A Project Report on

A NEURAL IMAGE CAPTION GENERATOR

Submitted in partial fulfillment for award of

Bachelor of Technology

Degree

in

Computer Science and Engineering

By

S. Sai Sree (Y20ACS553)   **Y. Pravallika (Y20ACS591)**

Y. Santhoshini (Y20ACS592)

Under the guidance of
Dr. N. Sudhakar, M. Tech, Ph.D.
Professor

Department of Computer Science and Engineering
Bapatla Engineering College
(Autonomous)
(Affiliated to Acharya Nagarjuna University)
BAPATLA – 522 102, Andhra Pradesh, INDIA
2023-2024

Department of
Computer Science and Engineering

CERTIFICATE

This is to certify that the project report entitled **A Neural Image Caption Generator** that is being submitted by S. Sai Sree (Y20ACS553), Y. Pravallika (Y20ACS591), Y. Santhoshini (Y20ACS592) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide	**Signature of the HOD**
Guide Name	**Dr. M. Rajesh Babu**
Designation	**Assoc. Prof. & Head**

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

S.Sai Sree (Y20ACS553)
Y.Pravallika (Y20ACS591)
Y.Santhoshini (Y20ACS592)

Acknowledgement

Abstract

In today's world of social media, almost everyone is a part of social platform and actively interacting with each other through internet. People on social media upload many pictures on their social media accounts with different captions. Thinking about the appropriate caption is a tedious process. Caption is important to effectively describe the content and meaning of a picture.

Image captioning is one of the applications of Deep Learning which involves fusion of work done in computer vision and natural language processing, and it is typically performed using Encoder-Decoder architecture. In this work, the aim is to improve the performance and explain ability of generating captions for images of different types and resolutions. A pretrained CNN (convolution neural network) model and an NLP model are used in the concept of encoder-decoder along with Attention network to build this model.

A pretrained CNN is used for image feature extraction purpose where only the important features are extracted from the resultant image. NLP model is used to predict the next word in the sentence. For capturing the relevant context from the image and generating more accurate prediction for the next word in the caption sequence Attention is used. The Flickr8k dataset is used to train and test the model.

Keywords- Image, Caption, RNN, CNN, Xception, VGG16, Densenet201, LSTM, GRU, Attention, Neural Networks

Table of Contents

List of Tables

List of Figures

List of Abbreviations

NLP	Natural Language Processing
DL	Deep Learning
ML	Machine Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
VGG	Visual Geometry Group
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit

1 Introduction

The image captioning method generates textual descriptions for a given input image which gives an actual understanding of the image provided. The objective is to teach a deep convolutional neural network and NLP model to understand the content of an image and produce a textual description that accurately reflects the visual content and context depicted in the image.

1.1 Introduction

In the past few years, computer vision in the image processing area has made significant progress, like image classification [1] and object detection [2]. Benefiting from the advances of image classification and object detection it becomes possible to automatically generate captions to understand visual content of an image. The goal of image captioning is to automatically generate descriptions for a given image.

Making a computer system detect objects and describe them using natural language processing (NLP) in an age-old problem of Artificial Intelligence. This was considered an impossible task by computer vision researchers till now. With the growing advancements in Deep learning techniques, availability of vast datasets, and computational power, models are often built which will generate captions for an image.

Image caption generation is a task that involves image processing and natural language processing concepts to recognize the context of an image and describe them in a natural language like English or any other language. It generates syntactically and semantically correct sentences. In this paper, we present a deep learning model

to describe images and generate captions using computer vision and machine translation. These generators can find applications in Image segmentation as used by Facebook and Google Photos, and even more so, its use can be extended to video frames. They will easily automate the job of a person who has to interpret images.
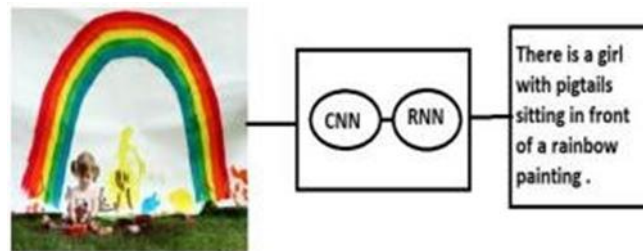
Figure 1.1 Our model is based on a deep learning neural network that consists of a pretrained CNN model followed by a language generating NLP model. It generates complete sentences in natural language as an output

A computer system takes input images as two-dimensional arrays and mapping is done from images to captions or descriptive sentences. In recent years a lot of attention has been drawn towards the task of automatically generating captions for images.The Figure 1.1 shows the basic model and algorithms used. However, while new datasets often spur considerable innovation, benchmark datasets also require fast, accurate, and competitive evaluation metrics to encourage rapid progress. Being able to automatically describe the content of a picture using properly formed English sentences may be a very challenging task, but it could have an excellent impact, as an example by helping visuallyimpaired people better understand the content of images online.

Deep learning has attracted a lot of attention because it's particularly good at a kind of learning that has the potential to be very useful for real-world applications. The ability to find out from unlabeled or unstructured data is a huge benefit for those curious about real-world applications.

1.2 Objective

The main objective of the image caption generation is to teach a deep convolution neural network and NLP model to understand the content of an image and produce a textual description that accurately reflects the visual content and context depicted in the any given image.

1.3 Problem Statement

Artificial Intelligence (AI) is now at the heart of innovation economy and thus is the base for our work. The main problem in the development of image description started with object detection using static object class libraries in the image and modelled using statistical language models.

1) Making use of CNN: It's a Deep Learning algorithm that will intake in a 2D matrix input image, assign importance (learnable weights and biases) to different aspects/objects in the image, and be intelligent enough to be able to differentiate one from the other.

2) This model was advantageous in naming the objects in an image but it could not tell us the relationship among them (that's plain image classification).

3) Present a generative model built on a deep recurrent architecture that unites recent advances in computer vision and machine translation and that can effectively generate meaningful sentences.

4) Making use of an RNN: They are networks with loops in them, allowing information to persist. LSTMs are a particular kind of RNN, capable of learning long-term dependencies.

5) Further an attention layer is used along with the RNN model to generate more relevant captions for any given image.

1.4 Feature Extraction

Feature extraction is a part of the dimensionality reduction process, in which, an initial set of the raw data is divided and reduced to more manageable groups. So, when you want to process it will be easier. The most important characteristic of these large data sets is that they have many variables. These variables require a lot of computing resources to process. So, Feature extraction helps to get the best feature from those big data sets by selecting and combining variables into features, thus, effectively reducing the amount of data. These features are easy to process, but still able to describe the actual data set with accuracy and originality.

The technique of extracting the features is useful when you have a large data set and need to reduce the number of resources without losing any important or relevant information. Feature extraction helps to reduce the amount of redundant data from the data set.

1.5 Machine Learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine learning algorithms are typically created using frameworks that accelerate solution development, such as TensorFlow and PyTorch.

1.5.1 Encoder

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is

particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes. The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information. The output of the pooling layers is then passed through one or morefully connected layers, which are used to make a prediction or classify the image.

1.5.2 Decoder

Long Short-Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is specifically designed to handle sequential data, such as time series, speech, and text. LSTM networks can learn long-term dependencies in sequential data, which makes them well suited for tasks such as language translation, speech recognition, and time series forecasting.

The Gated Recurrent Unit (GRU) is another type of Recurrent Neural Network (RNN) that in certain cases, has advantages over long short term memory (LSTM). GRU uses less memory and is faster than LSTM, however, LSTM is more accurate when using datasets with longer sequences.

1.5.3 Attention

The attention mechanism is a complex cognitive ability that human beings possess.

When people receive information, they can consciously ignore some of the main information while ignoring other secondary information. This ability of self-selection is called attention. The attention mechanism allows the neural network to have the ability to focus on its subset of inputs to select specific features. The attention mechanism the model to pay attention to certain parts of the data and to give them more weight when making predictions. It allows a nutshell, the attention mechanism helps preserve the context of every word in a sentence by assigning an attention weight relative to all other words.

The attention mechanism consists of three layers. The encoder creates a hidden state at each step. The attention component generates a context vector that captures the relevant information from the encoder's hidden states. The context vector is fed into the decoder along with the current hidden state, to predict the next token.

2 Literature Survey

In related work the relevant information is enhanced on prior study on image caption generation and attention. Several approaches for getting image descriptions have recently been presented. Automatic image captioning generation has emerged as a promising research area in recent years, because to advances in deep neural network models for Computer Vision (CV) and Natural Language Processing (NLP).

In this section we provide relevant background on previous work on image caption generation and attention. Recently, several methods have been proposed for generating image descriptions. Many of these methods are based on recurrent neural networks and inspired by the successful use of sequence-to-sequence training with neural networks for machine translation. The encoder-decoder framework of machine translation is well suited, because it is analogous to "translating" an image to a sentence.

In the paper [3] proposed by O. Vinyals, A. Toshev, S. Bengio and D. Erhan (2015) used CNN and RNN for image captioning with different datasets and analysed the results.

In the paper [4] proposed by P. Voditel, A. Gurjar, A. Pandey, A. Jain, N. Sharma and N. Dubey (2023) used a specific pretrained model called VGG16 which is trained on ImageNet dataset which has more than 14 million images and a LSTM NLP model to generate captions for images

In the paper [5] proposed by A. Aker and R. Gaizauskas (2010) used N-grams to generate image descriptions using dependency relational patterns. They used a simple model based on N-Gram graphs which does not require any end-to-end training on paired image captions is proposed. Starting with a set of image keywords considered

as nodes, the generator is designed to form a directed graph by connecting these nodes through overlapping n-grams as found in a given text corpus. The model then infers the caption by maximizing the most probable n-gram sequences from the constructed graph

In the paper [6] proposed by Haoran Wang, Yue Zhang, Xiaosheng Yu (2020) used the attention mechanism to perform the image caption generation. An attention mechanism can be used to improve the contextual aspect of natural language sequences. The use of attention to describe image content is consistent with human understanding.

In our work we created of an image captioning models that employs an NLP model with a soft attention decoder to predict the future sentences by selectively focusing over a specific parts of an image. We used different pretrained models VGG16, Xception, Densenet201 and different NLP models LSTM, GRU to analysis the performance of each of its combinations and identified which gives better captions for any given image. We did analysis with the help of BLEU score and Meteor score.

3 Technology Used

The Technology used in building the project image captioning method are as follows.

1)   Python:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming

2)   Jupyter Notebook:

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

3)   Google Colab:

Colaboratory, or Colab for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

4)   Python Libraries:

The python libraries used in our work are as follows.

1. Pandas

2. Numpy

3. Matplotlib

4. Keras

5. PIL

6. Tqdm

7. OS module

8. Nltk

9. String

10. Pickle

11. TensorFlow

4 Existing System

Techniques of Pre-trained Convolutional Neural Networks and a type of Recurrent Neural Network are combined together for image captioning.

CNN acts as an encoder that maps input images to a fixed-length feature vector.

The feature vector combined with the caption text which is fed to the RNN are passed through fully connected layers.

RNN acts as a decoder to decode the fixed-length vector into a caption.

4.1 Drawbacks

There are many drawbacks of RNN some are listed as follows

4.1.1 Vanishing Gradient Problem

RNNs often suffer from vanishing gradient problems, where gradients diminish as they are backpropagated through time, making it difficult for the model to learn long-range dependencies.

4.1.2 Limited Short-Term Memory

Traditional RNNs have a tendency to forget information from earlier time steps when processing long sequences. This limits their ability to capture long-term dependencies in sequential data.

5 Proposed System

Deep learning and AI based model are proposed to implement image captioning using CNNand LSTM along with Visual Attention. Image Captioning is the process of generating textual description of an image. It uses both Natural Language Processing and Computer Vision to generate the captions. The dataset will be in the form

[image → captions]

5.1 Task

The task is to build a system that will take an image input in the form of a dimensional array and generate an output consisting of a sentence that describes the image and is syntactically and grammatically correct.

We analyze different methods of performing image captioning. They are

Pretrained models: VGG16, Xception, Densenet

NLP models: LSTM, GRU, GRU+Attention.

VGG16		LSTM
Xception	+	GRU
Densenet		Attention+GRU

VGG16+LSTM

VGG16+GRU

VGG16+GRU+Attention

Xception+LSTM

Xception+GRU

Xception+GRU+Attention

Densenet+LSTM

Densenet+GRU

Densenet+GRU+Attention

These are the combinations of NLP models and pretrained models used for generating image caption generation.

5.2 Corpus

We have used the Flickr 8K dataset as the corpus. The dataset consists of 8000 images andfor every image, there are 5 captions. The 5 captions for a single image help in understanding all the various possible scenarios. The dataset has a predefined training dataset Flickr_8k.trainImages.txt (6,000images), development dataset Flickr_8k.devImages.txt (1,000 images), and test dataset Flickr_8k.testImages.txt (1,000 images).

Generating a caption for a given image is a challenging problem in the deep learningdomain. It is small in size. So, the model can be trained easily on low-end laptops/desktops.Data is properly labelled. For each image 5 captions are provided. The dataset is available for free. It is a new bookmark collection for sentence-base image description and search. It doesn't touch original photos that we upload.
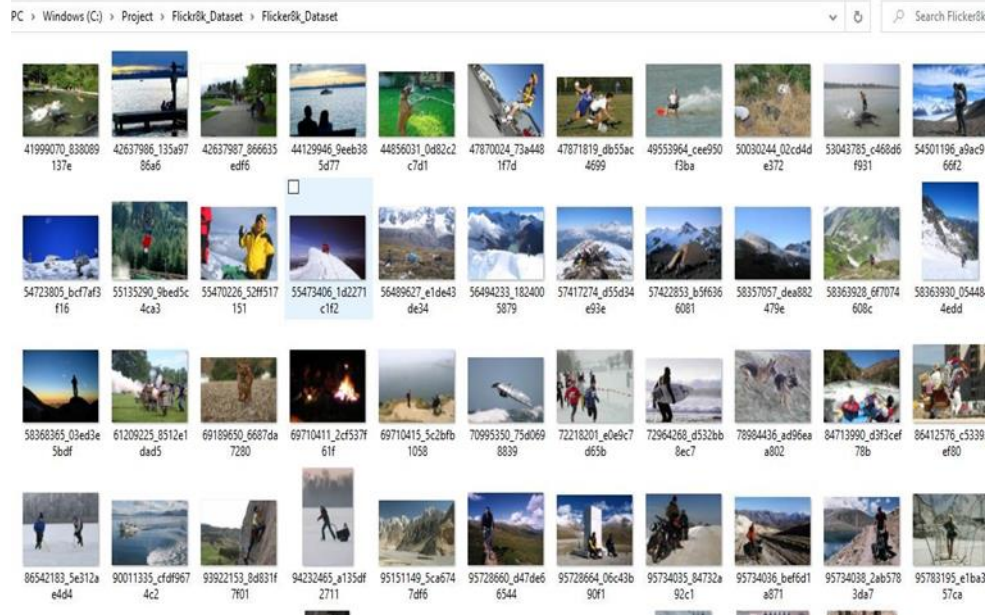
Figure 5.1 Glimpse of the Flickr8k Image Dataset

Figure 5.2 Glimpse of Flickr8k Text File

The architecture proposed is an encoder-decoder attention-based architecture. The encoder is one of the Xception, Densenet201, VGG16 models which is pretrained on ImageNet dataset. The decoder is one of the LSTM, GRU models. As shown in Figure 5.1 images are present in Flickr8k_dataset folder and Figure 5.2 shows a glimpse of how each image is associated with 5 captions.

14

5.3 Preprocessing

Data preprocessing is done in two parts, the images and the corresponding captions are cleaned and pre-processed separately. Image preprocessing is done by feeding the input data to the CNN pretrained application of the Keras API running on top of TensorFlow. Pretrained CNN model is pre-trained on ImageNet. This helped us train the images faster with the help of Transfer learning. The descriptions are cleaned using the tokenizer class in Keras, this will vectorize the text corpus and is stored in a separate dictionary. Then each word of the vocabulary is mapped with a unique index value.

5.4 Model

Deep learning uses an artificial neural network that is composed of several levels arranged in a hierarchy. The model is based on deep networks where the flow of information starts from the initial level, where the model learns   something simple and then the output of it is passed to layer two of the network and input is combined into something that is a bit more complex and passes it on to the third level. This process continues as each level in the network produces something more complex from the input it received from the ascendant level.

5.4.1 Convolutional Neural Networks (CNN)

Convolutional Neural networks are specialized deep neural networks that can process the data that has input shape like a 2D matrix. Images can be easily represented as a 2D matrix. CNN is crucial in working with images. It takes an image as input, assigns importance (weights and biases) to various aspects/objects in the image, and differentiates one from the other. The below Figure 4.3 demonstrates the architecture of CNN. The CNN makes use of filters which help in feature learning same   as a human

brain identifying objects in time and space. It has convolutional layer applies a set of learnable filters to the input image for extracting features hierarchically and Pooling layers reduce spatial dimensions. This architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and the reusability of weights.
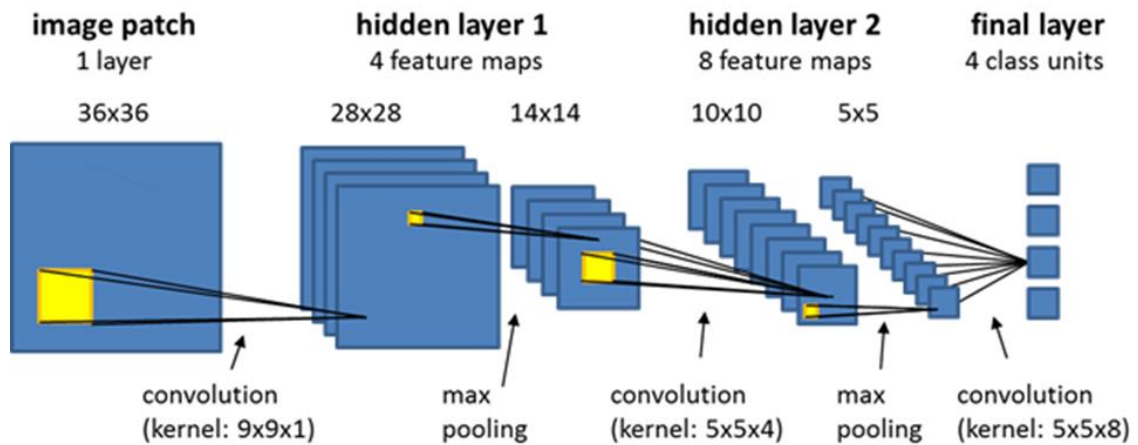
Figure 5.3 Architecture of Convolutional Neural Networks

5.4.1.1   Pretrained Models

A pretrained model refers to a machine learning model that has been trained on a large dataset for a specific task before being deployed for further fine-tuning or use in real-world applications. These models are trained using techniques like deep learning, where neural networks are trained on vast amounts of data to learn patterns and relationships within the data.

Examples of pretrained models include language models like OpenAI's GPT (Generative Pretrained Transformer) models, image classification models like ResNet or VGG, and other models used for tasks such as object detection, text generation,

language translation, and more.

Xception

Xception is a convolutional neural network that is 71 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database.

VGG16

VGG16 is a deep convolutional neural network model used for image classification tasks. The network is composed of 16 layers of artificial neurons, which each work to process image information incrementally and improve the accuracy of its predictions. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil and many animals.

Densenet

Densely Connected Convolutional Networks (DenseNet) is a feed-forward convolutional neural network (CNN) architecture that links each layer to every other layer. Each architecture consists of four Dense Blocks with varying number of layers. For example, the DenseNet-121 has [6,12,24,16] layers in the four dense blocks whereas DenseNet-169 has [6, 12, 32, 32] layers.

5.4.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. Remembering information for long periods is practically their default behavior, and this behavior is controlled with

the help of "gates". While RNNs process single data points, LSTMs can process entire sequences. Not only that, but they can also learn which point in the data holds importance, and which can be thrown away.

Hence, the only relevant information is passed on to the next layer. Their ability to remember information for extended periods of time which makes them widely used in various research areas.
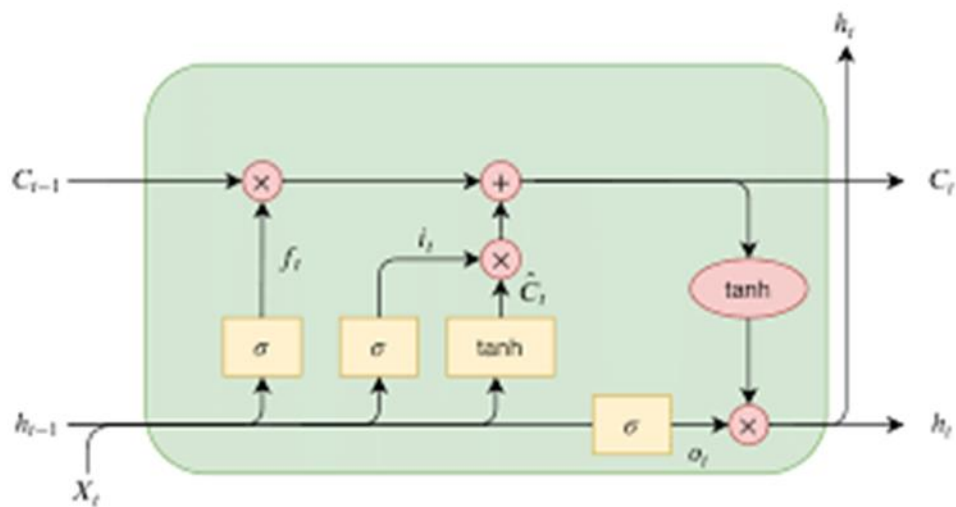
Figure 5.4 LSTM Structure

The Figure 5.4 shows the structure of LSTM. The 3 main gates involved are: input gate, output gate and forget gate. These gates decide whether to forget the currentcell value, read a value into the cell, or output the cell value. The hidden states play an important role since the previous hidden states are passed to the next step of the sequence.The hidden state acts as the neural network's memory, as it is storing the data that the neural network has seen before. Thus, it allows the neural network to function like a human braintrying to form sentences.

5.4.3 Gated Recurrent Unit (GRU)

Gated Recurrent Unit is a special kind of RNN designed for sequential data by allowing information to be selectively remembered or forgotten over time. However, GRU has a simpler architecture than LSTM, with fewer parameters, which can make it easier to train and more computationally efficient.

The main difference between GRU and LSTM is the way they handle the memory cell state. In LSTM, the memory cell state is maintained separately from the hidden state and is updated using three gates: the input gate, output gate, and forget gate. In GRU, the memory cell state is replaced with a "candidate activation vector," which is updated using two gates: the reset gate and update gate.
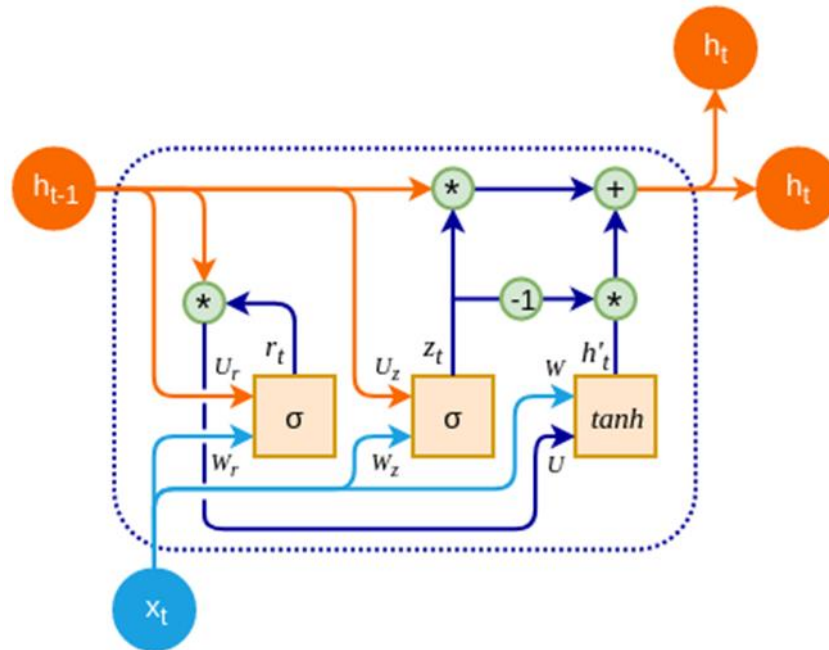
Figure 5.5 GRU Structure

The GRU architecture consists of the following components:

Input layer: The input layer takes in sequential data, such as a sequence of words or a

time series of values, and feeds it into the GRU.

Hidden layer: The hidden layer is where the recurrent computation occurs. At each time step, the hidden state is updated based on the current input and the previous hidden state. The hidden state is a vector of numbers that represents the network's "memory" of the previous inputs.

Reset gate: The reset gate determines how much of the previous hidden state to forget. It takes as input the previous hidden state and the current input, and produces a vector of numbers between 0 and 1 that controls the degree to which the previous hidden state is "reset" at the current time step.

Update gate: The update gate determines how much of the candidate activation vector to incorporate into the new hidden state. It takes as input the previous hidden state and the current input, and produces a vector of numbers between 0 and 1 that controls the degree to which the candidate activation vector is incorporated into the new hidden state.

Candidate activation vector: The candidate activation vector is a modified version of the previous hidden state that is "reset" by the reset gate and combined with the current input. It is computed using a tanh activation function that squashes its output between -1 and 1.

Output layer: The output layer takes the final hidden state as input and produces the network's output. This could be a single number, a sequence of numbers, or a probability distribution over classes, depending on the task at hand.

5.4.4 Attention

Attention mechanisms, inspired by human visual attention, have been widely used in neural network architectures to improve the performance of various tasks, including natural language processing and computer vision. The core idea behind attention mechanisms is to allow models to focus on relevant parts of the input data (e.g., words in a sentence, regions in an image) while performing a task.

In the context of sequence-to-sequence tasks such as machine translation or image captioning, where an input sequence (source) is mapped to an output sequence (target), attention mechanisms help the model to dynamically weigh the importance of different parts of the input sequence when generating each element of the output sequence. This allows the model to selectively attend to relevant information during the decoding process, rather than relying solely on fixed-length representations or context vectors.
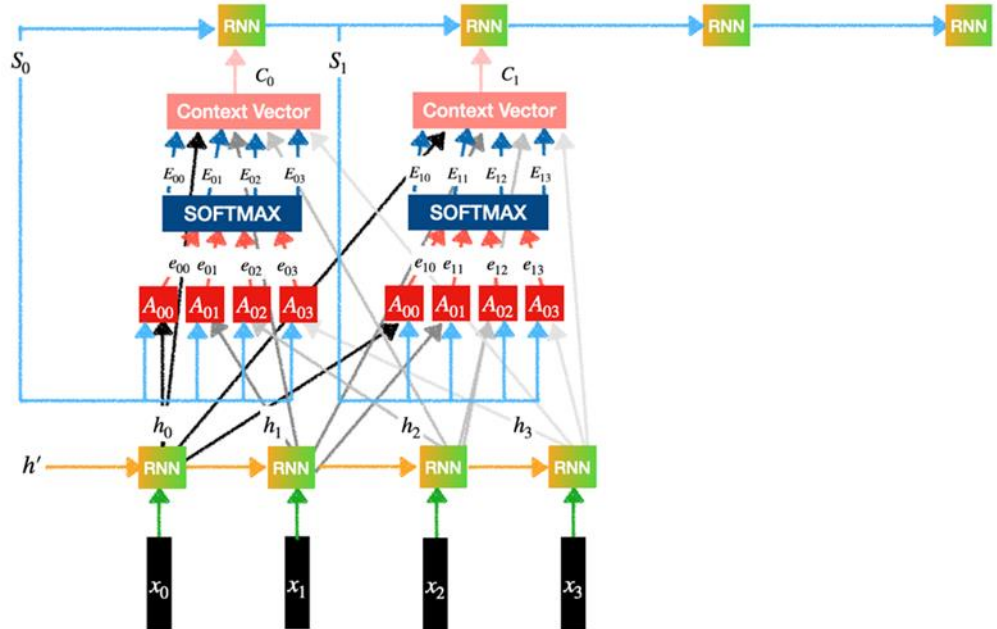
Figure 5.6 Attention Structure

Image Caption Model with Attention consists of four logical components:

1. **Encoder**: As the pre-trained CNN model encodes the image, the Encoder which consists of a Linear layer that takes the pre-encoded image features and passes them on to the Decoder.

2. **Sequence Decoder**: This is a recurrent network built with GRUs. The captions are passed in as the input after first going through an Embedding layer.

3. **Attention**: As the Decoder generates each word of the output sequence, the Attention module helps it to focus on the most relevant part of the image for generating that word.

4. **Sentence Generator**: this module consists of a couple of Linear layers. It takes the output from the Decoder and produces a probability for each word from the vocabulary, for each position in the predicted sequence.

The encoder creates a hidden state at each step. The attention component generates a context vector that captures the relevant information from the encoder's hidden states. The context vector is fed into the decoder along with the current hidden state, to predict the next token.

5.5 Architectures

We utilize a CNN + LSTM to take an image as input and output a caption. An "encoder" LSTM maps the source sentence (which is of variable length) and transforms it into a fixed-length vector representation, which in turn is used as the initial hidden state of a "decoder" LSTM which ultimately generates the final

meaningful sentence as a prediction. RNN's have become very powerful. Especially for sequential data modelling. It encode images toa high level representation then decodes this using a language generation model.
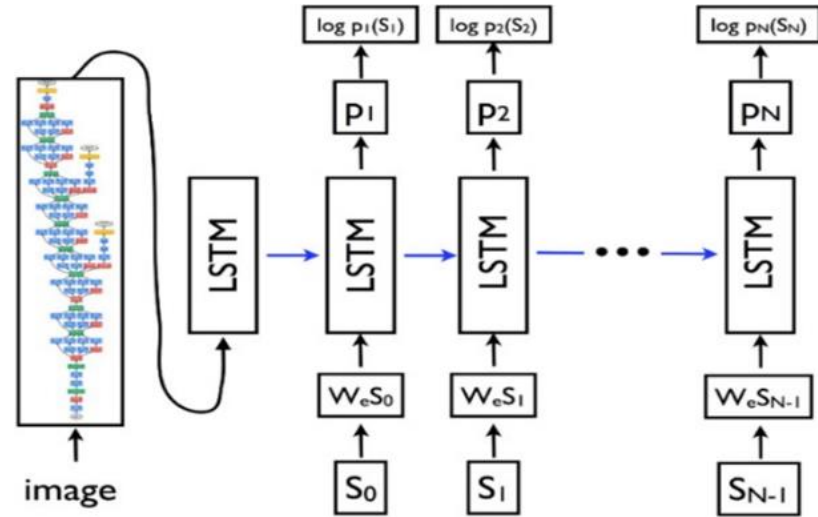
Figure 5.7 CNN-LSTM structure

However, replace this LSTM with a deep CNN - since it can produce a rich representationof the input image by embedding it to a fixed-length vector - by first pre-training it for an image classification task and using the last hidden layer as an input to the LSTM decoder that generates sentences. The above Figure 4.5 shows the working of CNN-LSTM.

We also utilized a CNN + GRU along with Attention mechanism. The GRU takes the responsibility of LSTM in the above architecture to generate the final meaningful sentence as a prediction, where addition attention layer is added to the decoder GRU. The architecture is as follows:
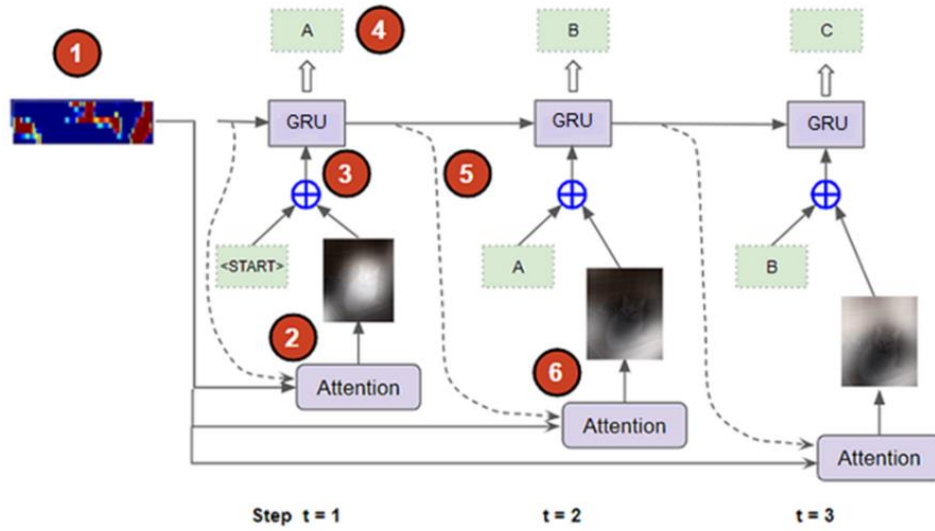
Figure 5.8 CNN-Attention-GRU structure

The Attention Module takes the encoded image from the Encoder, and the hidden state from the Sequence Decoder and computes the weighted Attention Score.

The input sequence is passed through the Embedding layer and then combined with the Attention Score. The combined input sequence is fed to the Sequence Decoder, which produces an output sequence along with a new hidden state.

The GRU processes the output sequence and generates its predicted word probabilities. The Decoder's new hidden state from this timestep is used for the next timestep. We continue doing this until an 'End' token is predicted or we reach the maximum length of the sequence.

6 System Design

The basic architecture of proposed model consists of CNN, LSTM. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data. LSTM is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies.

6.1 Use case Diagram

Use-case diagrams describe the high-level functions and scope of a system. These diagramsalso identify the interactions between the system and its actors. The Figure 6.1 shows the use case representation of the system.
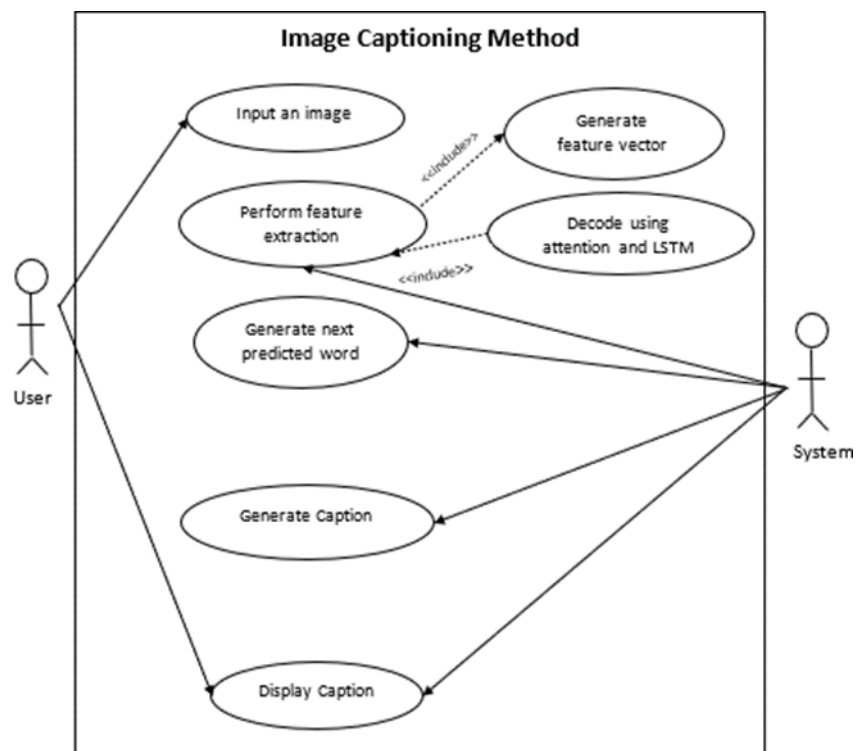
Figure 6.1 Use Case Diagram

6.2 Class Diagram

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. The Figure 6.2 shows the class diagram representation of the system.
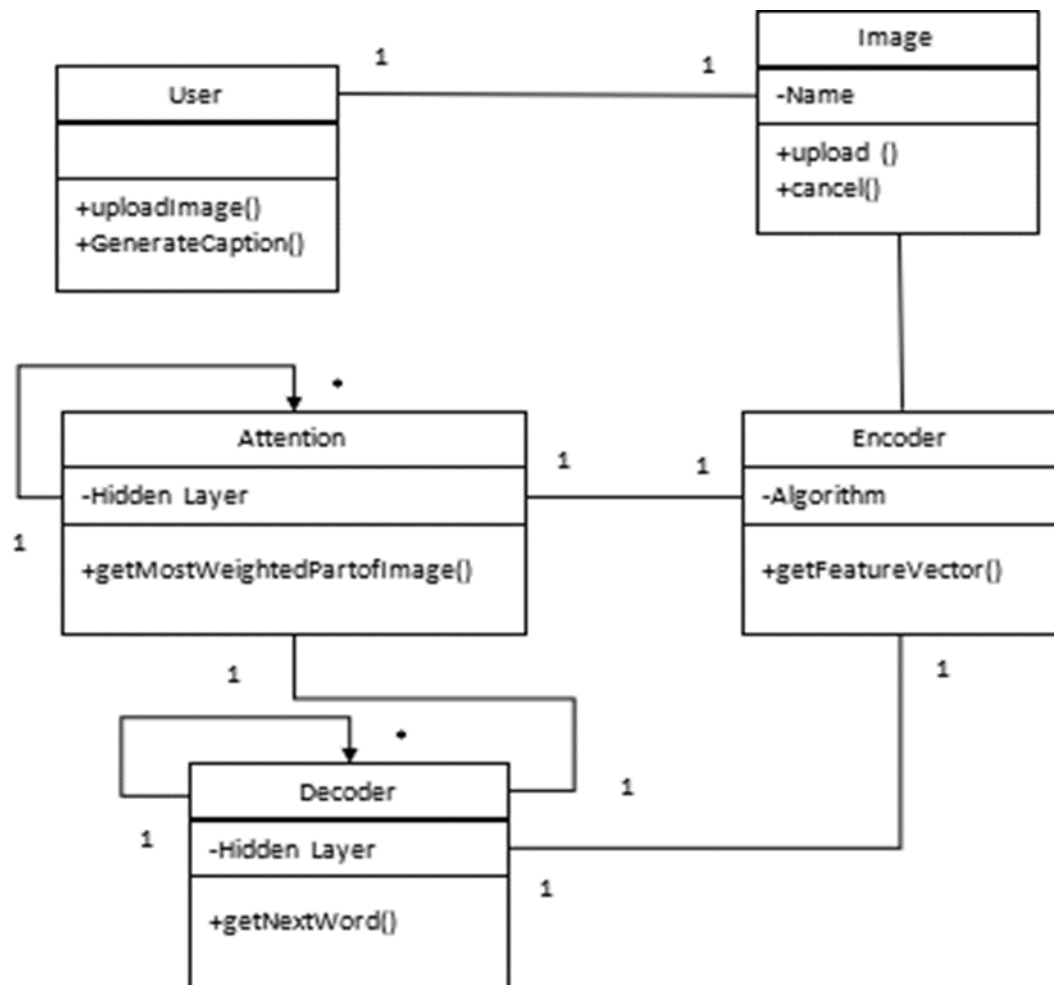
Figure 6.2 Class Diagram

6.3 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. In UML, an activity diagram provides a view of the behavior of a system by describing the sequence of actions in a process. The Figure 6.3 shows the activity diagram representation of the system.
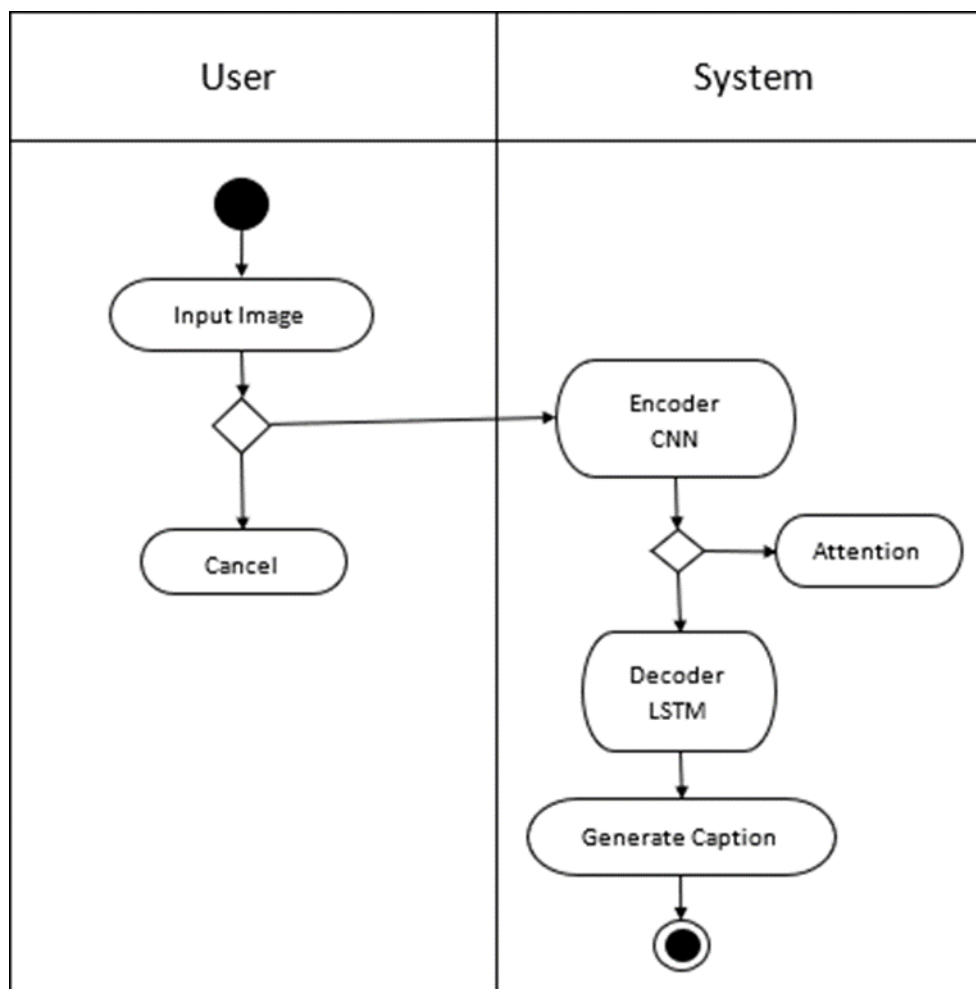
Figure 6.3 Activity Diagram

6.4 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. The Figure 6.4 shows the sequence diagram representation of the system. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them.
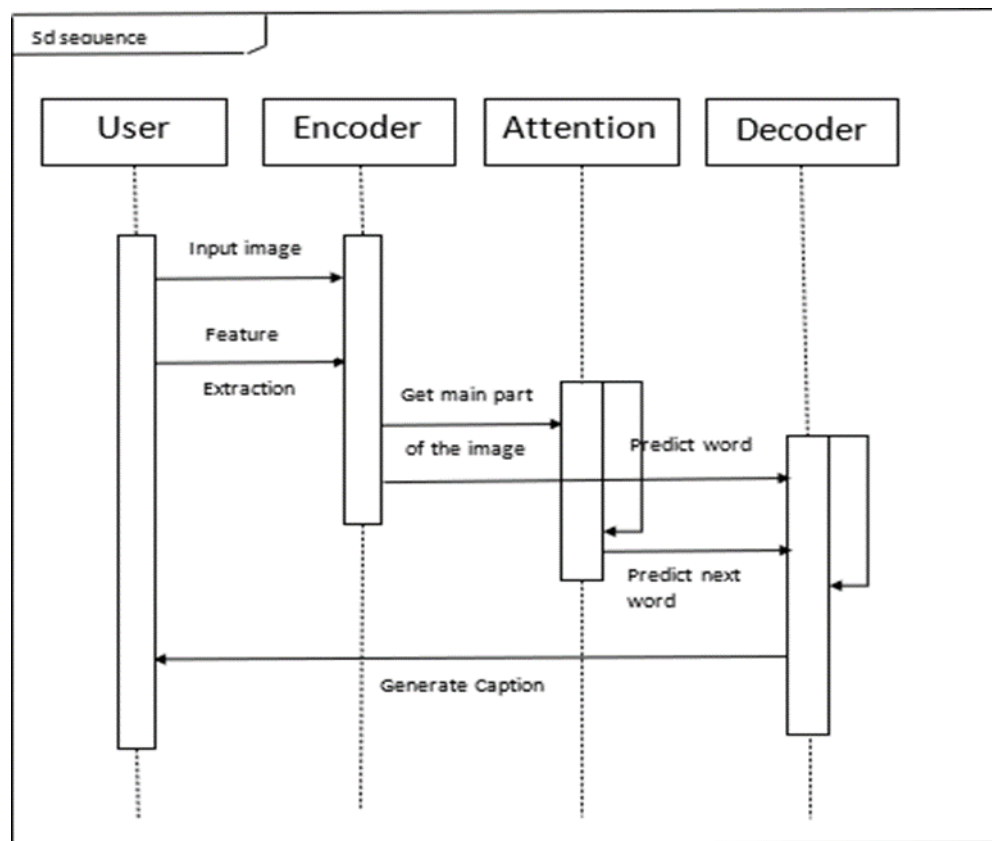
Figure 6.4 Sequence Diagram

6.5 State Chart Diagram

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). The Figure 6.5 shows the state chart diagram representation of the system.
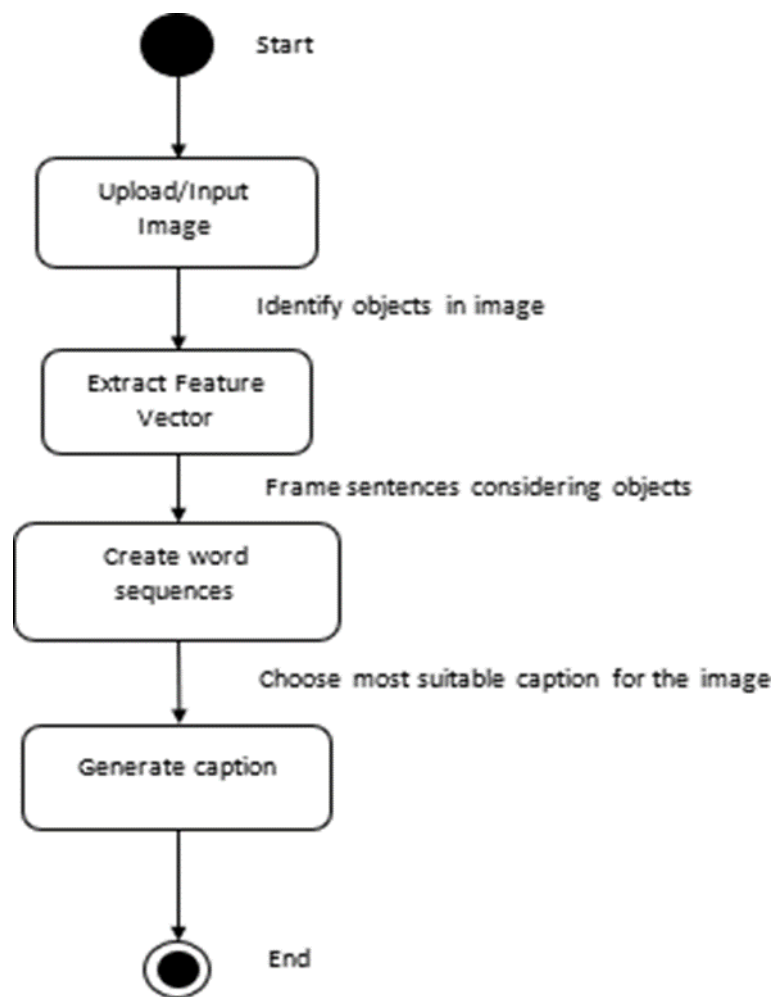
Figure 6.5 State Chart Diagram

7 Evaluation

Execution of the entire program takes place in 5 major steps. The implementation of the five major modules is as follows:

7.1 Data Cleaning and Preprocessing

This process involves filling of missing values, smoothing or removing noisy data and outliers along with removing inconsistencies.

1) For a comfortable and fast work experience, we use Google Colaboratory: a tool which provides free GPU/TPU processing power, over our local machines, which can take several hours to do the task that a GPU will take few minutes to do.

2) Our program starts with loading both, the text file, and the image file into separatevariables; the test file is stored in a string.

3) This string is used and manipulated in such a way so as to create a dictionary that maps each image with a list of 5 descriptions.

4) The main task of data cleaning involves removing punctuation marks, converting the whole text to lowercase, removing stop words, and removing words that containnumbers.

5) Further, a vocabulary of all unique words from all the descriptions is created, whichin the future will be used to generate captions to test images.

6) Another aspect of Preprocessing the data involves tokenizing our vocabulary with a unique index value. This is because a computer won't understand regular Englishwords, hence they need to be represented using numbers. The tokens are then storedin a pickle file. i.e. in the form of character stream, but with all the information necessary to reconstruct it into the original object type.

7) The above two preprocessing tasks can be achieved manually or by using the Keras.preprocessing.module for ease of writing the code.

8) We proceed to append the and identifier for each caption since these will act as indicators for our LSTM to understand where a caption is starting and where it's ending.

9) We will proceed with calculating the number of words in our vocabulary and finding the maximum length of the description, which will be used in later phases.

7.2 Extraction of feature vectors

A feature vector (or simply feature) is a numerical value in the matrix form, containing information about an object's important characteristics, e.g., intensity value of each pixel of the image in our case. These vectors, we'll ultimately store in a pickle file.

10) In our model we'll be using Transfer Learning, which simply means, using a pretrained model (in our case the Xception or VGG16 or DenseNet model) to extract features from it.

11) For example the Xception model is a Convolutional Neural Network that is 71 layers deep. It istrained on the famous ImageNet dataset which has millions of images and over 1000 different classes to classify from.

12) Python makes using this model in our code extremely easy with keras.applications.xceptionmodule. To use it in our code, we'll drop the classification layer from it, and hence obtain the 4096 feature vector.

13) Hence weights will be downloaded for each image, and then image names will be mapped with their respective feature array.

14) This process can take a few hours depending on your processor.

7.3 Layering the CNN-RNN model
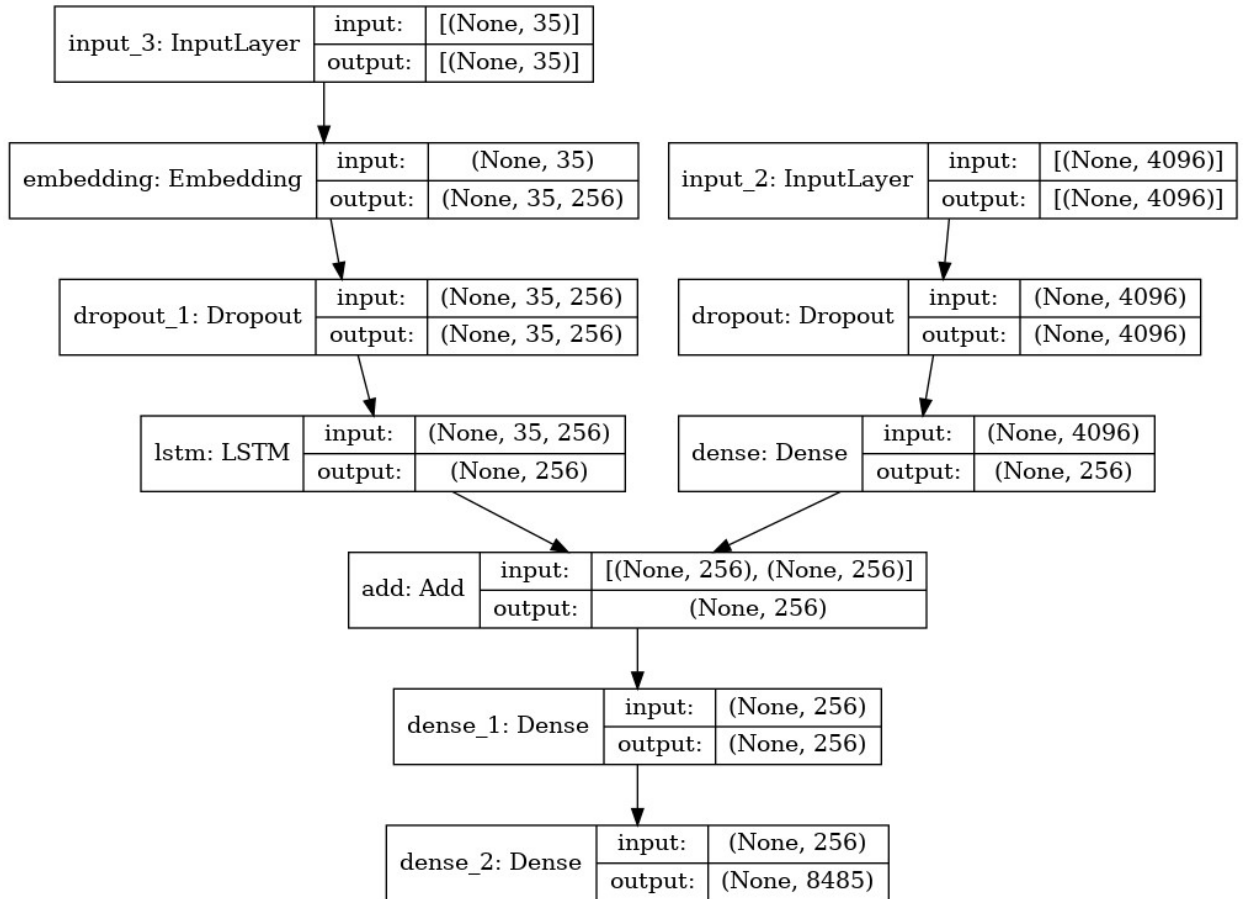
Figure 7.1 Structure of neural network

To stack the model, we'll use the Keras Model from Functional API. The structure will consist of 3 parts:

15) Feature Extractor: It will be used to reduce the dimensions from 4096 to 256. We'll make use of a Dropout Layer. One of these will be added in the CNN and the LSTMeach. We have pre-processed the photos with o n e o f the pretrained model (without the output layer) and will use

33

the extracted features predicted by this model as input. The Figure 7.1 demonstrates the model.

16) Sequence Processor: This Embedding layer will handle the textual input, followedby the LSTM layer.

17) Decoder: We will merge the output from above two layers and use a Dense layerto make the final predictions. Both the feature extractor and sequence processor output a fixed-length vector. These are merged and processed by a Dense layer. The number of nodes in the final layer will be the same as the size of our vocabulary.

7.4 Layering the CNN-Attention-RNN model

embedding: Embedding	input:	(None, 49, 512)
	output:	(None, 49, 256)

droupout_0: Droupout	input:	(None, 49, 256)
	output:	(None, 49, 256)

Features: Relu	input:	(None, 49, 256)
	output:	(None, 49, 256)
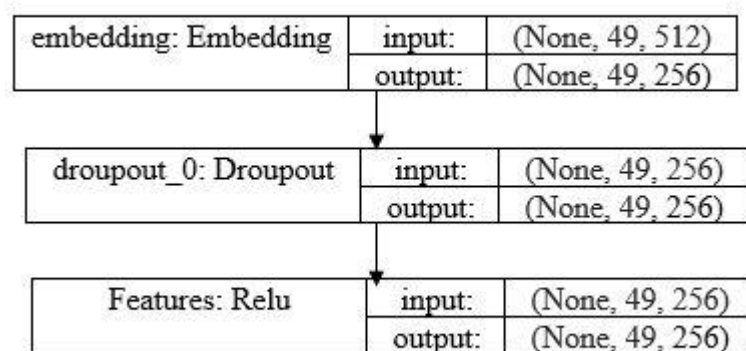
Figure 7.2 Encoder Layers

18) In addition attention layer is used in some models where the features are not flattened instead feature vectors are extracted with as many number

of locations identified. In our attention model 49 feature vectors are formed.

19) For the features the attention weights are multiplied to form context vector. Where the attention weights are calculated from the Attention layers (Uattn, Vattn,Wattn).

20) The context vector helps the decoder to identify more relevant words while framing the sentence as output.
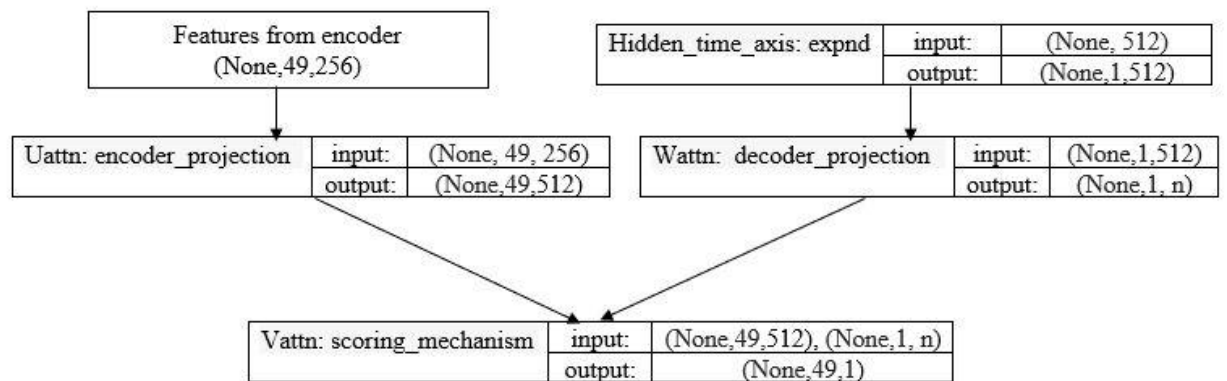
Figure 7.3 Attention Layers
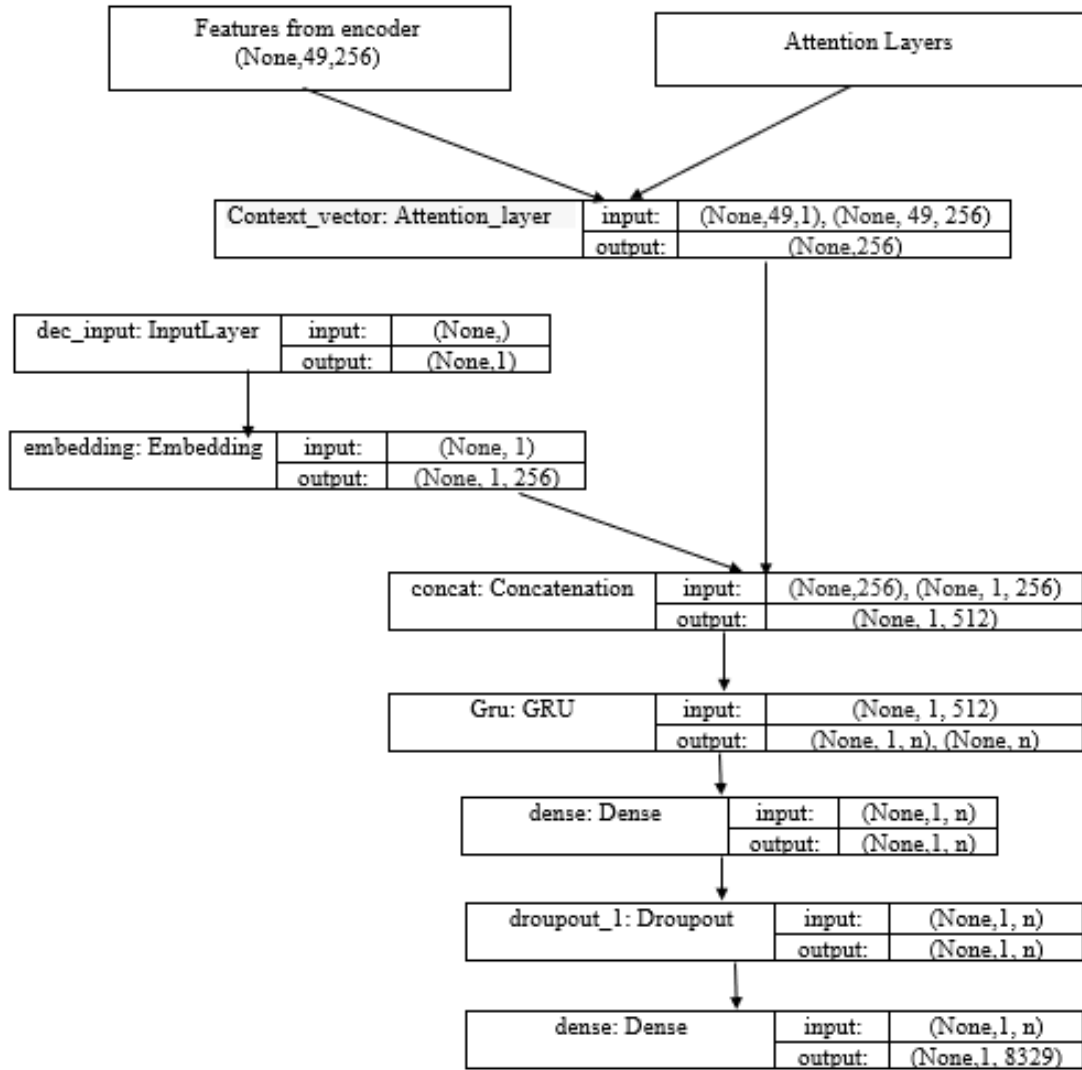
Figure 7.4 Neural Structure of Attention Mechanism

7.5 Training

Training a model means learning good values for all the weights and the bias from labelled examples.

```
dense_2 (Dense)              (None, 7577)        1947289      [ dense_1[0][0] ]

==================================================================================
Total params: 5,002,649
Trainable params: 5,002,649
Non-trainable params: 0
_____
None
<keras.engine.functional.Functional object at 0x7f345ee82130> model
<ipython-input-17-73e22a893c2c>:16: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future v
  model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
6000/6000 [==============================] - 573s 95ms/step - loss: 4.4967
6000/6000 [==============================] - 564s 94ms/step - loss: 3.6509
6000/6000 [==============================] - 570s 95ms/step - loss: 3.3635
6000/6000 [==============================] - 560s 93ms/step - loss: 3.1900
6000/6000 [==============================] - 561s 94ms/step - loss: 3.0724
6000/6000 [==============================] - 563s 94ms/step - loss: 2.9845
6000/6000 [==============================] - 571s 95ms/step - loss: 2.9165
6000/6000 [==============================] - 571s 95ms/step - loss: 2.8607
6000/6000 [==============================] - 557s 93ms/step - loss: 2.8176
6000/6000 [==============================] - 563s 94ms/step - loss: 2.7807
```

Figure 7.5 Model for Training

21) We'll be training our model on 6000 images each having 4096 long feature

vectors.

22) Since it is not possible to hold all this data in the memory at the same time,

we'll make use of a Data Generator. This will help us create batches of the

data, and willimprove the speed.

23) Along with this we'll be defining the number of epochs (i.e. iterations of the

trainingdataset) the model has to complete during its training as shown in

the Figure 6.2. This number must be selected in such a way that our model

is neither underfitted nor overfitted.

24) model.fit_generator() method will be used. And this whole process will

take sometime depending on the processor.

25) The maximum length of descriptions calculated earlier will be used as parameter value here. It will also take as input the clean and tokenized data.

26) Create a sequence creator, which will play the role of predicting the next word based on the previous work and feature vectors of the image.

27) While training our model, we can use the development dataset (It's provided with the rest of the files), to monitor the performance of the model, to decide when to save the model version to the file.

28) We will proceed to save several models, out of which the final one will be used fortesting in future. The Figure 7.6 represents the model.

```
Layer (type)                  Output Shape         Param #    Connected to
==================================================================================================
input_2 (InputLayer)          [(None, 32)]         0          []

input_1 (InputLayer)          [(None, 2048)]       0          []

embedding (Embedding)         (None, 32, 256)      1939712    ['input_2[0][0]']

dropout (Dropout)             (None, 2048)         0          ['input_1[0][0]']

dropout_1 (Dropout)           (None, 32, 256)      0          ['embedding[0][0]']

dense (Dense)                 (None, 256)          524544     ['dropout[0][0]']

lstm (LSTM)                   (None, 256)          525312     ['dropout_1[0][0]']

add (Add)                     (None, 256)          0          ['dense[0][0]',
                                                               'lstm[0][0]']

dense_1 (Dense)               (None, 256)          65792      ['add[0][0]']

dense_2 (Dense)               (None, 7577)         1947289    ['dense_1[0][0]']

==================================================================================================
Total params: 5,002,649
Trainable params: 5,002,649
Non-trainable params: 0
```

Figure 7.6   Model

7.6 Testing Phase

This phase helps spot problems in models that regular evaluation metrics might miss.

29)A separate Python notebook can be created, or the same can be used to perform testing. Either way, we'll load the trained model that we had saved in the previous step and generate predictions.

30)The sequence generator will come into play at this stage, besides the tokenizer file we created. The primary step of feature extraction for the particular image under observation will be performed. The path of one of the images from the remaining 2000 test images is passed to the function manually.

31)You can also iterate through the test data set and store the prediction for each image in a dictionary or a list. The actual functioning behind image generation involves using the start sequence and the end sequence, and to call the model recursively to generate meaningful sentences.

7.7 Code URL

https://github.com/Saisree5/A-neural-image-caption-generator.git

8 Results and Comparison

This section will show the results of the image caption generator with different methods and their comparison with the actual caption and with other method captions.

This comparison can be done in two ways. Either with human evaluation or with performance metrics.

8.1 Human Evaluation

For consideration, only two images have been subjected to testing, and the results can be seen in the following images:

1) Path of Figure 9.1.1:

/content/drive/MyDrive/ML/Flicker8k_Dataset/111537222_07e56d5a30.jpg

Output:


```
#Test the model using dumy following image and code
from PIL import Image
img = Image.open('/content/drive/MyDrive/ML/Flicker8k_Dataset/111537222_07e56d5a30.jpg')
# imagePath = '/content/drive/MyDrive/ML/Flicker8k_Dataset/3738685861_8dfff28760.jpg'
# img = Image.open(imagePath)  111537222_07e56d5a30.jpg

img
```

Figure 8.1 Image 1

```
!python3 '/content/drive/MyDrive/ML/testing_caption_generator.py' -i '/content/drive/MyDrive/ML/Flicker8k_Dataset/111537222_07e56d5a30.jpg'
#23445819_3a458716c1.jpg
# !python3 '/content/drive/MyDrive/ML/testing_caption_generator.py' -i '/content/drive/MyDrive/ML/Flicker8k_Dataset/3738685861_8dfff28760.jpg
```

```
2023-01-30 04:17:14.792175: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:42] Overriding orig_value setting because the TF_FORCE_
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop
83683744/83683744 [==============================] - 4s 0us/step
1/1 [==============================] - 7s 7s/step
```

start man is climbing up the side of cliff end

Figure 8.2 Output 1

2) Path of Figure 9.1.3:

/content/drive/MyDrive/ML/Flicker8k_Dataset/23445819_3a458716c1.jpg

Output:

Figure 8.3 Image 2

```
!python3 '/content/drive/MyDrive/ML/testing_caption_generator.py' -i '/content/drive/MyDrive/ML/Flicker8k_Dataset/23445819_3a458716c1.jpg'
#23445819_3a458716c1.jpg
# !python3 '/content/drive/MyDrive/ML/testing_caption_generator.py' -i '/content/drive/MyDrive/ML/Flicker8k_Dataset/3738685861_8dfff28760.
```

```
2023-02-10 15:03:06.351231: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:42] Overriding orig_value setting because the TF_FOF
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_no
83683744/83683744 [==============================] - 0s 0us/step
1/1 [==============================] - 8s 8s/step
```
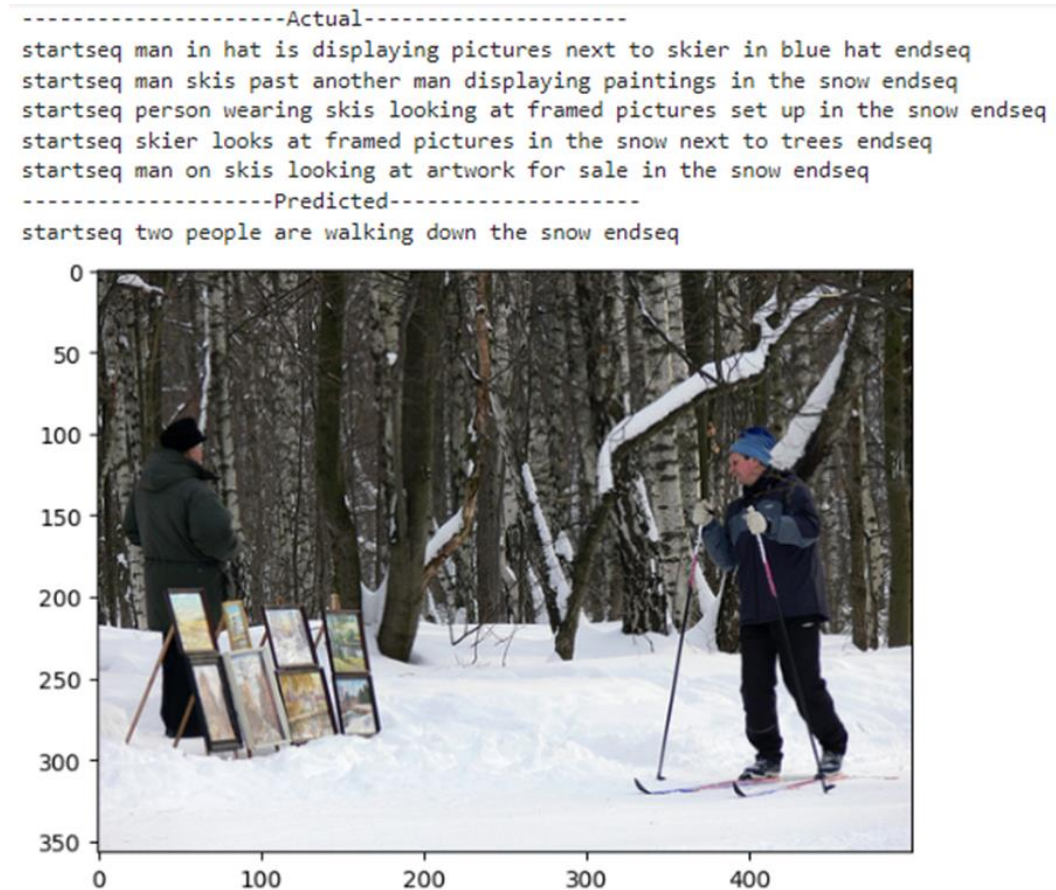
start two dogs are playing together on the grass end

Figure 8.4 Output 2

Table 8.1 Comparison with original captions

Image	The original description	Predicted description
111537222_07e56d5a30.jpg	A climber wearing a blue helmet and headlamp is attached to a rope on the rock face.	Man is climbing up theside of cliff
23445819_3a458716c1.jpg	a beagle and a goldenretriever wrestling in the grass	Two dogs are playing together on the grass

The above Table 8.1 differentiates between the original provided caption and the caption that is predicted by the image captioning method. The generated is far better and meaningful when compared to the original description. Hence image captioning method helped to create a caption of a given input image which is syntactically and semantically correct.

```
--------------------Actual--------------------
startseq man in hat is displaying pictures next to skier in blue hat endseq
startseq man skis past another man displaying paintings in the snow endseq
startseq person wearing skis looking at framed pictures set up in the snow endseq
startseq skier looks at framed pictures in the snow next to trees endseq
startseq man on skis looking at artwork for sale in the snow endseq
--------------------Predicted--------------------
startseq two people are walking down the snow endseq
```


Figure 8.5 Image 3

The above Figure 8.5 represent the difference between the original 5 captions present

in the dataset and the predicted caption by the image captioning method with VGG16

a white dog is running on grass

Figure 8.6 Image 4

Table 8.2 Comparison with results of different methods

Method	Generated Caption
Xception_LSTM	start two dogs are playing together in the grass end
Xception_GRU	start dog is running through the grass end
VGG16_LSTM	startseq white dog is flying through the air endseq
VGG16_GRU	startseq white dog is running through the air endseq
Attention_GRU_Xcep	two white dogs run across the woods <end>
Attention_GRU_VGG16	dark green grass <end>
Attention_GRU_DenseNet	white dog is running on grass <end>
DenseNet_GRU	startseq dog is running in the grass endseq
DenseNet_LSTM	startseq brown dog is running through the grass endseq

The above Table 8.2 represents the captions generated by different methods for the Figure 8.6. We can understand by human evaluation that the methods Attention_GRU_DenseNet, DenseNet_GRU, DenseNet_LSTM gives more relevant images when compared to other methods.

8.2 Performance Metrics

To assess the performance of model prediction, we can use various metrics.

The performance metrics gives score for each of the generated caption by comparing with the caption given by the user. It helps us to identify which model predicted the caption more relevant to user given caption. It only helps to measure the performance of the model.

1. BLEU (Bilingual evaluation understudy): a well-known machine translation statistic is used to measure the similarity of one sentence with reference to multiple sentences. It was proposed by Papineni et al. It returns a value where a higher value represents more similarity.

This method works by counting the number of n-grams in one sentence with the n-grams in the reference sentence. A unigram or 1-gram represents a token, whereas a bi-gram indicates each pair of a word. For calculating the BLEU score of multiple sentences, Corpus BLEU is employed, in which a reference list is indicated using a list of documents, and a candidate document is a list where the document is a list of tokens.

$$Unigram\ precision\ P = \frac{m}{w_t} \qquad \text{8-1}$$

$$Brevity\ penalty\ p = \begin{cases} 1\ if\ c > r \\ e^{\left(1-\frac{r}{c}\right)}\ if\ c \le r \end{cases} \qquad \text{8-2}$$

$$BLEU = p.e^{\sum_{n=1}^{N}\left(\frac{1}{N}*\log P_n\right)} \qquad \text{8-3}$$

Here in 8-1 m is number of words from the candidate that are found in the reference. w_t is the total number of words in the candidate. In 8-2 r is the effective length of the reference corpus, and c the total length of the translation corpus. P_n is geometric average of the modified n-gram precision. N is length of n-grams used to compute P_n.

2. METEOR (Metric for Evaluation for Translation with Explicit Ordering): The METEOR metric calculates F-score based on mapping unigrams. Meteor score is a metric used to evaluate machine translation by comparing it to human translations. It takes both accuracy and fluency of the translation, as well as the order in which words appear. It ranges from 0 to 1.

$$Unigram\ precision\ P = \frac{m}{w_t} \qquad \text{8-4}$$

$$Unigram\ recall\ R = \frac{m}{w_r} \qquad \text{8-5}$$

$$Harmonic\ mean\ F_{mean} = \frac{10PR}{R + 9P} \qquad \text{8-6}$$

45

$$penalty \; p = 0.5 * \left(\frac{c}{U_m}\right)^3 \qquad \textbf{8-7}$$

$$Final \; score \; for \; a \; segment \; M = F_{mean}(1 - p) \qquad \textbf{8-8}$$

Here in 8-4, 8-5 m is the number of unigrams in the candidate translation that are also found in the reference translation. w_t is the total number of unigrams in the candidate translation. w_r is the total number of unigrams in the reference translation. In 8-7 c is the number of chunks, where a chunk is defined as a set of unigrams that are adjacent in the candidate and in the reference. In 8-8 U_m is the number of unigrams that has been mapped.

**Table 8.3 Comparison with performance metrics of different methods**

Method	BLEU SCORE	METEOR SCORE
Xception_LSTM	0.09306272250443651	0.15463917525773196
Xception_GRU	0.4953587998572467	0.4746835443037974
VGG16_LSTM	0.2791881675133095	0.30355097365406647
VGG16_GRU	0.37225089001774603	0.40914948453608246
Attention_GRU_Xcep	0.08067401742967989	0.2410901467505241
Attention_GRU_VGG16	0.14285714285714285	0.07142857142857142
Attention_GRU_DenseNet	0.558376335026619	0.6171248568155785
DenseNet_GRU	0.4294155960430205	0.42613636363636365
DenseNet_LSTM	0.37225089001774603	0.38659793814432986

From the Table 8.3 we can clearly observe that, compared to other models Attention_GRU_DenseNet gives more relevant captions for the Figure 8.6.

8.3 Screens

Our application main aim is to make the users able to generate caption for any given image in 9 different methods so that user can understand which is better caption for the given image with human evaluation and also provides performance metrics if the user enters their own caption.

The developed system integrates image caption generator and a database that stores the user images in the gallery along with a user-friendly interface to analyze different caption generators. The complete application is built using Django.
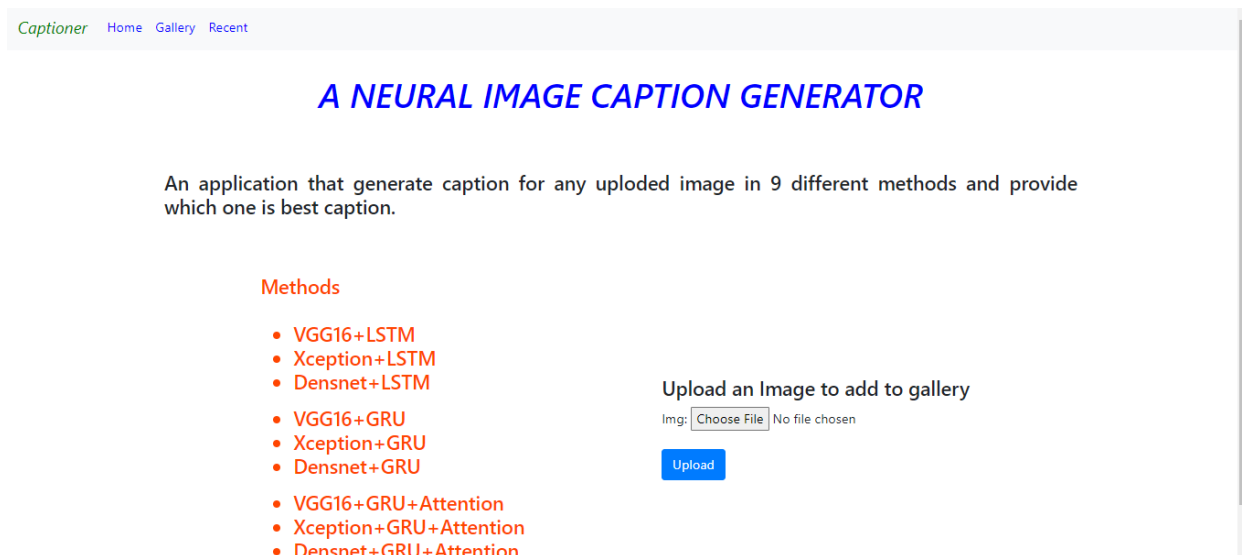
Figure 8.7 Home Page

The Figure 8.7 home page displays all the methods performed the image caption generator and also gives option to upload the image.
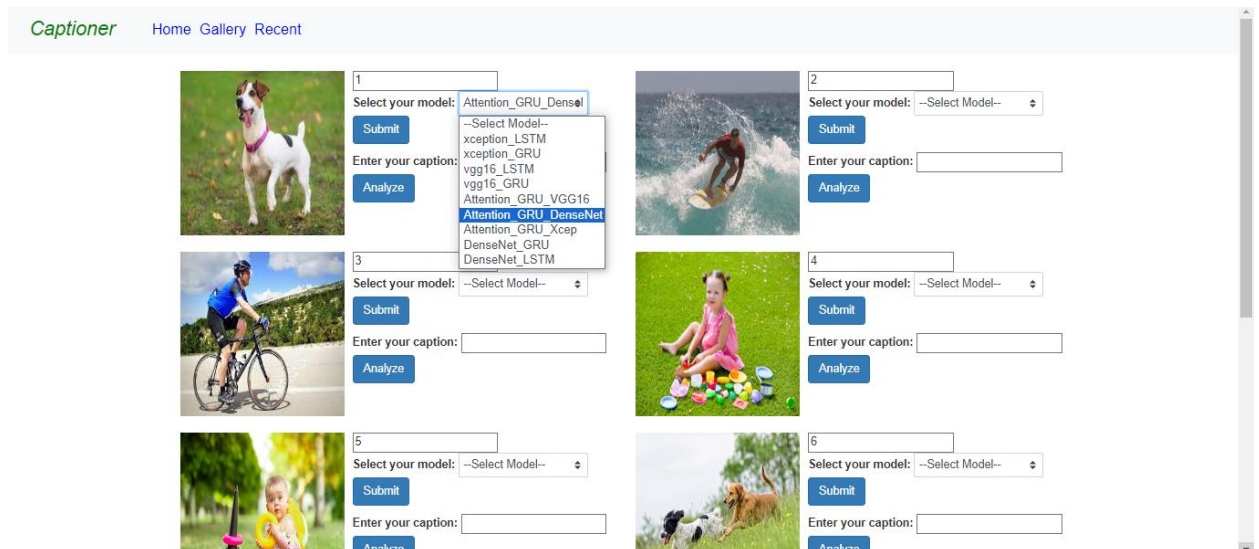
Figure 8.8 Gallery Page

The Figure 8.8 Gallery page displays all the images uploaded by the user and provides option to select the method for generate caption and also provides option to analyze.

Figure 8.9 Result Page

The Figure 8.9 shows the result page in which the caption will be displayed by the selected method for the given image.

Figure 8.10 Analysis Page

Figure 8.11 Analysis Page

Figure 8.12 Analysis Page

The Figure 8.10, Figure 8.11, Figure 8.12 Analysis page displays the caption generated

with 9 different methods and also provides the performance metrics if the user provides

the caption to know the performance of the methods
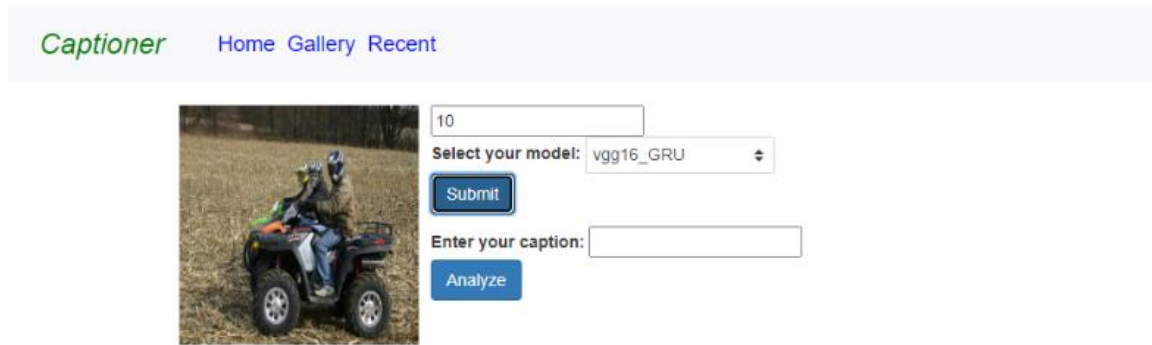
Figure 8.13 Recent Image Page

The Figure 8.13 shows the recent uploaded image and similar to the Figure 8.8 this page

also shows all options to perform caption generation by different methods.

9 Conclusion

Based on the results obtained we can see that the deep learning methodology used here bore successful results. The pretrained CNN model and the NLP model worked together in proper synchronization; they were able to find the relation between objects in images. To compare the accuracy of the predicted caption, we compared them with target captions in our Flickr8k test dataset and with the caption entered by the user, using BLEU (Bilingual Evaluation Understudy) score and METEOR (Metric for Evaluation for Translation with Explicit Ordering) score.

From our experiments and their evaluation with the help of performance metrics we conclude that, in NLP Models **Attention** + **GRU** gives best caption, and in pretrained models **DenseNet** gives better performance.

So we conclude that the method with **Attention+GRU+DenseNet** gives more relevant caption compared to other methods in most of the cases. This introduced us to various new developments in the field of Deep Learning and Artificial Intelligence and how vast this field is.

10 Future Scope

We can use image captioning in image indexing, for visually impaired persons, for social media, and several other natural language processing applications. Deep learning methods have demonstrated state-of-the-art results on caption generation problems. These social images are usually associated with a set of user-contributed tags, which can express users' attentiveness. The task of social image captioning is to generate descriptions of images with the help of available user tags.

When the speech in the video is not clear, captions help to understand what has been said. But even when the audio is clear, captions are still important for non-native speakers of the language. They help the watcher to better understand and follow conversations and events. Probably, it will be useful in cases/fields where text is most used and with the use of this, you can infer/generate text from images. As in our project, we can use the information directly from any particular image in a textual format automatically.

11 References

[1] A. I. S. a. G. E. H. Krizhevsky, "ImageNet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems Curran Associates Inc*, 2012.

[2] R. e. a. Girshick, "Region-based Convolutional Networks for Accurate Object Detection and Segmentation," in *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2015.

[3] O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and Tell: A Neural Image Caption Generator," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] A. G. A. P. A. J. N. S. N. D. Preeti Voditel, "Image Captioning-A Deep Learning Approach Using CNN and LSTM Network," in *3rd International Conference on Pervasive Computing and Social Networking (ICPCSN)*, 2023.

[5] R. G. Ahmet Aker, "Generating Image Descriptions Using Dependency Relational Patterns," in *48th Annual Meeting of the Association for Computational Linguistics*, 2010.

[6] Y. Z. X. Y. Haoran Wang, "An Overview of Image Caption Generation Methods," *Computational Intelligence and Neuroscience,* vol. 2020, p. 13, 2020.

[7] P. Y. Z. S. L. L. Y. Y. M. Q. &. G. R. Cao, "Image captioning with bidirectional semantic attention-based guiding of long short-term memory," *Neural Processing Letters,* vol. 50(1), pp. 103-119, 2019.

[8] G. &. H. H. Huang, "c-RNN: a fine-grained language model for image captioning," *Neural Processing Letters,* vol. 48(2), pp. 683-691, 2019.

[9] L. &. H. H. Yang, "Adaptive syncretic attention for constrained image captioning," *Neural Processing Letters,* vol. 50(1), pp. 549-564, 2019.

[10] X. W. N. Z. X. L. a. L.-J. L. Z. Ren, "Deep reinforcement learning- based image

captioning with embedding reward," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] E. M. Y. M. J. R. a. V. G. S. J. Rennie, "Self- critical sequence training for image," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[12] A. T. S. B. a. D. E. O. Vinyals, "Show and tell: A neural image caption generator," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.