Generate detailed caption of an image using deep learning

Submitted in partial fulfillment of the requirements of the degree

Bachelor of Engineering

By

Khan Shayaan Shakeel
Roll No - 3118030

Masalawala Murtaza Shabbir
Roll no - 3118033

Qazi Faizan Ahmed
Roll no - 3118039

Supervisor:

Er. Nafisa Mapari

University of Mumbai

Department of Computer Engineering

M. H. Saboo Siddik College of Engineering

Mumbai University

(2021 - 2022)

M.H. SABOO SIDDIK COLLEGE OF ENGINEERING
8, Saboo Siddik Road, Byculla, Mumbai – 400 008

This is to certify that,

Roll No.	Name
3118030	Khan Shayaan Shakeel
3118033	Masalawala Murtaza Shabbir
3118039	Qazi Faizan Ahmed

Of Final Year (B.E. Semester VIII) degree course in Computer Engineering, have completed the specified project synopsis on,

"Generate detailed caption of an image using deep learning"

As a partial fulfillment of the project work in a satisfactory manner as per the rules of the curriculum laid by the University of Mumbai, during the Academic Year August 2021 – June 2022.

Internal Guide
(Er.Nafisa Mapari)

Internal Examiner

External Examiner

Head of Department
(Dr. Zainab Pirani)

I/ C Principal
(Dr. Ganesh Kame)

Generate detailed caption of an image using deep learning

Submitted in partial fulfillment of the requirements of the degree of

Bachelor of Engineering

By

Khan Shayaan Shakeel - 3118030

Masalwala Murtaza Shabbir - 3118033

Qazi Faizan Ahmed - 3118039

Supervisor :

Prof. Nafisa Mapari

Department of Computer Engineering

M. H. Saboo Siddik College of Engineering

Mumbai University

(2021 - 2022)

Project Report Approval for B. E.

This project report entitled ***Image Caption Generation using Deep Learning*** by ***Shayaan Khan, Murtaza Masalawala, Faizan Qazi*** is approved for the degree of BE in Computer Engineering.

Examiners

1. ---

2. ---

Supervisors

1. ---

2. ---

Chairman

Date:

Place:

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Khan Shayaan Shakeel	**3118030**
Masalwala Murtaza Shabbir	**3118033**
Qazi Faizan Ahmed	**3118039**

Date:

Abstract

Computer vision has been an area of interest for engineers and scientists who have been spearheading in the field of artificial intelligence from the late 1960s as it was very essential to give machines or robots the power of visualizing objects and activities around them like the human visual system. The ability to visualize 2-Dimensional images and extracting features from them can be utilised for developing various applications. The involvement of deep learning has been successful in bolstering the field of computer vision even further. The abundance of images in today's digital world and the amount of information contained in them have made them a very valuable and research worthy data item.

A deep learning-based image caption generator model can incorporate the areas of natural language processing and computer vision with deep learning to give a solution in which the machine can extract features from an image and then describe those features in a natural language. Thus, explaining the contents of the image in a human readable format. This model has various applications ranging from social causes like being an aid to visually impaired to enhancing search experience of users over the web. This paper analyses the various state-of-the-art work in the field of image processing, computer vision and deep learning and presents a deep learning model that generates captions describing the images given as input to the system and further converts the text output into an audio format.

Acknowledgement

No project can be completed without the support of a lot of people. Today when we are concluding our project synopsis work by submitting this report, we reflect upon all the times when we needed support in various forms and were lucky enough to receive it.

We wish to express our sincere gratitude to our I / C principal Dr. Ganesh Kame, M. H. Saboo Siddik College of Engineering, Mumbai, for providing the facilities to carry out the project work.

A special mention here to Dr. Zainab Pirani (H.O.D., Computer Engineering Department, MHSSCOE) for his valuable support. We are also thankful to all staff members of Computer Department, without whom the completion of this report would have been impossible.

This entire journey would not have been possible without the efforts put in by our guides, Er. Nafisa Mapari. They have been a constant source of encouragement and guidance through the entire semester.

This acknowledgment would indeed be incomplete without rendering our sincere gratitude to our family. They have always been a pillar of strength and support in our past and current endeavors.

Khan Shayaan Shakeel
Murtaza Masalawala Shabbir
Qazi Faizan Ahmed

List of Figures

List of Tables

List of Abbreviations

1. **UI:** User Interface
2. **CNN:** Convolutional Neural Network
3. **RNN:** recurrent Neural Network
4. **LSTM:** Long Short Term Memory
5. **SDLC:** Software Development Life Cycle
6. **DFD:** Data Flow Diagram
7. **UML:** Unified Modified Language

Index

Chapter 1

Project Overview

Project Overview

Computer vision has been an area of interest for engineers and scientists who have been spearheading in the field of artificial intelligence from the late 1960s as it was very essential to give machines or robots the power of visualizing objects and activities around them like the human visual system. The ability to visualize 2-Dimensional images and extracting features from them can be utilised for developing various applications. The involvement of deep learning has been successful in bolstering the field of computer vision even further. The abundance of images in today's digital world and the amount of information contained in them have made them a very valuable and research worthy data item.

A deep learning-based image caption generator model can incorporate the areas of natural language processing and computer vision with deep learning to give a solution in which the machine can extract features from an image and then describe those features in a natural language. Thus, explaining the contents of the image in a human readable format. This model has various applications ranging from social causes like being an aid to visually impaired to enhancing search experience of users over the web.

Automatic image captioning is an important aspect of Computer Vision and Natural language processing. Image caption 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. It is widely used by search engines to retrieve and show relevant search results to the user over the annotation keywords, to categorize personal multimedia collections, for automatic product tagging in online catalogs, in computer vision development, robotic vision and other areas of business and research.

This project analyses the various state-of-the-art work in the field of image processing, computer vision and deep learning and presents a deep learning model that generates captions describing the images given as input to the system and further converts the text output into an audio format.

Chapter 2

Introduction and Motivation

2.1 Introduction

Computer vision, Image processing and machine learning have become very crucial needs and also an economical alternative in various fields and applications. These include applications ranging from signature recognition for authorization to iris and face recognition in forensics. Also, their combination is being widely used in military applications across the world. Each of these applications has its special basic requirements, which may be unique from the others. Any stakeholder of such systems or models is concerned and wants their system to be faster, more accurate than other counterparts as well as cheaper and equipped with more extensive computational powers.

All such traits from the systems are desirable as most of these systems are being used for mission-critical purposes and scope of any mistake should be very less. Such systems are required to handle the complexity of problems of the modern world like intelligent crimes, a smart city needs like smart traffic control systems, disaster control and management systems etc., thus a computer vision based model that is unbiased and free of any prejudice towards anything or anyone is required to generate a caption describing the images given to it as input. So that such description can be used to automate existing systems like traffic control systems, flood control systems or surveillance systems, this will reduce chances of errors in such critical works and also the surveillance can be conducted 24X7 without human interaction.

This project puts forward a task of extracting features from a digital image and then describing those extracted features in a natural language. Through the localization and description of salient regions of images using LSTM a meaningful sentence in natural language will be formed that will describe images i.e. given a set of images and prior knowledge about the content find the correct semantic label for the entire image.

The project was undertaken with a vision to use the deep learning algorithms and incorporate it with web technologies in order to create a user friendly system which generates correct caption of the image uploaded efficiently.

2.2 Motivation

Generating image captions is an important aspect of Computer Vision and Natural language processing. Image caption 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.

Automatic image captioning is widely used by search engines to retrieve and show relevant search results to the user over the annotation keywords, to categorize personal multimedia collections, for automatic product tagging in online catalogs, in computer vision development, robotic vision and other areas of business and research.

There are numerous other applications for which the project can be used or extended to but our primary motivation for the project is to create a product that will aid visually imparied people to navigate through everyday situations without depending on other people.

2.3 Aim and Objectives

Aim

To develop a system that generates detailed caption of an image efficiently using deep learning algorithms.

Objectives

- To implement a system that generates image captions on image upload efficiently.
- To incorporate it with web technologies to develop a mobile-friendly website
- To convert the generated captions into audio.
- Website should have simple and attractive UI for naive users and to improve ease of use.

2.4 Scope

Finding the best and most efficient way to generate detailed caption of an image is very crucial using which we will obtain the audio of the caption generated. This entire system will be incorporated into a website where a user can submit the image to obtain its caption and audio of the generated caption.

The product will be able handle form validation i.e. the form will only accept files of image format not any other file format. After successfully accepting the file when when user will click submit button the image will be sent to the server and then feed to the model to get the caption which will the sent to the modules used for text to speech conversion then the result i.e. caption and audio will be dispalyed to the user.

Assumptions and their management are important in software development. Not well managed assumptions may lead to undesirable problems. The only assumption we made for our model to generated the correct caption is that the user will upload the kind of image on which the model was trained. Because the vocabulary of the model is limited to the dataset on which it was trained on. So, if the user uploads the type of image on which the model was not trained then it can generate an irrelevant response.

Chapter 3

Problem Statement

Problem Statement

The problem statement is to generate detailed caption an image using deep learning algorithms and further convert the generated caption into voice form for the end user to listen. The goal is to incorporate the system into a user-friendly and mobile-friendly website.

A deep learning-based image caption generator model can incorporate the areas of natural language processing and computer vision with deep learning to give a solution in which the machine can extract features from an image and then describe those features in a natural language. Thus, explaining the contents of the image in a human readable format. This model has various applications ranging from social causes like being an aid to visually impaired to enhancing search experience of users over the web.

Chapter 4

Requirement Analysis

4.1 Requirement Analysis

4.1.1. Literature survey

Author(s)	Research Paper	Publication Year	Dataset and methodology	Conclusions
Simao Herdade, Armin Kappeler, Kofi Boakye, Joao Soares	Automatic Image Captioning using CNN and LSTM	2019	Dataset: MSCOCO Method: architecture model using CNN as well as NLP techniques	Using CNN and LSTM models the image's caption is generated.
B.Krishnakumar,K. Kousalya, S.Gokul,R.Karthik eyan, D.Kaviyarasu	Image Caption Generator using Deep Learning	2020	Method: Deep learningbased model using CNN to identify featured objects with the help of OpenCv	Proposed model could generate captions successfully in Jupyter Notebook using keras as well as tenserflow
Pranay Mathur, Aman Gill, Nand Kumar Bansode, Anurag Mishra	Camera2 Caption: A real time caption generator	2017	Dataset: MSCOCO Method: Advanced deep reinforcement learning based on NLP and Computer Vision	The model proposed generates the real time environment high quality captions with the help of tensorflow
R. Subash	Automatic Image Captioning using CNN and LSTM	2019	Dataset: MS COCO Method: NLP and CNNLSTM based mode	Using CNNLSTM and NLP techniques the model for image captioning is generated

4.2. Existing system

Image Caption Generator models is based on encoder-decoder architecture which use input vectors for generating valid and appropriate captions. This model bridges gap input vectors for generating valid and appropriate captions. This model bridges gap between natural language processing as well as computer vision. It"s a task of recognizing and interpreting the context described in the image and then describing everything in natural language such as English. Our model is developed using the two main models i.e., CNN (Convolutional Neural Network) and RNN-LSTM (Recurrent Neural Networks- Long Short-Term Memory). The encoder in the derived application is CNN which is used to extract the features from the photograph or image and RNN-LSTM works as a decoder that is used in organizing the words and generating captions. Some of the major applications of the application are self-driving cars wherein it could describe the scene around the car, secondly could be an aid to the people who are blind as it could guide them in every way by converting scene to caption and then to audio, CCTV cameras where the alarms could be raised if any malicious activity is observed while describing the scene, recommendations in editing, social media posts, and many more. The model can predict accurate captions for only those type of images for which it was trained. For filling the gap of inaccuracy the model needs to be trained on a larger dataset with a variety of captions.

Chapter 5

Project Design

5.1 Methodology

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.

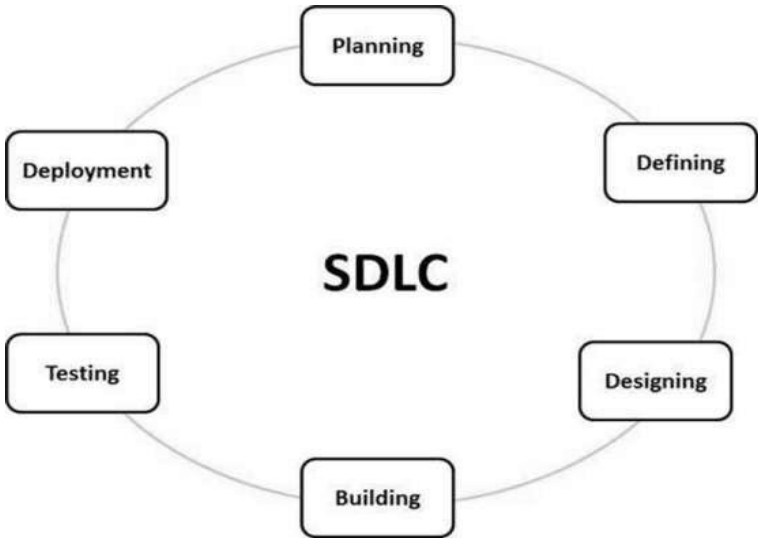


Fig 5.1 SDLC Diagram

A typical Software Development Life Cycle consists of the following stages –

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

Waterfall Model

In our proposed system, we are using Waterfall model as we need to complete our task. The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

When Should You Use It?

- Requirements are clear and fixed that may not change.
- There are no ambiguous requirements (no confusion).
- It is good to use this model when the technology is well understood.
- The project is short and cast is low.
- Risk is zero or minimum.

5.2 Design Details

5.2.1 Data Flow Diagram:

- **DFD Level-0**

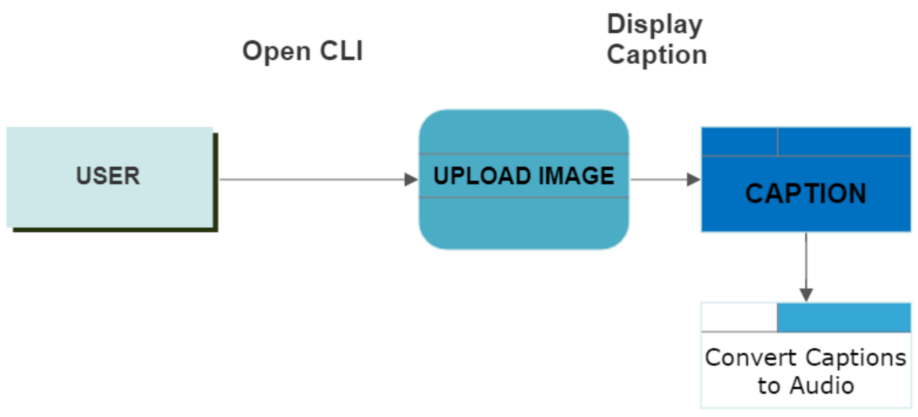


Figure 5.2 DFD Level-0

- **DFDLevel-1**

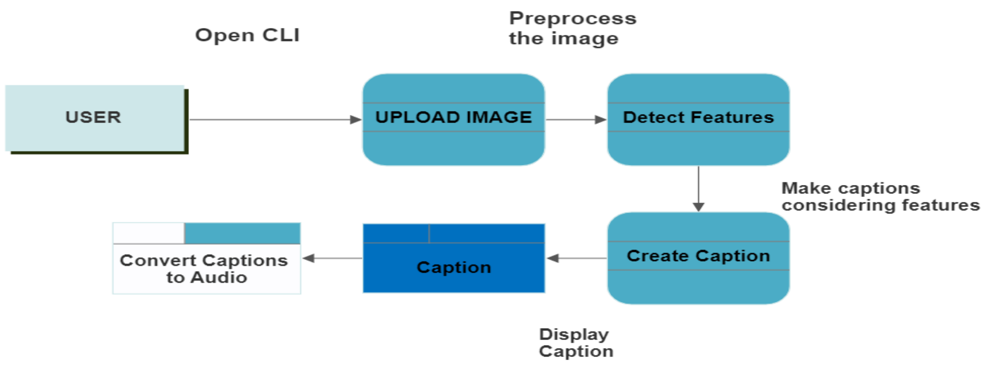


Figure 5.3 DFD Level

- **DFD Level-2**

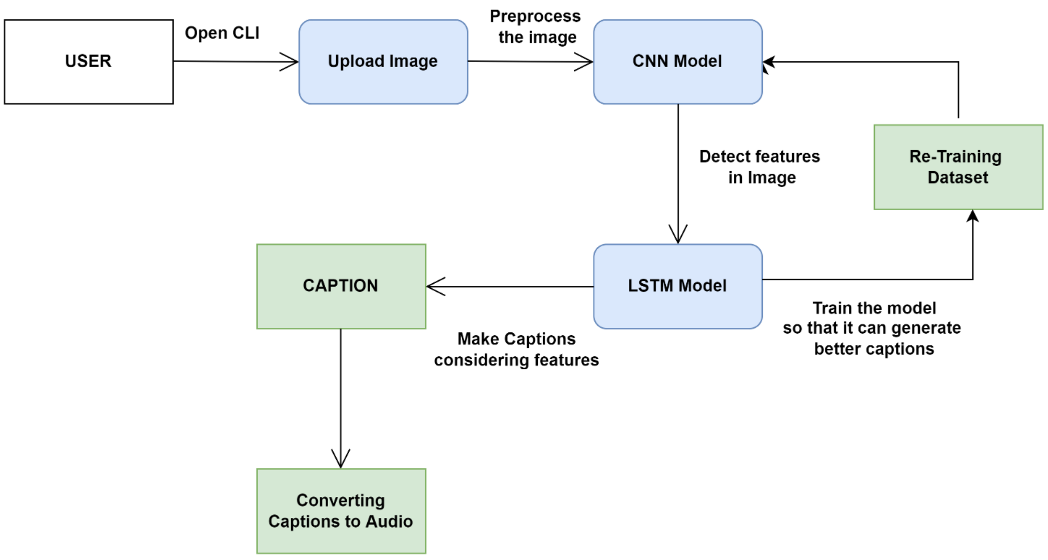

USER
Open CLI
Upload Image
Preprocess the image
CNN Model
Detect features in Image
Re-Training Dataset
CAPTION
LSTM Model
Make Captions considering features
Train the model so that it can generate better captions
Converting Captions to Audio

Figure 5.4 DFD Level-2

5.2.2 UML Diagrams:

- **Use Case Diagram**

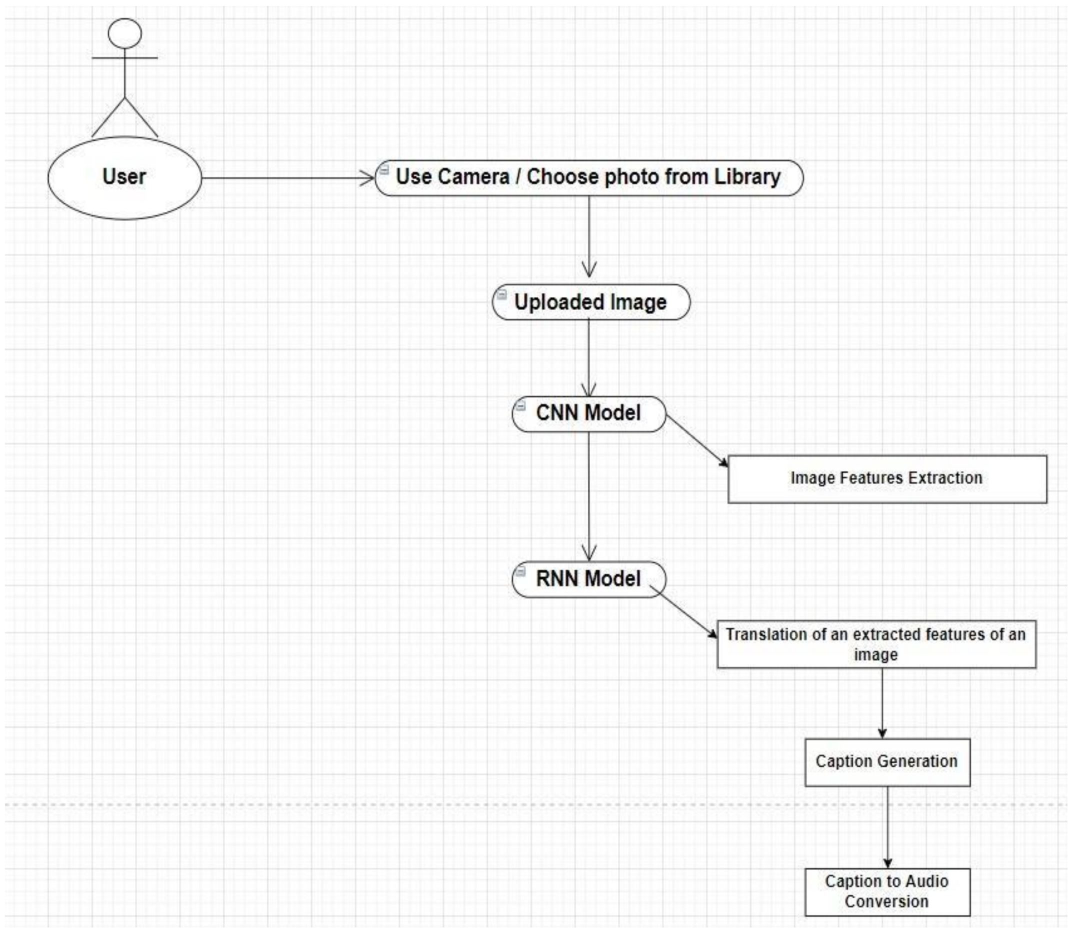

User
Use Camera / Choose photo from Library
Uploaded Image
CNN Model
Image Features Extraction
RNN Model
Translation of an extracted features of an image
Caption Generation
Caption to Audio Conversion

Figure 5.5 Use case Diagram

- **Activity Diagram**

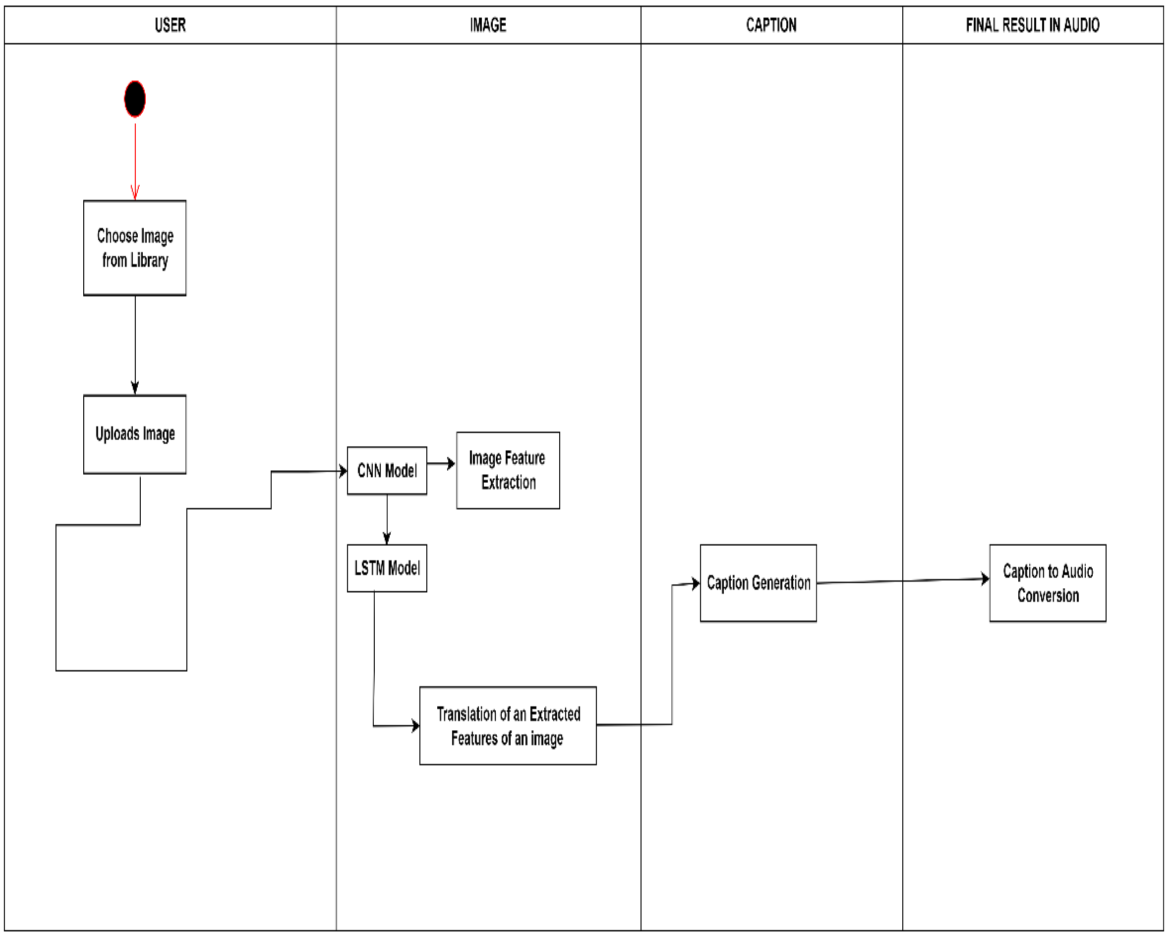


Figure 5.6 Activity Diagram

5.2.3 Flow Chart:

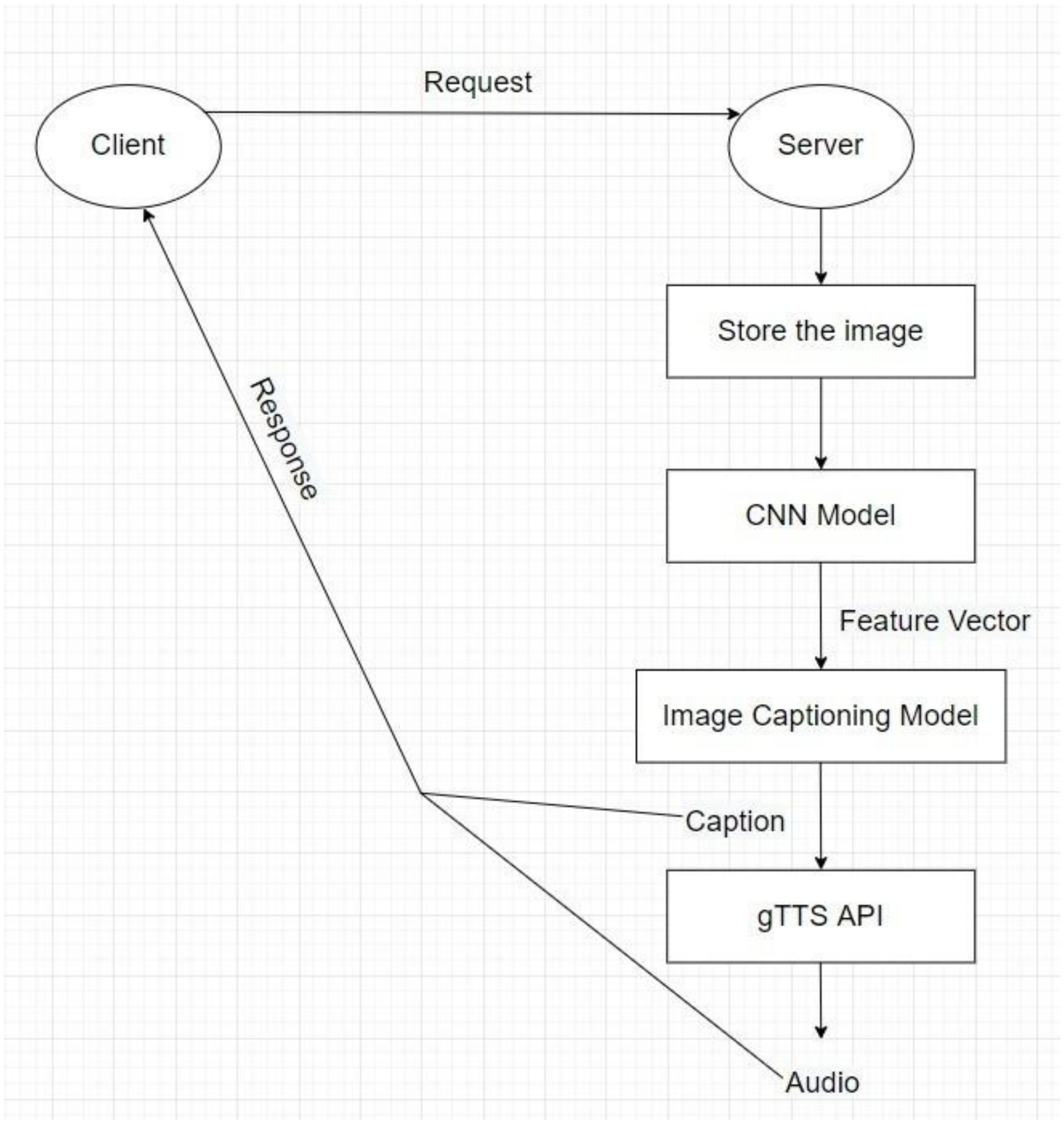


Fig 5.7 Flow Chart

Chapter 6

Implementation Details

6.1 Proposed System

Deep Learning

In this module we have deal with deep learning side of our project. Firstly the data analysis part is done on Jupyter notebook that converted into relevant
modules as required. The main code for this part is there in Model.py file.
It contains functions to pre-process image, encode the image, predict caption, etc.

Server Side Logic

In this module implemented the code to respond to the user's request which is nothing but an image. The image received from frontend will be sent to the model which will send the caption.
This caption will then be converted into an audio and both will be rendered on the frontend.

Front End Design

This module deals with the webpage structure and its design. It will help sending the user request to the server and rendering server's response to the user.

6.1.1 Analysis/Framework/Algorithm

The application is merged with two main architectures CNN and RNN which describes attributes, relationships, objects in the image and puts into words. CNN is an extractor that extracts features from the given image.
RNN- LSTM will be fed with the output of the CNN and following it will describe and generate a caption.

CNN is a Convolutional Neural Network which process the data having the input shape similar to two-dimensional matrix. CNN model has many layers including input layer, Convo Layer, Pooling Layer, Fully-connected layers, Softmax, and Output layers. Input 15layer in CNN is an image. Image data is presented in form of 3D form of matrix. Convo Layer also known as feature extractor where it performs the convolutional operation and calculate the dot products. ReLU is sub layer in Convo layer that converts all negative values to zero. Pooling layer is one where the volume of the image is being reduced once the convolution layer executes. Fully-Connected layers is connection layer that connects one neuron in a layer to other neuron in other layer involving neurons, biases and weights. Softmax layer is used for multi- classification of objects where using formula the objects are classified. Output layer is last layer at CNN model and has the encoded result to be fed to LSTM model.

RNN is Recurrent Neural Network where output of previous step is fed to ongoing step. LSTM (Long Short Term Memory) is an extended version of RNN that are used to the predict the sequence based on the previous step where in it remembers all the steps and also the predicted sequence at every step. It grasps the required information from the processing of inputs as well as forget gate and also it does remove the non-required data

6.2 ScreenShots

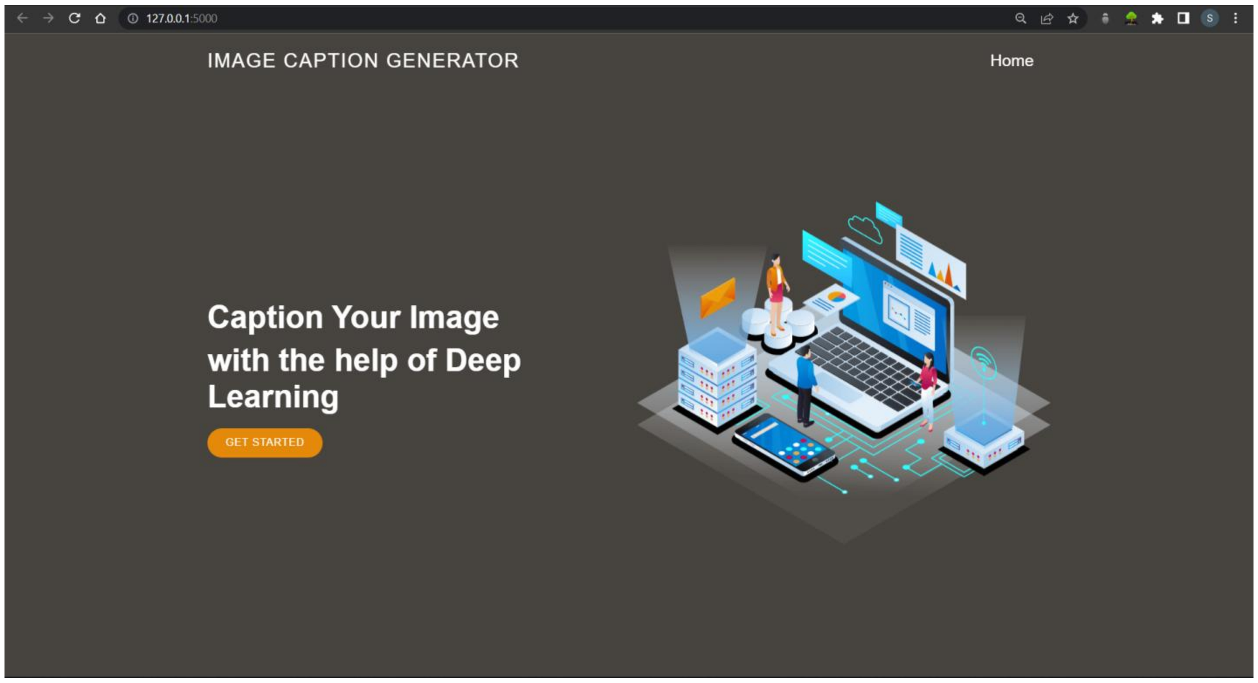


Fig 6.1 Output Screen 1

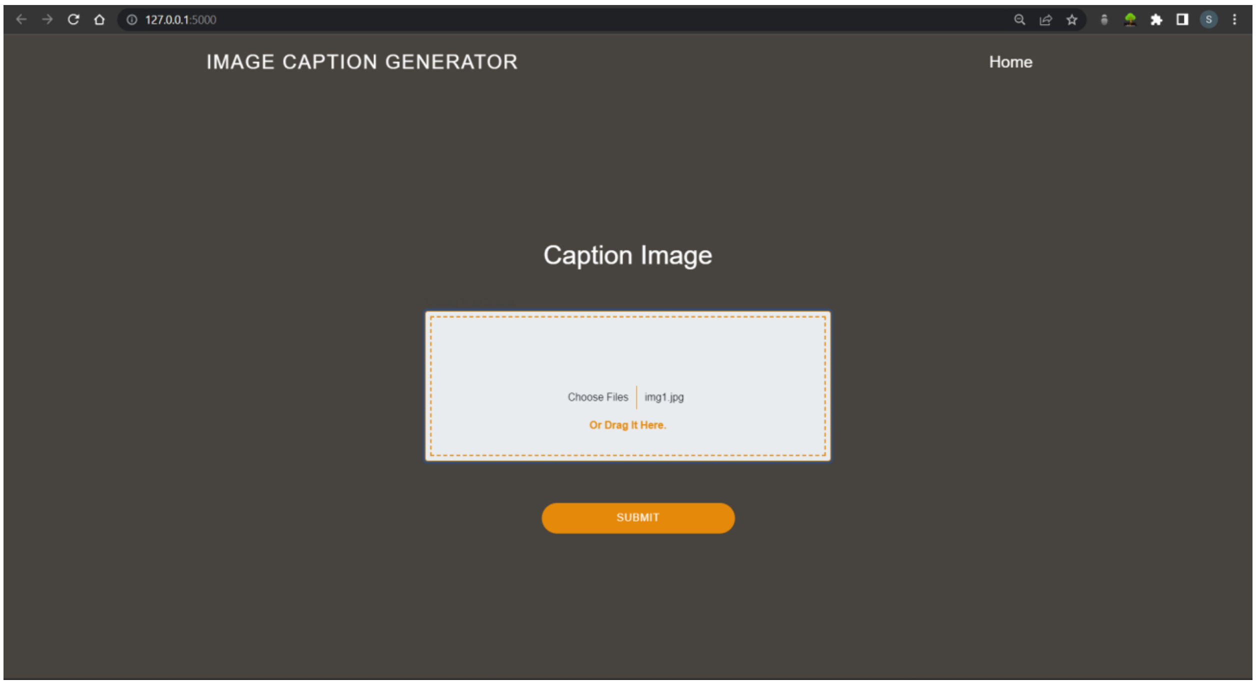
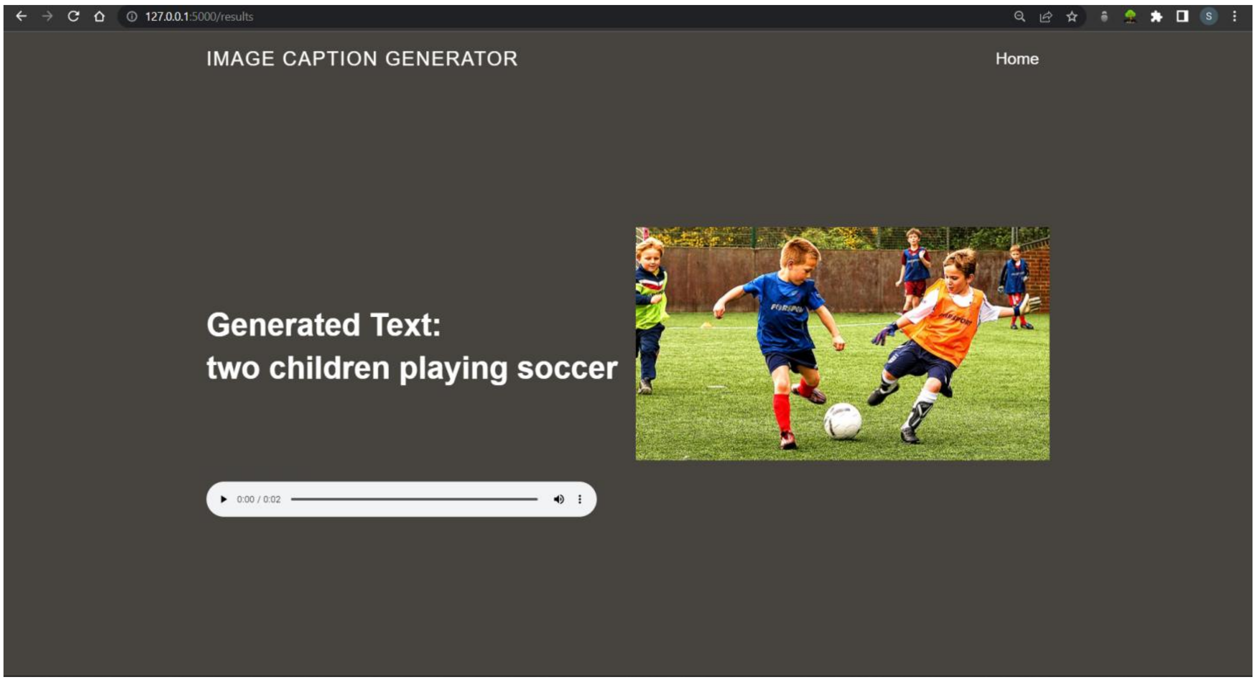
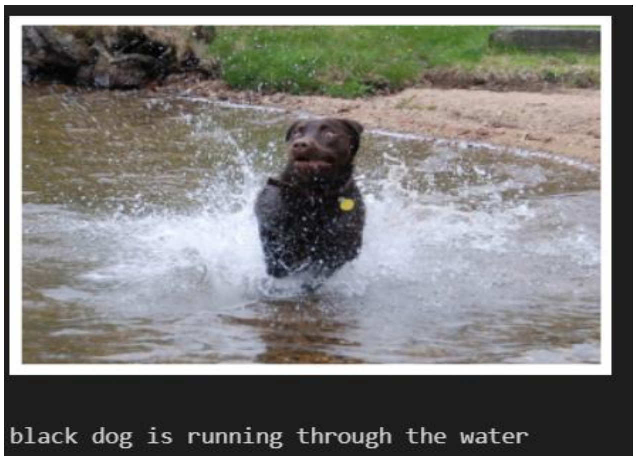


Fig 6.2 Output Screen 2



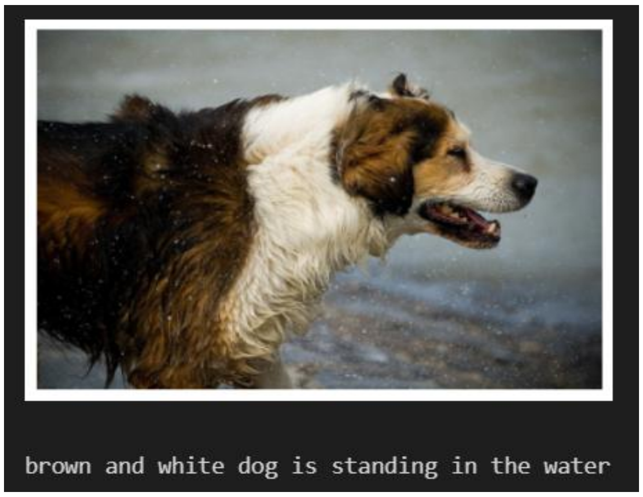
Generated Text:
two children playing soccer

Fig 6.3 Output Screen 3



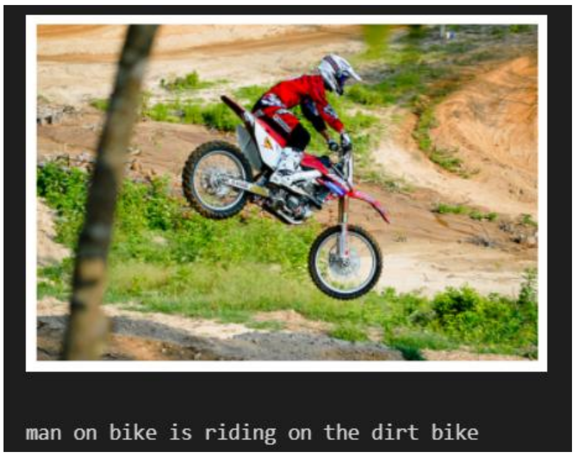
```
black dog is running through the water
```

Fig 6.4 Caption 1



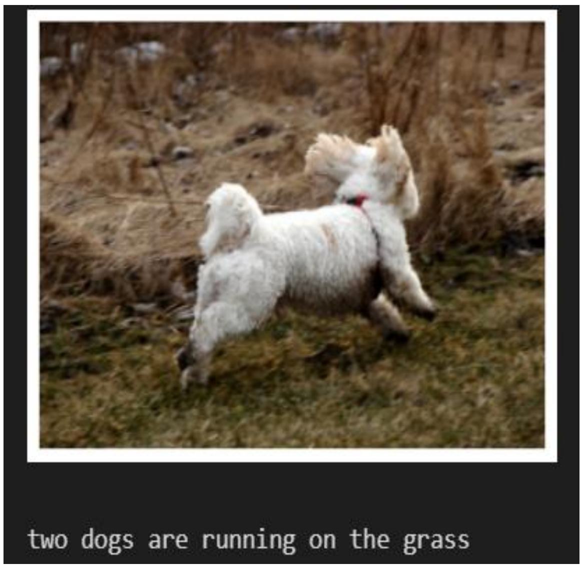
```
brown and white dog is standing in the water
```

Fig 6.5 Caption 2



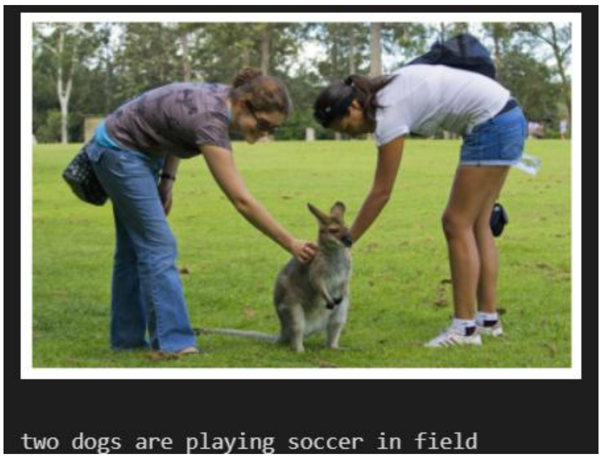
```
man on bike is riding on the dirt bike
```

Fig 6.5 Caption 3



```
two dogs are running on the grass
```

Fig 6.6 Caption 4



```
two dogs are playing soccer in field
```

Fig 6.7 Caption 5

Chapter 7

Technology Used

7.1 Tensorflow

TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries, and community resources.

Currently, the most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

Tensorflow architecture works in three parts:

- Preprocessing the data
- Build the model
- Train and estimate the model

It is called Tensorflow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

7.2 Keras

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend.

Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano. Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras in Python doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

7.3 Pillow

In today's digital world, we come across lots of digital images. In case, we are working with Python programming language, it provides lot of image processing libraries to add image processing capabilities to digital images.

Some of the most common image processing libraries are: OpenCV, Python Imaging Library (PIL), Scikit-image, Pillow. However, in this tutorial, we are only focusing on Pillow module and will try to explore various capabilities of this module.

Pillow is built on top of PIL (Python Image Library). PIL is one of the important modules for image processing in Python. However, the PIL module is not supported since 2011 and doesn't support python 3.

Pillow module gives more functionalities, runs on all major operating system and support for python 3. It supports wide variety of images such as "jpeg", "png", "bmp", "gif", "ppm", "tiff". You can do almost anything on digital images using pillow module. Apart from basic image processing functionality, including point operations, filtering images using built-in convolution kernels, and color space conversions.

7.4 Pandas

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data**.** It is used for data analysis in Python and developed by Wes McKinney in 2008.

Data analysis requires lots of processing, such as restructuring**,** cleaning or merging, etc. There are different tools are available for fast data processing, such as Numpy, Scipy, Cython**,** and Panda. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools. Pandas is built on top of the Numpy package, means Numpy is required for operating the Pandas.

Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze**.**

7.5 Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance

7.6 Matpotlib

Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython.

7.7 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools

7.8 HTML

HyperText Markup Language (HTML) is the set of markup symbols or codes inserted into a file intended for display on the Internet. The markup tells web browsers how to display a web page's words and images.

Each individual piece markup code (which would fall between "<" and ">" characters) is referred to as an element, though many people also refer to it as a tag. Some elements come in pairs that indicate when some display effect is to begin and when it is to end.

7.9 CSS

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

7.10 Javascript

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user. Common examples of JavaScript that you might use every day include the search box on Amazon, a news recap video embedded on The New York Times, or refreshing your Twitter feed.

Incorporating JavaScript improves the user experience of the web page by converting it from a static page into an interactive one.

Details of Hardware and Software

Hardware Requirements

- CPU: Intel processor with 64-bit support
- Disk Storage: 8GB of free disk space.

Software Requirements

- OS: Windows 7 and above, Recommended: Windows 10.
- For Execution: Anaconda Framework in Python and VS Code
- Web Browser (e.g. Google Chrome).

Chapter 8

Test Cases

8.1 Testing Strategy

What is Testing?

Testing is a group of techniques to determine the correctness of the application under the predefined script but, testing cannot find all the defect of application. The main intent of testing is to detect failures of the application so that failures can be discovered and corrected. It does not demonstrate that a product functions properly under all conditions but only that it is not working in some specific conditions.

Testing furnishes comparison that compares the behavior and state of software against mechanisms because the problem can be recognized by the mechanism. The mechanism may include past versions of the same specified product, comparable products, and interfaces of expected purpose, relevant standards, or other criteria but not limited up to these.

Testing includes an examination of code and also the execution of code in various environments, conditions as well as all the examining aspects of the code. In the current scenario of software development, a testing team may be separate from the development team so that Information derived from testing can be used to correct the process of software development.

The success of software depends upon acceptance of its targeted audience, easy graphical user interface, strong functionality load test, etc. For example, the audience of banking is totally different from the audience of a video game. Therefore, when an organization develops a software product, it can assess whether the software product will be beneficial to its purchasers and other audience.

What is software testing?

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.

Software testing provides an independent view and objective of the software and gives surety of fitness of the software. It involves testing of all components under the required services to confirm that whether it is satisfying the specified requirements or not. The process is also providing the client with information about the quality of the software.

Testing is mandatory because it will be a dangerous situation if the software fails any of time due to lack of testing. So, without testing software cannot be deployed to the end user.

Principles of Testing:

- All the test should meet the customer requirements
- To make our software testing should be performed by a third party
- Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
- All the test to be conducted should be planned before implementing it
- It follows the Pareto rule(80/20 rule) which states that 80% of errors come from 20% of program components.
- Start testing with small parts and extend it to large parts.

Types of Testing:

Many of these types of testing can be done manually — or they can be automated.

Accessibility Testing

Accessibility testing is the practice of ensuring your mobile and web apps are working and usable for users without and with disabilities such as vision impairment, hearing disabilities, and other physical or cognitive conditions.

Acceptance Testing

Acceptance testing ensures that the end-user (customers) can achieve the goals set in the business requirements, which determines whether the software is acceptable for delivery or not. It is also known as user acceptance testing (UAT).

Black Box Testing

Black box testing involves testing against a system where the code and paths are invisible.

End to End Testing

End to end testing is a technique that tests the application’s workflow from beginning to end to make sure everything functions as expected.

Functional Testing

Functional testing checks an application, website, or system to ensure it's doing exactly what it's supposed to be doing.

Interactive Testing

Also known as manual testing, interactive testing enables testers to create and facilitate manual tests for those who do not use automation and collect results from external tests.

Integration Testing

Integration testing ensures that an entire, integrated system meets a set of requirements. It is performed in an integrated hardware and software environment to ensure that the entire system functions properly.

Load Testing

This type of non-functional software testing process determines how the software application behaves while being accessed by multiple users simultaneously.

Non Functional Testing

Non functional testing verifies the readiness of a system according to nonfunctional parameters (performance, accessibility, UX, etc.) which are never addressed by functional testing.

Performance Testing

Performance testing examines the speed, stability, reliability, scalability, and resource usage of a software application under a specified workload.

Regression Testing

Regression testing is performed to determine if code modifications break an application or consume resources.

Sanity Testing

Performed after bug fixes, sanity testing determines that the bugs are fixed and that no further issues are introduced to these changes.

Security Testing

Security testing unveils the vulnerabilities of the system to ensure that the software system and application are free from any threats or risks. These tests aim to find any potential flaws and weaknesses in the software system that could lead to a loss of data, revenue, or reputation per employees or outsides of a company.

Single User Performance Testing

Single user performance testing checks that the application under test performs fine according to specified threshold without any system load. This benchmark can be then used to define a realistic threshold when the system is under load.

Smoke Testing

This type of software testing validates the stability of a software application, it is performed on the initial software build to ensure that the critical functions of the program are working.

Stress Testing

Stress testing is a software testing activity that tests beyond normal operational capacity to test the results.

Unit Testing

Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own, speeding up testing strategies and reducing wasted tests.

White Box Testing

White box testing involves testing the product's underlying structure, architecture, and code to validate input-output flow and enhance design, usability, and security.

8.2 White Box Testing

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.
The term “WhiteBox” was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings. Likewise, the “black box” in “Black Box Testing” symbolizes not

being able to see the inner workings of the software so that only the end-user experience can be tested.

- What do you verify in White Box Testing?
- White box testing involves the testing of the software code for the following:
- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into two basic steps. This is what testers do when testing an application using the white box testing technique:

Step 1) Understand the Source Code

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) Create Test cases and Execute

The second basic step to white box testing involves testing the application’s source code for proper flow and structure. One way is by writing more code to test the application’s source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include Manual Testing, trial, and error testing and the use of testing tools.

WhiteBox Testing Example

Consider the following piece of code

```
Printme (int a, int b) {
   int result = a+ b;
   If (result> 0)
         Print ("Positive", result)
   Else
         Print ("Negative", result)
   }
```

The goal of WhiteBox testing in software engineering is to verify all the decision branches, loops, statements in the code.

To exercise the statements in the above white box testing example, WhiteBox test cases would be

A = 1, B = 1
A = -1, B = -3

White Box Testing Techniques

A major White box testing technique is Code Coverage analysis. Code Coverage analysis eliminates gaps in a Test Case suite. It identifies areas of a program that are not exercised by a set of test cases. Once gaps are identified, you create test cases to verify untested parts of the code, thereby increasing the quality of the software product

There are automated tools available to perform Code coverage analysis. Below are a few coverage analysis techniques a box tester can use:

Statement Coverage: This technique requires every possible statement in the code to be tested at least once during the testing process of software engineering.

Branch Coverage: This technique checks every possible path (if-else and other conditional loops) of a software application.

Apart from above, there are numerous coverage types such as Condition Coverage, Multiple Condition Coverage, Path Coverage, Function Coverage etc. Each technique has its own merits and attempts to test (cover) all parts of software code. Using Statement and Branch coverage you generally attain 80-90% code coverage which is sufficient.

Following are important WhiteBox Testing Techniques:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Condition Coverage
- Multiple Condition Coverage
- Finite State Machine Coverage
- Path Coverage
- Control flow testing
- Data flow testing

8.3 Black Box Testing

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

Input
Black Box
Output

Fig 8.1 Black Box Block Diagram

The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

How to do BlackBox Testing?

Here are the generic steps followed to carry out any type of Black Box Testing.

Initially, the requirements and specifications of the system are examined. Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.

- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.

- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

8.4 White Box Testing Versus Black Box Testing

Black Box Testing	**White Box Testing**
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred as outer or external software testing.	It is the inner or the internal software testing.
It is functional test of the software.	It is structural test of the software.
This testing can be initiated on the basis of requirement specifications document.	This type of testing of software is started after detail design document.
No knowledge of programming is required.	It is mandatory to have knowledge of programming.
It is the behavior testing of the software.	It is the logic testing of the software.

Black Box Testing	**White Box Testing**
It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
It is also called closed testing.	It is also called as clear box testing.
It is least time consuming.	It is most time consuming.
It is not suitable or preferred for algorithm testing.	It is suitable for algorithm testing.
Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.
Example: search something on google by using keywords	Example: by input to check and verify loops
Types of Black Box Testing:	Types of White Box Testing:
A. Functional Testing	A. Path Testing
B. Non-functional testing	B. Loop Testing
C. Regression Testing	C. Condition testing

Test Cases:

Test Case Id	Case Description	Expected Result	Actual Result	Pass / Fail
1	Image Format	Various file format (jpeg, png, jpg)	Proper entry of image format	Pass
2	Other File Format	Mp3, mp4, etc	Input should not be accepted	Pass
3	Submitting image	The image should be submitted for caption generation	Caption Generated for input image.	Pass
4	Audio generation	Audio for the generated caption must be produced for visually impaired people	Audio generated along with the caption	Pass
5	Audio Sample Download	Downloading mp3 file for Audio	Mp3 file downloaded in user's download folder	Pass
6	Cleaning Captions	The captions for the image may contain numbers, special characters, underscore signs, etc	The captions must be free from the constraints for better captioning	Pass
7	Vocabulary	Unique Vocabulary	A dictionary that consists of	Pass

	creations	dictionary must be created.	unique words from the captions	
8	Tokenization	Assign unique index to each unique word	A dictionary with key as word and value as index	Pass
9	Start and End Sequence Identifier	Add start and end for marking the beginning and end of the caption	Start and end statement for each caption	Pass

Table 8.1 Test cases

Chapter 9

Project Timeline

9.1 Implementation and plan

- After submission of the project synopsis in this semester, next semester will contain all the implementation part of the project.
- Implementation starts with all the detailed study of algorithm that is going to be implemented.
- Along with the study of algorithm, coding will also be carried out simultaneously.
- Verification and discussion about the project tasks will be done weekly with the in-charge project guide.
- In due course, preparation of the project report will be done which will contain all the detailed information about the project.
- Finally, demonstration of the project will be done with all the requirements.
- At last, a final verification and conclusion of the project will be done with the concern of the in-charge project guide

9.2 Time Line Chart

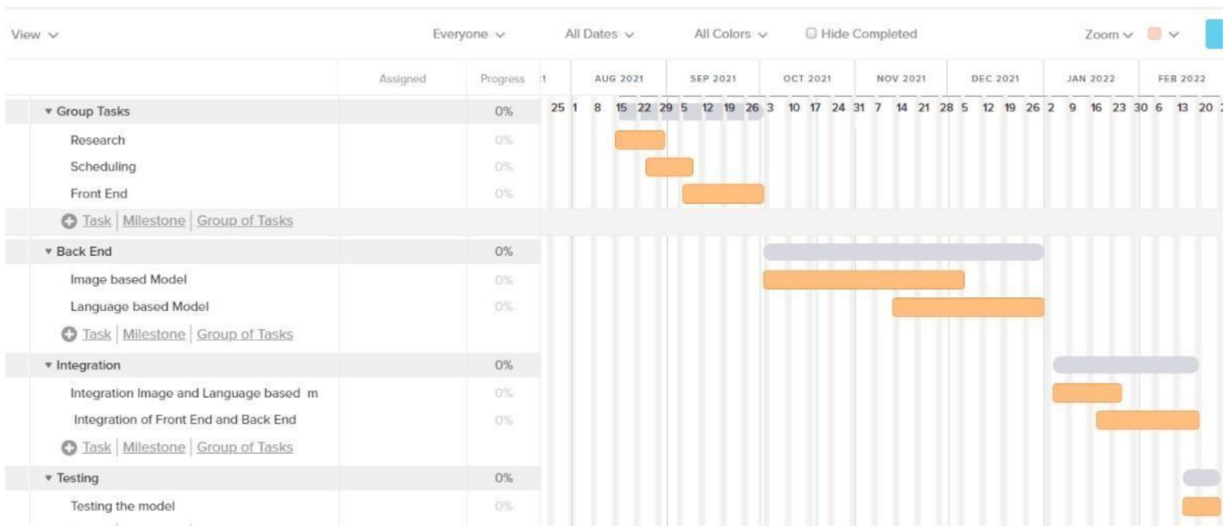


Fig 9.1 Gnatt Chart

Chapter 10

Task Distribution

Tasks	Faizan Qazi	Shayaan Khan	Murtaza Masalawala
Analysis			
Requirement Gathering	✓	✓	✓
Study of Existing System	✓	✓	✓
Study of Technologies	✓	✓	
Study of System Design	✓	✓	
Documentation	✓	✓	✓
Implementation			
UI Development	✓		
Backend Development	✓	✓	✓
Integration	✓	✓	
Testing		✓	
Documentation	✓	✓	✓

Table 10.1 Task Distribution

Chapter 11

Conclusion and Future Work

Conclusion

We have introduced a deep learning model that automatically generate image captions with the goal of helping visually impaired people to understand their environment better. Our proposed model is based upon a CNN feature extraction model that encodes the image into a vector representation followed by an RNN decoder model that generates respective sentences based on the image features learned. The performance is expected to increase on using a bigger dataset with more number of images. The presented model still provides considerable accuracy to aid visually imparied people get a better sense of their surroundings and with the text-to-speech technology we have implemented it makes it more user-friendly.

Future Work

Our model is not flawless and may generate erroneous captions occasionally. It is trained on the Flickr 8k dataset which is comparatively small and less diverse to bigger datasets available like Flickr30k and MSCOCO dataset. Hence to increase its accuracy and make better predictions we will be training our model on larger datasets. Other optimizations include tweaking the hyperparameters like batch size, number of epochs, learning rate, etc and understanding the effect of each of them on our model.

Furthermore, will be developing models using other CNN models like Inception V3, VGG, etc. as the feature extractor. Then we will comparing their results which will help us to analyze the influence of the CNN component over the entire network. This comparison can de using BLEU(Bilingual Evaluation Understudy) Score.

We would also like to explore different methods to generate better captions. Like using Beam Search, which selects a group of words with the maximum likelihood and parallel searches through all the sequence for generating the next word in the sequence unlike our approach which greedily selects the word with maximum probability.

Chapter 12

References

References

[1] V. Julakanti, "Image Caption Generator using CNN-LSTM Deep Neural Network", International Journal for Research in Applied Science and Engineering Technology, vol. 9, no., pp. 2968-2974, 2021.

[2] S.-H. Han and H.-J. Choi, "Domain-Specific Image Caption Generator with Semantic Ontology", IEEE International Conference on Big Data and Smart Computing (BigComp), 2020.

[3] M. Wang, L. Song, X. Yang, and C. Luo, " A parallel-fusion RNN-LSTM architecture for image caption generation ", IEEE International Conference on Image Processing (ICIP), 2016.

[4] S. Amirian, K. Rasheed, T. Taha and H. Arabnia, "Automatic Image and Video Caption Generation With Deep Learning: A Concise Review and Algorithmic Overlap", IEEE Access, vol. 8, pp. 218386-218400, 2020.

[5] S. Shukla, S. Dubey, A. Pandey, V. Mishra, M. Awasthi and V. Bhardwaj, "Image Caption Generator Using Neural Networks", International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp. 01-07, 2021.

[6] M. Panicker, V. Upadhayay, G. Sethi and V. Mathur, "Image Caption Generator", International Journal of Innovative Technology and Exploring Engineering, vol. 10, no. 3, pp. 87-92, 2021.

[7] J. Karan Garg and Kavita Saxena, "Image to Caption Generator", International Journal for Modern Trends in Science and Technology, vol. 6, no. 12, pp. 181-185, 2020.

[8] F. Fang, H. Wang, and P. Tang, "Image Captioning with Word Level Attention”, 25th IEEE International Conference on Image Processing (ICIP), 2018.

[9] Y. Huang, J. Chen, W. Ouyang, W. Wan and Y. Xue, "Image Captioning With End-to-End Attribute Detection and Subsequent Attributes Prediction", IEEE Transactions on Image Processing, vol. 29, pp. 4013-4026, 2020.

[10] P. Mathur, A. Gill, A. Yadav, A. Mishra, and N. K. Bansode, " Camera2Caption: A real-time image caption generator ", International Conference on Computational Intelligence in Data Science (ICCIDS), 2017.

[11] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[12] Y. Zhou, Y. Sun, and V. Honavar, " Improving Image Captioning by Leveraging Knowledge Graphs", IEEE Winter Conference on Applications of Computer Vision (WACV), 2019.
[13] Y. Zhenyu and Z. Jiao, “Research on Image Caption Method Based on Mixed Image Features", IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2019.
[14] M. Tanti, A. Gatt, and K. Camilleri, “What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?", Proceedings of the 10th International Conference on Natural Language Generation, 2017.
[15] Simao Herdade, Armin Kappeler, Kofi Boakye, Joao Soares, “Image Captioning: Transforming Objects into Words.", Proceedings of the 10th International Conference on Natural Language Generation, 2017.
[16] A. Deshpande, J. Aneja, L. Wang, A.G. Schwing, D. Forsyth, “Fast, diverse and accurate image captioning guided by part-of-speech”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[17] R. Subash, “Automatic Image Captioning Using Convolution Neural Networks and LSTM”, Journal of Physics Conference, 2019.

Chapter 13

Source Code

Index.html
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title>Image Captioning</title>

  <!-- Vendor CSS Files -->
  <link href="static/vendor/aos/aos.css" rel="stylesheet">
  <link href="static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="static/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
  <link href="static/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
  <link href="ststic/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="static/vendor/remixicon/remixicon.css" rel="stylesheet">
  <link href="static/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">

  <!-- Template Main CSS File -->

  <link href="static/css/style.css" rel="stylesheet">

</head>

<body>

  <!-- ======= Header ======= -->
  <header id="header" class="fixed-top ">
    <div class="container d-flex align-items-center">

      <h1 class="logo me-auto"><a href="/">Image Caption Generator</a></h1>
      <!-- Uncomment below if you prefer to use an image logo -->
      <!-- <a href="index.html" class="logo me-auto"><img src="assets/img/logo.png" alt=""
class="img-fluid"></a>-->

      <nav id="navbar" class="navbar">
        <ul>
          <li><a class="nav-link home scrollto active" href="/">Home</a></li>
         </ul>
        <i class="bi bi-list mobile-nav-toggle"></i>
      </nav><!-- .navbar -->

    </div>
  </header><!-- End Header -->

  <!-- ======= Hero Section ======= -->
  <section id="hero" class="d-flex align-items-center">

    <div class="container">
      <div class="row">
```

```
            <div class="col-lg-6 d-flex flex-column justify-content-center pt-4 pt-lg-0 order-2
order-lg-1" data-aos="fade-up" data-aos-delay="200">
              <h1>Caption Your Image</h1>
              <h1>with the help of Deep Learning</h1>
              <div class="d-flex justify-content-center justify-content-lg-start">
                <a href="#about" class="btn-get-started scrollto">Get Started</a>
                </div>
            </div>
            <div class="col-lg-6 order-1 order-lg-2 hero-img" data-aos="zoom-in" data-aos-
delay="200">
              <img src="static/img/hero-img.png" class="img-fluid animated" alt="">
            </div>
          </div>
        </div>

      </section><!-- End Hero -->

      <main id="main">

        <!-- ======= About Us Section ======= -->
        <section id="about" class="about">
          <div class="container" data-aos="fade-up">

            <div class="section-title">
              <h1 class="caption">Caption Image</h1>
            </div>


     <div class="container fileUpload">
            <div class="row">
              <div class="col-md-6">
                <form  action="/results" method="POST" enctype="multipart/form-data">
            <div class="form-group files">
              <label>Upload Your Image </label>
              <input type="file" name="userfile" class="form-control" multiple="">
            </div>
            <br>
            <button type="submit" class="btn submit" style="margin-left: 1rem; width:
30vh;">Submit</button>
          </form>
              </div>
      </div>
      <br>
      <br>


     </div>
        </section><!-- End About Us Section -->

      </main><!-- End #main -->
```

```
  <!-- Vendor JS Files -->
  <script src="static/vendor/aos/aos.js"></script>
  <script src="static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script src="static/vendor/glightbox/js/glightbox.min.js"></script>
  <script src="static/vendor/isotope-layout/isotope.pkgd.min.js"></script>
  <script src="static/vendor/php-email-form/validate.js"></script>
  <script src="static/vendor/swiper/swiper-bundle.min.js"></script>
  <script src="static/vendor/waypoints/noframework.waypoints.js"></script>

  <!-- Template Main JS File -->
  <script src="static/js/main.js"></script>

</body>

</html>
```

App.py

```
import os
from playsound import playsound
from gtts import gTTS
from flask import Flask, render_template, url_for, request, redirect
from FinalModel import *
import warnings
warnings.filterwarnings("ignore")


app = Flask(__name__)


@app.route('/')
def hello():
    return render_template('index.html')


@app.route('/results', methods=['POST'])
def upload_file():
    if request.method == 'POST':
        img = request.files['userfile']

        img.save("static/img/"+img.filename)

        caption = caption_this_image("static/img/"+img.filename)

        result_dic = {
            'image': "static/img/" + img.filename,
            'description': caption
        }

        myobj = gTTS(text=caption, lang='en', slow=False, tld='ie')
        myobj.save("static/audio/audio.mp3")

    return render_template('results.html', results=result_dic)
```

```
if __name__ == '__main__':
    app.run(threaded=False)
```

imageCaptioning.ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import keras
import re
import nltk
from nltk.corpus import stopwords
import string
import json
from time import time
import pickle
from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input, decode_predictions
from keras.preprocessing import image
from keras.models import Model, load_model
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from keras.layers import Input, Dense, Dropout, Embedding, LSTM
from keras.layers.merge import add
# Read Captions File
# Reading the Description file

with open("Flickr_8k_text/Flickr8k.token.txt") as filepath:
    captions = filepath.read()
    filepath.close()
captions = captions.split("\n")[:-1]
len(captions)
# creating a "descriptions" dictionary  where key is 'img_name' and value is list of captions corresponding to that image_file.

descriptions = {}

for ele in captions:
    i_to_c = ele.split("\t")
    img_name = i_to_c[0].split(".")[0]
    cap = i_to_c[1]

    if descriptions.get(img_name) == None:
        descriptions[img_name] = []

    descriptions[img_name].append(cap)
descriptions['1000268201_693b08cb0e']
# Data Cleaning
""" 1. lower each word
    2. remove puntuations
    3. remove words less than length 1 """
```

```
def clean_text(sample):
    sample = sample.lower()

    sample = re.sub("[^a-z]+"," ",sample)

    sample = sample.split()

    sample = [s for s in sample if len(s)>1]

    sample = " ".join(sample)

    return sample
clean_text("My noghsujf si am m cricket101 &8 mphi*&86%%&??,BY6fajdn 213 q rqu243 boy  32 ewr wO>>J DHD 34  asfb HHGY Gvg HgB   231 123")
#  modify all the captions i.e - cleaned captions

for key, desc_list in descriptions.items():
    for i in range(len(desc_list)):
        desc_list[i] = clean_text(desc_list[i])
#  clean descriptions

descriptions['1000268201_693b08cb0e']
# writing clean description to .txt file

f = open("descriptions.txt","w")
f.write( str(descriptions) )
f.close()
#  reading description file

f = open("storage/descriptions.txt", 'r')
descriptions = f.read()
f.close()

json_acceptable_string = descriptions.replace("'", "\"")
descriptions = json.loads(json_acceptable_string)
# finding the unique vocabulary

vocabulary = set()

for key in descriptions.keys():
    [vocabulary.update(i.split()) for i in descriptions[key]]

print('Vocabulary Size: %d' % len(vocabulary))

#  All words in description dictionary
all_vocab =  []

for key in descriptions.keys():
    [all_vocab.append(i) for des in descriptions[key] for i in des.split()]

print('Vocabulary Size: %d' % len(all_vocab))
print(all_vocab[:15])
```

# count the frequency of each word, sort them and discard the words having frequency lesser than threshold value

```
import collections


counter= collections.Counter(all_vocab)

dic_ = dict(counter)

threshelod_value = 10

sorted_dic = sorted(dic_.items(), reverse=True, key = lambda x: x[1])
sorted_dic = [x for x in sorted_dic if x[1]>threshelod_value]
all_vocab = [x[0] for x in sorted_dic]
len(all_vocab)
# Loading Training Testing Data
# TrainImagesFile
f = open("Flickr_8k_text/Flickr_8k.trainImages.txt")
train = f.read()
f.close()
train  = [e.split(".")[0] for e in train.split("\n")[:-1]]
# TestImagesFile
f = open("Flickr_8k_text/Flickr_8k.testImages.txt")
test = f.read()
f.close()
test  = [e.split(".")[0] for e in test.split("\n")[:-1]]
# create train_descriptions dictionary, which will be similar to earlier one, but having only train samples
# add startseq + endseq

train_descriptions = {}

for t in train:
    train_descriptions[t] = []
    for cap in descriptions[t]:
        cap_to_append = "startseq " + cap + " endseq"
        train_descriptions[t].append(cap_to_append)
train_descriptions['1000268201_693b08cb0e']
# Data Preprocessing - Images
"""
In this section, we will load our images and do some processing so that we can feed it in our network.
"""
model = ResNet50(weights="imagenet", input_shape=(224,224,3))
model.summary()
# Create a new model, by removing the last layer (output layer of 1000 classes) from the resnet50
model_new = Model(model.input, model.layers[-2].output)
images = "Flickr8k_Dataset/"
def preprocess_image(img):
    img = image.load_img(img, target_size=(224,224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    return img
def encode_image(img):
    img = preprocess_image(img)
    feature_vector = model_new.predict(img)
    feature_vector = feature_vector.reshape(feature_vector.shape[1],)
    return feature_vector
start = time()

encoding_train = {}

for ix, img in enumerate(train):

    img = "Flicker8k_Dataset/{}.jpg".format(train[ix])
    encoding_train[img[len(images):]] = encode_image(img)

    if ix%100==0:
        print("Encoding image- "+ str(ix))

print("Time taken in seconds =", time()-start)
# Save the bottleneck train features to disk

with open("./storage/encoded_train_images.pkl", "wb") as encoded_pickle:
    pickle.dump(encoding_train, encoded_pickle)

start = time()

encoding_test = {}

for ix, img in enumerate(test):

    img = "Flicker8k_Dataset/{}.jpg".format(test[ix])
    encoding_test[img[len(images):]] = encode_image(img)

    if ix%100==0:
        print("Encoding image- "+ str(ix))

print("Time taken in seconds =", time()-start)
# Save the bottleneck train features to disk

with open("./storage/encoded_test_images.pkl", "wb") as encoded_pickle:
    pickle.dump(encoding_test, encoded_pickle)

# Load the train images features from disk

with open("./storage/encoded_train_images.pkl", "rb") as encoded_pickle:
    encoding_train = pickle.load(encoded_pickle)
# Load the test images features from disk

with open("./storage/encoded_test_images.pkl", "rb") as encoded_pickle:
    encoding_test = pickle.load(encoded_pickle)

# Data Preprocessing - Captions
```

```
"""
word_to_idx is mapping between each unique word in all_vocab to int value
and idx_to_word is vice-versa
"""


ix = 1
word_to_idx = {}
idx_to_word = {}

for e in all_vocab:
    word_to_idx[e] = ix
    idx_to_word[ix] = e
    ix +=1
#  need to add these 2 words as well

word_to_idx['startseq'] = 1846
word_to_idx['endseq'] = 1847

idx_to_word[1846] = 'startseq'
idx_to_word[1847] = 'endseq'
#  vocab_size is total vocabulary len +1 because we will append 0's as well.

vocab_size = len(idx_to_word)+1
print(vocab_size)
all_captions_len = []

for key in train_descriptions.keys():
    for cap in train_descriptions[key]:
        all_captions_len.append(len(cap.split()))

max_len = max(all_captions_len)
print(max_len)

# Data Preparation using Generator Function

def data_generator(train_descriptions,encoding_train,word_to_idx,max_len,batch_size):
    X1,X2, y = [],[],[]

    n =0
    while True:
        for key,desc_list in train_descriptions.items():
            n += 1

            photo = encoding_train[key+".jpg"]
            for desc in desc_list:

                seq = [word_to_idx[word] for word in desc.split() if word in word_to_idx]
                for i in range(1,len(seq)):
                    xi = seq[0:i]
                    yi = seq[i]

                    #0 denote padding word
                    xi = pad_sequences([xi],maxlen=max_len,value=0,padding='post')[0]
                yi = to_categorcial([yi],num_classes=vocab_size)[0]

                X1.append(photo)
                X2.append(xi)
                y.append(yi)

            if n==batch_size:
                yield [[np.array(X1),np.array(X2)],np.array(y)]
                X1,X2,y = [],[],[]
                n = 0

# Word Embedding
f = open("GloVE/glove.6B.50d.txt", encoding='utf8')
embedding_index = {}

for line in f:
    values = line.split()

    word = values[0]
    word_embedding = np.array(values[1:],dtype='float')
    embedding_index[word] = word_embedding
## Converting words into vectors  Directly - (Embedding Layer Output)
def get_embedding_matrix():
    emb_dim = 50
    matrix = np.zeros((vocab_size,emb_dim))
    for word,idx in word_to_idx.items():
        embedding_vector = embedding_index.get(word)

        if embedding_vector is not None:
            matrix[idx] = embedding_vector

    return matrix
# embedding_output.shape
embedding_matrix = get_embedding_matrix()
embedding_matrix.shape

# Model Architecture
# image feature extractor model

input_img_fea = Input(shape=(2048,))
inp_img1 = Dropout(0.3)(input_img_fea)
inp_img2 = Dense(256, activation='relu')(inp_img1)
# partial caption sequence model

input_cap = Input(shape=(max_len,))
inp_cap1 = Embedding(input_dim=vocab_size, output_dim=50, mask_zero=True)(input_cap)
inp_cap2 = Dropout(0.3)(inp_cap1)
inp_cap3 = LSTM(256)(inp_cap2)
decoder1 = add([inp_img2 , inp_cap3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)

# Merge 2 networks
model = Model(inputs=[input_img_fea, input_cap], outputs=outputs)
```

```
model.summary()
# model.layers[2].set_weights([embedding_output])
# model.layers[2].trainable = False
# Embedding Layer most important
model.layers[2].set_weights([embedding_matrix])
model.layers[2].trainable = False
model.compile(loss="categorical_crossentropy", optimizer="adam")
# Train Our Model
epochs = 20
batch_size = 3
steps = len(train_descriptions)//64
def train():

   for i in range(epochs):
      generator                                                                =
data_generator(train_descriptions,encoding_train,word_to_idx,max_len,batch_size)
      model.fit_generator(generator,epochs=1,steps_per_epoch=steps,verbose=1)
      model.save('model_weights/model_'+str(i)+'.h5')
model = load_model("./model_weights/model_9.h5")
# Predictions
def predict_caption(photo):
   in_text = "startseq"

   for i in range(max_len):
      sequence = [word_to_idx[w] for w in in_text.split() if w in word_to_idx]
      sequence = pad_sequences([sequence], maxlen=max_len, padding='post')

      ypred =  model.predict([photo,sequence])
      ypred = ypred.argmax()
      word = idx_to_word[ypred]
      in_text+= ' ' +word

      if word =='endseq':
         break


   final_caption =  in_text.split()
   final_caption = final_caption[1:-1]
   final_caption = ' '.join(final_caption)

   return final_caption


for i in range(3):
   rn =  np.random.randint(0, 1000)
   img_name = list(encoding_test.keys())[rn]
   photo = encoding_test[img_name].reshape((1,2048))

   i = plt.imread("Flicker8k_Dataset/"+img_name)
   plt.imshow(i)
   plt.axis("off")
   plt.show()

   caption = predict_caption(photo)
```

```
    print(caption)
```

finalModel.py

```
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input, decode_predictions
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing import image
from keras.models import load_model, Model

import matplotlib.pyplot as plt
import pickle
import numpy as np

import warnings
warnings.filterwarnings("ignore")


model = load_model("./model_weights/model_9.h5")
model.make_predict_function()

model_temp = ResNet50(weights="imagenet", input_shape=(224, 224, 3))

# Create a new model, by removing the last layer (output layer of 1000 classes) from the resnet50
model_resnet = Model(model_temp.input, model_temp.layers[-2].output)
model_resnet.make_predict_function()


# Load the word_to_idx and idx_to_word from disk

with open("./storage/word_to_idx.pkl", "rb") as w2i:
    word_to_idx = pickle.load(w2i)

with open("./storage/idx_to_word.pkl", "rb") as i2w:
    idx_to_word = pickle.load(i2w)

max_len = 35


def preprocess_image(img):
    img = image.load_img(img, target_size=(224, 224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    return img


def encode_image(img):
    img = preprocess_image(img)
    feature_vector = model_resnet.predict(img)
    feature_vector = feature_vector.reshape(1, feature_vector.shape[1])
    return feature_vector


def predict_caption(photo):
    in_text = "startseq"

    for i in range(max_len):
        sequence = [word_to_idx[w]
                    for w in in_text.split() if w in word_to_idx]
        sequence = pad_sequences([sequence], maxlen=max_len, padding='post')

        ypred = model.predict([photo, sequence])
        ypred = ypred.argmax()
        word = idx_to_word[ypred]
        in_text += ' ' + word

        if word == 'endseq':
            break

    final_caption = in_text.split()
    final_caption = final_caption[1:-1]
    final_caption = ' '.join(final_caption)

    return final_caption


def caption_this_image(input_img):

    photo = encode_image(input_img)

    caption = predict_caption(photo)
    # keras.backend.clear_session()
    return caption
```

Chapter 14

Paper Proceedings

Volume: 10 Issue: IV Month of publication: April 2022

DOI:

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue IV Apr 2022- Available at www.ijraset.com

Generate Detailed Captions of an Image using Deep Learning

Khan Shayaan Shakeel[1], Masalawala Murtaza Shabbir[2], Qazi Faizan Ahmed[3], Nafisa Mapari[4]
[1, 2, 3, 4]Computer Engineering, M. H. Saboo. Siddik COE, Mumbai, India

***Abstract:** This paper shows the implementation of image caption generation using deep learning algorithm. The project is one of the primary example of computer vision. The main aim of computer vision is scene undertanding. The algorithms used in this model are CNN and LSTM. This model is an extension of the model based on CNN - RNN Model which suffers from the drawback of vanishing gradient. Xception model is used for image feature extraction and is a CNN model that is trained using ImageNet dataset. Extracted features from the Xception model is fed as the input to the LSTM model which in turn generates the caption for the image. The dataset used for training and testing is Flickr_8k dataset.*

I. INTRODUCTION

For humans the process of describing an image is a very simple process but for computers we achieve this with the help of computer vision. The main goal of computer is to understand the scenario in an image. Not only should it understand the image but also it should be able to express it in the human language. Image captioning is a process where the system must be capable enough to distinguish between the different objects and then later express it in the terms of language which is understood by humans. We create a system that links the objects in the image and creates a logical sequence. This logical sequence of description comes with the help of learning the data and the dataset consists of images along with descriptions that help us to train our model and predict the results.

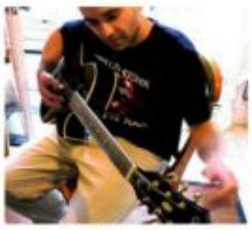

"man in black shirt is playing guitar."

Figure 1: Example of man playing guitar

In the above diagram we can see that the caption generated is very accurate. The general idea is to divide the system into logically 2 modules where the first module is an Image based model and the other is a Language based model.
Image based model is built with the help of Convolutional Neural Network (CNN). This model is used to extract the features from the image and identifies the different segments of an image and assigns weight to it, which helps in the classification of the image. CNN is found to be very useful in image classification however our main goal is to extract the features. CNN is generally used in layers where the output of the first layer is fed as the input to the second layer and so on. After a series of layers we get the vectorial representation of image which is fed as an input to the language based model.
Language based model is built with the help of a Long Short Term Memory Network (LSTM). LSTM is a type of Recurrent Neural Network and is generally is used in sequence prediction problems. RNN can also be used for sequence prediction but the limitation with RNN is short term memory, as a result LSTM is found to be more efficient for predicting sequence.

II. RELATED WORK

This section gives detailed information about the research work that has been done on Image Caption Generation. Recently the quality of image caption generation has improved considerably by using combinations of CNN to obtain vectorial representation of images and RNN to decode those representations into natural language sentences.

Yao et al have published a research paper that explains the process of Image and Video to Text conversion. The entire process is based on Image Comprehension.

The process is divided into three steps. In the first step visual features are extracted. In the second step the output of the first stage is given as input to second stage which converts it into textual description. In the final stage the description is transformed into semantically meaningful, human understandable captions. Users can not only obtain captions for images but for videos as well.

Li et al have published a paper that incorporates storytelling for videos. The main aim is to produce coherent and concise stories for long videos.

With the help of the Multimodal Embedding Research, they have designed a Residual Bidirectional RNN to use past and future contextual knowledge. Multimodal embedding is also used for video clip phrases.

O. Vinyals et al have developed a model known as NIC which is a end-to-end neural network model that automatically generates caption for the input image [4]. The entire model is dependent on CNN which is used for features extraction and then later it is trained by a RNN to generate sentences.

This system has proved to be producing accurate results for larger datasets. The model quantitative evaluations is done either by using BLEU or ranking metrics to assess the generated descriptions.

S. Shabir, S. Arafat et al have published that since there are many research is going on to find new ways for generating captions, they have given detailed overview over technical aspects and techniques of image captioning. The research paper is all about the most common process for image captioning to new ways that have been discovered. The research paper also talks about the all related points in detail. The paper has even proposed the fields where the potential efforts should be made in order to improve the results.

Hao Fang et al have published a system that divides the process of image caption generation into three major steps. First the system reasons with the image sub-regions rather than the entire image. Next with the help of the CNN the features from the sub-regions are extracted and then fine-tuned on the training data.

The training is done at Maximum Entropy (ME) from training data set descriptions. This training results in capturing of commonsense knowledge about the image through language statistics. The final stage is re-ranking of a set of high-likelihood sentences by a linear weighting.

These weights are assigned on the basis of Minimum Error Rate Training (MERT). In addition to this they have used Deep Multimodal Similarity Model (DMSM) that maps the similarity between text and image. This in turn improves the selection of quality captions.

Kelvin Xu el at have proposed a system with two approaches. The first one is soft deterministic attention mechanism that is trained on the basis of standard back-propagation methods and the second one is hard deterministic attention mechanism which is trained by maximizing an approximate variational lower bound or by REINFORCE. The paper showcase the how we can gain insights and interprets the results. It visualize where and what the attention is focused on in an image. The paper also show the usefulness of the caption generated by evaluating it against state of art performance.

III. METHODOLOGY

A. System Design

The entire module can be logically divided into two modules:

1) Image Based Model
2) Language Based model

In the Image Based Model the input image is converted into vectorial representations. For image based model convolutional neural networks are used in combinations and is also known as the Encoder. In the Language Based model the vectorial representations are converted into natural language. The vectorial representations are decoded with the help of the LSTM network and this model is also known as the Decoder.

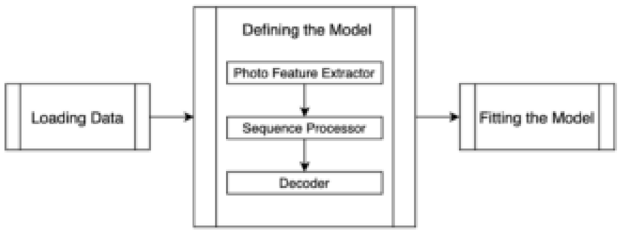


Figure 2: Model Structure

Both the models are integrated together to complete the entire process of image to text generation and later with the help of pyttsx3 or gtts we can convert the generated caption to speech, which will help the visually impaired people.

The process is divided into following logical steps:
Step1: Firstly after pre-processing the data, the data is loaded into the model for training
Step2: This step can be further bifurcated into three more steps:

- Photo Feature Extractor: In this module, it will extract the features from the image by using different combinations of convolutional neural networks.
- Sequence Processor: The output from the Photo Feature Extractor is then fed into the sequence processor and this uses Long Short Term Memory Network (LSTM) for managing texts.
- Decoder: It produces the most logically correct sequence by combining sequence processor and photo feature extractor.

B. Data Collection

The dataset used for training and testing model is flickr_8k dataset and is divided into 6000 images for training, 1000 images for validation and 1000 images for testing. The dataset consist of two directories:

1) Flickr8k_dataset: It consists of 8092 photographs in JPEG format.
2) Flickr8k_text: It consists of number of files having descriptions for the image.

C. Convolutional Neural Network (CNN)

A Convolutional layer is also known as CNN or CoonvNet and consist of three layers viz. convolutional layer, pooling layer and fully connected layer.

1) *Convolutional Layer:* All the load of computational work is handled by the convolutional layer. This layer performs dot product between two matrices, where one matrix represents the kernel i.e. learnable parameters and the other matrix represents restricted portion of the receptive field. The kernel is smaller than an image but it is more in depth. The Kernel slides across the height and width of the image, producing a two dimensional representation of the image.
2) *Pooling Layer:* The pooling layer is used to derive summary statistics of nearby outputs at certain locations. This results in reducing the spatial size of representation which in turn reduces the computation and weights. Every slice of the representation has its own pooling operation. Several pooling opeartions are there such as the average of the rectangular neighbourhood, L2 norm of the neighbourhood, weighted average based on the distance from the central pixel, but the most commonly used is the Max pooling, where the maximum of the neighbourhood is taken into consideration.
3) *Fully Connected Layer:* This layer helps to map the input and output of all the layers. The Neurons in this layer are fully connected with all the neurons in the preceding and succeeding layers. CNN is a Deep Learning Algorithm and uses the concept of weights for image classification. CNN assign weights to different objects present in the image, which helps in the classification of image. For vectorial representation of image, layers of CNN are used together. The output of first layer is fed as an input to the second layer and this process continues for all the subsequent layers. After a series of convolutional network, it is necessary to connect a fully connected layer and which result in a N dimensional vector which is in the encoded form.

D. Long Short Term Memory (LSTM)

LSTM is a type of a Recurrent Neural network. RNN and LSTM are generally used for predicting orders. The idea behind using LSTM is that when we go into deep neural networks, if the gradients are very low or zero then the training cannot take place which eventually leads to poor prediction performance.

Long Short Term Memory is an advanced RNN algorithm which overcomes the limitations of traditional RNN. RNN remembers the past information and uses it for it's current operation, but due to the short term memory (also known as vanishing gradient) it cannot remember long term dependencies. LSTM overcomes the limitations of the traditional RNN and proves to be more efficient in long term sequence prediction.

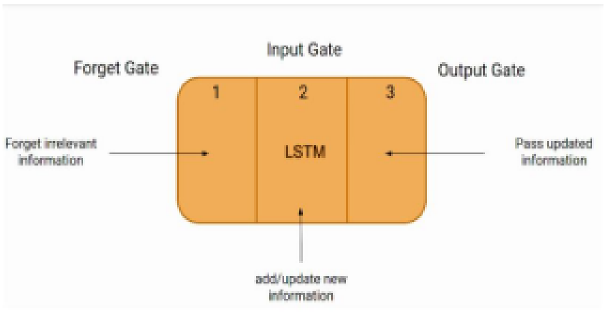
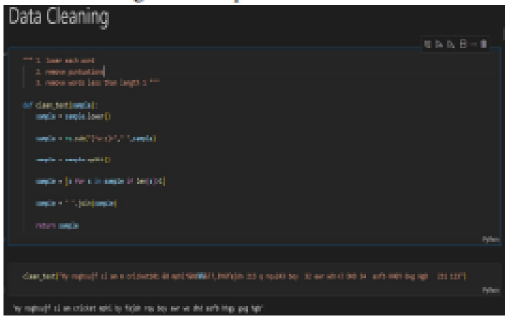


Figure 3: LSTM Gates

LSTM consists of three parts. The first part tells whether the information coming from the previous timestamp is relevant or not and if it is not then the information is discarded. The second part tries to learn new information from the input provided to it and finally the third part that helps in updating the information from current timestamp to the next timestamp. Depending upon the functionality of all the three parts they are known as the Forget gate, the Input gate and the Output gate respectively.

IV. IMPLEMENTATION

1) *Data Preprocessing and Cleaning:* This process starts by loading the text file for data cleaning. The motive behind data cleaning is to ensure that our text doesn't contains any punctuation marks, converting the entire text into lowercase and removing words that contains numbers or taking care of stop words.



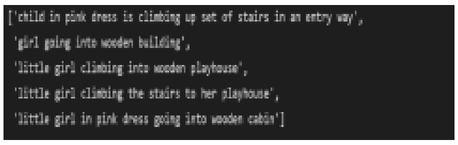
Data Cleaning

```
""" 1. lower each word
    2. remove punctuation
    3. remove words less than length 1 """

def clean_text(sample):
    sample = sample.lower()

    sample = re.sub("[^a-z]+"," ",sample)

    sample = sample.split()

    sample = [s for s in sample if len(s)>1]

    sample = " ".join(sample)

    return sample
```

```
clean_text([illegible])
```

```
[illegible]
```

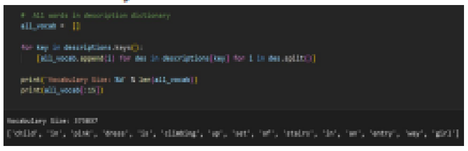
Figure 4: Data Cleaning



```
['child in pink dress is climbing up set of stairs in an entry way',
 'girl going into wooden building',
 'little girl climbing into wooden playhouse',
 'little girl climbing the stairs to her playhouse',
 'little girl in pink dress going into wooden cabin']
```

Figure 5: Example of clean descriptions

2) *Creating Vocabulary for the Image:* Machines cannot handle raw text. First the cleaning of the text is important which is done by splitting it into words, handling punctuations and removing words with the numbers. Each unique word is mapped to a unique index value which could be understood by the machines.

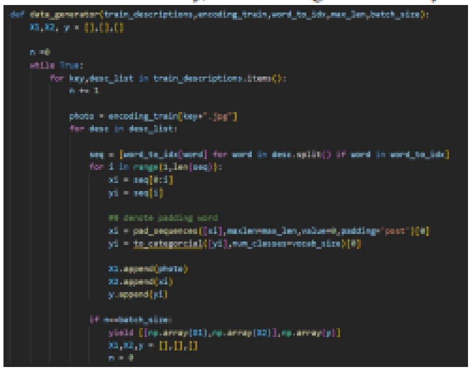


```
# All words in description dictionary
all_vocab = []

for key in descriptions.keys():
    [all_vocab.append(i) for des in descriptions[key] for i in des.split()]

print('Vocabulary Size: %d' % len(all_vocab))
print(all_vocab[:15])
```

```
Vocabulary Size: 373837
['child', 'in', 'pink', 'dress', 'is', 'climbing', 'up', 'set', 'of', 'stairs', 'in', 'an', 'entry', 'way', 'girl']
```

Figure 6: Vocabulary Creation

3) *Data Generator:* For this supervised model 6000 input images are provided and each image has 4096 length feature vector. This amount of large data cannot be stored in the memory, so we use a generator that yields batches.



```
def data_generator(train_descriptions,encoding_train,word_to_idx,max_len,batch_size):
    X1,X2, y = [],[],[]

    n =0
    while True:
        for key,desc_list in train_descriptions.items():
            n += 1

            photo = encoding_train[key+".jpg"]
            for desc in desc_list:

                seq = [word_to_idx[word] for word in desc.split() if word in word_to_idx]
                for i in range(1,len(seq)):
                    xi = seq[0:i]
                    yi = seq[i]

                    #0 denote padding word
                    xi = pad_sequences([xi],maxlen=max_len,value=0,padding='post')[0]
                    yi = to_categorical([yi],num_classes=vocab_size)[0]

                    X1.append(photo)
                    X2.append(xi)
                    y.append(yi)

                if n==batch_size:
                    yield [[np.array(X1),np.array(X2)],np.array(y)]
                    X1,X2,y = [],[],[]
                    n = 0
```

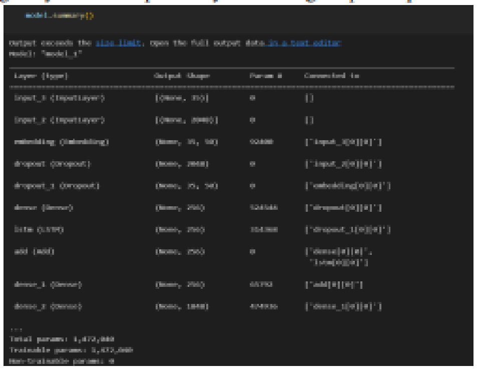
Figure 7: Data Generator

4) *CNN-LSTM Model:* With the help of CNN-LSTM we generate the captions. The Structure of the model is built with the help of keras library that consist of three vital components. These are as follows:

a) Photo Feature Extractor: This module will extract the features from the image by using different combinations of convolutional neural networks.

b) Sequence Processor: The output from the Photo Feature Extractor is fed to the sequence processor. It uses Long Short Term Memory Network (LSTM) for managing text.

c) Decoder: It produces the most logically correct sequence by combining sequence processor and photo feature extractor.

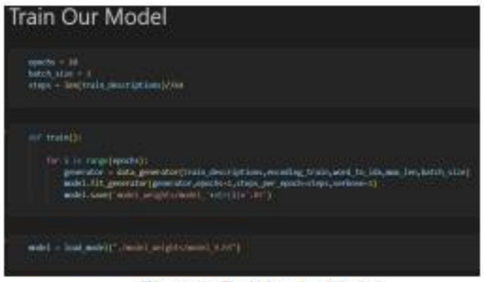


```
model.summary()
```

```
Output exceeds the size limit. Open the full output data in a text editor
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_3 (InputLayer)           [(None, 35)]         0           []

 input_2 (InputLayer)           [(None, 2048)]       0           []

 embedding (Embedding)          (None, 35, 50)       92400       ['input_3[0][0]']

 dropout (Dropout)              (None, 2048)         0           ['input_2[0][0]']

 dropout_1 (Dropout)            (None, 35, 50)       0           ['embedding[0][0]']

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

 lstm (LSTM)                    (None, 256)          314368      ['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, 1848)         474936      ['dense_1[0][0]']

...
Total params: 1,472,040
Trainable params: 1,472,040
Non-trainable params: 0
```

Figure 8: Model Summary

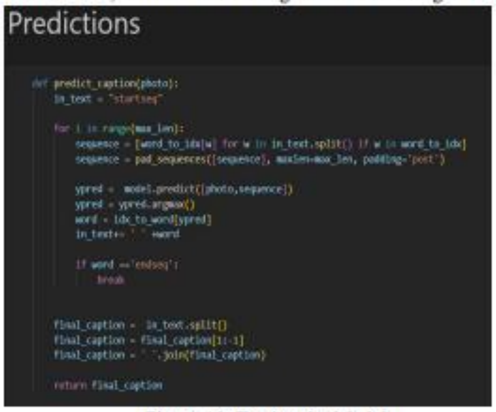
5) *Training the Model:* The flickr_8k dataset consists of 6000 images in jpeg format for training. Each image has 5 descriptive captions.


Train Our Model

```
epochs = 10
batch_size = 3
steps = len(train_descriptions)//64

def train():

    for i in range(epochs):
        generator = data_generator(train_descriptions,encoding_train,word_to_idx,max_len,batch_size)
        model.fit_generator(generator,epochs=1,steps_per_epoch=steps,verbose=1)
        model.save('model_weights/model_'+str(i)+'.h5')

model = load_model("./model_weights/model_9.h5")
```

Figure 9: Training the Model

6) *Testing the Model:* After the model is trained, we test the model against random images and evaluate the generated captions.


Predictions

```
def predict_caption(photo):
    in_text = "startseq"

    for i in range(max_len):
        sequence = [word_to_idx[w] for w in in_text.split() if w in word_to_idx]
        sequence = pad_sequences([sequence], maxlen=max_len, padding='post')

        ypred =  model.predict([photo,sequence])
        ypred = ypred.argmax()
        word = idx_to_word[ypred]
        in_text+= ' ' +word

        if word =='endseq':
            break


    final_caption =  in_text.split()
    final_caption = final_caption[1:-1]
    final_caption = ' '.join(final_caption)

    return final_caption
```

Figure 10: Testing the Model

V. RESULTS

A. *Perfectly Generated Captions*

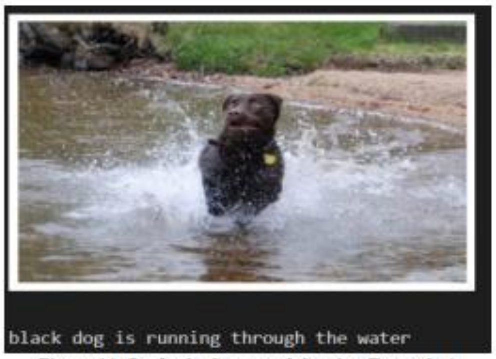
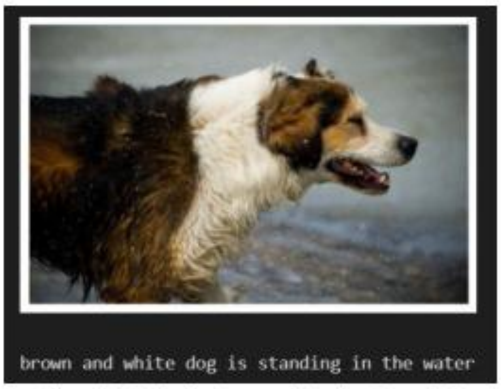


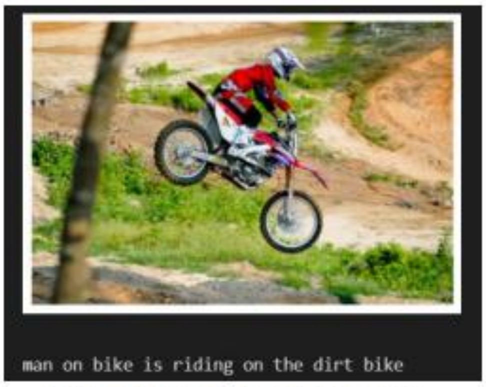
Figure 11: Perfectly Generated Caption Example 1



```
brown and white dog is standing in the water
```

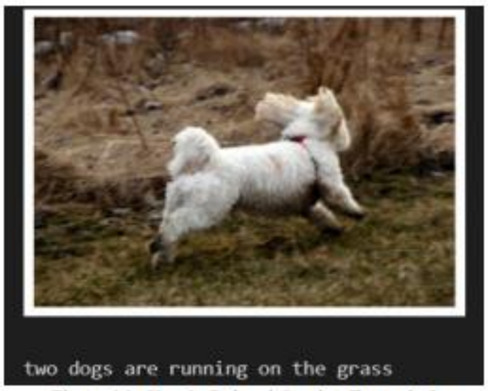
Figure 12: Perfectly Generated Caption Example 2

B. Closely Related Captions



```
man on bike is riding on the dirt bike
```

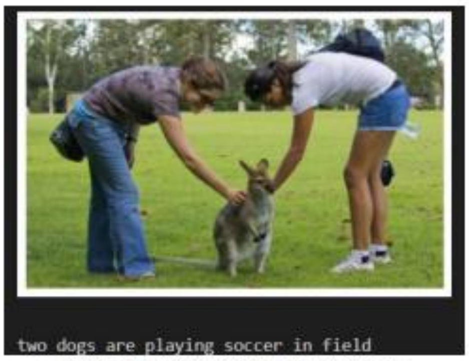
Figure 13: Closely Related Caption Example 1

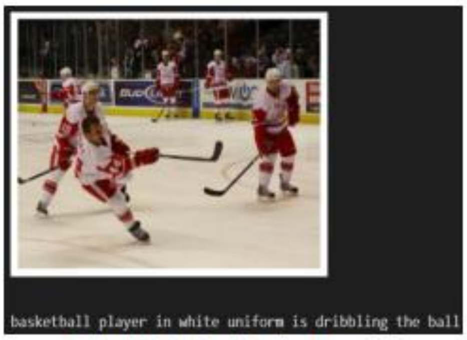


```
two dogs are running on the grass
```

Figure 14: Closely Related Caption Example 2

C. Unrelated Captions


```
two dogs are playing soccer in field
```
Figure 15: Unrelated Captions Example 1


```
basketball player in white uniform is dribbling the ball
```
Figure 16: Unrelated Caption Example 2

VI. FUTURE SCOPE

The model is currently trained with flickr_8k dataset. In future the CNN-LSTM model can be trained against the dataset containing much larger volume of images like 1000000 images which will improve the overall accuracy of the model. Instead of LSTM we can use another RNN algorithm known as Long Term Recurrent Convolutional Neural Network. LRCN combines a deep hierarchical visual feature extractor (such as NN) with a model that can learn to recognize temporal dynamics for task involving sequential data, linguistics etc.

VII. CONCLUSION

We have proposed a system that will generate logical captions for an image. The model can also be tested for its evaluation against BLEU and METEOR metric system. The system is so designed that it will be able to mimic human like behaviour for describing an image. In addition to that the model that we have proposed, uses very few hard coded assumptions.
We hope that our research will encourage and help students in their future researches and areas of work.

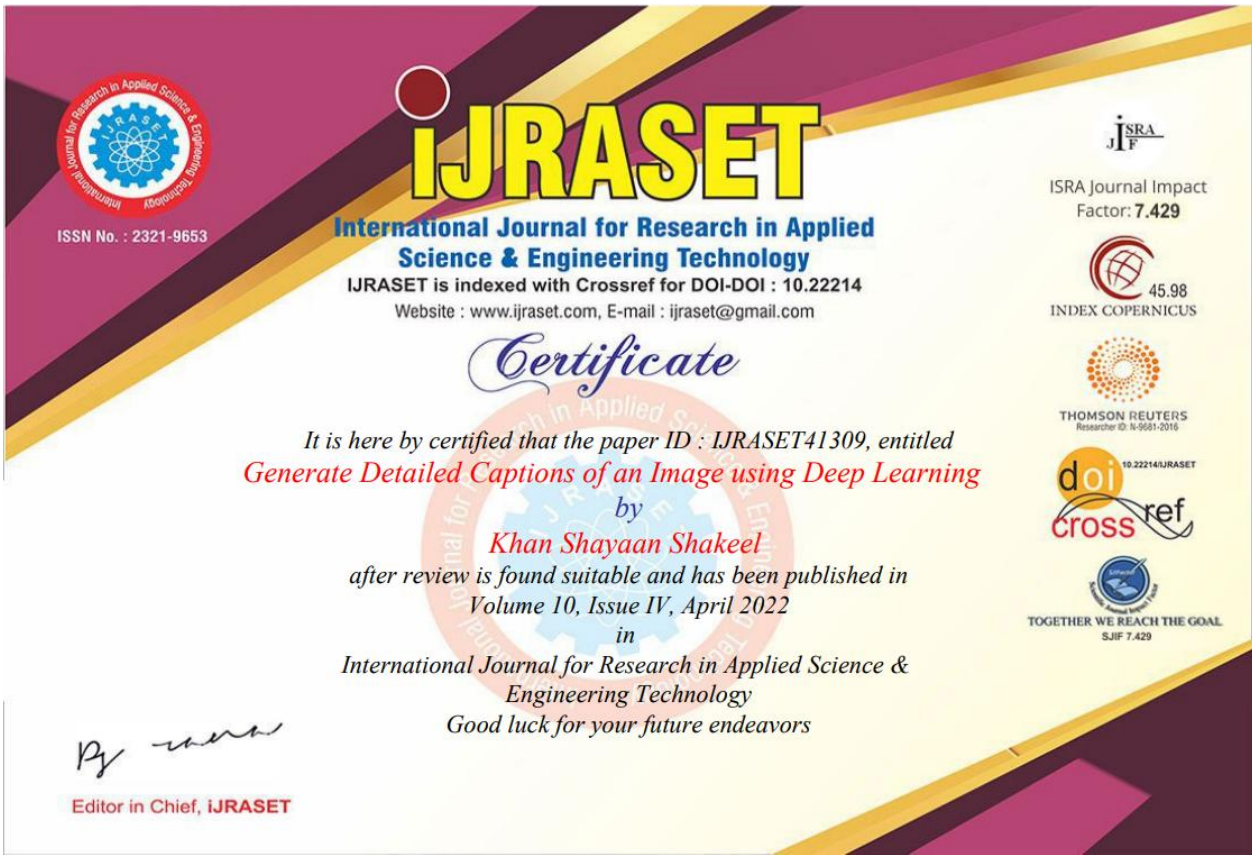
REFERENCES

[1] V. Julakanti, "Image Caption Generator using CNN-LSTM Deep Neural Network", International Journal for Research in Applied Science and Engineering Technology, vol. 9, no., pp. 2968-2974, 2021.
[2] S.-H. Han and H.-J. Choi, "Domain-Specific Image Caption Generator with Semantic Ontology", IEEE International Conference on Big Data and Smart Computing (BigComp), 2020.
[3] M. Wang, L. Song, X. Yang, and C. Luo, " A parallel-fusion RNN-LSTM architecture for image caption generation ", IEEE International Conference on Image Processing (ICIP), 2016.
[4] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[5] S. Shukla, S. Dubey, A. Pandey, V. Mishra, M. Awasthi and V. Bhardwaj, "Image Caption Generator Using Neural Networks", International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp. 01-07, 2021.
[6] M. Panicker, V. Upadhayay, G. Sethi and V. Mathur, "Image Caption Generator", International Journal of Innovative Technology and Exploring Engineering, vol. 10, no. 3, pp. 87-92, 2021.
[7] J. Karan Garg and Kavita Saxena, "Image to Caption Generator", International Journal for Modern Trends in Science and Technology, vol. 6, no. 12, pp. 181-185, 2020.
[8] F. Fang, H. Wang, and P. Tang, "Image Captioning with Word Level Attention", 25th IEEE International Conference on Image Processing (ICIP), 2018.
[9] Y. Huang, J. Chen, W. Ouyang, W. Wan and Y. Xue, "Image Captioning With End-to-End Attribute Detection and Subsequent Attributes Prediction", IEEE Transactions on Image Processing, vol. 29, pp. 4013-4026, 2020.
[10] P. Mathur, A. Gill, A. Yadav, A. Mishra, and N. K. Bansode, " Camera2Caption: A real-time image caption generator ", International Conference on Computational Intelligence in Data Science (ICCIDS), 2017.
[11] S. Amirian, K. Rasheed, T. Taha and H. Arabnia, "Automatic Image and Video Caption Generation With Deep Learning: A Concise Review and Algorithmic Overlap", IEEE Access, vol. 8, pp. 218386-218400, 2020.
[12] Y. Zhou, Y. Sun, and V. Honavar, " Improving Image Captioning by Leveraging Knowledge Graphs", IEEE Winter Conference on Applications of Computer Vision (WACV), 2019.
[13] Y. Zhenyu and Z. Jiao, "Research on Image Caption Method Based on Mixed Image Features", IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2019.
[14] M. Tanti, A. Gatt, and K. Camilleri, "What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?", Proceedings of the 10th International Conference on Natural Language Generation, 2017.
[15] Simao Herdade, Armin Kappeler, Kofi Boakye, Joao Soares, "Image Captioning: Transforming Objects into Words.", Proceedings of the 10th International Conference on Natural Language Generation, 2017.
[16] A. Deshpande, J. Aneja, L. Wang, A.G. Schwing, D. Forsyth, "Fast, diverse and accurate image captioning guided by part-of-speech", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[17] R. Subash, "Automatic Image Captioning Using Convolution Neural Networks and LSTM", Journal of Physics Conference, 2019.
[18] B.Krishnakumar,K.Kousalya, S.Gokul,R.Karthikeyan, D.Kaviyarasu, "Image Caption Generation Using Deep Learning", International Journal of Advanced Science and Technology, 2020.

Chapter 15

Paper Presentation Certificate



ISSN No. : 2321-9653

iJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISRA Journal Impact Factor: 7.429

45.98 INDEX COPERNICUS

THOMSON REUTERS Researcher ID: N-9681-2016

10.22214/IJRASET doi crossref

TOGETHER WE REACH THE GOAL SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET41309, entitled

Generate Detailed Captions of an Image using Deep Learning

by

Khan Shayaan Shakeel

after review is found suitable and has been published in

Volume 10, Issue IV, April 2022

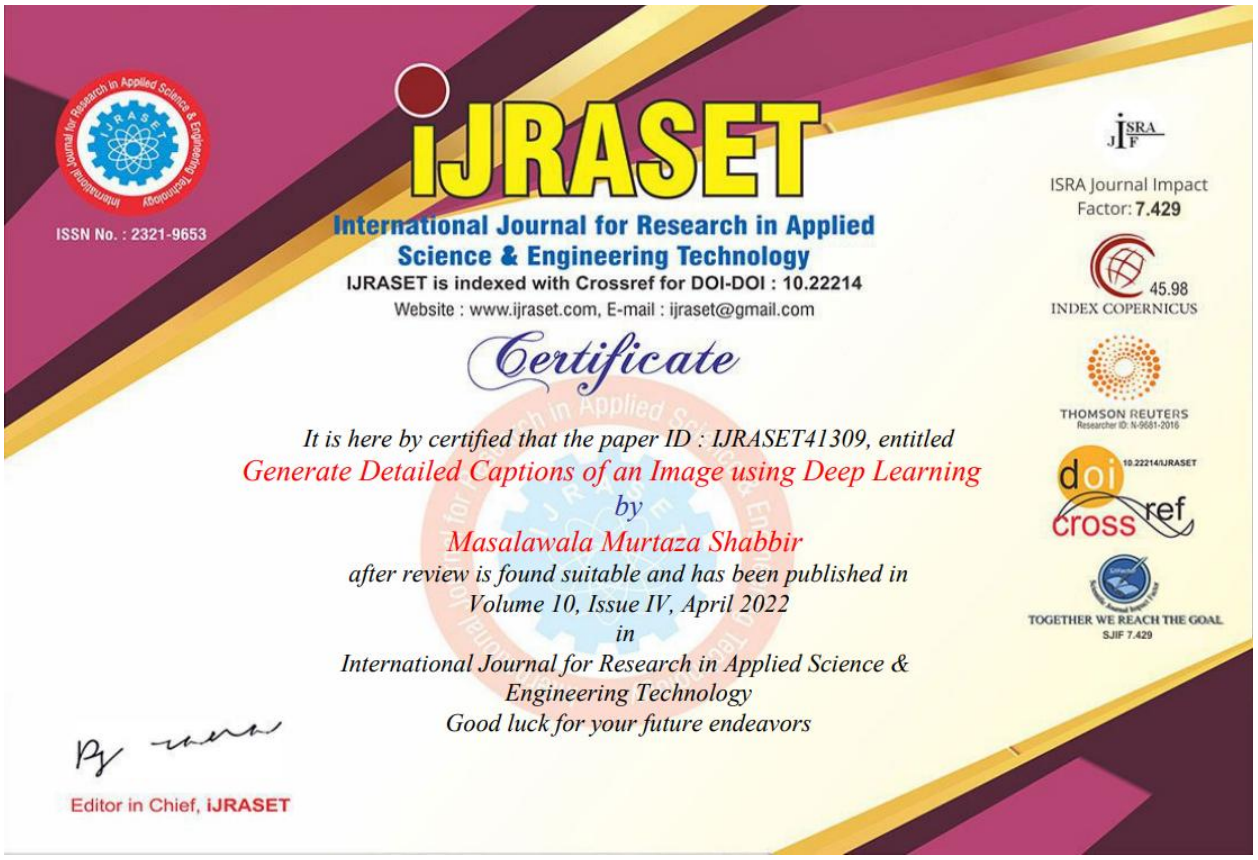
in

International Journal for Research in Applied Science & Engineering Technology

Good luck for your future endeavors

[illegible]

Editor in Chief, iJRASET



ISSN No. : 2321-9653

iJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISRA Journal Impact Factor: 7.429

45.98 INDEX COPERNICUS

THOMSON REUTERS Researcher ID: N-9681-2016

10.22214/IJRASET

TOGETHER WE REACH THE GOAL SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET41309, entitled

Generate Detailed Captions of an Image using Deep Learning

by

Masalawala Murtaza Shabbir

after review is found suitable and has been published in

Volume 10, Issue IV, April 2022

in

International Journal for Research in Applied Science & Engineering Technology

Good luck for your future endeavors

[illegible]

Editor in Chief, iJRASET

ISSN No. : 2321-9653

iJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISRA Journal Impact Factor: 7.429

45.98 INDEX COPERNICUS

THOMSON REUTERS Researcher ID: N-9681-2016

10.22214/IJRASET

TOGETHER WE REACH THE GOAL SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET41309, entitled

Generate Detailed Captions of an Image using Deep Learning

by

Qazi Faizan Ahmed

after review is found suitable and has been published in

Volume 10, Issue IV, April 2022

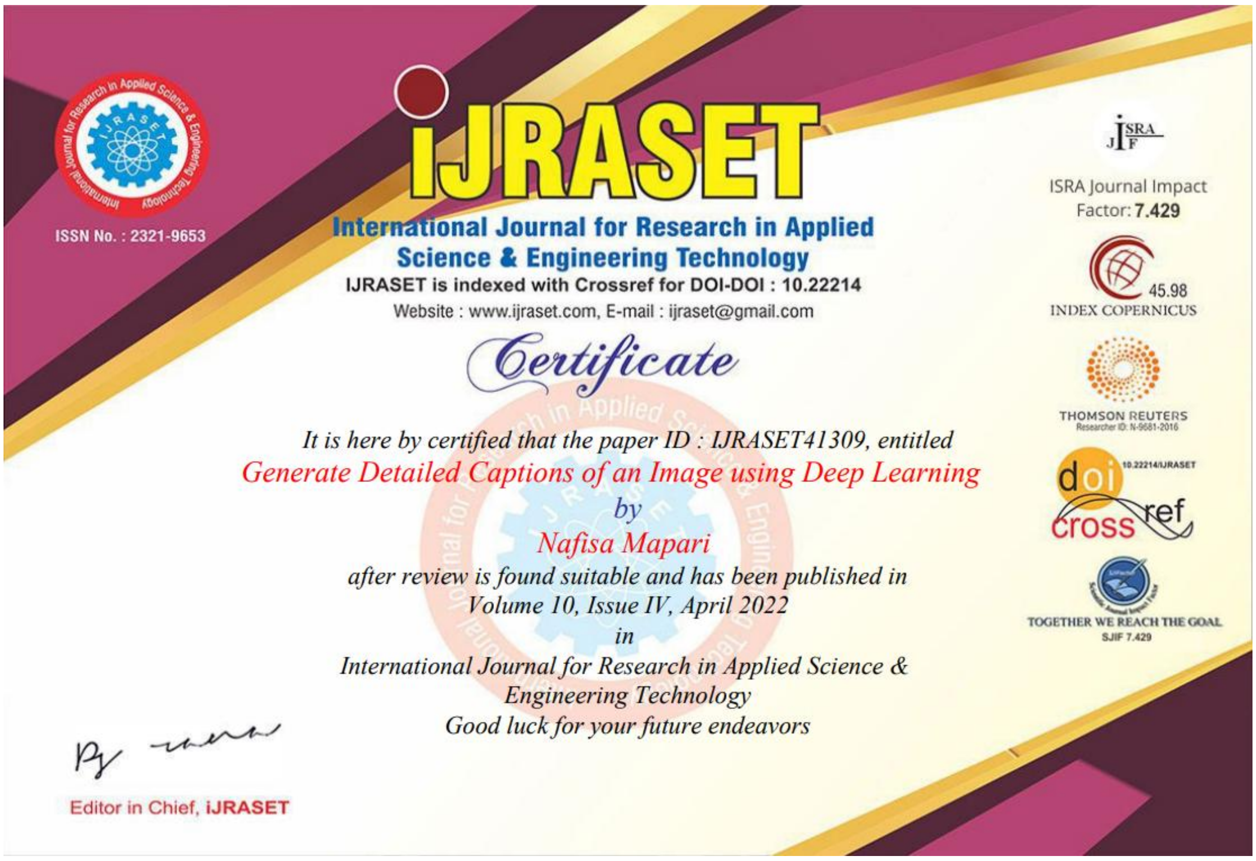
in

International Journal for Research in Applied Science & Engineering Technology

Good luck for your future endeavors

[illegible]

Editor in Chief, iJRASET



ISSN No. : 2321-9653

iJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISRA Journal Impact Factor: 7.429

45.98 INDEX COPERNICUS

THOMSON REUTERS Researcher ID: N-9681-2016

10.22214/IJRASET

TOGETHER WE REACH THE GOAL SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET41309, entitled

Generate Detailed Captions of an Image using Deep Learning

by

Nafisa Mapari

after review is found suitable and has been published in

Volume 10, Issue IV, April 2022

in

International Journal for Research in Applied Science & Engineering Technology

Good luck for your future endeavors

[illegible]

Editor in Chief, IJRASET