Mini Project Report on

TEXTUAL DESCRIPTION GENERATOR FOR A GIVEN IMAGE

Submitted by

Name of Student	Class	Roll No.
1. Unnati Shah	TE3	53
2. Aditi Tandel	TE3	59
3. Krutik Patel	TE4	46

Under the guidance of

Prof. Radhika Fulzele

DEPARTMENT OF COMPUTER ENGINEERING

SHAH AND ANCHOR KUTCHHI ENGINEERING COLLEGE

CHEMBUR, MUMBAI - 400088.

2020-2021

www.shahandanchor.com Tel: 022 2558 0854

Mahavir Education Trust's

SHAH & ANCHOR KUTCHHI ENGINEERING COLLEGE

Mahavir Education Trust Chowk, W.T. Patil Marg, Chembur, Mumbai 400 088

Affiliated to University of Mumbai, Approved by D.T.E. & A.I.C.T.E.

Awarded accreditation for Computer & Information Technology Engineering by NBA

(for 3 years w.e.f. 1st July, 2019)

Certificate

This is to certify that the report of the mini project entitled

Title of mini project

is a bonafide work of

Name of Student	Class	Roll No.
1. Unnati Shah	TE3	53
2. Aditi Tandel	TE3	59
3. Krutik Patel	TE4	46

submitted to the

UNIVERSITY OF MUMBAI

during semester VI

in

COMPUTER ENGINEERING DEPARTMENT

---------- ----------

(Prof. Uday Bhave)

Guide *I/c Head of Department*

Approval for Mini Project Report for T. E. Semester VI

This mini project report entitled "TEXTUAL DESCRIPTION GENERATOR FOR A GIVEN IMAGE" by Unnati Shah, Aditi Tandel, and Krutik Patel is approved for the partial fulfillment of the requirement for the completion of Semester VI.

Name and Sign of Internal Examiner _____

Name and Sign of External Examiner _____

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of Student	Class	Roll No.	Signature
1. Unnati Shah	TE3	53	Unnati Shah
2. Aditi Tandel	TE3	59	Aditi Tandel
3. Krutik Patel	TE4	46	Krutik Patel

Date:

Attendance Certificate

Date

To,
The Principal
Shah and Anchor Kutchhi Engineering College,
Chembur, Mumbai-88

Subject: Confirmation of Attendance

Respected Sir,

This is to certify that Third year (TE) students
Unnati Shah
Aditi Tandel
Krutik Patel

have duly attended the sessions on the day allotted to them during the period from _____to_____ for performing the Mini Project titled Textual Description Generator.

They were punctual and regular in their attendance. Following is the detailed record of the student's attendance.

Attendance Record:

Date	Unnati Shah	Aditi Tandel	Krutik Patel
	Present/Absent	Present/Absent	Present/Absent
8/01/2020	Present	Present	Present
15/01/2020	Present	Present	Present
22/01/2020	Present	Present	Present
29/01/2020	Present	Present	Present
5/02/2020	Present	Present	Present
4/03/2020	Present	Present	Present
11/03/2020	Present	Present	Present
18/03/2020	Present	Present	Present
1/04/2020	Present	Present	Present
15/04/2020	Present	Present	Present

Signature and Name of Internal Guide

Abstract

With the rapid development of artificial intelligence in recent years, image captioning has gradually drawn the attention of many researchers in the field of artificial intelligence and has become an interesting and difficult task. Image caption is an important part of scene understanding because it automatically generates natural language descriptions based on the content seen in an image. It necessitates both computer vision methods for comprehending the image's content and a natural language processing language model for converting the image's comprehension into words in the correct order. The use of image captions is widespread and important, for example, in the realization of human-computer interaction. Textual Description Generator is commonly used to act as a Caption Bot. It can, for example, generate relevant captions for Blog Posts. It also facilitates Google Image Search. This procedure has a wide range of potential applications in the real world. One notable example would be to save an image's captions so that it can be easily retrieved at a later stage based solely on this description. Deep learning methods have recently achieved cutting-edge results on examples of this problem. What's most exciting about these approaches is that instead of requiring complex data preparation or a pipeline of specially built models, a single end-to-end model can be established to predict a caption provided a picture. Alternative text for photographs is often absent in many types of media. The lack of image captions makes their content less available. It is also prohibitively costly to hire people to write captions for such images. A feasible option would be to use an automated method to write the captions. Advances in computer vision have resulted in the creation of feature extractors based on Artificial Neural Networks (ANN), such as the Convolutional Neural Network (CNN). ANNs may also be trained to model symbol sequences, such as sentences. The Recurrent Neural Network is one kind of ANN (RNN). The aim of this project is to learn the concepts of a CNN and LSTM model and to create a working model of an image caption generator using CNN and LSTM. We will use CNN (Convolutional Neural Networks) and LSTM to implement the caption generator (Long short-term memory). The image features will be extracted from Xception, which is a CNN model trained on the ImageNet dataset, and fed into the LSTM model, which will generate the image captions. Using a statistical model previously learned from labeled data, these feature extractors can transform digital images into rich high-level representations. Based on the previously obtained image attributes, the RNN will produce a caption.

Table of Contents

List of Figures

Chapter 1

Introduction

Deep Learning is a thriving environment right now, with new apps being released daily. And the only way to learn more about Deep Learning is to experiment with it. Take on as many tasks as you can and strive to complete them all on your own. This will help you understand the subjects more thoroughly and will help you become a stronger Deep Learning practitioner. In this project report, we will look at an intriguing multi-modal subject in which we will combine image and text processing to create a useful Deep Learning framework known as Image Captioning. The process of creating textual descriptions from an image based on the objects and behavior in the image is referred to as image captioning. For example:

black dog and white dog with brown spots are staring at each other in the street

small girl in the grass plays with fingerpaints in front of white canvas with rainbow on it

Figure 1.1: Textual Description Example

This procedure has a wide range of possible applications in the real world. One notable example will be to save an image's captions so that it can be conveniently retrieved at a later point based solely on this definition.

1.1 Advantages of Image Textual Description Generator

Globally, visuals and imagery continue to influence social and professional experiences. Manual efforts are falling short of monitoring, recognizing, and annotating massive volumes of visual data as the scale grows. With the introduction of artificial intelligence, multimedia companies can now accelerate the process of image captioning while still producing substantial value. AI-powered image caption generator automates image captioning processes by using various artificial intelligence services and technologies such as deep neural networks.

1.2 Limitations of this Model

The drawbacks of retrieval-based image captioning approaches are self-evident. These methods pass well-formed human-written sentences or phrases to generate descriptions for query images. Although the resulting outputs are normally grammatically correct and fluent, limiting image descriptions to previously existing sentences prevents them from adapting to new combinations of objects or novel scenes. In certain cases, provided descriptions can even be unrelated to the image contents. The ability of retrieval-based approaches to represent images is severely restricted. Our model will be based on the data, so it will be unable to predict terms that are not in its vocabulary.

1.3 Applications of Image Textual Description Generator

The AI-powered image captioning model is an automated method that efficiently produces succinct and meaningful captions for massive amounts of images. To extract detailed textual knowledge about the provided images, the model employs techniques from computer vision and Natural Language Processing (NLP).

1.3.1 Recommendations in Editing Applications

The image captioning model automates and accelerates the process of closed captioning for digital content creation, editing, distribution, and archival. In order to produce quality captions for photographs and videos, well-trained models replace manual efforts.

1.3.2 Assistance for Visually Impaired

The introduction of machine learning solutions such as image captioning is a blessing for visually disabled individuals who are unable to comprehend visuals. Image explanations can be read out to visually disabled people using an AI-powered image caption generator, allowing them to get a better understanding of their surroundings.

1.3.3 Media and Publishing Houses

Tens of thousands of visual data points are circulated across borders by the media and public relations industries in the form of newsletters, emails, and so on. The picture captioning model speeds up the production of subtitles, allowing executives to concentrate on more critical tasks.

1.3.4 Social Media Posts

In the context of social media, artificial intelligence is progressing from discussion forums to underlying frameworks for defining and representing terabytes of media data. It allows managers to track interactions and analysts to develop business strategies.

1.3.5 Medical Domain

Doctors can use this technology to find tumors or some defects in the images or used by people for understanding geospatial images where they can find out more details about the terrain.

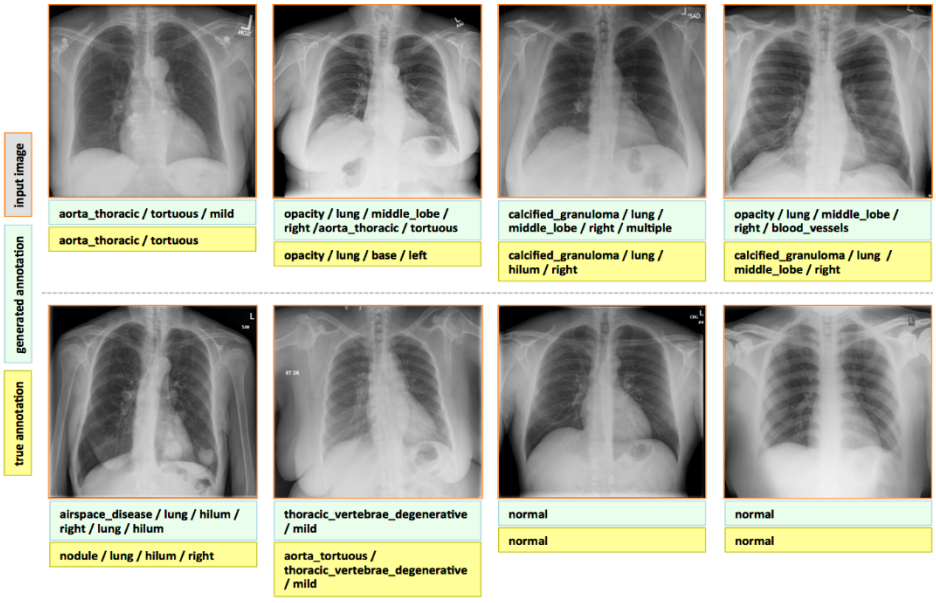


Figure 1.2: Use Case-Medical Domain

1.4 Technology Used

Python 3

Keras 2.4.3

Numpy

Pandas

NLTK

Matplotlib

Pre-trained weights of Xception Model

The model requires:

- Methods from Computer Vision (to understand the content of the image)
- Language model from NLP (to turn the understanding of image into Words)
- After training the model, its learning will be tested using Deep Learning.

Chapter 2

Literature Survey

In 2014 a number of high-profile AI labs began to release new approaches leveraging deep learning to improve performance. The first paper, to the best of our knowledge, to apply neural networks to the image captioning problem was Kiros et al. (2014a), who proposed a multi-layer perceptron (MLP) that uses a group of word representation vectors biased by features from the image, meaning the image itself conditioned the linguistic output. Vinyals et al. (2014) [13], successfully used a sequence-to-sequence model in which the typical encoder LSTM was replaced by a CNN. In their paper titled, "*Show and Tell: A Neural Image Caption Generator",* the CNN takes an input image and generates the feature representation which is then fed to the decoder LSTM for generating the output sentence (see fig. 2).

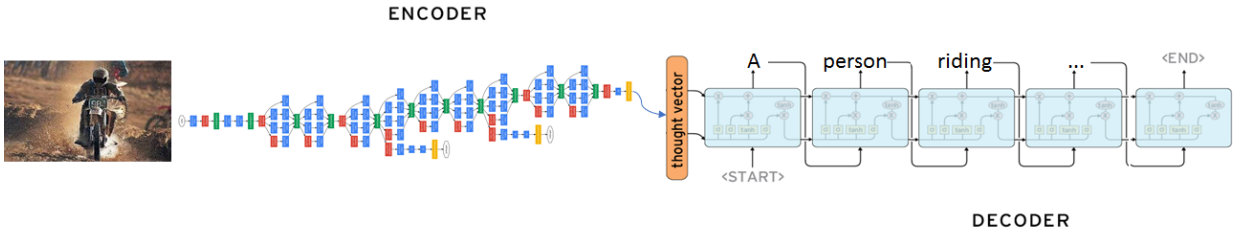


Figure 2.1: CNN encoder to LSTM decoder

Note: The image is encoded into a context vector by a CNN which can then be passed to a RNN decoder. Different neural *blocks* of computation are combined in new ways to handle different tasks.

Source: The M Tank [12]

Around the time *Show and Tell* came around, a similar, but distinct, approach was presented by Donahue et al. (2014): Long-term Recurrent Convolutional Networks for Visual Recognition and Description [15]. Instead of just using an LSTM for encoding a vector, as is typically done in *sequence-to-sequence* models, the feature representation is outputted by a CNN, in this case

VGGNet, and presented to the decoder LSTM. This work was also successfully applied to video captioning, a natural extension of image captioning.

From Captions to Visual Concepts and Back, by Fang et al. (2014) [16], is useful to explain multi-modality of the 2014 breakthroughs. Although distinct from the approaches of Vinyals et al. (2014) [17] and Donahue et al. (2014) [15], the paper represents an effective combination of some of these ideas. For readers, the working flow of the captioning process may bring a new appreciation of the modularity of these approaches.

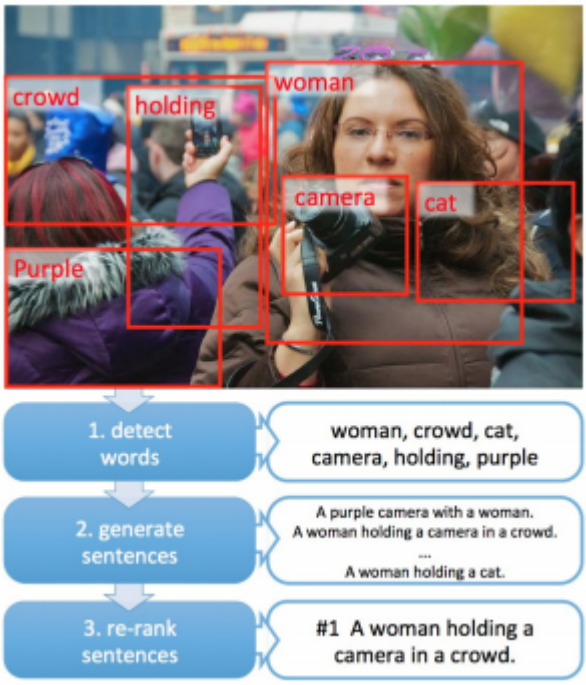


Figure 2.2: Creating captions from visual concepts

Source: Fang et al. (2014) [16]

With these technical improvements, Johnson et al. (2015) asked the question, **why are we describing an image with a single caption, when we can use the diversity of the captions in each region of interest to generate multiple captions with better descriptions than an individual image caption provides?** The authors introduce a variation to the image captioning task called ***dense captioning*** where the model describes individual parts of the image (denoted by bounding boxes). This approach produces results that may be more relevant, and accurate, when contrasted with captioning an entire image with a single sentence.

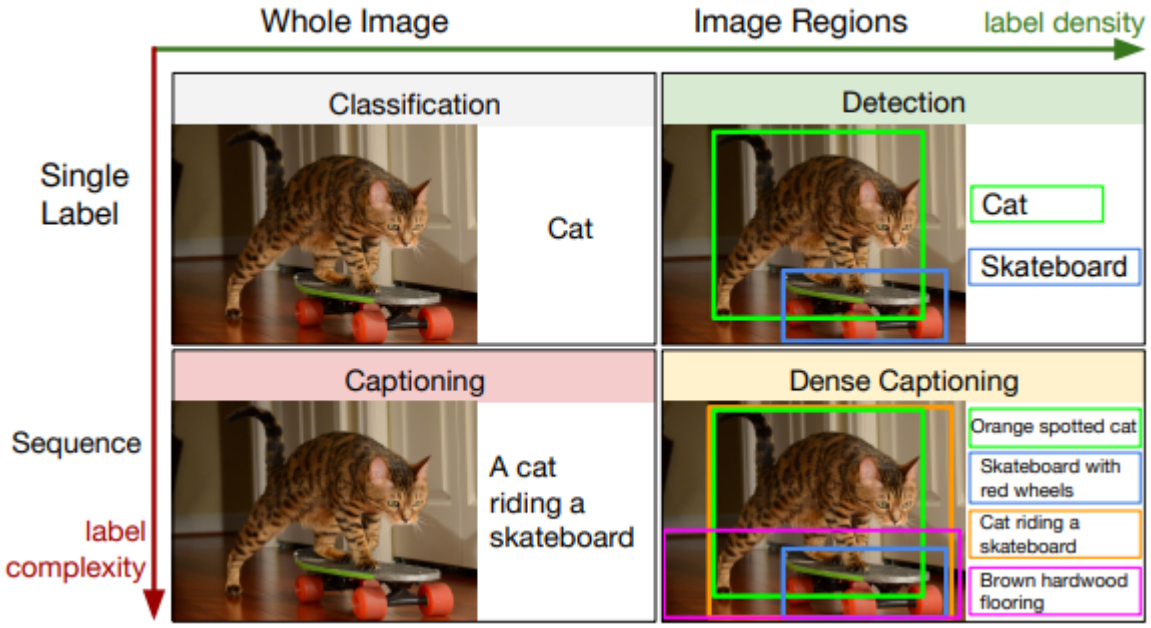


Figure 2.3: Dense captioning & labelling

Note from source: We address the Dense Captioning task (bottom right) by generating dense, rich annotations with a single forward pass.

Source: Johnson et al. (2015) [19]

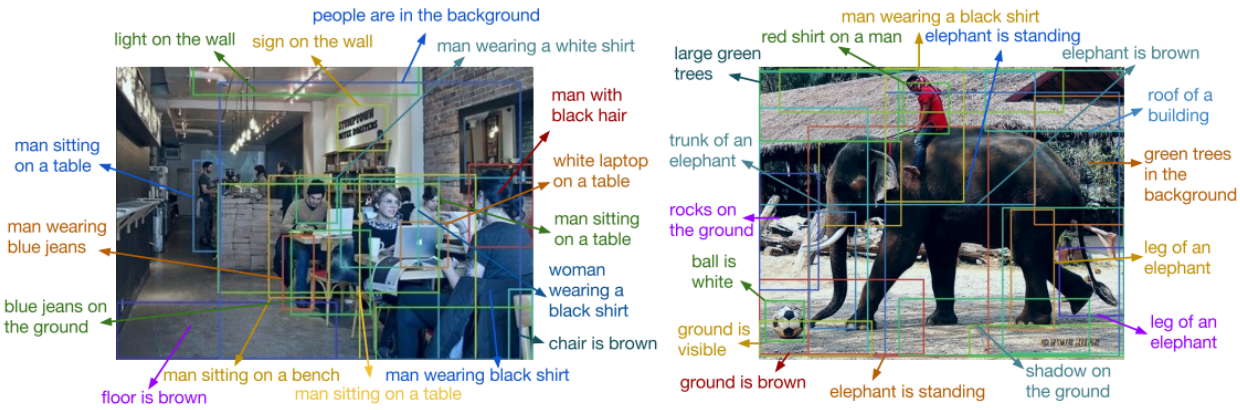


Figure 2.4: Dense captioning in action

Note from source: Example captions generated and localized by our model on test images. We render the top few most confident predictions.

Source: Johnson et al. (2015) [19]

Enter "*Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*" by Xu et al. (2015) [28] — the first paper, to our knowledge, that introduced the concept of attention into image captioning. The work takes inspiration from attention's application in

other sequence and image recognition problems. Building on seminal work from Kiros et al. (2014a; 2014b) [30][31], which incorporated the first neural networks into image captioning approaches, the impressive research team of Xu et al. (2015) implement hard and soft attention for the first time in image captioning. There are multiple ways to implement attention, but Xu et al. (2015) divide the image into a grid of regions after the CNN feature extraction and produce one feature vector for each. These features are used in different ways for soft and hard attention.

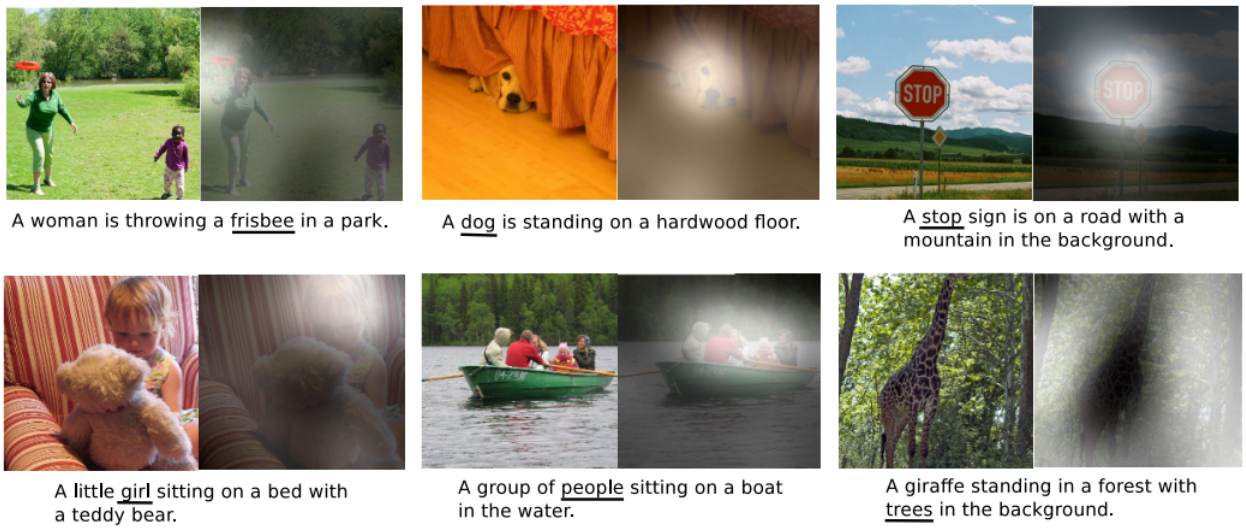


Figure 2.5: Attention in action

Note from source: Examples of attending to the correct object (white indicates the attended regions, underlines indicated the corresponding word).

Source: Xu et al. 2015 [28]

(You et al., 2016) [26] You et al. (2016) noted that traditional approaches to image captioning are either '*top-down, moving from a gist of an image which is converted to words, or bottom-up, which generate words describing various aspects of an image and then combine them*'. However, their contribution is the introduction of a novel algorithm that combines both approaches and learns to *selectively attend*. This is achieved through a model of semantic attention, which combines semantic concepts and the feature representation of the image/encoding. There are several important differences, by the authors' own admission, between their use of semantic attention and previous use-cases in image captioning. Comparing this work to Xu et al. (2015) [28], their attention algorithm learns to attend to the specific word concepts found within an image ***rather than*** words defined from specific

spatial locations. It is important to note that some concepts or words may not be related to a specific region, e.g., the word "exciting" which may encompass the entire image. This is the case even with concepts that are not directly seen in the image and can be expanded by '*leveraging external image data for training additional visual concepts as well as external text data for learning semantics between words'*.

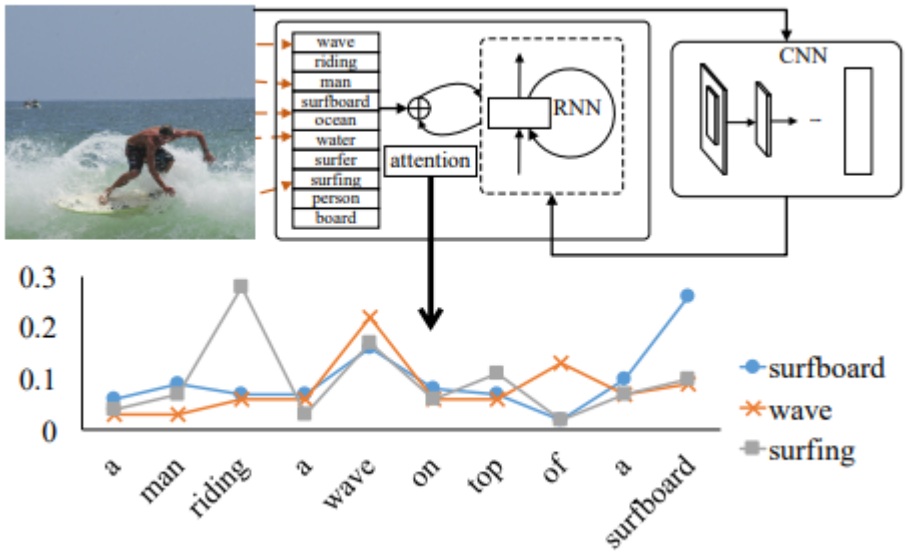


Figure 2.6: Semantic attention framework

Note from source: Top — an overview of the proposed framework. Given an image, we use a convolutional neural network to extract a top-down visual feature and at the same time detect visual concepts (regions, objects, attributes, etc.). We employ a semantic attention model to combine the visual feature with visual concepts in a recurrent neural network that generates the image caption.

Bottom — We show the changes of the attention weights for several candidate concepts with respect to the recurrent neural network iterations.

Source: You et al. (2016) [26]

Next, we introduce the concept of adaptive attention from Lu et al. (2017) [32]. ***Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning*** produced new benchmarks for state-of-the-art on both the COCO and Flickr30K datasets. Instead of forcing visual attention to be active for each generated word, Lu et al. (2017) reason that certain words in a sentence do not relate to the image, such as 'the', 'of', etc. Using a *visual sentinel,* the model learns *when* to use attention. Adaptive attention may also vary the amount

of attention supplied for each word. At each time step, our model decides whether to attend to the image (and if so, to which regions) or to the visual sentinel. The model decides whether to attend to the image and where, in order to extract meaningful information for sequential word generation".

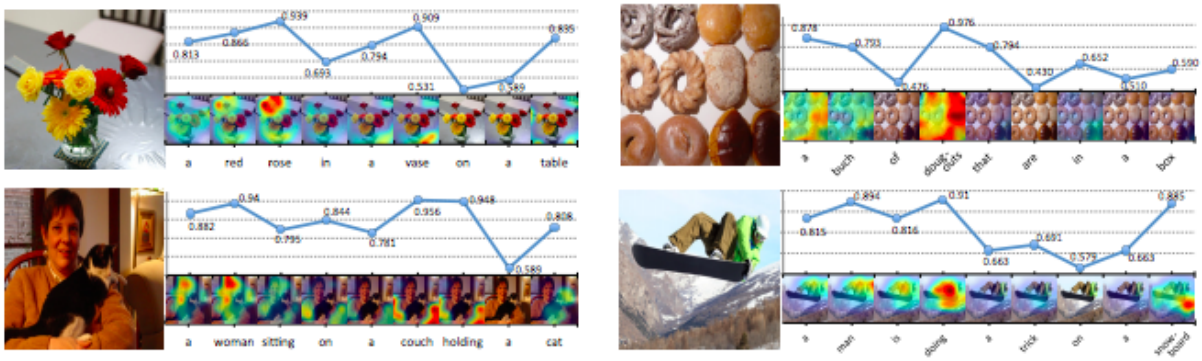


Figure 2.7: Visualization of caption generation

Note from source: Visualization of generated captions, visual grounding probabilities of each generated word, and corresponding spatial attention maps produced by our model.

Note: Knowing where and when, and how much to look. The probabilities graphed change depending on the 'importance' of the word, i.e., the attention that should be given to an image section when generating the sentence.

Source: Lu et al. (2017) [32]

Another interesting piece is ***SCA-CNN: Spatial and Channel-wise Attention in Convolutional Networks for Image Captioning*** from Chen et al. (2017) [25]. The authors go all out and make use of spatial, semantic, multilayer, and multi-channel attention in their CNN architecture, while also gently admonishing the use of traditional spatial attention mechanisms.

In 2017, ***Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering*** from Anderson et al. (2018) [24] proposed a more natural method for attention, inspired by neuroscience. Another paper that truly explores the difference between Bottom-Up and Top-Down attention, the authors examine attention in depth and present a method to efficiently fuse information from both types.

Chapter 3

Problem Statement

Suppose you see this picture –

Figure 3.1: A sample image

What is the first thing that comes to you mind? Here are a few sentences that people could come up with:

- man is on sailboard with green and clear sail in the middle of the ocean
- man is parasailing in the ocean with clear sky
- person rides white and green sail board across the water
- someone having fun on their surfboard in the clear blue ocean

A quick glance is all you need to understand and explain what is going on in the picture. Picture captioning is the job of automatically creating this textual definition from an artificial system. The task is simple – the produced output is required to explain what is shown in the image in a single sentence – the objects present, their properties, the actions being performed, and the interaction between the objects, and so on. But, as with any other image processing problem, replicating this behavior in an artificial system is a huge challenge, necessitating the use of sophisticated and advanced techniques such as Deep Learning to solve the problem.

Dataset: The dataset used here is the **FLICKR 8K** which consists of around 8091 images along with 5 captions for each image. If you have a powerful system with more than 16 GB RAM and a graphic card with more than 4 GB of memory, you can try to take **FLICKR 30K** which has around 30,000 images with captions.

Chapter 4

Project Design

1.1 System Architecture

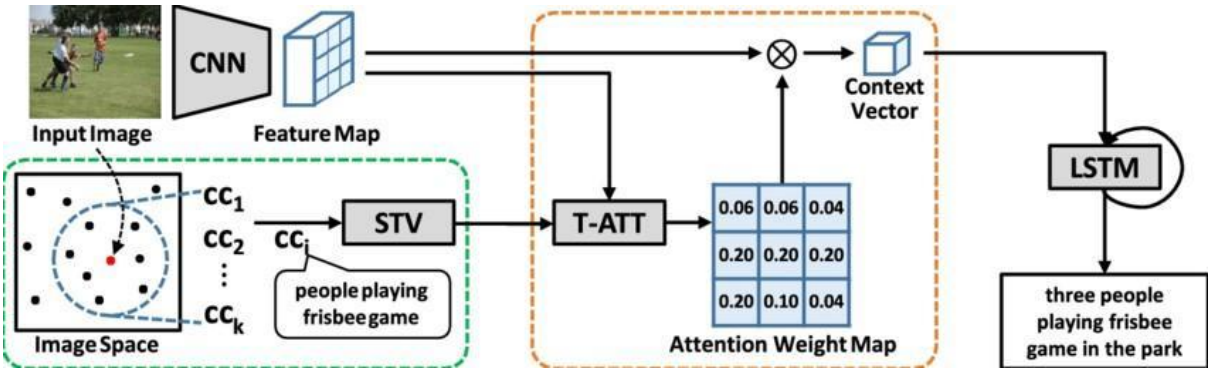


Figure 4.1: A functional CNN-RNN model.

Source: Research Gate

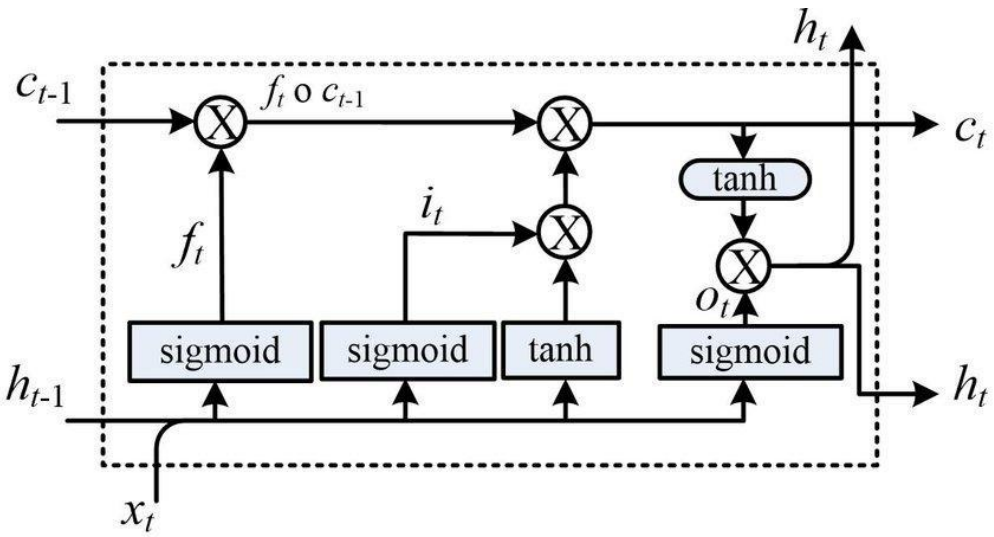


Figure 4.2: LSTM Cell

1.2 Phases of Image Caption Generator

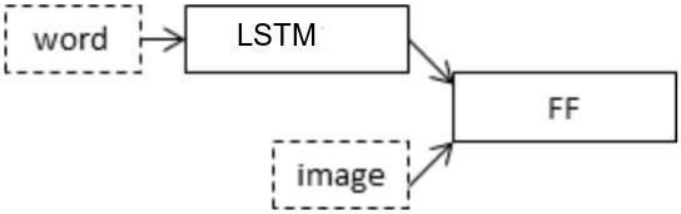


Figure 4.3: Schematic of the Merge Model for Image Captioning

1.2.1 Feature Extraction

CNNs derive distinct features from an image dependent on its spatial meaning as the first step. CNNs generate dense feature vectors, also known as embedding, which are used as feedback for the RNN algorithms that follow.

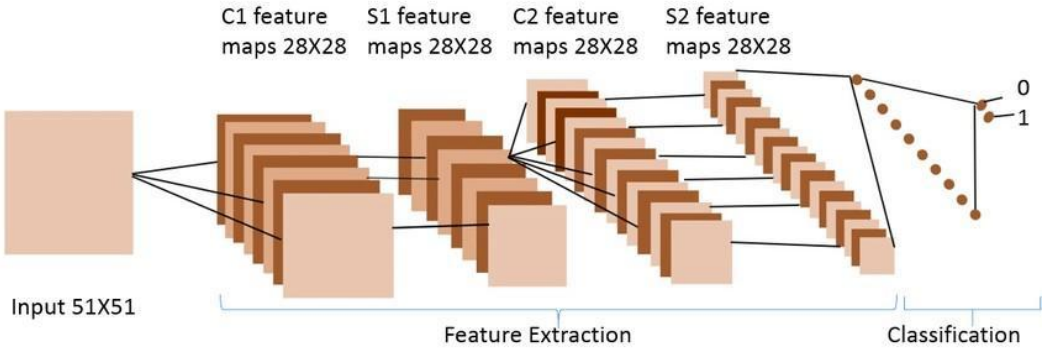


Figure 4.4: A-CNN-is-composed-of-two-basic-parts-of-feature-extraction-and-classification-Feature

1.2.2 Tokenization

The RNN module is used in the second step to ‘decode' the process vector inputs provided by the CNN module. The RNN model must be trained with a relevant dataset before it can begin the captioning process. The RNN model must be trained in order to predict the next word in the sentence. However, without definite numerical alphas values, training the model with strings is inefficient. It was necessary to translate the image captions into a list of tokenized words for this reason, as seen below-

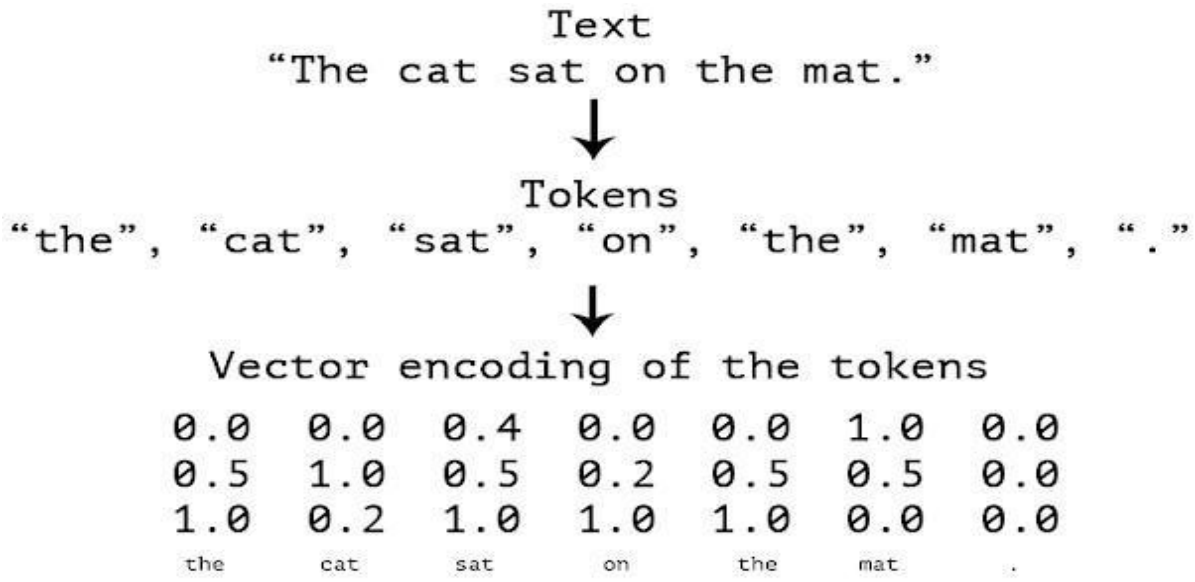


Figure 4.5: Source: Manning

1.2.3 Text Prediction

The final step of the model is activated using LSTM after tokenization. This move necessitates the use of an embedding layer to convert each word into the correct vector, which is then moved for decoding. The RNN model must recall spatial knowledge from the input function vector and anticipate the next term while using LSTM. After the LSTM has completed its functions, the (get prediction) function is used to produce the final output. Oodles, a startup, recently developed a deep neural network-powered image captioning model. Here is how it works for photographs to produce near-perfect results:

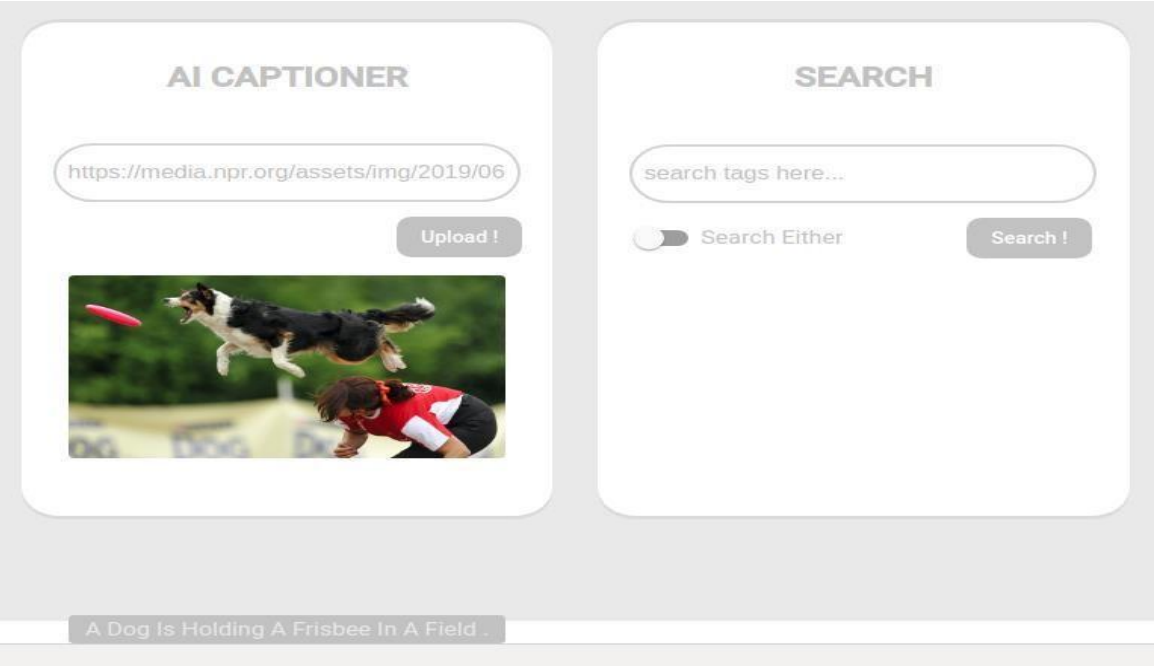


Figure 4.6: Image Captioning Model by Team Oodles

Source: Oodles AI

In addition to image captioning, the model can be used to search for relevant images with input in the form of tags such as “cars”, “books”, etc.

We also create a plot to visualize the structure of the network that better helps understand the two streams of input.

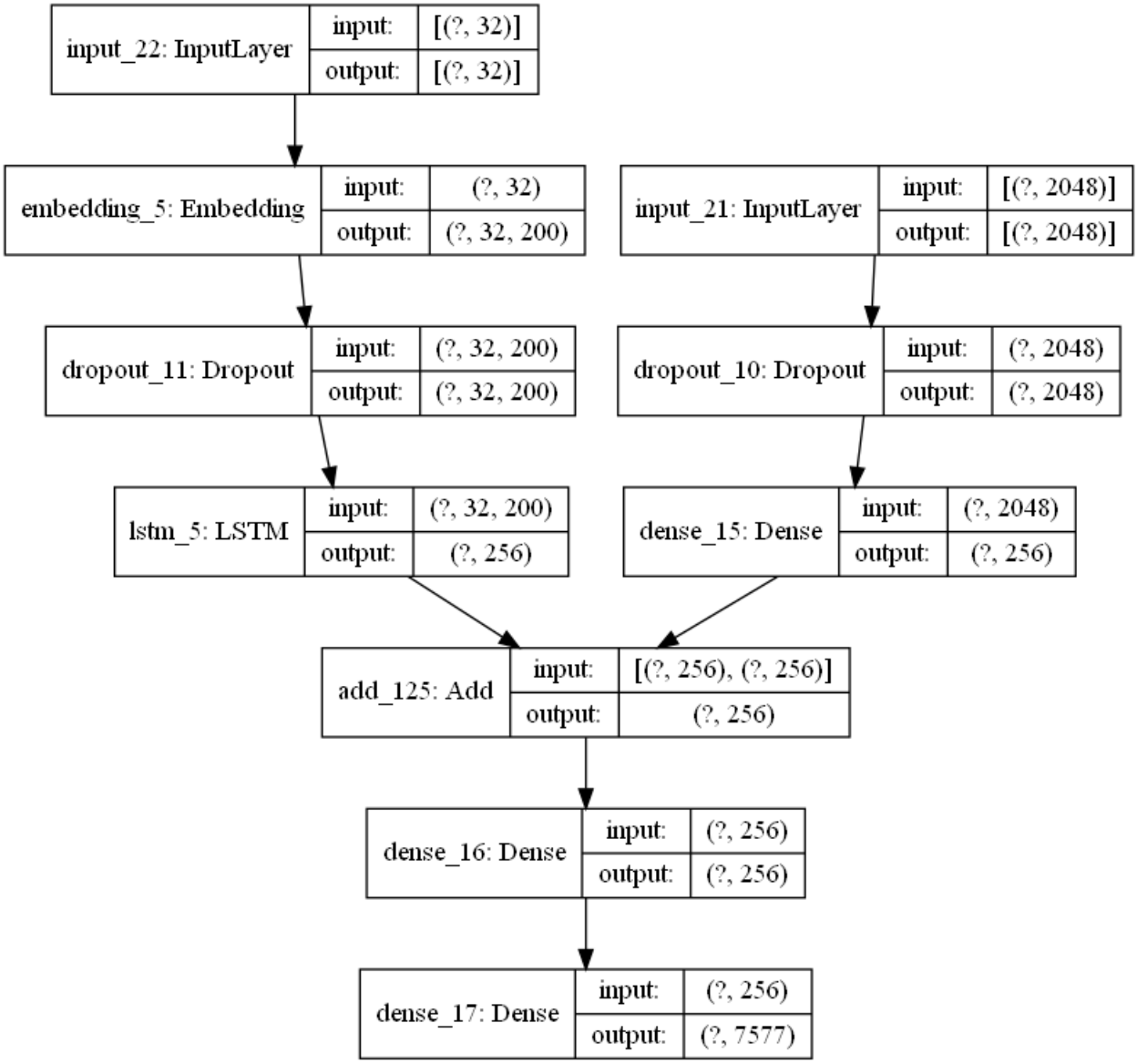


Figure 4.7: Plot of the Caption Generation Deep Learning Model

1.3 Flow Diagram

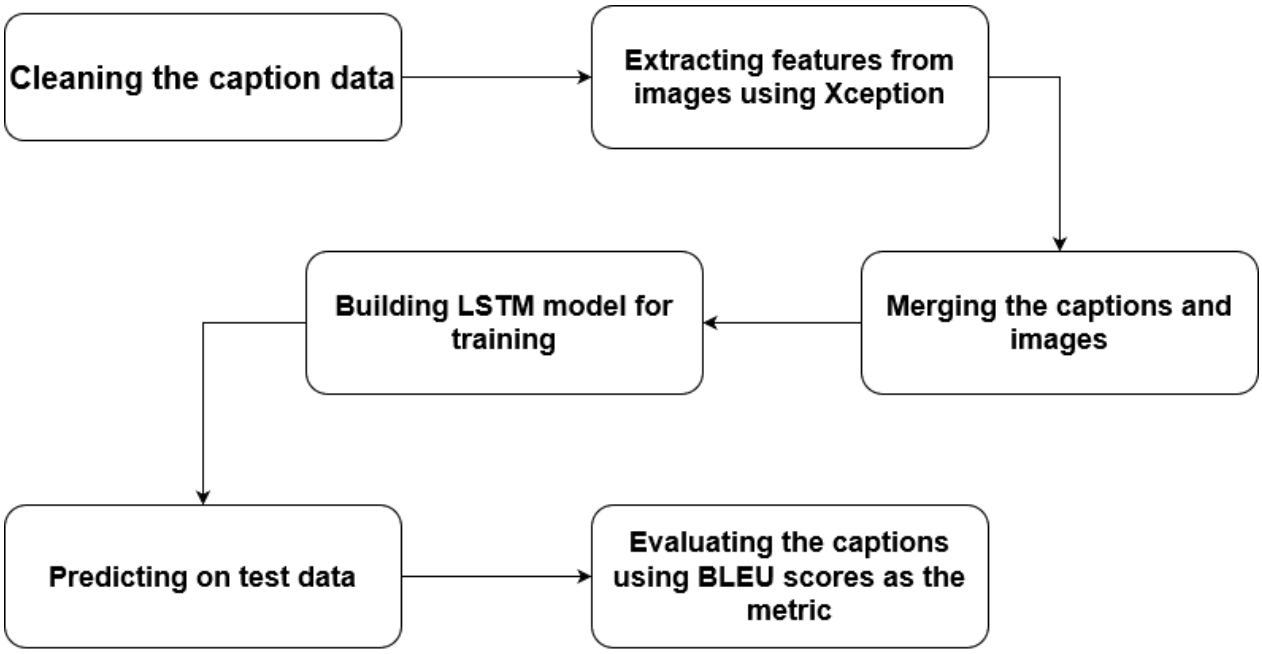


Figure 4.3: Flow Diagram

Chapter 5

Implementation Details

In [74]:
```
import string
import numpy as np
import pandas as pd
from PIL import Image
import matplotlib.pyplot as plt
import os
from pickle import dump, load
from collections import Counter
from keras.applications.xception import preprocess_input, Xception
from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical, plot_model
from keras.layers.merge import add
from keras.models import Model, load_model
from keras.layers import Input, Dense, LSTM, Embedding, Dropout
from keras.callbacks import ModelCheckpoint
from nltk.translate.bleu_score import corpus_bleu
from tqdm.notebook import tqdm_notebook as tqdm
```

In [67]:
```
## The location of the Flickr8K_ images
dir_Flickr_jpg = "Flickr8k_Dataset/"
## The location of the caption file
dir_Flickr_text = "Flickr8k.token.txt"

jpgs = os.listdir(dir_Flickr_jpg)
print("The number of jpg flies in Flicker8k: {}".format(len(jpgs)))
```

```
The number of jpg flies in Flicker8k: 8092
```

In [3]:
```
#Finding the captions for each image.
file = open(dir_Flickr_text,'r', encoding='utf8')
text = file.read()
file.close()


datatxt = []
for line in text.split('\n'):
    col = line.split('\t')
    if len(col) == 1:
        continue
    w = col[0].split("#") # Splitting the caption dataset at the required position
    datatxt.append(w + [col[1].lower()])

df_txt = pd.DataFrame(datatxt,columns=["filename","index","caption"])


uni_filenames = np.unique(df_txt.filename.values)
print("The number of unique file names : {}".format(len(uni_filenames)))
print("The distribution of the number of captions for each image:")
Counter(Counter(df_txt.filename.values).values())
print(df_txt[:5])
```

```
The number of unique file names : 8092
The distribution of the number of captions for each image:
                    filename index  \
0  1000268201_693b08cb0e.jpg     0
1  1000268201_693b08cb0e.jpg     1
2  1000268201_693b08cb0e.jpg     2
3  1000268201_693b08cb0e.jpg     3
4  1000268201_693b08cb0e.jpg     4

                                             caption
0  a child in a pink dress is climbing up a set o...
1              a girl going into a wooden building .
2   a little girl climbing into a wooden playhouse .
3  a little girl climbing the stairs to her playh...
4  a little girl in a pink dress going into a woo...
```

In [4]:
```
from keras.preprocessing.image import load_img, img_to_array
from IPython.display import display
from PIL import Image

npic = 5 # Displaying 5 images from the dataset
npix = 224
target_size = (npix,npix,3)

count = 1
fig = plt.figure(figsize=(10,20))
for jpgfnm in uni_filenames[-5:]:
    filename = dir_Flickr_jpg + '/' + jpgfnm
    captions = list(df_txt["caption"].loc[df_txt["filename"]==jpgfnm].values)
    image_load = load_img(filename, target_size=target_size)

    ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
    ax.imshow(image_load)
    count += 1

    ax = fig.add_subplot(npic,2,count)
    plt.axis('off')
    ax.plot()
    ax.set_xlim(0,1)
    ax.set_ylim(0,len(captions))
    for i, caption in enumerate(captions):
        ax.text(0,i,caption,fontsize=20)
    count += 1
plt.show()
```

a white crane stands tall as it looks out upon the ocean .
a water bird standing at the ocean 's edge .
a tall bird is standing on the sand beside the ocean .
a large bird stands in the water on the beach .
a grey bird stands majestically on a beach while waves roll in .

woman writing on a pad in room with gold , decorated walls .
the walls are covered in gold and patterns .
a woman standing near a decorated wall writes .
a woman behind a scrolled wall is writing
a person stands near golden walls .

a rock climber practices on a rock climbing wall .
a rock climber in a red shirt .
a person in a red shirt climbing up a rock face covered in assist handles .
a man is rock climbing high in the air .
a man in a pink shirt climbs a rock face

In [14]:
```
def load_doc(filename):
    file = open(filename, 'r')
    text = file.read()
    file.close()
    return text
```

In [15]:
```
def all_img_captions(filename):
    file = load_doc(filename)
    captions = file.split('\n')
    descriptions = {}
    for txt in captions[:-1]:
        img, caption = txt.split('\t')
        if img[:-2] not in descriptions:
            descriptions[img[:-2]] = [caption]
        else:
            descriptions[img[:-2]].append(caption)
    return descriptions


file = "Flickr8k.token.txt"
descriptions = all_img_captions(file)
len(descriptions.keys())
```

Out[15]: 8092

In [16]:
```
def cleaning_text(captions):
    table = str.maketrans('', '', string.punctuation)
    for img, caps in captions.items():
        for i, img_caption in enumerate(caps):
            img_caption.replace("-", " ")
            desc = img_caption.split()
            desc = [word.lower() for word in desc]
            desc = [word.translate(table) for word in desc]
            desc = [word for word in desc if (len(word) > 1)]
            desc = [word for word in desc if (word.isalpha())]
            img_caption = ' '.join(desc)
            captions[img][i] = img_caption
    return captions
```

In [17]:
```
def text_vocabulary(descriptions):
    vocab = set()
    for key in descriptions.keys():
        [vocab.update(d.split()) for d in descriptions[key]]
    return vocab
```

In [18]:
```
def save_descriptions(descriptions, filename):
    lines = list()
    for key, desc_list in descriptions.items():
        for desc in desc_list:
            lines.append(key + '\t' + desc)
    data = "\n".join(lines)
    file = open(filename, "w")
    file.write(data)
    file.close()
```

In [19]:
```
filename = "Flickr8k.token.txt"
descriptions = all_img_captions(filename)
print("Length of descriptions =", len(descriptions))
clean_descriptions = cleaning_text(descriptions)
vocabulary = text_vocabulary(clean_descriptions)
print("Length of vocabulary = ", len(vocabulary))
save_descriptions(clean_descriptions, "descriptions.txt")
```

```
Length of descriptions = 8092
Length of vocabulary =  8763
```

In [20]:
```
all_train_captions = []
for key, val in descriptions.items():
    for cap in val:
        all_train_captions.append(cap)

word_count_threshold = 8
word_counts = {}
nsents = 0
for sent in all_train_captions:
    nsents += 1
    for w in sent.split(' '):
        word_counts[w] = word_counts.get(w, 0) + 1

vocab = [w for w in word_counts if word_counts[w] >= word_count_threshold]
print('Preprocessed words=%d ' % len(vocab))
```

```
Preprocessed words=2246
```

In [45]:

```
def extract_features(directory):
    model = Xception(include_top=False, pooling='avg')
    features = {}
    for img in tqdm(os.listdir(directory)):
        filename = directory + "/" + img
        image = image.open(filename)
        image = image.resize((299, 299))
        image = np.expand_dims(image, axis=0)
        #image = preprocess_input(image)
        image = image / 127.5
        image = image - 1.0
        feature = model.predict(image)
        features[img] = feature
    return features


def generate_desc(model, tokenizer, photo, max_length):
    in_text = 'start'
    for i in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence], maxlen=max_length)
        pred = model.predict([photo, sequence], verbose=0)
        pred = np.argmax(pred)
        word = word_for_id(pred, tokenizer)
        if word is None:
            break
        in_text += ' ' + word
        if word == 'end':
            break
    return in_text
```

In [46]:
```
features = load(open("features.p", "rb"))
```

In [49]:
```
def load_photos(filename):
    file = load_doc(filename)
    photos = file.split("\n")[:-1]
    return photos


def load_clean_descriptions(filename, photos):
    file = load_doc(filename)
    descriptions = {}
    for line in file.split("\n"):
        words = line.split()
        if len(words) < 1:
            continue
        image, image_caption = words[0], words[1:]
        if image in photos:
            if image not in descriptions:
                descriptions[image] = []
            desc = '<start> ' + " ".join(image_caption) + ' <end>'
            descriptions[image].append(desc)
    return descriptions


def load_features(photos):
    all_features = load(open("features.p", "rb"))
    features = {k: all_features[k] for k in photos}
    return features
```

```
def max_length(descriptions):
    desc_list = dict_to_list(descriptions)
    return max(len(d.split()) for d in desc_list)


def word_for_id(integer, tokenizer):
    for word, index in tokenizer.word_index.items():
        if index == integer:
            return word
    return None


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


def create_tokenizer(descriptions):
    desc_list = dict_to_list(descriptions)
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(desc_list)
    return tokenizer
```

In [50]:
```
filename = "Flickr_8k.trainImages.txt"
train_imgs = load_photos(filename)
train_descriptions = load_clean_descriptions("descriptions.txt", train_imgs)
train_features = load_features(train_imgs)
```

In [51]:
```
def evaluate_model(model, descriptions, photos, tokenizer, max_length):
    actual, predicted = list(), list()
    for key, desc_list in tqdm(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)))
```

In [52]:
```
test_dire = "Flickr_8k.testImages.txt"
test_imgs = load_photos(test_dire)
test_descriptions = load_clean_descriptions("descriptions.txt", test_imgs)
test_features = load_features(test_imgs)
test_tokenizer = create_tokenizer(test_descriptions)
```

In [53]:
```
max_length_test=max_length(test_descriptions)
max_length_test
```

Out[53]:
```
30
```

In [54]:
```
tokenizer = create_tokenizer(train_descriptions)
dump(tokenizer, open('tokenizer.p', 'wb'))
vocab_size = len(tokenizer.word_index) + 1
vocab_size
```

Out[54]:
```
7577
```

In [55]:
```
max_length = max_length(descriptions)
max_length
```

Out[55]:
```
32
```

In [56]:
```
features['1000268201_693b08cb0e.jpg'][0]
```

Out[56]:
```
array([0.36452794, 0.12713662, 0.0013574 , ..., 0.221817  , 0.01178991,
       0.24176797], dtype=float32)
```

In [57]:
```
def data_generator(descriptions, features, tokenizer, max_length):
    while 1:
        for key, description_list in descriptions.items():
            feature = features[key][0]
            input_image, input_sequence, output_word = create_sequences(
                tokenizer, max_length, description_list, feature)
            yield [input_image, input_sequence], output_word


def create_sequences(tokenizer, max_length, desc_list, feature):
    X1, X2, y = list(), list(), list()
    for desc in desc_list:
        seq = tokenizer.texts_to_sequences([desc])[0]
        for i in range(1, len(seq)):
            in_seq, out_seq = seq[:i], seq[i]
            in_seq = pad_sequences([in_seq], maxlen=max_length)[0]
            out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]
            X1.append(feature)
            X2.append(in_seq)
            y.append(out_seq)
    return np.array(X1), np.array(X2), np.array(y)
```

In [58]:
```
[a, b], c = next(data_generator(train_descriptions, features, tokenizer, max_length))
```

In [59]:
```
def define_model(vocab_size, max_length):
    inputs1 = Input(shape=(2048, ))
    fe1 = Dropout(0.5)(inputs1)
    fe2 = Dense(256, activation='relu')(fe1)
    inputs2 = Input(shape=(max_length, ))
    se1 = Embedding(vocab_size, 200, mask_zero=True)(inputs2)
    se2 = Dropout(0.5)(se1)
    se3 = LSTM(256)(se2)
    decoder1 = add([fe2, se3])
    decoder2 = Dense(256, activation='relu')(decoder1)
    outputs = Dense(vocab_size, activation='softmax')(decoder2)
    model = Model(inputs=[inputs1, inputs2], outputs=outputs)
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    print(model.summary())
    plot_model(model,
               to_file='./model.png',
               show_shapes=True,
               show_layer_names=True)
    return model
```

In [61]:
```
print('Dataset: ', len(train_imgs))
print('Descriptions: train=', len(train_descriptions))
print('Photos: train=', len(train_features))
print('Vocabulary Size:', vocab_size)
print('Description Length: ', max_length)

epochs = 10
steps = len(train_descriptions)
model = define_model(vocab_size, max_length)
for i in range(epochs):
    generator = data_generator(train_descriptions, train_features, tokenizer,max_length)
    model.fit(generator, epochs=1, steps_per_epoch=steps, initial_epoch=0)
    model.save("model_" + str(i) + ".h5")
```

```
Dataset:  6000
Descriptions: train= 6000
Photos: train= 6000
Vocabulary Size: 7577
Description Length:  32
Model: "functional_3"
```

Layer (type)	Output Shape	Param #	Connected to
input_14 (InputLayer)	[(None, 32)]	0	
input_13 (InputLayer)	[(None, 2048)]	0	
embedding_1 (Embedding)	(None, 32, 200)	1515400	input_14[0][0]
dropout_2 (Dropout)	(None, 2048)	0	input_13[0][0]
dropout_3 (Dropout)	(None, 32, 200)	0	embedding_1[0][0]
dense_3 (Dense)	(None, 256)	524544	dropout_2[0][0]
lstm_1 (LSTM)	(None, 256)	467968	dropout_3[0][0]
add_121 (Add)	(None, 256)	0	dense_3[0][0] lstm_1[0][0]
dense_4 (Dense)	(None, 256)	65792	add_121[0][0]
dense_5 (Dense)	(None, 7577)	1947289	dense_4[0][0]

```
Total params: 4,520,993
Trainable params: 4,520,993
Non-trainable params: 0
```

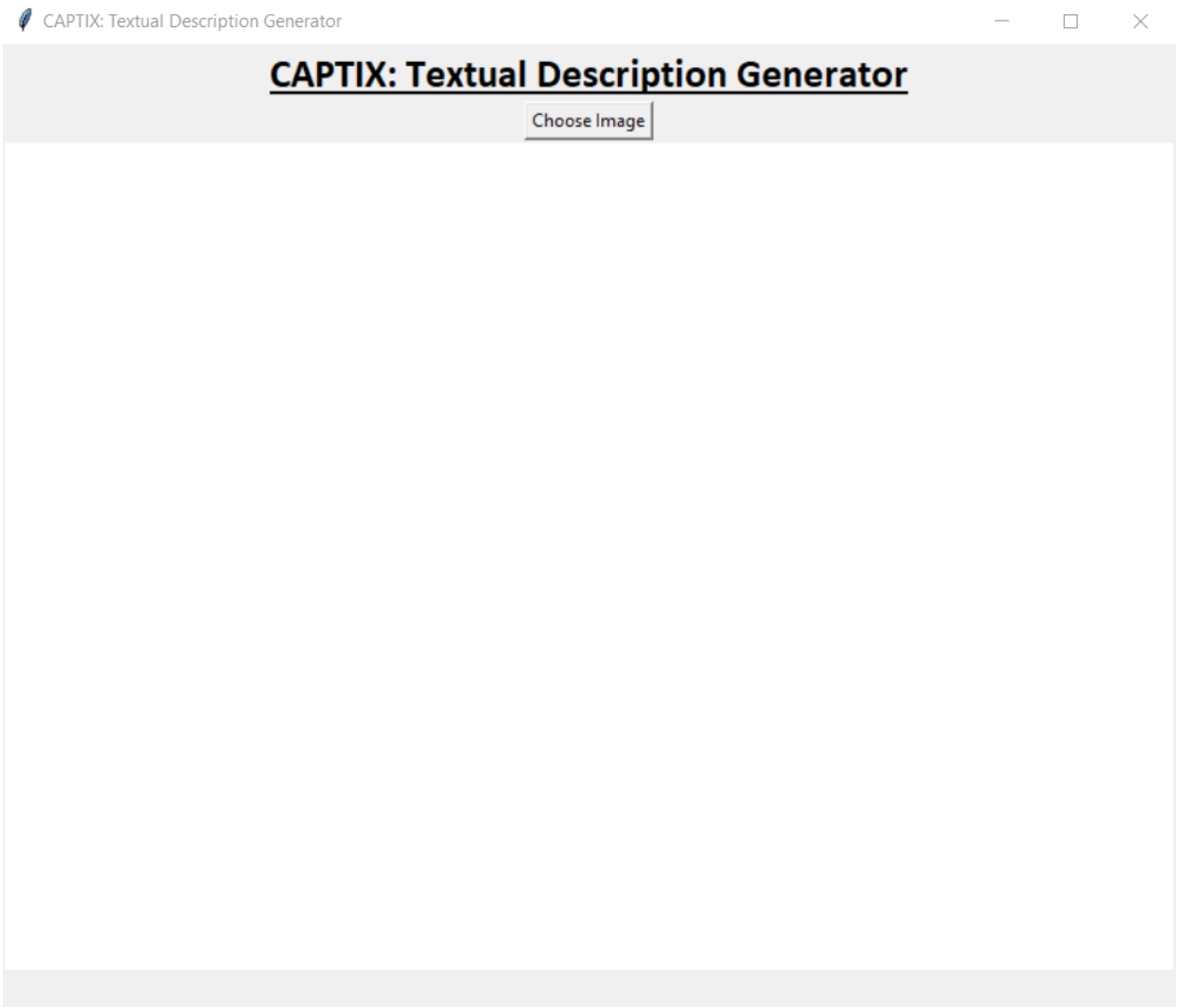
```
None
6000/6000 [==============================] - 964s 161ms/step - loss: 4.5442 - accuracy: 0.2325
6000/6000 [==============================] - 988s 165ms/step - loss: 3.7013 - accuracy: 0.2896
6000/6000 [==============================] - 969s 162ms/step - loss: 3.4129 - accuracy: 0.3098
6000/6000 [==============================] - 1113s 185ms/step - loss: 3.2436 - accuracy: 0.3216
```

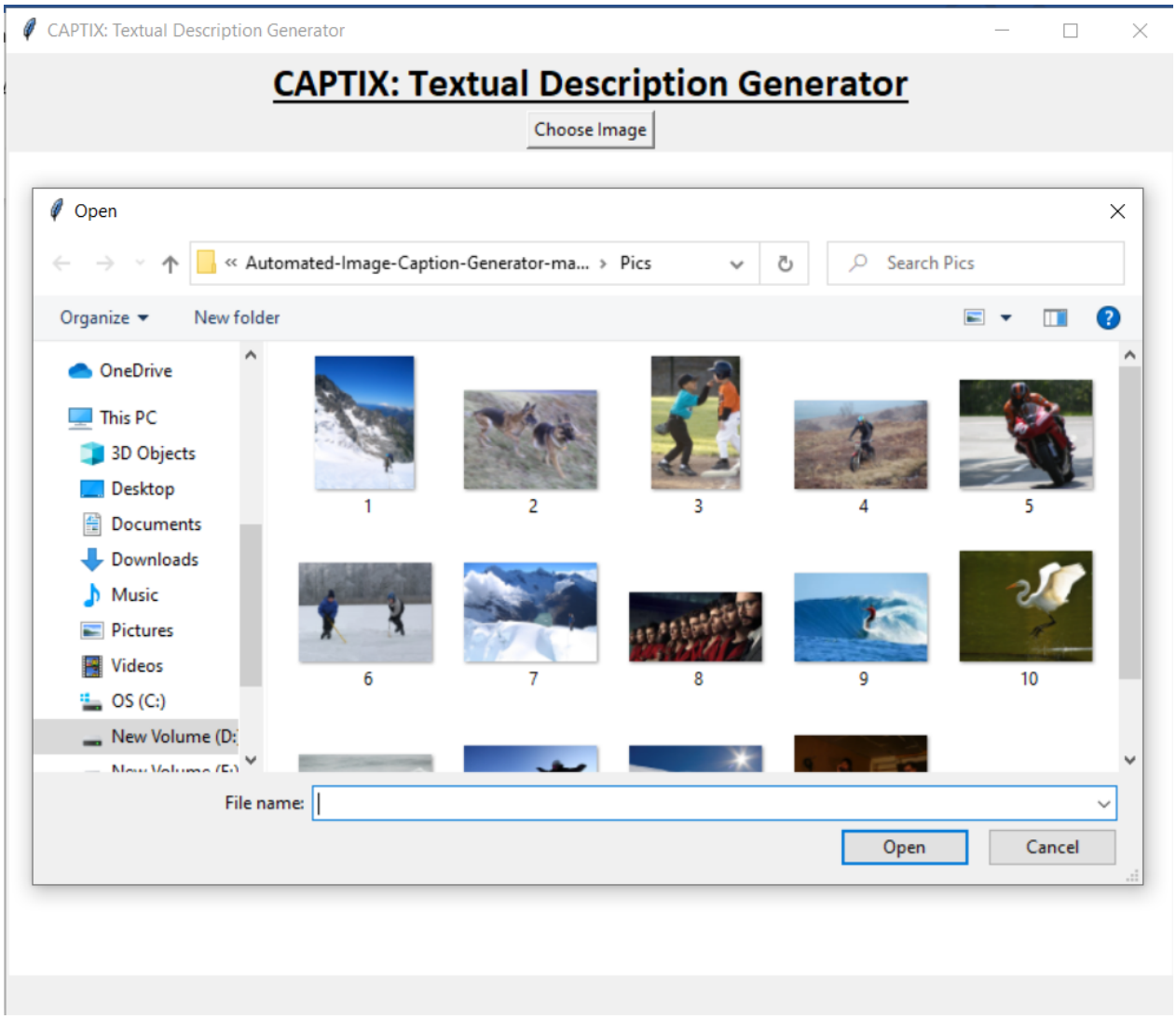
```
6000/6000 [==============================] - 964s 161ms/step - loss: 4.5442 - accuracy: 0.2325
6000/6000 [==============================] - 988s 165ms/step - loss: 3.7013 - accuracy: 0.2896
6000/6000 [==============================] - 969s 162ms/step - loss: 3.4129 - accuracy: 0.3098
6000/6000 [==============================] - 1113s 185ms/step - loss: 3.2436 - accuracy: 0.3216
6000/6000 [==============================] - 911s 152ms/step - loss: 3.1244 - accuracy: 0.3288
6000/6000 [==============================] - 917s 153ms/step - loss: 3.0384 - accuracy: 0.3358
6000/6000 [==============================] - 915s 153ms/step - loss: 2.9686 - accuracy: 0.3404
6000/6000 [==============================] - 906s 151ms/step - loss: 2.9190 - accuracy: 0.3459
6000/6000 [==============================] - 912s 152ms/step - loss: 2.8675 - accuracy: 0.3501
6000/6000 [==============================] - 922s 154ms/step - loss: 2.8362 - accuracy: 0.3525
```

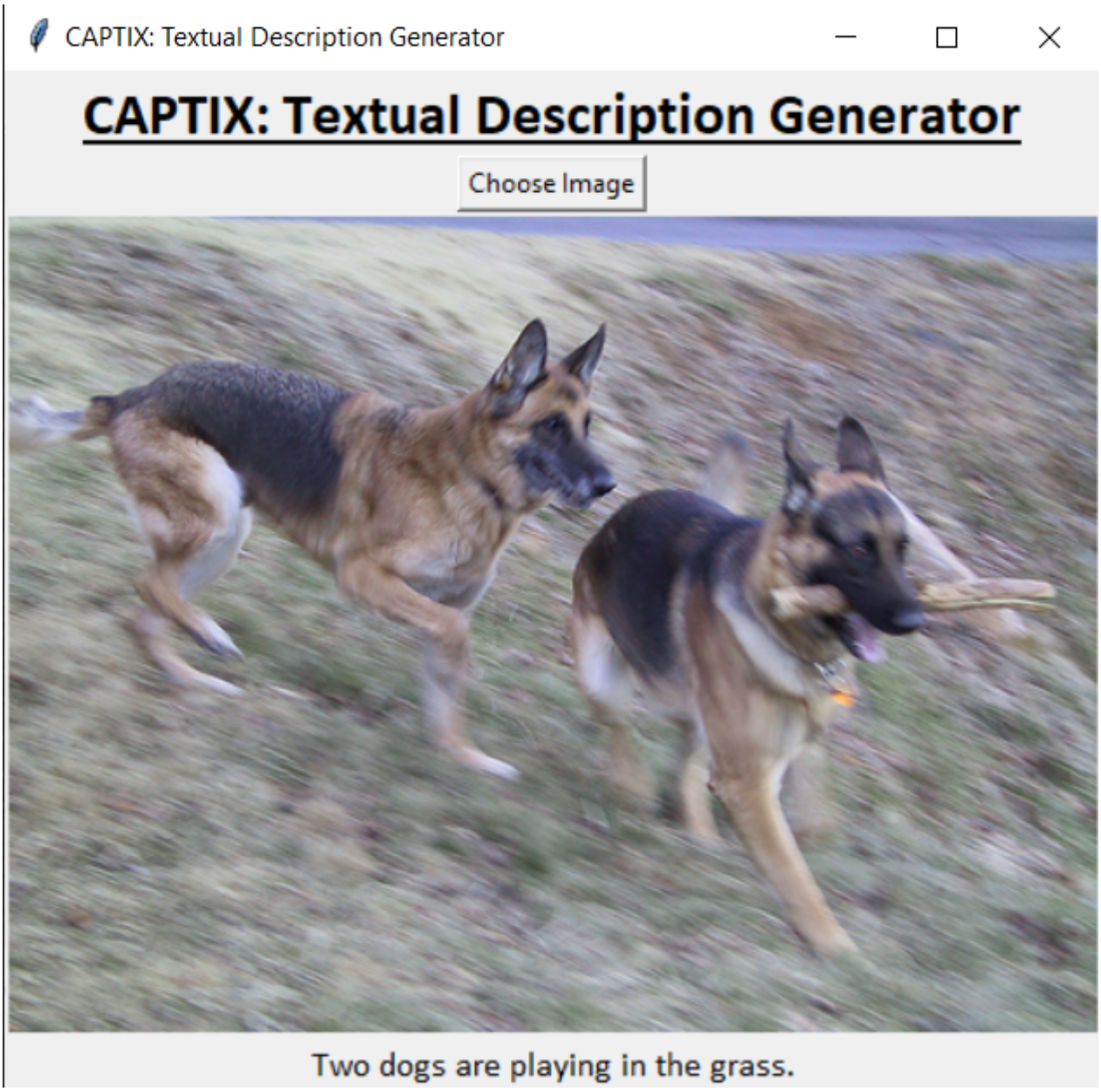
Chapter 6

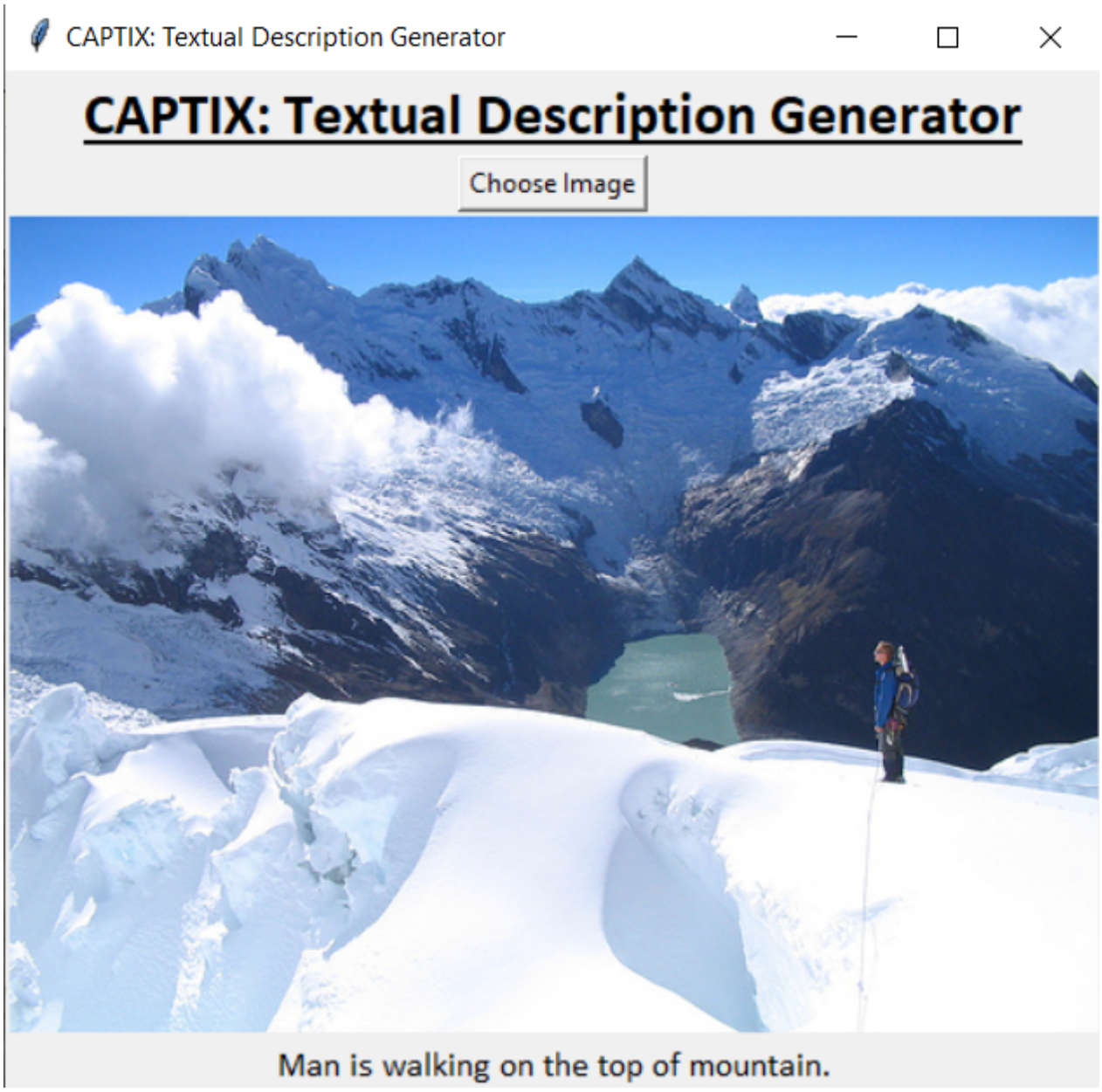
Result & Analysis

6.1 Results




CAPTIX: Textual Description Generator
CAPTIX: Textual Description Generator
Choose Image
Open
Automated-Image-Caption-Generator-ma... › Pics
Search Pics
Organize
New folder
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
New Volume (D:
1
2
3
4
5
6
7
8
9
10
File name:
Open
Cancel


CAPTIX: Textual Description Generator
CAPTIX: Textual Description Generator
Choose Image
Two dogs are playing in the grass.


CAPTIX: Textual Description Generator
CAPTIX: Textual Description Generator
Choose Image
Man is walking on the top of mountain.

6.2 Analysis

After the model is trained, we must test the model's prediction capabilities on a test dataset. Traditional accuracy metric cannot be used on predictions. For text evaluations we have a metric called as BLEU Score. BLEU stands for Bilingual Evaluation Understudy; it is a score for comparing a candidate text to one or more reference texts.

In [64]:
```
mod=load_model("model_9.h5")
```

In [65]:
```
evaluate_model(mod, test_descriptions, test_features, test_tokenizer, 32)
```

100% 1000/1000 [28:23<00:00, 1.70s/it]

```
d:\python\python36\lib\site-packages\nltk\translate\bleu_score.py:516: UserWarning:
The hypothesis contains 0 counts of 3-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
d:\python\python36\lib\site-packages\nltk\translate\bleu_score.py:516: UserWarning:
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
```

```
BLEU-1: 0.272195
BLEU-2: 0.030590
BLEU-3: 0.000000
BLEU-4: 0.000000
```

Chapter 7

Conclusion & Future Scope

The model has been successfully trained to generate the captions as expected for the images. The caption generation has constantly been improved by fine tuning the model with different hyper parameters. Higher BLEU score indicates that the generated captions are similar to those of the actual captions present on the images. Below you will find a table displaying different BLEU scores obtained by tuning the parameters:
With the help of the Tensor board, we were able to see how different training process had an impact on the model. The validation loss falls up to 5th epoch and then increases afterwards, while the training loss still continues falling the following were the major outcomes and observations of the training process and testing the model on the test data:

- The validation loss increases after the 5th epoch in most cases even though the training loss decreases over time. This indicates that the model is overfitting and the training needs to stop.
- Higher BLEU score does not always translate to better generated captions. If the model overfits on your training data, it will lead the model to go through details in the image and generate captions which do not make sense. It can be seen in the good and the bad captions generated above.

The model which we saw above was just the tip of the iceberg. There has been a lot of research done on this topic. Currently, the state-of-the-art model in image captioning is Microsoft's CaptionBot. You can look at a demo of the system on their official website (link : www.captionbot.ai).

I will list down a few ideas which you can use to build a better image captioning model.

- **Adding in more data –** Of course, this is the usual tendency of a Deep Learning model. More data you provide to your model, the better it will perform. You can refer to this resource for other image captioning datasets –
  – http://www.cs.toronto.edu/~fidler/slides/2017/CSC2539/Kaustav_slides.pdf
- **Using Attention models** – As we saw in this article (Essentials of Deep Learning – Sequence to Sequence modelling with Attention), using attention models help us in fine tuning our model performance.
- **Moving on to bigger and better techniques** – There are a few techniques which researchers have been investigating – such as using reinforcement learning for building end-to-end deep learning systems, or using novel attention model for visual sentinel.

References

[1] Lu et al. (2017). Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. [Online] arXiv: 1612.01887. Available: arXiv:1612.01887v2

[2] Kiros et al. (2014a): "Unlike many of the existing methods, our approach can generate sentence descriptions for images without the use of templates, structured prediction, and/or syntactic trees."

[3] Farhadi et al. (2010). Every Picture Tells a Story: Generating Sentences from Images. In: Daniilidis K., Maragos P., Paragios N. (eds) Computer Vision — ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol 6314. Springer, Berlin, Heidelberg. Available: https://www.cs.cmu.edu/~afarhadi/papers/sentence.pdf

[4] Kulkarni et al. (2013). BabyTalk: Understanding and Generating Simple Image Descriptions. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, No. 12, December. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6522402

[5] Kiros et al. (2014a). Multimodal Neural Language Models. Proceedings of the 31st International Conference on Machine Learning, PMLR 32(2):595–603. Available: http://proceedings.mlr.press/v32/kiros14.html

[6] Vedantam et al. (2014). CIDEr: Consensus-based Image Description Evaluation. [Online] arXiv: 1411.5726. Available: arXiv:1411.5726v2 (2015 version).

[7] ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation — Lin, C.Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain. Available: http://www.aclweb.org/anthology/W04-1013

[8] Banerjee, S., Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. [Online] Language Technologies Institute, Carnegie Mellon University (www.cs.cmu.edu). Available: https://www.cs.cmu.edu/~alavie/papers/BanerjeeLavie2005-final.pdf

[9] Cocodataset.org. (2018) COCO: Common Objects in Context. [Website] http://cocodataset.org/. Available: http://cocodataset.org/#captions-challenge2015

[10] Sutskever et al. (2014). Sequence to Sequence Learning with Neural Networks. [Online] arXiv: 1409.3215. Available: arXiv:1409.3215v3

[11] Britz, D. (2016). Deep Learning for Chatbots, Part 1 — Introduction. [Blog] WildML (http://www.wildml.com/). Available: http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/

[12] For shameless self-promotion, see previous report: "A Year in Computer Vision". Available: http://www.themtank.org/a-year-in-computer-vision

[13] Vinyals et al. (2014). Show and Tell: A Neural Image Caption Generator. [Online] arXiv:

1411.4555. Available: arXiv:1411.4555v2

[14] Geeky is Awesome. (2016). Using beam search to generate the most probable sentence. [Blog] Geeky is Awesome (geekyisawesome.blogspot.ie). Available: https://geekyisawesome.blogspot.ie/2016/10/using-beam-search-to-generate-most.html

[15] Donahue et al. (2014). Long-term Recurrent Convolutional Networks for Visual Recognition and Description. [Online] arXiv: 1411.4389. Available: arXiv:1411.4389v4 (2016 version)

[16] Fang et al. (2014). From Captions to Visual Concepts and Back. [Online] arXiv: 1411.4952. Available: arXiv:1411.4952v3 (2015 version)

[17] Vinyals et al. (2014). Show and Tell: A Neural Image Caption Generator. [Online] arXiv: 1411.4555. Available: arXiv:1411.4555v2 (2015 version).

[18] Karpathy, A., Fei-Fei, L. (2015). Deep Visual-Semantic Alignments for Generating Image Descriptions. [Online] Stanford Computer Science Department (cs.stanford.edu). Available: https://cs.stanford.edu/people/karpathy/cvpr2015.pdf. For additonal code, etc., see project page: https://cs.stanford.edu/people/karpathy/deepimagesent/

[19] Johnson, J., Karpathy, A., Fei-Fei., L. (2015). DenseCap: Fully Convolutional Localization Networks for Dense Captioning. [Online] arXiv: 1511.07571. Available: arXiv:1511.07571v1

[20] CNN computation only according to: Ren et al. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. [Online] arXiv: 1506.01497. Available: arXiv:1506.01497v3 (2016 version).

[21] Mun et al. (2016). Text-guided Attention Model for Image Captioning. [Online] arXiv: 1612.03557. Available: arXiv:1612.03557v1

[22] Liu et al. (2017). MAT: A Multimodal Attentive Translator for Image Captioning. [Online] arXiv: 1702.05658. Available: arXiv:1702.05658v3

[23] Anderson et al. (2017). Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. [Online] arXiv: 1707.07998. Available: arXiv:1707.07998v3 (2018 version)

[24] peteanderson80 (GitHub). (2018). Up-Down-Captioner. [Online] Automatic Image Captioning Model by PeteAnderson80 (Github.com). Available: https://github.com/peteanderson80/Up-Down-Captioner. See publication: Anderson et al. (2017). Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. [Online] arXiv: 1707.07998. Available: arXiv:1707.07998v3 (2018 version)

[25] Chen et al. (2016). SCA-CNN: Spatial and Channel-wise Attention in Convolutional Networks for Image Captioning. [Online] arXiv: 1611.05594. Available: arXiv:1611.05594v2

[26] You et al. (2016). Image Captioning with Semantic Attention. [Online] arXiv: 1603.03925. Available: arXiv:1603.03925v1

[27] Brtiz, D. (2016). Attention and Memory in Deep Learning and NLP. [Blog] WildML (www.wildml.com). Available: http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

[28] Xu et al. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. [Online] arXiv: 1502.03044. Available: arXiv:1502.03044v3 (2016 version)

[29] Olah, C., Carter, S. (2016). Attention and Augmented Recurrent Neural Networks. [Online] Distill (distill.pub). Available:https://distill.pub/2016/augmented-rnns/

[30] Kiros et al. (2014). Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. [Online] arXiv: 1411.2539. Available: arXiv:1411.2539v1

[31] Kiros et al. (2014). Multimodal Neural Language Models. Proceedings of the 31st International Conference on Machine Learning, PMLR 32(2):595–603. Available: http://proceedings.mlr.press/v32/kiros14.html

[32] Lu et al. (2016). Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. [Online] arXiv: 1612.01887. Available: arXiv:1612.01887v2 (2017 version).

Acknowledgments

We are thankful to our college Shah And Anchor Kutchhi Engineering College for considering our project and extending help at all stages needed during our work of collecting information regarding the project.

We are deeply indebted to our Principal, **Dr. Bhavesh Patel** and Head of Computer Department, **Mr. Uday Bhave** giving us this valuable opportunity to do this project. We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully.

It gives us immense pleasure to express our deep and sincere gratitude to our project guide **Prof. Radhika F.** for her kind help and valuable advice during the development of project synopsis and for her guidance and suggestions.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is a great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project and we also want to thank our families who provided moral support and encouragement to us at several times.