Voice Assistant for Image

Under the Guidance of

Prof. Vaibhav Dhore

Submitted By

Raj Chheda(161070046)
Pranay Samrit (161070048)
Dhiraj Bhoi(161070055)
Abhishek Toshniwal(161070065)

Department of Computer Engineering

Veermata Jijabai Technological Institute, Mumbai - 400019

(Autonomous Institute Affiliated to University of Mumbai)
2019 - 2020

CERTIFICATE

This is to certify that Mr. Raj Chheda(161070046),Mr. Pranay Samrit(161070048), Mr. Dhiraj Bhoi (161070055) and Mr Abhishek Toshniwal(161070065), students of B.Tech (Computer Engineering), VJTI , Matunga ,Mumbai /400019 have successfully completed the final year project under my supervision and studied some of the concepts on deep learning including RNN, CNN and LSTM and implemented the prototype model on Voice Assistant for Image in Department of Computer Engineering and IT ,VJTI, Matunga ,Mumbai 19 (MS) from Dated 15-July-2019 to 5-May-2020.

(Prof. V.D.Dhore)
Computer Engineering,VJTI.

(Dr. M. M. Chandane)
Head of Computer Engineering,VJTI.

DECLARATION OF STUDENT

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

Raj Chheda	Pranay Samrit	Dhiraj Bhoi	Abhishek Toshniwal
(161070046)	(161070048)	(161070055)	(161070065)

APPROVAL SHEET

The report ”Voice Assistant for image” submitted by,

1. Raj Chheda (161070046)
2. Pranay Samrit (161070048)
3. Dhiraj Bhoi (161070055)
4. Abhishek Toshniwal (161070065)

is found to be satisfactory and is approved for the Degree of Bachelor of Technology(Computer Engineering)

Internal Examiner 1	Internal Examiner 2	External Examiner

Abstract

Inspired by recent work in machine translation and object detection, we introduce CNN-LSTM model that automatically learns to describe the content of images. Existing approaches are either top-down, which start from a gist of an image and convert it into words, or bottom-up, which come up with words describing various aspects of an image and then combine them. Here, through a hybrid model our algorithm learns to selectively attend to semantic concept proposals and fuse them into hidden states and outputs of recurrent neural networks. We describe how we can train this model in a deterministic manner using standard backpropagation techniques and stochastically by maximizing a variational lower bound. We also show through visualization how the model is able to automatically learn to fix its gaze on salient objects while generating the corresponding words in the output sequence.We suggest improvements over pre-existing models and validate the use of hybrid model with state-of-art performance on two benchmark datasets: Flickr8k and MS COCO.

Contents

1 Introduction

Every day, we encounter a large number of images from various sources such as the internet, news articles, document diagrams and advertisements. These sources contain images that viewers would have to interpret themselves. Most images do not have a description, but the human can largely understand them without their detailed captions. However, a machine needs to interpret the image if humans need automatic image captions from it.

For the image captioning task, humans can easily understand the image content and express it in the form of natural language sentences according to specific needs; however, for computers, it requires the integrated use of image processing, computer vision, natural language processing and other major areas of research results. The challenge of image captioning is to design a model that can fully use image information to generate more human-like rich image descriptions. The meaningful description generation process of high level image semantics requires not only the understanding of objects or scene recognition in the image, but also the ability to analyse their states,understand the relationship among them and generate a semantically and syntactically correct sentence. It is currently unclear how the brain understands an image and organizes the visual information into a caption. Image captioning involves a deep understanding of the salient parts of the whole world.

1.1 Background

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 systems. For our course project, we design an image captioning system using convolutional neural networks(CNN) and recurrent neural networks (RNN). 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. RNN 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. RNN's tend to be computationally expensive to train and evaluate, so in practice memory is limited to just a few elements. In this work, we develop a system which extracts features from images using a hybrid convolutional neural network (CNN), and generates captions with an RNN

1.2 Current Scenario

Machine learning is a new age and future of computer science. Every company wants to do machine learning on a bigger scale and for less cost. Cloud service providers will continue to compete to drive down the costs and increase the capacity of machine learning systems. We've seen Google's cloud services grow from storage to include a suite of machine learning tools across language, speech and images. They have gone so far as to build custom hardwarethe tensor processing unitfor helping users train their own machine learning systems quickly. Amazon's AWS and Azure have similar offerings.And because of bigger, faster, cheaper machine learning, which is more accurate with less data, we will see the number of applications and use cases of machine learning continue to rise across all sectors.Seeing the trends in new age knowledge discovery from data we could deduce that a lot of information can be obtained from visuals and converting this data format into knowledge. Image processing has played and will continue to play an important role in science and industry.

1.3 Motivation

As mentioned earlier, a lot of study has gone into studying the field of image processing. We then stumbled upon a problem as to whom can this research benefit the most? Who would be our target audience?
Someone who is unable to process this data or someone who could be greatly benefited by this research.One answer was, blind people. But this research could also be used for cross-language/cross cultural social activities e.g: Tourists visiting the place unable to understand some local goods, machines, artifacts and their daily/normal name and usage.
Scanning this visual may guide them in their local language. Our other motivation to undergo a project in machine learning which would enhance our knowledge and exploration in this field.

1.4 Problem Statement

The Problem: Blind people have problems going out as they have to be dependent on other people to guide them through. They have to use their other senses to determine whether there is an obstacle around. Another problem is that tourists have difficulty understanding local languages and they can get lost while searching for some place.

The Proposal: We propose to develop a digital assistant for the visually impaired people as it can identify the nearby object and convert it into an audio.

2 Literature Review

In this section, state-of-the-art techniques used in current object detection and recognition methods are reviewed.Deep learning based methods[10] have become the mainstream for text detection.The relevant models and the model evaluation and performance strategies are discussed.

The very first image detection methods were based on traditional machine learning algorithms such as Local Binary Patterns (LBP)[14],Scale-Invariant Feature Transform (SIFT)[6], the Histogram of Oriented Gradients (HOG)[12],and a combination of such features are widely used.The output of these techniques is passed to SVM(Support Vector Machine) which classify the objects. However, we cannot use these techniques as these techniques are task-specific, so may not give the desired output and also our project uses real-life data such as images which may have different semantic interpretations. Hence we have to use deep-learning methods to classify the object.

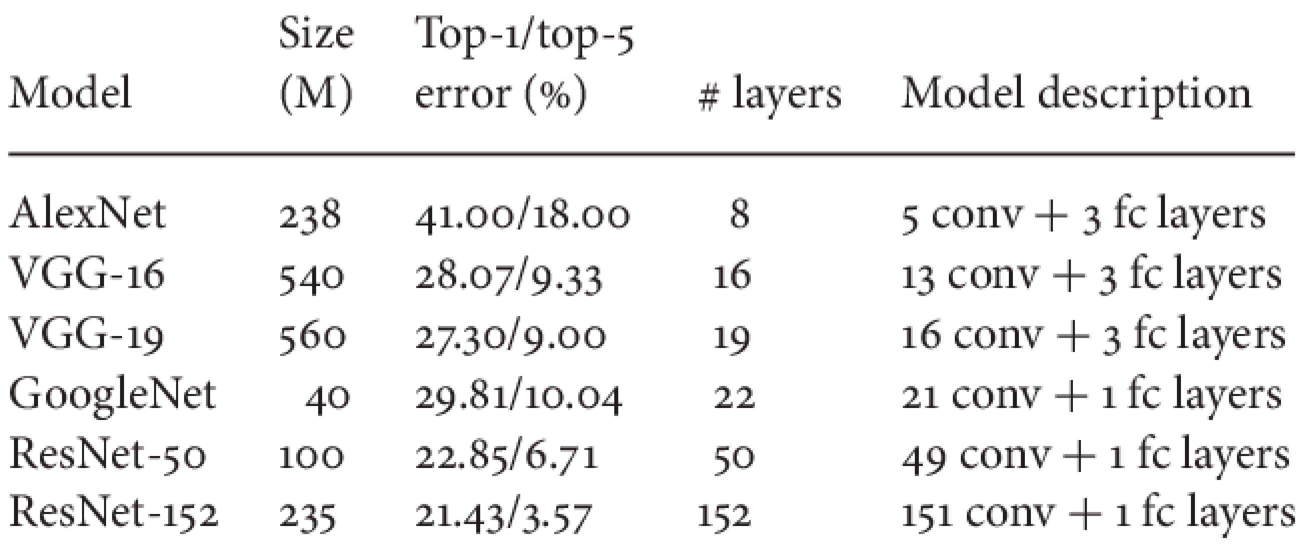
Image captioning algorithms are typically divided into three categories.The first category tackles this problem using the **retrieval-based methods**, which first retrieves the closest matching images, and then transfers their descriptions as the captions of the query images . These methods can produce grammatically correct sentences but cannot adjust the captions according to the new image. The advantage of these methods is that the generated captions are more natural than the generations by template-based methods. However, because they directly rely on retrieval results among training data, there is little flexibility for them to add or remove words based on the content of an image.

The second category typically uses **template-based methods** to generate descriptions with predefined syntactic rules and split sentences into several parts . These methods first take advantage of several classifiers to recognize the objects, as well as their attributes and relationships in an image, and then use a rigid sentence template to form a complete sentence. Though it can generate a new sentence, these methods either cannot express the visual context correctly or generate flexible and meaningful sentences.

With the extensive application of deep learning, most recent works fall into the third category called **neural network-based methods**.Inspired by machine learnings encoder-decoder architecture , recent years most image captioning methods employ a Convolutional Neural Network (CNN) as the encoder and a Recurrent Neural. Network (RNN) as the decoder, especially Long Short-Term Memory (LSTM) to generate captions, with the objective to maximize the likelihood of a sentence given the visual features of an image.

CNN model for extracting image features use models such as AlexNet[9], VGG[5], GoogleNet[7] and ResNet[4]. AlexNet is a model consisting of eight layers-five convolutional and three fully connected layers.The advantages of AlexNet are that it uses multiple GPU's cutting down on training time, it uses ReLu instead of Tanh which has a better accuracy rate and it also uses overlapped pooling which reduces the error faster. However, the main disadvantage of AlexNet model was that as it had 60 million parameters it had a overfitting problem. However, we can remove this problem using data augmentation and dropout. VGG is another model that is used in image classification. The main advantages is that it is simple to explain and easy to implement. However because of the large width of convolutional layers it becomes inefficient even on multiple GPU's. Also it has 180 million parameters which requires quite a lot computational time. Inception is a very deep neural network model and hence it's advantage is that it has a lot better performance than other models.This is because it uses multiple filters to get multiple features which results in better performance. It is much faster than VGG because it removes the fully connected layers at the end with a simple global average pooling which averages out the channel values across the 2D feature map, after the last convolutional layer. This results in massively reducing the number of parameters as it has only 7 million parameters compared to other models. It does not have any main disadvantage but, this model is quite difficult to understand as it has 42 layers. Lastly the ResNet is another model which is used for image classification. It has a great advantage that it does not have a vanishing gradient problem. The core idea of ResNet is introducing a so-called identity shortcut connection that skips one or more layers which bypasses the vanishing gradient problem. This is the reason the ResNet is computationally more efficient. It just has a disadvantage that it does not have a better accuracy than Inception models.

LSTM[13] stands for Long Short-term memory and it has three gates input, output and forget gate. The advantages of LSTM is that because of error backpropagation it can result into better captions and it can remember short term memory by forgetting some words.The disadvantage of LSTM is that it requires longer to train and requires more memory to train.GRU[3] stands for Gated Recurrent Unit has two gates reset and update. The main advantage of GRU is that it has less parameters to train and it trains faster, but it has a worse accuracy than LSTM as it does not have a good memory.



Model	Size (M)	Top-1/top-5 error (%)	# layers	Model description
AlexNet	238	41.00/18.00	8	5 conv + 3 fc layers
VGG-16	540	28.07/9.33	16	13 conv + 3 fc layers
VGG-19	560	27.30/9.00	19	16 conv + 3 fc layers
GoogleNet	40	29.81/10.04	22	21 conv + 1 fc layers
ResNet-50	100	22.85/6.71	50	49 conv + 1 fc layers
ResNet-152	235	21.43/3.57	152	151 conv + 1 fc layers

Figure 1: Different CNN models compared

Model Evaluation and Performance

Evaluation of the generated caption from the network is an important as it tells how effectively the network is generating caption.The evaluation is carried out at the time of training with the generated caption and the training caption.The evaluation can be done in various way:-

BLEU[11]

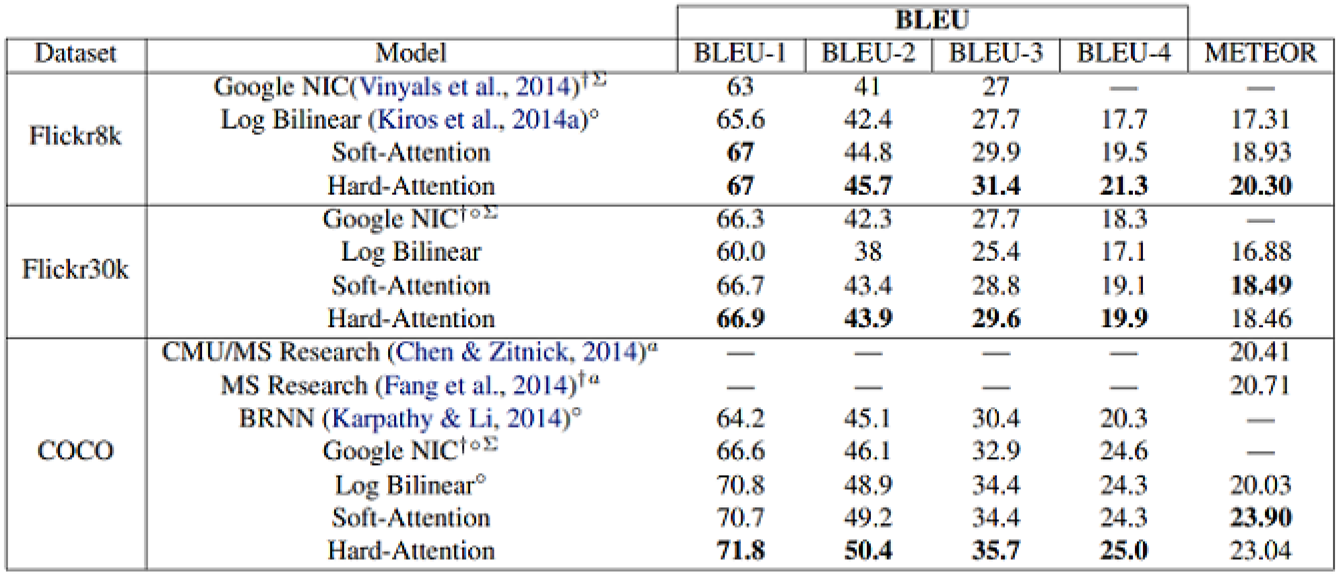
BLEU was one of the first metrics to report high correlation with human judgments of quality. The metric calculates scores for individual segments, generally sentences-then averages these scores over the whole corpus for a final score. BLEU uses a modified form of precision to compare a candidate translation against multiple reference translations.

NIST[8]
The NIST metric is based on the BLEU metric, but with some alterations. Where BLEU simply calculates n-gram precision adding equal weight to each one, NIST also calculates how informative a particular n-gram is. That is to say when a correct n-gram is found, the rarer that n-gram is, the more weight it is given.

Word error rate[15]
The Word error rate (WER) is a metric based on the Levenshtein distance, where the Levenshtein distance works at the character level, WER works at the word level. The metric is based on the calculation of the number of words that differ between a piece of machine translated text and a reference translation.

METEOR[1]
The metric is based on the weighted harmonic mean of unigram precision and unigram recall. METEOR also includes some other features not found in other metrics, such as synonym matching, where instead of matching only on the exact word form, the metric also matches on synonyms.

LEPOR[2]
A new MT evaluation metric LEPOR was proposed as the combination of many evaluation factors including existing ones (precision, recall) and modified ones (sentence-length penalty and n-gram based word order penalty).



Dataset	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
Flickr8k	Google NIC(Vinyals et al., 2014)$^{\dagger\Sigma}$	63	41	27	—	—
	Log Bilinear (Kiros et al., 2014a)$^{\circ}$	65.6	42.4	27.7	17.7	17.31
	Soft-Attention	**67**	44.8	29.9	19.5	18.93
	Hard-Attention	**67**	**45.7**	**31.4**	**21.3**	**20.30**
Flickr30k	Google NIC$^{\dagger\circ\Sigma}$	66.3	42.3	27.7	18.3	—
	Log Bilinear	60.0	38	25.4	17.1	16.88
	Soft-Attention	66.7	43.4	28.8	19.1	**18.49**
	Hard-Attention	**66.9**	**43.9**	**29.6**	**19.9**	18.46
COCO	CMU/MS Research (Chen & Zitnick, 2014)^{a}	—	—	—	—	20.41
	MS Research (Fang et al., 2014)$^{\dagger a}$	—	—	—	—	20.71
	BRNN (Karpathy & Li, 2014)$^{\circ}$	64.2	45.1	30.4	20.3	—
	Google NIC$^{\dagger\circ\Sigma}$	66.6	46.1	32.9	24.6	—
	Log Bilinear$^{\circ}$	70.8	48.9	34.4	24.3	20.03
	Soft-Attention	70.7	49.2	34.4	24.3	**23.90**
	Hard-Attention	**71.8**	**50.4**	**35.7**	**25.0**	23.04

Figure 2: Performance of models on BLEU-1,2,3,4/METEOR metrics

3 Proposed System

This is the proposed system which is shown in the diagram.

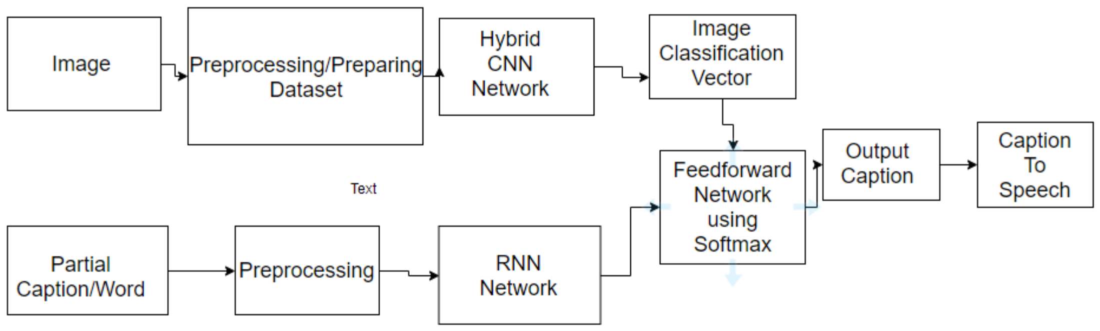


Figure 3: Proposed System : Block Diagram

3.1 Prepare the Dataset

Flickr and MSCOCO2014 dataset are used for this project, these 2 datasets are combined into one consistent dataset. In Flickr dataset captions are provided in text format while in MScoco2014 dataset captions are provided in JSON format. So, convert json format into text format by following below mentioned process:

1. As size of caption file in json format is in 60-70 mb, reading and understanding format of file becomes difficult task. So extensions on VS Code are used and python scripts are implemented to know the format of Json file.

2. There is lot of redundant unnecessary data in json file, so a python script is implemented to remove unnecessary json objects.

3. Python code is implemented to convert json file to csv format by choosing necessary json objects.
4. Formatting of csv file is done to make it consistent with Flickr caption file.
5. Convert Flickr file to csv format and merge it with MSCOCO csv file.
6. Caption file is divided into 80:10:10 ratio to form train, dev and test files are created.
7. This merged csv file is finally converted to text file.

Dataset:

Train set : 337682 Images

Development set : 42400 Images

Test set : 42031 Images

Here every Image has 5 captions

3.2 Preprocessing

Caption Preprocessing: We will clean the text in the following ways in order to reduce the size of the vocabulary of words we will need to work with:

1. Convert all words to lowercase.
2. Remove all punctuation.
3. Remove all words that are one character or less in length (e.g. a).
4. Remove all words with numbers in them.

Another thing we have to do is load the data onto the model. The steps of loading the data are:

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

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

3. Each description will be split into words. The model will be provided one word and the photo and generate the next word. Then the first two words of the description will be provided to the model as input with the image to generate the next word. This is how the model will be trained.

4. Later, when the model is used to generate descriptions, the generated words will be concatenated and recursively provided as input to generate a caption for an image.

5. There are two input arrays to the model: one for photo features and one for the encoded text. There is one output for the model which is the encoded next word in the text sequence.

6. The input text is encoded as integers, which will be fed to a word embedding layer. The photo features will be fed directly to another part of the model. The model will output a prediction, which will be a probability distribution over all words in the vocabulary.

7. The output data will therefore be a one-hot encoded version of each word, representing an idealized probability distribution with 0 values at all word

positions except the actual word position, which has a value of 1.

Image Preprocessing: We have to prepare the images in a format so that the features of the image complies with both the model. The process of image preprocessing is:

1. Image is converted into preferred size of the model using keras (e.g. 3 channel 224 x 224 pixel image).

2. We can load the VGG model in Keras using the VGG class. We will remove the last layer from the loaded model, as this is the model used to predict a classification for a photo. We are not interested in classifying images, but we are interested in the internal representation of the photo right before a classification is made. These are the features that the model has extracted from the photo.

3. The image features are a 1-dimensional 4,096 element vector.

4. Pre-compute the photo features using the pre-trained model and save them to a pickle file. We can then load these features later and feed them into our model as the interpretation of a given photo in the dataset.This is an optimization that will make training our models faster and consume less memory.

3.3 Creating the Hybrid Model

This hybrid model is made by combining VGG16 and ResNet50. ResNet50 is appended on the top of VGG16 and output is 1000 class classification vector.

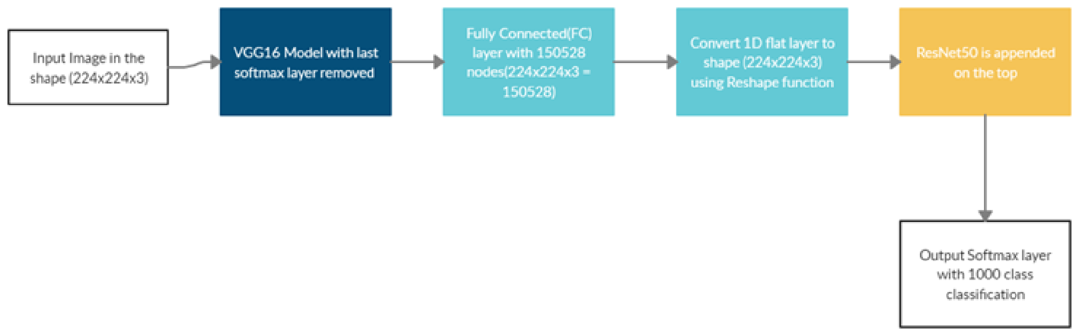

Input Image in the shape (224x224x3)
VGG16 Model with last softmax layer removed
Fully Connected(FC) layer with 150528 nodes(224x224x3 = 150528)
Convert 1D flat layer to shape (224x224x3) using Reshape function
ResNet50 is appended on the top
Output Softmax layer with 1000 class classification

Figure 4: Hybrid Model:Block diagram. Light blue coloured blocks depict we are joining two network.

3.4 Structure of Model

We have mainly three main structures in our model. The first is a Photo Feature Extractor which is a 16-layer VGG model followed by the ResNet model. We have pre-processed the photos with the VGG model (without the output layer) and will use the extracted features predicted by this model as input to the next model. The output of ResNet acts as a input to the next structure. The Photo Feature Extractor model expects input photo features to be a vector of 4,096 elements. These are processed by a Dense layer to produce a 256 element representation of the photo.

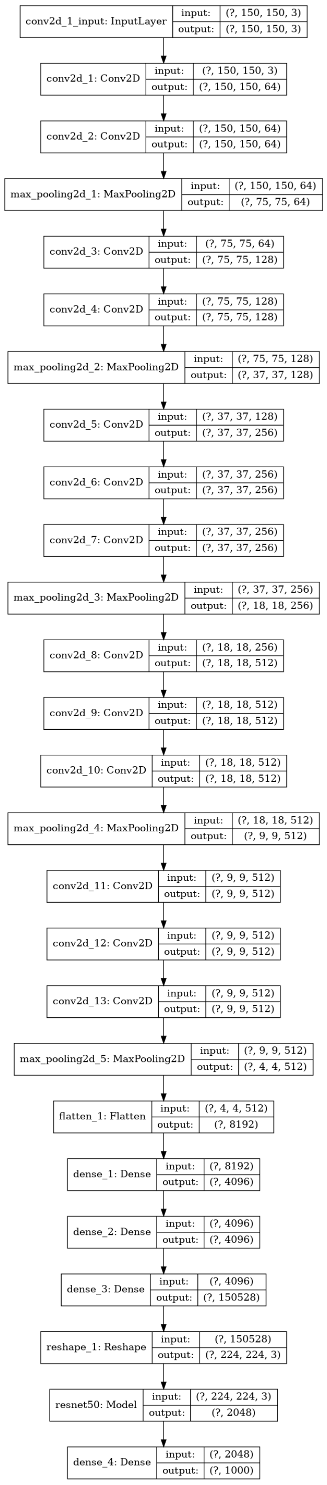
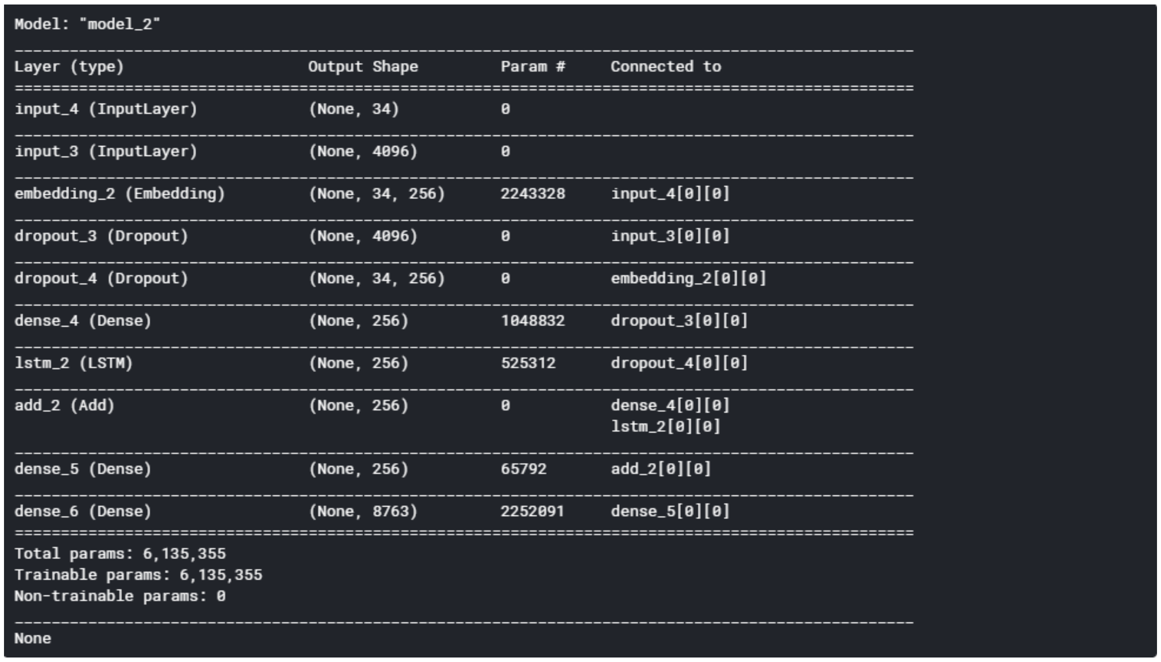

conv2d_1_input: InputLayer input: (?, 150, 150, 3) output: (?, 150, 150, 3)
conv2d_1: Conv2D input: (?, 150, 150, 3) output: (?, 150, 150, 64)
conv2d_2: Conv2D input: (?, 150, 150, 64) output: (?, 150, 150, 64)
max_pooling2d_1: MaxPooling2D input: (?, 150, 150, 64) output: (?, 75, 75, 64)
conv2d_3: Conv2D input: (?, 75, 75, 64) output: (?, 75, 75, 128)
conv2d_4: Conv2D input: (?, 75, 75, 128) output: (?, 75, 75, 128)
max_pooling2d_2: MaxPooling2D input: (?, 75, 75, 128) output: (?, 37, 37, 128)
conv2d_5: Conv2D input: (?, 37, 37, 128) output: (?, 37, 37, 256)
conv2d_6: Conv2D input: (?, 37, 37, 256) output: (?, 37, 37, 256)
conv2d_7: Conv2D input: (?, 37, 37, 256) output: (?, 37, 37, 256)
max_pooling2d_3: MaxPooling2D input: (?, 37, 37, 256) output: (?, 18, 18, 256)
conv2d_8: Conv2D input: (?, 18, 18, 256) output: (?, 18, 18, 512)
conv2d_9: Conv2D input: (?, 18, 18, 512) output: (?, 18, 18, 512)
conv2d_10: Conv2D input: (?, 18, 18, 512) output: (?, 18, 18, 512)
max_pooling2d_4: MaxPooling2D input: (?, 18, 18, 512) output: (?, 9, 9, 512)
conv2d_11: Conv2D input: (?, 9, 9, 512) output: (?, 9, 9, 512)
conv2d_12: Conv2D input: (?, 9, 9, 512) output: (?, 9, 9, 512)
conv2d_13: Conv2D input: (?, 9, 9, 512) output: (?, 9, 9, 512)
max_pooling2d_5: MaxPooling2D input: (?, 9, 9, 512) output: (?, 4, 4, 512)
flatten_1: Flatten input: (?, 4, 4, 512) output: (?, 8192)
dense_1: Dense input: (?, 8192) output: (?, 4096)
dense_2: Dense input: (?, 4096) output: (?, 4096)
dense_3: Dense input: (?, 4096) output: (?, 150528)
reshape_1: Reshape input: (?, 150528) output: (?, 224, 224, 3)
resnet50: Model input: (?, 224, 224, 3) output: (?, 2048)
dense_4: Dense input: (?, 2048) output: (?, 1000)

Figure 5: Detailed layers of the Hybrid Model



```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            (None, 34)           0
__________________________________________________________________________________________________
input_3 (InputLayer)            (None, 4096)         0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 34, 256)      2243328     input_4[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 4096)         0           input_3[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 34, 256)      0           embedding_2[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 256)          1048832     dropout_3[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   (None, 256)          525312      dropout_4[0][0]
__________________________________________________________________________________________________
add_2 (Add)                     (None, 256)          0           dense_4[0][0]
                                                                 lstm_2[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 256)          65792       add_2[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 8763)         2252091     dense_5[0][0]
==================================================================================================
Total params: 6,135,355
Trainable params: 6,135,355
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

Figure 6: Detailed layers of the LSTM model

Next, the Sequence Processor is a word embedding layer for handling the text input, followed by a Long Short-Term Memory (LSTM) recurrent neural network layer.The Sequence Processor model expects input sequences with a pre-defined length (34 words) which are fed into an Embedding layer that uses a mask to ignore padded values. This is followed by an LSTM layer with 256 memory units.Both the input models produce a 256 element vector. Further, both input models use regularization in the form of 50 percent dropout. This is to reduce overfitting the training dataset, as this model configuration learns very fast.

Finally, in decoder both the feature extractor and sequence processor output(which are a fixed-length vector) are merged together and processed by a Dense layer to make a final prediction.The Decoder model merges the vectors from both input models using an addition operation. This is then fed to a Dense 256 neuron layer and then to a final output Dense layer that makes a softmax prediction over the entire output vocabulary for the next word in the sequence.

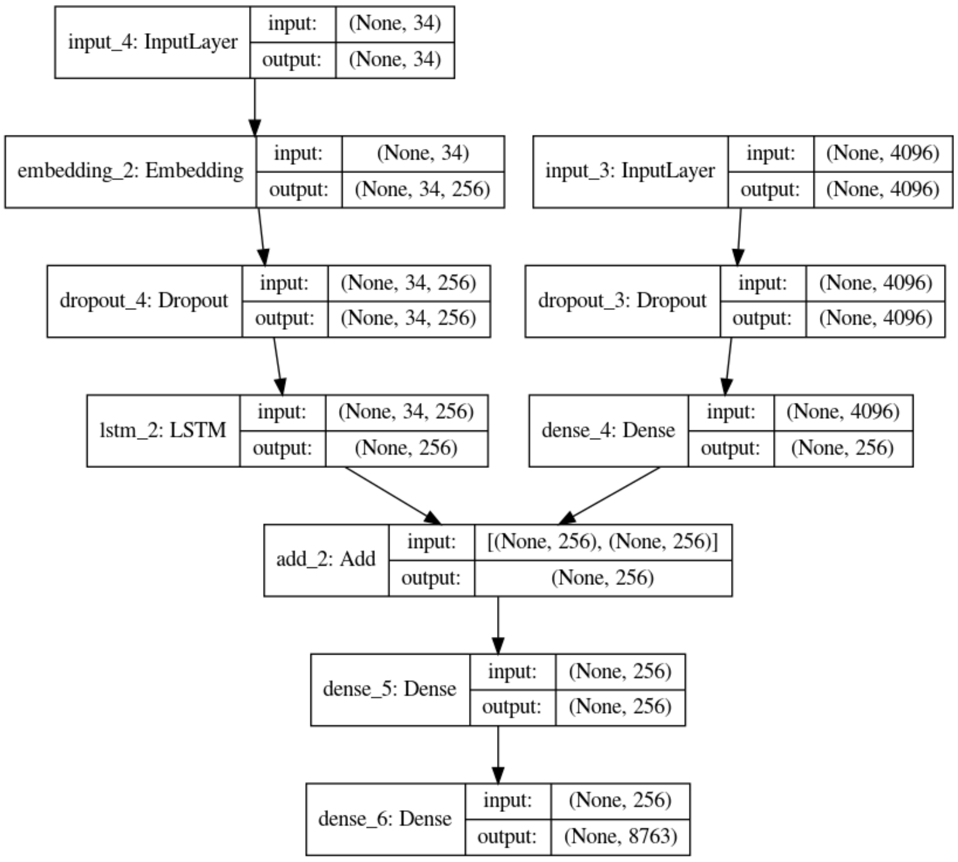


Figure 7: Detailed layers and working of Project Model

4 Analysis and Design

4.1 Hardware and Software Requirements

1. NVIDIA DGX GPU
2. System (PC/Laptop)
3. Kaggle/Colab for Python code implementation of CNN-LSTM model
4. React.js for front end of API
5. Node.js for Backend of API
6. Apache as a server
7. Python Libraries (Keras,Tensorflow,Pillow,matplotlib)
8. gTTs for text to speech
9. Dataset: Flick30k/ MSCOCO
10. Docker : A container to use Python libraries with DGX GPU

4.2 Feasibility Study

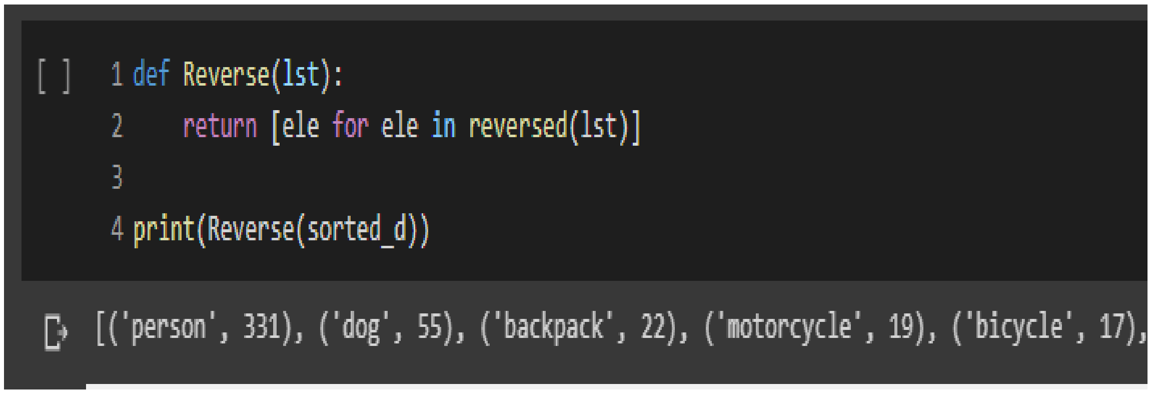
- Algorithms and libraries present make it possible to implement this project.
- This Project demands high computing power for creating and evaluating models. This project is not feasible to train on a local machine, but can be trained on external servers with high computational power.
- Success of this project depends on number of images trained, so it is practically not possible to train model with each and every type of image.

- As this product can successfully detect general types of images, areas where this can be used is limited.
- This application will be more successful and feasible if end stakeholder(users) help grow this application by providing input and training model.
- Visually impaired people can use this, but that will require development of new hardware devices.
- Economically feasibility of product will depend on the cost of hardware device.
- Market place for this application will be limited considering small user base and limitations of domain.

4.3 Scope of Dataset and Project

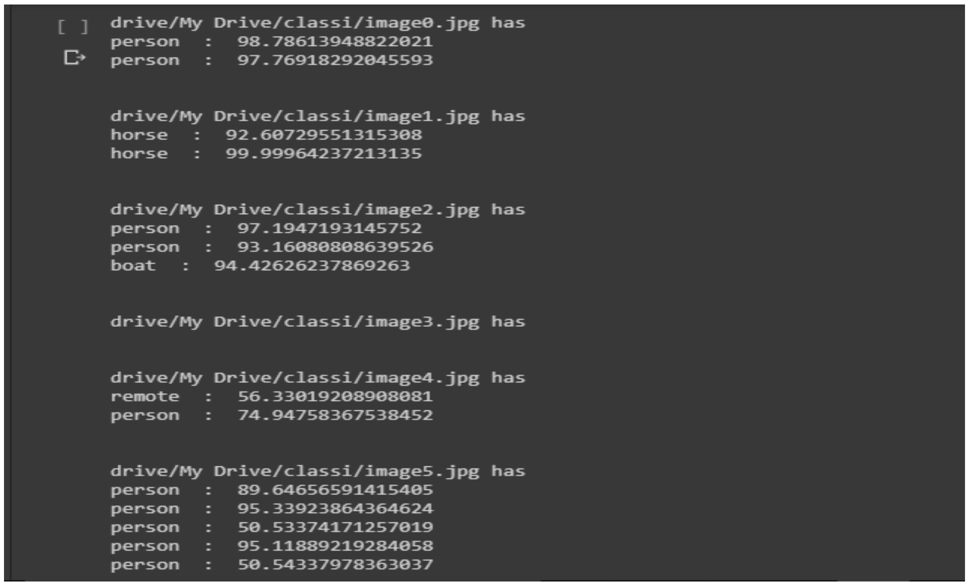
4.3.1 Scope of Images

Random sampling was used to get a small dataset to carry out a study to determine what kind of images and the objects in the image form the majority that will affect the result and accuracy of the model. Used Object Detection from Imageai.Detection to detect the objects along with their probability. Total number of objects for each different class present in the random sample was articulated and presented in analytical form.
Some of the most occurring objects detected were person, dog, motorcycle, bicycle, boat, etc in that order. Which clearly states that the model will give a better BLUE score if the images have above objects in them.



```
[ ] def Reverse(lst):
        return [ele for ele in reversed(lst)]

    print(Reverse(sorted_d))

[('person', 331), ('dog', 55), ('backpack', 22), ('motorcycle', 19), ('bicycle', 17),
```

Figure 8: Scope of the dataset

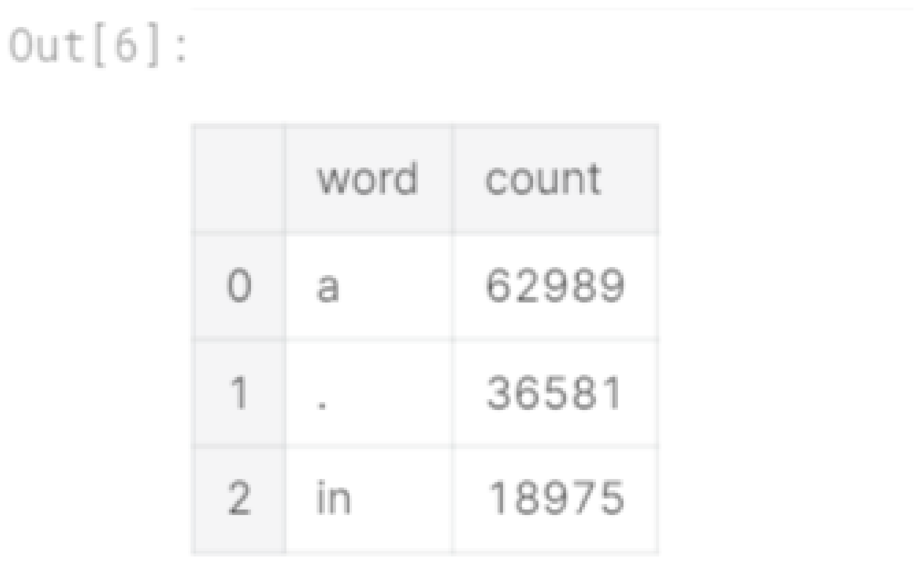


```
drive/My Drive/classi/image0.jpg has
person  :  98.78613948822021
person  :  97.76918292045593


drive/My Drive/classi/image1.jpg has
horse  :  92.60729551315308
horse  :  99.99964237213135


drive/My Drive/classi/image2.jpg has
person  :  97.1947193145752
person  :  93.16080808639526
boat  :  94.42626237869263


drive/My Drive/classi/image3.jpg has


drive/My Drive/classi/image4.jpg has
remote  :  56.33019208908081
person  :  74.94758367538452


drive/My Drive/classi/image5.jpg has
person  :  89.64656591415405
person  :  95.33923864364624
person  :  50.53374171257019
person  :  95.11889219284058
person  :  50.54337978363037
```

Figure 9: The figure represents occurence of various objects in the image with their probabilities.

4.3.2 Analysis of the captions

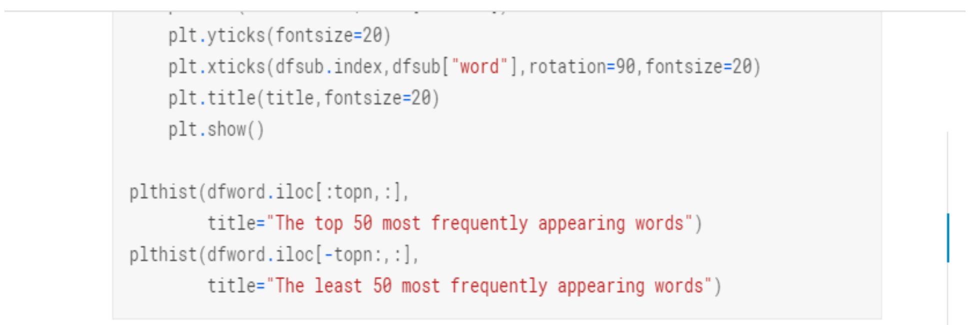
Used pandas and its DataFrame function to count the key-values of different words and the number of occurence of these various words of the vocabulary. Sorted in ascending order to visualize most occuring words of the vocbulary.



```
Out[6]:
```

	word	count
0	a	62989
1	.	36581
2	in	18975

Figure 10: The figure represents the top three characters in the caption dataset



```
    plt.yticks(fontsize=20)
    plt.xticks(dfsub.index,dfsub["word"],rotation=90,fontsize=20)
    plt.title(title,fontsize=20)
    plt.show()

plthist(dfword.iloc[:topn,:],
        title="The top 50 most frequently appearing words")
plthist(dfword.iloc[-topn:,:],
        title="The least 50 most frequently appearing words")
```

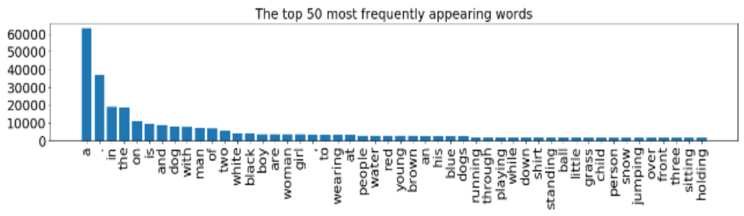


Figure 11: The figure represents the top 50 most frequently occurring words in the caption dataset.

4.4 UML diagrams

Use case:

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. The diagram shows the use cases or actions performed by different stakeholders.

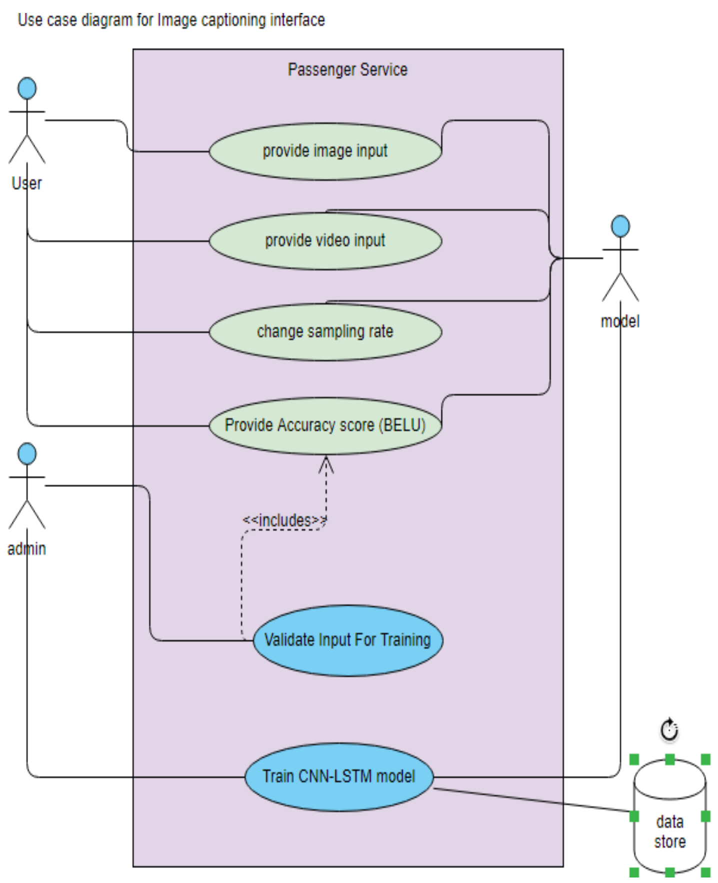


Figure 12: Use Case Diagram

Component Diagram:

In Unified Modeling Language, a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.The diagram depicts dependencies among various components of the system.

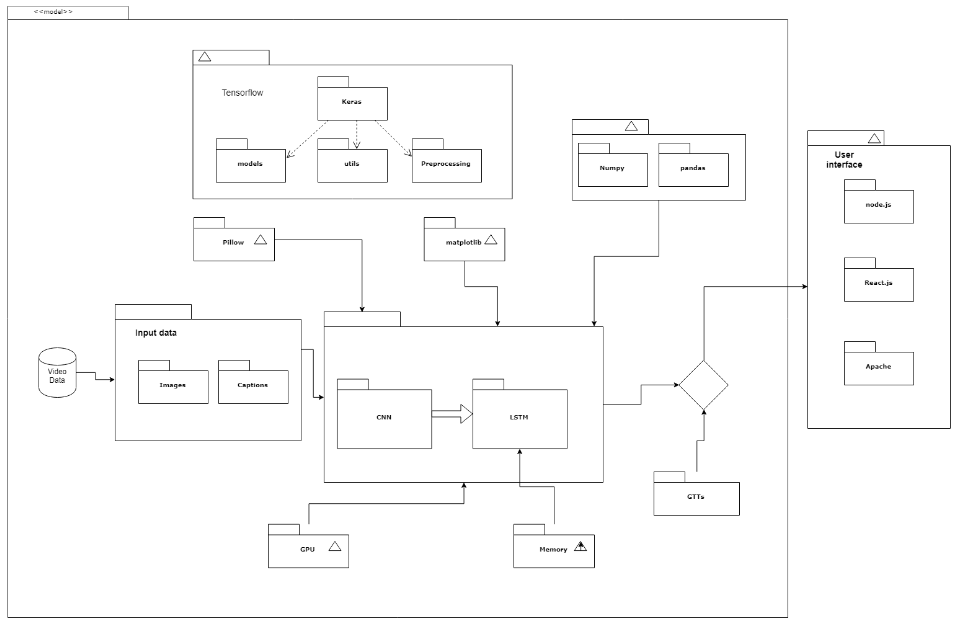


Figure 13: Component Diagram

Sequence Diagram:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. The diagram shows the sequence of activities among the user , interface, admin and model.

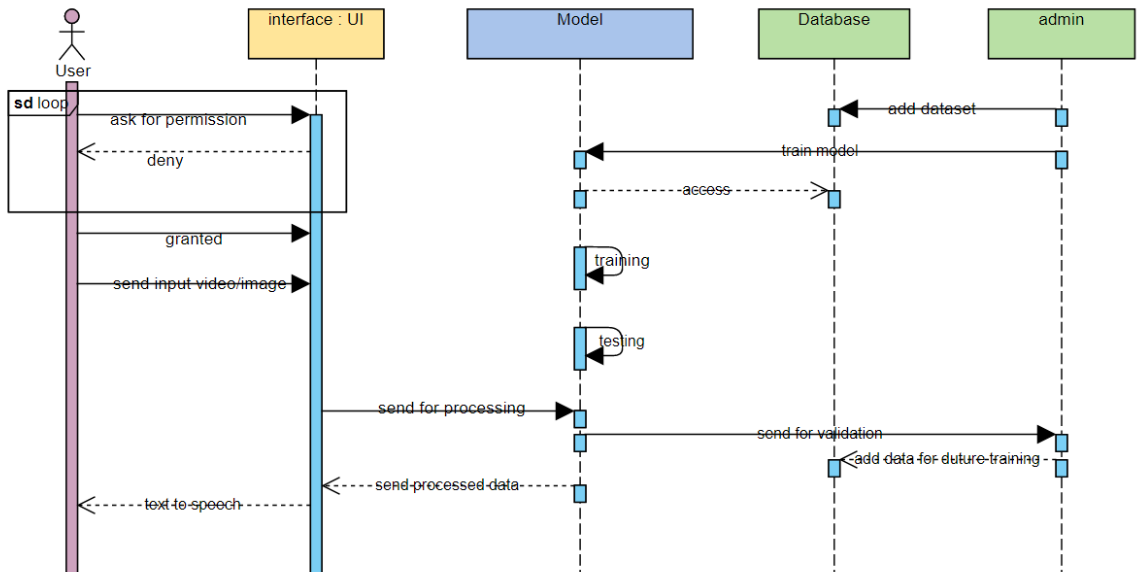


Figure 14: Sequence Diagram

Deployment Diagram:

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components exist, what software components run on each node, and how the different pieces are connected. The diagram shows relations among the required packages and modules.

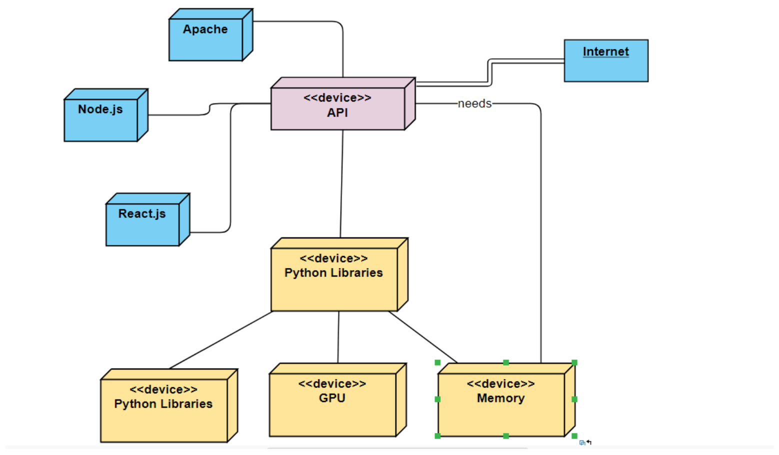


Figure 15: Deployment Diagram

System Architecture:

A system architecture diagram would be used to show the relationship between different components. Usually they are created for systems which include hardware and software and these are represented in the diagram to show the interaction between them. However, it can also be created for web applications.The diagram shows 3 tier architecture of our system.

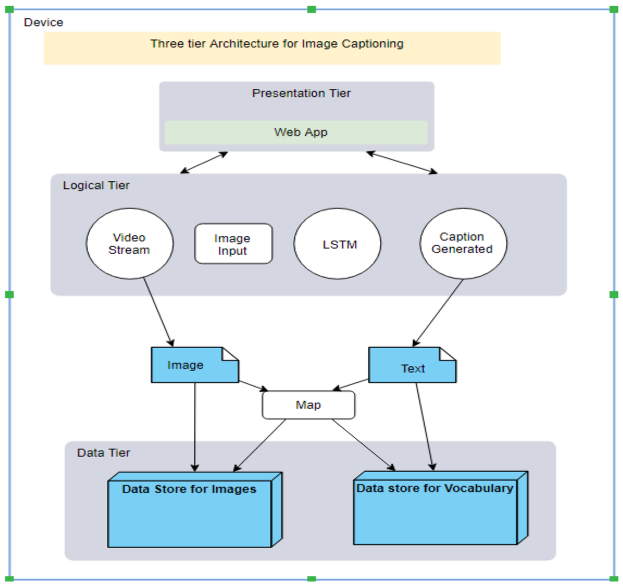


Figure 16: System Architecture

5 Implementation

5.1 Training the Model

Our model for Image Captioning has two parts. First part does the job of extracting features from image in 1024 sized vector. Second part uses output vector from first part to generate human language captions.

Feature Extractor

In order to get the feature vector of image, we use following steps:-

1. Input size of image to the model is 224x224 and use rgb filter
2. Load the vgg model and remove the last two layers which is used for classification purpose
3. Compute the feature vector for the image which gives the output of 1x4096 vector
4. Store these vector in pickle file for later use
5. Load and pass these vector into the our new model for further feature extraction

Storing the feature vector and then use it later is done only while training the model. For validation part images are not pass to vgg model. They are directly feed to our new model. This increases the accuracy , faster processing and consume less memory.

Caption Generator

We use LSTM to train our network to predict captions.

1. We need to encode each word into a fixed sized vector. So, create two Python Dictionaries word to index and index to word.

2. Represent every unique word in the vocabulary by an integer (index). Words that occur less than certain threshold value, they are discarded.

3. Image vector and partial caption goes as input iteratively to predict next word as the target word. For one image and one caption, network is trained for n times, where n is the number of words in the caption.

4. One image+caption is not a single data point but are multiple data points depending on the length of the caption.In every data point, its not just the image which goes as input to the system, but also, a partial caption which helps to predict the next word in the sequence.

5. Caption is padded with zeros to make every caption of same length.

5.2 Convert Text to Speech

There are several APIs available to convert text to speech in python. One of such APIs is the Google Text to Speech API commonly known as the gTTS API. gTTS is a very easy to use tool which converts the text entered, into audio which can be saved as a mp3 file. The gTTS API supports several languages including English, Hindi, Tamil, French, German and many more. The speech can be delivered in any one of the two available audio speeds, fast or slow. However, as of the latest update, it is not possible to change the voice of the generated audio.

5.3 Application

We have designed a web app which allows an image to be searched, uploaded and give a caption of the image which would be spelt out by the computer using a caption to speech library. After the image is uploaded, it is then processed and converted to the viable input of the image classification model, which generates the corresponding feature vector of the image. The feature vector along with <startseq> sequence are fed to the Captioning model for each iteration a new word is added to the sequence until it encounters <endseq> tag. The <startseq>, <endseq> are discarded and returned to the rendering page. Using gTTs library function, the text is converted to speech and simultaneously read aloud along with the text. Technologies used to develop application :

- Language:
  - Python
  - Javascript
- Libraries:
  - Keras
  - TensorFlow
  - Numpy
  - Pickle
  - gTTs
  - Werkzeug
  - jQuery

- FrameWork
  - Flask
- Technologies
  - HTML
  - CSS

5.4 Screenshots

Following screenshots illustrate the working of the system. These are the images captured when we input the image to the web application.

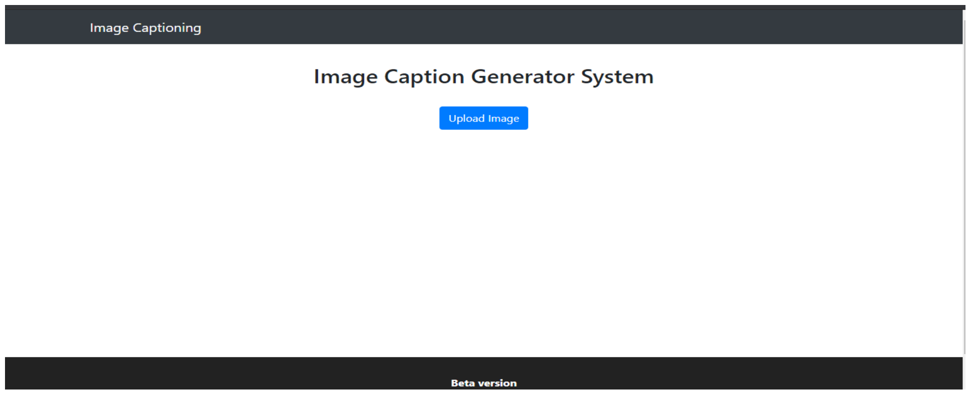


Figure 17: Screenshot of System: Index Page

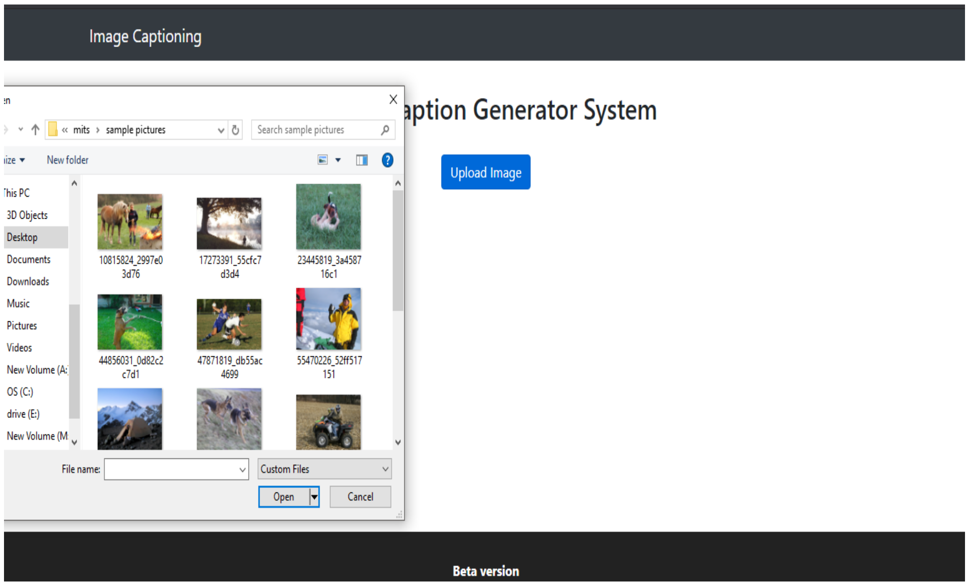
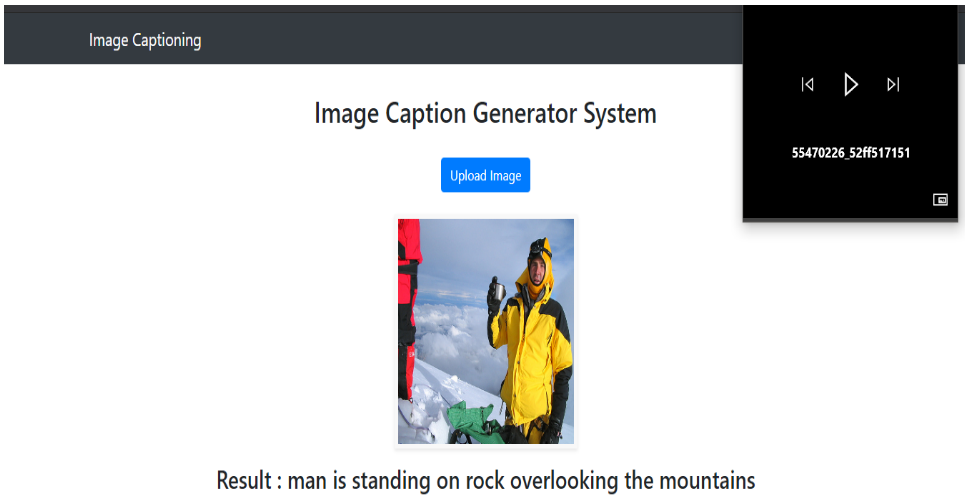


Figure 18: Screenshot of System: File upload



Result : man is standing on rock overlooking the mountains

Figure 19: Screenshot of System: Rendering of Predicted Text and system call to play the converted corresponding speech

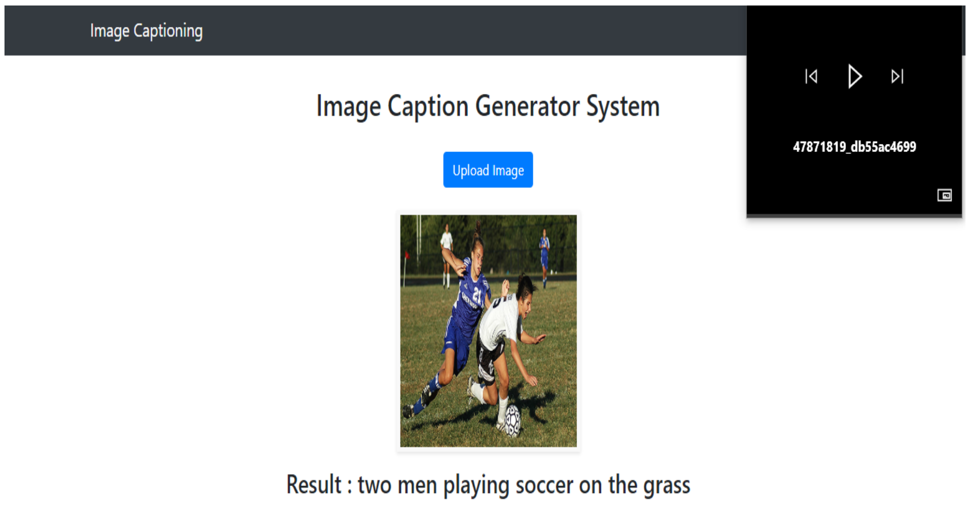


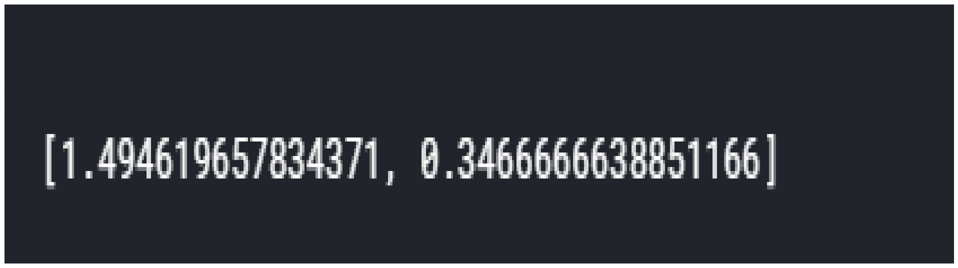
Image Caption Generator System

Result : two men playing soccer on the grass

Figure 20: Sample output of the Web Application

6 Results

Proposed Model is using Hybrid Model network for extracting features of image in Image vector. This hybrid model is trained and results we get by running this model on test set is as follows:



```
[1.494619657834371, 0.3466666638851166]
```

Figure 21: Hybrid model accuracy on test dataset - Loss and Accuracy

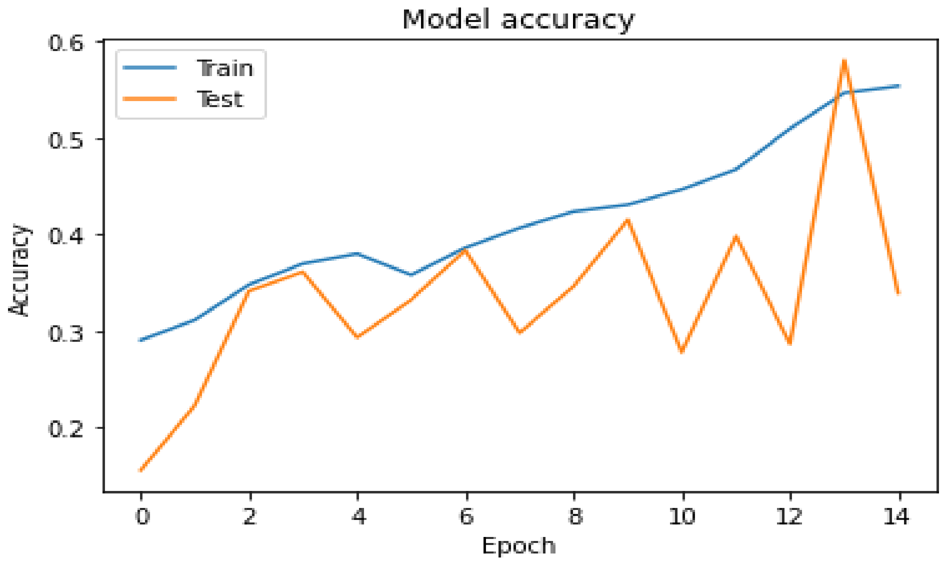


Figure 22: Plot of accuracy of train and test set data

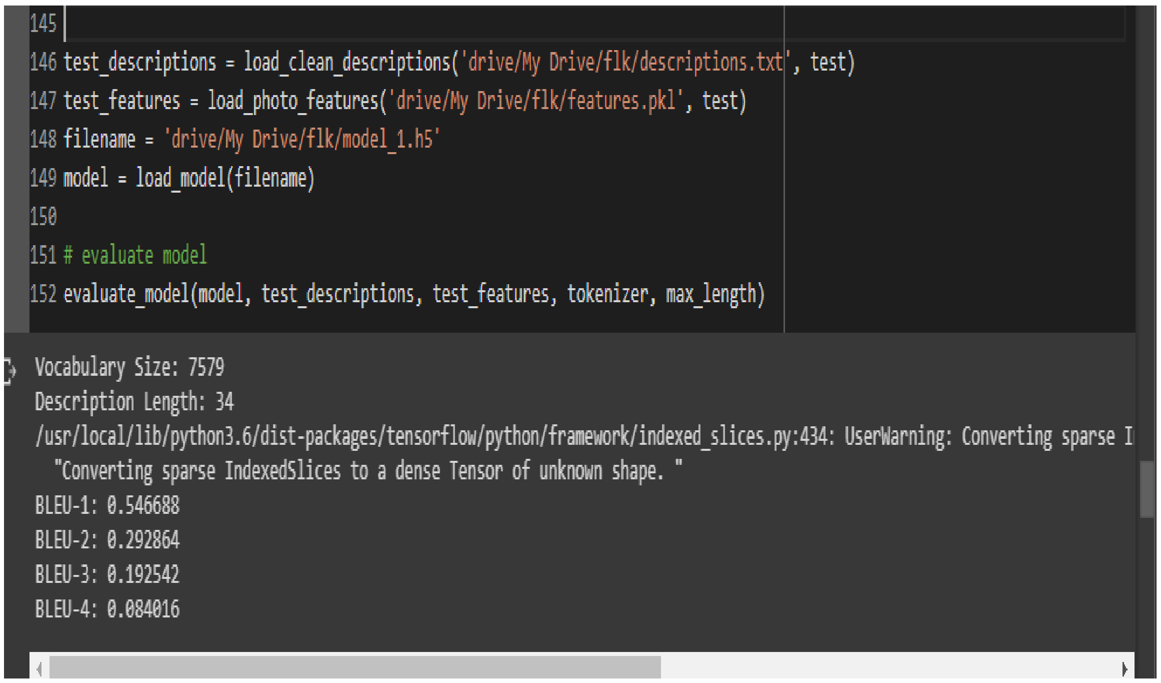
For the final Proposed model, BLEU scores obtained are as follows:

BLEU-1: 0.546688

BLEU-2: 0.292864

BLEU-3: 0.192542

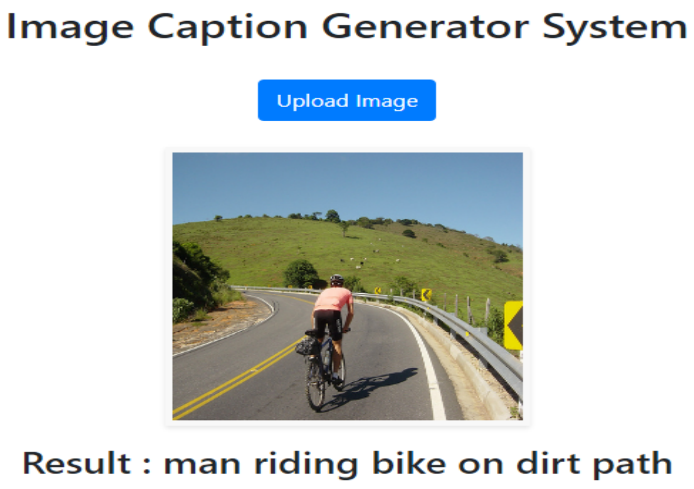
BLEU-4: 0.084016



```
test_descriptions = load_clean_descriptions('drive/My Drive/flk/descriptions.txt', test)
test_features = load_photo_features('drive/My Drive/flk/features.pkl', test)
filename = 'drive/My Drive/flk/model_1.h5'
model = load_model(filename)

# evaluate model
evaluate_model(model, test_descriptions, test_features, tokenizer, max_length)
```

```
Vocabulary Size: 7579
Description Length: 34
/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/indexed_slices.py:434: UserWarning: Converting sparse I
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
BLEU-1: 0.546688
BLEU-2: 0.292864
BLEU-3: 0.192542
BLEU-4: 0.084016
```

Figure 23: BLEU score results of the proposed model



Result : man riding bike on dirt path

Figure 24: True Caption: Man riding bike on curved road uphill.Predicted Caption: Man riding bike on dirt path. Therefore we get BLEU 1 : 0.571428.

7 Conclusion and Future Scope

We started our Final Year project by searching for an interesting topic which may help us learn new skills. We searched and studied various project ideas. And finally, we decided to pursue this project by considering the impact this project has on many applications and our interest towards machine learning.
We started off by studying various research papers, learning and understanding about the project. We tried to search for a better approach to improve the accuracy of the model as image captioning has made significant advances in recent years and already many architectures have been suggested to make a state-of-art model . Recent work based on deep learning techniques has resulted in a breakthrough in the accuracy of image captioning. We have studied and understood the architecture of image captioning mentioned in various papers and also implemented various sections during the project to understand it. We use this information to make a new hybrid model which has more scope than previous models and at the same time better accuracy.

The future scope for this project are:

1. Tuning hyper-parameters of LSTM of such as learning rate, structure layer can give different results.
2. The expanding application scope of visual understanding in the fields of medicine, security, military and other fields, which has a broad application prospect.
3. We can make an application which does live captioning i.e we can caption a video.

References

[1] S. Banerjee and A. Lavie. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments.* Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43rd Annual Meeting of the Association of Computational Linguistics, 2005.

[2] Aaron Li-Feng Han. *LEPOR: An Augmented Machine Translation Evaluation Metric.* Computation and Language, 2017.

[3] KyungHyun Cho Yoshua Bengio Junyoung Chung, Caglar Gulcehre. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.* NIPS 2014 Deep Learning and Representation Learning Workshop, 2014.

[4] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. *Deep Residual Learning for Image Recognition.* The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[5] Andrew Zisserman Karen Simonyan. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* Computer Vision and Pattern Recognition, 2017.

[6] Tony Lindsberg. *Scale invariant feature transform.* Computer Sciences Computer Vision and Robotics (Autonomous Systems), 2012.

[7] H. Al-Qassab M. Al-Qizwini, I. Barjasteh and H. Radha. *Deep learning algorithm for autonomous driving using GoogLeNet.* 2017 IEEE Intelligent Vehicles Symposium (IV), 2017.

[8] P. S. Bronsart Gregory A. Sanders Mark A. Przybocki, Kay Peterson. *The NIST 2008 Metrics for Machine Translation Challenge.* Machine Translation, 2010.

[9] Christopher Yakopcic Stefan Westberg Paheding Sidike Mst Shamima Nasrin Brian C Van Esesn Abdul A S. Awwal Vijayan K. Asari Md Zahangir Alom, Tarek M. Taha. *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches.* Computer Vision and Pattern Recognition, 2018.

[10] Mohd Fairuz Shiratuddin Md. Zakir Hossain, Ferdous Sohel and Hamid Laga. *A Comprehensive Survey of Deep Learning for Image Captioning.* ACM Comput. Surv. 0, 0, Article 0, 2018.

[11] S. Ward T. Zhu W. J. Papineni, K. Roukos. *BLEU: a method for automatic evaluation of machine translation.* ACL-2002: 40th Annual meeting of the Association for Computational Linguistics., 2002.

[12] Daniel.D.Morris Regis Hoffman Paul Rybski, Daniel Huber. *Visual classification of coarse vehicle orientation using Histogram of Oriented Gradients features.* 2010 IEEE Intelligent Vehicles Symposium, 2010.

[13] Alex Sherstinsky. *Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network.* Special Issue on Machine Learning and Dynamical Systems, 2018.

[14] M.Pietikainen T.Ahonen, A.Hadid. *Face Description with Local Binary Patterns: Application to Face Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006.

[15] A. Chelba C. Wang, Y. Acero. *Is Word Error Rate a Good Indicator for Spoken Language Understanding Accuracy*. IEEE Workshop on Automatic Speech Recognition and Understanding, 2003.