DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING - GUINDY

ANNA UNIVERSITY, CHENNAI- 600025

CS6301 – MACHINE LEARNING

IMAGE INSCRIPTION AND INTONATION

A NEURAL NETWORK APPROACH

NAME	REG.NO	AADHAR NO	E-MAIL	MOBILE NO
SRIHARI. S	**2018103601**	5850 2258 1455	srihari961112@gmail.com	9901017871
T.K.S. ARUNACHALAM	**2018103616**	8847 8977 1257	tksarunachalam2508@gmail.com	7550221044

INSTRUCTOR & MENTOR: **Dr AROCKIA XAVIER ANNIE R**

Table of Contents

Abstract

Digital communication technologies have greatly influenced and expanded the way humans interact. The progress of information technology has opened wider opportunities for communication. Social networks have become the modern-day social communities connecting people from different parts of the globe, sharing images and videos on these platforms. By creating virtual communities, digital communication has expanded the scope of communication eliminating barriers. We aim to make further progress in this arena by describing an image in the form of audio to visually impaired people. A certain section of differently abled people is unfortunately isolated from this world. In-order to combat this issue we have come up with a system that describes an image shown in the form of plain text using an encoder-decoder architecture and is integrated with an end-to-end lexical articulator which produces a vocal description of the given image.

Chapter 1: INTRODUCTION

1.1 Problem Statement

Technology has become an integrated part of our daily lives over the past decades. The efficient processing and association of different multimodal information is a very important research field with a great variety of applications, such as human computer interaction, knowledge discovery, document understanding, etc. Computerized elucidation of an image has been one of the primary goals of computer vision. Not only must description generator models be powerful enough to solve the computer vision challenges of determining which objects are in an image, but they must also be capable of capturing and expressing their relationships in a natural language. It is a very important challenge for machine learning algorithms, as it amounts to mimicking the remarkable human ability to compress huge amounts of salient visual information into descriptive language. Hence, to tackle this conundrum we present the development of a novel methodology to extract meaningful information from images, in the form of short

descriptions. The results can be further run through a lexical articulator engine to offer full sustainability. This way, a full independent experience could be delivered to visually impaired people. For an accurate algorithm, it must be fed with a good amount of data. The dataset which we are planning to use for training and testing the model is MS-COCO (MicrosoftCommon Objects in Context) which contains more than 80000 labelled images. For converting still images into natural language text sentences we must first start from the understanding of the context of an image and secondly how this context is expressed into natural language. Thus, for understanding the image a feature extracting convolutional neural network can be used. With the aid of recurrent neural networks, the extracted features can be transformed into a suitable textual description. These results are finally fed to a lexical articulator module which produces the output of the system in the form of speech.

1.2 *Problem Scope*

Producing properly structured sentences requires both syntactic and semantic comprehension of the language. Having the option to portray the substance of a picture using precisely framed sentences is an extremely difficult task, however it could likewise have an incredible effect, by helping visually impaired people better understand the content of images. This undertaking is fundamentally harder than the deeply scrutinized arenas like object recognition and classification of images. The greatest test is having the option to make a portrayal that should catch the articles contained in a picture, yet in addition express how these objects identify with one another.

The goal of image inscribing is to automatically depict an image in natural language sentences. This is a task that facilitates the amalgamation of computer vision and natural language processing, so its principle challenges emerge from the need of translating between two discrete, yet typically combined modalities. In the first place, it is important to identify objects on the scene and decide the relations among them and then express the picture content accurately with meaningful sentences. The manner in which

the descriptions are formulated is still very different from how humans depict pictures since individuals depend on good judgment and experience, and call attention to significant subtleties. Automatically produced depictions can likewise be utilized for content-based recovery or in social media communications.

Chapter 2 LITERATURE SURVEY

[2] A framework of encoder/decoder system with attention mechanism has been considered to overcome the limitation of considering the image's scene as a whole.Attention mechanism considers the spatial aspects of the image and allows the decoding process to focus on the emphasis and details of the input image at each time step while the output sequences are being produced.Inception V3 model has been used as the encoder and a GRU with state size of 512 is being using in the decoder. The model has been compared with Microsoft's CaptionBot, an online caption generator. This does not reflect the true nature of the system since a standard metric has to be found to calculate the performance of the system

[5] A neural framework has been proposed for generating captions from images derived from probability theory.By using a powerful mathematical model, the probability of the correct translation for both inference and training has been maximized and better results are obtained. The accuracy of model and command of language model learnt from image descriptions is tested on different datasets.Sometimes the generated sentence seems to lose track or differ completely from that of the original image content.

[6] Analyzed different deep neural network-based image caption generation approaches and pretrained models to conclude on the efficient model with fine-tuning. A number of pre-trained models VGG 16, RESNET and Inception were used to compare the results. Adopted RMS prop optimizer instead of Adam as RMSprop is an adaptive learning optimization algorithm, in case of RNN we use it because the model of neural network is sequential. Performed a comparative study by which the approach encompassing attention is found to perform better. The GRU every time looks at the

entire image vector representation and this seems inefficient as different words in a caption are generated by looking at specific parts of the image. The 'tanh' activation in attention gives less choppy and more smoother parts of the sub region over the inner dot product approach. Used BLEU-1 as the sole metric to evaluate the performance of the system which fails to provide the appropriate view.

[7] Based on a recurrent neural network with modified LSTM cells with an additional gate responsible for image features. This modification results in generation of more accurate captions. In this work, we will select Mixed_7c layer from Google Inception and append average pooling layer which will have 2048-dimensional output for image description.maximize the probability of the correct caption for the given image by optimizing the sum of the log probabilities for the whole training set using stochastic gradient descent

[19] The attention based approach improved caption generation accuracy, but it did not always focus on appropriate image parts. The attention accuracy is uncorrelated to the caption accuracy directly. The ordinary attention mechanism does not help us to understand how a caption generation system generates a word.

Convolutional Neural Networks:

Convolutional Neural Networks have the ability to capture the spatial and temporal features of complex images with the help of kernels/filters. CNN transforms images into simpler intermediate forms which are easy for further computation without losing important features. Initial layers of CNN are responsible for capturing the low level features such as points, lines, colour etc. Subsequent convolutional layers extract the high level features and provide interpretation of the image as a whole

Recurrent Neural Networks:

A recurrent neural network (RNN) is a type of neural network which uses sequential data or time series data. It allows previous inputs to be passed as inputs. Thus historical information can be considered. RNNs are used in a variety of natural language processing tasks like speech recognition, language modeling, translation etc

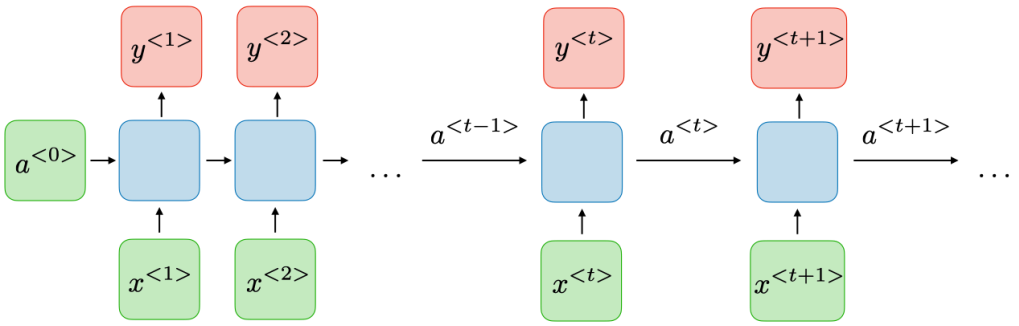

Figure.1 Vanilla RNN

For each timestep t, we have activation $a^{<t>}$ and output $y^{<t>}$

Issues with RNN

vanishing and exploding gradient are two phenomena RNNs often suffer from.This is because it is difficult to capture long term dependencies. As the no of layers increase we either get an exponential decrease or increase in the Gradient. In order to resolve the vanishing gradient problem certain modifications are done in the RNN by defining specific gates . This led to the introduction of LSTMs and GRUs.

Chapter 3: PROBLEM SOLUTION

3.1 Dataset Description:

The dataset used is COCO (Common Objects In Context) sponsored by Microsoft for various computer vision tasks like Object detection, segmentation, key-point detection, captioning etc. The images can be grouped in 91 categories. The dataset consists of 328K images. The first version of MS COCO dataset was released in 2014. It contains 164K images split into training (83K), validation (41K) and test (41K) sets. In 2015 an additional test set of 81K images was released, including all the previous test images and 40K new images.

The dataset has annotations for object detection - bounding boxes and per-instance segmentation masks with 80 object categories, captioning - natural language descriptions of the images, keypoints detection - containing more than 200,000 images and 250,000 person instances labeled with keypoints (17 possible keypoints, such as

left eye, nose, right hip, right ankle), stuff image segmentation – per-pixel segmentation masks with 91 stuff categories, such as grass, wall, sky (see MS COCO Stuff), panoptic: full scene segmentation, with 80 thing categories (such as person, bicycle, elephant) and a subset of 91 stuff categories (grass, sky, road), dense pose: more than 39,000 images and 56,000 person instances labeled with DensePose annotations – each labeled person is annotated with an instance id and a mapping between image pixels that belong to that person body and a template 3D model. The annotations are publicly available only for training and validation images.

3.2 Overview

Chapter 2 conducts extensive analysis on the challenging tasks with five recent independent works, demonstrating the superiority and wide application scope of our framework. Chapter 3 covers the refined overall system design along with module-wise design and their corresponding implementation methodology, pseudocode and block diagrams. Chapter 4 discusses the experimental results by summarizing the overall system performance with respect to five evaluation metrics attacking each and every qualitative aspect of the generated output. It also includes a detailed workflow of the system with some execution snapshots. The last chapter formally concludes the overall argument, gives few insights regarding the future scope and mentions the references to the aforementioned significant works of contemporaries.

Chapter 4 SYSTEM DESIGN

4.1 Abstract System Architecture

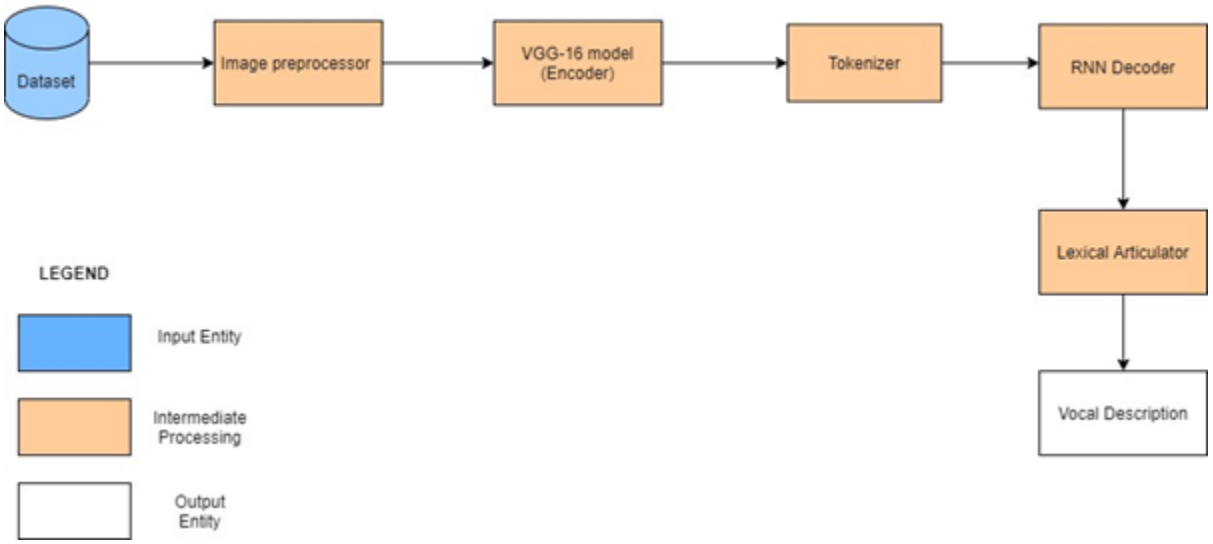

Figure.2 Overall Block Diagram

The system is modularized into five sections as shown in the block diagram.

Module	**Input**	**Output**
Image Pre-processor	MS-COCO Dataset	Pre-processed Images and Captions
VGG-19 Model (Encoder)	Pre-processed input image	Features of the input image
Tokenizer	Captions for the image from the dataset	Sequence of real valued vectors (embeddings)
RNN Decoder	Features of the input image and embeddings from the tokenizer	Textual Description of the image
Lexical Articulator	Textual Description of the image	Vocal Description of the input image

Chapter 5 Detailed Module Design

5.1. Image Preprocessor:

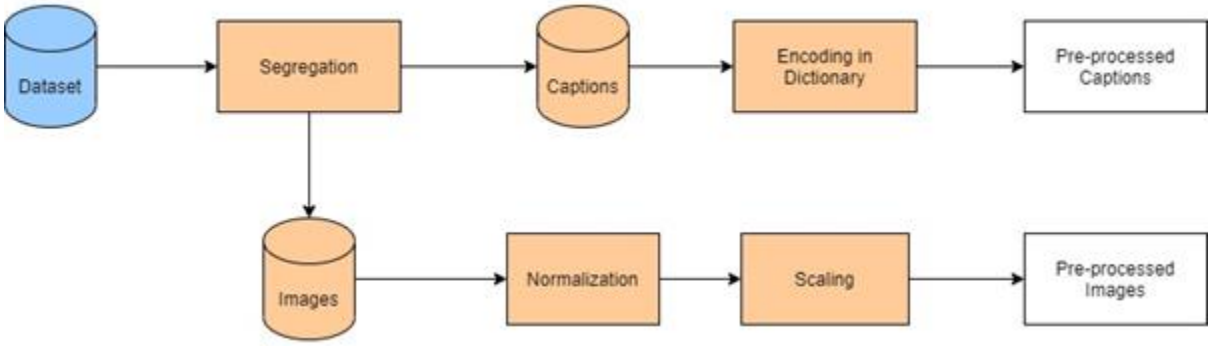

Figure.3 Image Preprocessor - Module Block Diagram

An iterative approach has been chosen for the implementation of the problem statement. The first goal was to understand the nature of the dataset and pre-process it. The input MS-COCO 2014 dataset is of size 25 GB. In-order to deal with this huge dataset and the constrained computing resources we make use of the dynamic programming paradigm by caching the values, the first time the dataset is downloaded, in-order to make access faster the subsequent times.

On loading the dataset, the images and the corresponding captions are then segregated and stored separately. The images then undergo normalization followed by scaling to finish the pre-processing. On the other hand, the captions are encoded in a dictionary and are thus pre-processed so that it could be used by the tokenizer. Caching is used to persist the data so it can be reloaded very quickly and easily. The COCO data-set contains a large number of images and various data for each image stored in a JSON-file. On loading the image it is resized as specified. Further the images are scaled so that their pixels fall between 0.0 and 1.0.

The pycocotools have been put into use. It is a Python API that assists in loading, parsing and visualizing the annotations in COCO.

5.2. VGG-16 Encoder:

In caption generation, we have to extract features from raw images. A pretrained convolutional neural network is used with the help of transfer learning thereby reducing the training time and utilizing the large amounts of training data. We employ VGG16 as the pretrained convolutional neural network. VGG16 has been trained with ImageNet dataset already and can classify images into 1000 object classes.

It extracts enough features from an input image and the features are useful to generate the description of an image. The architecture of VGG16 consists of two modules: a feature extraction module which consists of 13 convolutional layers and a classification module which consists of 3 fully-connected layers. We capture the 15th layer in VGG16 as a feature vector of an input image representing the final output from this feature extraction module.The size of an image feature vector is 4096x1x1.

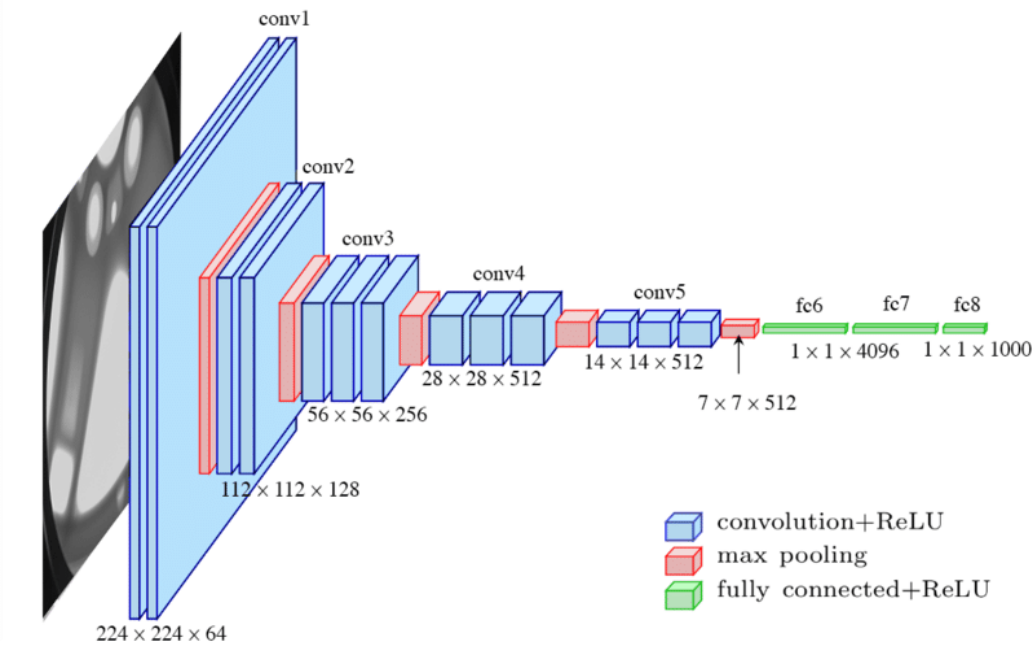

Figure.4 Architecture of VGG-16

Total no of Parameters in VGG 16

conv1-64 x (2) : 38,720

conv2-128 x (2) : 221,440

conv3-256 x (3) : 1,475,328

conv4-512 x (3) : 5,899,776

conv5-512 x (3) : 7,079,424

fc6 : 102,764,544

fc7 : 16,781,312

fc8 : 4,097,000

TOTAL : 138,357,544

5.3. Tokenizer:

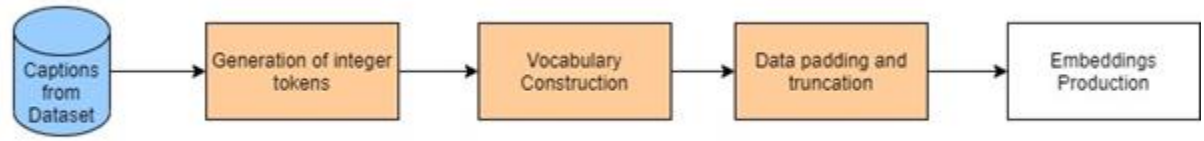

Figure.5 Tokenizer - Module Block Diagram

As a consequence of the inability of neural networks to straightforwardly work on textual-information, a two-step process has been adopted to translate the text into numerical form. Before handling the content, the start and the end of each text series is set apart to monitor the captions. Then the process is commenced by changing the text into integer. The token-sequences are padded with zeros such that each has the same

length before being fed to the decoder. This is followed by converting the integer-tokens into vectors of floating-point numbers using an embedding-layer.

Integer tokens take upon values among 0 and the size of the vocabulary, but the recurrent neural network can't deal with values in such a wide range.Word embeddings is an alternate to one-hot encoding along with dimensionality reduction. This sequence of real valued vectors obtained from the embedding layer helps us to understand the words, their correlation in a better way .

Pseudocode:

```
getDescription(model, tokenizer, image, max_length):
   description := 'ssss'
   for each i of range(max_length) do:
      sequence := tokenizer.texts_to_sequences(description)
      pad_sequences(sequence, max_length)
      yhat := model.predict(image,sequence)
      word := word_for_id(yhat, tokenizer)
      description += ' ' + word
      if word is 'eeee':
        then  break
    end for
   return description
```

```
1 captions_train_marked = mark_captions(captions_train)
```

```
1 captions_train_marked[0]
```

```
['ssss Closeup of bins of food that include broccoli and bread. eeee',
 'ssss A meal is presented in brightly colored plastic trays. eeee',
 'ssss there are containers filled with different kinds of foods eeee',
 'ssss Colorful dishes holding meat, vegetables, fruit, and bread. eeee',
 'ssss A bunch of trays that have different food. eeee']
```

Fig3 - Input before Tonkenizer

```
tokens_train[0]
```

```
[[2, 844, 5, 2845, 5, 60, 25, 1933, 248, 9, 438, 3],
 [2, 1, 423, 10, 3661, 7, 1020, 395, 492, 1076, 3],
 [2, 64, 19, 988, 144, 8, 191, 954, 5, 717, 3],
 [2, 296, 731, 26, 343, 209, 262, 9, 438, 3],
 [2, 1, 170, 5, 1076, 25, 450, 191, 60, 3]]
```

Fig4 -Text Converted to Tokens

Every integer in this is further converted to a vector of 128 floating point values using the Embedding layer.

5.4. RNN Decoder:

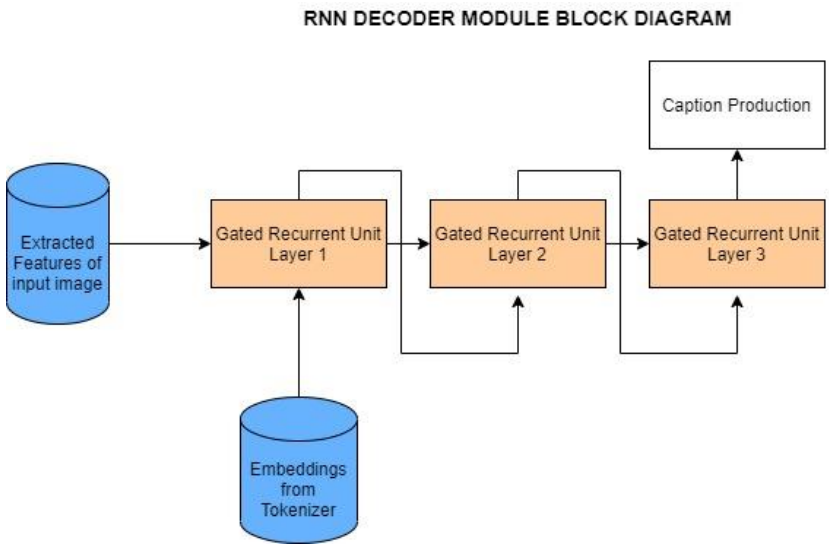

Figure.6 Decoder Module Block Diagram - Algorithm 1

Recurrent neural networks are used for decoding the extracted features from the image. Gated Recurrent Unit (GRU) is a sophisticated recurrent unit used to capture dependencies of various time scales, process memories of sequential data by storing previous inputs in the internal state of networks and plan from the history of previous

inputs to target vectors in principle. GRU consists of Update and reset gates, these gates are responsible for regulating the information to be kept or discarded at each time step.

GRU:

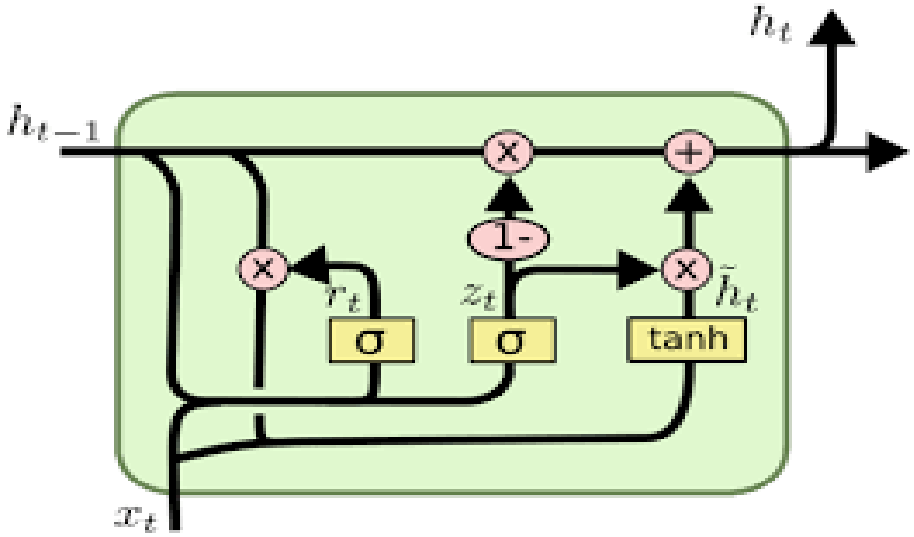

Figure.7 Architecture of Gated Recurrent Unit

Pseudocode:

```
gru_cell(prev_ct, input):
    combine := prev_ct + input
    candidate := candidate_layer(combine)
    ut := update_gate(combine)
    Ct := ut * candidate + (1 - ut) * prev_ct
    return Ct
```

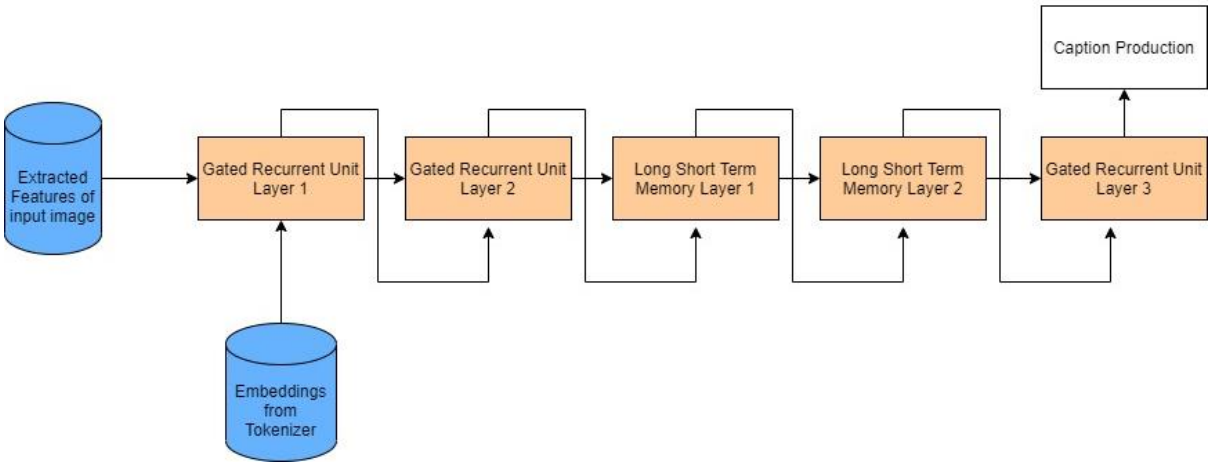

Figure.8 Decoder Module Block Diagram - Algorithm 2

LSTM:

Pseudocode:

```
lstm_cell(prev_ct, prev_ht, input):
    combine := prev_ht  + input
    ft  := forget_layer(combine)
    candidate := candidate_layer(combine)
    it := input_layer(combine)
    Ct := prev_ct * ft + candidate * it
    ot := output_layer(combine)
    ht := ot * tanh(Ct)
    return ht,Ct
```

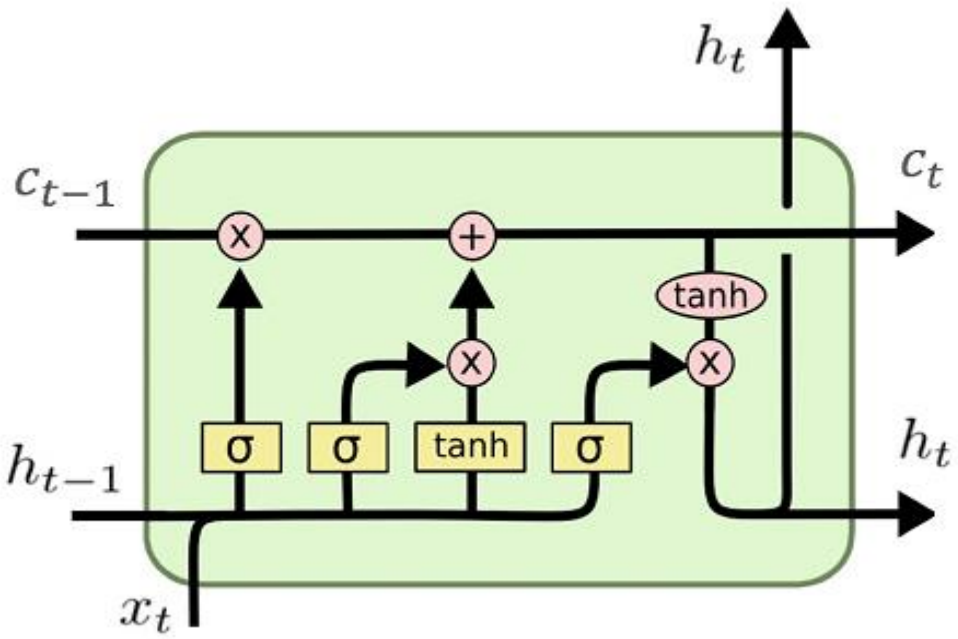

Figure.9 Architecture of Long Short Term Memory

Equations involved in a LSTM

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(W_f\,[h_{t-1}, x_t] + b_f\,)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot \tanh(c_t)$$

C is the memory cell output and h is the Hidden cell state . i_t, f_t, o_t are the activations outputs of input gate, forget gate,and output gate, respectively.

LSTMs have the ability to control the information added/removed to the cell state with the help of update and forget gate. The output is based on the product of the output gate and the cell state. The controlling properties of these gates give LSTM its power to handle long

5.5. Lexical Articulator:

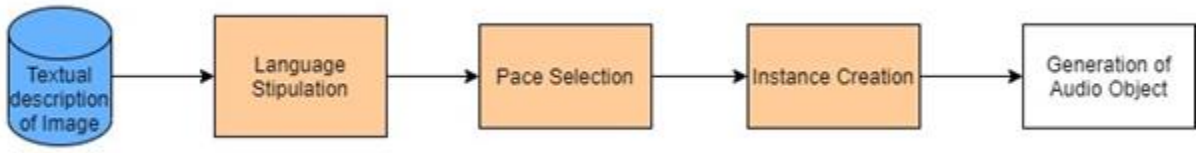

Figure.10 Lexical Articulator - Module Block Diagram

The final module of this system brings into play a state-of-the-art Lexical Articulator. Being fed with the textual description of images from the decoder, the language to be translated into is stipulated. Further we specify the pace of the vocal description we

wish to obtain. Next, we create an mp3 instance of the description using the gtts package. The terminal step in this module is the creation of an audio object which produces the required vocal output and enables us to play the articulated description.

Pseudocode:

```
articulator(text):
    language := 'en'
    pace := slow
    articulator := gTTS(text)
    articulator.setLanguage(language)
    articulator.setPace(pace)
    articulator.save(articulated.wav)
    with open(articulated.wav) as audio:
        IPython.display.Audio(audio)
```

Chapter 6 Implementation

6.1. Tools Used

Major Tools and Packages used in implementation of the project

1. Python 3

2. Keras

3. Pycocotools (to load coco dataset and annotations)

4. OpenCV

5. TensorFlow

6. Pillow

7. Matplotlib

8. PyWebIO (for creating the user interface)

6.2 Initial Setup:

Our system consists of deep neural networks with a lot of trainable parameters which require high computation power, thus using GPU for this task is essential. We have utilised Google Colab's runtime with GPU accelerator. This enables faster training and prediction. However Colab offers only a maximum of 15GB of persistent storage and since our dataset is of size 25GB we had to leverage caching techniques to load the entire dataset into memory. Packages like nltk.blue_score, pycocotools, pycocoeval, gtts, ngrok, pywebio has be installed and setup

6.3 Code Snippets

M1 - 1.1Cache function

```
def cache(cache_path, fn, *args, **kwargs):
    if os.path.exists(cache_path):
        # Load the cached data from the file.
        with open(cache_path, mode='rb') as file:
            obj = pickle.load(file)

        print("- Data loaded from cache-file: " + cache_path)
    else:
        obj = fn(*args, **kwargs)

        with open(cache_path, mode='wb') as file:
            pickle.dump(obj, file)

        print("- Data saved to cache-file: " + cache_path)

    return obj
```

M1 - 1.2 load record function

```
data_dir = "/content/drive/MyDrive/data/lstm_gru/"
train_dir = "/content/drive/MyDrive/data/lstm_gru/train2014"
val_dir = "/content/drive/MyDrive/data/lstm_gru/val2014"
data_url = "http://images.cocodataset.org/"
def _load_records(train=True):
    if train:
        filename = "captions_train2014.json"
    else:
        filename = "captions_val2014.json"
    path = os.path.join(data_dir, "annotations", filename)
    with open(path, "r", encoding="utf-8") as file:
        data_raw = json.load(file)
    images = data_raw['images']
    annotations = data_raw['annotations']
    records = dict()
    for image in images:
        image_id = image['id']
        filename = image['coco_url']
        record = dict()
        record['coco_url'] = filename
        record['captions'] = list()
        records[image_id] = record
    for ann in annotations:
        image_id = ann['image_id']
        caption = ann['caption']
        record = records[image_id]
        record['captions'].append(caption)
    records_list = [(key, record['coco_url'], record['captions'])
```

```
                    for key, record in sorted(records.items())]

    ids, filenames, captions = zip(*records_list)

    return ids, filenames, captions
```

M2 - 2.1 process_image function

```
def process_images_train():

    print("Processing {0} images in training-set
...".format(len(filenames_train)))

    cache_path = os.path.join(data_dir,

                              "transfer_values_train.pkl")

    transfer_values = cache(cache_path=cache_path,

                            fn=process_images,

                            data_dir=train_dir,

                            filenames=filenames_train)

    return transfer_values
```

M3 - 3.1 connect decoder function

```
def connect_decoder(transfer_values):

    initial_state = decoder_transfer_map(transfer_values)

    net = decoder_input

    net = decoder_embedding(net)

    net = decoder_gru1(net, initial_state=initial_state)

    net = decoder_gru2(net, initial_state=initial_state)

    net                    =                    decoder_lstm1(net,
initial_state=[initial_state,initial_state])

    net                    =                    decoder_lstm2(net,
initial_state=[initial_state,initial_state])
```

```
        net = decoder_gru3(net, initial_state=initial_state)
        print(net)
        decoder_output = decoder_dense(net)
return decoder_output
```

Chapter 7: EXPERIMENTAL RESULTS AND ANALYSIS

7.1 Results obtained on the two different algorithms

Using VGG16 model as an encoder with 16 hidden layers and GRU network (using 3 GRU layers) as decoder with number of epochs set to 300 for optimum performance taking 82783 images from MS-COCO dataset as training dataset and vocabulary size of 10000 unique words, Figure 6-9 shows prediction for an image in validation set.

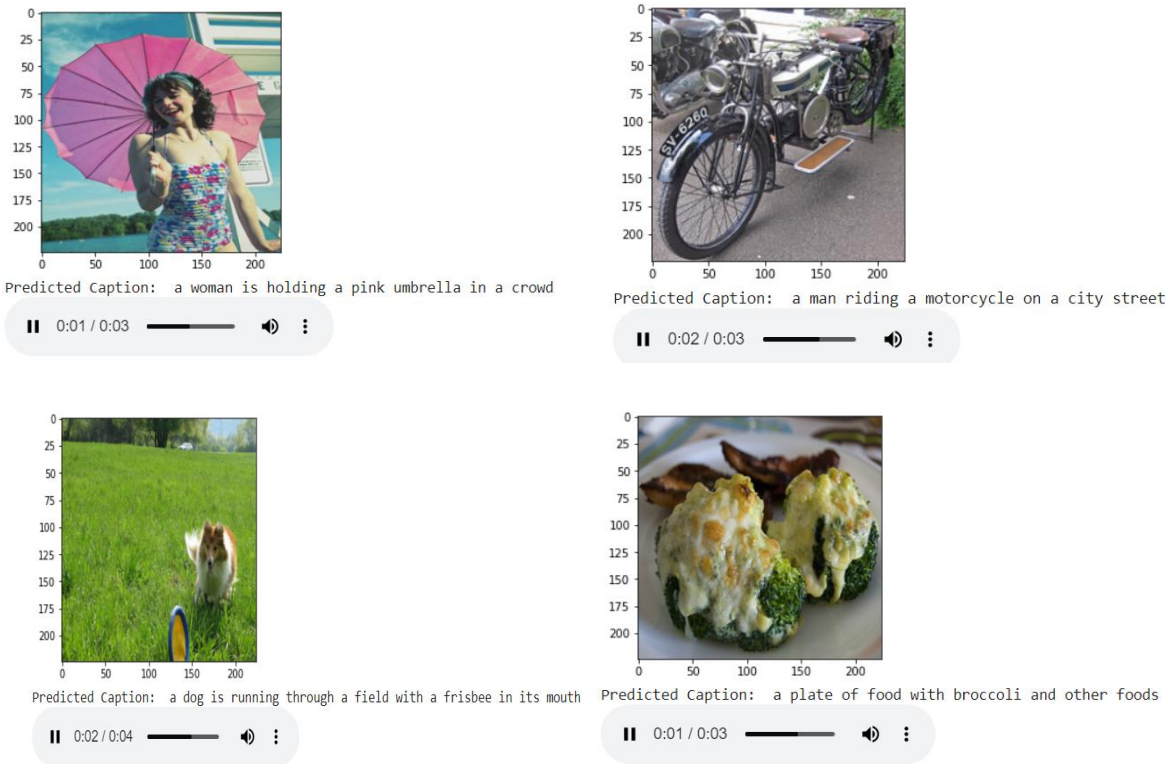

Figure.11 Predicted caption and generated audio using Algorithm - 1

Using VGG16 model as an encoder with 16 hidden layers and the decoder network comprising of a fusion of 2 GRU layers, 2 LSTM layers and 1 GRU layer, with number of epochs set to 300 taking 82873 images from MS-COCO dataset as training dataset

and vocabulary size of 10000 unique words, Figure 10 to Figure 13 outputs are observed.

Figure.12 Predicted caption and generated audio using Algorithm - 2

There are certain images where the performance of the model is just above par like those in Figure5. While for some images it needs more training.

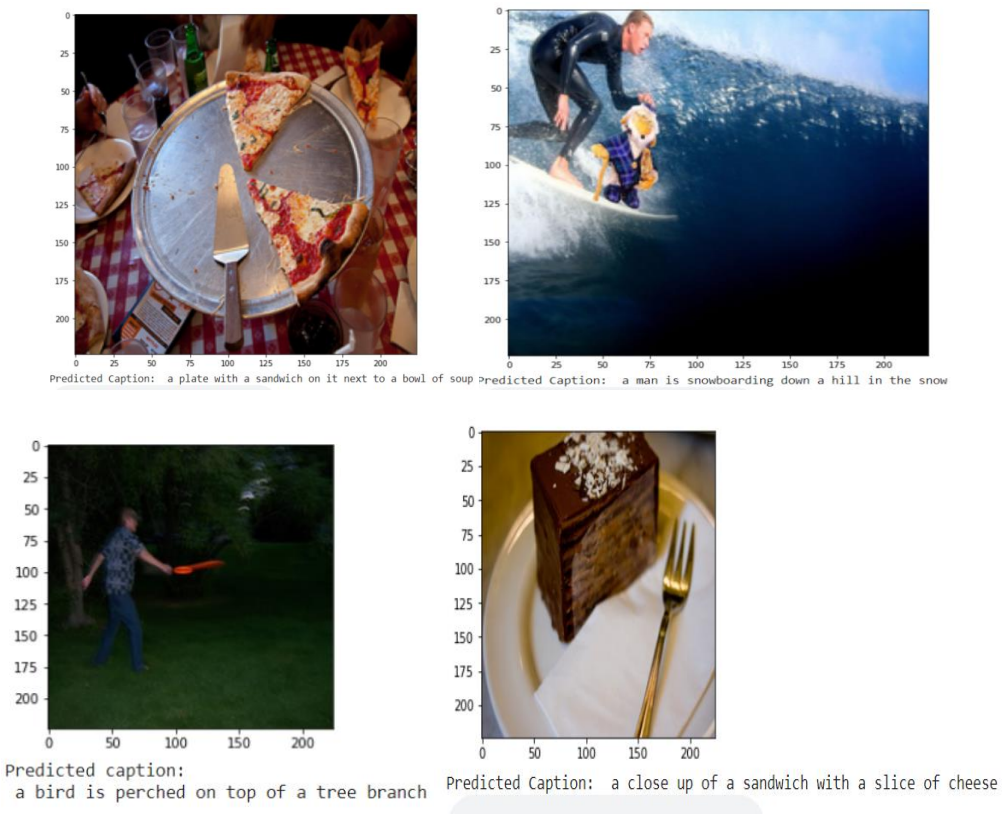

Figure.13 Images where the model underperforms

7.2. Evaluation Metrics

BLEU (Bilingual Evaluation Understudy) is a score for comparing a candidate translation of text to one or more reference translations.Blue score is a number between 0 and 1.This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts. A perfect score of 1 indicates that the candidate is identical to one of the reference translations.

Table.1 **Comparison of BLEU scores**

METRIC	ALGORITHM - 1	ALGORITHM - 2
BLEU-1	0.497	0.580
BLEU-2	0.297	0.406
BLEU-3	0.179	0.280
BLEU-4	0.103	0.193

Meteor (Metric for Evaluation of Translation with Explicit Ordering) automatic evaluation metric scores machine translation hypotheses by aligning them to one or more reference translations. Alignments are based on exact, stem, synonym, and paraphrase matches between words and phrases. Segment and system level metric scores are calculated based on the alignments between hypothesis-reference pairs.

CIDEr (Consensus-based Image Description Evaluation).

CIDEr metric measures the similarity of a generated sentence against a set of ground truth sentences written by humans. The metric shows high agreement with consensus as assessed by humans.

Table.1 **Comparison of performance metrics**

METRIC	ALGORITHM - 1	ALGORITHM - 2
BLEU-4	0.103	0.193
METEOR	0.165	0.195
ROUGE-L	0.318	0.396
CIDER	0.471	0.600
SPICE	0.128	0.133

SPICE (Semantic Propositional Image Caption Evaluation)

In the first stage, syntactic dependencies between words in the caption are established using a dependency parser pretrained on a large dataset. In the second stage, dependency trees to scene graphs are mapped using a rule-based system Given candidate and reference scene graphs,SPICE metric computes an F-score

ROUGE-L (**Recall-Oriented Understudy for Gisting Evaluation**) – measures longest matching sequence of words using LCS. An advantage of using LCS is that it does not require consecutive matches but in-sequence matches that reflect sentence level word order. Since it automatically includes longest in-sequence common n-grams, we don’t need a predefined n-gram length.

7.3. Analyzing the effect of Dropout

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

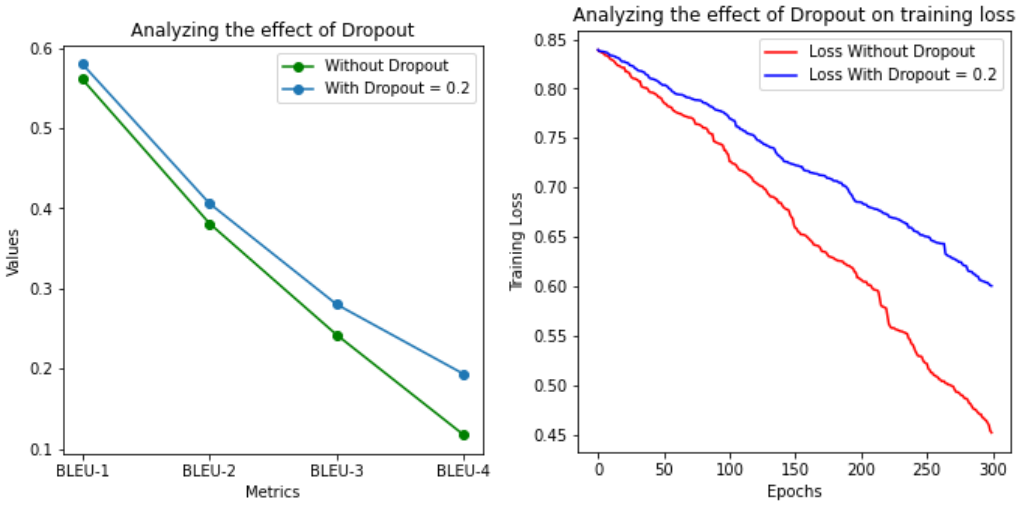

Figure.14 effect of dropout

The dropout parameter in the Long Short Term Memory cells has been tuned to 0.2. This led to an increase in the training loss due to the penalty imposed by regularization. We have observed that adding dropout makes the model generalize well for unseen data as indicated by Figure where the BLEU scores are calculated before and after applying dropout.

7.4. Analyzing the effect of the number of layers

It is observed that the time taken to train the model for an epoch increases as the number of layers in the neural network increases. In the Algorithm-1 where the decoder consisted of 3 Gated Recurrent Units it took around 480 seconds to train for an epoch. Upon the addition of 2 Long Short Term Memory cells, there was a 25% increase in the training time as observed from the figure 8, where it took around 600 seconds to train for an epoch.

The larger the number of epochs, the more time it takes for the algorithm to train, but it also raises the chances of resulting in a better map function. The evolution of the loss function in both algorithms is depicted below.

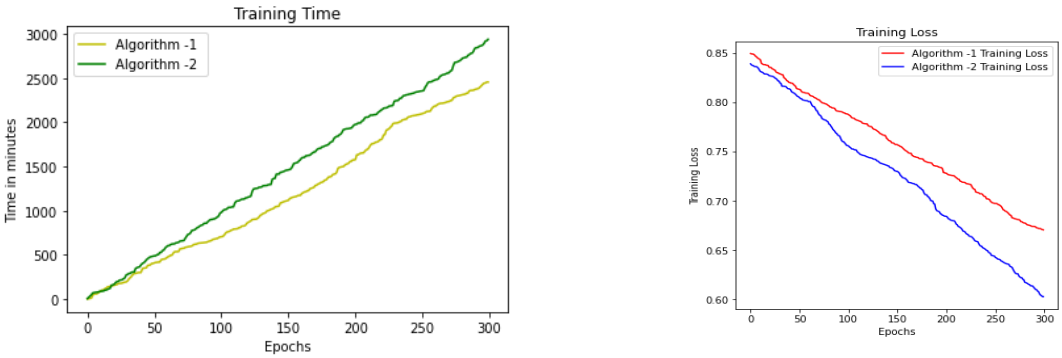

Figure.15 Loss graph and Training time of the model

Sparse_categorical_crossentropy is the best lost function to use as it produces the index of a category index of the most likely matching category and not the one hot encoder(10,000 sized vector). This saves a lot of memory used for computation and representation.

Generally ,Adam optimizer is believed to work well for Computer Vision tasks , but we found out RMSprop to Converge efficiently for our tasks. We have used RMSprop with the learning rate of .001

7.5. Graphical Analysis of Performance Metrics

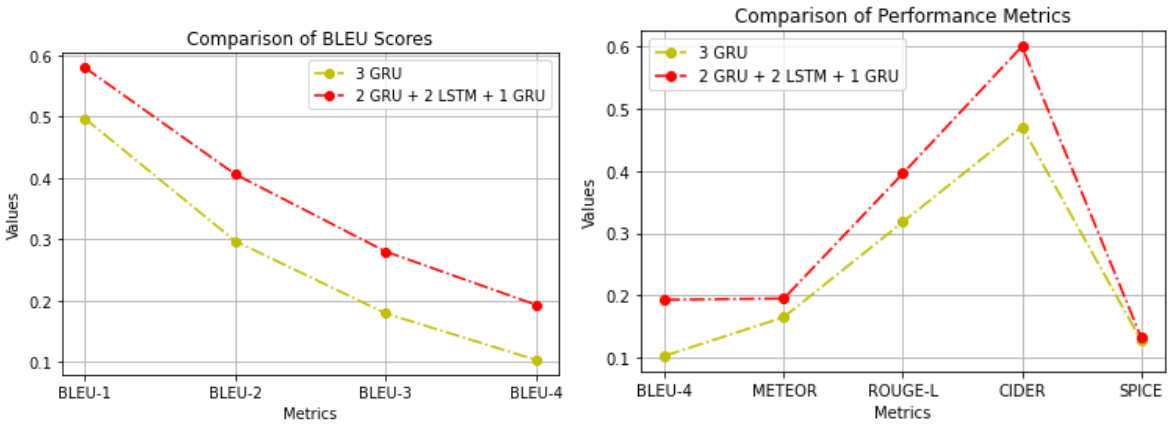

Figure.16 Graphical Analysis of BLEU Scores ***Graphical Analysis of Performance Metrics***

From the above graphs it can be inferred that the algorithm 2 where the decoder is composed of a combination of 2 Gated Recurrent Units, 2 Long Short Term Memory and 1 Gated Recurrent Unit performs better than the algorithm 1 where the decoder is composed of 3 Gated Recurrent Units. The two algorithms are compared among 5 metrics - BLEU, METEOR, ROUGE-L, CIDER and SPICE and the Algorithm 2 outperforms the other in all the metrics.

Chapter 8: CONCLUSION

8.1 Future Scope

The system can be further enhanced by using attention techniques which pay attention to the spatial aspects of the image by using all the hidden state output of the input sequence. Attention mechanism allows the RNN to focus more on certain important parts of the input. This generally leads to increase in the performance of the model. The basic principle we followed in Image Captioning can also be extended to Video captioning with few changes modifications in the encoder-decoder architecture. Extra features such as Optical character recognition(OCR), a responsive voice assistant can be embedded on top of the system.

8.2 Conclusion

We have fabricated an end-to-end neural network system which mechanically views an image and provides a suitable description in simple language. It is based on a convolution neural network which, by encoding the input image into a compact representation, forwards it to a recurrent neural network that generates a corresponding description. The generated description is then being fed to a state-of-the-art lexical articulator which produces the corresponding vocalized version. Thus a visual image is mapped to a textual narrative

In this work a Recurrent Neural Network (RNN) with Gated Recurrent Units and Long Short-Term Memory (LSTM) cells have been developed. It is observed that the

additional LSTM cells added to work increases the model efficiency. Two models, one comprising only the GRU and the other the fusion of LSTM and GRU have been trained on the same MSCOCO image train dataset. The final loss value recorded is 0.67 for the algorithm involving only GRU and 0.60 for the other algorithm involving the amalgamation of GRUs and LSTMs. The measurement of average value of different metrics on the same dataset with these different models was done. These metrics have shown that the new RNN model can generate the image captions more accurately. This new framework provides users with controllability in generating intended captions for images, which may inspire exciting applications.

Albeit the depictions produce energizing outcomes, we believe this is only the start. The future heading will consider a framework that can all the more explicitly portray the recognizable depictions of traffic signs and clinical pictures.

Applying an unsupervised approach for both images as well as text to enhance the image inscribing process seems quite promising. More explorations can be done to the natural language processing when dealing with the formation of sentences in-order to ensure a cogent description.

REFERENCES

[1] A. Puscasiu, A. Fanca, D. Gota and H. Valean, "Automated image captioning," 2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), 2020, pp. 1-6, doi: 10.1109/AQTR49680.2020.9129930.

[2] A. Hani, N. Tagougui and M. Kherallah, "Image Caption Generation Using A Deep Architecture," 2019 International Arab Conference on Information Technology (ACIT), 2019, pp. 246-251, doi: 10.1109/ACIT47987.2019.8990998.

[3] Show, Attend and Tell: Neural Image Caption Generation with Visual Attention: Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, Yoshua Bengio Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:2048-2057, 2015.

[4] Q. You, H. Jin, Z. Wang, C. Fang and J. Luo, "Image Captioning with Semantic Attention," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 4651-4659, doi: 10.1109/CVPR.2016.503

[5] C. Amritkar and V. Jabade, "Image Caption Generation Using Deep Learning Technique," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-4, doi: 10.1109/ICCUBEA.2018.8697360.
[6] V. Kesavan, V. Muley and M. Kolhekar, "Deep Learning based Automatic Image Caption Generation," 2019 Global Conference for Advancement in Technology (GCAT), 2019, pp. 1-6, doi: 10.1109/GCAT47503.2019.8978293.
[7] A. Poghosyan and H. Sarukhanyan, "Short-term memory with read-only unit in neural image caption generator," 2017 Computer Science and Information Technologies (CSIT), 2017, pp. 162-167, doi: 10.1109/CSITechnol.2017.8312163.
[8] V. Atliha and D. Šešok, "Comparison of VGG and ResNet used as Encoders for Image Captioning," 2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), 2020, pp. 1-4, doi: 10.1109/eStream50540.2020.9108880.
[9] M. Wang, L. Song, X. Yang and C. Luo, "A parallel-fusion RNN-LSTM architecture for image caption generation," 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 4448-4452, doi: 10.1109/ICIP.2016.7533201.
[10] C. Yin et al., "Automatic Generation of Medical Imaging Diagnostic Report with Hierarchical Recurrent Neural Network," 2019 IEEE International Conference on Data Mining (ICDM), 2019, pp. 728-737, doi: 10.1109/ICDM.2019.00083.
[11] B. Wang, X. Zheng, B. Qu and X. Lu, "Retrieval Topic Recurrent Memory Network for Remote Sensing Image Captioning," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 256-270, 2020, doi: 10.1109/JSTARS.2019.2959208.
[12] G. Hoxha, F. Melgani and J. Slaghenauffi, "A New CNN-RNN Framework For Remote Sensing Image Captioning," 2020 Mediterranean and Middle-East Geoscience and Remote Sensing Symposium (M2GARSS), 2020, pp. 1-4, doi: 10.1109/M2GARSS47143.2020.9105191.
[13] M. Yang et al., "An Ensemble of Generation- and Retrieval-Based Image Captioning With Dual Generator Generative Adversarial Network," in IEEE Transactions on Image Processing, vol. 29, pp. 9627-9640, 2020, doi: 10.1109/TIP.2020.3028651.
[14] N. Yu, X. Hu, B. Song, J. Yang and J. Zhang, "Topic-Oriented Image Captioning Based on Order-Embedding," in IEEE Transactions on Image Processing, vol. 28, no. 6, pp. 2743-2754, June 2019, doi: 10.1109/TIP.2018.2889922.
[15] M. Zhang, Y. Yang, H. Zhang, Y. Ji, H. T. Shen and T. Chua, "More is Better: Precise and Detailed Image Captioning Using Online Positive Recall and Missing Concepts Mining," in IEEE Transactions on Image Processing, vol. 28, no. 1, pp. 32-44, Jan. 2019, doi: 10.1109/TIP.2018.2855415.
[16] S. Takada, R. Togo, T. Ogawa and M. Haseyama, "Generation of Viewed Image Captions From Human Brain Activity Via Unsupervised Text Latent Space," 2020 IEEE International Conference on Image Processing (ICIP), 2020, pp. 2521-2525, doi: 10.1109/ICIP40778.2020.9191262.
[17] S. M. Xi and Y. I. Cho, "Image caption automatic generation method based on weighted feature," 2013 13th International Conference on Control, Automation and Systems (ICCAS 2013), 2013, pp. 548-551, doi: 10.1109/ICCAS.2013.6703998.

[18] Y. Feng and M. Lapata, "Automatic Caption Generation for News Images," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 4, pp. 797-812, April 2013, doi: 10.1109/TPAMI.2012.118.
[19] M. Shozu and H. Yanagimoto, "Attention Analysis in Caption Generation," 2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI), 2019, pp. 95-98, doi: 10.1109/IIAI-AAI.2019.00029.
[20] Y. Zhenyu and Z. Jiao, "Image Caption Method Combining Multi-angle with Multi-modality," 2019 IEEE 11th International Conference on Advanced Infocomm Technology (ICAIT), 2019, pp. 24-30, doi: 10.1109/ICAIT.2019.8935913.

APPENDIX A:

The undersigned acknowledge they have completed implementing the project “Image Inscription and Intonation - A Neural Network Approach” and agree with the approach it presents.

Signature : _____ Date: 24/05/2021

Name : Srihari. S

Signature : _____ Date: 24/05/2021

Name : T.K.S. Arunachalam

APPENDIX B:

The audio output obtained by the system for the input images shown in Figure.12 is as follows (The link for the generated audio files are inserted below)

A.1 Articulated-Caption

A.2 Articulated-Caption

A.3 Articulated-Caption

A.4 Articulated-Caption

A.5 Articulated-Caption