INDUSTRIAL TRAINING REPORT

ON

PIXEL PHRASER

Submitted in partial fulfillment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Submitted By

Archi Agrawal
03815603119

Department of Information Technology
Dr. Akhilesh Das Gupta Institute of Technology & Management
Guru Gobind Singh Indraprastha University
Dwarka, Delhi-110078

CERTIFICATE

OF INTERNSHIP

September 09, 2022

To Whom It May Concern

This is to certify that **Archi Agarwal** (emp#**00012**) has completed internship program with TECHYAI as part of academic requirements.

The particulars of the internship program are as follows:
Project start date : **July 08, 2022**
Project end date : **September 08, 2022**
Project Role: **Machine Learning**

[illegible]

Yash Jha
Human Rresource

B-2, Manomay bunglow, Anjali Nagar, Hendre Pada, Badlapur| Maharashtra 421503| India| contact@techyai.com|https://g.co/kgs/EjFKfL

ACKNOWLEDGEMENT

I would like to acknowledge the contributions of the following people without whose help and guidance this industrial training report would not have been completed.

I acknowledge the counsel and support of my mentor, **Aakash Jadhav**, whose expertise, guidance, support, encouragement, and enthusiasm made this internship an enriching and enthralling experience. I am indeed proud and fortunate to be supervised by him. I would like to thank **Yash Jha**, **Human Resource, TechyAI** for the smooth on boarding process and the entire interview panel of TechyAI for recognizing my potential.

I extend my respect and gratitude to **Ms. Charul Dewan, Departmental In-charge, IT department,** for her constant encouragement, valuable suggestions and moral support and blessings.

I am immensely thankful to our esteemed, **Prof. (Dr.) Sanjay Kumar, Director, Dr. Akhilesh Das Gupta Institute of Technology & Management, New Delhi** for his never-ending motivation and support.

Finally, yet importantly, I would like to express my heartfelt thanks to everyone for his/her hard work and/or wishes for the successful completion of this project.

ABSTRACT

Image captioning is the task of generating text descriptions of images. This is a quickly-growing research area in computer vision, suggesting more intelligence of the machine than mere classification or detection. In this project named Pixel Phraser, I have designed an image captioning system using recurrent neural networks (RNNs) and attention models. This model uses Computer Vision to identify patterns or details of images and Natural Language Processing for understanding human language in order to generate captions in English language. RNNs are variants of the neural network paradigm that make predictions over sequences of inputs. For each sequence element, outputs from previous elements are used as inputs, in combination with new sequence data. This gives the networks a sort of memory which might make captions more informative and context aware. RNNs tend to be computationally expensive to train and evaluate, so in practice memory is limited to just a few elements. Attention models help address this problem by selecting the most relevant elements, focusing attention on a small fraction of the visual scene, from a larger bank of input data. In this work, I developed a system which extracts features from images using a convolutional neural network (CNN), combines the features with an attention model, and generates audio captions with an RNN using Google Text-to-Speech mechanism.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1.1 INTRODUCTION

In the past few years, neural networks have fueled dramatic advances in image classification. Emboldened, researchers are looking for more challenging applications for computer vision and artificial intelligence systems. They seek not only to assign numerical labels to input data, but to describe the world in human terms. Image and video captioning is among the most popular applications in this trend toward more intelligent computing.

Image Captioning is the task of describing the content of an image in words. This task lies at the intersection of computer vision and natural language processing. Most image captioning systems use an encoder-decoder framework, where an input image is encoded into an intermediate representation of the information in the image, and then decoded into a descriptive text sequence. The most popular benchmarks are nocaps and COCO, and models are typically evaluated according to a BLEU or CIDER metric. In this project, attention based architecture for image captioning has been introduced.

Attention mechanism has been a go-to methodology for practitioners in the Deep Learning community. It was originally designed in the context of Neural Machine Translation using Seq2Seq Models, but today we'll take a look at its implementation in Image Captioning. Rather than compressing an entire image into a static representation, the Attention mechanism allows for salient features to dynamically come to the forefront as and when needed. This is especially important when there is a lot of clutter in an image.

1.2 BASIC TERMS OF PROJECT

1.2.1 ARTIFICIAL INTELLIGENCE

While a number of definitions of artificial intelligence (AI) have surfaced over the last few decades, John McCarthy offers the following definition in his 2004 paper, "It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable."

1.2.2 MACHINE LEARNING

According to IBM, machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, and to uncover key insights in data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics.

1.2.3 DEEP LEARNING

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amount of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies.

1.2.4 RECURRENT NEURAL NETWORKs

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate. Like feed forward and convolutional neural networks (CNNs), recurrent neural networks utilize training data to learn. They are distinguished by their “memory” as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depends on the prior elements within the sequence.

1.2.5 COMPUTER VISION

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

1.2.6 NATURAN LANGUAGE PROCESSING

Web Natural language processing (NLP) refers to the branch of computer science and more specifically, the branch of Artificial Intelligence concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning, complete with the speaker or writer's intent and sentiment. NLP drives computer program that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time.

1.2.7 ATTENTION MECHANISM

Attention mechanism has been a go-to methodology for practitioners in the Deep Learning community. It was originally designed in the context of Neural Machine Translation using Seq2Seq Models, but today we'll take a look at its implementation in Image Captioning. Rather than compressing an entire image into a static representation, the Attention mechanism allows for salient features to dynamically come to the forefront as and when needed. This is especially important when there is a lot of clutter in an image.

1.3 LITERATURE OVERVIEW

Recently several high-quality surveys on image captioning with deep neural net-works [3, 7] and other techniques such as template generation and slot filling [6, 8,1] have been published. Instead of reviewing all methods used for image captioning, we record the success the use of attention mechanisms has brought to image captioning. After briefly reviewing the history of attention mechanisms in image captioning, we review the state-of-the-art literature to highlight the most successful attention mechanisms. This is what mainly differentiates our work from previous surveys on deep learning models for image captioning.

In general, researchers have recently noticed the usefulness of self-attention and multi-head attention over recurrent neural networks. The Transformer [9] model utilizes self-attention (scaled dot-attention) inside multi-head attention to obviate the use of recurrences and convolutions, relying fully on attention for neural machine translation. We provide a brief review of the methods that use the Transformer model or its variants for image captioning as well as other methods that use LSTMs with attention mechanisms.

A comprehensive survey of deep learning models for image captioning by Hos- sain et. al. [39], creates a well-defined taxonomy for the models. However, the most successful methods published after this paper have used Transformers [9] or variations of this model. The same applies to the survey by Liu et al. [7].

Although these surveys provide good background on evaluation metrics, datasets and methods used for automatic image caption generation, for the most part they do not discuss the subtle differences among the attention mechanisms used in the deep learning models for image captioning. This is mainly because the methods published after these surveys used bottom-up attention encoders and Transformer or components of this model such as multi-head attention, which improves perfor- mance by adding more attention units in a parallel manner. Multi-head attention was introduced by Vaswani et al. [9].
Another survey by Bai and An [6], reviews methods that use deep learning as well as other non-empirical methods. This survey and other similar surveys performed by Sharma et al. [8] and Li et al. [4] on automatic image caption generation do not perform a review of attentive

deep learning models for image captioning.

A noteworthy survey by Pavlopoulos et al. [7], reviews, deep learning models and other techniques used for medical image captioning. Given a large dataset of medical images and relevant diagnosis sentences for each image, it is possible to train deep learning models used for image captioning to generate candidate diagnosis sentences. This is likely to help physicians and save them considerable time when performing diagnosis procedures using visual medical data. This survey also reviews the datasets used for medical image captioning. The comprehensive survey performed by Hossain et al. [3], offers useful in- formation regarding common datasets, metrics and categories for deep learning models used for image captioning. We avoid repeating the same information and instead focus on the state-of-the-art models that use attention mechanisms. We offer a new taxonomy for attentive deep learning models used for image captioning.

A computer screen with a Windows message about Microsoft license terms.

A can of green beans is sitting on a counter in a kitchen.

A photo taken from a residential street in front of some homes with a stormy sky above.

A blue sky with fluffy clouds, taken from a car while driving on the highway.

A hand holds up a can of Coors Light in front of an outdoor scene with a dog on a porch.

A digital thermometer resting on a wooden table, showing 38.5 degrees Celsius.

A Winnie The Pooh character high chair with a can of Yoohoo sitting on it in front of a white wall.

A cup holder in a car holding loose change from Canada.

Figure 1. Image Captioning

METHODOLOGY ADOPTED

1.1 STRATEGY

The first phase of this project was to define a strategy for evolving the idea into a successful application. In this phase we tried to:

- Identify our users
- Assess resources, objectives and capabilities
- Research the competition
- Established the goals

1.2 ARCHITECTURE

1.2.1 Attentive Deep Learning for Image Captioning

In this section, first we discuss the early use of attention mechanisms in image captioning deep models. We follow by the introduction of bottom-up and top-down attention [3] (Up-Down Attention), which became a source of inspiration for most of the later work. In recent years, the use of generative adversarial networks (GANs) for image captioning has also led to good results [19, 9, 69]. In Comparison with the encoder-decoder architecture, which is usually trained with cross-entropy loss, GAN architectures are trained with adversarial loss, making it impossible to perform a direct comparison of performance. Although recently attention mech-anism was employed inside a GAN model [100], this model utilizes the encoder- decoder architecture inside the generator and discriminator modules in order to make use of attention. Considering that attention mechanisms have been widely used in encoder-decoder architectures, we present the way attention is calculated and used in these architectures.

After a review of history of attention mechanisms used for image captioning in the context of encoder-decoder architecture, we elaborate upon the state-of-the-art attention mechanisms in the context of different kinds of encoders in which they were used. Different types of encoders provide the attention mechanisms and paired decoders with different kinds of input information. Therefore, it is Necessary to analyze the differences among attention mechanisms in the context of the associated encoders.

We have referred to the deep learning models that use attention mechanisms as attentive deep learning models. A taxonomy graph of technologies associated with attentive deep image captioning is illustrated below. In our survey, we focus on attention mechanisms in the context of state-of-the-art encoder-decoder architectures for image captioning.

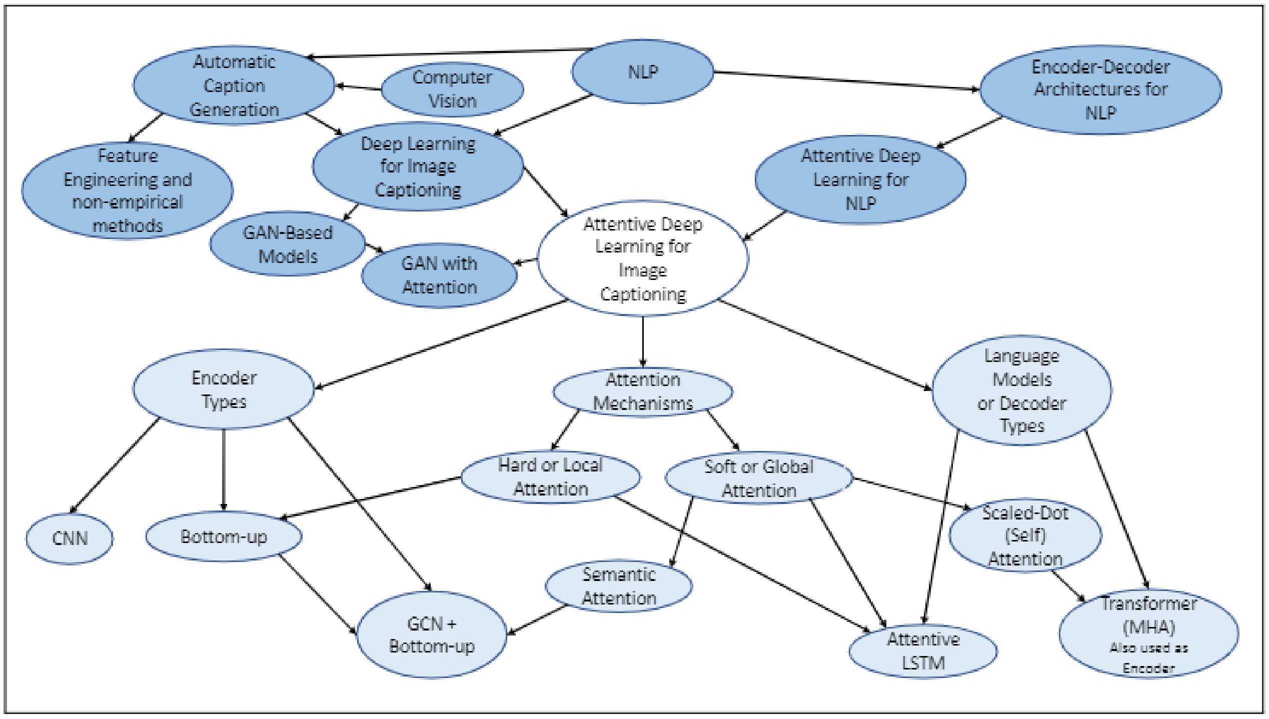

Automatic Caption Generation
Computer Vision
NLP
Encoder-Decoder Architectures for NLP
Feature Engineering and non-empirical methods
Deep Learning for Image Captioning
Attentive Deep Learning for NLP
GAN-Based Models
GAN with Attention
Attentive Deep Learning for Image Captioning
Encoder Types
Attention Mechanisms
Language Models or Decoder Types
Hard or Local Attention
Soft or Global Attention
CNN
Bottom-up
Scaled-Dot (Self) Attention
Semantic Attention
GCN + Bottom-up
Attentive LSTM
Transformer (MHA) Also used as Encoder

Figure 2. (Self) Attention - NLP: Natural Language Processing

1.2.2 CNN Encoders & Attentive Decoders

There are a few state-of-the-art models reviewed in this survey that use the traditional CNN encoder, only using a CNN for feature extraction over the input image as whole. Fig. 3 shows the general framework for models that employ CNNs for feature extraction alongside an attentive decoder for image captioning. The Recurrent Fusion Network model is an example of using only a CNN backbone as the encoder part of the model and utilizing attentive LSTM as the decoder. This model employs multiple CNN networks for feature extraction, where the extracted features are sent to multiple LSTMs in the first fusion stage to create the soft multi-attention mechanism. The average of hidden states from LSTMs in the first fusion stage is sent to LSTMs in the second fusion stage alongside multi-attention values. The attention values from each stage are concatenated to form the multi attention values. The hidden states of LSTMs in the second fusion stage are used in a soft-attention mechanism used in LSTM unit as decoder.

The captioning model considers different personality categories while generating captions, thus resulting in personalized captions that carry certain sentimental information. For this purpose, a CNN (ResNext 32 × 48d [101]) is used as image encoder and a linear transformation of personality vector is used for encoding personality information. In the decoder, LSTM is used like how it was utilized in Show and Tell, Show Attend and Tell, and the soft top-down decoder in Up-Down attention model as caption decoders. They achieve the best results from the soft top-down attention decoder in Up-Down atttention model. The visual information is used in soft spatial attention. The embedding of personality trait is used as additional input for LSTM in the decoder. This model employs ResNet152 and ResNext32 × 48d as image encoders. As mentioned earlier, most of the state-of-the-art methods use ResNet101 or Faster-RCNN with ResNet101 as image encoder.

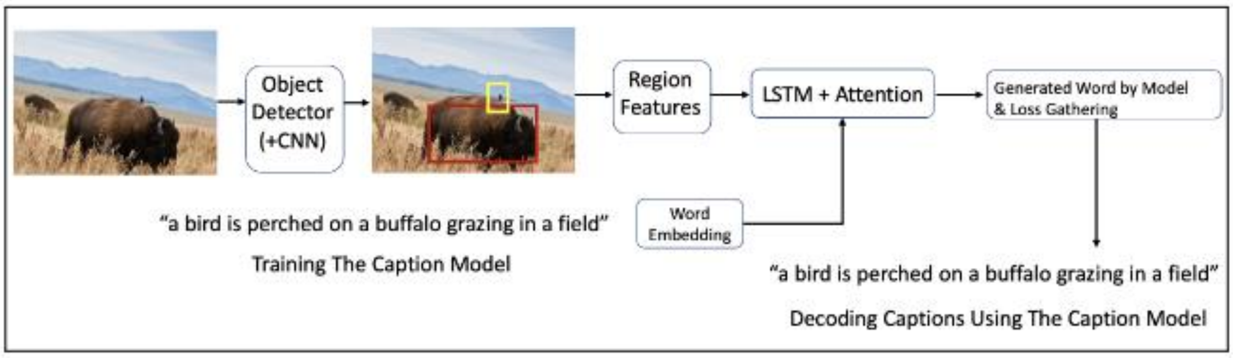

Object Detector (+CNN)
Region Features
LSTM + Attention
Generated Word by Model & Loss Gathering
"a bird is perched on a buffalo grazing in a field"
Training The Caption Model
Word Embedding
"a bird is perched on a buffalo grazing in a field"
Decoding Captions Using The Caption Model

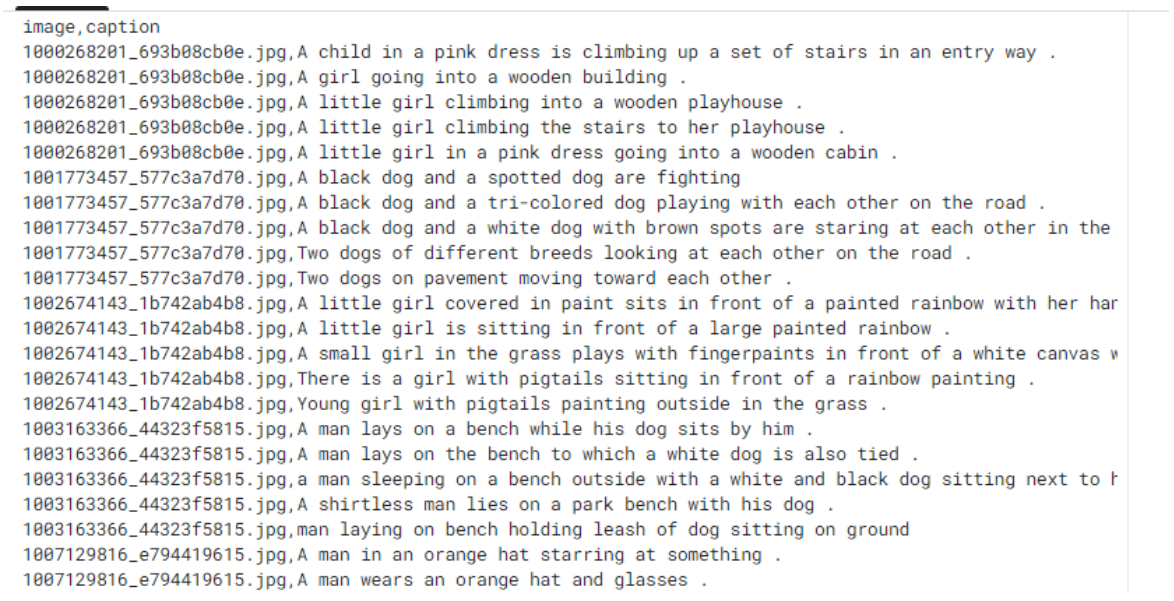
Figure 3. Attentive LSTM decoder with bottom-up attention encoder

1.3 DATASET USED

The dataset used is called Flickr8k. It introduced a new benchmark collection for sentence-based image description and search, consisting of 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events. The images were chosen from six different Flickr groups, and tend not to contain any well-known people or locations, but were manually selected to depict a variety of scenes and situations.

It is small in size. So, the model can be trained easily on low-end laptops/desktops. Data is properly labelled. For each image 5 captions are provided. The dataset is available for free. Following are the two subsets of this dataset:

- Flickr8k_Dataset: Contains a total of 8092 images in JPEG format with different shapes and sizes. Of which 6000 are used for training, 1000 for test and 1000 for development.
- Flickr8k_text : Contains text files describing train_set ,test_set. Flickr8k.token.txt contains 5 captions for each image i.e. total 40460 captions.



```
image,caption
1000268201_693b08cb0e.jpg,A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg,A girl going into a wooden building .
1000268201_693b08cb0e.jpg,A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg,A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg,A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg,A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg,A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg,A black dog and a white dog with brown spots are staring at each other in the
1001773457_577c3a7d70.jpg,Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg,Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg,A little girl covered in paint sits in front of a painted rainbow with her har
1002674143_1b742ab4b8.jpg,A little girl is sitting in front of a large painted rainbow .
1002674143_1b742ab4b8.jpg,A small girl in the grass plays with fingerpaints in front of a white canvas w
1002674143_1b742ab4b8.jpg,There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_1b742ab4b8.jpg,Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg,A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg,A man lays on the bench to which a white dog is also tied .
1003163366_44323f5815.jpg,a man sleeping on a bench outside with a white and black dog sitting next to h
1003163366_44323f5815.jpg,A shirtless man lies on a park bench with his dog .
1003163366_44323f5815.jpg,man laying on bench holding leash of dog sitting on ground
1007129816_e794419615.jpg,A man in an orange hat starring at something .
1007129816_e794419615.jpg,A man wears an orange hat and glasses .
```

Figure 4. Captions.txt file of Flickr8k dataset

"A man holding money

A man holds a dollar bill in front of his face while posing in front of a street band

A man holds money in the air

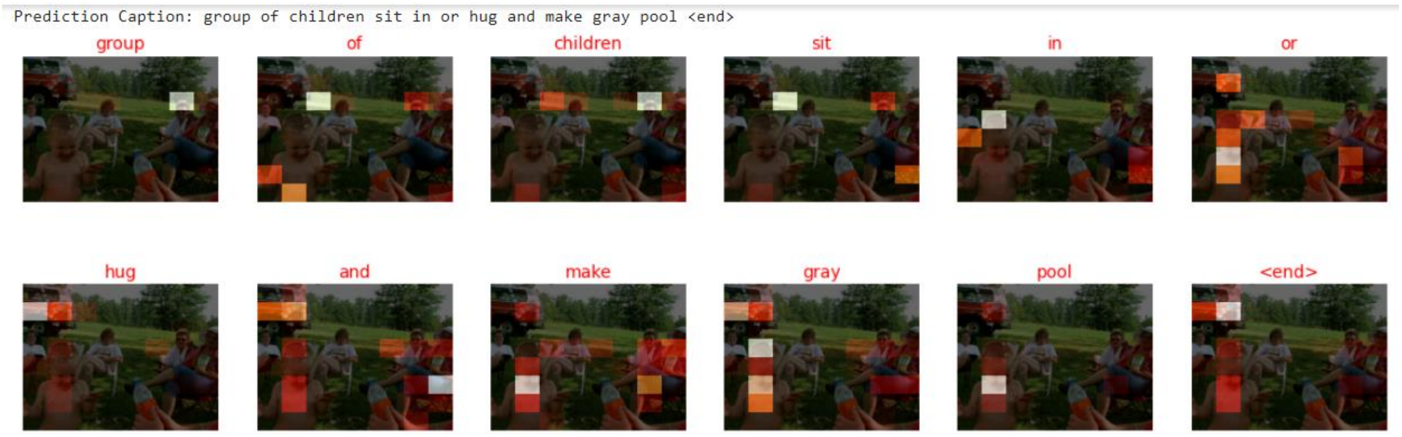
A man in a green jacket is standing outside a store holding some money in front of his face

"An African-American man wearing a green sweatshirt and blue vest is holding up 2 dollar bills in front of his face

Figure 5. Example picture with captions

1.4 DIGITAL ANALYSIS

1.4.1 Training



Prediction Caption: group of children sit in or hug and make gray pool <end>

Figure 6. Caption generation by attention model

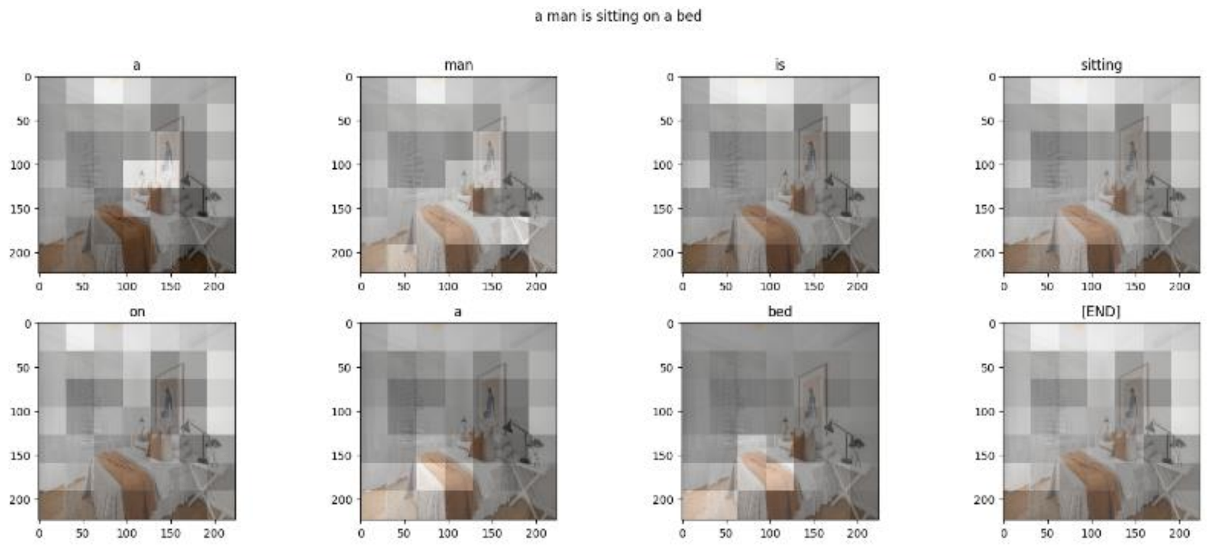


Figure 7. Caption generation by attention model

DESIGNING AND RESULT ANALYSIS

3.1 PROJECT INSIGHTS

3.1.1 BACK END LANGUAGES USED IN OUR APPLICATION

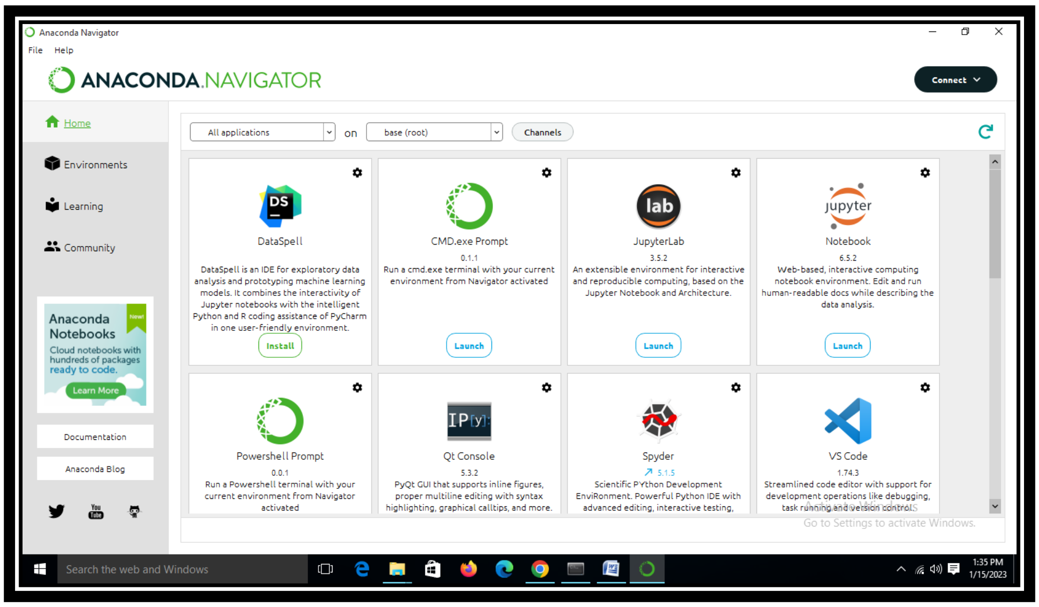
- **Python:** Python is a high-level, general-purpose and a very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting edge technology in Software Industry. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

- **Flask:** Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

3.1.2 SOFTWARES USED IN OUR APPLICATION

- **ANACONDA**

  Anaconda is an open-source package and environment management system that runs on Windows, macOS, and Linux. Conda quickly installs, runs, and updates packages and their dependencies. It also easily creates, saves, loads, and switches between environments on local computer. We have specifically used **Jupyter Notebook** on Conda for running python files and implementing opencv library.

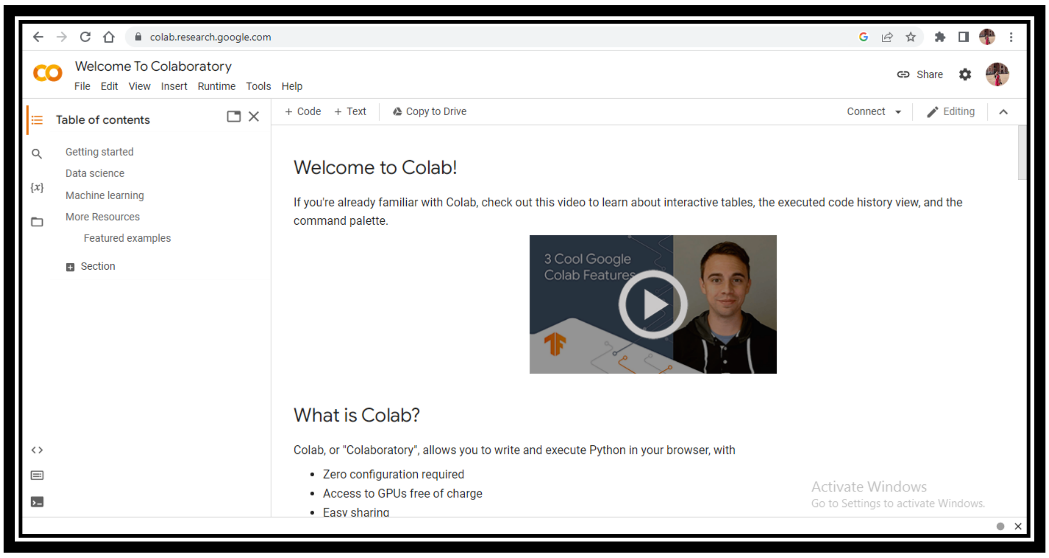


Anaconda Navigator

Figure 8

- **GOOGLE COLAB**

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. We used Colab hosted Jupyter notebook service that requires no setup to use, for training our model free of charge, accessing computing resources like GPUs.



Welcome to Colab!

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette.

What is Colab?

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

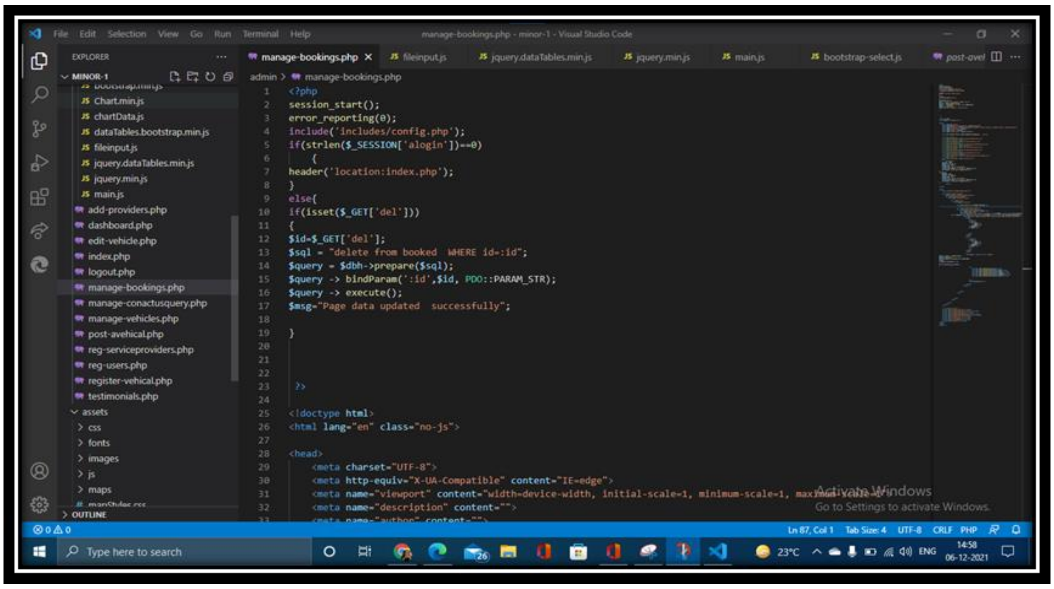
- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Google Colab

Figure 9

- **<u>VISUAL STUDIO CODE</u>**

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.



```
<?php
session_start();
error_reporting(0);
include('includes/config.php');
if(strlen($_SESSION['alogin'])==0)
    {
header('location:index.php');
}
else{
if(isset($_GET['del']))
{
$id=$_GET['del'];
$sql = "delete from booked  WHERE id=:id";
$query = $dbh->prepare($sql);
$query -> bindParam(':id',$id, PDO::PARAM_STR);
$query -> execute();
$msg="Page data updated  successfully";

}



 ?>

<!doctype html>
<html lang="en" class="no-js">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1, max
    <meta name="description" content="">
```

Visual Studio Code

Figure 10

3.1.3 Model Architecture

Attention Model

```
class Attention(tf.keras.Model):
  def __init__(self, units):
    super(Attention, self).__init__()
    self.W1 = tf.keras.layers.Dense(units)
    self.W2 = tf.keras.layers.Dense(units)
    self.V = tf.keras.layers.Dense(1)

  def call(self, features, hidden):

    hidden_with_time_axis = tf.expand_dims(hidden, 1)
    attention_hidden_layer = (tf.nn.tanh(self.W1(features) +
                                                self.W2(hidden_with_t
ime_axis)))
    score = self.V(attention_hidden_layer)
    attention_weights = tf.nn.softmax(score, axis=1)
    context_vector = attention_weights * features
    context_vector = tf.reduce_sum(context_vector, axis=1)

    return context_vector, attention_weights
```

Encoder-Decoder

```
class Encoder(tf.keras.Model):
    def __init__(self, embedding_dim):
        super(Encoder, self).__init__()
        self.fc = tf.keras.layers.Dense(embedding_dim)

    def call(self, features):
        features = self.fc(features)
        features = tf.nn.relu(features)
        return features

class Decoder(tf.keras.Model):
    def __init__(self, embed_dim, units, vocab_size):
        super(Decoder, self).__init__()
        self.units=units
        self.attention = Attention(self.units)
        self.embed = tf.keras.layers.Embedding(vocab_size, emb
ed_dim)
        self.gru = tf.keras.layers.GRU(self.units,return_seque
nces=True,return_state=True,recurrent_initializer='glorot_unif
orm')
        self.d1 = tf.keras.layers.Dense(self.units)
        self.d2 = tf.keras.layers.Dense(vocab_size)
```

3.2 RESULTS AND DISCUSSION

As mentioned earlier, data is encoded to numbers via a dictionary of words. After the data is consumed by the pipeline, the output will also be in the encoded format which needs to be reversed back into English words in order to make sense to human. The other important thing is the output from RNN network is a series of likelihoods of words (likelihoods). Picking highest likelihood word at each decode step in RNN tend to yield a sub-optimal result. Instead, we have implemented Beam Search introduced in Section 2.7 which is a popular solution for finding optimal path for decoding natural language sentences. Some generated captions on test images are presented below. Apparently, the network generated captions are not all perfect, some of which miss important information in the image and others have misidentified visual features. For example, the top left image in Figure 11 is "A little girl in a white dress is playing a hunki". The ladder in the image is recognized by the CNN network but the RNN failed to generate the word "ladder". The use of hunki suggests that training images has few number of images with ladders or the word "ladder" is very rare in training captions.

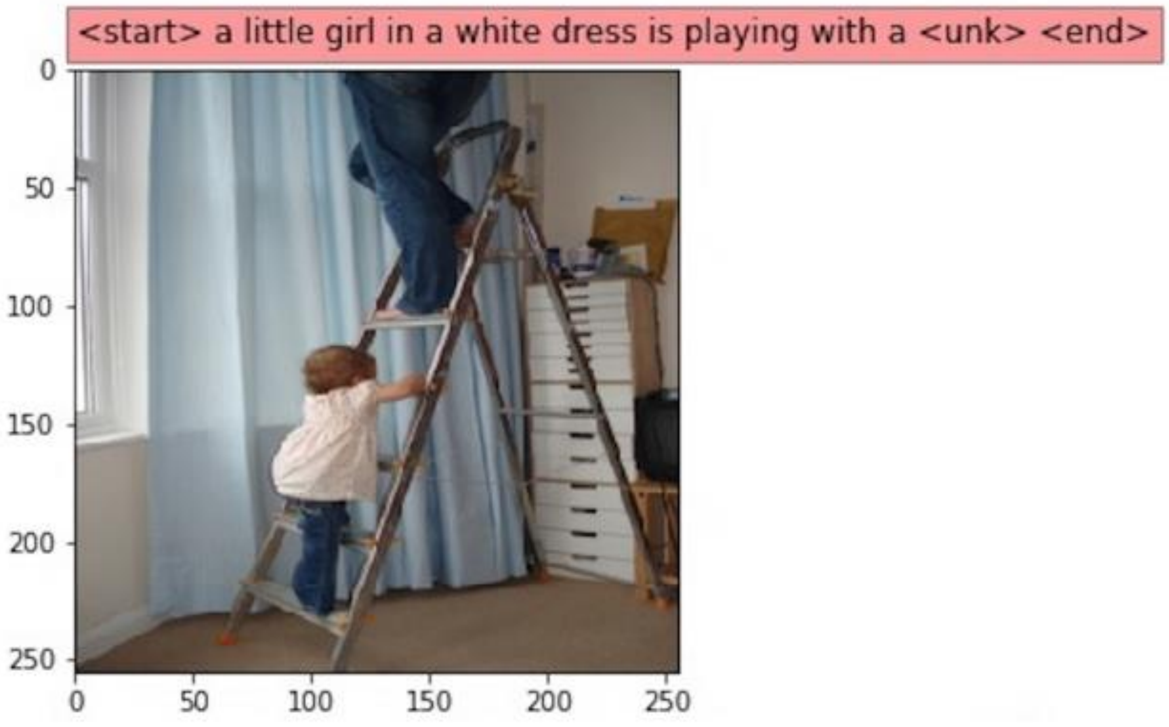

<start> a little girl in a white dress is playing with a <unk> <end>
0
50
100
150
200
250
0
50
100
150
200
250

Figure 11. An example of an incorrectly generated caption

The other flaw we observed is the network sometimes fails to separate objects. For example, in the right-most image on the fourth row has two dogs racing. But our network fails to recognize two dogs, mainly because two dogs are too close and have some parts appearing connected in the image. Therefore, the network thinks there is one black and white dog. Interestingly, we discovered that images of dogs catching Frisbee often have very accurately generated captions in test images and is likely due to large amount of training images are dogs jumping to catch a Frisbee. Also the pretrained CNN may also has lots of knowledge about what a dog looks like.

3.2.1 Generated Captions

As shown in Figure 12, Figure 13 and Figure 14, we have successfully generated mostly grammarly correct and human readable captions describing what is happening in the image. Most of the images correctly state what objects appear in the scene, count number of appearance and gives an intuitively correct verb to logically complete the sentence. Colors of the object and spatial relationships between object are also well captured. For example, the bottom right image has a caption ”A man in a red shirt is standing on a grassy hill”. Objects in the images are identified as ”A Man”, ”Shirt” and ”Grassy hill”. Then CNN is also able to capture properties of objects such as shirt’s color is ”red”. Lastly, RNN and Attention Mechanism will connect these words logically by conjunctions into a valid sentence.

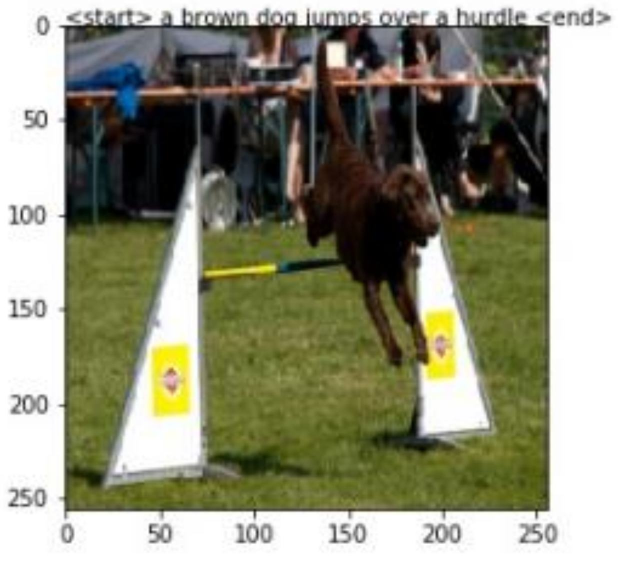


Figure 12 . Generated Caption :A brown dog jumps over a hurdle

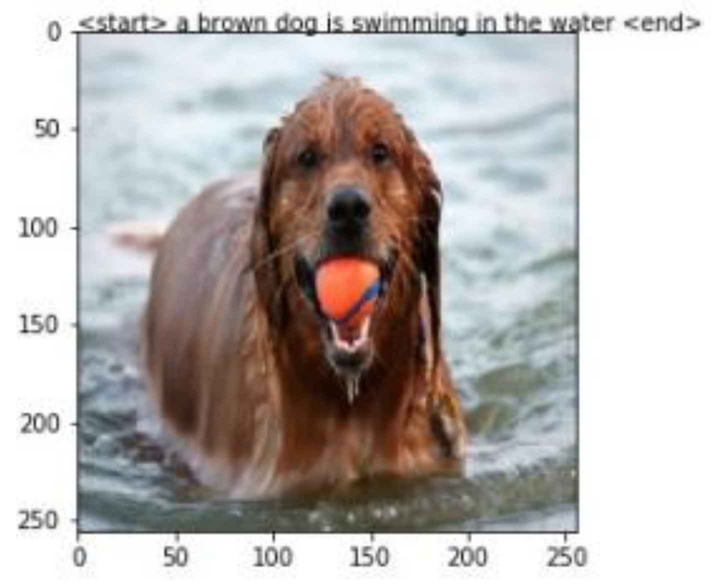


Figure 13. Generated Caption: A brown dog is swimming in the water

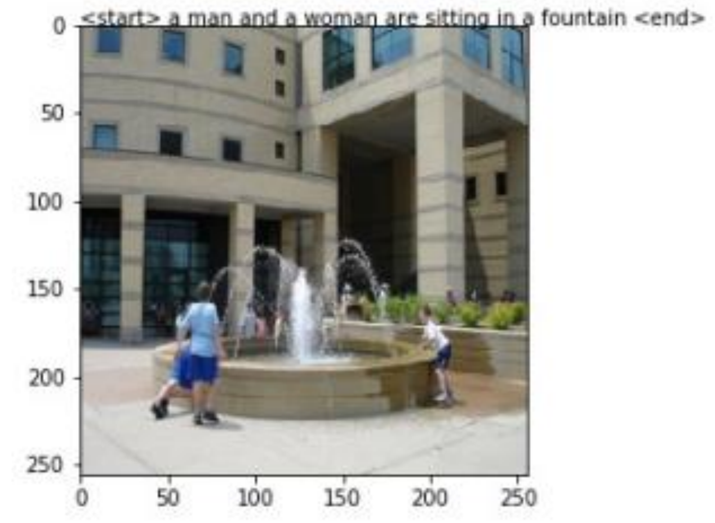


Figure 14. Generated Caption man and a woman are sitting in a fountain

3.2.2 Attention Mechanism Visualization

Visualizing attention mechanism allows us to understand which part of the image is being focused when generating a specific word, which is essential for improving the network's scene understanding capability. We implemented our visualization by adding a semi-transparent mask on top of the original image at each RNN node (each word), such that the bright region denotes the focus area and dark region has little influence on the generated word. As we can see above, the network has different focuses when generating different words.

MERITS, DEMERITS AND APPLICATIONS

4.1 MERITS

1. Recent advances in deep learning methods on perceptual tasks, such as image classification and object detection have encouraged researchers to tackle even more difficult problems for which recognition is just a step towards to more complex reasoning about our visual world. Image captioning is one of such tasks.
2. The aim of image captioning is to automatically describe an image with one or more natural language sentences. This is a problem that integrates computer vision and natural language processing, so its main challenges arise from the need of translating between two distinct, but usually paired, modalities.
3. First, it is necessary to detect objects on the scene and determine the relationships between them and then, express the image content correctly with properly formed sentences. The generated description is still much different from the way people describe images because people rely on common sense and experience, point out important details and ignore objects and relationships that they imply. Moreover, they often use imagination to make descriptions vivid and interesting. Regardless of the existing limitations, image captioning has already been proven to have useful applications, such as helping visually impaired people in performing daily tasks.
4. Automatically generated descriptions can also be used for content-based retrieval or in social media communications. Early image captioning approaches relied on the use of predefined templates, which were filled in based on the results of the detection of elements on the scene.
5. Generated sentences were too simple, lacking the fluency of human writing. Moreover, such systems were heavily hand-designed, which constrained their flexibility. Some authors have reformulated image captioning as a ranking task. Ranking-based approaches always return well-formed sentences, but they cannot generate new sentences or to describe compositionally new images , i.e., those containing objects that were observed during training but appear in different combinations on the test image.

4.2 DEMERITS

1. **Compositionality and Naturalness**

The first challenge stems from the compositional nature of natural language and visual scenes. While the training dataset contains co-occurrences of some objects in their context, a captioning system should be able to generalize by composing objects in other contexts. Traditional captioning systems suffer from lack of compositionality and naturalness as they often generate captions in a sequential manner, i.e., next generated word depends on both the previous word and the image feature. This can frequently lead to syntactically correct, but semantically irrelevant language structures, as well as to a lack of diversity in the generated captions. We propose to address the compositionality issue with a context-aware Attention captioning model, which allows the captioner to compose sentences based on fragments of the observed visual scenes. Specifically, we used a recurrent language model with a gated recurrent visual attention that gives the choice at every generating step of attending to either visual or textual cues from the last generation step

2. Generalization

The second challenge is the dataset bias impacting current captioning systems.
The trained models overfit to the common objects that co-occur in a common context (e.g., bed and bedroom), which leads to a problem where such systems struggle to generalize to scenes where the same objects appear in unseen contexts (e.g., bed and forest). Although reducing the dataset bias is in itself a challenging, open research problem, we propose a diagnostic tool to quantify how biased a given captioning system is.

3. Evaluation and Turing Test

The third challenge is in the evaluation of the quality of generated captions. Using automated metrics, though partially helpful, is still unsatisfactory since they do not take the image into account. In many cases, their scoring remains inadequate and sometimes even misleading — especially when scoring diverse and descriptive captions. Human evaluation remains a gold standard in scoring captioning systems.

4.3 APPLICATIONS

There are a number of applications wherever images and deep learning involved. Mapping natural language to images and vice versa is more effective than any other thing. my favorite application would be medical image understanding that conveys the clinical physician that the algorithm had found certain things fishy in the patient by mapping the physiological parameters and images which may require further investigations for confirmation. It is just one example. So you venture out, explore in literature and surely will find a lot of problem are open and yet to be solved but here are a few other applications listed below:

1. SkinVision : Lets you confirm weather a skin condition can be skin cancer or not.
2. Google Photos: Classify your photo into Mountains, sea etc.
3. Deepmind: Achieved superhuman level playing Game Atari.
4. Facebook: Using AI to classify, segmentate and finding patterns in pictures.
5. A U.S. company is predicting crop yield using images from satellite.
6. Fed Ex and other courier services: Are using hand written digit recognition system from may times now to detect pin code correctly.
7. Picasa : Using facial Recognition to identify your friends and you in a group picture.
8. Tesla/Google Self Drive Cars: All the self drive cars are using image/video processing with neural network to attain their goal.
9. Automatic Image captioning requires both Image analysis and neural network.

CONCLUSION AND SCOPE OF THE PROJECT

5.1 CONCLUSION

In this work we have designed an Image Captioning system using Attention model and NLP to generate audio captions of the fed image by virtue of Google text-to-speech library. The model can be converted or incorporated into a web application using Flask for public use.
We observed that attention based network along with encoders and decoders give us a fair accuracy over other models such as CNN which trains an image as a whole rather than being attentive towards certain features and extracting those features.
Image captioning has become a major translation task in vision-language, combining the principles of computer vision and neural machine translation. In the future, we are likely to witness widespread use of attention mechanisms in neural image captioning systems in various tasks. Medical image captioning is an instance of a task which may be beneficial to the medical community. Image captioning performed on mobile devices may become useful to visually impaired, especially when communicated verbally. Bringing visual intelligence for robots and enhancing visual recommendation systems and visual assistant systems are among other applications of neural image captioning. Attention mechanisms have shown the ability to map important parts of multi-modal information for use in downstream tasks such as image captioning. Improving visual attention mechanisms (such as bottom- up, SCA-CNN, Areas-of-Attention) and multi-modal (soft) attention mechanisms as well as multi-head attention and self-attention for image captioning remains an interesting challenge for the research community.

5.2 SCOPE OF THE PROJECT

Automatically image captioning is far from mature and there are a lot of ongoing research projects aiming for more accurate image feature extraction and semantically better sentence generation. We successfully completed what we mentioned in the project proposal, but used a smaller dataset (Flickr8k) due to limited computational power. There can be potential improvements if given more time. First of all, we directly used pre-trained CNN network as part of our pipeline without fine-tuning, so the network does not adapt to this specific training dataset. Thus, by experimenting with different CNN pre-trained networks and enabling fine-tuning, we expect to achieve a slightly higher BLEU4 score. Another potential improvement is by training on a combination of Flickr8k, Flickr30k, and MSCOCO. In general, the more diverse training dataset the network has seen, the more accurate the output will be. We all agree this project ignites our interest in application of Machine Learning knowledge in Computer Vision and expects to explore more in the future. To further adapt our models to the real world, we hope to benefit blind people via this project.

There are still many challenges to be addresses, such as dealing with unseen images or objects. Moreover, more experimentation could be done with the problem of data imbalance. We would like to improve the robustness and accuracy of our model by applying different data augmentation techniques to address varying camera brightness and angle issues. For better deployment results, one could use a quantized model that could run on the GPU/TPU. Additionally, we would like to apply our work to benefit humanity, such as by employing it to support shared empathy.

REFERENCES

[1] Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA. ACL (2002)

[2] Anderson, P., Fernando, B., Johnson, M., Gould, S.: Spice: Semantic propositional image caption evaluation. In: ECCV (2016)

[3] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L.: Bottom- up and top-down attention for image captioning and visual question answering. In: CVPR (2018)

[4] Aneja, J., Deshpande, A., Schwing, A.G.: Convolutional image captioning. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5561–5570 (2018). DOI 10.1109/CVPR.2018.00583

[5] Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: 3rd International Conference on Learning Representations, ICLR 2015 (2015)

[6] Bai, S., An, S.: A survey on automatic image caption generation. Neurocomputing 311, 291 – 304 (2018). DOI https://doi.org/10.1016/j.neucom.2018.05.080. URL http://www.sciencedirect.com/science/article/pii/S0925231218306659

[7] Bengio, Y., LeCun, Y., Henderson, D.: Globally trained handwritten word recognizer us- ing spatial representation, convolutional neural networks, and hidden markov models. In: Advances in Neural Information Processing Systems 6, pp. 937–944. Morgan-Kaufmann (1994)

[8] Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: Y. Bengio, Y. LeCun (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014). URL http://arxiv.org/abs/1312.620

[9] Chen, C., Mu, S., Xiao, W., Ye, Z., Wu, L., Ju, Q.: Improving image captioning with conditional generative adversarial nets. Proceedings of the AAAI Conference on Artificial Intelligence 33(01), 8142–8150 (2019). DOI 10.1609/aaai.v33i01.33018142

[10] Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W., Chua, T.S.: 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) (2017)

[11] Chen, S., Jin, Q., Wang, P., Wu, Q.: Say as you wish: Fine-grained control of

image caption generation with abstract scene graphs. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[12] Chen, S., Zhao, Q.: Boosted attention: Leveraging human attention for image captioning. In: The European Conference on Computer Vision (ECCV) (2018

[13] Cho, K., Courville, A., Bengio, Y.: Describing multimedia content using attention-based encoder-decoder networks. IEEE Transactions on Multimedia 17(11), 1875–1886 (2015). DOI 10.1109/TMM.2015.2477044

[14] Cho, K., Courville, A.C., Bengio, Y.: Describing multimedia content using attention- based encoder-decoder networks. IEEE Trans. Multimedia 17(11), 1875–1886 (2015)

[15] Cho, K., van Merri ̈enboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural ma- chine translation: Encoder–decoder approaches. In: Proceedings of SSST-8, Eighth Work- shop on Syntax, Semantics and Structure in Statistical Translation, pp. 103–111. Association for Computational Linguistics, Doha, Qatar (2014). DOI 10.3115/v1/W14-4012. URL https://www.aclweb.org/anthology/W14-4012

[16] Cho, K., van Merri ̈enboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734. ACL, Doha, Qatar (2014). DOI10.3115/v1/D14-1179

[17] Cornia, M., Baraldi, L., Cucchiara, R.: Show, control and tell: A framework for generating controllable and grounded captions. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

[18] Cornia, M., Stefanini, M., Baraldi, L., Cucchiara, R.: Meshed-memory transformer for image captioning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[19] Dai, B., Fidler, S., Urtasun, R., Lin, D.: Towards diverse and natural image descriptions via a conditional gan. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017)

[20] Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, vol. 29, pp. 3844–3852 (2016)

[21] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)

[22] Deubel, H., Schneider, W.X.: Saccade target selection and object recognition: Evidence for a common attentional mechanism. Vision Research 36(12), 1827 – 1837 (1996). DOI https://doi.org/10.1016/0042-6989(95)00294-4

[23] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)

[24] Fang, H., Gupta, S., Iandola, F., Srivastava, R.K., Deng, L., Dollar, P., Gao, J., He, X., Mitchell, M., Platt, J.C., Lawrence Zitnick, C., Zweig, G.: From captions to visual concepts and back. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)

[25] Farhadi, A., Hejrati, M., Sadeghi, M.A., Young, P., Rashtchian, C., Hockenmaier, J., Forsyth, D.: Every picture tells a story: Generating sentences from images. In: Computer Vision – ECCV 2010, pp. 15–29. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

[26] Feng, Y., Ma, L., Liu, W., Luo, J.: Unsupervised image captioning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

[27] Fu, K., Jin, J., Cui, R., Sha, F., Zhang, C.: Aligning where to see and what to tell: Image captioning with region-based attention and scene-specific contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence 39(12), 2321–2334 (2017). DOI 10.1109/TPAMI.2016.2642953

[28] Gao, L., Fan, K., Song, J., Liu, X., Xu, X., Shen, H.T.: Deliberate attention networks for image captioning. Proceedings of the AAAI Conference on Artificial Intelligence 33(01), 8320–8327 (2019). DOI 10.1609/aaai.v33i01.33018320

[29] Girshick, R.: Fast r-cnn. In: The IEEE International Conference on Computer Vision (ICCV) (2015)

[30] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014). DOI 10.1109/CVPR.2014.8

APPENDIX

main

```
import tensorflow as tf
import matplotlib.pyplot as plt
import collections
import random
import numpy as np
import os
import time
import json
from PIL import Image
pathi = os.path.join('/content/ImageCap', 'images')
pathf= os.path.join('/content/ImageCap', 'annotations')
os.makedirs(pathi)
os.makedirs(pathf)

!unzip '/content/drive/MyDrive/ImgCaptioning/Images.zip' -
d '/content/ImageCap/images'
!unzip '/content/drive/MyDrive/ImgCaptioning/captions.txt.zip' -
d '/content/ImageCap/annotations'

import glob
images='/content/ImageCap/images'
all_imgs = glob.glob(images + '/*.jpg',recursive=True)
len(all_imgs)
import imageio
Display_Images = all_imgs[0:5]
figure, axes = plt.subplots(1,5)
figure.set_figwidth(20)
for ax, image in zip(axes, Display_Images):
  ax.imshow(imageio.imread(image), cmap=None)
text_file = '/content/ImageCap/annotations/captions.txt'

def load_doc(filename):
    open_file = open(text_file, 'r', encoding='latin-1' )
    text = open_file.read()
    open_file.close()
    return text
doc = load_doc(text_file)
print(doc[:300])

# Can be ignored for csv files
import pandas as pd
img_path = '/content/ImageCap/images/'
```

```
all_img_id = []
all_img_vector = []
annotations = []

with open('/content/ImageCap/annotations/captions.txt' , 'r') as fo:
  next(fo)
  for line in fo :
    split_arr = line.split(',')
    all_img_id.append(split_arr[0])
    annotations.append(split_arr[1].rstrip('\n.'))
    all_img_vector.append(img_path+split_arr[0])

df = pd.DataFrame(list(zip(all_img_id, all_img_vector,annotations)),col
umns =['ID','Path', 'Captions'])

df

from collections import Counter
vocabulary = [word.lower() for line in annotations for word in line.spl
it()]

val_count = Counter(vocabulary)

def caption_with_img_plot(image_id, frame) :
  capt = ("\n" *2).join(frame[frame['ID'] == image_id].Captions.to_list
())
  fig, ax = plt.subplots()
  ax.set_axis_off()
  idx = df.ID.to_list().index(image_id)
  im =  Image.open(df.Path.iloc[idx])
  w, h = im.size[0], im.size[-1]
  ax.imshow(im)
  ax.text(w+50, h, capt, fontsize = 18, color = 'navy')
caption_with_img_plot(df.ID.iloc[8049], df)
import string
#data cleaning
rem_punct = str.maketrans('', '', string.punctuation)
for r in range(len(annotations)) :
  line = annotations[r]
  line = line.split()

  # converting to lowercase
  line = [word.lower() for word in line]

  # remove punctuation from each caption and hanging letters
  line = [word.translate(rem_punct) for word in line]
  line = [word for word in line if len(word) > 1]
  # remove numeric values
  line = [word for word in line if word.isalpha()]

  annotations[r] = ' '.join(line)
  #add the <start> & <end> token to all those captions as well
annotations = ['<start>' + ' ' + line + ' ' + '<end>' for line in annot
ations]

#Create a list which contains all the path to the images
all_img_path = all_img_vector
##list contatining captions for an image
annotations[0:5]

# Creating the tokenizer
from keras.preprocessing.text import Tokenizer
top_word_cnt = 5000
tokenizer = Tokenizer(num_words = top_word_cnt+1, filters= '!"#$%^&*()_
+.,:;-?/~`{}[]|\=@ ',
                      lower = True, char_level = False,
                      oov_token = 'UNK')
tokenizer.fit_on_texts(annotations)
train_seqs = tokenizer.texts_to_sequences(annotations)
tokenizer.word_index['PAD'] = 0
tokenizer.index_word[0] = 'PAD'
tokenizer_top_words = [word for line in annotations for word in line.sp
lit() ]

#tokenizer_top_words_count
tokenizer_top_words_count = collections.Counter(tokenizer_top_words)
tokens = tokenizer_top_words_count.most_common(30)
most_common_words_df = pd.DataFrame(tokens, columns = ['Word', 'Count']
)

train_seqs_len = [len(seq) for seq in train_seqs]
longest_word_length = max(train_seqs_len)
cap_vector= tf.keras.preprocessing.sequence.pad_sequences(train_seqs, p
adding= 'post', maxlen = longest_word_length,
                                                          dtype='int32'
, value=0)
print("The shape of Caption vector is :" + str(cap_vector.shape))
from sklearn.model_selection import train_test_split


def load_images(image_path):
    img = tf.io.read_file(image_path)
    img = tf.io.decode_jpeg(img, channels=3)
    img = tf.keras.layers.Resizing(299, 299)(img)
    img = tf.keras.applications.inception_v3.preprocess_input(img)
    return img, image_path

training_list = sorted(set(all_img_vector))
New_Img = tf.data.Dataset.from_tensor_slices(training_list)
New_Img = New_Img.map(load_images, num_parallel_calls = tf.data.experim
ental.AUTOTUNE)

New_Img = New_Img.batch(64, drop_remainder=False)

path_train, path_test, caption_train, caption_test = train_test_split(a
ll_img_vector, cap_vector,
                                                                       t
est_size = 0.2, random_state = 42)
image_model = tf.keras.applications.InceptionV3(include_top=False, weig
hts='imagenet')
new_input = image_model.input
hidden_layer = image_model.layers[-1].output
image_features_extract_model = tf.keras.Model(new_input, hidden_layer)

# extract features from each image in the dataset
from tqdm import tqdm
img_features = {}
for image, image_path in tqdm(New_Img) :
  batch_features = image_features_extract_model(image)
  batch_features_flattened = tf.reshape(batch_features, (batch_features
.shape[0], -1, batch_features.shape[3]))
  for batch_feat, path in zip(batch_features_flattened, image_path) :
    feature_path = path.numpy().decode('utf-8')
    img_features[feature_path] = batch_feat.numpy()

def map(image_name, caption):
    img_tensor = img_features[image_name.decode('utf-8')]
    return img_tensor, caption

BUFFER_SIZE = 1000
BATCH_SIZE = 64
embedding_dim = 256
units = 512
vocab_size = 5001
train_num_steps = len(path_train) // BATCH_SIZE
test_num_steps = len(path_test) // BATCH_SIZE
max_length = 31
feature_shape = batch_feat.shape[1]
attention_feature_shape = batch_feat.shape[0]

def gen_dataset(img, capt):
    data = tf.data.Dataset.from_tensor_slices((img, capt))
    data = data.map(lambda ele1, ele2 : tf.numpy_function(map, [ele1, e
le2], [tf.float32, tf.int32]),
                    num_parallel_calls = tf.data.experimental.AUTOTUNE)


    data = (data.shuffle(BUFFER_SIZE, reshuffle_each_iteration= True).b
atch(BATCH_SIZE, drop_remainder = False)
    .prefetch(tf.data.experimental.AUTOTUNE))
    return data

train_dataset = gen_dataset(path_train,caption_train)
test_dataset = gen_dataset(path_test,caption_test)

class Attention(tf.keras.Model):
  def __init__(self, units):
    super(Attention, self).__init__()
    self.W1 = tf.keras.layers.Dense(units)
    self.W2 = tf.keras.layers.Dense(units)
    self.V = tf.keras.layers.Dense(1)

  def call(self, features, hidden):

    hidden_with_time_axis = tf.expand_dims(hidden, 1)
    attention_hidden_layer = (tf.nn.tanh(self.W1(features) +
                                         self.W2(hidden_with_time_axis)
))
    score = self.V(attention_hidden_layer)
    attention_weights = tf.nn.softmax(score, axis=1)
    context_vector = attention_weights * features
    context_vector = tf.reduce_sum(context_vector, axis=1)

    return context_vector, attention_weights

class Encoder(tf.keras.Model):
    def __init__(self, embedding_dim):
        super(Encoder, self).__init__()
        self.fc = tf.keras.layers.Dense(embedding_dim)

    def call(self, features):
        features = self.fc(features)
        features = tf.nn.relu(features)
        return features

class Decoder(tf.keras.Model):
    def __init__(self, embed_dim, units, vocab_size):
        super(Decoder, self).__init__()
        self.units=units
        self.attention = Attention(self.units)
        self.embed = tf.keras.layers.Embedding(vocab_size, embed_dim)
        self.gru = tf.keras.layers.GRU(self.units,return_sequences=True
,return_state=True,recurrent_initializer='glorot_uniform')
        self.d1 = tf.keras.layers.Dense(self.units)
        self.d2 = tf.keras.layers.Dense(vocab_size)


    def call(self,x,features, hidden):
        context_vector, attention_weights = self.attention(features, hi
dden)
        embed = self.embed(x)
        embed = tf.concat([tf.expand_dims(context_vector, 1), embed], a
xis = -1)
        output,state = self.gru(embed)
        output = self.d1(output)
        output = tf.reshape(output, (-1, output.shape[2]))
        output = self.d2(output)

        return output, state, attention_weights

    def init_state(self, batch_size):
        return tf.zeros((batch_size, self.units))
encoder = Encoder(embedding_dim)
decoder=Decoder(embedding_dim, units, vocab_size)
optimizer = tf.keras.optimizers.Adam()
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True, reduction='none')

def loss_function(real, pred):
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  loss_ = loss_object(real, pred)
  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask
  return tf.reduce_mean(loss_)
checkpoint_path = "./checkpoints/train"
ckpt = tf.train.Checkpoint(encoder=encoder,
                           decoder=decoder,
                           optimizer = optimizer)
ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path, max_to
_keep=5)

start_epoch = 0

if ckpt_manager.latest_checkpoint:
    start_epoch = int(ckpt_manager.latest_checkpoint.split('-')[-1])

@tf.function
def train_step(img_tensor, target):
    loss = 0
    hidden = decoder.init_state(batch_size=target.shape[0])
    dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * targ
et.shape[0], 1)

    with tf.GradientTape() as tape:

        encoder_op = encoder(img_tensor)
        for r in range(1, target.shape[1]) :
          predictions, hidden, _ = decoder(dec_input, encoder_op, hidde
n)
          loss = loss + loss_function(target[:, r], predictions)
          dec_input = tf.expand_dims(target[:, r], 1)

    avg_loss = (loss/ int(target.shape[1])) #avg loss per batch
    trainable_vars = encoder.trainable_variables + decoder.trainable_va
riables
    grad = tape.gradient (loss, trainable_vars)
    optimizer.apply_gradients(zip(grad, trainable_vars))

    return loss, avg_loss

@tf.function

def test_step(img_tensor, target):
    loss = 0
    hidden = decoder.init_state(batch_size = target.shape[0])
    dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * targ
et.shape[0], 1)
    with tf.GradientTape() as tape:
      encoder_op = encoder(img_tensor)
      for r in range(1, target.shape[1]) :
        predictions, hidden, _ = decoder(dec_input, encoder_op, hidden)
        loss = loss + loss_function(target[:, r], predictions)
        dec_input = tf.expand_dims(target[: , r], 1)
    avg_loss = (loss/ int(target.shape[1])) #avg loss per batch
    trainable_vars = encoder.trainable_variables + decoder.trainable_va
riables
    grad = tape.gradient (loss, trainable_vars)
    optimizer.apply_gradients(zip(grad, trainable_vars))

    return loss, avg_loss

def test_loss_cal(test_dataset):
    total_loss = 0
    for (batch, (img_tensor, target)) in enumerate(test_dataset) :
      batch_loss, t_loss = test_step(img_tensor, target)
      total_loss = total_loss + t_loss
```

```
        avg_test_loss = total_loss/ test_num_steps

    return avg_test_loss

loss_plot = []
test_loss_plot = []
EPOCHS = 30
best_test_loss=100

for epoch in tqdm(range(0, EPOCHS)):
    start = time.time()
    total_loss = 0
    for (batch, (img_tensor, target)) in enumerate(train_dataset):
        batch_loss, t_loss = train_step(img_tensor, target)
        total_loss += t_loss
        avg_train_loss=total_loss / train_num_steps
    loss_plot.append(avg_train_loss)
    test_loss = test_loss_cal(test_dataset)
    test_loss_plot.append(test_loss)
    print ('For epoch: {}, the train loss is {:.3f}, & test loss is {:.
3f}'.format(epoch+1,avg_train_loss,test_loss))
    print ('Time taken for 1 epoch {} sec\n'.format(time.time() -
 start))
    if test_loss < best_test_loss:
        print('Test loss has been reduced from %.3f to %.3f' % (best_te
st_loss, test_loss))
        best_test_loss = test_loss
        ckpt_manager.save()

def evaluate(image):
    attention_plot = np.zeros((max_length, attention_feature_shape))

    hidden = decoder.init_state(batch_size=1)

    temp_input = tf.expand_dims(load_images(image)[0], 0)
    img_tensor_val = image_features_extract_model(temp_input)
    img_tensor_val = tf.reshape(img_tensor_val, (img_tensor_val.shape[0
], -1, img_tensor_val.shape[3]))

    features = encoder (img_tensor_val)

    dec_input = tf.expand_dims([tokenizer.word_index['<start>']], 0)
    result = []

    for i in range(max_length):
        predictions, hidden, attention_weights = decoder(dec_input, fea
tures, hidden)
```

```
        attention_plot[i] = tf.reshape(attention_weights, (-
1, )).numpy()

        predicted_id = tf.argmax(predictions[0]).numpy()
        result.append (tokenizer.index_word[predicted_id])

        if tokenizer.index_word[predicted_id] == '<end>':
            return result, attention_plot,predictions

        dec_input = tf.expand_dims([predicted_id], 0)

    attention_plot = attention_plot[:len(result), :]
    return result, attention_plot,predictions

def plot_attention_map (caption, weights, image) :

  fig = plt.figure(figsize = (20, 20))
  temp_img = np.array(Image.open(image))

  cap_len = len(caption)
  for cap in range(cap_len) :
    weights_img = np.reshape(weights[cap], (8,8))
    wweights_img = np.array(Image.fromarray(weights_img).resize((224,22
4), Image.LANCZOS))

    ax = fig.add_subplot(cap_len//2, cap_len//2, cap+1)
    ax.set_title(caption[cap], fontsize = 14, color = 'red')

    img = ax.imshow(temp_img)

    ax.imshow(weights_img, cmap='gist_heat', alpha=0.6, extent=img.get_
extent())
    ax.axis('off')
  plt.subplots_adjust(hspace=0.2, wspace=0.2)
  plt.show()

# captions on the validation set
rid = np.random.randint(0, len(image_test))
test_image = image_test[rid]
real_caption = ' '.join([tokenizer.index_word[i] for i in cap_test_data
[rid] if i not in [0]])
result, attention_plot, pred_test = evaluate(test_image)

print('Real Caption:', real_caption)
print('Prediction Caption:', ' '.join(result))
plot_attention_map(result, attention_plot, test_image)
```