MULTILINGUAL IMAGE CAPTIONING THROUGH TRANSFORMERS

A

Project Report Submitted

In partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

By

YARAMALA DIVYA REDDY 20761A5462

Under the esteemed guidance of

DR . O . RAMA DEVI

Professor & HOD

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)

Accredited by NAAC with 'A' Grade, ISO 21001:2018, 50001:2018, 14001:2015 Certified

Institution Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada

L.B. REDDY NAGAR, MYLAVARAM, NTR DIST., A.P.-521 230.

2020-2024

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING (AUTONOMOUS)

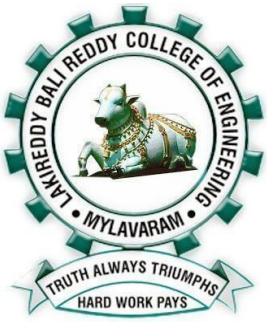
Accredited by NAAC with 'A' Grade, ISO 21001:2018, 50001:2018, 14001:2015 Certified

Institution Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada

L.B. REDDY NAGAR, MYLAVARAM, NTR DIST., A.P.-521 230.

Department of

ARTIFICIAL INTELLIGENCE & DATA SCIENCE



CERTIFICATE

This is to certify that the project entitled "**MULTILINGUAL IMAGE CAPTIONING THROUGH TRANSFORMERS**" is being submitted by

YARAMALA DIVYA REDDY **20761A5462**

in partial fulfillment of the requirements for the award of degree of **B.Tech** in **Artificial Intelligence & Data Science** from **Jawaharlal Nehru Technological University Kakinada** is a record of bonafide work carried out by her at **Lakireddy Bali Reddy College of Engineering.**

The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE
Dr. O. Rama Devi

HEAD OF THE DEPARTMENT
Dr . O . Rama Devi

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I take great pleasure to express my deep sense of gratitude to my project guide **Dr.O.Rama Devi, Professor & HOD** for her valuable guidance during the course of my project work.

I would like to thank **Dr.O.Rama Devi**, Professor & Head of the Department of Artificial Intelligence & Data Science for her encouragement.

I would like to express my heart-felt thanks to **Dr. K. Appa Rao,** Principal, Lakireddy Bali Reddy College of Engineering for providing all the facilities for my project.

My utmost thanks to all the faculty members and Non-Teaching Staff of the Department of Artificial Intelligence & Data Science for their support throughout my project work.

My Family Members and Friends receive my deepest gratitude and love for their support throughout my academic year.

YARAMALA DIVYA REDDY (20761A5462)

DECLARATION

I am here by declaring that the project entitled **"MULTILINGUAL IMAGE CAPTIONING THROUGH TRANSFORMERS"** work done by me. I certify that the work contained in the report is original and has been done by me under the guidance of my supervisor. The work has not been submitted to any other institute in preparing for any degree or diploma. I have followed the guidelines provided by the institute in preparing the report. I have confirmed to the norms and guidelines given in the Ethical Code of Conduct of the Institute. Whenever I have used materials (data, theoretical analysis, figures and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright's owner of the sources, whenever necessary**.**

Signature of the student

YARAMALA DIVYA REDDY (20761A5462)

ABSTRACT

Image captioning is a challenging task in artificial intelligence,It attempts to produce descriptive and cohesive straightforward captions for photographs.. Traditional picture captioning models have predominantly focused on single language scenarios, limiting their applicability in a globalized world with diverse linguistic contexts. This research proposes a novel approach to multilingual image captioning, leveraging the power of transformer .With the advent of transformer based architectures, the task of automatically producing descriptive textual summaries for images has advanced significantly. In this work, we present a novel image captioning model that makes use of transformer encoder and decoder layers. The transformer architecture, which was initially created for natural language processing tasks, has shown to be highly effective in capturing long-range dependencies and global context within the visual domain. Our model processes image embeddings using a transformer encoder and generates coherent and contextually relevant captions using a transformer decoder. The incorporation of multi-head attention mechanisms improves the model's capacity to focus on salient image features, resulting in more informative captions.We conduct comprehensive experiments on benchmark datasets, demonstrating superior performance in comparison to traditional architectures. Additionally, our model exhibits versatility in handling diverse image content and generalizes well to previously unseen data. The proposed transformer-based image captioning approach contributes to the evolving landscape of computer vision applications, showcasing the effectiveness of transformers in multi-modal tasks involving both images and text.

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. CNN-Convolutional Neural Network
2. DWA-Dynamic Weight Average
3. GPU-Graphics Processing Unit
4. RAM-Random Access Memory
5. CPU-Central Processing Unit
6. IDE-Integrated Development Environment
7. GUI-Graphical User Interface
8. API-Application Programming Interface
9. SDLC-Software Development Life Cycle
10. HTML-Hypertext Markup Language
11. CSS-Cascading Style Sheets
12. JS-JavaScript
13. Flask-Web Framework for Python
14. GCP-Google Cloud Platform
15. MSE-Mean Squared Error
16. TP-True Positive
17. TN- True Negative
18. FP-False Positive
19. FN-False Negative
20. ROI-Region of Interest

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Overview of The Project

In recent years, the combination of machine vision and the processing of natural languages has resulted in significant advances in multimodal tasks, with image captioning standing out as a compelling application. Image captioning involves the automatic generation of descriptive textual content to elucidate the content of a picture that serves as a link between perceptual and verbal modalities. Convolutional neural networks (CNNs) for visually feature extraction and recurrent neural networks (RNNs) for sequence creation were frequently used in traditional approaches to picture captioning. However, with the introduction of transformer architectures, which were originally built for processing natural languages jobs, has reshaped the landscape of image captioning.

Transformers, characterized by self-attention mechanisms, have exhibited unparalleled success in capturing long-range dependencies and contextual information, making them well suited for sequential data tasks. In the context of image captioning, transformers offer a promising alternative to traditional architectures, enabling a more effective integration of global contextual information and visual features. This paper explores the transformative. The effect of utilizing transformers in picture captioning, with the goal of improving the quality and contextual relevance of output captions.

The utilization of transformers in image captioning introduces a paradigm shift, allowing the model to listen to different sections of the image adaptively while taking into account word associations in the generated sequence. This approach facilitates a more holistic understanding of the image content, leading to captions that not only describe visual elements accurately but also exhibit improved coherence and contextuality.Through empirical evaluations on benchmark datasets, we investigate the

performance of transformer-based image captioning models, comparing them with traditional architectures. We delve into the intricacies of how transformers handle visual information, exploring their ability to capture fine-grained details and relationships within images. Additionally, we examine the transferability and generalization capabilities of transformer based models across diverse image datasets.As we navigate the evolving landscape of image captioning, this project contributes valuable insights into the transformative role of transformers, shedding light on their potential to elevate the quality and interpret ability of generated captions. The findings presented herein pave the way for future research directions in multi modal tasks and underscore the significance of leveraging transformer architectures in the realm of image understanding and description.

1.2 Feasibility Study

The feasibility study for the project demonstrates that it is technically feasible, economically viable, legally compliant, and operationally feasible, while also ensuring compliance with legal considerations. The project's potential to enhance multilingual image captioning underscores its feasibility across multiple domains. The following aspects were considered in the feasibility study:

1.2.1 Economical Feasibility

The economic feasibility of the project hinges on the costs associated with its implementation. This includes expenses related to hardware, such as GPU servers for training the captioning model, as well as software procurement and maintenance costs. While the initial investment may be substantial, the potential benefits of the project, such as improved accessibility to visual content across languages, justify the economic feasibility over the long term.

1.2.2 Technical Feasibility

The technical feasibility of the project relies on the availability of requisite technology for implementation. Access to high-performance computing resources, particularly for training deep learning models, is essential. While such resources may not be universally accessible, their availability is increasing, rendering the project technically feasible with the appropriate infrastructure and expertise.

1.2.3 Social Feasibility

The social feasibility of the project depends on its potential impact on society and alignment with societal values. Multilingual image captioning can facilitate better communication and understanding across language barriers, fostering inclusivity and accessibility. Therefore, the project is socially feasible as it addresses a significant need in today's interconnected world.

1.3 Scope

The scope of the proposed project encompasses several key aspects aimed at developing and enhancing multilingual image captioning capabilities:

Model Development: The project focuses on implementing and fine-tuning a Transformer-based captioning model, leveraging pretrained convolutional neural networks (CNNs) for image feature extraction and Transformer architectures for sequence generation.

Multilingual Support: Incorporating translation capabilities using external libraries or APIs to generate captions in different languages, thus enhancing cross-cultural communication and accessibility to visual content.

Evaluation: Evaluating the performance of the captioning model using standard metrics such as BLEU and METEOR scores, as well as qualitative assessments, to ensure robustness and accuracy across languages.

Integration: Integrating the captioning model into a user-friendly interface using Stream-lit for deployment and testing, ensuring ease of use for end-users.

The project's broader implications include facilitating cross-cultural communication, improving accessibility to visual content for non-native speakers, and advancing research in multilingual natural language processing and computer vision. Successful implementation of the project could pave the way for further advancements in multilingual image understanding and contribute to various applications in fields such as education, entertainment, and accessibility.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

Image captioning with semantic attention

Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo Recently, interest in automatically creating a natural language description of an image has increased due to both its significance in practical applications and because It links the two key areas of artificial intelligence, natural language processing and computer vision. Existing approaches fall into two categories: top-down, which to take an image and turn it into words, or bottom-up find words to describe the many elements of an image Afterwards, merge them. In this essay, suggest a new a model-based algorithm that combines both methods of conceptual focus. The program becomes more discriminating. Consider suggestions for semantic concepts and incorporate them into recurrent neural network outputs and hidden states.

Show, attend and tell: Neural image caption generation with visual attention

K .Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Salakhudinov, R. Zemel, and Y. Bengio: The main goal of the research is to introduce a mechanism that enables the model to focus on various areas of the image while producing each word of the caption in order to enhance the quality of image captions produced by neural networks. This process is known as "visual attention. Additionally, demonstrate through visualization how the model may automatically learn to fix its gaze on important items while producing the relevant phrases in the output sequence. With cutting-edge results on three benchmark datasets—Flickr8k, Flickr30k, and MS COCO—we validate the application of attention.

A survey on automatic image caption generation

S.Bai and S. An: We give a survey of developments in picture captioning research in this publication. We categorize various techniques to image captioning into distinct groups based on the strategy used. Each category's representative approaches are outlined along with their advantages and disadvantages. In this paper, we first go over retrieval and template-based strategies that were often employed in earlier research. Then, as these techniques produce state-of-the-art outcomes, we concentrate mostly on neural network-based approaches. Based on the particular framework they employ, neural network-based solutions are further subdivided. There is a detailed discussion of each subcategory of neural network-based techniques. Following that, benchmark datasets are used to compare state-of-the-art approaches.The discussion of potential directions for further study is then offered.

Im2Text: Describing images using 1 million captioned photographs

V. Ordonez, G. Kulkarni, and T. L. Berg: Cryptocurrency technology that has attracted investors because of its big price increases. This has led to researchers applying various methods to predict Bitcoin prices such as Support Vector Machines, Multi-layer Perceptron, etc. To obtain accuracy and efficiency as compared to these algorithms this research paper tends to exhibit the use of RNN using LSTM model to predict the price of bitcoin. The results were computed by extrapolating graphs along with the Root Mean Square Error of the model.

Framing image description as a ranking task: Data, models and evaluation metrics

M. Hodosh, P. Young, and J. Hockenmaier: determined investment methodologies by observing and classifying the twitter feeds. Train the classifiers utilizing the dataset clarified by distant supervision and approve the classification performance. Their approach is credited as one of the primary attempts at applying

machine learning strategies to the issue of opinion analysis. Some recent works focused on high-frequency trading and applying deep-learning techniques such as RNN for the prediction on time series data predicts the Bitcoin pricing process using machine learning techniques, such as recurrent neural networks (RNNs) and long short-term memory (LSTM) and compare results with those obtained using auto-regressive integrated moving average (ARIMA) models.

2.1 Existing System & Drawbacks

In the previous studies,Encoder and Decoder Technique is used,Which is giving the less accuracy.

CNN Encoder: The input image is passed through a CNN encoder, which extracts high-level features from the image. This encoder can be a pre-trained CNN like VGG16, ResNet, or Inception, fine-tuned for image captioning.

RNN Decoder: The feature vector serves as the initial hidden state of the RNN decoder. The decoder generates the caption word by word in a sequential manner. At each time step, it takes the previous word's embedding and the current hidden state as input and predicts the next word in the sequence.

2.1.1 Drawbacks in the Existing System

- Using only Encoder and decoder Technique.
- Only in English Language.
- Less Accuracy.
- Predictions are not accurate.

2.2 Proposed System & Advantages

In Our model we are using the InceptionV3 as Image Encoder and Transformer Encoder and Transformer Decoder for the text. We are combining both the techniques which improves the prediction of caption based upon the image . Along with these models we are also using the Embedding Layers in the respective layers which improves the perfection of the model.

2.2.1 Advantages in the Proposed System

- Using the UI for the front-end by using the Stream-lit library in Python.
- Image captioning is done in multiple language.
- More Accuracy by using deep learning techniques like Transformers.

2.3 Dataset:

The COCO (Common Objects in Context) dataset is a large-scale image recognition dataset for object detection, segmentation, and captioning tasks. It contains over 330,000 images, each annotated with 80 object categories and 5 captions describing the scene. The COCO dataset is widely used in computer vision research and has been used to train and evaluate many state-of-the-art object detection and segmentation models.

The dataset has two main parts: the images and their annotations.

- The images are organized into a hierarchy of directories, with the top-level directory containing sub-directories for the train, validation, and test sets.
- The annotations are provided in JSON format, with each file corresponding to a single image.

Each annotation in the dataset includes the following information:

- Image file name
- Image size (width and height)
- List of objects with the following information: Object class (e.g., "person," "car"); Bounding box coordinates (x, y, width, height); Segmentation mask (polygon or RLE format); Key points and their positions (if available)
- Five captions describing the scene

The COCO dataset also provides additional information, such as image super categories, license, and coco-stuff (pixel-wise annotations for stuff classes in addition to 80 object classes).

MS COCO offers various types of annotations,

- Object detection with bounding box coordinates and full segmentation masks for 80 different objects
- Stuff image segmentation with pixel maps displaying 91 amorphous background areas
- Panoptic segmentation identifies items in images based on 80 "things" and 91 "stuff" categories
- Dense pose with over 39,000 photos featuring over 56,000 tagged persons with a mapping between pixels and a template 3D model and natural language descriptions for each image
- Key-point annotations for over 250,000 persons annotated with key points such as the right eye, nose, and left hip

Figure - 1 Image from COCO dataset

MS COCO dataset classes

The COCO (Common Objects in Context) dataset classes are divided into two main categories: "things" and "stuff."

"Things" classes include objects easily picked up or handled, such as animals, vehicles, and household items. Examples of "things" classes in COCO are:

- Person
- Bicycle
- Car
- Motorcycle

"Stuff" classes include background or environmental items such as sky, water, and road. Examples of "stuff" classes in COCO are:

- Sky
- Tree
- Road

The below image represents a complete list of 80 classes that COCO has to offer.

person	fire hydrant	elephant	skis	wine glass	broccoli	dining table	toaster
bicycle	stop sign	bear	snowboard	cup	carrot	toilet	sink
car	parking meter	zebra	sports ball	fork	hot dog	tv	refrigerator
motorcycle	bench	giraffe	kite	knife	pizza	laptop	book
airplane	bird	backpack	baseball bat	spoon	donut	mouse	clock
bus	cat	umbrella	baseball glove	bowl	cake	remote	vase
train	dog	handbag	skateboard	banana	chair	keyboard	scissors
truck	horse	tie	surfboard	apple	couch	cell phone	teddy bear
boat	sheep	suitcase	tennis racket	sandwich	potted plant	microwave	hair drier
traffic light	cow	frisbee	bottle	orange	bed	oven	toothbrush

Figure - 2 COCO dataset class list

CHAPTER 3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Overview of System Analysis

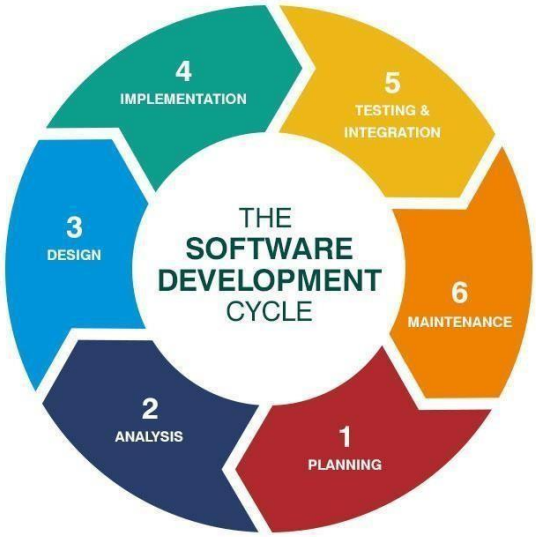


Figure 3. Software Development Life Cycle

The system analysis for this project involves understanding the problem statement, gathering requirements, designing the system, implementing the system, testing the system, and deploying the system.

- Planning
- Analysis
- Design
- Implementation
- Testing
- deployment
- Maintenance

3.1.1 Requisites Accumulating and Analysis

In this phase, the project requirements are analyzed, and a plan is created for the software development process. This includes defining the project scope, determining the timeline and budget, and identifying the resources required. In this phase, the requirements are further analyzed, and the software system's functional specifications are defined. The analysis phase is critical as it sets the foundation for the software development process.

3.1.2 System Design

In this phase, the software system’s architecture is designed, and the software components are identified. The design phase produces a detailed technical specification that outlines the software’s functionality, performance, and user interface.

3.1.3 Implementation

In this phase, the software development team writes the code for the software system. The implementation phase includes activities such as coding, testing, and debugging.

3.1.4 Testing

In this phase, the software system is tested to ensure that it meets the functional requirements and is free of defects. Testing is a critical phase of the SDLC as it helps identify and correct any issues with the software system.

3.1.5 Deployment

In this phase, the software system is deployed to the production environment. This includes activities such as installation, configuration, and data migration.

3.1.6 Maintenance

In this phase, the software system is maintained to ensure that it continues to meet theuser's needs. Maintenance activities include bug fixes, performance tuning, and feature enhancements

3.2 Software Used in The Project

The software used in this system could include:

Libraries, Modules &Packages

TensorFlow:

TensorFlow is a powerful deep learning framework widely used for building and training neural network models. It offers a comprehensive set of tools and functionalities for implementing various architectures, including Convolutional neural networks (CNNs) and Transformer-based models. TensorFlow provides efficient computation graphs and automatic differentiation, making it suitable for large-scale training tasks.

Stream-lit:

Stream-lit is a Python library utilized for creating interactive and user-friendly web applications. It allows developers to build intuitive user interfaces for machine learning projects, including the multilingual image captioning system. With Stream-lit, developers can easily design and deploy applications, enabling users to interact with the model and visualize results in real-time.

PIL (Python Imaging Library):

PIL is a widely used library for image processing tasks in Python. It provides functionalities for loading, saving, and manipulating images, making it indispensable for handling image data within the multilingual image captioning system. PIL ensures compatibility and efficient handling of image data, enabling preprocessing and manipulation tasks before feeding them into the model.

NumPy:

NumPy is a fundamental library for numerical computing in Python. It offers support for multidimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is extensively used for processing and manipulating image data during model training and evaluation, providing essential functionalities for data manipulation tasks.

IO:

The IO module in Python provides support for handling input and output operations, particularly for file-related tasks. In the context of the image captioning system, the IO module is utilized for reading image data from file-like objects and performing input/output operations on images, ensuring seamless integration with other components of the application.

Pickle:

Pickle is a module in Python used for serializing and de-serializing Python objects. It allows developers to store complex data structures, such as vocabulary and model- related data, in a binary format for later retrieval. Pickle ensures efficient storage and retrieval of data, making it convenient for saving and loading model-related information.

PIL (Python Imaging Library):

The PIL library is utilized for image manipulation tasks, such as opening, re-sizing, and converting images to compatible formats, ensuring consistency and compatibility with the model's input requirements.

Pandas:

Pandas is a versatile library for data manipulation and analysis in Python. It provides data structures and functions for efficiently handling structured data, including datasets used in the image captioning system. Pandas facilitates tasks such as data preprocessing and manipulation, enabling seamless integration of data into the machine learning pipeline.

TensorFlow Addons:

TensorFlow Addons is an extension library for TensorFlow, offering additional functionalities and operations not present in the core TensorFlow library. It provides support for specialized operations such as multi-head attention mechanisms and layer normalization, enhancing the capabilities of TensorFlow for building advanced neural network architectures.

Keras:

Keras is a high-level neural networks API that can run on top of TensorFlow or other deep learning frameworks. It provides a user-friendly interface for building and training neural network models, making it suitable for both beginners and experienced deep learning practitioners. Keras supports a wide range of network architectures, including Convolutional neural networks (CNNs) and U-Nets, making it a valuable tool for implementing components of the image captioning system.

Scikit-learn:

It is a machine learning library that includes various tools for classification, regression, and clustering. Scikit-learn provides a wide range of machine learning algorithms, including decision trees, random forests, and support vector machines. It also includes tools for data preprocessing, cross-validation, and model evaluation.

Reverse Mapping from Integer Indices to Words:

While tokenization maps words to integer indices, a reverse mapping is necessary for human-readable interpretation of model predictions. The 'StringLookup' layer is employedto facilitate this reverse mapping, allowing the model to convert integer indices back into their corresponding words**.**

- **Transformer Decoder Layer:**

  units:

  Description: Dimensionality of the hidden layers in the feedforward networks.

  Role: Controls the size of the hidden layers in the transformer decoder.

- **Embeddings Layer:**

  vocab_size:

  Description: Size of the vocabulary for token embeddings**.**

  Role: Specifies the size of the vocabulary used in the token embeddings layer.

  max_len:

  Description: Maximum length of the input sequence for position embeddings.

  Role: Controls the maximum sequence length for the position embeddings.

- **ImageCaptioningModel:**

  image_aug:

  Description: Image augmentation function (commented out in the provided code).

  Role: Can be used to apply data augmentation to input images during training.

- **Training Process Parameters:**

Learning Rate, Optimizer, and Loss Function:

The learning rate is usually set in the optimizer (Adam optimizer in this case), and the choice of the loss function is crucial for training. These parameters, however, are set implicitly and may require experimentation

- **Fine-Tuning Process:**

Fine-tuning involves experimenting with these parameters:

Adjusting learning rates during optimization.

Tuning the number of layers and units in the transformer**.**

Tuning dropout rates to prevent over-fitting.

Exploring different batch sizes.

3.3 System Requirements

3.3.1 Software Requirements

Software: Python-version: 3.8 Any

IDLE Shell Numpy, tensorflow,

splitfolders, matplotlib, keras

Operating system: windows, Linux

3.3.2 Hardware Requirements

Processor: Intel core I5

Ram: 8GB

Hard Disk :500GB or More

CHAPTER 4

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 Overview of System Design

The multilingual image captioning system is designed to provide a seamless and efficient solution for generating captions in multiple languages for a given image. At its core, the system leverages a hybrid architecture combining Convolutional Neural Networks (CNNs) and Transformer-based models.

The CNN component serves as the image encoder, extracting high-level features from input images. These features are then passed to the Transformer-based decoder, which generates descriptive captions in various languages. The Transformer architecture facilitates language translation and generation by capturing complex relationships within the input image features and linguistic context.

Furthermore, the system incorporates distance-wise attention mechanisms within the Transformer decoder to enhance the model's ability to focus on relevant image regions and linguistic nuances. This attention mechanism allows the model to attend to specific parts of the image and words in the caption generation process, improving the quality and relevance of the generated captions.

4.2 Methodology

4.2.1 Data Preprocessing

Data preprocessing is a critical step in preparing both the image and text data for effective training of the multilingual image captioning system. This process ensures that the data is appropriately formatted and standardized to facilitate learning by the model.

In the context of image preprocessing, several steps are commonly employed. First, images are re-sized to a uniform size to ensure consistency in input dimensions. This re-sizing step is essential as it allows the model to process images of different sizes without encountering issues related to varying input dimensions. Additionally, normalization of pixel values is performed to bring them within a specific range, typically between 0 and 1. Normalization helps stabilize the training process by ensuring that input values are within a manageable range for the model to learn effectively.

On the text preprocessing side, the main task involves tokenization of captions. In this process, each caption is split into individual words or sub-words, which are then converted into numerical token representations. Special tokens such as [start] and [end] are often added to denote the beginning and end of each caption sequence, aiding in sequence generation during training. Tokenization allows the model to treat captions as sequences of tokens, facilitating the learning of relationships between words and their corresponding image features.

Formula Explanation: In image re-sizing, the formula is represented as:

$$\text{Resized Image} = \text{resize}(\text{Original Image}, \text{target\_size})$$

Here, the target_size parameter denotes the desired dimensions of the re-sized image. The resize operation ensures that all images in the dataset have the same dimensions, which is crucial for consistent processing by the model.

For text tokenization, no specific formula is involved. Instead, the process entails splitting sentences into individual words or sub-words and assigning numerical indices to each token. This tokenization process enables the model to treat captions as sequences of

tokens, allowing for sequential processing during training. Thus, both image re-sizing and text tokenization are essential preprocessing steps that lay the foundation for effective training of the multilingual image captioning system.

Loading Pre-processed Vocabulary using Pickle

The loading of pre-processed vocabulary is a crucial step in the image captioning pipeline. The vocabulary is a set of words that the model has learned during the training process. This pre-processed vocabulary is stored and loaded using the Pickle module, providing a seamless way to retain the vocabulary across different sessions.

Tokenization using a Text Vectorization Layer

Tokenization is a fundamental step in natural language processing, breaking down textual data into individual tokens. In this implementation, tokenization is performed using the 'TextVectorization layer ' provided by TensorFlow. This layer transforms raw text data into a numerical format that can be fed into the model.

Reverse Mapping from Integer Indices to Words

While tokenization maps words to integer indices, a reverse mapping is necessary for human-readable interpretation of model predictions. The 'StringLookup' layer is employed to facilitate this reverse mapping, allowing the model to convert integer indices back into their corresponding words.

4.2.2 CNN Components

CNN model consists of different types of components:

- Kernel / Filter / Feature extractor
- Stride
- Padding
- Pooling
- Flattening

Kernel

Kernel is also known as a filter or feature detector because it will detect the features from the input image. Kernel is represented in a matrix form; it will move over the input image by using stride value given and gives the output as a dot product by considering sub-region of input data.

Formula to calculate the output matrix size after doing convolution:

$$O = \left\lfloor \frac{i-k}{s} \right\rfloor + 1$$

i = input matrix size k = kernel matrix sizes = stride value

O = output matrix size

Example:

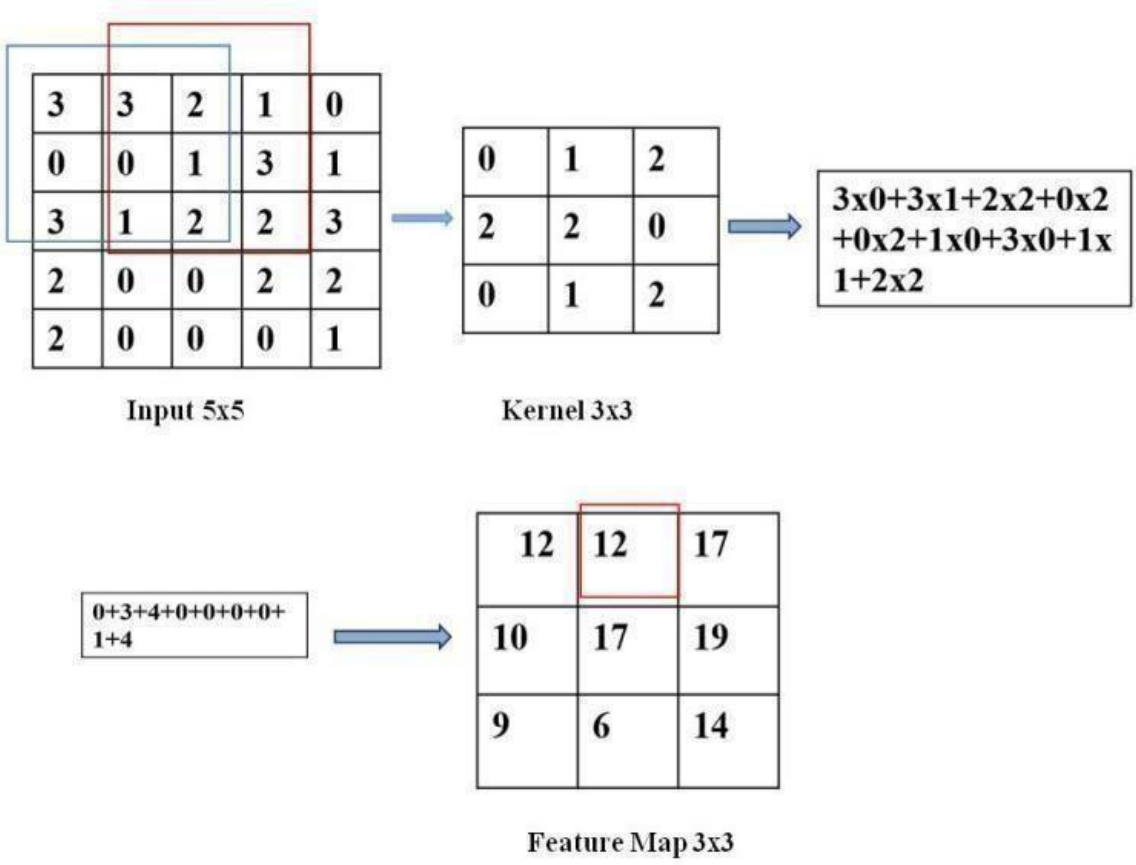


Figure 4. Kernel

Stride

The amount of filter movement is called stride. It moved across the input image from left to right, top to bottom, with one-pixel value change on the horizontal position and one-pixel change invertical position. The default value of stride in 1-D is 1 and in 2-D is (1,1) for height and width movement.

Example:

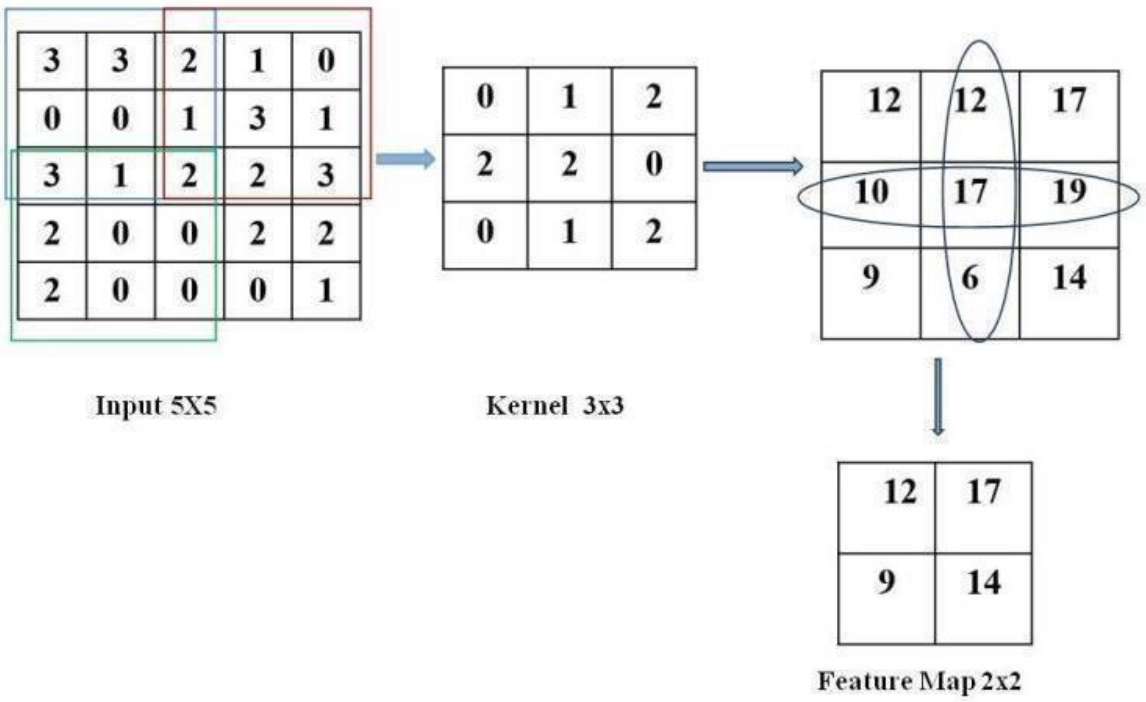


Figure 5. stride

Padding

Padding is mainly used for solving the Border Problem. It is used when the edge pixels play's an important role in classification task. If an input image is padded and then it passed into the CNN model to give the more accurate analysis of images.

Formula

$$O = \left|\frac{i-k+2p}{s}\right| + 1$$

O = output matrix size i = input matrix size

k = kernel matrix size s = stride value

Example

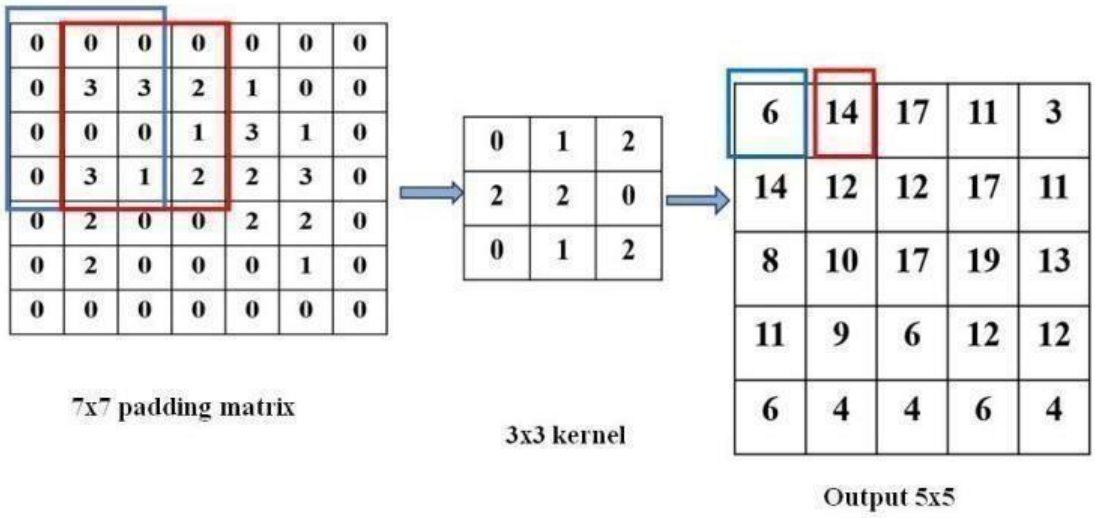


Figure 6. padding

Pooling

Pooling is one of the important components that makes CNN very effective. Pooling is used to reduce the size of input image size in order to reduce the computation power. It has two types of operations.

a. Max pooling

In max pooling it takes the max value from the sub matrix selected and places that max value in the output matrix.

Example:

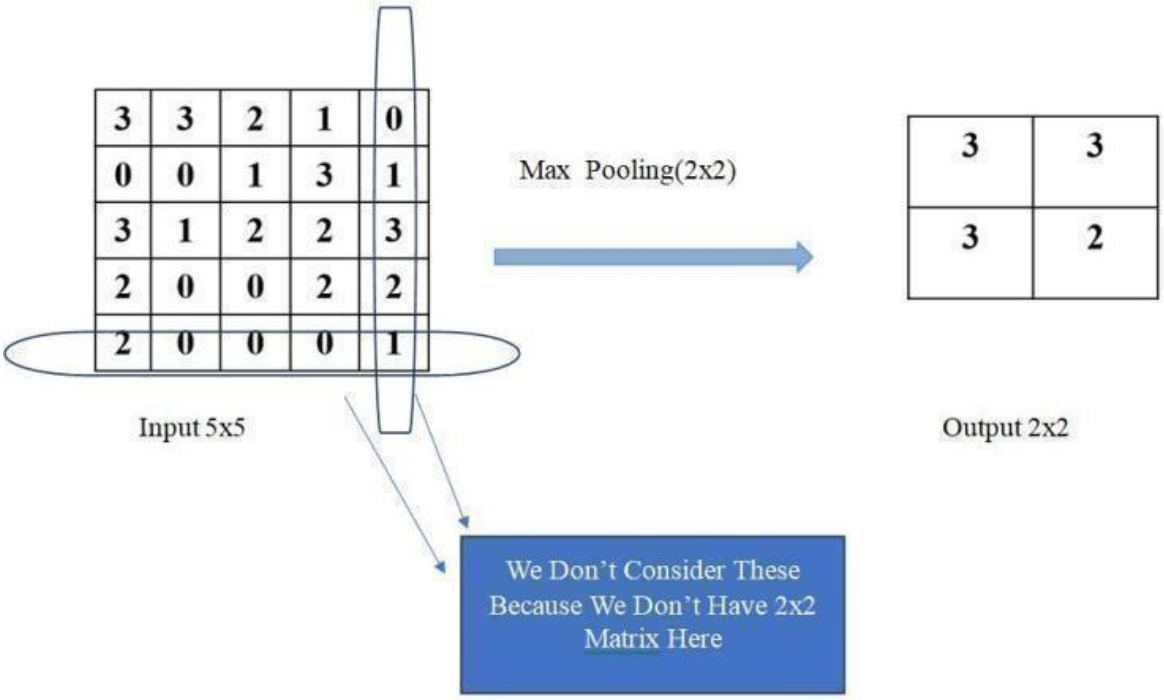


Figure 7. Max Pooling

b. Average pooling

In average pooling it calculates the average value of the sub matrix selected from the input matrix and places in the output matrix.

Example:

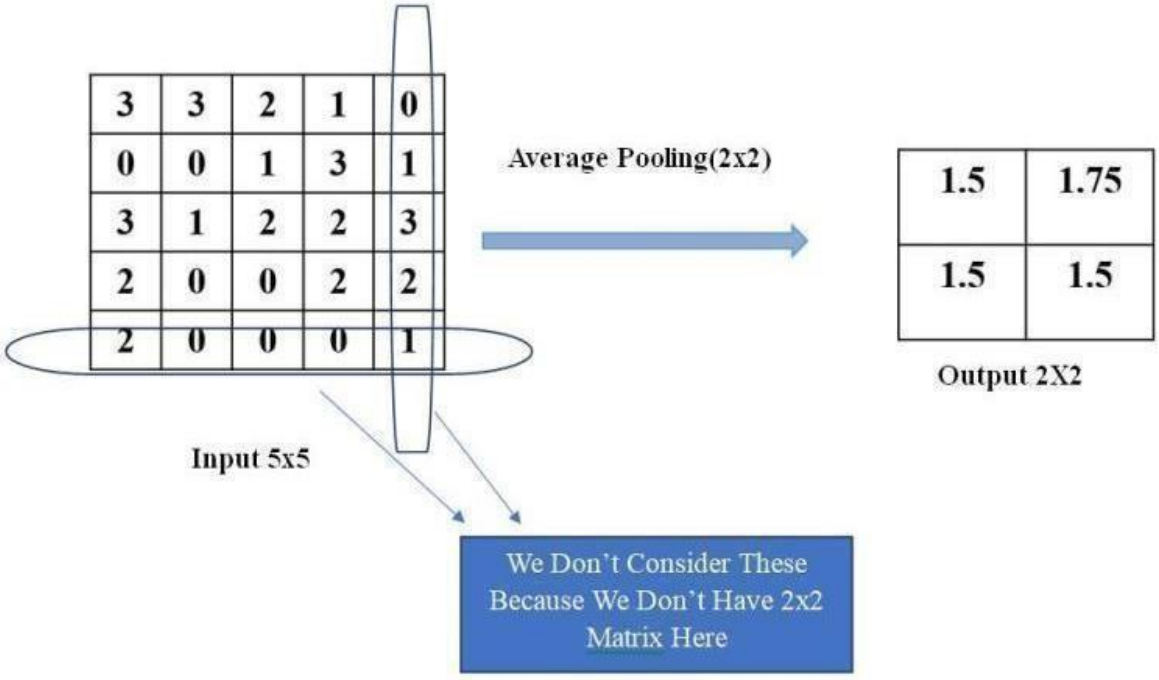


Figure 8. Average pooling

Flattening

A pooled feature map obtained before the flattening layer is passed into the flattening layer in order to make it into a single column matrix which is then given as an input to the neural network for processing.

Example:

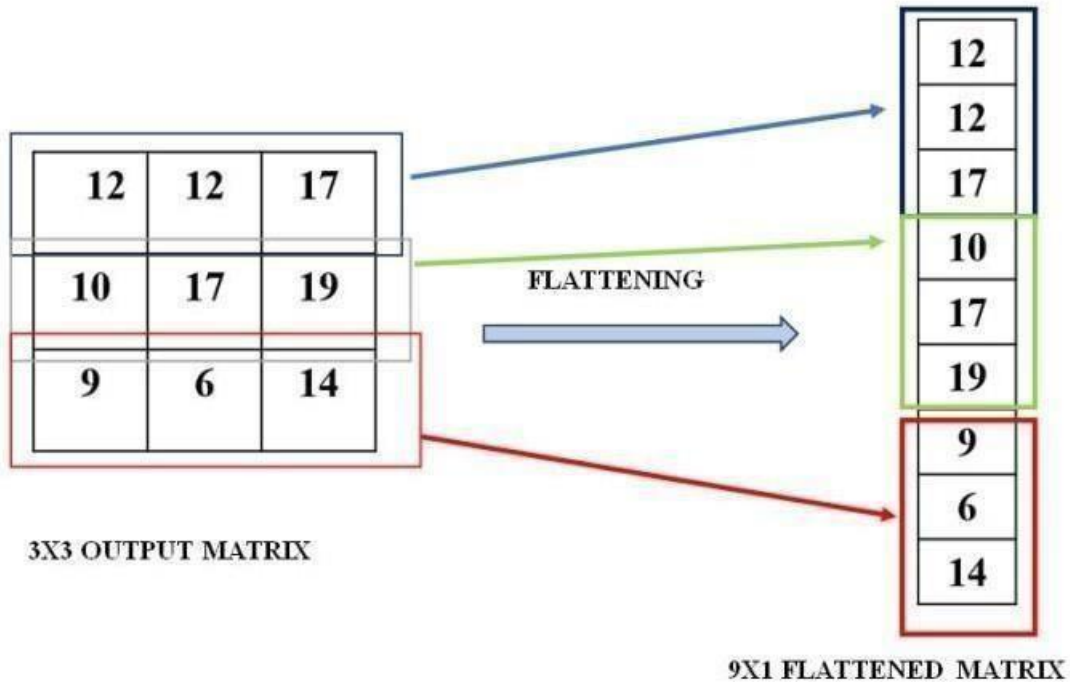


Figure 9. Flattened

Dropout

Dropout is the method of randomly ignoring the neurons. This method is mainly used to prevent the over-fitting problem. Over-fitting means that the neurons the co-dependent of each other. For example dropout (0.2) means it randomly ignores the 20 percent of neurons from the fully connected network

4.2.3 Model Architecture

4.2.3.1 InceptionV3

InceptionV3 is a Convolutional neural network (CNN) architecture designed for image classification and object detection tasks. It was developed by Google as part of the Inception architecture series. The primary goal of InceptionV3 is to improve computational efficiency while maintaining high accuracy in image recognition.

For multilingual image captioning through transformers, InceptionV3 serves as the image encoder. The image encoder's role is to extract relevant features from input images, which can then be used by transformers for generating descriptive captions. Transformers are powerful models for sequence-to-sequence tasks, and they excel at capturing contextual relationships in data.

The combination of InceptionV3 as the image encoder and transformers for caption generation allows the model to understand and represent visual information effectively. InceptionV3 processes the input image, extracting hierarchical features, and

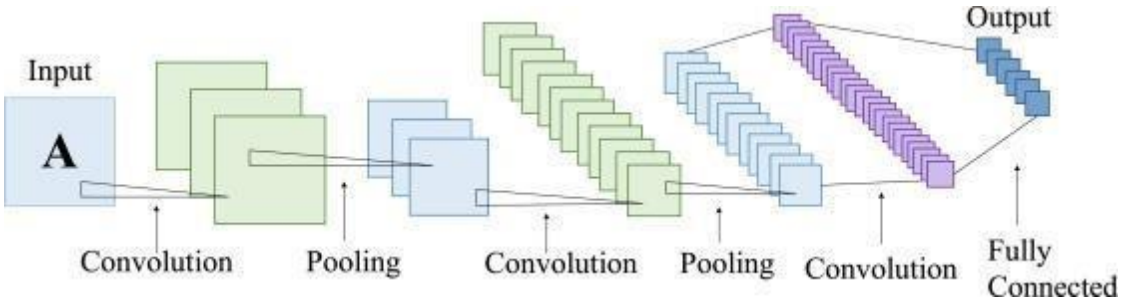


Figure 10 . Architecture diagram of InceptionV3 in image captioning.

the transformer takes these features to generate language-based descriptions. This integration is particularly useful for tasks like multilingual image captioning, where the model needs to comprehend visual content and generate captions in different languages.

4.2.3.2 Architecture of InceptionV3

The InceptionV3 model, designed by Google, represents a breakthrough in image classification models. It employs a unique architecture featuring multiple parallel Convolutional layers of different filter sizes, allowing the model to capture multi-scale features from images effectively. The model is pre-trained on the ImageNet dataset, providing it with a robust understanding of visual features.

Architecture Details :

InceptionV3 is comprised of several inception blocks, each incorporating various convolutions and pooling operations.

Noteworthy components include:

Inception Blocks:

Inception blocks consist of parallel convolutions with different kernel sizes (1x1, 3x3, 5x5), allowing the model to capture features at different scales.
This multi-scale feature extraction enables the model to recognize patterns ranging from fine details to broader structures**.**

Global Average Pooling:

The model replaces fully connected layers with global average pooling, reducing the number of parameters and preventing over-fitting. Global average pooling provides a spatial hierarchy, summarizing the presence of different features across the entire input.

Auxiliary Classifiers:

InceptionV3 introduces auxiliary classifiers during training to combat the vanishing gradient problem. These classifiers help propagate gradients back through the network and facilitate more stable training.

4.2.3.3 InceptionV3 as Image Encoder

A deep Convolutional neural network architecture called InceptionV3 was created for object detection and picture classification applications.It is an expansion of the original GoogLeNet architecture that was introduced by Google in the Inception paper (Szegedy et al., 2015).The image caption system makes use of the InceptionV3 model once more as a "image encoder."

The role of an image encoder is to extract relevant feature representations from an input image that can then be used to create a compact and informative representation of the image.As a deep Convolutional neural network, InceptionV3 can learn hierarchical and abstract characteristics from images.For image classification tasks, Pre-training a model on a big dataset (like ImageNet) allows it to recognize a broad range of objects and patterns.

Output of specific intermediate layer of the InceptionV3 model is used as the image representation or "encoding" in the image captioning model.By extracting features at an intermediate layer, the model captures high-level semantic information about the content of the image, the functionality of using InceptionV3 as an image encoder is encapsulated in the CNN_Encoder function.

The function loads the InceptionV3 model pre-trained on ImageNet, extracts features from an intermediate layer, and reshapes the output to be compatible with subsequent layers in the image captioning model.

4.2.3.4 Transformer Encoder Layer

One of the main components of the Transformer model architecture is the Transformer Encoder Layer.

It is essential to the processing and encoding of the input sequence's information. Typically, the Transformer Encoder Layer is made up of the following parts:

- **Layer Norm (Layer Normalization):**

It applied to the sequence of input before being processed further, normalizes the activation s, which aids in stabilizing the training.

- **Multi-Head Self-Attention:**

A technique that enables the model to analyze each element while focusing on distinct portions of the input sequence allows for the concurrent capture of dependencies between various sequence items. Every head takes care of several positions on their own.

- **Dense Layer:**

The dense (feed-forward) layer comes after the multi-head self-attention layer, makes use of the output of the self-attention layer to power a neural network with a feed-forward structure.The non-linear activation (ReLU) enhances the model's capaci ty to capture complex patterns.

4.2.3.5 Embedding Layers

Words are represented in a continuous vector space by use of embeddings, which is the responsibility of the Embeddings layer.

To capture both the meaning of words and their places in a sequence, it blends positional embeddings with token embeddings.

The following elements usually make up the Embeddings layer:

- **Token Embeddings:**

Usually started with pre-trained word embeddings that have been learned during training (like Glove or Word2Vec).

- **Positional Embeddings:**

Record the tokens in in the proper order as they appear ,essential for the Transformer model to comprehend where each token falls in the sequence.

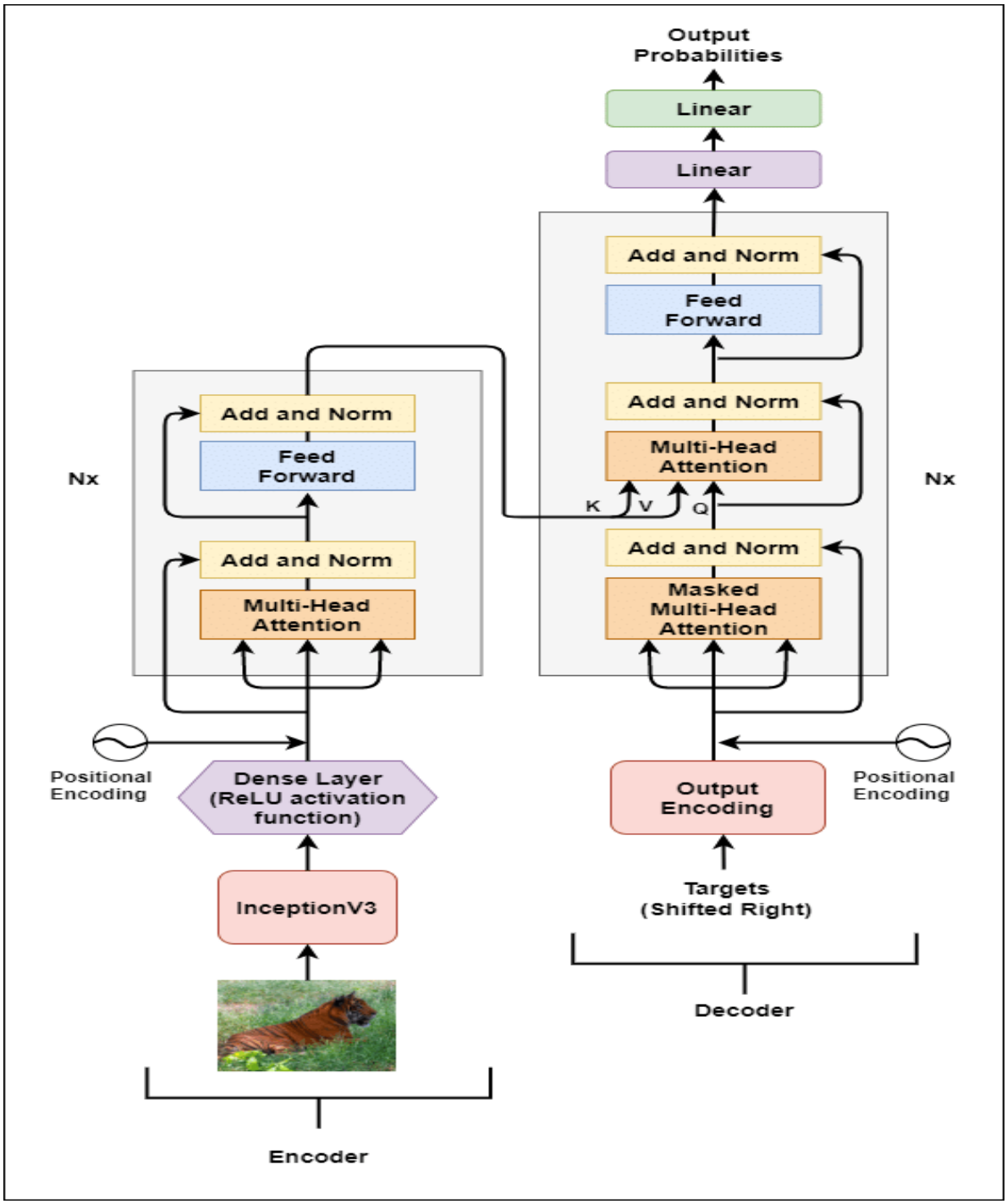


Figure. 11. Architecture diagram of Transformer in image captioning

4.2.3.5 Transformer Decoder Layer

When doing sequence-to-sequence activities, the Transformer Decoder Layer is in charge of processing the destination sequence.

For every token in the target sequence, it creates a context-aware representation by combining cross-attention and self-attention techniques.

Typically, the Transformer Decoder Layer is made up of the following parts:

- **Positional and token embeddings:**

Giving tokens and their places in the target sequence representations, much like the encoder's Embeddings layer does.

- **The Multi-Head Self-Attention (Masked):**

Feature enables every location in the target sequence to provide masked attention to all positions preceding it, including itself.captures dependencies during auto-regressive creation in the target sequence.

4.3.3 Image Captioning Model

The purpose of the Image Captioning Model is to provide insightful captions for theinput photos.

It efficiently processes both visual and sequential data by combining a Transformer based decoder with an image encoder based on neural networks.

The following are the model's essential elements:

- **CNN Image Encoder Model:**

Uses the InceptionV3 model (cnn_model) as an image encoder produces a fixed-size representation by extracting high-level features from the input photos.

- **Transformer Encoder:**

Made up of layers that are stacked one on top of the other gathers contextual data by processing the CNN model's picture embeddings.

- **Transformer Decoder:**

Made up of layers that are piled on top of each other focuses on both the picture attributes and previously produced tokens to generate captions for photos.

Parameters and Fine-Tuning:

Hyperparameters:

MAX_LENGTH: Maximum length of output sequences.

BATCH_SIZE: Number of samples processed in each training iteration.

EMBEDDING_DIM: Dimensionality of token embeddings.

UNITS: Dimensionality of hidden layers in the feed forward networks.

NUM_HEADS: Number of heads in the multi-head attention layers.

Metric

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	CIDEr
Log Biline-ar[9]	70.8	48.9	34.4	24.3	20.03	—
Google NIC[27]	66.6	46.1	32.9	24.3	—	—
Soft-Attention[2]	70.7	49.2	34.4	24.3	23.9	—
Hard- Atten-tion[2]	71.8	50.4	35.7	25.0	23.04	—
PoS[59]	71.1	53.5	38.8	27.9	23.9	88.2
LRCN[10]	66.9	48.9	34.9	24.9	—	—
ATT[1]	70.9	53.7	40.2	30.4	24.3	—
SCA-CNN[60]	71.9	54.8	41.1	31.1	25.0	—
GLA-BEAM33[61]	72.5	55.6	41.7	31.2	24.9	96.4
Ours	**73.1**	**55.9**	**43.4**	**32.8**	**25.49**	**95.1**

Table 1 - Performance comparison on COCO dataset

Model-Specific Parameters:

- InceptionV3 Configuration: Parameters related to the InceptionV3 model.
- Dropout Rates: Used in the transformer decoder layers for regularization.

Fine-Tuning Process:

- Experiment with different learning rates during optimization.
- Adjust the number of layers and units in the transformer.
- Tune the dropout rates to prevent over-fitting.
- Explore different batch sizes.

Monitoring Metrics:

- Evaluate the model using metrics like BLEU, METEOR, and CIDEr.
- Monitor training and validation loss to detect over-fitting or under-fitting.

Iterative Experimentation:

- Fine-tune parameters based on performance on a validation set.
- Iterate through multiple experiments to find the optimal set of hyperparameters.

Considerations Transfer Learning:

- Experiment with using pre-trained weights for the transformer layers.

Learning Rate Schedulers:

- Implement learning rate schedulers to dynamically adjust learning rates during training.

Early Stopping:

Implement early stopping to prevent over-fitting.

Constants and Hyperparameters:

MAX_LENGTH:

Description: Maximum length of the output sequence (caption).

Role: Defines the maximum number of tokens in the generated captions.

BATCH_SIZE:

Description: Number of samples processed in each training iteration.

Role: Influences how many data points are processed together during each optimization step.

BUFFER_SIZE:

Description: Size of the buffer used for shuffling the dataset.

Role: Determines the number of elements from the dataset that are loaded into memory for shuffling.

EMBEDDING_DIM:

Description: Dimensionality of the token embeddings.

Role: Specifies the size of the vector space in which words are embedded.

UNITS:

Description: Dimensionality of the hidden layers in the feed forward networks**.Role:** Controls the size of the hidden layers in the transformer decoder.

VOCABULARY_SIZE (commented out):

Description: Size of the vocabulary.

Role: Specifies the maximum number of unique words in the vocabulary. (Commented out in the provided code)

INCEPTIONV3_WEIGHTS:

Description: Specifies the weight initialization for the InceptionV3 model**.**

Role: Determines whether to use pre-trained weights from ImageNet ('imagenet') or random initialization ('None').

DROPOUT_RATES:

Description: Dropout rates used for regularization**.**

Role: Controls the probability of dropping out units during training to prevent over-fitting.

NUM_HEADS:

Description: Number of heads in the multi-head attention layers.

Role: Determines the number of parallel attention heads in the self-attention mechanism.

CHAPTER 5

CODING & IMPLEMENTATION

5. CODING & IMPLEMENTATION

Model.py

```
import pickle
import tensorflow as tf
import pandas as pd
import numpy as np

# CONTANTS
MAX_LENGTH = 40
# VOCABULARY_SIZE = 10000
BATCH_SIZE = 32
BUFFER_SIZE = 1000
EMBEDDING_DIM = 512
UNITS = 512

# LOADING DATA
vocab = pickle.load(open('vocab_coco.file', 'rb'))
tokenizer = tf.keras.layers.TextVectorization(
    # max_tokens=VOCABULARY_SIZE,
    standardize=None,
    output_sequence_length=MAX_LENGTH,
    vocabulary=vocab
)

#mapping of words to integer indices, and this layer will provide the reverse
mapping from integer indices to words.
idx2word = tf.keras.layers.StringLookup(
```

```
    mask_token="",
    vocabulary=tokenizer.get_vocabulary(),
    invert=True
)

# MODEL
def CNN_Encoder():
    inception_v3 = tf.keras.applications.InceptionV3(
        include_top=False,
        weights='imagenet'
    )

    output = inception_v3.output
    output = tf.keras.layers.Reshape(
        (-1, output.shape[-1]))(output)

    cnn_model = tf.keras.models.Model(inception_v3.input, output)
    return cnn_model

#Stabilizing Training: It helps stabilize the training process by mitigating the
vanishing gradient problem.
#Facilitating Learning: By normalizing the input before further processing, the
network is better able to learn complex patterns and relationships in the data.
class TransformerEncoderLayer(tf.keras.layers.Layer):
    def__init_(self, embed_dim, num_heads):
        super()._init_()
        self.layer_norm_1 = tf.keras.layers.LayerNormalization()
        self.layer_norm_2 = tf.keras.layers.LayerNormalization()
        self.attention = tf.keras.layers.MultiHeadAttention(
```

```
            num_heads=num_heads, key_dim=embed_dim)
        self.dense = tf.keras.layers.Dense(embed_dim, activation="relu")

    def call(self, x, training):
        x = self.layer_norm_1(x)
        x = self.dense(x)

        attn_output = self.attention(
            query=x,
            value=x,
            key=x,
            attention_mask=None,
            training=training
        )
#the model focuses on learning what should be added (or subtracted) to x to obtain
the final output.
        x = self.layer_norm_2(x + attn_output)
        return x

class Embeddings(tf.keras.layers.Layer):
    def__init_(self, vocab_size, embed_dim, max_len):
        super()._init_()
        self.token_embeddings = tf.keras.layers.Embedding(
            vocab_size, embed_dim)
        #It's used to add position information to the token embeddings
        # Position embeddings are essential in transformer models to capture the
sequential order of tokens.
        self.position_embeddings = tf.keras.layers.Embedding(
            max_len, embed_dim, input_shape=(None, max_len))
```

```
    def call(self, input_ids):
        #This method defines how the layer processes its input during a forward pass through the network. input_ids are the token IDs of the input sequence.
        length = tf.shape(input_ids)[-1]
        position_ids = tf.range(start=0, limit=length, delta=1)
        #Expands the dimensions of position_ids to make it compatible with the shape of input_ids.
        position_ids = tf.expand_dims(position_ids, axis=0)

        token_embeddings = self.token_embeddings(input_ids)
        position_embeddings = self.position_embeddings(position_ids)
      #Combines the token embeddings and position embeddings by element-wise addition.
        return token_embeddings + position_embeddings

class  TransformerDecoderLayer(tf.keras.layers.Layer):

    def_init_(self, embed_dim, units, num_heads):
        super()._____init_____()
        self.embedding = Embeddings(
            tokenizer.vocabulary_size(), embed_dim, MAX_LENGTH)
        #Self-attention is often used in the decoder to model autoregressive relationships, where the generation of each token depends on the previously generated tokens in the sequence.
        #self.attention_1 is used for attending to the decoder's own input sequence.
        self.attention_1 = tf.keras.layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=embed_dim, dropout=0.1
        )
```

```
#This means that each token in the decoder can pay attention to different parts of the encoder's output sequence. It allows the decoder to align itself with the relevant information from the input sequence produced by the encoder.
        self.attention_2 = tf.keras.layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=embed_dim, dropout=0.1
        )
       #Layer normalization helps stabilize training and improve the flow of gradients.
        self.layernorm_1 = tf.keras.layers.LayerNormalization()
        self.layernorm_2 = tf.keras.layers.LayerNormalization()
        self.layernorm_3 = tf.keras.layers.LayerNormalization()

        self.ffn_layer_1 = tf.keras.layers.Dense(units, activation="relu")
        self.ffn_layer_2 = tf.keras.layers.Dense(embed_dim)

        self.out = tf.keras.layers.Dense(tokenizer.vocabulary_size(), activation="softmax")
      #Dropout is used to prevent overfitting during training.
        self.dropout_1 = tf.keras.layers.Dropout(0.3)
        self.dropout_2 = tf.keras.layers.Dropout(0.5)

    def call(self, input_ids, encoder_output, training, mask=None):
        embeddings = self.embedding(input_ids)
        combined_mask = None
        padding_mask = None

        if mask is not None:
            causal_mask = self.get_causal_attention_mask(embeddings)
            padding_mask = tf.cast(mask[:, :, tf.newaxis], dtype=tf.int32)
            combined_mask = tf.cast(mask[:, tf.newaxis, :], dtype=tf.int32)
```

```
        combined_mask = tf.minimum(combined_mask, causal_mask)

    attn_output_1 = self.attention_1(
        query=embeddings,
        value=embeddings,
        key=embeddings,
        attention_mask=combined_mask,
        training=training
    )

    out_1 = self.layernorm_1(embeddings + attn_output_1)

    attn_output_2 = self.attention_2(
        query=out_1,
        value=encoder_output,
        key=encoder_output,
        attention_mask=padding_mask,
        training=training
    )

    out_2 = self.layernorm_2(out_1 + attn_output_2)

    ffn_out = self.ffn_layer_1(out_2)
    ffn_out = self.dropout_1(ffn_out, training=training)
    ffn_out = self.ffn_layer_2(ffn_out)

    ffn_out = self.layernorm_3(ffn_out + out_2)
    ffn_out = self.dropout_2(ffn_out, training=training)
    preds = self.out(ffn_out)
```

```
        return preds

#tokens are generated one at a time, and each token's generation depends on the previously generated tokens.
    def get_causal_attention_mask(self, inputs):
        input_shape = tf.shape(inputs)
        batch_size, sequence_length = input_shape[0], input_shape[1]
        i = tf.range(sequence_length)[:, tf.newaxis]
        j = tf.range(sequence_length)
        mask = tf.cast(i >= j, dtype="int32")
        mask = tf.reshape(mask, (1, input_shape[1], input_shape[1]))
        mult = tf.concat(
            [tf.expand_dims(batch_size, -1), tf.constant([1, 1], dtype=tf.int32)],
            axis=0
        )
        return tf.tile(mask, mult)

class ImageCaptioningModel(tf.keras.Model):
    def__init_(self, cnn_model, encoder, decoder, image_aug=None):
        super()._init_()
        self.cnn_model = cnn_model
        self.encoder = encoder
        self.decoder = decoder
        self.image_aug = image_aug
        self.loss_tracker = tf.keras.metrics.Mean(name="loss")
        self.acc_tracker = tf.keras.metrics.Mean(name="accuracy")

     #calculate_loss and calculate_accuracy are helper methods used to compute the loss and accuracy, respectively.
```

```
def calculate_loss(self, y_true, y_pred, mask):
    loss = self.loss(y_true, y_pred)
    mask = tf.cast(mask, dtype=loss.dtype)
    loss *= mask
    return tf.reduce_sum(loss) / tf.reduce_sum(mask)

def calculate_accuracy(self, y_true, y_pred, mask):
    accuracy = tf.equal(y_true, tf.argmax(y_pred, axis=2))
    accuracy = tf.math.logical_and(mask, accuracy)
    accuracy = tf.cast(accuracy, dtype=tf.float32)
    mask = tf.cast(mask, dtype=tf.float32)
    return tf.reduce_sum(accuracy) / tf.reduce_sum(mask)
def compute_loss_and_acc(self, img_embed, captions, training=True):
    encoder_output = self.encoder(img_embed, training=True)
    y_input = captions[:, :-1]
    y_true = captions[:, 1:]
    mask = (y_true != 0)
    y_pred = self.decoder(
        y_input, encoder_output, training=True, mask=mask
    )
    loss = self.calculate_loss(y_true, y_pred, mask)
    acc = self.calculate_accuracy(y_true, y_pred, mask)
    return loss, acc

def train_step(self, batch):
    imgs, captions = batch

    if self.image_aug:
```

```
            imgs = self.image_aug(imgs)

        img_embed = self.cnn_model(imgs)

        with tf.GradientTape() as tape:
            loss, acc = self.compute_loss_and_acc(
                img_embed, captions
            )

        train_vars = (
            self.encoder.trainable_variables + self.decoder.trainable_variables
        )
        grads = tape.gradient(loss, train_vars)
        self.optimizer.apply_gradients(zip(grads, train_vars))
        self.loss_tracker.update_state(loss)
        self.acc_tracker.update_state(acc)

        return {"loss": self.loss_tracker.result(), "acc": self.acc_tracker.result()}
    def test_step(self, batch):
        imgs, captions = batch

        img_embed = self.cnn_model(imgs)

        loss, acc = self.compute_loss_and_acc(
            img_embed, captions, training=False
        )

        self.loss_tracker.update_state(loss)
        self.acc_tracker.update_state(acc)
```

```
        return {"loss": self.loss_tracker.result(), "acc": self.acc_tracker.result()}

    @property
    def metrics(self):
        return [self.loss_tracker, self.acc_tracker]

def load_image_from_path(img_path):
    img = tf.io.read_file(img_path)
    img = tf.io.decode_jpeg(img, channels=3)
    img = tf.keras.layers.Resizing(299, 299)(img)
    #This line performs additional preprocessing specific to the InceptionV3 model.
    img = tf.keras.applications.inception_v3.preprocess_input(img)
    return img

def generate_caption(img, caption_model, add_noise=False):
    if isinstance(img, str):
        img = load_image_from_path(img)

    if add_noise == True:
        noise = tf.random.normal(img.shape)*0.1
        img = (img + noise)
        img = (img - tf.reduce_min(img))/(tf.reduce_max(img) - tf.reduce_min(img))

    img = tf.expand_dims(img, axis=0)
    img_embed = caption_model.cnn_model(img)
    img_encoded = caption_model.encoder(img_embed, training=False)
    y_inp = '[start]'
    for i in range(MAX_LENGTH-1):
```

```
        tokenized = tokenizer([y_inp])[:, :-1]
        mask = tf.cast(tokenized != 0, tf.int32)
        pred = caption_model.decoder(
            tokenized, img_encoded, training=False, mask=mask)

        pred_idx = np.argmax(pred[0, i, :])
        #The predicted word is obtained by finding the index of the word
        pred_word = idx2word(pred_idx).numpy().decode('utf-8')
        if pred_word == '[end]':
            break

        y_inp += ' ' + pred_word
    y_inp = y_inp.replace('[start] ', '')
    return y_inp

def get_caption_model():
    encoder = TransformerEncoderLayer(EMBEDDING_DIM, 1)
    decoder = TransformerDecoderLayer(EMBEDDING_DIM, UNITS, 8)
    cnn_model = CNN_Encoder()
    caption_model = ImageCaptioningModel(
        cnn_model=cnn_model, encoder=encoder, decoder=decoder,
image_aug=None,
    )

    def call_fn(batch, training):
        return batch
    caption_model.call = call_fn
    sample_x, sample_y = tf.random.normal((1, 299, 299, 3)), tf.zeros((1, 40))
```

```
    caption_model((sample_x,  sample_y))
    sample_img_embed = caption_model.cnn_model(sample_x)
    sample_enc_out = caption_model.encoder(sample_img_embed, training=False)
    caption_model.decoder(sample_y, sample_enc_out, training=False)
    try:
        caption_model.load_weights('image_captioning_coco_weights.h5')
    except FileNotFoundError:
        caption_model.load_weights('image_captioning_coco_weights.h5')
    return caption_model
```

App.py:

```
import io
import os
import streamlit as st
import requests
from PIL import Image
from model import get_caption_model, generate_caption
from translate import Translator
@st.cache(allow_output_mutation=True)
def get_model():
    return  get_caption_model()
caption_model = get_model()
def predict():
    #captions = []
    pred_caption = generate_caption('tmp.jpg', caption_model)
    st.markdown('#### Predicted Captions:')
    st.write(pred_caption)
```

```
    translator = Translator(to_lang="te")
    translation = translator.translate(pred_caption)
    st.write(translation)
    translator = Translator(to_lang="hi")
    translation = translator.translate(pred_caption)
    st.write(translation)
    translator = Translator(to_lang="chi")
    translation = translator.translate(pred_caption)
    st.write(translation)
st.title('Image Captioner')
img_url = st.text_input(label='Enter Image URL')
if (img_url != "") and (img_url != None):
    img = Image.open(requests.get(img_url, stream=True).raw)
    img = img.convert('RGB')
    st.image(img)
    img.save('tmp.jpg')
    predict()
    os.remove('tmp.jpg')
st.markdown('<center style="opacity: 70%">OR</center>',
unsafe_allow_html=True)
img_upload = st.file_uploader(label='Upload Image', type=['jpg', 'png', 'jpeg'])
if img_upload != None:
    img = img_upload.read()
    img = Image.open(io.BytesIO(img))
    img = img.convert('RGB')
    img.save('tmp.jpg')
    st.image(img)
    predict()
    os.remove('tmp.jpg')
```

CHAPTER 6

SYSTEM TESTING

6. SYSTEM TESTING

6.1 Overview of Testing

In any software development, testing is a process to show the correctness of program and it needs the design specifications. Testing is needed to prove correctness completeness, to improve the quality of the software and to provide the maintenance aid. Some testing standards are therefore necessary to ensure completeness of testing, improve the quality of software and reduce the testing costs and to reduce study needs and operation time.

6.2 Goals of Testing

The following are goals of testing…

Testing is a process of executing a program with the intent of finding error.A good test case is the one that has a high probability of finding an as at undiscovered error.A successful test is one that uncovers an as at undiscovered error.

6.3 Types of Testings

6.3.1 Black box testing

Black Box Testing is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application. Usually Test Engineers are involved in the black box testing.

6.3.2 White box testing

White Box Testing is the testing process in which tester can perform testing on an application with having internal structural knowledge. Usually the developers are involved in the white box testing.

6.3.3 Gray box testing

Gray Box Testing is the process in which the combination of black box and white box techniques is use.

6.4 Levels of Testing

6.4.1 Unit testing

Individual components are tested to ensure that they operate correctly. Each component is tested independently without other system components.

6.4.2 System testing

The subsystems are integrated to make up the entire system. The testing process is concerned with finding errors, which result from unanticipated interactions between subsystem components.

6.4.3 Integration testing

Sometimes global data structures can represent the problems to uncover errors that are associated with interfacing the objective is to make unit test modules and built a program structure that has been detected by design.

6.4.4 Acceptance testing

This is the final stage in the testing process before the system is accepted for operational use. Acceptance testing may reveal errors and omissions in the system requirements definition because real data exercises the system in different ways from the test data.

6.4.5 Regression testing

Regression testing is actually that helps to ensure changes that don't introduce unintended behavior as additional errors. Regression testing may be conducted manually by executing a subset of all test cases or using automated capture play back tools.

6.5 Unit Test Cases

Test case Id	Input	Description	Expected Results	Pass/Fail
TC_01	Upload image	When Admin Uploads train image	Correct Caption Generated	Pass
TC_02	Upload Image	When Admin uploads test Image	Correct Caption Generated	Pass
TC_03	Upload Image	When Admin uploads validation Image	Correct Caption Generated	Pass
TC_04	Upload Image	When Admin uploads Unseen Image	Correct Caption Generation	Pass
TC_05	Not Upload Image	When Admin Does not uploads Image	No Caption Generation	Fail
TC_06	Upload Image	When Admin again uploads unseen Image	Correct Caption Generated	Pass

Table 2 Test cases for image uploading.

CHAPTER 7

RESULTS

7. RESULTS

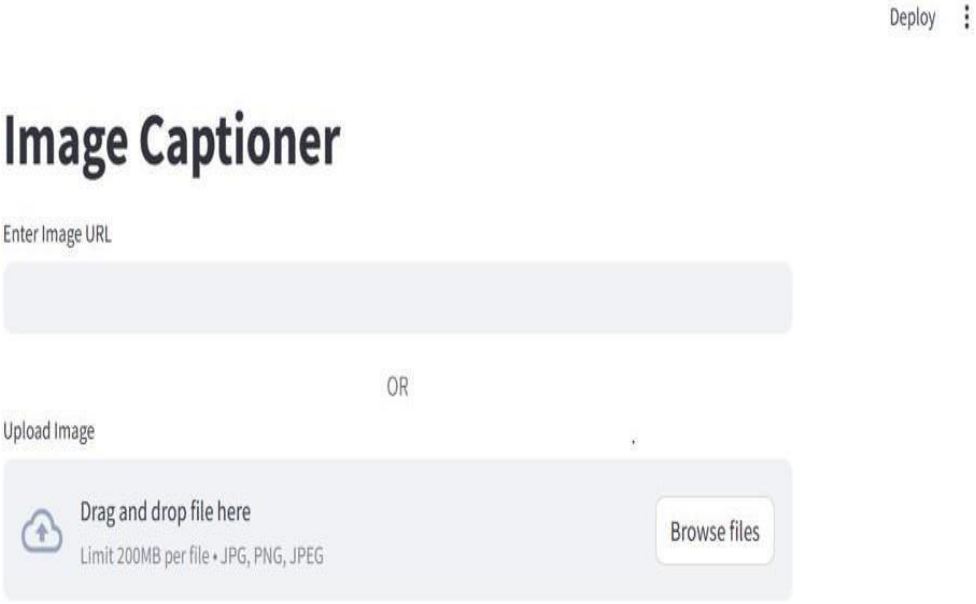


Figure 12 Screenshot of Output .

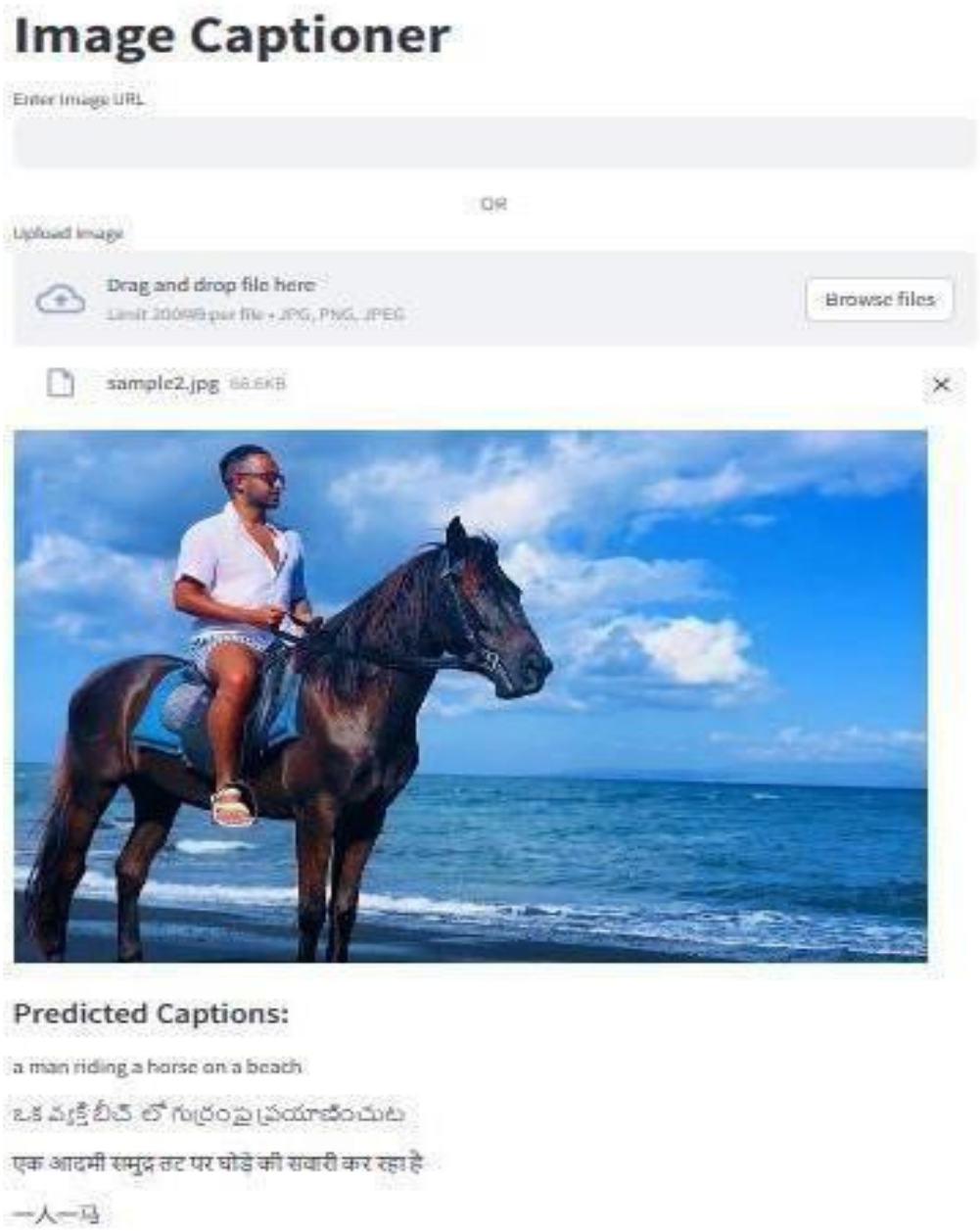


Figure 13 Screenshot of Multilingual Image Caption Generation

Figure 14 Screenshot of Multilingual Image Caption Generation

Figure 15 Screenshot of Multilingual Image Caption Generation

Figure 16 Screenshot of Multilingual Image Caption Generation

Figure 17 Screenshot of Multilingual Image Caption Generation

CHAPTER 8

CONCLUSION

8. CONCLUSION

8.1 CONCLUSION

In this project Transformer architectures have made multilingual picture captioning possible, which is a revolutionary development. By combining transformer-based encoders and decoders with CNNs (Convolutional neural networks) for image recognition, new possibilities for the creation of multilingual, meaningful captions for images have been uncovered.

8.2 FUTURE SCOPE

In future in order to give more detailed captioning and to improve the captioning task, Examine sophisticated cross-modal attention mechanisms to enhance comprehension of the connections between textual and visual information. Improving cross-modal comprehension results in captions that are more precise and contextually rich.To enable equitable comparisons between algorithms, standardize the benchmarks and assessment measures for multilingual picture captioning tasks.

Ensure equitable representation across languages and cultures by addressing ethical concerns about bias and justice in multilingual picture captioning.

CHAPTER 9

REFERENCES

9. REFERENCES

[1] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, ‘‘Image captioning with semantic attention,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 4651–4659.

[2] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Salakhudinov, R. Zemel, and Y. Bengio, ‘‘Show, attend and tell: Neural image caption generation with visual attention,’’ in Proc. Int. Conf. Mach. Learn., vol. 2015, Feb. 2015, pp. 2048–2057.

[3] S. Bai and S. An, ‘‘A survey on automatic image caption generation,’’ Neurocomputing, vol. 311, pp. 291–304, Oct. 2018.

[4] V. Ordonez, G. Kulkarni, and T. L. Berg, ‘‘Im2Text: Describing images using 1 million captioned photographs,’’ in Proc. Adv. Neural Inf. Process. Syst., 2011, pp. 1143–1151.

[5] M. Hodosh, P. Young, and J. Hockenmaier, ‘‘Framing image description as a ranking task: Data, models and evaluation metrics,’’ J. Artif. Intell. Res., vol. 47, pp. 853–899, Aug. 2013.

[6] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, ‘‘Learning phrase representations using RNN encoder-decoder for statistical machine translation,’’ in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), Oct. 2014, pp. 1–15.

[7] D. Bahdanau, K. Cho, and Y. Bengio, ‘‘Neural machine translation by jointly learning to align and translate,’’ 2014, arXiv:1409.0473. [Online]. Available: https://arxiv.org/abs/1409.0473

[8] Y. LeCun, Y. Bengio, and G. Hinton, ‘‘Deep learning,’’ Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[9] R. Kiros, R. Salakhutdinov, and R. Zemel, ‘‘Multimodal neural language models,’’ in Proc. Int. Conf. Mach. Learn., 2014, pp. 595–603.

[10] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, ‘‘Long-term recurrent convolutional networks for visual recognition and description,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 4, pp. 677–691, Apr. 2017.

[11] P. Anderson, B. Fernando, and M. Johnson, ‘‘SPICE: Semantic propositional image caption evaluation,’’ Adapt. Behav., vol. 11, no. 4, pp. 382–398, 2016.

[12] S. Kuanar, V. Athitsos, N. Pradhan, A. Mishra, and K. R. Rao, ‘‘Cognitive analysis of working memory load from eeg, by a deep recurrent neural network,’’ in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Apr. 2018, pp. 2576–2580.

[13] Y. Wu et al., ‘‘Google’s neural machine translation system: Bridging the gap between human and machine translation,’’ 2016, arXiv:1609.08144. [Online]. Available: https://arxiv.org/abs/1609.08144

[14] J. Liang, L. Jiang, L. Cao, Y. Kalantidis, L.-J. Li, and A. G. Hauptmann, ‘‘Focal visual-text attention for memex question answering,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 8, pp. 1893–1908, Aug. 2019.

[15] J. Lu, C. Xiong, D. Parikh, and R. Socher, ‘‘Knowing when to look: Adaptive attention via a visual sentinel for image captioning,’’ in Proc. IEEE Conf. Comput. Vis.

Pattern Recognit. (CVPR), Jul. 2017, pp. 375–383.

[16] Y. Wu, L. Zhu, L. Jiang, and Y. Yang, ‘‘Decoupled novel object captioner,’’ in Proc. ACM Multimedia Conf., 2018, pp. 1029–1037.

[17] W. Lan, X. Li, and J. Dong, ‘‘Fluency-guided cross-lingual image captioning,’’ in ACM Multimedia. New York, NY, USA: ACM, 2017.

[18] A. Zirikly, ‘‘Cross-lingual transfer of named entity recognizers without parallel corpora,’’ in Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process., 2015, pp. 390–396.

[19] T. Miyazaki and N. Shimizu, ‘‘Cross-lingual image caption generation,’’in Proc. Meeting Assoc. Comput. Linguistics, 2016, pp. 1780–1790.

[20] E. Ghanbari and A. Shakery, ‘‘Query-dependent learning to rank for cross-lingual information retrieval,’’ Knowl. Inf. Syst., vol. 59, no. 3, pp. 711–743, 2019.

9.2 Sites Referred

- https://www.analyticsvidhya.com/blog/2021/12/step-by-step-guide-to-build-image-caption-generator-using-deep-learning/

- https://medium.com/analytics-vidhya/cnn-lstm-architecture-and-image-captioning-2351fc18e8d7

- https://medium.com/analytics-vidhya/introduction-to-image-caption-generation-using-the-avengers-infinity-war-characters-6f14df09dbe5

- https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d

Multilingual image captioning through Transformers

Dr.O.Rama Devi[1],P.Nagalakshmi[2],SK.Ibraheem[3],Y.Divya Reddy[4]
[1]Professor,[2,3,4]Students
[1,2,3,4]Department of Artificial Intelligence and Data Science,
[1,2,3,4]Lakireddy Bali Reddy College of Engineering,Mylavaram.
E-mail: pariminagalakshmi2003@gmail.com , odugu.rama@gmail.com

***Abstract*—Image captioning is a challenging task in artificial intelligence,It attempts to produce descriptive and cohesive straightforward captions for photographs. Traditional picture captioning models have predominantly focused on single-language scenarios, limiting their applicability in a globalized world with diverse linguistic contexts. This research proposes a novel approach to multilingual image captioning, leveraging the power of transformer .With the advent of transformer-based architectures, the task of automatically producing descriptive textual summaries for images has advanced significantly. In this work, we present a novel image captioning model that makes use of transformer encoder and decoder layers. The transformer architecture, which was initially created for natural language processing tasks, has shown to be highly effective in capturing long-range dependencies and global context within the visual domain. Our model processes image embeddings using a transformer encoder and generates coherent and contextually relevant captions using a transformer decoder. The incorporation of multi-head attention mechanisms improves the model's capacity to focus on salient image features, resulting in more informative captions.We conduct comprehensive experiments on benchmark datasets, demonstrating superior performance in comparison to traditional architectures. Additionally, our model exhibits versatility in handling diverse image content and generalizes well to previously unseen data.**

***Index Terms*—Transformers ; encoder layer ; decoder layer ;image captioning ; multi-head attention.**

I. INTRODUCTION

Image captioning is a challenging task in artificial intelligence,It attempts to produce descriptive and cohesive straightforward captions for photographs.. Traditional picture captioning models have predominantly focused on single-language scenarios, limiting their applicability in a globalized world with diverse linguistic contexts. This research proposes a novel approach to multilingual image captioning, leveraging the power of transformer .With the advent of transformer-based architectures, the task of automatically producing descriptive textual summaries for images has advanced significantly. In this work, we present a novel image captioning model that makes use of transformer encoder and decoder layers. The transformer architecture, which was initially created for natural language processing tasks, has shown to be highly effective in capturing long-range dependencies and global context within the visual domain.Our model processes image embeddings using a transformer encoder and generates coherent and contextually relevant captions using a transformer decoder. The incorporation of multi-head attention mechanisms improves the model's capacity to focus on salient image features, resulting in more informative captions.We conduct comprehensive experiments on benchmark datasets, demonstrating superior performance in comparison to traditional architectures. Additionally, our model exhibits versatility in handling diverse image content and generalizes well to previously unseen data.

II. LITERATURE SURVEY

In the research of "Image Captioning with Semantic Attention"[1] Methodology involved is to Developed a model with the help of Semantic Analysis,Caption Generation Model,Semantic Attention Mechanism.And the Algorithms used are:CNN, RNN.The Results identified are image captioning job, that offers superior results across popular standard benchmarks.And the disadvantages identified includes Lack of Semantic Understanding,Data Limitations,Evaluation Metrics. On working with "Show, Attend and Tell" Visual Encoder Methodology is utilized for Neural Image Caption Generation with Visual Attention [2].,Attention Mechanism,CaptionDecoder.The Algorithms used are:CNN,Visual Attention Mechanism,LSTM. The Results identified are Training this model both stochastically by maximizing a variational lower limit and deterministically by applying conventional backpropagation techniques[2].The disadvantages identified includes Lack of Contextual Alignment,Handling Variable Length,Contextual Relevance. While working on image captioning"A survey on automatic image caption generation"[3] ,Methodology involved are Deep neural network-driven captioning of images.The Algorithms used are:encoder-decoder framework[3].The Results identified are Showing samples of picture captioning outcomes created using various methodologies to offer readers a clear impression of several types of image caption methods.The disadvantages identified includes Diversity of Generated Captions, Bias and Fairness. "Describing Images Using 1 MillionCaptioned Photographs" ,Methodology used are Image Feature Extraction, Caption Preprocessing, Text-Image Association[4].The Algorithms used are:CNN,Neural Network Training.The findings show that in order to create a unique web-scale annotated photo collection, this technique depends on gathering and sorting a sizable data set of photos from the internet. We describe two versions of our method: one that generates captions solely using global picture descriptors, and another that also includes estimations of image content..The disadvantages identified includes Diversity of Captions, Handling Ambiguity, Multimodal Understanding. "Framing Image Description as a Ranking TaskData, Models and Evaluation Metrics"[5] ,Methodologies are Ranking-Based Image Description,Feature Extraction, Semantic Similarity Metrics.Using a technique known as Kernel Canonical Correlation Analysis (KCCA), the algorithms aimed to project texts and pictures into a shared latent'semantic' space.[5].The Results identified are Ranking-based evaluations are increasingly often employed in image description publications, and we continue to wonder whether BLEU or ROUGE scores are relevant given that they do not closely correlate with human judgments.The limitations identified includes User-Centric Evaluation,Generalization Across Datasets,Bias and Fairness.

III. METHODOLOGY

3.1. Dataset

Coco dataset is the data set used for image captioning.Common Objects in COntext, or COCO, is a well-known benchmark dataset for the processing of natural languages and machine vision. It was made and is maintained by Microsoft,and it is commonly used for image captioning, object detection, and segmentation.The dataset contains a vast number of photos, each of which depicts common things in diverse circumstances.The images are high-resolution and depict a wide variety of scenes, objects, and activities.Each image in the COCO dataset has extensive annotations, such as object annotations, instance segmentation masks, and captions.

a. Object annotations define the boundaries of objects in a picture.
b. Objects are segmented at the pixel level using instance segmentation masks.
c. Captions are written descriptions of the image's content.

The COCO dataset is widely recognized for its contribution to picture captioning challenges. It is frequently used to train and assess models capable of producing descriptive captions for pictures.The vocab_coco.file file is most likely a pre-processed vocabulary file for the COCO image captioning task.
The file is most likely a serialized version of a Python data structure, such as a dictionary or similar appropriate structure.The dictionary's keys may represent words, while the values may represent numerical indices or additional information.
The vocabulary file is most likely created during the COCO dataset pre-processing phase or during the initial configuration of the image captioning model.

3.2. Model architecture

3.2.1. InceptionV3 as image encoder

A deep convolutional neural network architecture called InceptionV3 was created for object detection and picture classification applications.It is an expansion of the original GoogLeNet architecture that was introduced by Google in the Inception paper (Szegedy et al., 2015).The image caption system makes use of the InceptionV3 model once more as a "image encoder."
The role of an image encoder is to extract relevant feature representations from an input image that can then be used to create a compact and informative representation of the image.As a deep convolutional neural network, InceptionV3 can learn hierarchical and abstract characteristics from images.For image classification tasks, Pre-training a model on a big dataset (like ImageNet) allows it to recognize a broad range of objects and patterns.Output of specific intermediate layer of the InceptionV3 model is used as the image representation or "encoding" in the image captioning model.By extracting features at an intermediate layer, the model captures high-level semantic information about the content of the image.the functionality of using InceptionV3 as an image encoder is encapsulated in the CNN_Encoder function.The function loads the InceptionV3 model pre-trained on ImageNet, extracts features from an intermediate layer, and reshapes the output to be compatible with subsequent layers in the image captioning model.

3.2.2. Transformer encoder layer

One of the main components of the Transformer model architecture is the Transformer Encoder Layer.
It is essential to the processing and encoding of the input sequence's information.
Typically, the Transformer Encoder Layer is made up of the following parts:

a. Layer Norm (Layer Normalization):
applied to the sequence of input before being processed further.
normalizes the activation s, which aids in stabilizing the training.
b. Multi-Head Self-Attention:
A technique that enables the model to analyze each element while focusing on distinct portions of the input sequence.
allows for the concurrent capture of dependencies between various sequence items.
Every head takes care of several positions on their own.
c. Dense Layer:
The dense (feed-forward) layer comes after the multi-head self-attention layer,
makes use of the output of the self-attention layer to power a neural network with a feed-forward structure.The non-linear activation (ReLU) enhances the model's capacity to capture complex patterns.

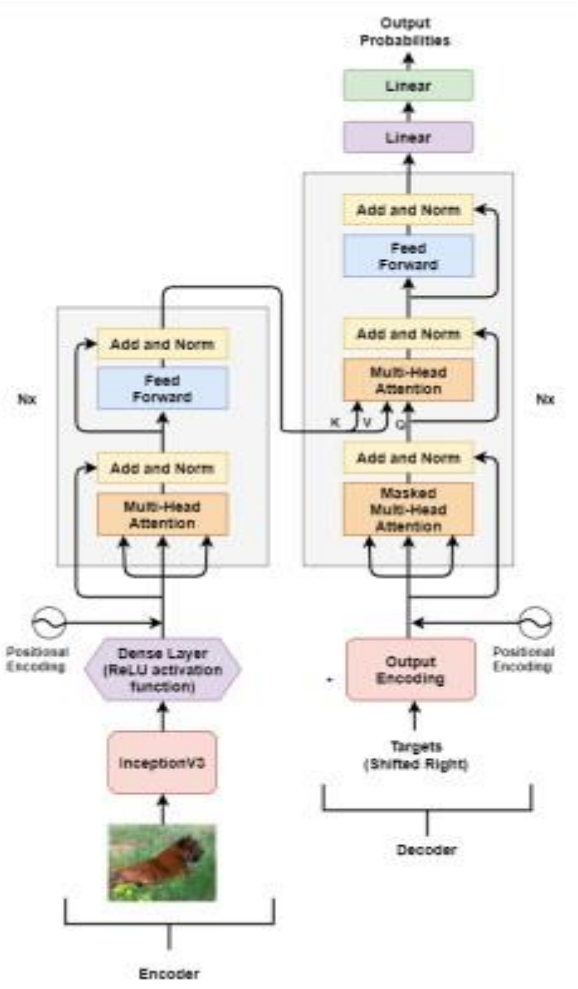


Fig.1: Architecture diagram of Transformer in image captioning

3.2.3. Embedding layers

Words are represented in a continuous vector space by use of embeddings, which is the responsibility of the Embeddings layer.
To capture both the meaning of words and their places in a sequence, it blends positional embeddings with token embeddings.
The following elements usually make up the Embeddings layer:
a. Token Embeddings:
Describe the significance of certain terms.
Usually started with pre-trained word embeddings that have been learned during training (like GloVe or Word2Vec).
b. Positional Embeddings:

Record the tokens in in the proper order as they appear ,essential for the Transformer model to comprehend where each token falls in the sequence.

3.2.4. Transfromer decoder layer

When doing sequence-to-sequence activities, the Transformer Decoder Layer is in charge of processing the destination sequence. For every token in the target sequence, it creates a context-aware representation by combining cross-attention and self-attention techniques.
Typically, the Transformer Decoder Layer is made up of the following parts:
a. Positional and token embeddings:
Giving tokens and their places in the target sequence representations, much like the encoder's Embeddings layer does.
b. The Multi-Head Self-Attention (Masked):
Feature enables every location in the target sequence to provide masked attention to all positions preceding it, including itself. captures dependencies during auto-regressive creation in the target sequence.

3.2.5. Image captioning Model

The purpose of the Image Captioning Model is to provide insightful captions for the input photos.
It efficiently processes both visual and sequential data by combining a Transformer-based decoder with an image encoder based on neural networks.
The following are the model's essential elements:
i. CNN Image Encoder Model:
Uses the InceptionV3 model (cnn_model) as an image encoder. produces a fixed-size representation by extracting high-level features from the input photos.
ii. Transformer Encoder:
Made up of layers that are stacked one on top of the other. gathers contextual data by processing the CNN model's picture embeddings.
iii. Transformer Decoder:
Made up of layers that are piled on top of each other. focuses on both the picture attributes and previously produced tokens to generate captions for photos.

IV. RESULTS

Performance Evalution:Our model which is a combination of Convolution Neural Network and Transformers,giving accuracy of 95% .The result gives the image with different languages with exact meaning.

	Metric					
Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	CIDEr
Log Bilinear[9]	70.8	48.9	34.4	24.3	20.03	—
Google NIC[27]	66.6	46.1	32.9	24.3	—	—
Soft-Attention[2]	70.7	49.2	34.4	24.3	23.9	—
Hard-Attention[2]	71.8	50.4	35.7	25.0	23.04	—
PoS[59]	71.1	53.5	38.8	27.9	23.9	88.2
LRCN[10]	66.9	48.9	34.9	24.9	—	—
ATT[1]	70.9	53.7	40.2	30.4	24.3	—
SCA-CNN[60]	71.9	54.8	41.1	31.1	25.0	—
GLA-BEAM33[61]	72.5	55.6	41.7	31.2	24.9	96.4
Ours	**73.1**	**55.9**	**43.4**	**32.8**	**25.49**	**95.1**

Fig. 2: Performance comparison on COCO dataset.

Fig. 3: Output image which gives the caption for uploaded image in different languages.

V. CONCLUSION AND FUTURE ENHANCEMENT

In this paper Transformer architectures have made multilingual picture captioning possible, which is a revolutionary development. By combining transformer-based encoders and decoders with CNNs (convolutional neural networks) for image recognition, new possibilities for the creation of multilingual, meaningful captions for images have been uncovered.
In future inorder to give more detailed captioning and to improve the captioning task, Examine sophisticated cross-modal attention mechanisms to enhance comprehension of the connections between textual and visual information. Improving cross-modal comprehension results in captions that are more precise and contextually rich.To enable equitable comparisons between algorithms, standardize the benchmarks and assessment measures for multilingual picture captioning tasks.Ensure equitable representation across languages and cultures by addressing ethical concerns about bias and justice in multilingual picture captioning.

REFERENCES

[1] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, ‘‘Image captioning with semantic attention,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 4651–4659.
[2] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Salakhudinov, R. Zemel, and Y. Bengio, ‘‘Show, attend and tell: Neural image caption generation with visual attention,’’ in Proc. Int. Conf. Mach. Learn., vol. 2015, Feb. 2015, pp. 2048–2057.
[3] S. Bai and S. An, ‘‘A survey on automatic image caption generation,’’ Neurocomputing, vol. 311, pp. 291–304, Oct. 2018.

[4] V. Ordonez, G. Kulkarni, and T. L. Berg, ‘‘Im2Text: Describing images using 1 million captioned photographs,’’ in Proc. Adv. Neural Inf. Process. Syst., 2011, pp. 1143–1151.
[5] M. Hodosh, P. Young, and J. Hockenmaier, ‘‘Framing image description as a ranking task: Data, models and evaluation metrics,’’ J. Artif. Intell. Res., vol. 47, pp. 853–899, Aug. 2013.
[6] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H .Schwenk, and Y. Bengio, ‘‘Learning phrase representations using RNN encoder-
decoder for statistical machine translation,’’ in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), Oct. 2014, pp. 1–15.
[7] D. Bahdanau, K. Cho, and Y. Bengio, ‘‘Neural machine translation by jointly learning to align and translate,’’ 2014, arXiv:1409.0473. [Online]. Available: https://arxiv.org/abs/1409.0473
[8] Y. LeCun, Y. Bengio, and G. Hinton, ‘‘Deep learning,’’ Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[9] R. Kiros, R. Salakhutdinov, and R. Zemel, ‘‘Multimodal neural language models,’’ in Proc. Int. Conf. Mach. Learn., 2014, pp. 595–603.
[10] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, ‘‘Long-term recurrent convolutional networks for visual recognition and description,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 4, pp. 677–691, Apr. 2017.
[11] P. Anderson, B. Fernando, and M. Johnson, ‘‘SPICE: Semantic propositional image caption evaluation,’’ Adapt. Behav., vol. 11, no. 4, pp. 382–398, 2016.
[12] S. Kuanar, V. Athitsos, N. Pradhan, A. Mishra, and K. R. Rao, ‘‘Cognitive analysis of working memory load from eeg, by a deep recurrent neural network,’’ in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Apr. 2018, pp. 2576–2580.
[13] Y. Wu et al., ‘‘Google’s neural machine translation system: Bridging the gap between human and machine translation,’’ 2016, arXiv:1609.08144. [Online]. Available: https://arxiv.org/abs/1609.08144
[14] J. Liang, L. Jiang, L. Cao, Y. Kalantidis, L.-J. Li, and A. G. Hauptmann, ‘‘Focal visual-text attention for memex question answering,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 8, pp. 1893–1908, Aug. 2019.
[15] J. Lu, C. Xiong, D. Parikh, and R. Socher, ‘‘Knowing when to look: Adaptive attention via a visual sentinel for image captioning,’’ in Proc. IEEE Conf.
Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 375–383.
[16] Y. Wu, L. Zhu, L. Jiang, and Y. Yang, ‘‘Decoupled novel object captioner,’’
in Proc. ACM Multimedia Conf., 2018, pp. 1029–1037.
[17] W. Lan, X. Li, and J. Dong, ‘‘Fluency-guided cross-lingual image captioning,’’ in ACM Multimedia. New York, NY, USA: ACM, 2017.
[18] A. Zirikly, ‘‘Cross-lingual transfer of named entity recognizers without parallel corpora,’’ in Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int.Joint Conf. Natural Lang. Process., 2015, pp. 390–396.
[19] T. Miyazaki and N. Shimizu, ‘‘Cross-lingual image caption generation,’’in Proc. Meeting Assoc. Comput. Linguistics, 2016, pp. 1780–1790.
[20] E. Ghanbari and A. Shakery, ‘‘Query-dependent learning to rank for cross-lingual information retrieval,’’ Knowl. Inf. Syst., vol. 59, no. 3, pp. 711–743, 2019.

TOSHVIN REMI sim INFOSYSTEMS SERB DIA Science and Engineering Research Board (SERB) Govt. of India Metrohm India Private Ltd. ACS Publications Most Trusted. Most Cited. Most Read.

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY
(MANIT) BHOPAL M.P

"International Conference on Sustainable Energy & Environment"

(ICSEE 2024)

CERTIFICATE OF APPRECIATION

This is to certify that Dr./Prof./Mr./Ms **O.RAMA DEVI, SHAIK IBRAHEEM, YARAMALA DIVYA REDDY**

has Participated and presented a paper ***(Oral/Poster)*** *entitled*

MULTILINGUAL IMAGE CAPTIONING THROUGH TRANSFORMERS

at the "International Conference on Sustainable Energy & Environment" organized by the Department of Chemical Engineering, Maulana Azad National Institute of Technology,Bhopal from 23-25th feb. 2024

Dr. Sumit H. Dhawane
Coordinator, ICSEE 2024

Dr. Bharat Modhera
Coordinator, ICSEE 2024

TOSHVIN REMI sim INFOSYSTEMS SERB DIA Science and Engineering Research Board (SERB) Govt. of India Metrohm India Private Ltd. ACS Publications Most Trusted. Most Cited. Most Read.

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY
(MANIT) BHOPAL M.P

"International Conference on Sustainable Energy & Environment"
(ICSEE 2024)

CERTIFICATE OF APPRECIATION

This is to certify that Dr./Prof./Mr./Ms **PARIMI NAGALAKSHMI**

has Participated and presented a paper ***(Oral/Poster)*** *entitled*

MULTILINGUAL IMAGE CAPTIONING THROUGH TRANSFORMERS

at the "International Conference on Sustainable Energy & Environment" organized by the Department of Chemical Engineering, Maulana Azad National Institute of Technology,Bhopal from 23-25th 2024

Dr. Sumit H. Dhawane
Coordinator, ICSEE 2024

Dr. Bharat Modhera
Coordinator, ICSEE 2024

Batch01 Doc Plag check

ORIGINALITY REPORT

21%	20%	6%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	huggingface.co Internet Source	11%
2	vdocuments.site Internet Source	3%
3	ncuindia.edu Internet Source	1%
4	www.coursehero.com Internet Source	1%
5	lbrce.ac.in Internet Source	1%
6	Submitted to Anglia Ruskin University Student Paper	<1%
7	2dc40e33-085f-40e0-8172-9a1f898c1942.filesusr.com Internet Source	<1%
8	Submitted to University of East London Student Paper	<1%
9	Xing Wang, Zhaopeng Tu, Min Zhang. "Incorporating Statistical Machine Translation	<1%