KLE Society's
KLE Technological University, Hubballi.

A Machine Learning (22ECSC306) Course Project Report
on

ATTENTION BASED IMAGE CAPTION GENERATOR USING ENCODER-DECODER ARCHITECTURE

ABSTRACT

The image captioning is utilized to develop the explanations of the sentences describing the series of scenes captured in the image or picture forms. The practice of using image captioning is vast although it is a tedious task for the machine to learn what a human is capable of. The model must be built in a way such that when it reads the scene, it recognizes and reproduce to-the-point captions or descriptions. The generated descriptions must be semantically and syntactically accurate. Hence, availability of Artificial Intelligence (AI) and Machine Learning algorithms viz. Natural Language Processing (NLP), Deep Learning (DL) etc. makes the task easier. In the proposed paper, anew introduction to attention mechanism called Bahdanau's along with Encoder-Decoder architecture is being used so as to reflect the image captions. It uses a pre-trained Convolutional Neural Network (CNN) called InceptionV3 architecture to gather the features of images and then a Recurrent Neural Network (RNN) called Gated Recurrent Unit (GRU) architecture in order to develop captions. The results obtained from this model trained on Flickr8k dataset with the improvement in accuracy around 10

Keywords- Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), Encoder, Decoder, Attention mechanism, Image captioning.

ACKNOWLEDGEMENTS

The sense of contentment and elation that accompanies the successful completion of our project and its report would be incomplete without mentioning the names of the people who helped us in accomplishing this.

We are indebted to our guide Prof. Uday N Kulkarni who was nothing but a constant source of enthusiasm and whose profound guidance, valuable suggestions, and beneficent direction was mainly responsible for us to complete this project.

We take this opportunity to express our deep sense of gratitude and sincere thanks to our Head Dr. Meena S. M. for her tremendous source of inspiration and help in challenging our effort in the right direction.

Last but not least we like to thank all the course faculty, teaching and non-teaching staff for helping us during the project.

Kushagra Tomar
Mayuri Kalmat
Pranav Jadhav
Rakshita Bandi

Contents

List of Figures

1 INTRODUCTION

Language is the medium through which the society constantly interacts, be it written or spoken.It typically describes the perceptible world around us. The photos and symbols are otherwise to speak and perceive by the physically disabled individuals. Automatic description generation from an image in proper sentences is a tedious task, nevertheless it can have an ample impression on visually challenged individuals for better understanding of the description of pictures on the web. Long back, "Image or Picture Captioning" [**2 3 4**] has always been a rigid mission and the generated captions for the given image were not so pertinent.In conjunction with the progress of Deep Learning [**28**], CNN and techniques for processing text like NLP[**29**], a number of previously challenging pieces of work became straightforward using Machine Learning. These are profitable in recognition, classification and captioning of images and several additional AI [**26**] applications. This "Image Captioning" [**2**] is pragmatic in various applications viz. The concept of self-driving cars, which is now the matter of the moment. In comparison with classification and object recognition, the task of automatically generating captions and describing images is significantly more complicated. A description of an image must include more than just the objects in it, also how the objects are related to their attributes and activities.

1.1 Motivation

Image Caption Generator could help visually impaired people better understand the content of images on the web. It could provide more accurate and compact information of images/videos in scenarios such as image sharing in social networks or video surveillance systems. This task of automatically generating captions and describing the image is significantly harder than image classification and object recognition. The description of an image must involve not only the objects in the image, but also relation between the objects with their attributes and activities shown in images. It reads the scene, then recognizes and reproduces to-the-point captions or descriptions in the human readable form. A caption for an image or picture describes the series of scenes captured in the picture or image.

1.2 Objectives

- To generate meaningful and grammatically valid captions for a given image.
- To develop an efficient image captioning model using encoder-decoder architecture.
- To utilize the attention mechanism in improving efficiency.
- To focus on sensitive local features of images while generating the captions.

1.3 Literature Survey

[1]Image Caption Generator Using Attention Mechanism

The authors of [1] have made use of CNN as an encoder to extract the characteristics or attributes from the images. CNN is a pre-trained InceptionV3. Owing to the InceptionV3's fact that it is a deep network for object detection, it demands to be altered slightly to assist in encoding. A vector which containes features is obtained from this deep network by removing the terminating layer. The feature vector so obtained is of the size (8x8x2048). The feature vector is the input to RNN. The RNN employed for decoding is GRU [17]. To generate more focused captions,Bahdanau's attention is used.

[2]Image Captioning Using Inception V3 Transfer Learning Model

The writers of [2] have proposed a model where the input sent is Flickr8k dataset, and the output obtained is passed to the latest layer which is completely connected and is introduced at the termination of the InceptionV3 model. The task of this layer is to transform the model's output into a vector which embed words. It serves as an input to an LSTM cell order by implanting a vector. The LSTM unit attaches the series of information and collects it progressively hence enabling the establishment of meaningful captions. The start-V3 component of this model is trained to recognize complete possible objects in a picture. Each word in the picture is predicted using the previous words in the phrase. The main intention of training is to reduce the failure function. They have used the Flickr8k dataset which has nearly 8000 images and each image is tagged with five unique captions or descriptions which offer compact reports of the noteworthy features.

[3]Image Caption Generation Using Deep Learning Technique

In [3], the authors have put forth a model that allows neural networks to view an image automatically and yield meaningful captions similar to natural English sentences. It is a well-trained model to perform the above-mentioned tasks. Here, the pre-trained CNN is utilized to classify images. This network handles the task of encoding images. The input to the RNN (decoder here) is the hindmost hidden layer of the encoder. The decoder generates sentences. The dataset being used here is Flickr8k[21] consists of about 8000 images and five descriptions tagged to every image. They used VGG [15] for large-scale image recognition. They conclude that using a bulky dataset boosts the performance of the model. In addition to reducing losses, it also improves accuracy.

[4]Image Caption Generation Using Deep Learning

To achieve better results, the authors of [4] worked on a model that combines CNN architecture and LSTM for image captioning. The proposed model uses three CNN architectures: ResNet-50 [26], Xception [25], and InceptionV3 [9]. The aptest combination of CNN and LSTM is chosen based on the model's accuracy. Training is performed onthe Flickr8k data. Combining Xception with LSTM has the highest accuracy of 75

[5]Visual mage Caption Generator Using Deep Learning

The authors of [5] proposed a model where CNN features are extracted from an image and encoded into vector representations using 16 convolutional layers of the VGG-16. Next, a RNN decoder model is used to develop corresponding sentences based on the learned image features i.e., training the features with captions or descriptions provided in the dataset. The input images are processed using two architectures viz, GRU and LSTM. Through the results, it is evident that the LSTM model achieves better results than the GRU [17] model. Although, it takes a longer time to train and generate captions due to the model's complexity.

[6]Deep Learning Based Image caption Generator

In [6], the authors have developed a model where preprocessed images are fed into the Inception V3 model, and the features are extracted. Later, a D-dimensional representation of each and every part of the image is produced by the extractor such as L vectors. With the spatial features of a CNN convolution layer, the decoder calculates the context vector according to the certain sectors of the input image. For the decoder's job, GRU is utilized which has a simpler structure than LSTM [16] . The vanishing gradient problem does not affect GRU, unlike RNN. Thus, it proves that usage of GRU gives better results than LSTM.

[7] Image Caption Generating Deep Learning Model

The authors display a method to overcome the vanishing gradient problem which hinders the existing CNN-RNN models in [7] . They have proposed ResNet-LSTM as an encoder-decoder technique for image captioning. The ResNet (encoder) extracts the features and the LSTM (decoder) generates the caption from the extracted features. For this, the images are resized to (224x224x3) and subjected to several pre-processing steps. They have used the Flickr8k dataset for training the model. After a minimum of 20 epochs, meaningful captions begin to generate. It is better than VGG and CNN-RNN models.

[8] Encoder-Decoder based Multi-Feature Fusion Model For Image Caption Generation

The authors of [8] explain a multi-feature fusion model to generate image captions. Models that currently exist focus on the global characteristics of an image, but with the comprehensive features, this model also considers the localized features of images. A

global feature extraction of global features is performed using the VGG16 network and Faster-CNN is used to excerpt the local characteristics. The local and global features are mixed and fed as input, through an attention layer to the Bi-LSTM [29]. The caption obtained is corrected if any error occurs. ImageNet dataset with image size (224x224x3) is used to train VGG16 [15], Pascal VOC dataset is used to train Faster-RCNN [30] (1:1 positive and negative sample ratio maintained). Bi-LSTM is trained with the MSCOCO dataset [28]. The fused features have turned out to be superior to global or local features alone. The test accuracies in the training set and verification set are 78.20

1.4 Problem definition

Develop an Image Captioning model using Encoder-Decoder Architecture and Attention Mechanism which generates semantically correct captions for the images.

2 PROPOSED SYSTEM

2.1 Data set description

The dataset being used is Flickr8k. It consists of 8092 images in JPEG format. Each image has 5 descriptions tagged with it in the caption.txt file. The Flickr8k dataset has been downloaded from kaggle.com. Since each image in the dataset has 5 captions, there are nearly 40460 captions available in the caption.txt file.

A close-up of a young girl with a pink shirt laying on grass
A girl in a pink shirt lies on her back in the grass
A girl wearing a pink shirt lays on the green grass and looks up
A young girl is laying in the grass posing for the camera
Girl in pink shirt with white writing lying in grass

Figure 1: sample 1.

A baseball is recoiling from an action taken on a treated field watched by others
A baseball player on a playing field springs into action
A baseball player standing on the mound
A Philadelphia Phillie pitcher on the pitchers mound with his left leg up behind him
A pitcher in a red and white uniform in a baseball game has just thrown the ball

Figure 2: sample 2.

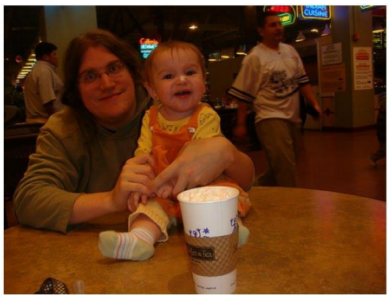
Inference: The images are tagged with five captions which describe the image.

2.2 Understanding the Data

Flickr8k dataset consists images of assorted scenes and instances, not confined to the well known people or places. This exposes the model to various scenes which results in better training.

a. white bird flaps its wings above the water

b. hockey player in helmet



c. man holds baby on the table at restaurant

d. three dogs playing in the green water

Figure 3: Sample images.

Merits of using Flickr8k dataset:

- It is a light sized dataset that can be used to train the model on laptops and other low end devices.
- It is available for free.
- The data is labeled really well, as each image is tagged with 5 captions.

Then, Python's TensorFlow library is used to preprocess the images

2.3 Initial Approach

There are various ways of image captioning which include ResNet-LSTM, VGG-LSTM and other models. Based on the survey, it is evident that the majority of the existing models are hindered by the vanishing gradient problem. Also, the captions generated do not focus on the sensitive features of the image.

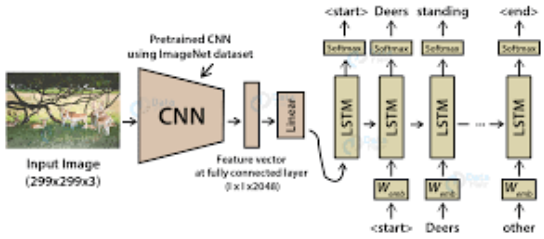


Figure 4: CNN-LSTM model

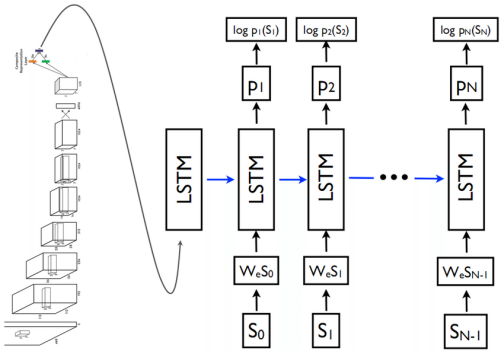


Figure 5: ResNet-LSTM model

To tackle these problems, the solution proposed is the usage of attention mechanism with the InceptionV3-GRU(encoder-decoder) model.

2.4 Proposed methodology

In ABICG, the CNN used is InceptionV3 [9] pretrained on ImageNet weights, which serves as an encoder. This extracts the features from the receptive fields of the images and forwards it to the decoder. Here, the RNN used is GRU, which is used as a decoder. The use of the decoder is to decode the sentence from the encoding. The Bahdanau's attention model [11] is used to enhance the capability of the decoder by allowing it to focus on the important aspects of the images while producing the captions. Thus, taking care that the sensitive parts of the image are not left out in the generated caption.

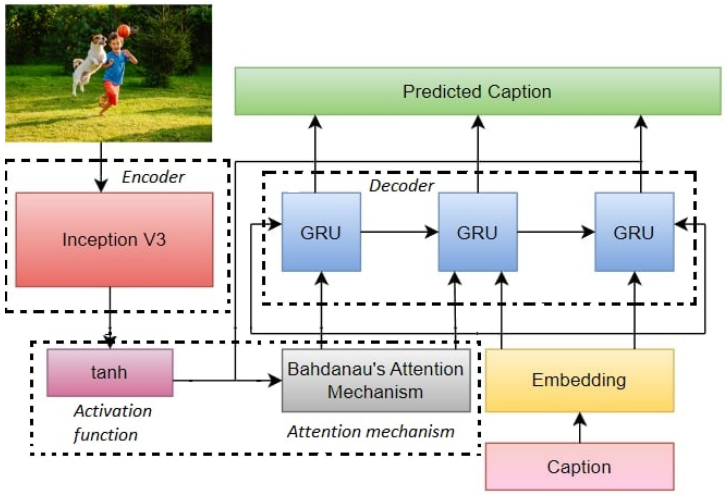


Figure 6: ABICG Architecture

The output of the encoder is fed to the decoder which predicts the captions based on the output of previous timestamp. Attention mechanism used, allows the decoder to focus on the more important part of the image when generating the captions.

Convolutional Neural Network (CNN)

InceptionV3 [9] is often used for image recognition and has been very popular in field of image processing because of its up to the mark accuracy on different datasets. The InceptionV3 encompasses the building blocks which are of types asymmetric and symmetric, along with convolutions, max pooling, average pooling, concatenations, dropouts and various fully connected layers as shown in the Fig 3.

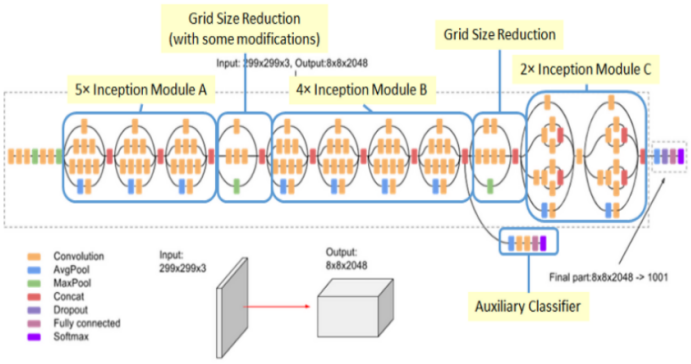


Figure 7: InceptionV3

It was built for the purpose of object detection on receiving a (299x299x3) image. Since InceptionV3 [9] is mostly used for object detection, the refinement of it to some extent is required so as to make it an encoder for extracting the image features. The last layer is eliminated which is used for classifying the images into the labels since classification of images is not required. Thus, a feature vector is obtained which is of the size (8x8x2048). The resulting feature vector is static and does not alter at each timestamp. Therefore, this vector is passed to the attention model along with the hidden state of the decoder to create the context vector.

The benefits of using InceptionV3 CNN for the encoder part is that it generates fewer parameters for computation which makes it computationally less expensive in comparison to the other models and is memory efficient. There is no comparison between InceptionV3 and the other models when it comes to depth and accuracy.

Attention mechanism

Attention model is a deep learning technique that makes use of attention mechanism which provides attention or additional focus on specific components. The Bahdanau's Attention Model [11] is used in this paper. It selectively highlights the relevant features of the input data. It is also referred to as an interface that connects the encoder and decoder.

It instructs the decoder with the relevant details from each and every encoder hidden state. Decoding begins with the context vector generated by the attention model to predict the word at that particular timestamp. The context vector changes with each timestamp since it is adaptive in nature. Attention model Fig 8 does a linear transformation of the input by applying tanh to it so as to introduce non-linearities henceforth achieving a smoother distribution.

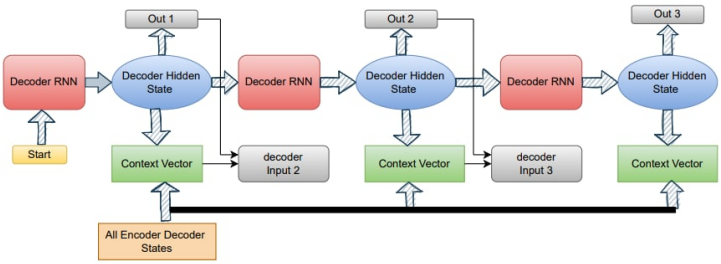


Figure 8: Bahdanau's Attention

Then, the attention score a s is computed. The output is required in the range (0,1). The softmax function is applied to the attention score and the final attention weights are obtained. This model intends to overcome the curb of the orthodox CNN–RNN models. This model facilitates passing various parts of the image instead of the whole. This also makes it swift and hikes its accuracy.

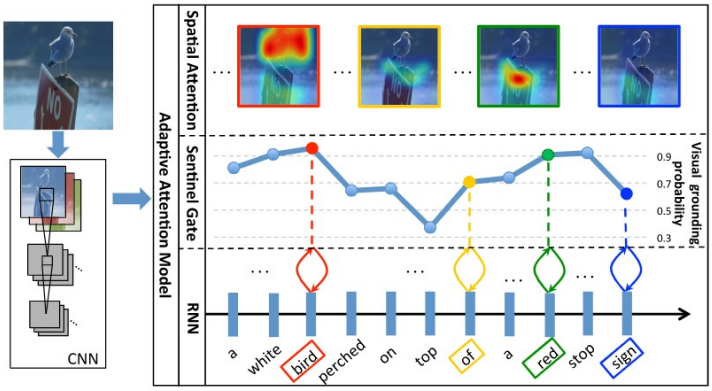


Figure 9: Complete architecture of proposed ABICG model

In the above image, there is a white color bird sitting on the sign board. The image is fed into the encoder which extracts the image features and gives it as an input to the decoder which transforms the image feature vector into concise caption. So here, the caption generated would be "a white bird perched on top of a red stop sign" all in lower case. The project aims at mimicking the human brain because of its abilities to generate a caption for every scene it senses. Therefore, it becomes crucial to add an attention mechanism using which the CNN-RNN model focuses on the more important parts of the image. There is no static vector encoding of the whole image in the attention mechanism. Instead, it adds the spatial information corresponding to the image to the extraction of image features. As a result, statements are described in a more detailed manner as shown in the above Fig. 5. By such means, while generating sentences, simulation of the human vision using the attention mechanism can be encouraged with the generation process of word sequence. This is to ensure that the generated sentence will reflect the expression

habits of the people.

Recurrent Neural Network (RNN)
GRU is used as a decoder. It works on the mechanism of RNN which anticipates expressions in the natural language. The other probably used RNNs are LSTM, Vanilla RNN, and the GRU. Vanilla RNN is not preferred due to its *vanishing gradient problem.*
GRU is similar to LSTM. It owns a few key differences from LSTM: GRU has only two gates whereas, LSTM has three gates. It exposes its total memory and also the hidden layers. GRU not only requires fewer parameters for training, but also has way more effective computation. Thus making it computationally efficient.
GRU is composed of two gates, viz., The update gate and the reset gate. Both these gates in GRU together act as a convex combination which gives the verdict of which information or the data of the hidden state is to be updated and which is to be forgotten. A large number of layers in the network lead to a fall in the derivative product till the partial derivative of the loss function tends to zero, and the partial derivative is abolished. This phenomenon is known as the vanishing gradient problem. In simple words, this means that the initially predicted words are wiped out as the new words are predicted therefore giving less weightage to the initial words and vice versa in the output generated. To tackle this problem, the LSTM was inaugurated. There is not much difference between GRU and LSTM. But GRU has a simpler network cell architecture as shown in Fig.10 as compared to that of LSTM. Hence, the GRU is used in this caption generator model.

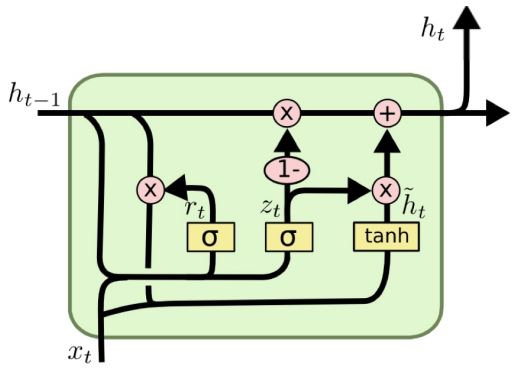


Figure 10: GRU

Fig.10 displays the working principle of GRU with a diagram. Here, x_t is an input vector, z_t is an update gate vector, h_{t-1} is a previous output, h_t is the current output, r_t is the reset gate vector and $\tilde{h_1}$ is an activation vector. Sigma (σ) represents the sigmoid activation function and *tanh* represents the tan hyperbolic operation.

Firstly, the update gate vector z_t is calculated for time step t using (1).

$$z_t = \sigma(Weight_{input_{update}} * X_t + Weight_{hidden_{update}} * h_{t-1}) \tag{1}$$

The input vector x_t and previous vector h_{t-1} are multiplied with their respective own weights viz. $Weight_{input_{update}}$ and $Weight_{hidden_{update}}$. The obtained products are added and the sum is squashed between 0 and 1 using the sigmoid activation function. Update gate enables the model to decide how much of the previous content needs to be sent to the future.
Then, the forget gate vector (2) r_t is calculated using the same formula as used in (1). This gate helps the model to decide the quantity of the past information that needs to be forgotten.

$$r_t = \sigma(Weight_{input_{reset}} * X_t + Weight_{hidden_{reset}} * h_{t-1}) \tag{2}$$

For the current memory content, the input x_t is multiplied with a weight W_{h_1} and h_{t-1} is multiplied with a weight W_{X1}. Then, the Hadamard (element-wise) product is calculated is between the reset gate r_t and $W_{h_1}h_{t-1}$. All these are added and a nonlinear activation function $tanh$ is applied as shown in (3).

$$\tilde{h_t} = \tanh(r_t \odot W_{h_1} * h_{t-1} + W_{X1} * X_t) \tag{3}$$

For the final memory content at current time step, element-wise multiplication is applied to the update gate z_t and h_{t-1}, and $1 - z_t$ and h_t. Then, these two products are added, as shown in (4).

$$h_t = (1 - z_t) \odot h_t + z_t \odot h_{t-1} \tag{4}$$

The implementation of ABICG has been discussed in next section.

3 IMPLEMENTATION

```
import string
rem_punct = str.maketrans('', '', string.punctuation)

for r in range(len(annotations)) :
    line = annotations[r].split()
    line = [word.lower() for word in line]

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

    annotations[r] = ' '.join(line)
```

Figure 11: Preprocessing

Performing caption preprocessing, converting all words of each caption to lowercase. Removing punctuation and any alpha-numeric values from each caption has been shown in above snippet.

```
#add the <start> & <end> token to all captions
annotations = ['<start>' + ' ' + line + ' ' + '<end>' for line in annotations]
```

Figure 12: Adding Tags

Adding $< start >$ and $< end >$ tag to each captions.

```
top_word_cnt = 5000

tokenizer = Tokenizer(num_words = top_word_cnt+1, filters= '!"#$%^&*()_+.,:;-?/~`{}[]|\=@ ', lower = True, char_level = False,
                      oov_token = 'UNK') #defining out-of-vocabulary word as 'UNK'

tokenizer.fit_on_texts(annotations)
train_seqs = tokenizer.texts_to_sequences(annotations) #transform each text into a sequence of integers
```

Figure 13: Tokenization

Performing tokenizations, considering only top 5000 words for memory efficiency and making rest word as $< unk >$ tag. This gives us vocabulary.

```
# we add PAD token for zero

tokenizer.word_index['PAD'] = 0
tokenizer.index_word[0] = 'PAD'
```

Figure 14: Padding

Considering 0 index as $< pad >$ tag and giving $< pad >$ tag a zero value, i.e. zero padding value.

```
# Inputs should be in similar in shape and size
# Padding each vector to the max_length of the captions

train_seqs_len = [len(seq) for seq in train_seqs] # store the length of all lists
longest_word_length = max(train_seqs_len)         # store elements from list with maximum length
cap_vector= tf.keras.preprocessing.sequence.pad_sequences(train_seqs , padding= 'post', maxlen = longest_word_length, dtype='int

cap_vector[:5]
```

Figure 15: Zero Padding

Performing zero padding at the end of each captions so that all captions are of equal shape and size.

```
image_model = tf.keras.applications.InceptionV3(include_top=False, weights='imagenet')

input_layer = image_model.input #to get the input of the image_model
hidden_layer = image_model.layers[-1].output  #to get the output of the image_model

image_features_extract_model = tf.compat.v1.keras.Model(input_layer, hidden_layer) #build the final model using
```

```
class Encoder(Model):
    def __init__(self,embed_dim):
        super(Encoder, self).__init__()
        self.dense = tf.keras.layers.Dense(embed_dim) #build your Dense layer with relu activation

    def call(self, features):
        features =  self.dense(features) # extract the features from the image shape: (batch, 8*8, embed_dim)
        features =  tf.keras.activations.relu(features, alpha=0.01, max_value=None, threshold=0.0)
        return features
```

Figure 16: Encoder

Designing encoder using the InceptionV3 architecture for generating images features.

```
class Attention_model(Model):
    def __init__(self, units):
        super(Attention_model, self).__init__()
        self.W1 = tf.keras.layers.Dense(units) #build your Dense layer
        self.W2 = tf.keras.layers.Dense(units) #build your Dense layer
        self.V = tf.keras.layers.Dense(1) #build your final Dense layer with unit 1
        self.units=units

    def call(self, features, hidden):
        hidden_with_time_axis = hidden[:, tf.newaxis]
        score = tf.keras.activations.tanh(self.W1(features) + self.W2(hidden_with_time_axis))
        attention_weights = tf.keras.activations.softmax(self.V(score), axis=1)
        context_vector = attention_weights * features
        context_vector = tf.reduce_sum(context_vector, axis=1)
        return context_vector, attention_weights
```

Figure 17: Attention

Designing the attention model which calculates attention score for each decoder timestamp and apply softmax to generate attention weights or attention distributions and getting the context vector form attention weights and features form encoder.

```
class Decoder(Model):
    def __init__(self, embed_dim, units, vocab_size):
        super(Decoder, self).__init__()
        self.units=units
        self.attention = Attention_model(self.units) #iniitalise your Attention model with units
        self.embed = tf.keras.layers.Embedding(vocab_size, embed_dim) #build your Embedding layer
        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) #build your Dense layer
        self.d2 = tf.keras.layers.Dense(vocab_size) #build your Dense layer


    def call(self,x,features, hidden):
        context_vector, attention_weights = self.attention(features, hidden) #create your context vector & attention weights from
        embed = self.embed(x) # embed your input to shape: (batch_size, 1, embedding_dim)
        embed = tf.concat([tf.expand_dims(context_vector, 1), embed], axis = -1) # Concatenate your input with the context vector
        output,state = self.gru(embed) # Extract the output & hidden state from GRU layer. Output shape : (batch_size, max_length
        output = self.d1(output)
        output = tf.reshape(output, (-1, output.shape[2])) # shape : (batch_size * max_length, hidden_size)
        output = self.d2(output) # shape : (batch_size * max_length, vocab_size)

        return output, state, attention_weights

    def init_state(self, batch_size):
        return tf.zeros((batch_size, self.units))
```

Figure 18: Decoder

```
def pred_caption(random, autoplay=False, weights=(0.5, 0.5, 0, 0)) :
    cap_test_data = caption_test.copy()
    rid = np.random.randint(0, random)
    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)

    real_caption = filt_text(real_caption)
    pred_caption = ' '.join(result).rsplit(' ', 1)[0]

    real_appn = []
    real_appn.append(real_caption.split())
    reference = real_appn
    candidate = pred_caption.split()

    score = sentence_bleu(reference, candidate, weights = weights) #set your weights
    print(f"BLEU score: {score*100}")
    print ('Real Caption:', real_caption)
    print ('Prediction Caption:', pred_caption)
    plot_attention_map(result, attention_plot, test_image)

    return test_image
```

Figure 19: Prediction

Function to predict captions, it displays the BLUE score, and compares the real human generated caption with machine generated caption.

4 RESULTS AND DISCUSSION

The following are the results obtained on the trained model:

Figure 20: "two girls hanging upside down on monkey-bars at a park"

The above image is fed to the model and the caption generated is "two girls hanging upside down on monkey-bars at a park" in comparison with the human generated annotation – "two girls are hanging upside down".

Figure 21: "large bird swooping down towards the ground"

The above image is fed to the model and the generated caption is "large bird swooping down towards the ground" in comparison with the human generated annotation – "a white bird swooping down the ground".

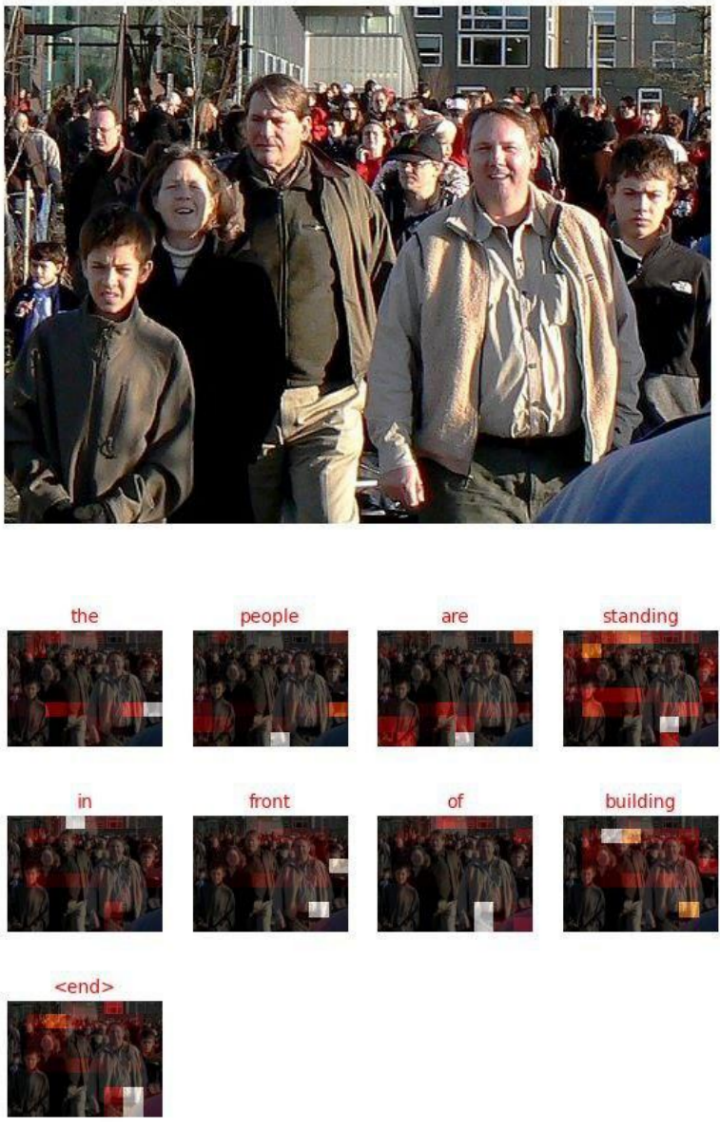


Figure 22: "The people are standing in front of building"

The above image is fed to the model and the caption generated is "the people are standing in front of the building" in comparison with the human generated annotation – "the people are standing before a building".

4.1 Graph of Loss vs Epoch for ABICG model

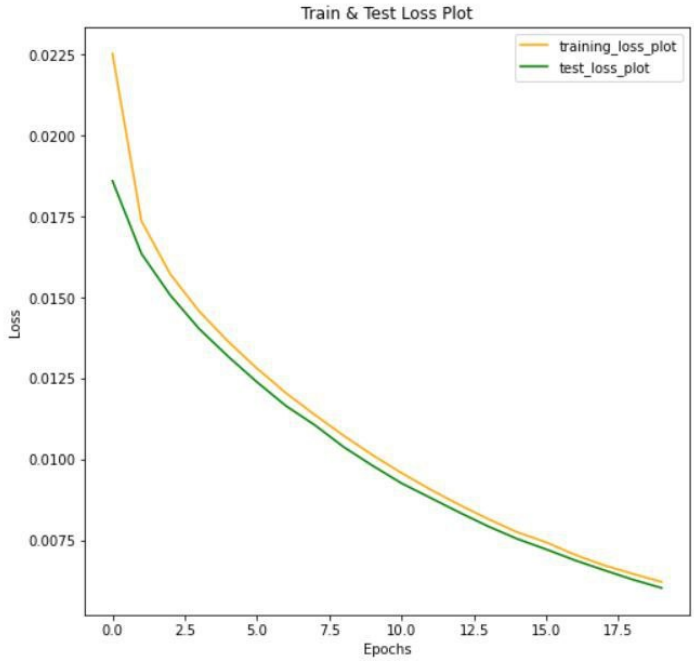
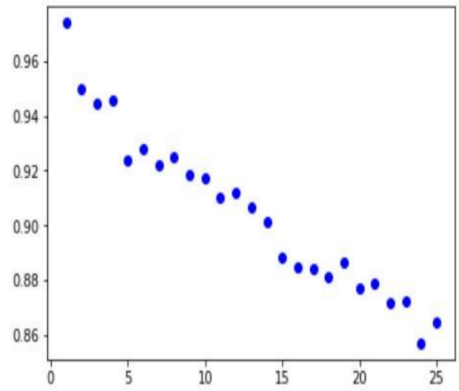


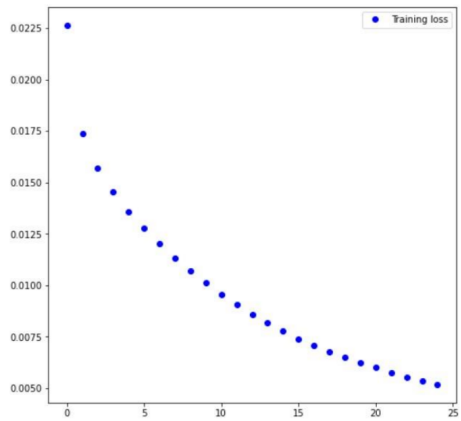
Figure 23: Train-Test loss vs Epoch for ABICG Model

4.2 Comparison of Training Losses

'Train Loss VS Epoch' graph was plotted for 25 epochs on both Traditional InceptionV3-GRU model (without attention) and InceptionV3-GRU model with attention (ABICG model). From the below comparison, it is evident that the train loss is higher in the InceptionV3-GRU model without attention [33] (Loss = 0.8647) as compared to the InceptionV3-GRU model with attention (ABICG model) (Loss = 0.0050).



24.1 InceptionV3-GRU (Without Attention): Train Loss= 0.8647



24.2 InceptionV3-GRU ABICG model(With Attention) : Train Loss=0.0050

Figure 24: Comparision of train losses

4.3 Metrics

The volumetric data hinders the way to show the result for each image. Hence, it becomes essential to look for a method to assess the system's average accuracy on the entire dataset. There are multiple ways to evaluate the quality of machine-generated text. For this model, the Bilingual Evaluation Understudy (BLEU)[**b38**] has been chosen as the evaluation metric, owing to its popularity and ease of usage. Before giving introduction to BLEU, knowledge about 'precision', a simpler and more well-known metric is customary. Let machine-generated n-grams and ground truth n-grams be denoted by the vectors x and y respectively. For instance, x could be taken as the words of a caption generated from an image, with x_i representing an individual word, and y could be the words from actual captions describing the same scene. It is expected always to denote the several possible captions of a single idea.

$$p = \frac{1}{N} \sum_{i=1}^{N} 1\{x_i \in y\}$$

The BLEU score and precision are equivalent, except the fact that there can only be a single instance of an n-gram in x for every incidence of an n-gram in y. Say, the statement "is is is is is" would receive an absolute precision if the word 'is' was present in the reference translation, but not compulsorily a perfect BLEU score, as it limits to counting only the number of occurrences of 'is' as it appears in y.

For ABICG model, a BLEU score of 90% was obtained for the weights (0.75, 0.25, 0, 0) and (0.50, 0.25, 0, 0).

Model	Accuracy
InceptionV3-GRU	79%
InceptionV3-GRU with Attention	90%

Figure 25: Comparision of accuracies

From the above table, it is evident that the accuracy of the InceptionV3-GRU model with attention (ABICG model) is approximately 10% more than that of traditional InceptionV3-GRU model (without attention).

5 Conclusion

In this paper, the caption generator for any given input image is being proposed using the encoder decoder techniques. The novel attention mechanism is the prime focus of the paper. The attention mechanism which is introduced after the InceptionV3 layer of networks here, makes the model focus on the highlighted receptive fields in the image to facilitate the decoder to produce captions solely for those parts. This greatly hikes the performance or the process of spawning the captions as compared to the orthodox encoder-decoder models. Results fetched from the model are budding and generated captions are clear.

Since the model had been exposed to a confined training set and vocabulary, the model may be deficient in connecting the input images to those features or the characteristics which are not present in the vocabulary. So, words like these are replaced with $< UNK >$ tag which means that these are unknown to the model. The model might not

do well with such kinds of input images where the $< UNK >$ tag occurs. In such cases, the captions reproduced might be too trivial.

Future scope of this work includes the usage of the transformer based models instead of the existing Encoder-Decoder based models, with the multi-head attention coupled with the positional embedding helps render information regarding how the different words are related in correct order.

References

[1] V. Agrawal, S. Dhekane, N. Tuniya and V. Vyas, "Image Caption Generator Using Attention Mechanism," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2021, pp. 1-6, doi: 10.1109/ICCCNT51525.2021.9579967.

[2] S. Degadwala, D. Vyas, H. Biswas, U. Chakraborty and S. Saha, "Image Captioning Using Inception V3 Transfer Learning Model," 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021, pp. 1103-1108, doi: 10.1109/ICCES51350.2021.9489111.

[3] Amritkar, Chetan Jabade, Vaishali. (2018). Image Caption Generation Using Deep Learning Technique. 1-4. 10.1109/ICCUBEA.2018.8697360.

[4] C. S. Kanimozhiselvi, K. V, K. S. P and K. S, "Image Captioning Using Deep Learning," 2022 International Conference on Computer Communication and Informatics (ICCCI), 2022, pp. 1-7, doi: 10.1109/ICCCI54379.2022.9740788.

[5] Sharma, Grishma Kalena, Priyanka Malde, Nishi Nair, Aromal Parkar, Saurabh. (2019). Visual Image Caption Generator Using Deep Learning. SSRN Electronic Journal. 10.2139/ssrn.3368837.

[6] Manish Raypurkar, Abhishek Supe, Pratik Bhumkar, Pravin Borse, Dr. Shabnam Sayyad5(2021). Deep Learning Based Image Caption Generator.

[7] Aishwarya Maroju , Sneha Sri Doma, Lahari Chandarlapati, 2021, Image Caption Generating Deep Learning Model, INTERNATIONAL JOURNAL OF ENGINEER-

ING RESEARCH TECHNOLOGY (IJERT) Volume 10, Issue 09 (September 2021).

[8] M. Duan, J. Liu and S. Lv, "Encoder-decoder based multi-feature fusion model for image caption generation," Journal on Big Data, vol. 3, no.2, pp. 77–83, 2021

[9] Szegedy, Christian Vanhoucke, Vincent Ioffe, Sergey Shlens, Jon Wojna, ZB. (2016). Rethinking the Inception Architecture for Computer Vision. 10.1109/CVPR.2016.308.

[10] Cho, Kyunghyun Merrienboer, Bart Gulcehre, Caglar Bougares, Fethi Schwenk, Holger Bengio, Y.. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 10.3115/v1/D14-1179.

[11] Bahdanau, Dzmitry Cho, Kyunghyun Bengio, Y.. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. ArXiv. 1409.

[12] P. G. Shambharkar, P. Kumari, P. Yadav and R. Kumar, "Generating Caption for Image using Beam Search and Analyzation with Unsupervised Image Captioning Algorithm," 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 857-864, doi: 10.1109/ICICCS51141.2021.9432245.

[13] X. Zou, C. Lin, Y. Zhang and Q. Zhao, "To be an Artist: Automatic Generation on Food Image Aesthetic Captioning," 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), 2020, pp. 779-786, doi: 10.1109/ICTAI50040.2020.001

[14] P. G. Shambharkar, P. Kumari, P. Yadav and R. Kumar, "Generating Caption for Image using Beam Search and Analyzation with Unsupervised Image Captioning Algorithm," 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 857-864, doi: 10.1109/ICICCS51141.2021.9432245.

[15] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[16] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[17] Chung, J., Gulcehre, C., Cho, K., Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[18] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2016.

[19] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[20] Chaitin, G.. (2013). Computing Machinery and Intelligence. Alan Turing: His Work and Impact. 551-621. 10.1016/B978-0-12-386980-7.50023-X.

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

[22] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[23] J., Weizenbaum. 1966. ELIZA—a computer program for the study of natural language communication between man and machine. Commun. ACM 9, 1 (Jan. 1966), 36–45. https://doi.org/10.1145/365153.365168

[24] Liu, S., Bai, L., Hu, Y., Wang, H. (2018). Image captioning based on deep neural networks. In MATEC Web of Conferences (Vol. 232, p. 01052). EDP Sciences.

[25] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

[26] Mo, Nan Yan, Li Zhu, Ruixi Xie, Hong. (2019). Class-Specific Anchor Based and Context-Guided Multi-Class Object Detection in High Resolution Remote Sensing Imagery with a Convolutional Neural Network. Remote Sensing. 11. 272. 10.3390/rs11030272.

[27] Abadi, Martín Barham, Paul Chen, Jianmin Chen, Zhifeng Davis, Andy Dean, Jeffrey Devin, Matthieu Ghemawat, Sanjay Irving, Geoffrey Isard, Michael Kudlur, Manjunath Levenberg, Josh Monga, Rajat Moore, Sherry Murray, Derek Steiner, Benoit Tucker, Paul Vasudevan, Vijay Warden, Pete Zhang, Xiaoqiang. (2016). TensorFlow: A system for large-scale machine learning.

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

[29] Graves, Alex Fernández, Santiago Schmidhuber, Jürgen. (2005). Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition.. 799-804.

[30] Ren, Shaoqing He, Kaiming Girshick, Ross Sun, Jian. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39. 10.1109/TPAMI.2016.2577031.

[31] Hubel, David Wiesel, Torsten. (2012). David Hubel and Torsten Wiesel. Neuron. 75. 182-4. 10.1016/j.neuron.2012.07.002.

[32] Papineni, Kishore Roukos, Salim Ward, Todd Zhu, Wei Jing. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. 10.3115/1073083.1073135.

[33] Hyunju1 (2018) Image-Captioning [Source code] `https://github.com/HyunJu1/Image-Captioning`