

IMAGE CAPTIONING USING TRANSFORMER ARCHITECTURE

by

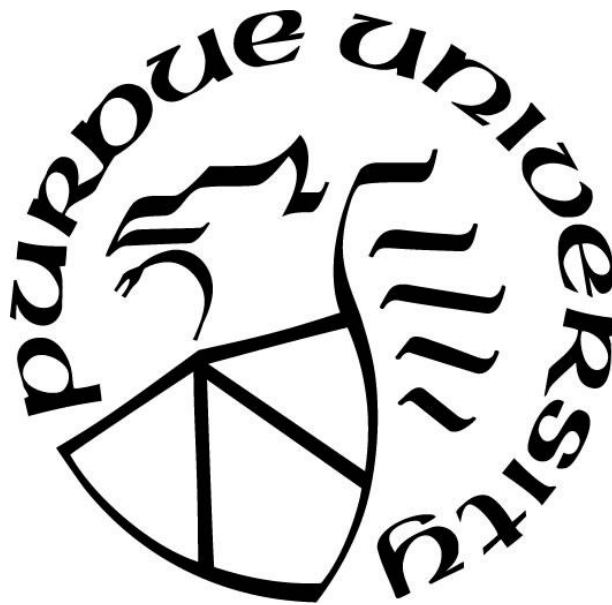
Wrucha Ameet Nanal

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science



Department of Electrical and Computer Engineering

Fort Wayne, Indiana

December 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Mohommadreza Hajiababi

Assistant Professor of computer science

Dr. Venkata Inokollu

Assistant Professor of computer science

Dr. Jin Soung Yoo

Professor of computer science

Approved by:

Dr. Jin Soung Yoo

Dedicated to my parents, Savita & Yeshwant Andurkar for supporting my dreams and aspirations. To my husband, Ameet Nanal for all the love and encouragement. And thank you to my in-laws Shubhada & Nitin Nanal for taking away my worries so I could dedicate my time here.

ACKNOWLEDGMENTS

I would like to thank my guide, Dr M Hajiarbabi for letting me explore the topics of my choice, help me out when I was stuck and supporting me throughout the semesters. I would like to thank my department for providing a supportive academic environment.

TABLE OF CONTENTS

LIST OF TABLES	7
LIST OF FIGURES	8
LIST OF EQUATIONS.....	9
LIST OF ABBREVIATION.....	11
ABSTRACT	12
1. INTRODUCTION	13
2. LITERATURE SURVEY	15
2.1 Generating text snippets for parts of images	15
2.2 A CNN-LSTM based Image Captioning Model	18
2.3 Attention based Image Captioning Model.....	19
3. DATASET	22
4. TRANSFORMER	24
4.1 Multi Head Attention Layer.....	25
4.2 Add & Norm Layer	28
4.3 Feed Forward Layer	29
4.4 Positional Encoding.....	29
4.5 Encoder.....	29
4.6 Decoder.....	31
4.7 Linear & Softmax Layer.....	32
5. CONTEXTUALLY AWARE TRANSFORMER BASED IMAGE CAPTIONING MODE.....	33
5.1 Encoder Input Preparation – Image Feature Extraction using Resnet 50.....	33
5.2 Decoder Input Preparation – Text Feature Extraction using BERT.....	34
5.3 Contextually Aware Transformer based Image Captioning Model	37
6. EXPERIMENT & EVALUATION	41
6.1 Experimental Setup	41
6.2 Evaluation Techniques	44
6.2.1 BLEU Score	45
6.2.2 METEOR Score	47
6.2.3 ROGUE Score.....	47

6.3 CATIC Results	48
7. DISCUSSION & FUTURE WORK	57
8. CONCLUSION.....	59
REFERENCES	60

LIST OF TABLES

Table 3.1 Number of images with unique caption count	23
Table 6.1 Experiment Parameters	41
Table 6.2 Test and Train Dataset after clenaing	42
Table 6.3 Performance comparison between CATIC and CNN-LSTM based model.....	48
Table 6.4 Performance comparison between CATIC and Attention based Image Captioning model	48
Table 6.5 CATIC Results for various categories	53

LIST OF FIGURES

Figure 2.1 Identified objects in the image are sorrounded by bounding boxes with a text description attached to them [8].....	15
Figure 2.2 Representing Image Caption model using multimodal RNN [8].....	17
Figure 2.3 LSTM model combined with CNN image encoder [9]	19
Figure 2.4 High level attention based model [11]	20
Figure 3.1 Sample of the types of images in the dataset along with one of their captions [12]	22
Figure 3.2 Example of the image and their captions from RSICD dataset [12]	23
Figure 4.1 Transformer Architecture [4]	24
Figure 4.2 Scaled Dot Product Attention [4]	26
Figure 4.3 Multi Head Attention is a parallelelized Scaled Dot Product Attention [4]	27
Figure 4.4 Process of obatinig Z matrix. Z become input to the next Transformer steps	28
Figure 4.5 Encoder Structure. Encoder implements Self Attention as $Q=K=V$	30
Figure 4.6 Decoder Stack and Internal Structure. Decoder implements two kinds of Multihead Attention layers.....	31
Figure 4.7 Working of Masked Self-Attention [4]	32
Figure 5.1 Skip connection shown in Resnet-50 architecture [16]	34
Figure 5.2 Language Feature Extraction using Hugginface	36
Figure 5.3 Contextually Aware Transformer based Image Captioning Model	37
Figure 5.4 (a) Internal representation of 1 layer of encoder & decoder (b) Shows internal representation for 4-layer encoder & decoder Transformer	40
Figure 6.1 Leaning Rate vs Train Step graph.....	43
Figure 6.2 Confusion Matrix with TN, TP,FN and FP	44
Figure 6.3 Best captions predictions by CATIC.....	50
Figure 6.4 Average captions predicted by CATIC	51
Figure 6.5 Poor captions predicted by CATIC	52

LIST OF EQUATIONS

Equation 2.3.1 Calculate the score between image k and sentence l	16
Equation 3.2.2 Get context vector of image I using CNN	17
Equation 2.3.3 Calculate the hidden state of RNN	17
Equation 2.3.4 Calculate unnormalized log probability of predicted word	17
Equation 2.3.5 Hidden value of LSTM block at $t=t$	19
Equation 2.3.6 Probability distribution over vocabulary	19
Equation 2.3.7 Equation to calculate weight for each annotation vector	20
Equation 2.3.8 Equation to calculate context vector.....	21
Equation 4.1 Formula for calculating Attention	26
Equation 4.2 Calculate the Multihead Attention with Q , K and V values	28
Equation 4.3 Mathematical representation of Add & Norm layer.....	28
Equation 4.4 Mathematical representation of Feed Forward Layer	29
Equation 4.5 Formula for calculating positional encoding for even and odd position of words in the sentence.....	29
Equation 5.1 Set of equations to create input for skip connection	33
Equation 5.2 Equation to calculate the skip connection in Resnet-50	34
Equation 5.3 Extract Image Features	38
Equation 5.4 Equation for first Add & Norm layer of Encoder	38
Equation 5.5 Equation for Add & Norm layer of Encoder	38
Equation 5.6 Equation for creating decoder input	39
Equation 5.7 Equation for first Add & Norm layer of decoder	39
Equation 5.8 Equation for second Add & Norm layer of decoder	39
Equation 5.9 Equation for third Add & Norm layer of decoder	39
Equation 6.1 Formula for calculating Sparse Categorical Cross Entropy.....	42
Equation 6.2 Formula for calculating learning rate during the training.....	43
Equation 6.3 Formula for Precision	44
Equation 6.4 Formula for Recall.....	45
Equation 6.5 Formula for F1-Score	45

Equation 6.6 Precision for n-gram	46
Equation 6.7 Precision for n-gram	46
Equation 6.8 Brevity Penalty	46
Equation 6.9 Final BLEU(N) score formula	46
Equation 6.10 Formula for METEOR Score calculation	47
Equation 6.11 Formula for ROGUE_L F1 Score calculation	48

LIST OF ABBREVIATION

Abbreviation	Definition
IC	Image Captioning
CV	Computer Vision
NLP	Natural Language Processing
CNN	Convolution Neural Network
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
RCNN	Regional Convolution Neural Network
BRNN	Bi-directional Recurring Neural Network
RSICD	Remote Sensing Image Captioning Dataset
MHA	Multi Head Attention
CATIC	Contextually Aware Transformer based Image Captioning
BERT	Bidirectional Encoder Representation of Transformers
BLEU	Bilingual Evaluation Understudy
METEOR	Metric for Evaluation of Translation with Explicit ORdering
ROGUE	Recall Oriented Understudy for Gisting Evaluation
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

ABSTRACT

The domain of Deep Learning that is related to generation of textual description of images is called ‘Image Captioning.’ The central idea behind Image Captioning is to identify key features of an image and create meaningful sentences that describe the image. The current popular models include image captioning using Convolution Neural Network - Long Short-Term Memory (CNN-LSTM) based models and Attention based models. This research work first identifies the drawbacks of existing image captioning models namely – sequential style of execution, vanishing gradient problem and lack of context during training.

This work aims at resolving the discovered problems by creating a Contextually Aware Image Captioning (CATIC) Model. The Transformer architecture, which solves the issues of vanishing gradients and sequential execution, forms the basis of the suggested model. In order to inject the contextualized embeddings of the caption sentences, this work uses Bidirectional Encoder Representation of Transformers (BERT). This work uses Remote Sensing Image Captioning Dataset. The results of the CATIC model are evaluated using BLEU, METEOR and ROGUE scores. On comparison the proposed model outperforms the CNN-LSTM model in all metrics. When compared to the Attention based model’s metrics, the CATIC model outperforms for BLEU2 and ROGUE metrics and gives competitive results for others.

1. INTRODUCTION

When humans look at an image, they recognize the objects and the spatial relationship between them. After that humans create a description of that image in a natural language. This is easy for humans; However, this process involves several steps when it is being implemented for machines. The stream of Deep Learning that implements the process of generating image descriptions is called ‘Image Captioning’ (IC). Due to the utilization of two top-tier domains, Computer Vision (CV) and Natural Language Processing (NLP), IC is a cross-field implementation. The model's CV side is used to pinpoint an image's features and interpret them as feature maps or vectors. The NLP side focusses on the generating description words. IC has gained popularity as its application spans across various fields such as captioning ariel images, social media image captioning and classification, medical image description generation, etc.

At present an end-to-end IC model has two sections- one that extracts image features and the second that will generate a caption using those features. To extract the image features, a pretrained Convolution Neural Network (CNN) [1] is employed. These features are then passed through a Recurrent Neural Network (RNN) [2] based models such as Long Short-Term Memory (LSTM) [3] which help generate a caption.

However, this research work was able to identify certain disadvantages in present day models. First, the existing models performed a series of RNN executions to generate an ‘ n ’ word sentence. RNN computations are memory expensive and sequential in nature. Secondly, the existing models trained using tokenized word formats. Location of words and their relative position to each provide a sematic meaning to a sentence. The existing models lacked attention to context.

This work is devoted to the creation of IC models using Transformer architecture which try to alleviate the problems faced by the present-day standard models. Objectives of this thesis outlined as follows –

- Create an IC model using Transformer Architecture
- Retain context of words while training

- Compare proposed model’s performance with popular IC models

The proposed model is trained on Remote Sensing Image Captioning Dataset [13] and the resultant captions are assessed using standard NLP evaluations methodologies such as BLEU [5], METEOR [6] and ROGUE [7] scores. Based on the aforementioned scoring methods, the performance of the proposed model is also contrasted with the CNN-LSTM and Attention-based CNN-LSTM models on the same dataset.

This paper has been split into following sections: section 2 discusses the literature survey conducted for the domain. Section 3 highlights the dataset and gives details about it. Section 4 highlights the Transformer architecture and Section 5 introduces the ‘Contextually Aware Transformer based Image Captioning’ (CATIC) model. Section 6 contains the details of the experimental setup and investigates the results of the CATIC model. Finally, section 7 and 8 include discussion and conclusion respectively.

2. LITERATURE SURVEY

In this section of the paper, we will go over the published works that have inspired the development of Contextually Aware Transformer Based Image Captioning model. This section is divided into 3 parts. Each subsection describes an IC model and their approach. Finally, the drawbacks of the popular models are identified.

2.1 Generating text snippets for parts of images

In Deep Visual-Semantic Alignments for Generating Image Descriptions [8] we see that the authors have tried to implement IC in two parts. In the first part, the model identified objects in the image and attached a text snippet describing that object. For the second part, the model extracted image features into a feature map using a pretrained CNN [1] and provided them to an RNN [2] which predicted the caption word by word.



Figure 2.1 Identified objects in the image are surrounded by bounding boxes with a text description attached to them [8]

In order to create a bounding box with text, the model created a shared multi modal embedding space for the two modalities (image and sentence). The objects of the image were identified using

Regional Convolution Neural Network (RCNN). The CNN used in this study was pre-trained on ImageNet [17] and fine-tuned on 200 ImageNet detection challenge classes. After that the model selected the top 19 detected locations and computed the pixels in each bounding box along with the whole image. Finally, the image was represented as a set of vectors of h (ranging between 1000-1600) dimension where h represented the size of the multimodal embedding space. The sentences were represented in the shared multimodal space using Bi-directional Recurring Neural Network (BRNN). A BRNN converted a sequence of N words into N h -dimensional vectors. The word vector representation used a 300-dimensional word2vec embedding. Next, the image and sentence embeddings were mapped in a shared h dimensional space. The authors introduced an image sentence score function that represented the correlation between the sentence-image pair. The words were said to have a confident support in the image if the score was high. The score for image k and sentence l is given by equation 2.1.

Equation 2.3.1 Calculate the score between image k and sentence l .

$$Skl = \sum_{t \in gl} \sum_{i \in gk} \max(0, v_i^T s_t)$$

The above equation interprets a dot product $v_i^T s_t$ between the vector of i^{th} region and the t^{th} word as a measure as a measure of similarity. If the RCNN created a bounding box in the i^{th} region of the image, then there would be a corresponding word(s) as well for that region. This model could ultimately be used for generating text snippets for a single bounding box.

In the second section of ‘Deep Visual-Semantic Alignments for Generating Image Descriptions’ [8] paper the authors describe how they have used the first multimodal model to train the second model which is to generate image snippets by taking an image as an input and giving description as the output.

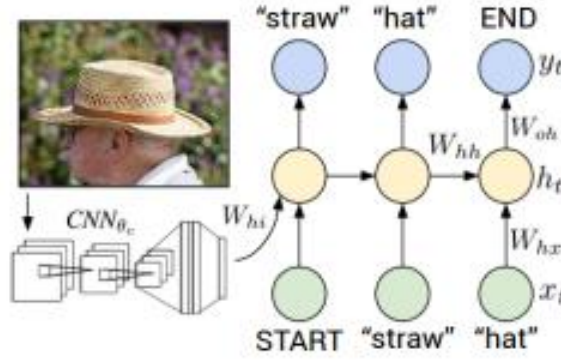


Figure 2.2 Representing Image Caption model using multimodal RNN [8]

For second model, the authors introduced a multimodal RNN. The context vector of an image was created using a CNN which would be fed to the RNN at $t=0$ as represented in Equation 2.2. The image was given to the RNN only once. Therefore, the first word would be predicted using image vector and ‘START’ token. The rest of the sentence would be generated token by token by using the word x_t and previous context (hidden state of RNN) $h_{(t-1)}$. The final output from each cycle in the RNN was the unnormalized log probabilities of the predicted words represented by equation 2.4. The following equations sum up the model steps.

Equation 3.2.2 Get context vector of image I using CNN

$$b_v = W_{hi}[CNN(I)]$$

Equation 2.3.3 Calculate the hidden state of RNN

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + 1(t=1) \odot b_v)$$

Equation 2.3.4 Calculate unnormalized log probability of predicted word

$$y_t = softmax(W_{oh}h_t + b_o)$$

where, b_v is the image vector, h_t are hidden states, y_t is the sequence of output and W_{hi} , W_{hx} , W_{hh} , W_{oh} , x_i and b_h , and b_0 are learnable parameters.

The experiments of this model were done on multiple datasets and the results showed that the region caption generation outperformed the full frame caption generator model. Therefore, this

model could not generate a full-fledged caption for an entire image instead it just described objects in the image with text snippets.

2.2 A CNN-LSTM based Image Captioning Model

A complete image captioning model was introduced for the first time in the paper titled ‘Show and Tell: Neural Image Caption Generator’ [9]. The authors of this paper introduced the ‘Encoder-Decoder’ style of IC model. The encoder encoded the image while the decoder predicted the words of the captions.

The image representations were created with help of CNN [1] which was pretrained on ImageNet. The output from the CNN [1] was adjusted such that it matched the embedding size of the decoder’s hidden state. The image and caption sentences were mapped into a shared embedding space. In order to represent the words in that space, the one hot encoded version of each word is multiplied by We which is the embedding matrix. The authors created decoder as a series of LSTM network.

Consider a sentence S of ‘ n ’ length. The tokens were converted into a one-hot encoded representation given as $(S_0, S_1, \dots, S_{n-1})$. To indicate the start and conclusion of a sentence at the LSTM, START and END tokens were inserted to the beginning and end of each sentence. Figure 2.3 shows a representation of the model. The encoded image was passed to the first LSTM block at $t=-1$. For the next block, the word embedding was passed to the LSTM [3] which computed the probability distribution of the words.

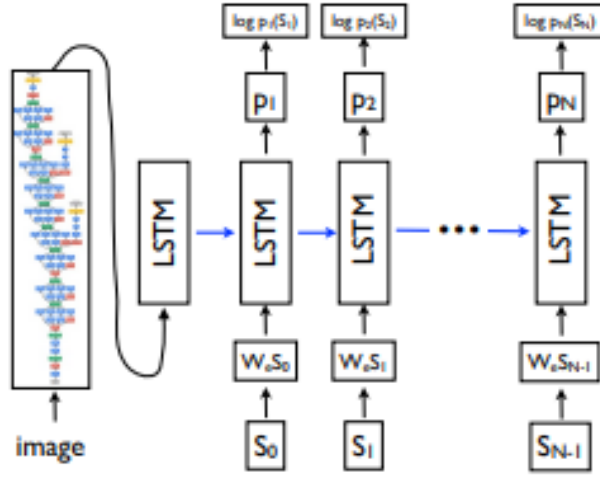


Figure 2.3 LSTM model combined with CNN image encoder [9]

Equation 2.5 describes the hidden state h_t calculation of the LSTM. Equation 2.6 calculates the final probability distribution p_t of the predicted words of the caption. W_o is the LSTM training parameter.

Equation 2.3.5 Hidden value of LSTM block at $t=t$

$$h_t = LSTM(W_e S_t, h_{t-1})$$

Equation 2.3.6 Probability distribution over vocabulary

$$p_t = W_o h_t$$

This models popularly known as the CNN-LSTM based model and will be referred to as such in the following sections. This model remains popular among developers. As seen above, the image features are used only once during the decoder's execution. Thus, the longer the predicted sentence, bleaker will be the presence of image features in the respective LSTM block.

2.3 Attention based Image Captioning Model

In order to alleviate the drawbacks of the CNN-LSTM based model, the author of 'Show, Attend and Tell: Neural Image Captioning with Visual Attention' [10] created an ingenious

solution by focusing on part of the image and used that part for predicting the sentence token. This model followed an ‘Encoder-Decoder’ model along with an attention module in pipeline.

Unlike previous models, the input image was first converted into a feature map using the lower convolution layers instead of the fully connected layers. These vectors were called annotation vectors. The annotation vectors were then processed by the decoder by passing them through an attention mechanism that generated a context vector z_t . After which the context vectors were passed through the LSTM in order to generate probability distribution of the predicted words. Figure 2.4 shows the high-level implementation of an Attention-based model.

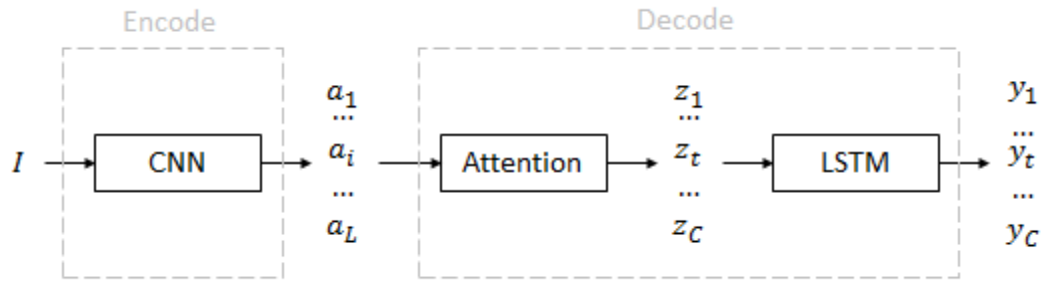


Figure 2.4 High level attention based model [11]

The central idea behind attention mechanism was understanding from previous state and deciding where should the model look next in order to predict the next word. The weight α_i for each annotation vector a_i depends on previous hidden state which is calculated with help of f_{att} as follows –

Equation 2.3.7 Equation to calculate weight for each annotation vector

$$e_{ti} = f_{att}(a_i, h_{t-1})$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^L \exp(e_{tk})}$$

Once the weights were calculated, the model calculated the ϕ function which in turn helped in calculating the z_t vector which is done as follows –

Equation 2.3.8 Equation to calculate context vector

$$z_t = \Phi(\{a_i\}, \{\alpha_i\})$$

The authors introduced two attention methods – deterministic method also referred to as soft attention and stochastic method also referred to as hard attention method. The 'where to look' in soft attention is determined by the weights assigned to different parts of the image based on previous state information. Hard attention, on the other hand, employs a sampling strategy to examine different parts of the image.

Finally, the LSTM generates the caption one word at a time using the context vector z , the previous hidden state h_{t-1} , and previously generated words. This model is considered state of the art as it produces some of the best results for image captioning use case. However, the major drawback is that a new context vector is required to select which part of image is most important for word prediction. Thus, the entire image is not attended by the decoder.

After understanding the industry standard works, we were able to identify some major areas where IC models could be improved. Firstly, the field of NLP has been revolutionized by the introduction of Transformers. Therefore, there was an absence of IC models that used the Transformers instead of LSTMs for NLP side. Secondly, the existing models performed a sequential execution to predict the caption words. Thirdly, the existing model did not utilize the entire image features during the decoder life-cycle which could lead to some important correlation between the image and word being overlooked. Finally, none of the existing models take relative or absolute position of the caption words, thereby not taking context into account while training or prediction.

3. DATASET

The dataset being used in this experiment was introduced in the ‘Exploring models and Data for Remote Sensing Image’ [12] paper. The authors created the dataset, known as the "Remote Sensing Picture Captioning Dataset" (RSICD) [13], in order to promote the discipline of remote sensing image captioning. The dataset consists of images which have been photographed from airplanes or satellites. Each image has 5 captions attached which describe the contents of the image. Following image shows the nature of images that are being considered.



Many planes are parked near several terminals in an airport with parking lots and runways.



A piece of green ocean is near yellow beach and many green trees.



A bridge is on a river with some green trees in two sides of it.

Figure 3.1 Sample of the types of images in the dataset along with one of their captions [12]

The images in this dataset are 224X224 in size. The dataset contains 10921 images. The authors have provided an 80:20 split of training and testing data. As a result, it was simple to create a training dataset and a testing dataset. Following figure 3.2 shows an example of image and its 5 captions.

Each caption in the dataset has been created such that all important features of the image are addressed in the captions. The authors have ensured that relative location words such as up,

down, above, below or nouns such as left, right etc. are not used for captions. The dataset offers 30 varieties of image classes such as beach, airports, rivers, stadiums, churches, parking etc.



1. An old court is surrounded by white houses.
2. A playground is surrounded by many trees and long buildings.
3. A playground with basketball fields next to it is surrounded by many green trees and buildings.
4. Many green trees and several long buildings are around a playground.
5. This narrow, oval football field and closing basketball court, tennis court, parking lot together form this area, with plants wreathing it.



1. Four planes are stopped on the open space between the parking lot.
2. Four white planes are between two white buildings.
3. Some cars and two buildings are near four planes.
4. Four planes are parked next to two buildings on an airport.
5. Four white planes are between two white buildings.

Figure 3.2 Example of the image and their captions from RSICD dataset [12]

Although all images are unique, it was observed that some images had repetitive captions. Table 3.1 below displays the number of images with distinct captions. Due to development environment restrictions, the dataset for experiment has been streamlined such that each image will have only unique captions.

Table 3.1 Number of images with unique caption count

Number of unique captions	Image count
5	724
4	1495
3	2182
2	1667
1	4853

The authors for ‘Exploring models and Data for Remote Sensing Image’ [12] have tried to create IC models with both CNN-LSTM and Attention-based methodologies and evaluated with help of various metrics. They have shown that RSICD can be effectively used for image captioning use case.

4. TRANSFORMER

Transformers were introduced in the paper – *Attention is all you need* [4]. It is backbone of some leading variations such as GPT-2 [14], BERT [15]. Transformer models are very popular and some of their applications include language translations models and question-answer based models. Due to their adaptable nature of the architecture, Transformers can be used for a wide range of use cases.

Figure 4.1 shows Transformer architecture. The Encoder stack of N identical layers is shown on the left side of the image. The right-side is the Decoder stack, which has N identical layers. Let us first understand the function of each subtask listed in the layers before we understand the purpose of each side of the transformer.

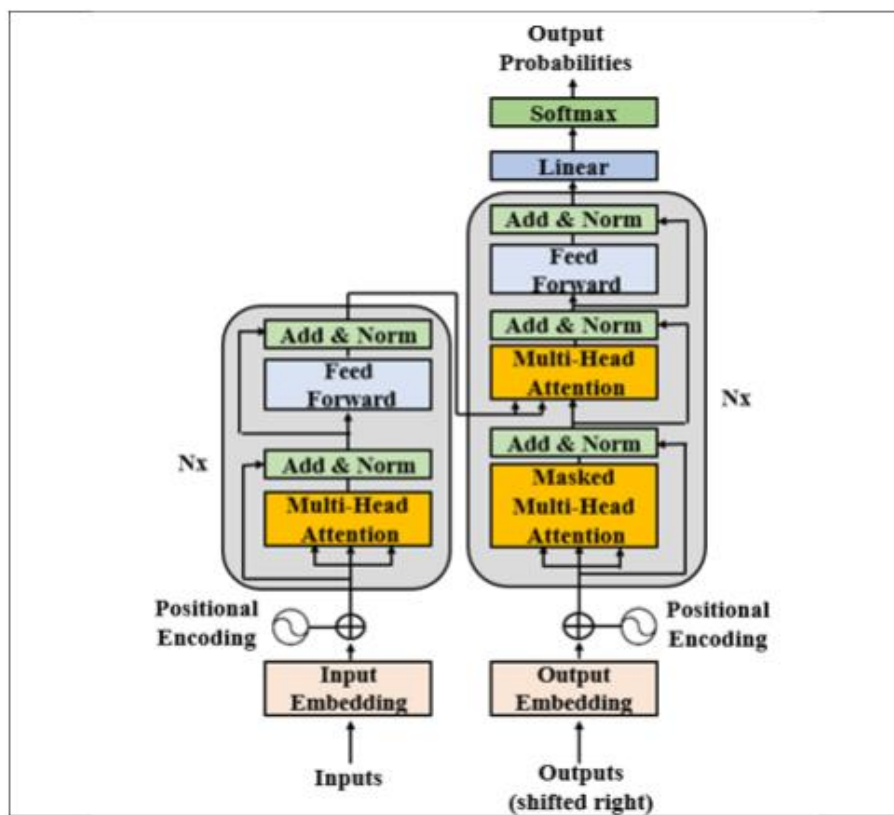


Figure 4.1 Transformer Architecture [4]

4.1 Multi Head Attention Layer

Let us consider a sentence – ‘The animal didn’t want to cross the street because it was tired.’ For a machine to associate ‘it’ with the animal the Transformer uses ‘Attention’ mechanism. Attention enables a Transformer to focus on other features from input that are related closely to the feature in focus therefore creating a relation between them.

In NLP, words must be transformed into vectors which by using embedding algorithms. For a sentence of n length, the vector would be of the size $n \times d_{model}$, where d_{model} is the embedding size of the features. These vectors are fed to the first layer of the encoder/decoder that performs a series of ‘Scaled-Dot Product Attention’ which is the ‘Multi Head Attention’ on them.

To understand purpose of Attention at high level we can say that attention enables a transformer to focus on one input features and evaluate how other features are closely related to the it. Thereby, creating a relation between them. Attention in Transformers is an implementation of ‘Scaled-Dot Product Attention.’ For each word of the sentence, we create Q, K, V which are Query, Key and Value vectors respectively. They can be defined as –

- Q – The vector whose value needs to be determined
- K – The vector that represents the features
- V – The vector for actual values of the input

Figure 4.2 depicts the implementation of the 'Scaled-Dot Product Attention' mechanism.

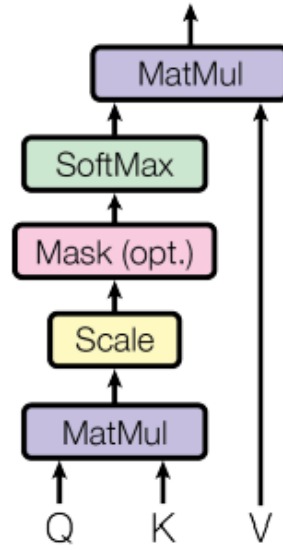


Figure 4.2 Scaled Dot Product Attention [4]

As per figure 4.2 the mechanism will firstly compute the dot product for the Q and K vectors. Secondly, the dot product is divided by $\sqrt{d_k}$ to alleviate the exploding gradient problem. Finally, the mechanism applies a softmax function to the normalized dot product. By doing so, weights used to scale V are obtained. We can condense Scaled-Dot Product Attention with following equation 3.1-

Equation 4.1 Formula for calculating Attention

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

A Multi Head Attention (MHA) layer is combination of ‘h’ Self Attention heads. Each head will calculate their Scaled Dot Product Attention. The results are concatenated which gives the value of the Multi Head Attention. The process is depicted in the diagram below.

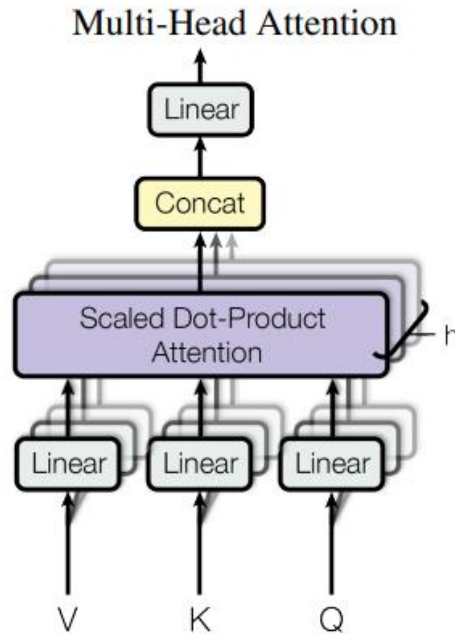


Figure 4.3 Multi Head Attention is a parallelized Scaled Dot Product Attention [4]

Using multiple heads for attention improves the performance of attentions layer. It allows the model to focus on different positions. It also gives multiple sets of weight matrices for Q , K and V . The attention calculated in each step is also known as '*z-score*' or '*z.*' Suppose we have 8 attention heads, then we will have $(z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7)$. These values cannot be given directly to the next layer of the encoder/decoder. They need to be transformed by concatenating all the results and multiplying it to W^O where W^O is the additional weights matrix that is trained jointly with the model.

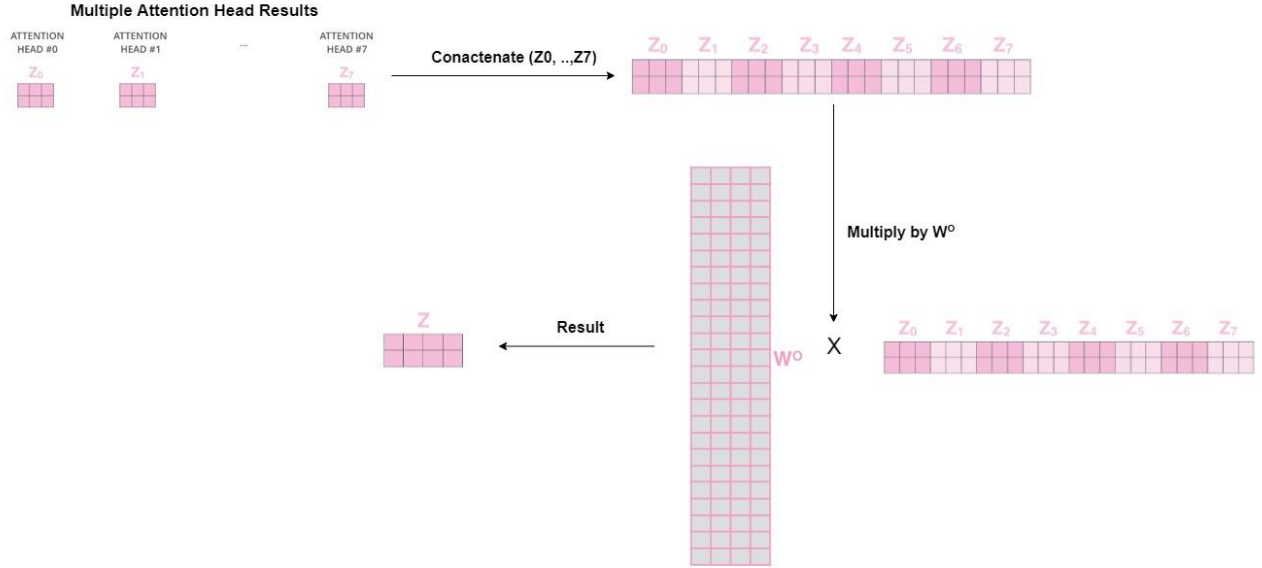


Figure 4.4 Process of obtaining Z matrix. Z become input to the next Transformer steps

We can mathematically define Multi Head Attention as follows.

Equation 4.2 Calculate the Multihead Attention with Q, K and V values

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \dots, head_h)W^O$$

$$where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

4.2 Add & Norm Layer

As the name of the layer suggests, it performs two operations. The first step is the ‘Add’ part that performs the flow control by Residual Connections. The second step is ‘Norm’ that performs layer normalization. Therefore, output of this layer would be as per the following equation.

Equation 4.3 Mathematical representation of Add & Norm layer

$$Add \& Norm = LayerNorm(x + Sublayer(x))$$

where x is input to the any sublayer (MHA or Feed Forward) and Sublayer(x) is its output.

4.3 Feed Forward Layer

Each layer has a point wise feed forward network which is a fully connected. It performs two linear transformations with ReLU activation. This layer calculates the weights during training. It can be mathematically be defined as follows –

Equation 4.4 Mathematical representation of Feed Forward Layer

$$FF(x) = ReLU(xW_1 + b_1)W_2 + b_2$$
$$ReLU(x) = \max(0, x)$$

where W_1 and W_2 are network weight matrix and b_1 and b_2 are biases.

4.4 Positional Encoding

The Transformers use Positional Encoding to inject the relative or absolute position of the embeddings into the model. This maintains token sequence in parallel execution format. The values for positional encoding are calculated using sine and cos representation of the position and training parameter. The positional encodings are added to the language features in order to create a position-aware embeddings. The formula for positional encoding proposed by the authors is as follows –

Equation 4.5 Formula for calculating positional encoding for even and odd position of words in the sentence

$$PE_{(pos, 2i)} = \sin(pos/1000^{2i/d_{model}})$$
$$PE_{(pos, 2i+1)} = \cos(pos/1000^{2i/d_{model}})$$

where i is the position of the embedding and d_{model} is the dimension of the embedding.

4.5 Encoder

Figure 4.5 shows that Encoder is stack of a Multi Head Attention layer, Add & Norm layer, a Feed-Forward layer, and a final Add & Norm layer which are repeated N times. The input to the encoder layer is in $n \times d_{model}$, where n are the number of features and d_{model} is the embedding

size of the features, to which positional encoding is added. This input is given only to the first layer of the encoder stack.

Following diagram shows an Encoder stack where $N = 4$. The input to the encoder is given such that $Q = K = V$. This case of attention mechanism where Q , K and V are equal is called ‘Self-Attention.’ Self-Attention creates richer representation of the input.

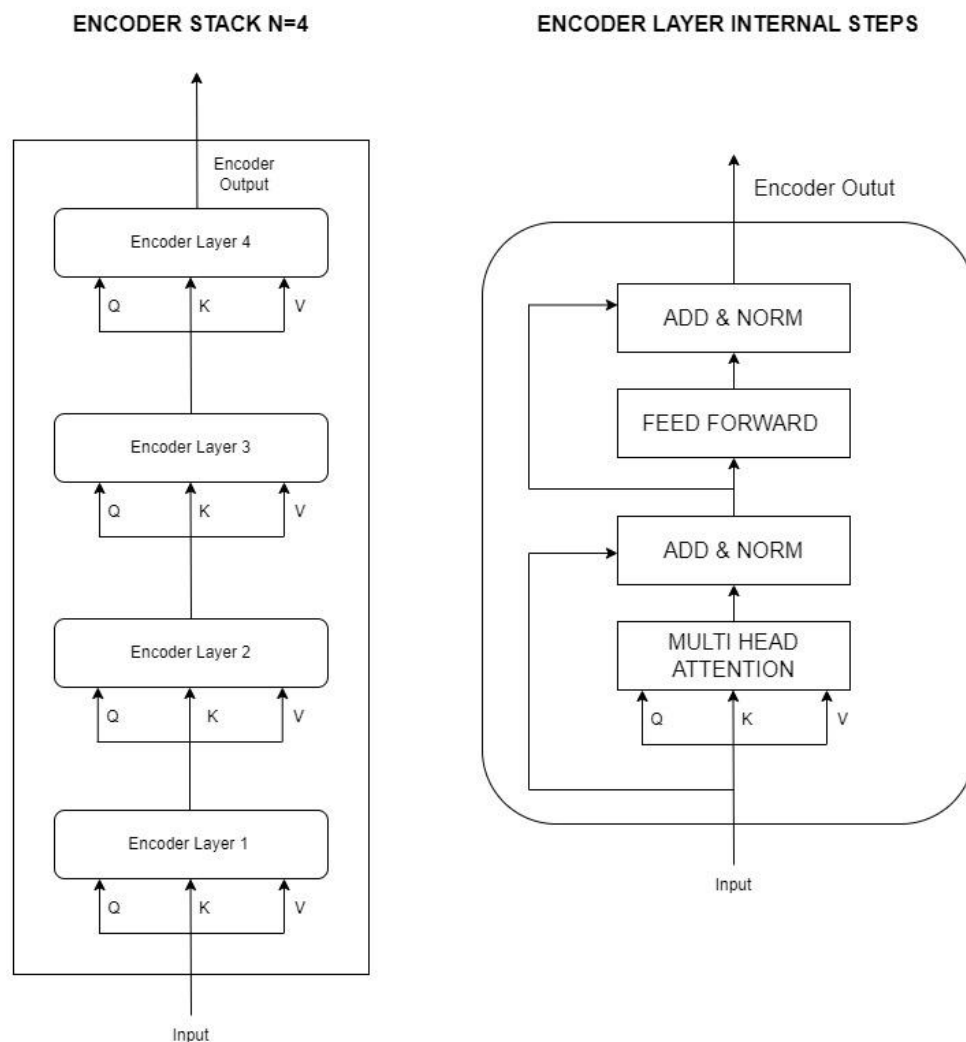


Figure 4.5 Encoder Structure. Encoder implements Self Attention as $Q=K=V$

4.6 Decoder

The right part of Figure 4.1 shows decoder part of the transformer which is also composed of N identical layers. The Decoder consists of 3 sublayers – Mask Self Attention layer, a multi head attention between ‘encoder output’ and output of Masked Self Attention layer and finally a Feed Forward Layer. Each of the sublayer is followed by an Add & Norm layer. The intermediate MHA layer of the decoder is also called Encoder-Decoder Attention Layer as it creates a relation between encoder output and decoder input. Following figure shows the overall structure of the decoder.

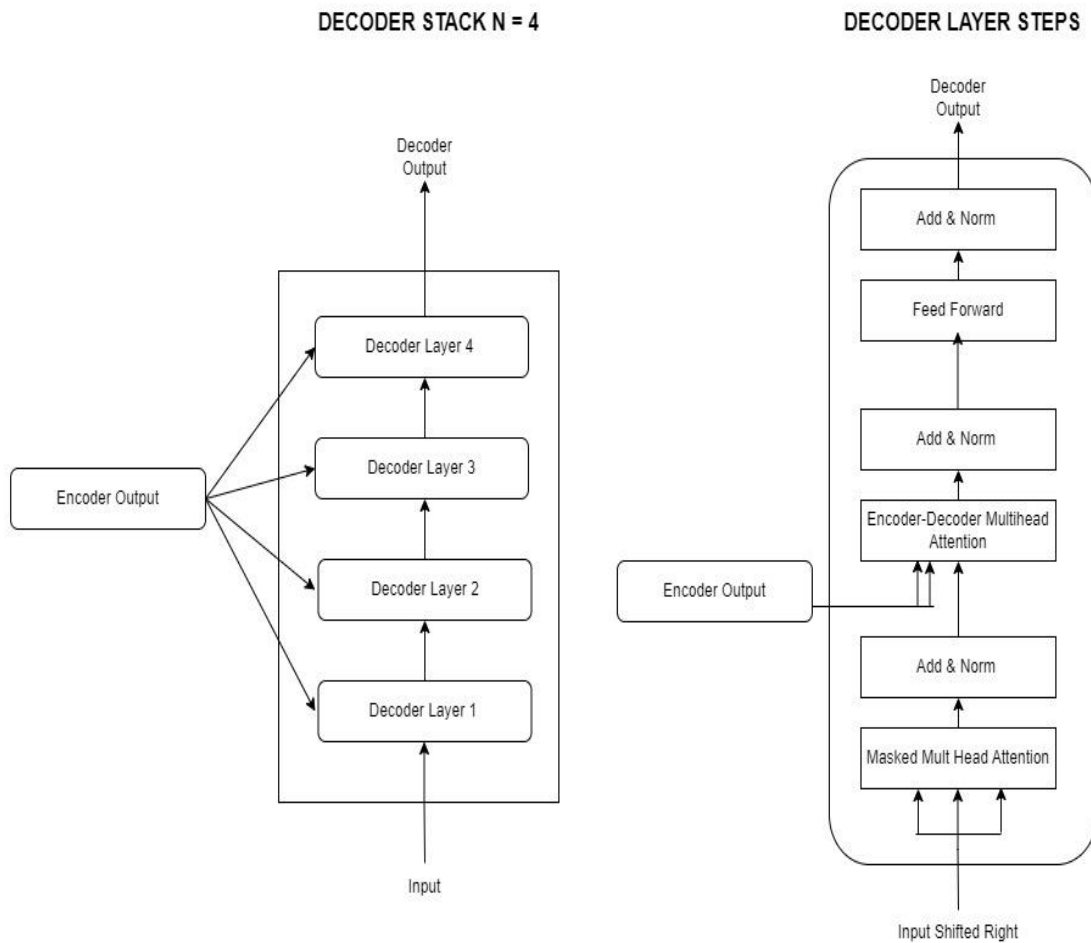


Figure 4.6 Decoder Stack and Internal Structure. Decoder implements two kinds of Multihead Attention layers

The first layer of decoder uses Masked Self Attention. During this time, Self-Attention can only attend to previous positions. The decoder input is shifted to the right and masked in such a

way that the word at i^{th} position can only attend to the words that have been before the position i . As all the computations are done in parallel, masking helps in preventing the decoder from peeking ahead and ‘cheating’ during the training. Following diagram shows how masking takes place. The future embeddings are hidden from the Self-Attention head. To implement masked self-attention a look ahead mask is created which is a lower triangular matrix.

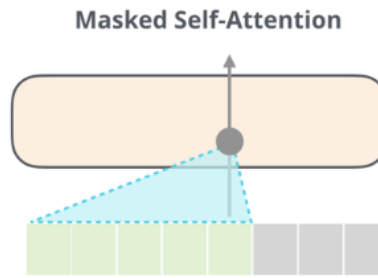


Figure 4.7 Working of Masked Self-Attention [4]

The second layer is a normal Multi Head Attention layer; however, it uses Encoder output as ‘K’ and ‘V’ along with output of Masked Self Attention layer as ‘Q.’ Hence the name of Encoder-Decoder MHA layer. This layer allows all positions of the decoder to attend to all positions of the input sequence. The final layer of this stack is the Feed Forward network.

4.7 Linear & Softmax Layer

This layer is like any other seq2seq model. The decoder output is passed through fully connected linear layer which projects the output of size $n \times vocab_size$ where n is the predicted sentence length and vocab size is the size of the language's vocabulary. The resultant matrix is then passed through a softmax layer which give a probability distribution over the vocabulary for each of the word in the output sentence.

5. CONTEXTUALLY AWARE TRANSFORMER BASED IMAGE CAPTIONING MODEL

The model that we are proposing in this thesis work is called ‘Contextually Aware Transformer Based Image Captioning’ (CATIC) model. This is a complete model that addresses the shortcomings of existing IC models.

As we saw in previous section the Transformer model does not enforce hard and fast rules for the type of use case for which it can be implemented. However, neither is an image ready to be ingested by the encoder part of transformer nor is the caption ready for decoder. Therefore, we need to modify the input in a more ingestible format. Following sections describe the input transformation for images, captions, and final CATIC model.

5.1 Encoder Input Preparation – Image Feature Extraction using Resnet 50

In any image captioning model, the image features need to be converted into a machine understandable format. The image features have been extracted using Resnet-50 for this use case. Resnet-50 [16] is a pretrained Deep Learning model for image classification. A pretrained Resnet-50 was trained on ImageNet [17] dataset which contains 1000 categories and over a million images. The model was trained to identify visual features of image and use them to classify the image. The reason for choosing Resnet-50 was its fewer error rates for recognition. A lower recognition rate is achieved because the ‘Skip Connection’ architecture implementation. Assume that in a regular network, if x_0 is the starting point then the neural network would work as follows-

Equation 5.1 Set of equations to create input for skip connection

$$\begin{aligned}z_1 &= w_1x_0 + b_1 \\x_1 &= ReLU(z_1) \\z_2 &= w_2x_1 + b_2 \\x_2 &= ReLU(z_2)\end{aligned}$$

Where let us assume that w_1 and w_2 are the weights and b_1 and b_2 are the biases. However, when we introduce the residual skip connection, the value of x_2 are updates as per following equation –

Equation 5.2 Equation to calculate the skip connection in Resnet-50

$$x_2 = \text{ReLU}(z_2 + x_0)$$

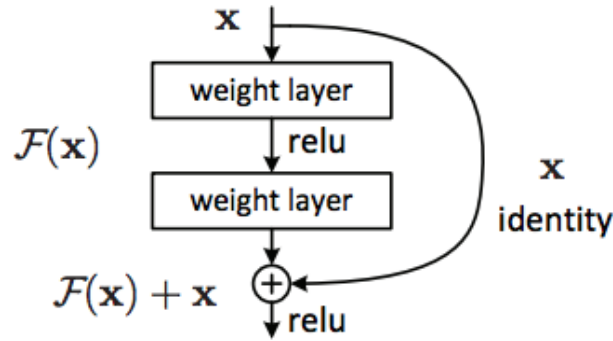


Figure 5.1 Skip connection shown in Resnet-50 architecture [16]

As we can see from the Figure 5.1, the network allows for the output of the previous layers to be added to the output of stacked layers. This addition does not have any extra parameters; however, it helps in alleviating the vanishing gradients and degrading accuracy problem. The CNN shrinks the image dimensions with each step. In Resnet50 there are 49 CNN layers followed by 1 fully connected layer. Unlike other models, the skip connection layer is introduced after three layers. As we are interested in extracting the features of the images, we take the outputs of the penultimate layer. Using the weight from the pretrained model, we extract features of the image in shape of (7, 7, 2048). The encoder side of the transformer expects the data to be in format of $n \times d_{model}$. After performing a 2D Convolution layer on the extracted features and reshaping them with d_{model} filter size the encoder input is ready.

5.2 Decoder Input Preparation – Text Feature Extraction using BERT

The input for decoder is captions in English language. A tokenized form of the caption words needs to be created. This can be accomplished using a variety of techniques, such word2Vec [18] and TF/IDF [19], although these pay no regard to the placement location of each word inside

the sentence. As mentioned above, the existing models do not incorporate context of the words. Relative positions of the words impact the meaning of a sentence. Consider an example – ‘There is plane ready on the runway.’ If we only rearrange the order of the words and recreate it as ‘There is runway ready on the plane,’ the semantic meaning behind the sentence completely changes. It is observed that the arrangement of words provides a different set of language features. None of the previous IC models paid attention to the relationship between the words of the sentence.

In order to create a contextually aware language features, CATIC uses Bidirectional Encoder Representation of Transformers [15] (BERT). BERT is a stacked transformer encoder implementation. The bidirectionality in BERT arises from the fact that it reads the entire word sequence at once unlike the other language models that take one word at a time. This bidirectionality helps the model to learn the context of the words based on their neighbors in a sentence. The authors introduced two models of BERT in their paper – BERT_{base} and BERT_{large} where the number of the layers involved in the model (12 and 24 respectively). Each layer has its own output which is passed to the next layer.

BERT is trained to perform two major tasks – Masked Language Model (MLM) and Next Sentence Prediction (NSP). In the MLM, BERT tries to find out the token that has been masked. In NSP, BERT takes a pair of sentences – sentence A and sentence and tries to find out if the B semantically follows A. Both the tasks are trained simultaneously. After pretraining BERT can be used for numerous fine-tuning tasks such as Sentence Pair classification, Question-Answering, Sentiment Analysis etc.

A pretrained BERT is also used for the extracting language features from the caption sentence and create contextual embeddings from a tokenized sentence. To do so, the authors of the BERT have recommended taking the sum of output of the last 4 hidden layers of a pretrained BERT. If the sentence is of length ‘ n ’ then each vector is of d_{model} size. Therefore, a sum of last 4 output layers gives rich content vectors of size $n \times d_{model}$. Huggingface is a library that has created pretrained BERT-base model which can be readily used for many uses cases. CATIC also uses Huggingface library to extract language features.

Following diagram depicts how the BERT Huggingface library has been used to extract the language features of the captions. Both the BERT tokenizer and pretrained BERT can be downloaded from the Huggingface interface [20]. The caption sentence is first tokenized using BERT tokenizer [20]. The tokenized sentence is passed through a pretrained BERT base model.

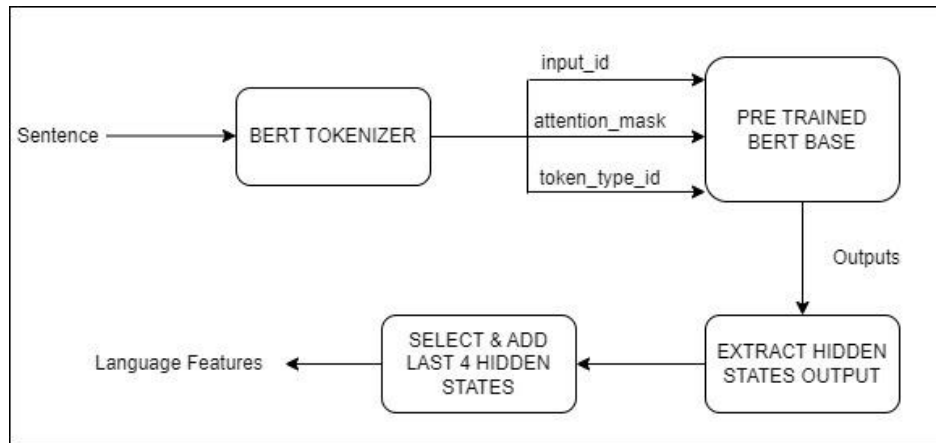


Figure 5.2 Language Feature Extraction using Huggingface

5.3 Contextually Aware Transformer based Image Captioning Model

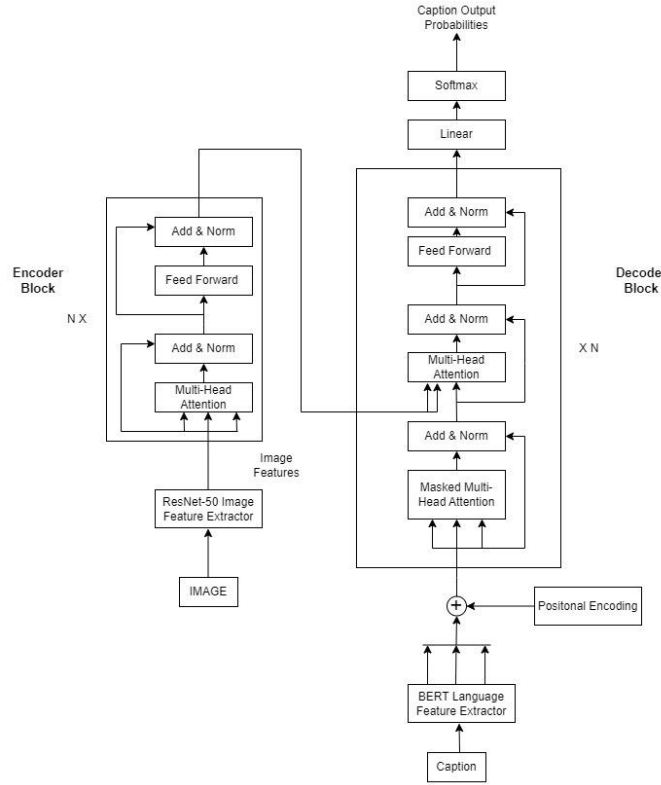


Figure 5.3 Contextually Aware Transformer based Image Captioning Model

Above Figure 5.3 gives model design for CATIC. The encoder block is shown on the left side of the figure, and the decoder block is shown on the right. Each one is a stack of N identical layers. Following steps briefly describe the steps of the model –

1. Extract image features using ResNet-50.
2. Feed the image features to the encoder block. The features are passed through multi-head attention layer and feed forward layer.
3. Extract caption features using BERT which is later summed with the positional encoding.
4. Shift the output from previous step (3) to the right and feed it to the decoder block's first masked multi-head attention layer.
5. The output of (4) is given to multi-head attention layer along with output of the encoder block.

6. After the feed forward and add & norm the output of the decoder block is passed through linear and softmax layer which yield the probabilities of the words that make the image caption.

Now let us look at each step descriptively. On the encoder side, let us consider an image '*Image*' and which is passed through the CNN function to extract the features of the image as follows–

Equation 5.3 Extract Image Features

$$f = CNN(Image)$$

Then f become the input which is passed to a Multi-Head Attention (MHA) layer of size $n \times d_{model}$ as seen in equation 5.3. An MHA layer focusses on the entire feature sequence rather than each feature at a time. Thereby, improving the quality of result from neural network. Thus, a stack of MHA will help in understanding different aspects of input data. The Add & Normalize steps simply adds the input from previous layer to the output of previous layer and performs a layer normalization. Equation 5.4 shows the first block of MHA block of encoder followed by the Add & Norm layer. The point wise feed forward network is a fully connected layer. It performs two linear transformations with ReLU activation. Equaton 5.5 depicts the second layer of encoder block. Let output of 1 encoder block can be defined as $y_{encoder}$. So, following equations help in achieving the $y_{encoder}$ –

Equation 5.4 Equation for first Add & Norm layer of Encoder

$$y'_{encoder} = LayerNormalization (f + MultiHeadAttention(f))$$

Equation 5.5 Equation for Add & Norm layer of Encoder

$$y_{encoder} = LayerNormalization(y'_{encoder} + FeedForward(y'_{encoder}))$$

On the decoder side, let us represent '*Caption*' as the English language sentence input to the decoder. This is first passed through BERT in order to get language features. These features are

summed with positional encoding which finally becomes the decoder input. Following equation 5.6 shows how caption is converted to become decoder input.

Equation 5.6 Equation for creating decoder input

$$f' = BERT(caption) + positional_encoding(caption)$$

The input f' is passed through a masked multi head attention. During execution of this layer, the decoder will ‘hide’ future inputs to ensure that outputs from this stage can only know of the words from the previous stage. After the first masked MHA and layer normalization. Equation 5.7 sums up the first block of decoder. The second MHA is the Encoder-Decoder MHA. This combines the output of encoder block with output of previous MHA. This establishes a correlation between the image features and caption features. After another layer normalization a position wise feed forward layer is executed followed by a layer normalization as seen in equation 5.7 and 5.8. Once repeated N times, this concludes the transformer decoder block.

Equation 5.7 Equation for first Add & Norm layer of decoder

$$y'_{decoder} = LayerNormalization(f' + Masked MultiHeadAttention(f'))$$

Equation 5.8 Equation for second Add & Norm layer of decoder

$$y''_{decoder} = LayerNormalisation(y'_{decoder} + MultiHeadAttention(y_{encoder}, y'_{decoder}))$$

Equation 5.9 Equation for third Add & Norm layer of decoder

$$y_{decoder} = LayerNormalizatio(y''_{decoder} + FeedForward(y''_{decoder}))$$

The $y_{decoder}$ is passed through a fully connected network of size of the vocabulary of the model, which creates a probability distribution matrix of a caption words (vocab_size) which is then passed through a Softmax function.

If we refer section 2.2, the CNN-LSTM model faces a vanishing gradient problem. It is seen that the image features are input only at the beginning of the decoder execution. Transformer architecture alleviates that drawback by using image encoder vectors at every decoder layer's execution, as seen in Figure 5.4 (b). From section 2.3, we observed that the Attention-based model only paid attention to part of an image and used image features from that section during the LSTM decoder execution cycle. Transformers decoders on the other hand, use the entire image features during caption prediction (as seen in Figure 5.4 (a)), thereby resolving the drawback of Attention-based models.

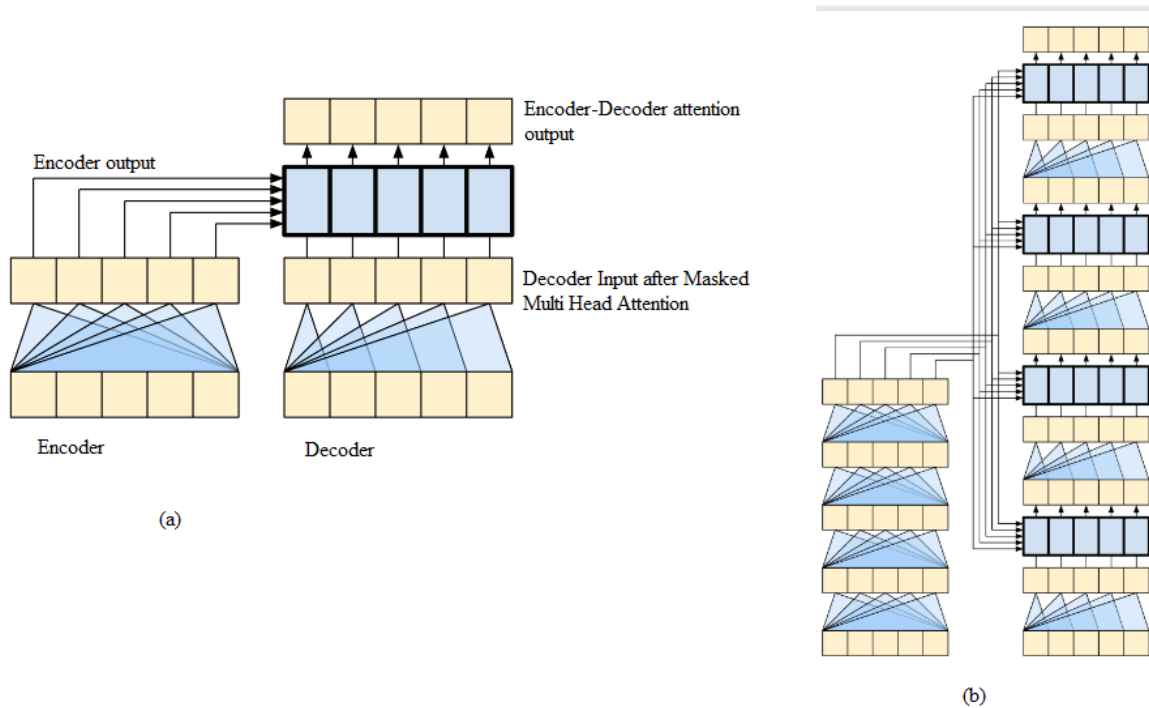


Figure 5.4 (a) Internal representation of 1 layer of encoder & decoder (b) Shows internal representation for 4-layer encoder & decoder Transformer

6. EXPERIMENT & EVALUATION

This section discusses is divided into 3 parts. Part 6.1 will describe the experimental set up and discuss the parameters that have been selected for the development of the CATIC model. In section 6.2 the model evaluation techniques and finally in 6.3 we compare the performance of the CATIC models with that of other IC models for RSICD. Also, we see the captions obtained from the model.

6.1 Experimental Setup

The Transformer model allows the user to configure some parameters according to their experiment requirements -such as the number of attention heads or number of encoder or decoder layers. All the experiment setup details are listed in the table below.

Table 6.1 Experiment Parameters

Experiment Parameter	Value
Encoder Layers	4
Decoder Layer	4
Attention Heads	8
Feed Forward Dimension	2048
Max Length of Sentence	50
Number Image/Language Feature Filters	768
Dropout Rate	0.1
Epochs	60
Vocabulary Size	3073

The dataset used for this model is called Remote Sensing Image Captioning [12] dataset. The dataset is split in ratio of 80:20 by the authors where in 80 percent of dataset is used for model training and 20 percent for testing. As mentioned above, the data set is cleaned such that each image only has unique captions attached to them. The model is trained using Google Colab platform for 60 epochs with 16 GB RAM using TPU over a span of 18 hours.

The language feature extraction is accomplished by using Huggiface library [20]. All these computations are quite expensive with regards to memory and the billing rates of the Google Cloud Platform. Therefore, a clean dataset reduced both computational time and billing. Each datapoint is considered as images (model input) and the corresponding caption (model output). Therefore, the new number of datapoints after cleaning are mentioned in Table 6.2.

Table 6.2 Test and Train Dataset after clenaing

	Train	Test
Original Dataset	43670	10935
Cleaned Dataset	17813	6519

The loss has been calculated using the Sparse Categorical Cross Entropy. The loss is calculated between the predicted values from the softmax function say ' p_{model} ' and the real values of the caption as C. The formula for calculating is the loss for Sparse Categorical Cross Entropy is same as the Categorical Cross Entropy (CCE) function as given below.

Equation 6.1 Formula for calculating Sparse Categorical Cross Entropy

$$Loss = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C 1_{y_i \in C_c} \log(p_{model}) [y_i \in C_c]$$

where, i iterates on i observations, c iterates on c classes, 1 is an indicator function and p_{model} is the predicted probability of i th observation of class c . The loss calculation function is same, however, the categories are represented in an integer format in Sparse CCE. In CCE the categories are represented in one-hot-encoded format. The integer representation is easier to implement in code and it requires lesser time and computation.

The model training is optimized using Adam Optimizer [21] with β values as $\beta_1 = 0.9$, $\beta_2 = 0.98$. The learning rate was varied through the training using the formula below –

Equation 6.2 Formula for calculating learning rate during the training

$$lrate = d_{model}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5})$$

This formula corresponds to incrementing the learning rate linearly for the first warmup counts of the training steps and then reducing it proportionally to the negative square root of the step number. The *warmup_step* was set to 4000. The following graph in Figure 6.1 shows the varying learning rate with number of steps.

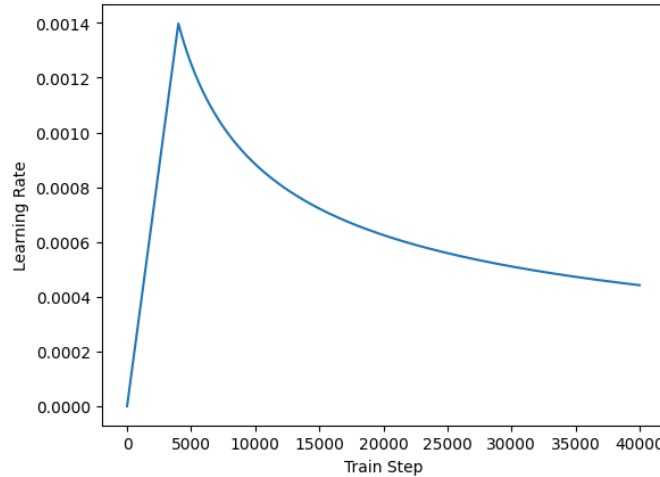


Figure 6.1 Learning Rate vs Train Step graph

For evaluation of the model BLEU6 [5], METEOR [6] and ROGUE [7] techniques are used. NLTK library [22] is used for evaluating BLEU score. Python packages for ROGUE have been used to calculate the score. Finally, the METEOR score has been calculated by using the METEOR jar of 1.5 version has been used as given by the authors on their website [23]. The

evaluation metric results are compared to the popular CNN-LSTM IC model and the cutting-edge Attention-based IC model.

6.2 Evaluation Techniques

As an IC model predicts a caption in a human-readable format, we used NLP evaluation techniques such as Bilingual Evaluation Understudy Score (BLEU), Recall Oriented Understudy for Gisting Evaluation scoring (ROGUE), and Metric for Evaluation of Translation with Explicit ORdering (METEOR) to assess the model's performance .

These methodologies use some common metrics such as Precision, Recall and F1 Score for calculating the scores of the predictions. These metrics use measure performance using True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The TP, TN, FP, and FN values are found by developing a confusion matrix by comparing the actual values to the predicted values as shown below.

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Figure 6.2 Confusion Matrix with TN, TP, FN and FP

Using the TP, TN, FP, and FN values we can calculate the values for Precision, Recall and F1 Scores.

- **Precision** – Measures how many values are accurately predicted positive out of the total positive predicted values.

Equation 6.3 Formula for Precision

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** – Measures how many values were accurately predicted positive out of the total positive values.

Equation 6.4 Formula for Recall

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score** - It is a weighted average of Precision and Recall.

Equation 6.5 Formula for F1-Score

$$F1 - Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

With help of these metrics the BLEU, ROGUE and METEOR scores are easier to define. Let us look at each score. The score values lie between 0 and 1. A model is said to be doing well when the score values are closer to 1.

6.2.1 BLEU Score

The Bilingual Evaluation Understudy score is precision-based scoring methodology. As per the definition of it can be interpreted as how much the words from the predicted sentences appeared in the reference sentences. BLEU score is not expensive on the memory and very easy to implement. As it is language independent, it can be adopted for any model readily.

BLEU score allows the user to calculate N-Gram score. This means that the score is evaluated on basis of how many words match for a specific order. Such as BLEU-1 would mean a 1-Gram score that see if a single word match(s) is(are) found or not. Similarly, BLEU-2 is for word pairs and BLEU 3 for group of 3 words and so on.

Though BLEU score is straight forward, it also incorporates 'brevity penalty.' That is, even if the n-grams give a high score and the predicted sentence is shorter than the reference sentence, the translation is penalized by multiplying it by a brevity measure. If the predicted sentence is shorter than reference sentence, then we calculate brevity by dividing length of closest sentence by the predicted sentence length and subtract 1 from that value. The remaining values is used to

raise the value of e which is finally the brevity measure. If the length of predicted sentence is longer, then brevity penalty measure is 1.

Let us understand how BLEU 1-4 scores are calculated with an example. Precision for n-gram would be calculated as follows –

Equation 6.6 Precision for n-gram

$$\text{Precision } n - \text{gram} = \frac{\text{Number of correct predicted } n - \text{gram}}{\text{Number of total predicted } n - \text{grams}}$$

After calculating the Precision for n-grams, we calculate the Geometric Average Precision Scores for each n-gram using the formula shown in equation 6.7. As mentioned above the Geometric Average Precision is multiplied by the Brevity penalty which is given by equation 6.8 which finally give us the BLEU-N score as mentioned in equation 6.9.

Equation 6.7 Precision for n-gram

$$\text{Geometric Average Precision Score } (N) = \exp\left(\sum_{n=1}^N w_n \log p_n\right) = \prod_{n=1}^N p_n^{w_n}$$

Equation 6.8 Brevity Penalty

$$\text{Brevity Penalty} = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c \leq r \end{cases}$$

Where, c is the length of predicted sentence and r is the length of reference sentence.

Equation 6.9 Final BLEU(N) score formula

$$\text{BLEU}(N) = \text{Brevity Penalty} \cdot \text{Geometric Average Precision Score}(N)$$

6.2.2 METEOR Score

The METEOR score tries to compensate the shortcomings of the BLEU score by incorporating Recall in the calculation. METEOR calculates the harmonic mean for a unigram, based on precision and recall. It is calculated as follows.

Equation 6.10 Formula for METEOR Score calculation

$$METEOR\ Score = \frac{10 * Precision * Recall}{Recall + 9Precision}$$

METEOR lays a lot of importance of the ordering of the sentence which automatically stresses the semantic similarity of the sentence. It is also a great evaluation metric as it accounts for stemming and synonyms. The authors of METEOR updated their model by addition of paraphrase matching which are automatically extracted. In the newer version of the METEOR score calculation, the words have weights attached to them such that they can be discriminated as functional word or content word.

6.2.3 ROGUE Score

ROGUE is an acronym for the Recall Oriented Understudy for Gisting Evaluation scoring algorithm. ROGUE produces a set of metrics – ROGUE-1, ROGUE-2, ROGUE-L. Despite the name suggesting recall, ROGUE scores are both recall and precision based. It calculates how many words from the human reference sentence are present in the model generated sentence.

Like BLEU n-gram, ROGUE-1 and ROGUE-2 measures the match rate of single word or pair of words (respectively) which are present in model generated sentence and reference sentence. The ROGUE-L metric distinguishes itself by measuring recall, precision, and F1 score for the 'longest common subsequence' between the predicted and reference sentences. A longest common sequence is a sequence of words which may or may not be in consecutive but are present in both reference and predicted sentence. In simple terms, it would mean that the longer the shared

sequence then higher will be the similarity between the sequences. For comparison ROGUE F1 score has been considered.

Equation 6.11 Formula for ROGUE_L F1 Score calculation

$$\text{ROGUE-L F1} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

6.3 CATIC Results

In this section we will first compare the results of CATIC model to the results of CNN-LSTM IC model (Table 6.3) and Attention-based IC model (Table 6.4) for the BLEU1-4 score, METEOR score and ROGUE score evaluation metrics.

In Table 6.3 we can observe that CATIC model outperforms the CNN-LSTM based models. In Table 6.4 represents the comparison between the Attention-Based models. CATIC outperforms them in BLEU2 and ROGUE_L metrics and performs competitively for all other matrices.

Table 6.3 Performance comparison between CATIC and CNN-LSTM based model

Model Name	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	ROGUE_L
VGG19 - LSTM	0.58330	0.42259	0.33098	0.27022	0.26133	0.51891
AlexNet - LSTM	0.57905	0.41871	0.32628	0.26552	0.26103	0.51913
CATIC	0.68486	0.54936	0.44	0.3324	0.30948	0.63009

Table 6.4 Performance comparison between CATIC and Attention based Image Captioning model


Model	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	ROGUE_L
AlexNet – Hard Attn	0.68768	0.5446	0.44396	0.36895	0.33521	0.62673
GoogleNet – Hard Attn	0.68813	0.54523	0.44701	0.3725	0.33324	0.62837
Our Model	0.68486	0.54936	0.44	0.3324	0.30948	0.65515


The scores that are documented in the table provide an average corpus score, which implies that in some of the cases where the model is predictions are closer to 1 may get washed out or the poor predictions made by the model may get hidden. As a result, it is also necessary to examine the captions generated by the model and compare them to the reference captions provided. These captions are being evaluated by humans on a case-by-case basis. The predicted captions are being judged on two basis – identification of image features and preservation of context or semantic meaning in prediction. Identification of image features will quantify if the caption has recognized objects of the image correctly. Preservation of context evaluates if the predicted caption was able to understand the semantic meaning of the image.


The following section discusses the best, average, and poor predictions made by the CATIC model.

- Best Captions generated by CATIC-

Figure 6.3 displays an example of the best captions generated. These captions come close to original caption in length and context but also capture the image features correctly. In the first image we see that the ‘trees’ and ‘woods’ are exchanged synonymously. Similarly, for third image the context is maintained by exchanging the word ‘across’ with ‘on.’ For the second image we observe that the context of the original caption is same as that of the predicted caption.

1. 

Original Caption: there is a factory near a road and a woods .
Predicated Caption: there is a factory near a road and a trees .
2. 


Original Caption: a park with some buildings, a pond and green trees is near a river .
Predicated Caption: a river and buildings are in the around park near to river .
3. 


Original Caption: a bridge across a big river .
Predicated Caption: a bridge on a big river .

Figure 6.3 Best captions predictions by CATIC

- Average Captions generated by CATIC-

The average captions would at least identify the objects of the image correctly and give some semblance of the context being same in the predicted caption. In following examples shown in Figure 6.4, we see that CATIC can predict the features such football field and colors of objects correctly. However, the meaning behind the original captions is only present in some extent.

1. 


Original Caption: the four baseball field forms into a round .
Predicated Caption: the two football field is to a square .
2. 


Original Caption: a church with roofs of black and white or brick red has some cars parked in the yard .
Predicated Caption: a house with white of white and white and white green is some cars in in the lawn .


Figure 6.4 Average captions predicted by CATIC

- Poor Captions generated by CATIC –

Some of the captions generated by CATIC model could not recognize the image features therefore not predict the right captions. Following are examples shown in Figure 6.5 have poor captions generated by the model. As we can observe in all the images do not possess any distinct features or objects in them. As a result, the model cannot anticipate the appropriate words to describe the image.

1. 

Original Caption: surging waves flapped in rock .
Predicated Caption: the and and is are a green in in green on .
2. 

Original Caption: texture of the mountain is very beautiful .
Predicated Caption: the of the trees is large large .
3. 

Original Caption: several sand hills scetch across this desert .
Predicated Caption: a green trees are of around through the two .

Figure 6.5 Poor captions predicted by CATIC

Let us now look at some general results from CATIC model. As mentioned above, the dataset creators have tried to use many categories (approximately 30) of images such as beaches, farmlands, forests, deserts, rivers, churches, etc. Following table is a collection of captions created by CATIC along with their original captions. Following Table shows results from CATIC

Table 6.5 CATIC Results for various categories




Category	Image	Captions
Airport		<p>Original Caption: four planes are near several buildings in an airport with several runways .</p> <p>Predicated Caption: two planes are near two buildings in a airport with two roads .</p>
Bare land		<p>Original Caption: it is a large piece of yellow bareland .</p> <p>Predicated Caption: there are green and white land land on the piece of the green land . some some green and barren - shaped land land .</p>
Baseball field		<p>Original Caption: many tall trees are planted around the baseball field .</p> <p>Predicated Caption: a green trees are on around the baseball field .</p>

Table 6.5 Continued




Beach		<p>Original Caption: the sunlight is very good on the beach .</p> <p>Predicated Caption: a large piece of green ocean is near a white beach .</p>
Church		<p>Original Caption: a church with pitched brick red roofs and a regular octagon pyramidal roof of blue green is located near a parking lot and a sports field .</p> <p>Predicated Caption: the white square ##oy shaped houses with a white ##an roof ##tangle and roof is surrounded by some trees .</p>
Commercial		<p>Original Caption: rows of houses are in this commercial area .</p> <p>Predicated Caption: some buildings are in in a residential area .</p>

Table 6.5 Continued

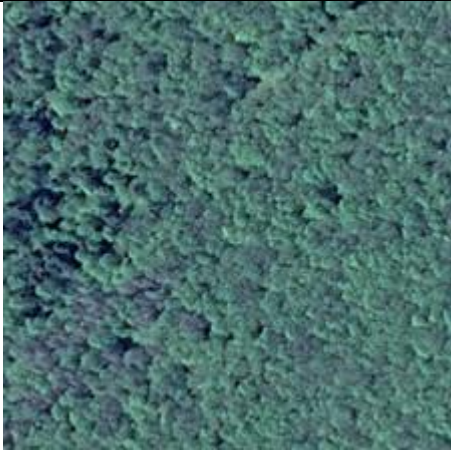


Forest		<p>Original Caption: the dark green forest is dense and airtight .</p> <p>Predicated Caption: it is a green green with forest with dense white plants .</p>
Industrial		<p>Original Caption: industrial use of large area .</p> <p>Predicated Caption: some buildings are in a industrial area .</p>
Parking		<p>Original Caption: the parking includes a few white and ren cars .</p> <p>Predicated Caption: some cars are in a parking lot .</p>

Table 6.5 Continued

Playground		<p>Original Caption: several parallel roads are near a playground .</p> <p>Predicated Caption: a roads roads are near a playground .</p>
------------	--	--

7. DISCUSSION & FUTURE WORK

As highlighted in the Literature Survey section the previous image captioning models showed 3 major problems – vanishing gradient, sequential execution, and lack of context. With help of Transformer architecture, we can address the vanishing gradient and sequential execution drawbacks. Also, by extracting language features using pre trained BERT we have tried to inject contextually rich captions while training. On evaluating CATIC with CNN-LSTM IC models we saw that CATIC supersedes it in all evaluation metrics. When compared with Attention-based IC models CATIC supersedes in BLEU2 and ROGUE_L evaluation metrics and gives competitive values for rest of the metrics.

Despite promising results there are few challenges that still need to be resolved for CATIC. Since the pretrained BERT is a large model, it adds significant computational cost of the training. Customizing it for our use case, would reduce the time and effort of computation, which will positively impact the model. Similarly, presence of duplicate data increases the size of the dataset. Which in turn takes a toll on training resources. As the development environment used is Google Colab, using high RAM configuration runs a huge billing cost. If the budget for development environment were increased then usage of higher RAM and TPUs would allow the model to be trained on the entire dataset. As the creators of database have made sure that the captions do not use relative positional words for description, we would also like to increase the scope of dataset by performing image augmentation activities such as rotation and taking a mirror image. In future we would like to expand the scope by training the model on bigger datasets. As of now the big datasets include MSCOCO, FLICKER8, FLICKER30. These datasets contain various categories of images and would make the model richer.

For a purely NLP use case there are special libraries that can be used for visualizing the hidden stages of the Transformer model. These visualizations are called attention plots and they help in understanding the extent of correlation between the encoder output and decoder input. However, there is no way such a visualization can create visualization for image captioning use case. The scope of the project could be increased such that visualization techniques can be

developed for the viewing the correlation between the image features and the decoder input. This would help us in understanding which features of the image have a higher impact in selecting the respective word from the vocabulary. This could also help in creating a visual comparison between attention-based models and CATIC.

Named Entity Recognition is one of the methods for extracting language features. This NLP technique highlights some basic concepts of text as it identifies nouns, locations, dates, etc. Using Named Entity Recognition along with powerful object recognition algorithms, we can caption images more specifically than using generic descriptors.

8. CONCLUSION

This thesis performs an Image Captioning using Transformer architecture. Significant drawbacks were identified in present day popular models which use Convolution Neural Network, Long-Short Term Memory and Stochastic/Deterministic based attention for Image Captioning. The work presented here tries to overcome those challenges by building Contextually Aware Transformer based Image Captioning (CATIC) model. The CATIC model uses pretrained BERT for extracting language features instead of using standard word tokenization techniques to inject context rich captions while training. Finally, the performance of CATIC is evaluated using metrics such as BLEU 1-4 score, METEOR score and ROGUE_L score. Our model does better than a plain CNN-LSTM model in all evaluation metrics. CATIC performs competitively in comparison to Attention-based model outperforms Attention-based model for BLEU2 and ROGUE_L metrics.

REFERENCES

- [1] P.Gopika; C.S.Krishnendu; M. Hari Chandana; S. Ananthakrishnan; V.Sowmya; E.A. Gopalakrishnan; K.P. Soman (2020). "Chapter two - Single-layer convolution neural network for cardiac disease classification using electrocardiogram signals".
- [2] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986.
- [3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.
- [4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.
- [5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.
- [6] Denkowski M, L.A.: Meteor universal: language specific translation evaluation for any target language. In: *Proceedings of the Ninth Workshop on Statistical Machine Translation* (2014)
- [7] Lin, C.Y.: Rouge: a package for automatic evaluation of summaries. In: *Text Summarization Branches Out*, pp. 74–81 (2004)
- [8] Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015)

- [9] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. CoRR, abs/1411.4555, 2014.
- [10] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. CoRR, abs/1502.03044, 2015.
- [11] Medium, Review Show, Attend Tell: Neural Image Caption Generator Available at: <https://sh-tsang.medium.com/review-show-attend-and-tell-neural-image-caption-generation-f56f9b75bf89>
- [12] Lu, X., Wang, B., Zheng, X., et al. 2017. “Exploring models and data for remote sensing image.” IEEE Trans. Geosci. Remote Sens.
- [13] Github, Remote Sensing Image Caption Dataset, Available at: https://github.com/201528014227051/RSICD_optimal
- [14] K. Luu, X. Wu, R. Koncel-Kedziorski, K. Lo, I. Cachola, and N. Smith, “Explaining Relationships Between Scientific Documents.”
- [15] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.
- [17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9:1735–80, 12 1997.

- [18] Yoav Goldberg and Omer Levy, word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method, 15 Feb 2014
- [19] Luhn, Hans Peter (1957). "A Statistical Approach to Mechanized Encoding and Searching of Literary Information" (PDF). *IBM Journal of Research and Development*. 1 (4): 309317. doi:10.1147/rd.14.0309. Spärck Jones, K. (1972). "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". *Journal of Documentation*. 28 (1): 11–21. CiteSeerX 10.1.1.115.8343. doi:10.1108/eb026526.
- [20] Huggingface, Tensorflow BERT Model Available at:
https://huggingface.co/docs/transformers/model_doc/bert#transformers.TFBertModel
- [21] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980 Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
- [22] NLTK library, BLEU Evaluation library Available at:
https://www.nltk.org/api/nltk.translate.bleu_score.html
- [23] METEOR jar, METEOR Evalutaion Availabel at:
<https://www.cs.cmu.edu/~alavie/METEOR/>