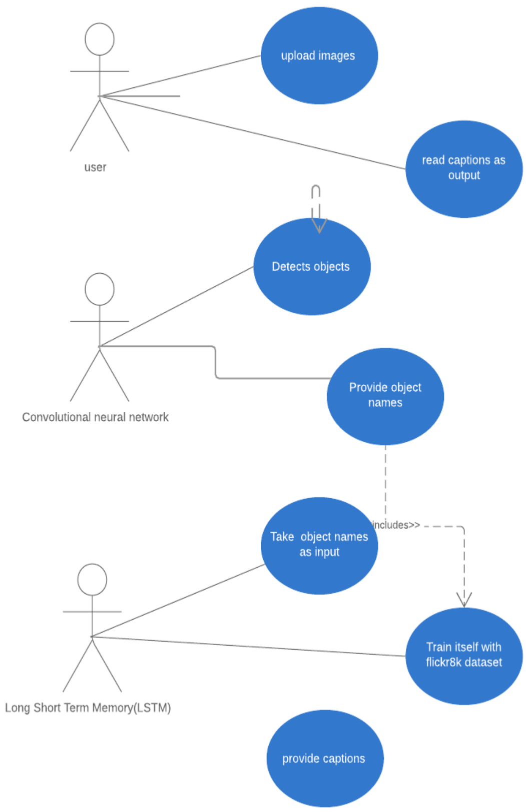


Fig 2.1: Use Case diagram

4.2.2 Class Diagram:

Class diagrams are the backbone of virtually each object-oriented methodology as well as UML. They describe the static structure of a system. Categories represent associate degree abstraction of entities with common characteristics. Associations represent the relationships between categories.

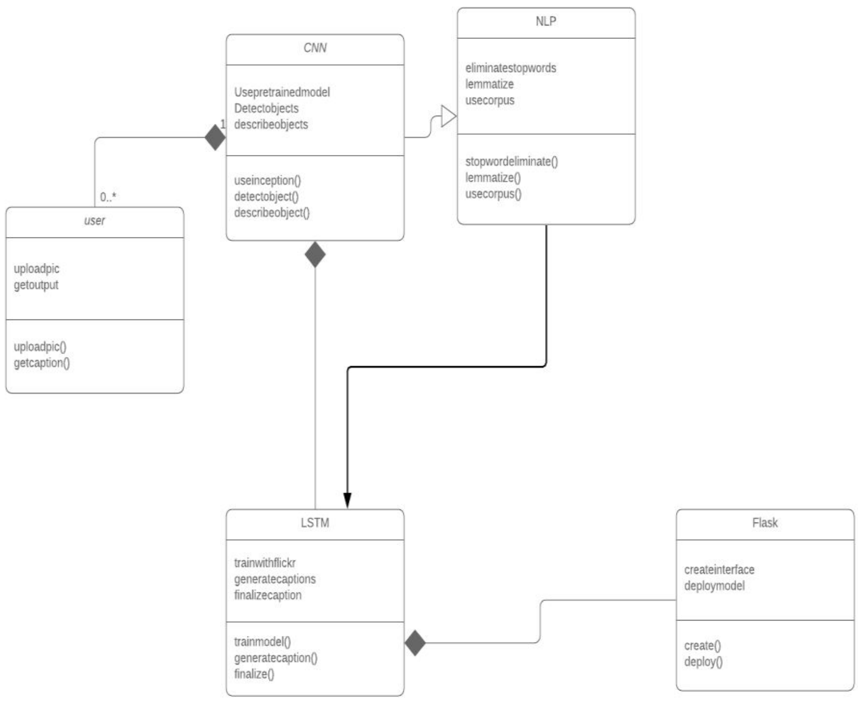


Fig 2.2: Class Diagram

4.2.3 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.

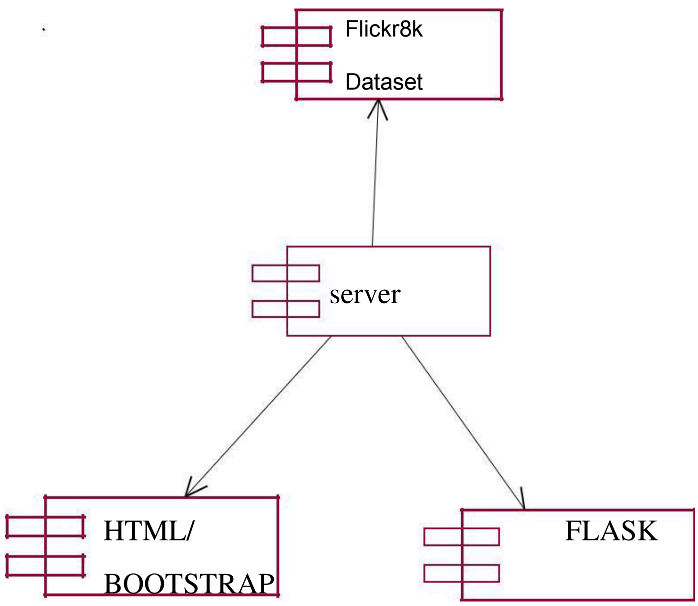


Fig 2.3: Component Diagram

4.2.4 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.

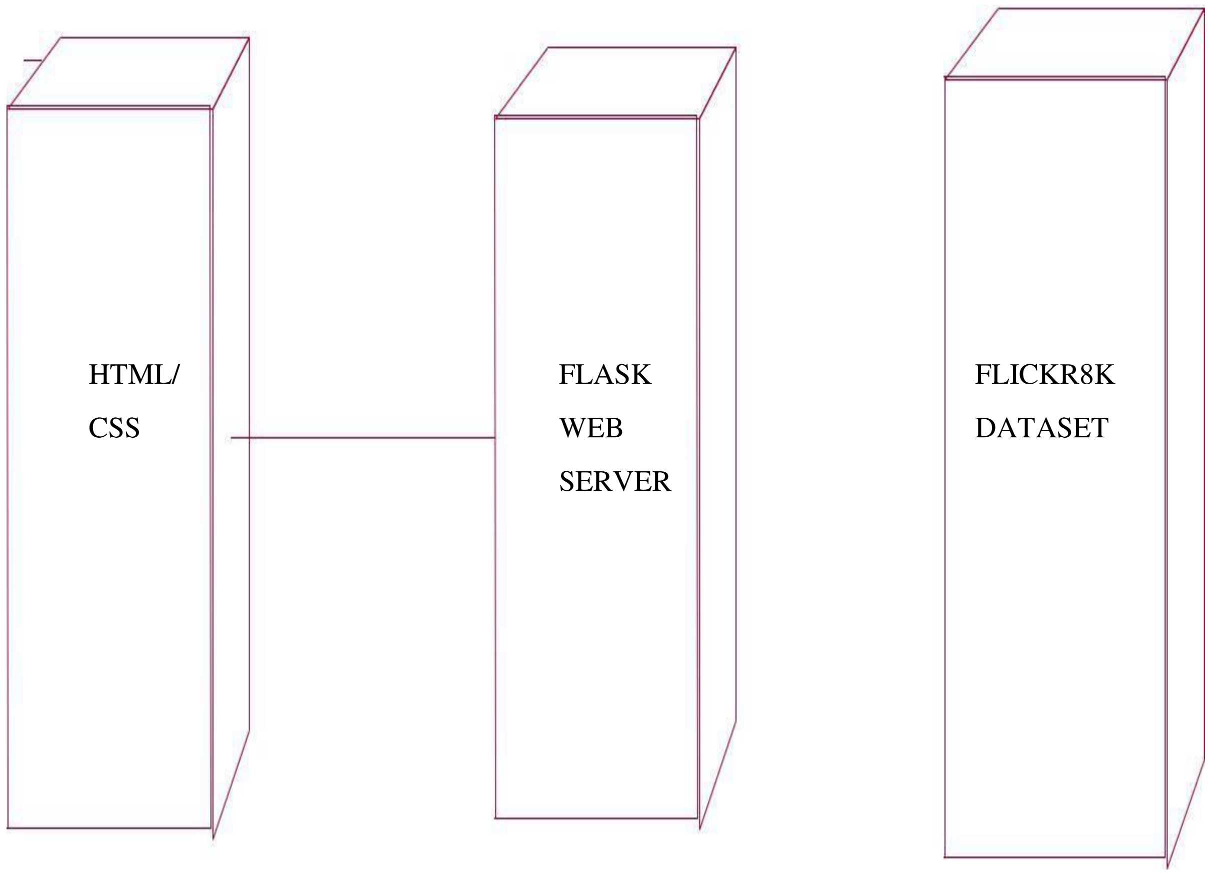


Fig 2.4: Deployment Diagram

4.2.5 Sequence Diagram:

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

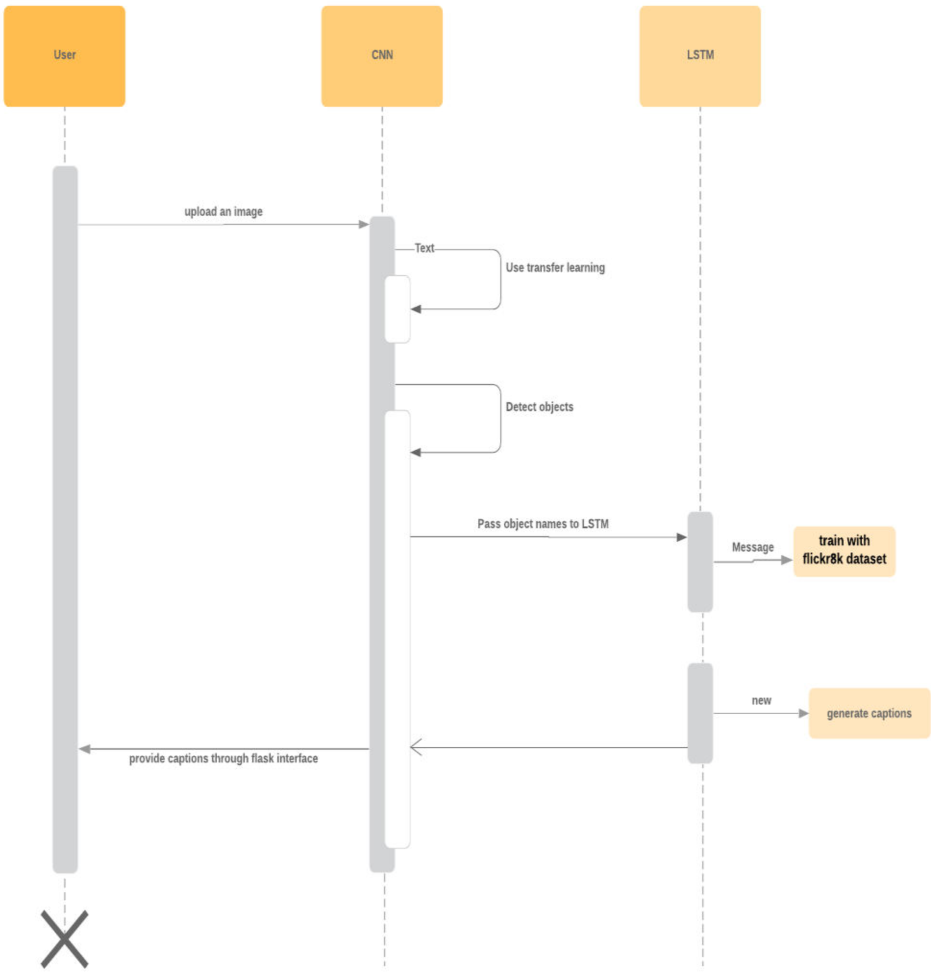


Fig 2.5: Sequence Diagram

4.2.6 Activity Diagram:

An activity diagram illustrates the dynamic nature of a system by modeling the flow of management from activity to activity. An activity represents AN operation on some category within the system that leads to an amendment within the state of the system. Typically, activity diagrams are accustomed model progress or business processes and internal operation. As a result of AN activity diagram may be a special quite state chart diagram, it uses a number of constant modeling conventions.

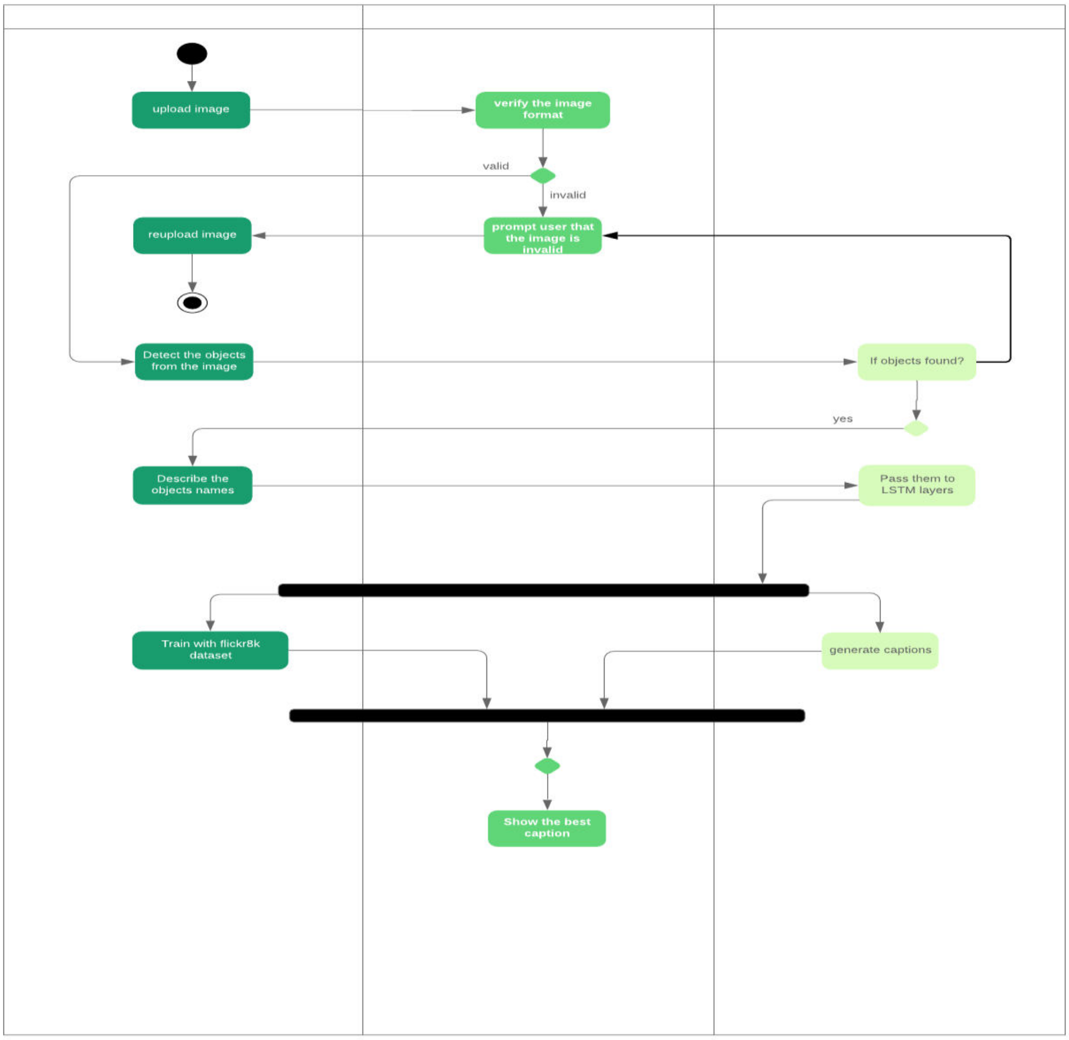


Fig 2.6: Activity Diagram

4.2.7 Dataflow Diagram:

A data-flow diagram is a way of representing a flow of a data of a process or a system The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops.

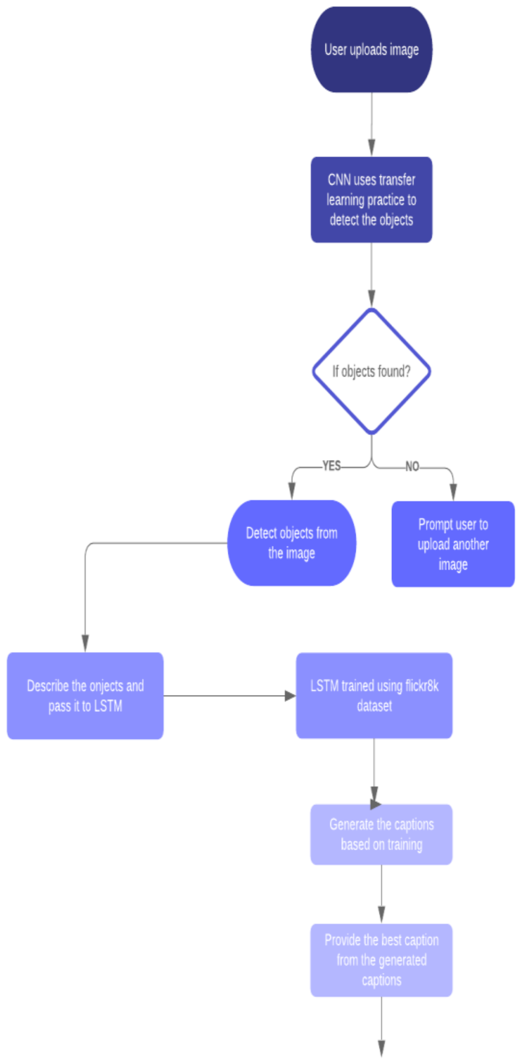


Fig 2.7: Data Flow Diagram

5. IMPLEMENTATION

5.1.Modules

5.1.1.Object Detection:

In this module, Convolutional Neural Network performs the task of Object Detection from the images. In this phase, Transfer learning methodology is used to extract the previously used knowledge.We have used pre-trained model named ssd_mobilenet_coco to detect the objects from the image , which contains the functionality of convolutional neural network.

The code for detecting the objects with Flask interface is shown below:

app.py

```
import os

from flask import Flask, render_template, request, redirect, url_for, send_from_directory

from flask_bootstrap import Bootstrap

from werkzeug import secure_filename

import numpy as np

import os

import six.moves.urllib as urllib

import sys

import tensorflow as tf

from collections import defaultdict

from io import StringIO

from PIL import Image
```

```
sys.path.append("..")

from utils import label_map_util

from utils import visualization_utils as vis_util

MODEL_NAME = 'ssd_mobilenet_v1_coco_11_06_2017'

PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')

NUM_CLASSES = 90


detection_graph = tf.Graph()

with detection_graph.as_default():

  od_graph_def = tf.GraphDef()

  with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:

    serialized_graph = fid.read()

    od_graph_def.ParseFromString(serialized_graph)

    tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)

categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)

category_index = label_map_util.create_category_index(categories)



def load_image_into_numpy_array(image):

  (im_width, im_height) = image.size

  return np.array(image.getdata()).reshape(

      (im_height, im_width, 3)).astype(np.uint8)
```

```
app = Flask(__name__)
bootstrap = Bootstrap(app)

app.config['UPLOAD_FOLDER'] = 'uploads/'
app.config['ALLOWED_EXTENSIONS'] = set(['png', 'jpg', 'jpeg'])

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in app.config['ALLOWED_EXTENSIONS']


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/upload', methods=['POST'])
def upload():
    file = request.files['file']
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
```

```
        return redirect(url_for('uploaded_file',
                                filename=filename))



@app.route('/uploads/<filename>')
def uploaded_file(filename):
    PATH_TO_TEST_IMAGES_DIR = app.config['UPLOAD_FOLDER']
    TEST_IMAGE_PATHS = [
os.path.join(PATH_TO_TEST_IMAGES_DIR,filename.format(i)) for i in range(1, 2) ]
    IMAGE_SIZE = (12, 8)

    with detection_graph.as_default():
        with tf.Session(graph=detection_graph) as sess:
            for image_path in TEST_IMAGE_PATHS:
                image = Image.open(image_path)
                image_np = load_image_into_numpy_array(image)
                image_np_expanded = np.expand_dims(image_np, axis=0)
                image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
                boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
                scores = detection_graph.get_tensor_by_name('detection_scores:0')
                classes = detection_graph.get_tensor_by_name('detection_classes:0')
                num_detections = detection_graph.get_tensor_by_name('num_detections:0')
                (boxes, scores, classes, num_detections) = sess.run(
                    [boxes, scores, classes, num_detections],
                    feed_dict={image_tensor: image_np_expanded})
```

```
            vis_util.visualize_boxes_and_labels_on_image_array(
                image_np,
                np.squeeze(boxes),
                np.squeeze(classes).astype(np.int32),
                np.squeeze(scores),
                category_index,
                use_normalized_coordinates=True,
                line_thickness=8)
            im = Image.fromarray(image_np)
            im.save('uploads/'+filename)

    return send_from_directory(app.config['UPLOAD_FOLDER'],
                               filename)
if __name__ == '__main__':
    app.run(debug=True,host='0.0.0.0',port=5000)
```

5.1.2. Loading Flickr8k Dataset:

The code for loading the data from Flickr8k dataset is shown below:

load_data.py

```
import numpy as np
from preprocessing import *
```

```
from pickle import load, dump

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad_sequences

from keras.utils import to_categorical


'''

We have Flickr_8k.trainImages.txt and Flickr_8k.devImages.txt files which consist of unique
identifiers which can be used to filter the images and their descriptions

'''

# load a pre-defined list of photo identifiers

def load_set(filename):

	file = open(filename, 'r')

	doc = file.read()

	file.close()

	dataset = list()

	# process line by line

	for line in doc.split('\n'):

		# skip empty lines

		if len(line) < 1:

			continue

		# get the image identifier

		identifier = line.split('.')[0]

		dataset.append(identifier)

	return set(dataset)
```

'''

The model we will develop will generate a caption given a photo, and the caption will be generated one word at a time.

The sequence of previously generated words will be provided as input. Therefore, we will need a ‘first word’ to kick-off the generation process

and a ‘last word‘ to signal the end of the caption.

We will use the strings ‘startseq‘ and ‘endseq‘ for this purpose. These tokens are added to the loaded descriptions as they are loaded.

It is important to do this now before we encode the text so that the tokens are also encoded correctly.

'''

```
# load clean descriptions into memory

def load_clean_descriptions(filename, dataset):

    file = open(filename, 'r')

    doc = file.read()

    file.close()

    descriptions = dict()

    for line in doc.split('\n'):

        # split line by white space

        tokens = line.split()

        # split id from description

        image_id, image_desc = tokens[0], tokens[1:]

        # skip images not in the set

        if image_id in dataset:

            # create list

            if image_id not in descriptions:
```

```
                    descriptions[image_id] = list()
                # wrap description in tokens
                desc = 'startseq ' + ' '.join(image_desc) + ' endseq'
                # store
                descriptions[image_id].append(desc)
    return descriptions
```

'''

The description text will need to be encoded to numbers before it can be presented to the model as in input or compared to the model’s predictions.

The first step in encoding the data is to create a consistent mapping from words to unique integer values. Keras provides the Tokenizer class that

can learn this mapping from the loaded description data.

'''

```
# convert a dictionary of clean descriptions to a list of descriptions


def to_lines(descriptions):
    all_desc = list()
    for key in descriptions.keys():
        [all_desc.append(d) for d in descriptions[key]]
    return all_desc


# fit a tokenizer given caption descriptions
```

```
def create_tokenizer(descriptions):
    lines = to_lines(descriptions)
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(lines)
    return tokenizer

'''
Each description will be split into words. The model will be provided one word and the photo
and generate the next word.
Then the first two words of the description will be provided to the model as input with the
image to generate the next word.
This is how the model will be trained.
For example, the input sequence "little girl running in field" would be
split into 6 input-output pairs to train the model:

X1,     X2 (text sequence),                                   y (word)
photo   startseq,                                             little
photo   startseq, little,                                     girl
photo   startseq, little, girl,                               running
photo   startseq, little, girl, running,                      in
photo   startseq, little, girl, running, in,                  field
photo   startseq, little, girl, running, in, field, endseq
'''
```

```
# create sequences of images, input sequences and output words for an image

def create_sequences(tokenizer, max_length, desc_list, photo):
    #X1 : input for photo features
    #X2 : input for text features
    X1, X2, y = list(), list(), list()
    vocab_size = len(tokenizer.word_index) + 1
    # walk through each description for the image
    for desc in desc_list:
        # encode the sequence
        seq = tokenizer.texts_to_sequences([desc])[0]
        # split one sequence into multiple X,y pairs
        for i in range(1, len(seq)):
            # split into input and output pair
            in_seq, out_seq = seq[:i], seq[i]
            # pad input sequence
            in_seq = pad_sequences([in_seq], maxlen=max_length)[0]
            # encode output sequence
            out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]
            # store
            X1.append(photo)
            X2.append(in_seq)
```

```
                    y.append(out_seq)

        return np.array(X1), np.array(X2), np.array(y)


# calculate the length of the description with the most words



def max_lengthcalc(descriptions):

        lines = to_lines(descriptions)

        return max(len(d.split()) for d in lines)


# load photo features



def load_photo_features(filename, dataset):

        # load all features

        all_features = load(open(filename, 'rb'))

        # filter features

        features = {k: all_features[k] for k in dataset}

        return features



# data generator, intended to be used in a call to model.fit_generator()
```

```
def data_generator(photos, descriptions, tokenizer, max_length):
	# loop for ever over images
	while 1:
		for key, desc_list in descriptions.items():
			# retrieve the photo feature
			photo = photos[key][0]
			in_img, in_seq, out_word = create_sequences(tokenizer, max_length,
desc_list, photo)
			yield [[in_img, in_seq], out_word]




def loadTrainData(path =
'train_val_data/Flickr_8k.trainImages.txt',preprocessDataReady=True):

	train = load_set(path)
	print('Dataset: %d' % len(train))

	# check if we already have preprocessed data saved and if not, preprocess the data.
	if preprocessDataReady is False:
		preprocessData()

	# descriptions
	train_descriptions = load_clean_descriptions('model_data/descriptions.txt', train)
	print('Descriptions: train=%d' % len(train_descriptions))
```

```
    # photo features
    train_features = load_photo_features('model_data/features.pkl', train)
    print('Photos: train=%d' % len(train_features))

    # prepare tokenizer
    tokenizer = create_tokenizer(train_descriptions)
    # save the tokenizer
    dump(tokenizer, open('model_data/tokenizer.pkl', 'wb'))

    # determine the maximum sequence length
    max_length = max_lengthcalc(train_descriptions)

    return train_features, train_descriptions, max_length


def loadValData(path = 'train_val_data/Flickr_8k.devImages.txt'):

    val = load_set(path)
    print('Dataset: %d' % len(val))

    # descriptions
    val_descriptions = load_clean_descriptions('descriptions.txt', val)
    print('Descriptions: val=%d' % len(val_descriptions))
```

```
    # photo features
    val_features = load_photo_features('features.pkl', val)
    print('Photos: val=%d' % len(val_features))

    return val_features, val_descriptions
```

5.1.3.Preprocess the data:

The code for preprocessing the data extracted from Flickr8k dataset is show below:

preprocessing.py

```
import numpy as np
import string
from os import listdir
from pickle import dump
from model import *
from keras.applications.inception_v3 import preprocess_input
from keras.preprocessing.image import load_img, img_to_array

#The function returns a dictionary of image identifier to image features.
def extract_features(path):
    model = defineCNNmodel()
```

```
    # extract features from each photo
  features = dict()
  for name in listdir(path):
        # load an image from file
      filename = path + '/' + name
      image = load_img(filename, target_size=(299, 299))
        # convert the image pixels to a numpy array
      image = img_to_array(image)
        # reshape data for the model
      image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
        # prepare the image for the VGG model
      image = preprocess_input(image)
        # get features
      feature = model.predict(image, verbose=0)
        # get image id
      image_id = name.split('.')[0]
        # store feature
      features[image_id] = feature
  return features


# extract descriptions for images
def load_descriptions(filename):
    file = open(filename, 'r')
    doc = file.read()
```

```
    file.close()
    mapping = dict()
    # process lines by line
    for line in doc.split('\n'):
        # split line by white space
        tokens = line.split()
        if len(line) < 2:
            continue
        # take the first token as the image id, the rest as the description
        image_id, image_desc = tokens[0], tokens[1:]
        # remove filename from image id
        image_id = image_id.split('.')[0]
        # convert description tokens back to string
        image_desc = ' '.join(image_desc)
        # create the list if needed
        if image_id not in mapping:
            mapping[image_id] = list()
        # store description
        mapping[image_id].append(image_desc)
    return mapping

def clean_descriptions(descriptions):
    # prepare translation table for removing punctuation
    table = str.maketrans('', '', string.punctuation)
```

```
	for key, desc_list in descriptions.items():
		for i in range(len(desc_list)):
			desc = desc_list[i]
			# tokenize
			desc = desc.split()
			# convert to lower case
			desc = [word.lower() for word in desc]
			# remove punctuation from each token
			desc = [w.translate(table) for w in desc]
			# remove hanging 's' and 'a'
			desc = [word for word in desc if len(word)>1]
			# remove tokens with numbers in them
			desc = [word for word in desc if word.isalpha()]
			# store as string
			desc_list[i] =  ' '.join(desc)


# save descriptions to file, one per line
def save_descriptions(descriptions, filename):
	lines = list()
	for key, desc_list in descriptions.items():
		for desc in desc_list:
			lines.append(key + ' ' + desc)
	data = '\n'.join(lines)
	file = open(filename, 'w')
```

```
        file.write(data)
        file.close()


def preprocessData():
        # extract features from all images
        path = 'train_val_data/Flicker8k_Dataset'
        print('Generating image features...')
        features = extract_features(path)
        print('Completed. Saving now...')
        # save to file
        dump(features, open('model_data/features.pkl', 'wb'))
        print("Save Complete.")


        # load descriptions containing file and parse descriptions
        descriptions_path = 'train_val_data/Flickr8k.token.txt'


        descriptions = load_descriptions(descriptions_path)
        print('Loaded Descriptions: %d ' % len(descriptions))


        # clean descriptions
        clean_descriptions(descriptions)
        # save descriptions
        save_descriptions(descriptions, 'model_data/descriptions.txt')
```

Now descriptions.txt is of form :

Example : 2252123185_487f21e336 stadium full of people watch game

5.1.4.Training the model:

The LSTM model, which is the advanced model of RNN should be trained with the dataset. The code for training the model is shown below:

model.py

```
from numpy import argmax

from keras.applications.inception_v3 import InceptionV3

from keras.models import Model

from keras.layers import Input

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Embedding

from keras.layers import Dropout

from keras.layers.merge import add

from keras.preprocessing.sequence import pad_sequences

from keras.preprocessing.image import load_img, img_to_array

from nltk.translate.bleu_score import corpus_bleu
```

```
from keras.models import Sequential

from keras.layers import LSTM, Embedding, TimeDistributed, RepeatVector, Activation

from keras.layers.core import Layer, Dense

from keras.optimizers import Adam, RMSprop

from keras.layers.wrappers import Bidirectional



# define the CNN model

def defineCNNmodel():

        model = InceptionV3()

        model.layers.pop()

        model = Model(inputs=model.inputs, outputs=model.layers[-1].output)

            #print(model.summary())

        return model


# define the RNN model

def defineRNNmodel(vocab_size, max_len):

    embedding_size = 300

    # Input dimension is 2048 since we will feed it the encoded version of the image.

    image_model = Sequential([

        Dense(embedding_size, input_shape=(2048,), activation='relu'),

        RepeatVector(max_len)

    ])

    # Since we are going to predict the next word using the previous words(length of previous
words changes with every iteration over the caption), we have to set return_sequences = True.
```

```
    caption_model  = Sequential([
        Embedding(vocab_size, embedding_size, input_length=max_len),
        LSTM(256, return_sequences=True),
        TimeDistributed(Dense(300))
    ])
    # Merging the models and creating a softmax classifier
    final_model = Sequential([
        keras.layers.Concatenate([image_model, caption_model], mode='concat',
concat_axis=1),
        Bidirectional(LSTM(256, return_sequences=False)),
        Dense(vocab_size),
        Activation('softmax')
    ])
    final_model.compile(loss='categorical_crossentropy', optimizer=RMSprop(),
metrics=['accuracy'])
    final_model.summary()
    return final_model


# map an integer to a word
def word_for_id(integer, tokenizer):
        for word, index in tokenizer.word_index.items():
                if index == integer:
                        return word
        return None
```

generate a description for an image, given a pre-trained model and a tokenizer to map integer back to word

```
def generate_desc(model, tokenizer, photo, max_length):
	# seed the generation process
	in_text = 'startseq'
	# iterate over the whole length of the sequence
	for i in range(max_length):
		# integer encode input sequence
		sequence = tokenizer.texts_to_sequences([in_text])[0]
		# pad input
		sequence = pad_sequences([sequence], maxlen=max_length)
		# predict next word
		yhat = model.predict([photo,sequence], verbose=0)
		# convert probability to integer
		yhat = argmax(yhat)
		# map integer to word
		word = word_for_id(yhat, tokenizer)
		# stop if we cannot map the word
		if word is None:
			break
		# append as input for generating the next word
		in_text += ' ' + word
		# stop if we predict the end of the sequence
		if word == 'endseq':
```

```
                break
        return in_text


def evaluate_model(model, photos, descriptions, tokenizer, max_length):
        actual, predicted = list(), list()

        for key, desc_list in descriptions.items():
                yhat = generate_desc(model, tokenizer, photos[key], max_length)
                references = [d.split() for d in desc_list]
                actual.append(references)
                predicted.append(yhat.split())

        print('BLEU-1: %f' % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
        print('BLEU-2: %f' % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
        print('BLEU-3: %f' % corpus_bleu(actual, predicted, weights=(0.3, 0.3, 0.3, 0)))
        print('BLEU-4: %f' % corpus_bleu(actual, predicted, weights=(0.25, 0.25, 0.25, 
0.25)))
```

5.1.5.Generating descriptions for trained images:

The code for generating descriptions for the trained images from the Flickr8k dataset is shown below:

train_val.py

```
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)
```

5.1.5.Generating descriptions for test images:

The code for generating descriptions for the test images from the Flickr8k dataset is shown below:

test.py

```
from pickle import load

from model import *

from keras.models import load_model

import matplotlib.pyplot as plt

import numpy as np

from PIL import Image




# extract features from each photo in the directory


def extract_features(filename):

        model = defineCNNmodel()

            # load the photo

        image = load_img(filename, target_size=(224, 224))

            # convert the image pixels to a numpy array

        image = img_to_array(image)

            # reshape data for the model

        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))

            # prepare the image for the VGG model

        image = preprocess_input(image)

            # get features

        feature = model.predict(image, verbose=0)
```

```
        return feature



# load the tokenizer

tokenizer_path = 'model_data/tokenizer.pkl'

tokenizer = load(open(tokenizer_path, 'rb'))


# pre-define the max sequence length (from training)


max_length = 34


# load the model


model_path = 'model_data/model_19.h5'

model = load_model(model_path)


# load and prepare the photograph


test_path = 'test_data'

for image_file in os.listdir(test_path):

        try:

            image_type = imghdr.what(os.path.join(test_path, image_file))

            if not image_type:

                continue
```

```
        except IsADirectoryError:
            continue
image = extract_features(image_file)

# generate description

description = generate_desc(model, tokenizer, image, max_length)

# remove startseq and endseq

caption = 'Caption: ' + description.split()[1].capitalize()
for x in description.split()[2:len(description.split())-1]:
    caption = caption + ' ' + x
caption += '.'

# Show image and it's caption
pil_im = Image.open(image_file, 'r')
fig, ax = plt.subplots(figsize=(8, 8))
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
_ = ax.imshow(np.asarray(pil_im), interpolation='nearest')
_ = ax.set_title(caption,fontdict={'fontsize': '20','fontweight' : '40'}
```

5.1.7.Building the web interface:

The code required for building the interface of the deployable web application is shown in the below files:

index.html:

```
<!DOCTYPE html>

<html lang="es">

<head>

    <link rel="stylesheet" type="text/css" href="/static/style.css">

    <link href="static/css/landing-page.min.css" rel="stylesheet">

    <link href="static/vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet"
type="text/css">

    <link href="static/vendor/simple-line-icons/css/simple-line-icons.css" rel="stylesheet"
type="text/css">

    <link
href="https://fonts.googleapis.com/css?family=Lato:300,400,700,300italic,400italic,700italic
" rel="stylesheet" type="text/css">

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
alpha.6/css/bootstrap.min.css"

          integrity="sha384-
rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ"
crossorigin="anonymous">

    <script src="https://code.jquery.com/jquery-3.1.1.slim.min.js"

            integrity="sha384-
A7FZj7v+d/sdmMqp/nOQwliLvUsJfDHW+k9Omg/a/EheAdgtzNs3hpfag6Ed950n"
```

```
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/tether.min.js"
        integrity="sha384-
DztdAPBWPRXSA/3eYEEUWrWCy7G5KFbe8fFjk5JAIxUYHKkDx6Qin1DkWx51bBrb"
        crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/js/bootstrap.min.js"
        integrity="sha384-
vBWWzlZJ8ea9aCX4pEW3rVHjgjt7zpkNpZk+02D9phzyeVkE+jo0ieGizqPLForn"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"
type="text/javascript"></script>
</head>
<body>
  <nav class="navbar fixed-top navbar-toggleable-md navbar-inverse bg-warning">
     <button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <h4><ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="btn btn-primary nav-link" href="http://localhost:5000"><b>DETECTION
</b><span class="sr-only">(current)</span></a>
        </li>  
        <li class="nav-item">
```

```
<a class="btn btn-success nav-link"
href="http://localhost:3000"><b>CAPTIONING</b><span class="sr-
only">(current)</span></a>

</li>  

<li class="nav-item">

<h3><strong><a class="nav-link text-primary"
align="center">  &nbspOBJECT DETECTION AND CAPTION GENERATION
FROM IMAGES</a></strong></h3>

</li>   

<li class="nav-item">


<a class="btn btn-danger nav-link" href="#speakers"><b>GITHUB</b><span class="sr-
only">(current)</span></a>

</li>  

<li class="nav-item">


<a class="btn btn-danger nav-link" href="#schedule"><b>RESOURCES</b><span
class="sr-only">(current)</span></a>

</li>   

<li class="nav-item">


<a class="btn btn-danger nav-link"
href="http://www.yaswanthpalaghat.site"><b>DEVELOPERS</b><span class="sr-
only">(current)</span></a></h4>

</li>

</ul>
```

```
    </div>
  </nav><br><br></h4>
<div class="jumbotron bg-primary">
<div class="container-fluid">
    <header class="masthead text-white text-center">
    <div class="overlay"></div>

        <div class="row">
            <div class="col-xl-9 mx-auto">
             <h1 class="mb-5">UPLOAD YOUR IMAGE</h1>
            </div>
            <div class="col-md-10 col-lg-8 col-xl-7 mx-auto">

 <form action="upload" method="post" enctype="multipart/form-data">
                <div class="form-row">
            <div class="col-12 col-md-9 mb-2 mb-md-0">
                <input class="btn btn-primary btn-lg" type="file" name="file"
id="inputFile1"><br/><br/>
            </div>
            <div class="col-12 col-md-3">
                <input class="btn btn-success btn-lg" type="submit" value="Upload">
             </div>
            </div><br>
            <h3 class="bg-warning"><STRONG>{{image_caption}}</STRONG></h3>
```

```
          <div class="col-lg-8">
              <img id="imagen"><br/>
          </div>
        </div>



      </form>


</header>
</div>
</div>
</body>
<script>

 function init() {
      var inputFile = document.getElementById('inputFile1');
      inputFile.addEventListener('change', mostrarImagen, false);
   }

  function mostrarImagen(event) {
      var file = event.target.files[0];
      var reader = new FileReader();
      reader.onload = function (event) {
          var img = document.getElementById('imagen');
```

```
            img.src = event.target.result;
            img.width = 500;
            img.height = 500;
        }
        reader.readAsDataURL(file);
    }
    window.addEventListener('load', init, false);
</script>
</html>
```

style.css

```
body,
html {
  width: 100%;
  height: 100%;
}

body {
  font-family: 'Source Sans Pro';
}

.btn-xl {
  padding: 1.25rem 2.5rem;
```

```
}

.content-section {
  padding-top: 7.5rem;
  padding-bottom: 7.5rem;
}

.content-section-heading h2 {
  font-size: 3rem;
}

.content-section-heading h3 {
  font-size: 1rem;
  text-transform: uppercase;
}

h1,
h2,
h3,
h4,
h5,
h6 {
  font-weight: 700;
}
```

```
.text-faded {
  color: rgba(255, 255, 255, 0.7);
}

/* Map */
.map {
  height: 30rem;
}

@media (max-width: 992px) {
  .map {
    height: 75%;
  }
}

.map iframe {
  pointer-events: none;
}

.scroll-to-top {
  position: fixed;
  right: 15px;
  bottom: 15px;
```

```
  display: none;
  width: 50px;
  height: 50px;
  text-align: center;
  color: white;
  background: rgba(52, 58, 64, 0.5);
  line-height: 45px;
}

.scroll-to-top:focus, .scroll-to-top:hover {
  color: white;
}

.scroll-to-top:hover {
  background: #343a40;
}

.scroll-to-top i {
  font-weight: 800;
}

.masthead {
  min-height: 10rem;
  position: relative;
```

```
  display: table;
  width: 100%;
  height: auto;
  padding-top: 2rem;
  padding-bottom: 8rem;
  background: -webkit-gradient(linear, left top, right top, from(rgba(255, 255, 255, 0.1)), to(rgba(255, 255, 255, 0.1))), url("../img/bg-masthead.jpg");
  background: linear-gradient(90deg, rgba(255, 255, 255, 0.1) 0%, rgba(255, 255, 255, 0.1) 100%), url("../img/bg-masthead.jpg");
  background-position: center center;
  background-repeat: no-repeat;
  background-size: cover;
  -webkit-background-size: cover;
  -moz-background-size: cover;
  -o-background-size: cover;
  background-size: cover;
}

.masthead h1 {
  font-size: 4rem;
  margin: 0;
  padding: 0;
}

@media (min-width: 992px) {
```

```
  .masthead {
    height: 100vh;
  }
  .masthead h1 {
    font-size: 5.5rem;
  }
}

/* Side Menu */
#sidebar-wrapper {
  position: fixed;
  z-index: 2;
  right: 0;
  width: 250px;
  height: 100%;
  -webkit-transition: all 0.4s ease 0s;
  transition: all 0.4s ease 0s;
  -webkit-transform: translateX(250px);
  transform: translateX(250px);
  background: #1D809F;
  border-left: 1px solid rgba(255, 255, 255, 0.1);
}

.sidebar-nav {
```

```
  position: absolute;
  top: 0;
  width: 250px;
  margin: 0;
  padding: 0;
  list-style: none;
}

.sidebar-nav li.sidebar-nav-item a {
  display: block;
  text-decoration: none;
  color: #fff;
  padding: 15px;
}

.sidebar-nav li a:hover {
  text-decoration: none;
  color: #fff;
  background: rgba(255, 255, 255, 0.2);
}

.sidebar-nav li a:active,
.sidebar-nav li a:focus {
  text-decoration: none;
```

```
}

.sidebar-nav > .sidebar-brand {
  font-size: 1.2rem;
  background: rgba(52, 58, 64, 0.1);
  height: 80px;
  line-height: 50px;
  padding-top: 15px;
  padding-bottom: 15px;
  padding-left: 15px;
}

.sidebar-nav > .sidebar-brand a {
  color: #fff;
}

.sidebar-nav > .sidebar-brand a:hover {
  color: #fff;
  background: none;
}

#sidebar-wrapper.active {
  right: 250px;
  width: 250px;
```

```
  -webkit-transition: all 0.4s ease 0s;
  transition: all 0.4s ease 0s;
}

.menu-toggle {
  position: fixed;
  right: 15px;
  top: 15px;
  width: 50px;
  height: 50px;
  text-align: center;
  color: #fff;
  background: rgba(52, 58, 64, 0.5);
  line-height: 50px;
  z-index: 999;
}

.menu-toggle:focus, .menu-toggle:hover {
  color: #fff;
}

.menu-toggle:hover {
  background: #343a40;
}
```

```
.service-icon {
  background-color: #fff;
  color: #1D809F;
  height: 7rem;
  width: 7rem;
  display: block;
  line-height: 7.5rem;
  font-size: 2.25rem;
  -webkit-box-shadow: 0 3px 3px 0 rgba(0, 0, 0, 0.1);
  box-shadow: 0 3px 3px 0 rgba(0, 0, 0, 0.1);
}

.callout {
  padding: 15rem 0;
  background: -webkit-gradient(linear, left top, right top, from(rgba(255, 255, 255, 0.1)), to(rgba(255, 255, 255, 0.1))), url("../img/bg-callout.jpg");
  background: linear-gradient(90deg, rgba(255, 255, 255, 0.1) 0%, rgba(255, 255, 255, 0.1) 100%), url("../img/bg-callout.jpg");
  background-position: center center;
  background-repeat: no-repeat;
  background-size: cover;
}

.callout h2 {
```

```
  font-size: 3.5rem;
  font-weight: 700;
  display: block;
  max-width: 30rem;
}

.portfolio-item {
  display: block;
  position: relative;
  overflow: hidden;
  max-width: 530px;
  margin: auto auto 1rem;
}

.portfolio-item .caption {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  height: 100%;
  width: 100%;
  background-color: rgba(33, 37, 41, 0.2);
  position: absolute;
  top: 0;
  bottom: 0;
```

```
  z-index: 1;
}

.portfolio-item .caption .caption-content {
  color: #fff;
  margin: auto 2rem 2rem;
}

.portfolio-item .caption .caption-content h2 {
  font-size: 0.8rem;
  text-transform: uppercase;
}

.portfolio-item .caption .caption-content p {
  font-weight: 300;
  font-size: 1.2rem;
}

@media (min-width: 992px) {
  .portfolio-item {
    max-width: none;
    margin: 0;
  }
  .portfolio-item .caption {
```

```
  -webkit-transition: -webkit-clip-path 0.25s ease-out, background-color 0.7s;
  -webkit-clip-path: inset(0px);
  clip-path: inset(0px);
}
.portfolio-item .caption .caption-content {
  -webkit-transition: opacity 0.25s;
  transition: opacity 0.25s;
  margin-left: 5rem;
  margin-right: 5rem;
  margin-bottom: 5rem;
}
.portfolio-item img {
  -webkit-transition: -webkit-clip-path 0.25s ease-out;
  -webkit-clip-path: inset(-1px);
  clip-path: inset(-1px);
}
.portfolio-item:hover img {
  -webkit-clip-path: inset(2rem);
  clip-path: inset(2rem);
}
.portfolio-item:hover .caption {
  background-color: rgba(29, 128, 159, 0.9);
  -webkit-clip-path: inset(2rem);
  clip-path: inset(2rem);
```

```
  }
}

footer.footer {
  padding-top: 5rem;
  padding-bottom: 5rem;
}

footer.footer .social-link {
  display: block;
  height: 4rem;
  width: 4rem;
  line-height: 4.3rem;
  font-size: 1.5rem;
  background-color: #1D809F;
  -webkit-transition: background-color 0.15s ease-in-out;
  transition: background-color 0.15s ease-in-out;
  -webkit-box-shadow: 0 3px 3px 0 rgba(0, 0, 0, 0.1);
  box-shadow: 0 3px 3px 0 rgba(0, 0, 0, 0.1);
}

footer.footer .social-link:hover {
  background-color: #155d74;
  text-decoration: none;
```

```
}

a {
  color: #1D809F;
}

a:hover, a:focus, a:active {
  color: #155d74;
}

.btn-primary {
  background-color: #1D809F !important;
  border-color: #1D809F !important;
  color: #fff !important;
}

.btn-primary:hover, .btn-primary:focus, .btn-primary:active {
  background-color: #155d74 !important;
  border-color: #155d74 !important;
}

.btn-secondary {
  background-color: #ecb807 !important;
  border-color: #ecb807 !important;
```

```
  color: #fff !important;
}

.btn-secondary:hover, .btn-secondary:focus, .btn-secondary:active {
  background-color: #ba9106 !important;
  border-color: #ba9106 !important;
}

.btn-dark {
  color: #fff !important;
}

.btn {
  -webkit-box-shadow: 0px 3px 3px 0px rgba(0, 0, 0, 0.1);
  box-shadow: 0px 3px 3px 0px rgba(0, 0, 0, 0.1);
  font-weight: 700;
}

.bg-primary {
  background-color: #1D809F !important;
}

.text-primary {
  color: #1D809F !important;
```

```
}

.text-secondary {
  color: #ecb807 !important;
}
```

5.1.8. Caption Generation:

The code for generating captions using Flask interface which allows user to upload the images and retrieve captions is shown below:

cap.py

```
import requests

import os

from flask import Flask

from flask import render_template

from flask import request

from captionbot import CaptionBot

from flask_bootstrap import Bootstrap

from werkzeug import secure_filename

import numpy as np

import os
```

```
import six.moves.urllib as urllib

import sys

import tensorflow as tf

from collections import defaultdict

from flask_gtts import gtts

from gtts import gTTS

from playsound import playsound

from io import StringIO

from PIL import Image

sys.path.append("..")

from utils import label_map_util

from utils import visualization_utils as vis_util



language = 'en'


c=CaptionBot()


app = Flask(__name__)

bootstrap = Bootstrap(app)


UPLOAD_FOLDER = os.path.basename('/uploads')
```

```
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

@app.route('/')

def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])

def upload_file():
  file = request.files['file']

  f = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)

  # add your custom code to check that the uploaded file is a valid image and not a malicious

file (out-of-scope for this post)
  file.save(f)
  print(f)
  image_caption = c.file_caption(f)
  return render_template('index.html', image_caption=image_caption)

@app.route("/success", methods = ['POST','GET'])
```

```
def success():
    text = c.file_caption(f)
    tts = gTTS(text=text, lang='en')
    return render_template("success.html",value = text)
    tts.save("text.mp3")


if __name__ == '__main__':
  app.run(debug=True, port=3000)
```

6. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

6.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

6.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.4 Performance Testing

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

6.5 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

6.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the auditor after the data is received by the cloud server
- The auditors audit operation is done only when there is a request from user
- The Status of data information on the cloud is viewed only by the cloud server

6.7 Test Cases

Test case Number	1
Test case Name	Unit Test Case for uploading images by the user.
Feature to be tested	Image upload
Description	When a new image is uploaded by the user, the model verifies the image format.
Sample Input	Example.jpg, Example.jpeg, Example.png
Expected output	The image should be uploaded successfully without any errors and the uploaded image should satisfy the mentioned format described above.
Actual output	If the image uploaded is invalid, the prompt shows "Please upload valid image".
Remarks	Success

Table 1: Unit Test Case for Image Upload

Test case Number	2
Test case Name	Unit Test Case for caption generation
Feature to be tested	Caption generation
Description	When an image is uploaded by the user using the web interface, the caption should be generated after clicking the upload button.
Sample Input	Caption of the image.
Expected output	The caption of the image, which descibes the image based on the objects detected from the image.
Actual output	If there is no image selected, it prompts "Please upload the image to generate caption"
Remarks	Successful

Table 2: Unit Test Case for Caption generation

Sr-no	Test-case scenario	Expected result	Actual result
1	Invalid image	Prompt for "Please upload Valid image".	Focus on image format .
2	No image is selected	Prompt for "Please upload an image".	Focus on caption. .

Table 3: Acceptance Test Case for generating captions.

7. RESULTS

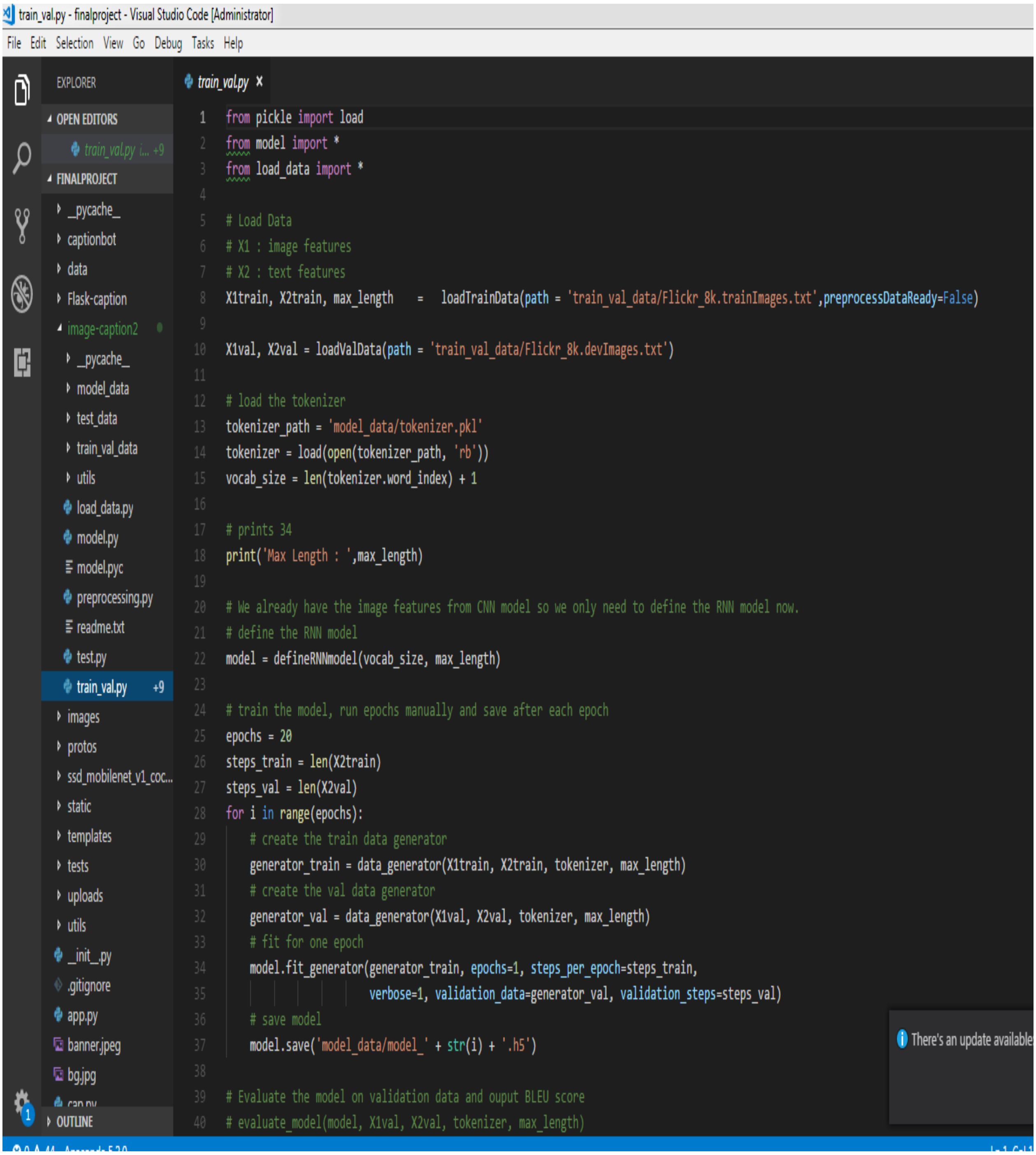


fig 3.1: Visual Studio Code Interface

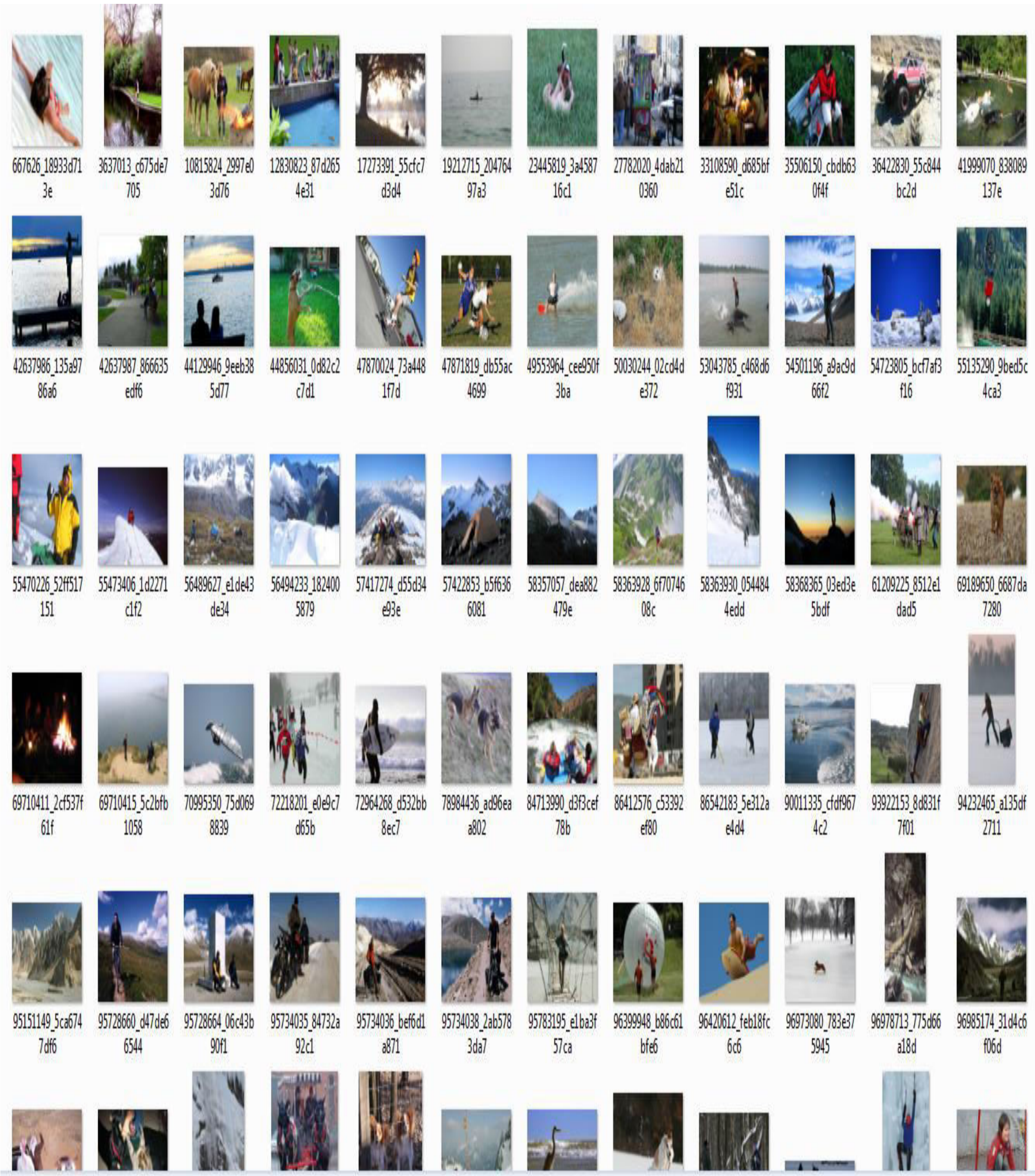


Fig 3.2: Flickr8k Dataset

```
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 3.3: Descriptions

Fig 3.4: Web Application Interface

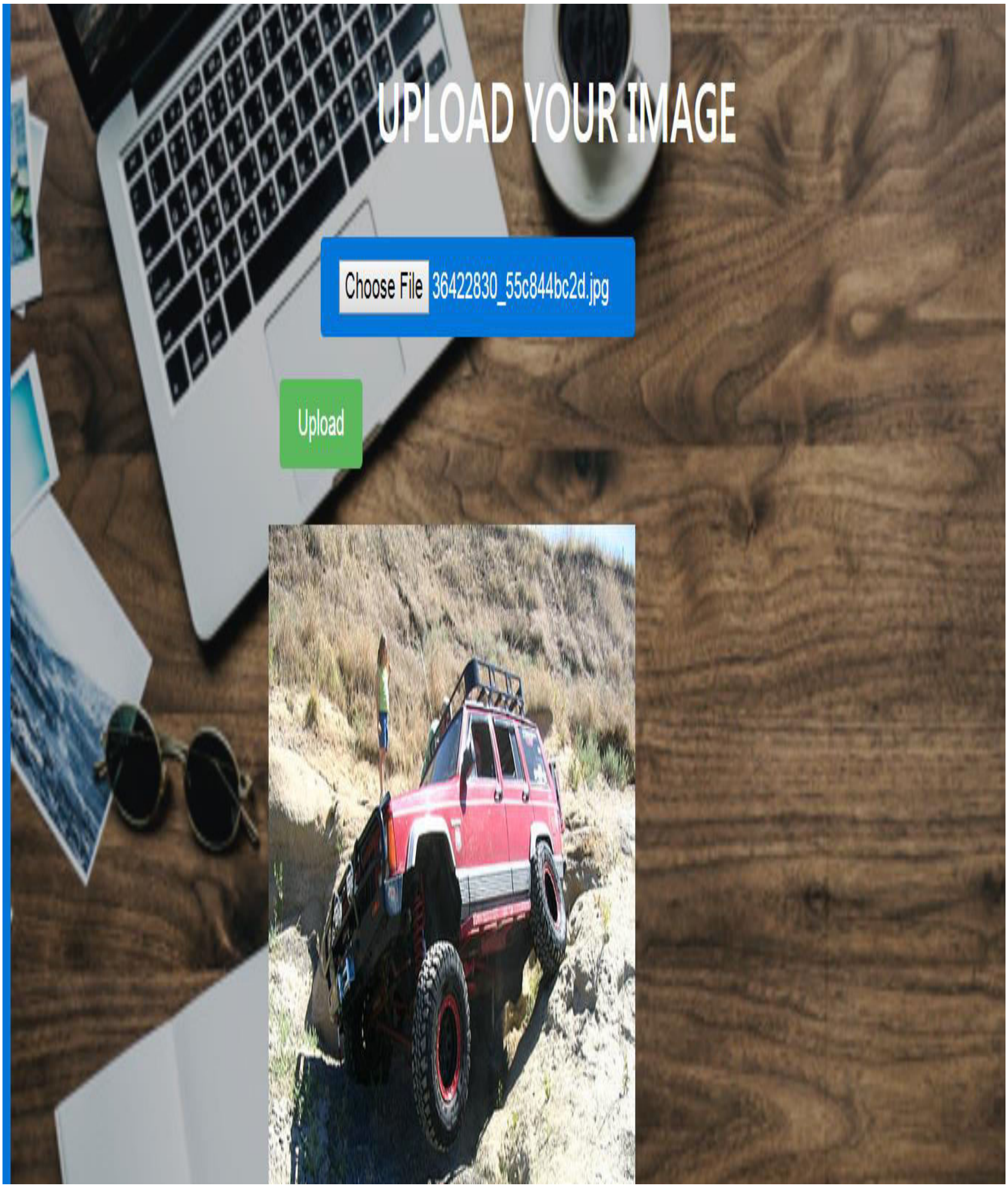


Fig 3.5: User Image Upload

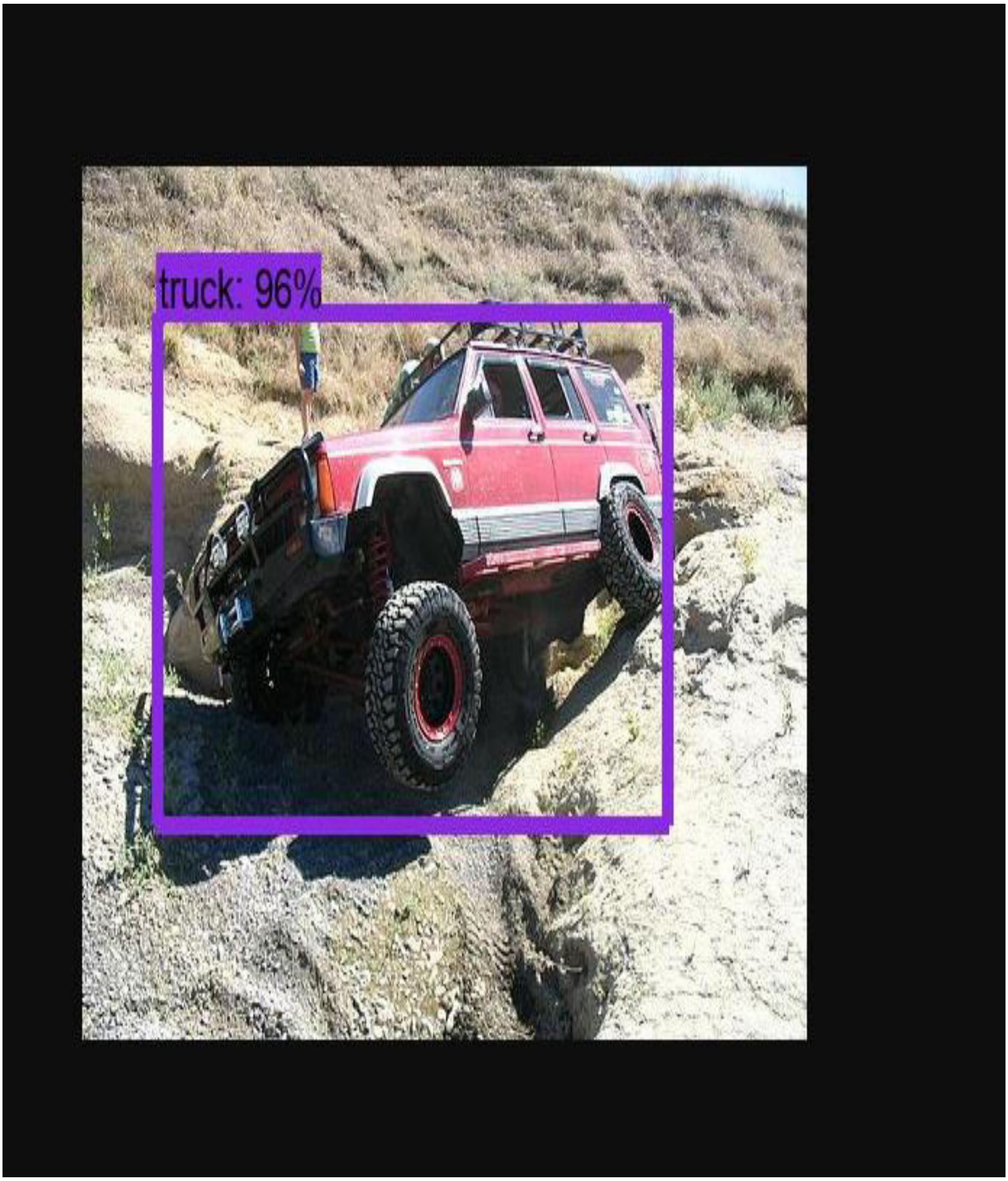


Fig 3.6: Object Detection

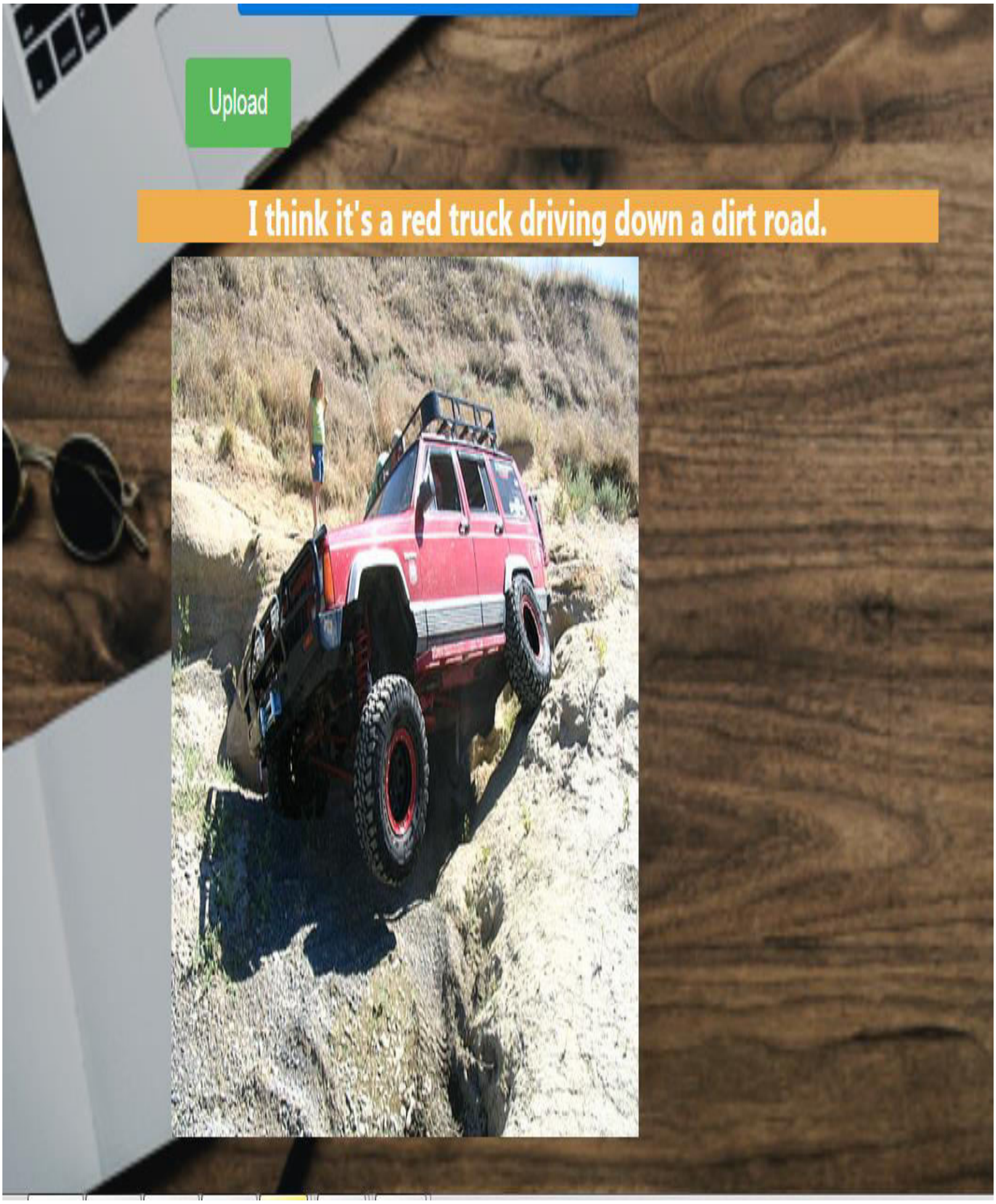


Fig 3.7: Caption Generation Example1

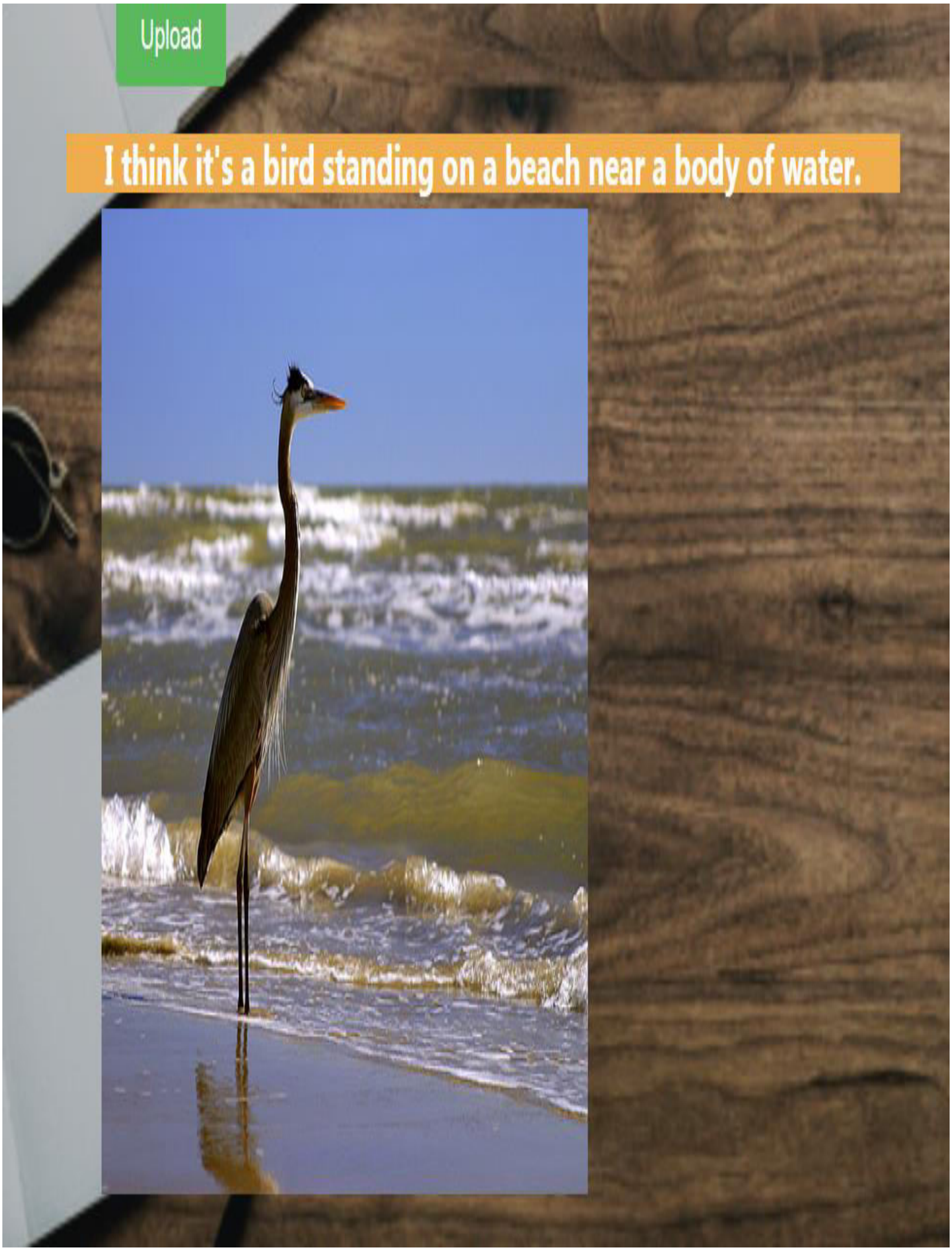


Fig 3.8: Caption Generation Example2

8. 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.

9. BIBLIOGRAPHY

References for this project development were taken from the following books and websites:

[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] Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, Trevor Darrell, Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, et al. 2016. Deep compositional captioning: Describing novel object categories without paired training data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[7] 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).

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

[9] 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.

[10] 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.

[11] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. Journal of machine learning research 3, Feb, 1137–1155.

[12] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. Computational linguistics 22, 1 (1996), 39–71.

[13] Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, Barbara Plank, et al. 2016. Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures. Journal of Artificial Intelligence Research (JAIR) 55, 409–442.

[14] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. Journal of machine Learning research 3, Jan (2003), 993–1022.

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

[16] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data. AcM, 1247–1250.

[17] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. 1992. A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory. ACM, 144–152.

[18] Andreas Bulling, Jamie A Ward, Hans Gellersen, and Gerhard Troster. 2011. Eye movement analysis for activity recognition using electrooculography. IEEE transactions on pattern analysis and machine intelligence 33, 4 (2011), 741–753.

[19] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. 2011. Learning photographic global tonal adjustment with a database of input/output image pairs. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 97–104.

[20] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluation the Role of Bleu in Machine Translation Research.. In EACL, Vol. 6. 249–256.

[21] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, and Tat-Seng Chua. 2017. SCA-CNN: Spatial and Channel-wise Attention in Convolutional Networks for Image Captioning. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 6298–6306.

[22] Tseng-Hung Chen, Yuan-Hong Liao, Ching-Yao Chuang, Wan-Ting Hsu, Jianlong Fu, and Min Sun. 2017. Show, Adapt and Tell: Adversarial Training of Cross-domain Image Captioner. In The IEEE International Conference on Computer Vision (ICCV), Vol. 2.

[23] Xinlei Chen and C Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2422–2431.

[24] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In Association for Computational Linguistics. 103–111.

[25] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[26] Bo Dai, Dahua Lin, Raquel Urtasun, and Sanja Fidler. 2017. Towards Diverse and Natural Image Descriptions via a Conditional GAN. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 2989–2998.

[27] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1. IEEE, 886–893.

[28] Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D Manning. 2006. Generating typed dependency parses from phrase structure parses. In Proceedings of LREC, Vol. 6. Genoa Italy, 449–454.

[29] Etienne Denoual and Yves Lepage. 2005. BLEU in characters: towards automatic MT evaluation in languages without word delimiters. In Companion Volume to the Proceedings of the Second International Joint Conference on Natural Language Processing. 81–86.

[30] Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. arXiv preprint arXiv:1505.01809 (2015).

[31] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2625–2634.

[32] Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 1292–1302.

[33] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr DollÃąr, Jianfeng Gao, Xiaodong He, Margaret Mitchell, and John C Platt. 2015. From captions to visual concepts and back. In Proceedings of the IEEE conference on computer vision and pattern recognition. 1473–1482.

[34] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In European conference on computer vision. Springer, 15–29.

[35] Alireza Fathi, Yin Li, and James M Rehg. 2012. Learning to recognize daily actions using gaze. In European Conference on Computer Vision. Springer, 314–327.

[36] William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the _. arXiv preprint arXiv:1801.07736.

[37] Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In Proceedings of the 2013 conference on empirical methods in natural language processing. 1914–1925.

[38] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In Advances in neural information processing systems. 2121–2129.

[39] Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. 2017. Stylenet: Generating attractive visual captions with styles. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3137–3146.

[40] Chuang Gan, Tianbao Yang, and Boqing Gong. 2016. Learning attributes equals multi-source domain generalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 87–97.

[41] Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. 2017. Semantic compositional networks for visual captioning. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 1141–1150.

[42] Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. arXiv preprint arXiv:1611.02344.

[43] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122.

[44] Ross Girshick. 2015. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision. 1440–1448.

[45] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition. 580–587.

[46] Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In Proceedings of the 2010 conference on empirical methods in natural language processing. Association for Computational Linguistics, 410–419.

[47] Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. 2014. Improving image-sentence embeddings using large weakly annotated photo collections. In European Conference on Computer Vision. Springer, 529–545.

[48] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Advances in neural information processing systems. 2672–2680.

[49] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. In Proceedings of Machine Learning Research. 1462–1471.

[50] Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. 2006. The iapr tc-12 benchmark: A new evaluation resource for visual information systems. In International workshop ontoImage, Vol. 5. 10.

[51] Jiuxiang Gu, Gang Wang, Jianfei Cai, and Tsuhan Chen. 2017. An empirical study of language cnn for image captioning. In Proceedings of the International Conference on Computer Vision (ICCV). 1231–1240.

[52] Yahong Han and Guang Li. 2015. Describing images with hierarchical concepts and object class localization. In Proceedings of the 5th ACM on International Conference on Multimedia Retrieval. ACM, 251–258.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.

[54] Sepp Hochreiter and JÃijrgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.

[55] Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. Journal of Artificial Intelligence Research 47 (2013), 853–899.

[56] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision (CVPR). 5967–5976.

[57] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. 2015. Spatial transformer networks. In Advances in Neural Information Processing Systems. 2017–2025.

[58] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In International Conference on Learning Representations (ICLR).

[59] Xu Jia, Efstratios Gavves, Basura Fernando, and Tinne Tuytelaars. 2015. Guiding the long-short term memory model for image caption generation. In Proceedings of the IEEE International Conference on Computer Vision. 2407–2415.

[60] Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. 2015. Salicon: Saliency in context. In Proceedings of the IEEE conference on computer vision and pattern recognition. 1072–1080.

[61] Junqi Jin, Kun Fu, Runpeng Cui, Fei Sha, and Changshui Zhang. 2015. Aligning where to see and what to tell: image caption with region-based attention and scene factorization. arXiv preprint arXiv:1506.06272.

[62] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 4565–4574.

[63] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. 2015. Image retrieval using scene graphs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3668–3678.

[64] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410 (2016).

[65] Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3128–3137.

[66] Andrej Karpathy, Armand Joulin, and Fei Fei F Li. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In Advances in neural information processing systems. 1889–1897.

[67] S Karthikeyan, Vignesh Jagadeesh, Renuka Shenoy, Miguel Ecksteinz, and BS Manjunath. 2013. From where and how to what we see. In Proceedings of the IEEE International Conference on Computer Vision. 625–632.

[68] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes.. In EMNLP. 787–798.

[69] Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In Proceedings of the 31st International Conference on Machine Learning (ICML-14). 595–603.

[70] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. In Workshop on Neural Information Processing Systems (NIPS)).

[71] Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In Advances in neural information processing systems. 1008–1014.

[72] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision 123, 1 (2017), 32–73.

[73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems. 1097–1105.

[74] Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2011. Baby talk: Understanding and generating image descriptions. In Proceedings of the 24th CVPR. Citeseer.

[75] Akshi Kumar and Shivali Goel. 2017. A survey of evolution of image captioning techniques. International Journal of Hybrid Intelligent Systems Preprint, 1–19.

[76] Polina Kuznetsova, Vicente Ordonez, Alexander C Berg, Tamara L Berg, and Yejin Choi. 2012. Collective generation of natural image descriptions. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. Association for Computational Linguistics, 359–368.

[77] Polina Kuznetsova, Vicente Ordonez, Tamara L Berg, and Yejin Choi. 2014. TREETALK: Composition and Compression of Trees for Image Descriptions. TACL 2, 10 (2014), 351–362.

[78] RÃľmi Lebret, Pedro O Pinheiro, and Ronan Collobert. 2015. Simple image description generator via a linear phrase-based approach. Workshop on International Conference on Learning Representations (ICLR).

[79] Yann LeCun, LÃľon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86, 11 (1998), 2278–2324.

[80] Siming Li, Girish Kulkarni, Tamara L Berg, Alexander C Berg, and Yejin Choi. 2011. Composing simple image descriptions using web-scale n-grams. In Proceedings of the Fifteenth Conference on Computational Natural Language Learning. Association for Computational Linguistics, 220–228.

[81] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out: Proceedings of the ACL-04 workshop, Vol. 8. Barcelona, Spain.

[82] Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 605.

[83] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr DollÃąr, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In European conference on computer vision. Springer, 740–755.

[84] Chenxi Liu, Junhua Mao, Fei Sha, and Alan L Yuille. 2017. Attention Correctness in Neural Image Captioning. In AAAI. 4176–4182.

[85] Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2017. Improved image captioning via policy gradient optimization of spider. In Proceedings of the IEEE Internationa Conference on Computer Vision (ICCV), Vol. 3. 873–881.

[86] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3431–3440.

[87] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. International journal of computer vision 60, 2 (2004), 91–110.

[88] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via A visual sentinel for image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 3242–3250.

[89] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In International Conference on Learning Representations (ICLR).

[90] Shubo Ma and Yahong Han. 2016. Describing images by feeding LSTM with structural words. In Multimedia and Expo (ICME), 2016 IEEE International Conference on. IEEE, 1–6.

[91] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In International Conference on Learning Representations (ICLR).

[92] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition. 11–20.

[93] Junhua Mao, Xu Wei, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L Yuille. 2015. Learning like a child: Fast novel visual concept learning from sentence descriptions of images. In Proceedings of the IEEE International Conference on Computer Vision. 2533–2541.

[94] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). In International Conference on Learning Representations (ICLR).

[95] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L Yuille. 2014. Explain images with multimodal recurrent neural networks. arXiv preprint arXiv:1410.1090 (2014).

[96] Oded Maron and TomÃąs Lozano-PÃľrez. 1998. A framework for multiple-instance learning. In Advances in neural information processing systems. 570–576.

[97] Alexander Patrick Mathews, Lexing Xie, and Xuming He. 2016. SentiCap: Generating Image Descriptions with Sentiments.. In AAAI. 3574–3580.

[98] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[99] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocky,` and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association.

[100] Ajay K Mishra, Yiannis Aloimonos, Loong Fah Cheong, and Ashraf Kassim. 2012. Active visual segmentation. IEEE transactions on pattern analysis and machine intelligence 34, 4 (2012), 639–653.

[101] Margaret Mitchell, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, Kota Yamaguchi, Tamara Berg, Karl Stratos, and Hal Daumé III. 2012. Midge: Generating image descriptions from computer vision detections. In

Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 747–756.

[102] Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2010. Natural reference to objects in a visual domain. In

Proceedings of the 6th international natural language generation conference. Association for Computational Linguistics, 95–104.

[103] Margaret Mitchell, Kees Van Deemter, and Ehud Reiter. 2013. Generating Expressions that Refer to Visible Objects.. In HLT-NAACL. 1174–1184.

[104] Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In Proceedings of the 24th international conference on Machine learning. ACM, 641–648.

[105] Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In Proceedings of the 24th international conference on Machine learning. ACM, 641–648.

[106] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1. Association for Computational Linguistics, 160–167.

[107] Timo Ojala, Matti PietikÃďinen, and Topi MÃďenpÃďÃď. 2000. Gray scale and rotation invariant texture classification with local binary patterns. In European Conference on Computer Vision. Springer, 404–420.

[108] Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In Advances in Neural Information Processing Systems. 1143–1151.

[109] Dim P Papadopoulos, Alasdair DF Clarke, Frank Keller, and Vittorio Ferrari. 2014. Training object class detectors from eye tracking data. In European Conference on Computer Vision. Springer, 361–376.

[110] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting

on association for computational linguistics. Association for Computational Linguistics, 311–318.

[111] Cesc Chunseong Park, Byeongchang Kim, and Gunhee Kim. 2017. Attend to You: Personalized Image Captioning with Context Sequence Memory Networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 6432–6440.

[112] Marco Pedersoli, Thomas Lucas, Cordelia Schmid, and Jakob Verbeek. 2017. Areas of Attention for Image Captioning. In Proceedings of the IEEE international conference on computer vision. 1251–1259.

[113] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In Proceedings of the IEEE international conference on computer vision. 2641–2649.

[114] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In International Conference on learning Representations (ICLR).

[115] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In Proceedings of Machine Learning Research, Vol. 48. 1060–1069.

[116] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems. 91–99.

[117] Zhou Ren, Hailin Jin, Zhe Lin, Chen Fang, and Alan Yuille. 2015. Multi-instance visual-semantic embedding. arXiv preprint arXiv:1512.06963 (2015).

[118] Zhou Ren, Hailin Jin, Zhe Lin, Chen Fang, and Alan Yuille. 2016. Joint image-text representation by gaussian visual-semantic embedding. In Proceedings of the 2016 ACM on Multimedia Conference. ACM, 207–211.

[119] Zhou Ren, Xiaoyu Wang, Ning Zhang, Xutao Lv, and Li-Jia Li. 2017. Deep Reinforcement Learning-based Image Captioning with Embedding Reward. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 1151–1159.

[120] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 1179–1195.

[121] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. Journal of documentation 60, 5 (2004), 503–520.

[122] Hosnieh Sattar, Sabine Muller, Mario Fritz, and Andreas Bulling. 2015. Prediction of search targets from fixations in open-world settings. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 981–990.

[123] Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. 2015. Generating se-mantically precise scene graphs from textual descriptions for improved image retrieval. In Proceedings of the fourth workshop on vision and language, Vol. 2.

[124] Karthikeyan Shanmuga Vadivel, Thuyen Ngo, Miguel Eckstein, and BS Manjunath. 2015. Eye tracking assisted extraction of attentionally important objects from videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3241–3250.

[125] Naeha Sharif, Lyndon White, Mohammed Bennamoun, and Syed Afaq Ali Shah. 2018. Learning-based Composite Metrics for Improved Caption Evaluation. In Proceedings of ACL 2018, Student Research Workshop. 14–20.

[126] Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, and Bernt Schiele. 2017. Speaking the Same Language: Matching Machine to Human Captions by

Adversarial Training. In IEEE International Conference on Computer Vision (ICCV). 4155–4164.

[127] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR).

[128] Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded composi-tional semantics for finding and describing images with sentences. Transactions of the Association for Computational Linguistics 2 (2014), 207–218.

[129] Yusuke Sugano and Andreas Bulling. 2016. Seeing with humans: Gaze-assisted neural image captioning. arXiv preprint arXiv:1608.05203 (2016).

[130] Chen Sun, Chuang Gan, and Ram Nevatia. 2015. Automatic concept discovery from parallel text and visual corpora. In Proceedings of the IEEE International Conference on Computer Vision. 2596–2604.

[131] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems. 3104–3112.

[132] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems. 1057–1063.

[133] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition. 1–9.

[134] Hamed R Tavakoli, Rakshith Shetty, Ali Borji, and Jorma Laaksonen. 2017. Paying Attention to Descriptions Generated by Image Captioning Models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2487– 2496.

[135] Kenneth Tran, Xiaodong He, Lei Zhang, Jian Sun, Cornelia Carapcea, Chris Thrasher, Chris Buehler, and Chris Sienkiewicz. 2016. Rich image captioning in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 49–56.

[136] Kees van Deemter, Ielka van der Sluis, and Albert Gatt. 2006. Building a semantically transparent corpus for the generation of referring expressions. In Proceedings of the Fourth International Natural Language Generation Conference. Association for Computational Linguistics, 130–132.

[137] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. 2016. Conditional image generation with pixelcnn decoders. In Advances in Neural Information Processing Systems. 4790–4798.

[138] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems. 5998–6008.

[139] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In Proceedings of the IEEE conference on computer vision and pattern recognition. 4566–4575.

[140] Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2017. Captioning images with diverse objects. In Proceedings of the IEEE conference on computer vision and pattern recognition. 1170–1178.

[141] Jette Viethen and Robert Dale. 2008. The use of spatial relations in referring expression generation. In Proceedings of the Fifth International Natural Language Generation Conference. Association for Computational Linguistics, 59–67.

[142] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition. 3156–3164.

[143] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2017. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. IEEE transactions on pattern analysis and machine intelligence 39, 4 (2017), 652–663.

[144] Cheng Wang, Haojin Yang, Christian Bartz, and Christoph Meinel. 2016. Image captioning with deep bidirectional LSTMs. In Proceedings of the 2016 ACM on Multimedia Conference. ACM, 988–997.

[145] Heng Wang, Zengchang Qin, and Tao Wan. 2018. Text Generation Based on Generative Adversarial Nets with Latent Variables. In Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 92–103.

[146] Minsi Wang, Li Song, Xiaokang Yang, and Chuanfei Luo. 2016. A parallel-fusion RNN-LSTM architecture for image caption generation. In Image Processing (ICIP), 2016 IEEE International Conference on. IEEE, 4448–4452.

[147] Qingzhong Wang and Antoni B Chan. 2018. CNN+ CNN: Convolutional Decoders for Image Captioning. arXiv preprint arXiv:1805.09019.

[148] Yufei Wang, Zhe Lin, Xiaohui Shen, Scott Cohen, and Garrison W Cottrell. 2017. Skeleton Key: Image Captioning by Skeleton-Attribute Decomposition. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 7378–7387.

[149] Qi Wu, Chunhua Shen, Anton van den Hengel, Lingqiao Liu, and Anthony Dick. 2015. Image captioning with an intermediate attributes layer. arXiv preprint arXiv:1506.01144 (2015).

[150] Qi Wu, Chunhua Shen, Peng Wang, Anthony Dick, and Anton van den Hengel. 2018. Image captioning and visual question answering based on attributes and external knowledge. IEEE transactions on pattern analysis and machine intelligence 40, 6, 1367–1381.

[151] Zhilin Yang Ye Yuan Yuexin Wu and Ruslan Salakhutdinov William W Cohen. 2016. Encode, Review, and Decode: Reviewer Module for Caption Generation. In 30th Conference on Neural Image Processing System(NIPS).

[152] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning. 2048–2057.

[153] Linjie Yang, Kevin Tang, Jianchao Yang, and Li-Jia Li. 2016. Dense Captioning with Joint Inference and Visual Context. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1978–1987.

[154] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. 2017. Incorporating copying mechanism in image captioning for learning novel objects. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 5263–5271.

[155] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. 2017. Boosting image captioning with attributes. In IEEE International Conference on Computer Vision (ICCV). 4904–4912.

[156] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 4651–4659.

[157] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu Seqgan. 2017. Sequence generative adversarial nets with policy gradient.. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence.

[158] Kiwon Yun, Yifan Peng, Dimitris Samaras, Gregory J Zelinsky, and Tamara L Berg. 2013. Exploring the role of gaze behavior and object detection in scene understanding. Frontiers in psychology 4 (2013).

[159] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In European conference on computer vision. Springer, 818–833.

[160] Gregory J Zelinsky. 2013. Understanding scene understanding. Frontiers in psychology 4 (2013).

[161] Li Zhang, Flood Sung, Feng Liu, Tao Xiang, Shaogang Gong, Yongxin Yang, and Timothy M Hospedales. 2017. Actor-critic sequence training for image captioning. arXiv preprint arXiv:1706.09601.