***DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIOUR***

***A PROJECT REPORT***

***Submitted in partial fulfillment of the requirements for the award of the Degree of***

**BACHELOR OF TECHNOLOGY**

in

**INFORMATION TECHNOLOGY**

by

**R. PAVAN KUMAR T. CHARAN**

(18BQ1A12D2) (18BQ1A12G0)

**S. SRINIVASA RAO**

(18BQ1A12D9)

**Under the guidance of**

**Ms. SK. MULLA ALMAS, M. Tech**, Assistant Professor.

Department of Information Technology



**Vasireddy Venkatadri Institute of Technology**

Jawaharlal Nehru Technological University, Kakinada, AP, India

***June-2022***

APPENDIX 2

(A typical specimen of Bonafide Certificate)

**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY: : NAMBUR**

**BONAFIDE CERTIFICATE**

<Font Style Times New Roman – size -12>

This is to certify that this project report **“DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIOUR*”*** is the bonafide work of “**T. ALEKHYA (17BQ1A12G2), T. BHAVANA (17BQ1A12G7), V. SAI SRUJANA (17BQ1A12E6),**

**P.S.L. MANASA (17BQ1A12D2)”**, who carried out the project under the supervision of **Dr. Kalavathi A** during the academic year 2020-21 towards partial fulfillment of the Degree of Bachelor of Technology in Information Technology from Jawaharlal Nehru Technological University, Kakinada. The results embodied in this report have not been submitted to any other University for the award of any degree.

Signature of the Head of the Department Signature of the Supervisor

**<<Name>> <<Name>>**

**HEAD OF THE DEPARTMENT GUIDE**

Department of <<Designation>>, <<Department>>

<<Address of the Dept & College >>

<<Date>>

External Viva voce conducted on

**Internal Examiner External Examiner**

**EXTERNAL EXAMINER**

**DECLARATION**

We here by declare that the project report entitled **“IMAGE CAPTION GENERATION USING DEEP LEARNING”** submitted to the JNTUK, Kakinada, is a record of an original work done by us under the guidance of **Mr. G. MAHESH,** Assistant Professor and this project report is submitted in the fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology. The results embodied in this project report are not submitted to any other University or Institute for the award of any Degree or Diploma.

**PLACE:**

**DATE:**

**PROJECT ASSOCIATES**  **SIGNATURES**

**S. PRIYATHAM** (18FE1A1249) **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**R. SHARATH KUMAR** (18FE1A1237) **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**P. DEEPAK** (18FE1A1233) **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**S. ANNIE ARCHANA** (18FE1A1243) **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **ACKNOWLEDGEMENT**

The satisfaction that accompanies with the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are glad to express our deep sense of gratitude to **Mr. G. MAHESH**, Assistant professor, Information Technology for guiding through this project and for encouraging right from the beginning of the project. Every interaction with him was an inspiration. At every step she was there to help us to choose right path.

We also express our deep gratitude to the beloved **Mr. R. VEERABABU,** Assistant Professor and Head of the Department of Information Technology for his encouragement and kind support in carrying out our work.

We also express our deep gratitude to the beloved chairman **Dr. L. RATHAIAH,** and to our Principal **Dr. K. PHANEENDRA KUMAR** and for their encouragement and kind support in carrying out our work.

We thank our parents and others who have rendered help to us directly or indirectly in the completion of project work.

**PROJECT MEMBERS**

**S. PRIYATHAM** (18FE1A1249)

**R. SHARATH KUMAR** (18FE1A1237)

**P. DEEPAK** (18FE1A1233)

**S. ANNIE ARCHANA** (18FE1A1243)

# ABSTRACT

In the past few years, the problem of generating descriptive sentences automatically for images has garnered a rising interest in natural language processing and computer vision research. Image captioning is a fundamental task that requires a semantic understanding of images and the ability to generate description sentences with the proper and correct structure. In this study, the authors propose a hybrid system employing the use of a multilayer Convolutional Neural Network (CNN) to generate vocabulary describing the images and a Long Short-Term Memory (LSTM) to accurately structure meaningful sentences using the generated keywords. The convolutional neural network compares the target image to a large dataset of training images, then generates an accurate description using the trained captions. We showcase the efficiency of our proposed model using the Flickr8K dataset and show that their model gives superior results compared with the state-of-the-art models utilizing the Bleu metric. The Bleu metric is an algorithm for evaluating the performance of a machine translation system by grading the quality of text translated from one natural language to another. The performance of the proposed model is evaluated using standard evaluation matrices, which outperform previous benchmark models.

**CONTENTS**

**PAGE NO.**

**CERTIFICATE ii**

**DECLARATION iii**

**ACKNOWLEDGEMENT iv**

**ABSTRACT v**

**CONTENTS vi-vii**

**LIST OF FIGURES viii**

**CHAPTER 1 INTRODUCTION 1-3**

1.1 OBJECTIVE 1

1.2 SCOPE 1

1.3 USECASES 2

1.4 HOW IMAGE CAPTIONING WORKS? 2

**CHAPTER 2 LITERATURE REVIEW 4-5**

**CHAPTER 3 SYSTEM ANALYSIS 6-9**

3.1 EXISTING SYSTEM 6

3.1.1 DISADVANTAGES OF EXISTING SYSTEM 6

3.2 PROPOSED SYSTEM 6

3.2.1 ADVANTAGES & APPLICATIONS OF PROPOSED SYSTEM 7

3.3 FEASIBILITY STUDY 7

3.3.1 ECONOMICAL FEASIBILITY 8

3.3.2 TECHNICAL FEASIBILITY 8

3.3.3 SOCIAL FEASIBILITY 8

3.3.4 OPERATIONAL FEASIBILITY 9

**CHAPTER 4 SYSTEM REQUIREMENTS SPECIFICATION 10-13**

4.1 HARDWARE REQUIREMENTS 10

4.2 SOFTWARE REQUIREMENTS 10

4.3 SYSTEM REQUIREMENT SPECIFICATION 10

4.4 PURPOSE 11

4.5 FUNCTIONAL REQUIREMENTS 11

4.6 NON-FUNCTIONAL REQUIREMENTS 12

**CHAPTER 5 METHODOLOGY 14-25**

5.1 DATA COLLECTION 14-15

5.2 ALGORITHM 15-21

5.3 LIBRARIES DESCRIPTION 22-25

**CHAPTER 6 SYSTEM DESIGN 26-33**

6.1 INPUT AND OUTPUT DESIGN 26-27

6.2 UML DIAGRAMS 27-33

**CHAPTER 7 IMPLEMENTATION 34-39**

7.1 INTRODUCTION 34-35

7.2 SAMPLE CODE 36-39

**CHAPTER 8 TESTING 40-47**

8.1 TESTING INTRODUCTION 40

8.2 TESTING PROCESS 40

8.3 LEVELS OF TESTING 41-47

**CHAPTER 9 RESULTS AND DISCUSSION 48-55**

**CONCLUSION AND FUTURE SCOPE 56**

**REFERENCES 57**

# LIST OF FIGURES

**FIGURE NO** **FIGURE NAME** **PAGE NO**

Figure 1.4.2 Working Representation of Model 3

Figure 5.2.1 A Functional CNN- RNN Model 16

Figure 5.2.2 In-depth working of RNN Layers 17

Figure 5.2.3 Schematic of Merge Model for IC 18

Figure 5.2.4 Feature extraction and Classification 18

Figure 5.2.5 Conversion of captions into a list of tokenized words 19

Figure 5.2.6 Image Captioning Model by Team Oodles 20

Figure 5.2.7 Plot of Caption Generation Deep Learning Model 20

Figure 5.2.8 In-depth working of Word Embedding 21

Figure 6.2.1 Use case diagram 29

Figure 6.2.2 Activity diagram 30

Figure 6.2.3 Sequence diagram 31

Figure 6.2.4 Class Diagram 32

Figure 6.2.5 Deployment diagram 32

Figure 6.2.6 Data Flow diagram 33

Figure 8.2 Testing process 41

Figure 8.3 Levels of Testing 42

Figure 8.3.1 Unit Testing 42

Figure 8.3.2 Top-down Integration 44

Figure 9.2.1 Interpretation of BLEU scores 49

Figure 9.3.1 Accuracy of the Model 50

Figure 9.3.2 Train Model Accuracy 51

Figure 9.3.3 Train Model Loss 51

Figure 9.3.4 BLEU Score of the model 51

Figure 9.5.1 Testcase- 1 53

Figure 9.5.2 Testcase- 2 53

**CHAPTER- 1**

**INTRODUCTION**

**1.1 OBJECTIVE**

Machine Learning is a field of technology development with immense abilities and applications in automating tasks, where neither human intervention is needed nor explicit programming.

The power of ML is such great that we can see its applications trending almost everywhere in our day-to-day lives. ML has solved many problems that existed earlier and have made businesses in the world progress to a great extent. Automatically generating captions to an image shows the understanding of the image by computers, which is a fundamental task of intelligence. Caption Generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph.

Image captioning can be regarded as an end-to-end Sequence to Sequence problem, as it converts images, which are regarded as a sequence of pixels to a sequence of words. For this purpose, we need to process both the language or statements and the images. For the Language part, we use recurrent Neural Networks, and for the Image part, we use Convolutional Neural Networks respectively.

**1.2 SCOPE**

Image Captioning is the process of generating a textual description for given images. It has been a very important and fundamental task in the Deep Learning domain. Image captioning has a huge number of applications. NVIDIA is using image captioning technologies to create an application to help people who have low or no eyesight.

In our project, we do image-to-sentence generation. This application bridges vision and natural language. If we can do well in this task, we can then utilize natural language processing technologies to understand the world in images. In addition, we introduced an attention mechanism, which can recognize what a word refers to in the image, and thus summarize the relationship between objects in the image. This will be a powerful tool to utilize the massive unformatted image data, which dominates the whole data in the world.

**1.3 USECASES**

* Some detailed use cases would be like a visually impaired person taking a picture from his phone and then the caption generator will turn the caption to speech for him to understand.
* Advertising industry trying the generate captions automatically without the need to make them separately during production and sales.
* Doctors can use this technology to find tumors or some defects in the images or used by people for understanding geospatial images where they can find out more details about the terrain

**1.4 HOW IMAGE CAPTIONING WORKS**

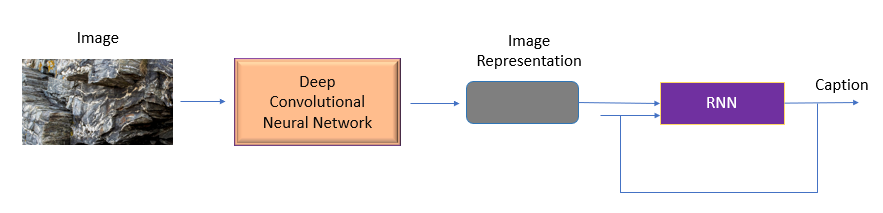


**Figure 1.4.1. A puppy on a blue towel**

If we are told to describe it, maybe we will describe it as: “A puppy on a blue towel” or “A brown dog playing with a green ball”. So, how are we doing this? While forming the description, we are seeing the image, but at the same time, we are looking to create a meaningful sequence of words. The first part is handled by CNNs and the second is handled by RNNs.

The task of image captioning can be divided into two modules logically–one is an **image-based model**– which extracts the features and nuances out of our image, and the other is a **language-based model**– which translates the features and objects given by our image-based model to a natural sentence.

For our image-based model (via encoder), we usually rely on a Convolutional Neural Network model. And for our language-based model (via decoder), we rely on a Recurrent Neural Network. The image below summarizes the approach given above.



**Figure-1.4.2. Working Representation of Model**

Usually, a pre-trained CNN extracts the features from our input image. The feature vector is linearly transformed to have the same dimension as the input dimension of the RNN/LSTM network. This network is trained as a language model on our feature vector. Convolutional Neural Network (CNN) to generate vocabulary describing the images and a Long Short-Term Memory (LSTM) to accurately structure meaningful sentences using the generated keyword.

**CHAPTER- 2**

**LITERATURE SURVEY**

**Andrej Karpathy and Li Fei-Fei [1],** presented a model that generates natural language descriptions of images and their regions. Described a Multimodal Recurrent Neural Network architecture that uses the inferred alignments to learn to generate novel descriptions of image regions as an approach leverage dataset of images and their sentence descriptions to learn about the inter-modal correspondences between language and visual data. The model describes Learning to align visual and language data Multimodal Recurrent Neural Network for generating descriptions. The model can only generate a description of one input array of pixels at a fixed resolution. RNN receives the image information only through additive bias interactions, which are known to be less expressive than more complicated multiplicative interactions.

(**Andrej Karpathy and Li Fei-Fei**, 2016)

**Jyothi Aneja, Aditya Deshpande, and Alexander Schwing [2],** proposed their work on Convolutional Image Captioning. This is an application for various scenarios, recommendations in editing applications, usage in Virtual assistants, image indexing, and support of the disabled. It is also a basic ingredient for more complex operations such as storytelling and visual summarization. Here we use Inference to disallow convolution operations from using the information of future word tokens. The training process is inherently sequential for a particular image-caption pair. This results from unrolling the recurrent relation in time.

(**Jyothi Aneja, Aditya Deshpande, and Alexander Schwing**, 2017)

**Chetan Amritkar and Vaishali Jabade [3]**, proposes a model capable of generating novel descriptions from images. Presented a generative model based on a deep recurrent architecture that combines recent advances in computer vision and machine translation and that can be used to generate natural sentences describing an image. Designed as a single model which takes an image as an input and is trained for producing a sequence of words where each word belongs to the dictionary that describes the image suitably. This task of automatically generating captions and describing the image is significantly harder than image classification and object recognition.

(**Chetan Amritkar and Vaishali Jabade**, 2018)

**Zakir Hossain, Ferdous Sohel [4],** proposes the model generating textual descriptions of images. Several techniques based on deep learning have been proposed on this topic. These models require many training data to perform at their full potential. Using Generative Adversarial Network (GAN) based text to image generator to generate synthetic images. We use an attention-based image captioning method trained on both real and synthetic images to generate the captions. It demonstrates the effectiveness of image captioning for synthetic images. It further improves the quality of the generated captions for real images, understandably, because we use additional images for training.

(**Zakir Hossain and Ferdous Sohel,** 2021)

**CHAPTER- 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

The existing system generates a description of one input array of pixels at a fixed resolution. Retrieval-based methods have large limitations in their capability to describe images. Under certain conditions, generated descriptions may even be irrelevant to image contents. The existing model will depend on the data, so it cannot predict the words that are out of the scope of its vocabulary. Although the yielded outputs are usually grammatically correct and fluent, constraining image descriptions to sentences that have already existed cannot adapt to new combinations of objects or novel scenes. Under certain conditions, generated descriptions may even be irrelevant to image contents.

**3.1.1 DISADVANTAGES OF EXISTING SYSTEM**

1. The disadvantages of retrieval-based image captioning methods are obvious. Such methods transfer well-formed human-written sentences or phrases for generating descriptions for query images.
2. Although the yielded outputs are usually grammatically correct and fluent, constraining image descriptions to sentences that have already existed cannot adapt to new combinations of objects or novel scenes.
3. Under certain conditions, generated descriptions may even be irrelevant to image contents. Retrieval-based methods have large limitations in their capability to describe images.
4. Our model will depend on the data, so, it cannot predict the words that are out of the scope of its vocabulary.

**3.2 PROPOSED SYSTEM**

The proposed system overcomes the drawbacks of the existing system by using modern technology. It uses Deep neural networks to avoid all the mismatches created by the existing system. Here we use BLEU Metric for automatically evaluating machine-translated text. Using CNN and RNN models, we get more confidence in the image than in the existing system. Implemented the final model with the real-time application using GUI.

**Development of DL model**

It has three steps: loading, defining, and fitting the model.

**Implementation of the system**

1. **Object Detection:**Objects are detected from the image with the help of CNN Encoder, which then is used as input for the RNN-LSTM algorithm.
2. **Tokenization:**A tokenizer is mapped from words to unique integer values.
3. **Text Prediction:**Using the LSTM embedding layer, each predicted word is employed to get subsequent words.

**3.2.2 ADVANTAGES & APPLICATIONS OF PROPOSED SYSTEM**

1. **Recommendations for Editing Applications**

The image captioning model automates and accelerates the close captioning process for digital content production, editing, delivery, and archival. Well-trained models replace manual efforts for generating quality captions for images as well as videos.

1. **Assistance for Visually Impaired**

The advent of machine learning solutions, like image captioning, is a boon for visually impaired people who are unable to comprehend visuals. With an AI-powered image caption generator, image descriptions can be read out to the visually impaired, enabling them to get a better sense of their surroundings.

1. **Media and Publishing Houses**

The media and public relations industry circulates tens of thousands of visual data across borders in the form of newsletters, emails, etc. The image captioning model accelerates subtitle creation and enables executives to focus on more important tasks.

4. **Social-Media**

For social media, artificial intelligence is moving from discussion rooms to underlying mechanisms for identifying and describing terabytes of media files. It enables community administrators to monitor interactions and analysts formulate business strategies.

**3.3** **FEASIBILITY STUDY**

A feasibility study is a preliminary study which investigates the information of prospective users and determines the resources requirements, costs, benefits and feasibility of proposed system. A feasibility study takes into account various constraints within which the system should be implemented and operated. In this stage, the resource needed for the implementation such as computing equipment, manpower and costs are estimated. The estimated are compared with available resources and a cost benefit analysis of the system is made. The feasibility analysis activity involves the analysis of the problem and collection of all relevant information relating to the project. The main objectives of the feasibility study are to determine whether the project would be feasible in terms of economic feasibility, technical feasibility and operational feasibility and schedule feasibility or not. It is to make sure that the input data which are required for the project are available. Thus, we evaluated the feasibility of the system.

* + 1. **ECONOMIC FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be useful point of reference to compare actual costs as the project progresses.

**3.3.2 TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipment have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and data security?
  + 1. **SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

**3.3.4 OPERATIONAL FEASIBILITY**

The project is operationally feasible as the user having basic knowledge about computer and

Internet. It is based on operational performance of the model depending on the training dataset.

* Is there sufficient support for the management from the users?
* Will the system be used and work properly if it is being developed and implemented?
* Will there be any resistance from the user that will undermine the possible application benefits?

**CHAPTER- 4**

**SYSTEM REQUIREMENT SPECIFICATION**

**4.1 HARDWARE REQUIREMENTS**

System: Intel (i5) and Above

Hard Disk: 256GB and Above

Monitor: 14’ Colour Monitor.

Ram: 6GB and Above

**4.2 SOFTWARE REQUIREMENTS**

Operating system: Windows 8, 10(64bit), 11, MacOS

Software: Python

Tools: Anaconda (Jupyter Notebook IDE)

**4.3 SYSTEM REQUIREMENT SPECIFICATION**

A Software Requirements Specification (SRS) – a requirements specification for a software system – is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development life cycle domain, typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers.

Projects are subject to three sorts of requirements:

* Business requirements describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product which could be one of several ways to accomplish a set of business requirements.
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify specific methodologies that must be followed, and constraints that the organization must obey.

Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, a maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement); a requirement that the product be maintainable (a Product requirement) often is addressed by imposing requirements to follow particular development styles.

**4.4 PURPOSE**

A systems engineering, a requirement can be a description of what a system must do, referred to as a Functional Requirement. This type of requirement specifies something that the delivered system must be able to do. Another type of requirement specifies something about the system itself, and how well it performs its functions. Such requirements are often called Non-functional requirements, or 'performance requirements' or 'quality of service requirements.' Examples of such requirements include usability, availability, reliability, supportability, testability and maintainability. A collection of requirements defines the characteristics or features of the desired system. A 'good' list of requirements as far as possible avoids saying how the system should implement the requirements, leaving such decisions to the system designer.

Specifying how the system should be implemented is called "implementation bias" or "solution engineering". However, implementation constraints on the solution may validly be expressed by the future owner, for example for required interfaces to external systems; for interoperability with other systems; and for commonality (e.g., of user interfaces) with other owned products. In software engineering, the same meanings of requirements apply, except that the focus of interest is the software itself.

**4.5 FUNCTIONAL REQUIREMENTS**

Functional requirement are the functions or features that must be included in any system to satisfy the business needs and be acceptable to the users. User or the customer has to satisfy for features

and also needs to be accepted in the market and should rely customers to buy the product. These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected.

They are basically the requirements stated by the user which one can see directly in the final product.

**4.6 NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements are a description of features, characteristics and attribute of the system as well as any constraints that may limit the boundaries of the proposed system. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioural requirements. The non-functional requirements are essentially based on the performance, information, economy, control and security efficiency and services.

**Efficiency**: Specifies how well the software utilizes scarce resources: CPU cycles, disk space, memory, bandwidth etc. All of the above-mentioned resources can be effectively used by performing most of the validations through the link available

**Flexibility**: If the organization intends to increase or extend the functionality of the software after it is deployed, that should be planned from the beginning; it influences choices made during the design, development, testing and deployment of the system. New modules can be easily integrated to our system without disturbing the existing modules or modifying the logical database schema of the existing applications.

**Portability**: Portability specifies the ease with which the software can be installed on all necessary platforms, and the platforms on which it is expected to run. By using appropriate server versions released for different platforms our project can be easily operated on any operating system, hence can be said highly portable.

**Scalability**: Software that is scalable has the ability to handle a wide variety of system configuration sizes. The non-functional requirements should specify the ways in which the system may be expected to scale up (by increasing hardware capacity, adding machines etc.). Our system can be easily expandable. Any additional requirements such as hardware or software which increase the performance of the system can be easily added. An additional server would be useful to speed up the application.

**Integrity**: Integrity requirements define the security attributes of the system, restricting access to features or data to certain users and protecting the privacy of data entered into the software. Certain features access must be disabled to normal users such as adding the details of files, searching etc.

which is the sole responsibility of the server. Access can be disabled by providing appropriate logins to the users for only access.

**Usability**: Ease-of-use requirements address the factors that constitute the capacity of the software to be understood, learned, and used by its intended users. Hyperlinks will be provided for each and every service the system provides through which navigation will be easier. A system that has high usability coefficient makes the work of the user easier.

**Performance**: This system is developing in the high-level languages and using the advanced front-end and back-end technologies it will give response to the end user. Performance of the system is essential.

Based on these the non-functional requirements are as follows:

• User friendly

• System should provide better accuracy

• To perform with efficient throughput and response time.

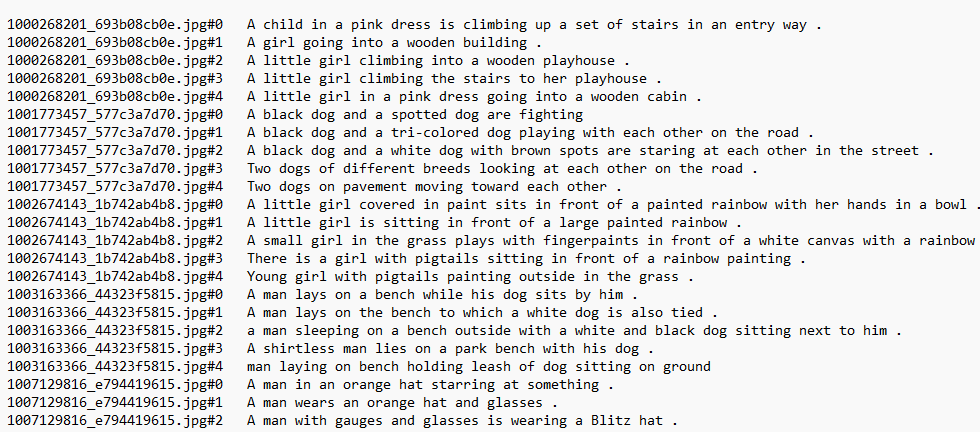
**CHAPTER- 5**

**METHODOLOGY**

**5.1 DATA COLLECTION**

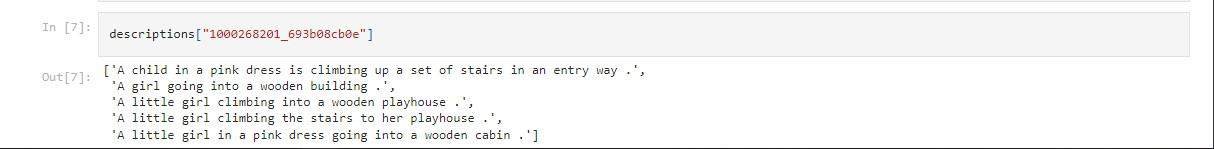
One of the files is “Flickr8k.token.txt” which contains the name of each image along with its

Five captions. (<image name>#i <caption>, where 0≤i≤4)



**Figure 5.1.1: Screenshot of file in the dataset**

We will create a dictionary named “descriptions” which contains the name of the image (without the .jpg extension) as keys and a list of the five captions for the corresponding image as values.



**Figure 5.1.2: Screenshot of my code on google colab**

**Why Flickr8k dataset…?**

1. It is small. So, the model can be trained easily on low-end laptops/desktops...

2. Data is properly labeled. For each image, 5 captions are provided.

3. The dataset is available for free.

**Let’s understand the data.**

Data pre-processing and cleaning is an important part of the whole model building process. Understanding the data helps us to build more accurate models.

After extracting zip files, you will find the below folders…

**Flickr8k\_Dataset:** This contains a total of 8092 images in JPEG format, with different shapes and sizes. Of which 6000 are used for training, 1000 for test and 1000 for development.

**Flickr8k\_text:** Contains text files describing train\_set, test\_set. Flickr8k.token.txt contains 5 captions for each image, i.e., total 40460 captions.

We have mainly two types of data.

1. Images
2. Captions (Text)

The size of the training vocabulary is 7976.

The top 10 most frequent words are:

('a', 46784), ('in', 14094), ('the', 13509), ('on', 8007), ('is', 7196),

('and', 6678), ('dog', 6160), ('with', 5763), ('man', 5383), ('of', 4974)

Since the words which occur very less do not carry much information. We are considering words with a frequency of more than 10.

**5.2 ALGORITHM**

The AI-infused image caption generator is packed with deep learning neural networks; namely, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM), wherein

1) CNNs are deployed for extracting spatial information from the images

2) RNNs are harnessed for generating sequential data of words

3) LSTM is good at remembering lengthy sequences of words

**Convolutional Neural Networks**

A Convolutional Neural network or CNN is a type of deep neural network that is efficient at extracting meaningful information from visual imagery. When it comes to us, humans, evolution has gifted us with very complex yet efficient techniques to view and detect several objects. Our brain keeps on learning continuously without our notice. There are several organs and parts of our brain involved in the process, like the eyes, receptors, and visual cortex.

In this era, with the resources and immense computational power, it would be pointless not to explore computer vision. With so many applications of computer vision services, we can take current generation technology to the next level. A great example is the upcoming Tesla Robo-taxi, which gives us a glimpse into the future.

A very popular machine learning algorithm, especially for Object Detection, is Convolutional Neural Networks (CNN).

CNN consists of four hidden layers, such as

1. **Convolutional layers**
2. **Pooling layers**
3. **Fully connected layers**
4. **Normalization layers**

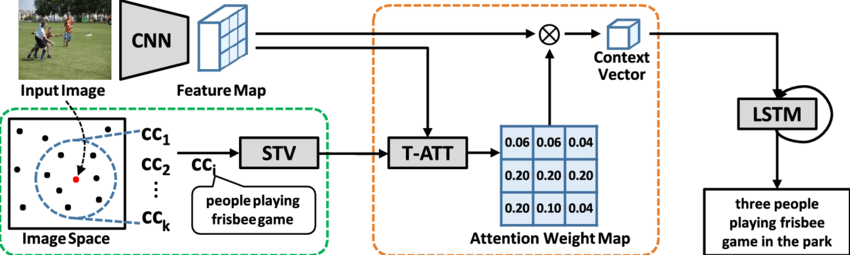
**Convolutional Layers** take two input layers - a part of the image and an equally sized filter called the kernel. The output of this layer is the dot product of both inputs.

The idea of pooling is to down-sample data.

The **Pooling Layer** takes the input (an image) and reduces its size in terms of the number of pixels. There are two ways to perform this - Max Pooling and Min Pooling. Max Pooling picks the maximum value from the selected region, whereas Min Pooling picks up the minimum value.

Under **Fully Connected Layers**, as the name suggests, all the outputs from one layer are connected to the input of another layer. These layers are useful in the classification of the data.

**Normalization Layers** are used to stabilize the neural networks. It performs normalization on the input data.

CNN performs incredibly when it comes to analyzing a single image, but it lacks one essential quality - they only consider spatial features and visual data, ignoring the temporal and time features, i.e., how a frame is related to the previous frame. This is where Recurrent Neural Networks or RNNs come into play. The term ‘recurrent’ suggests that the neural network repeats the same tasks for every sequence. RNN can also be used in Natural Language Processing.

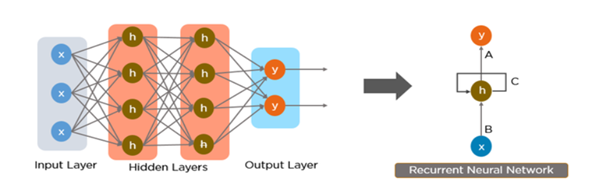
**Figure 5.2.1: A Functional CNN- RNN Model**

**Recurrent Neural Network**

A recurrent neural network (RNN) is a special type of artificial neural network adapted to work for time series data or data that involve sequences. Ordinary feed-forward neural networks are only meant for data points that are independent of each other.

RNNs are harnessed for generating sequential data of words. In neural image captioning systems, a recurrent neural network (RNN) is typically viewed as the primary generation component. This view suggests that the image features should be `injected' into the RNN. This is, in fact, the dominant view in the literature. Alternatively, the RNN can instead be viewed as only encoding the previously generated words.

RNNs have the concept of 'memory' that helps them store the states or information of previous inputs to generate the next output of the sequence. RNN is the extension of feed-forward NN with the presence of loops in hidden layers. RNN takes the input with the sequence of samples and identifies the time relationship between the samples.

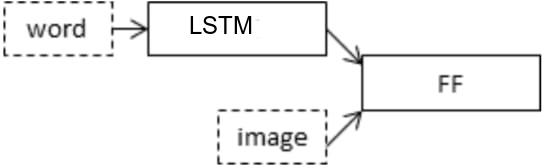


**Figure-5.2.2: In-depth working of RNN Layers**

**Long-Short-Term Memory**

We use a deep convolutional neural network to generate a vectorized representation of an image that we then feed into a Long-Short-Term Memory (LSTM) network, which then generates captions.

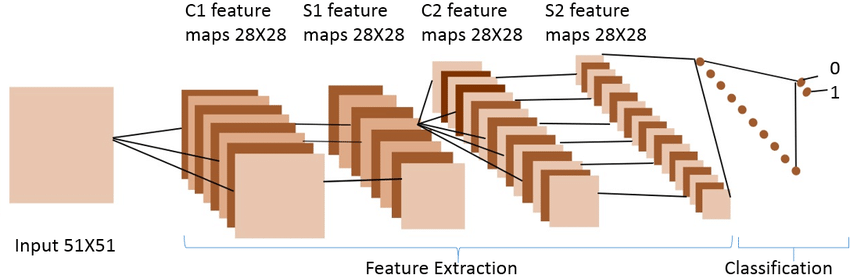
We use a deep convolutional neural network to create a semantic representation of an image, which we then decode using an LSTM network. (Right) An unrolled LSTM network for our CNN-LSTM model. All LSTMs share the same parameters. The vectorized image representation is fed into the network, followed by a special start of the sentence token. The hidden state produced is then used by the LSTM to predict/generate the caption for the given image.

**PHASES OF AUTOMATED IMAGE CAPTION GENERATOR**

**Figure 5.2.3: Schematic of the Merge Model for Image Captioning**

**Feature Extraction**

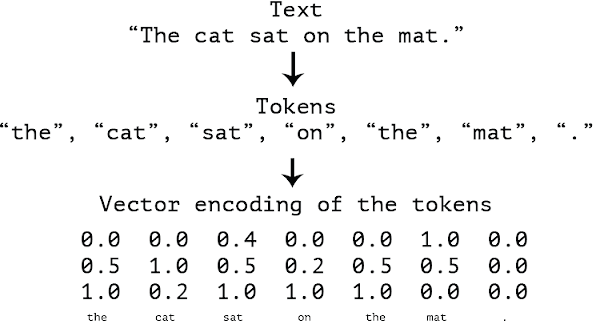
The first move is made by CNNs to extract distinct features from an image based on its spatial context. CNNs create dense feature vectors, also called embedding, that are used as an input for the following RNN algorithms. The CNN is fed with images as inputs in different formats, including png, jpg, and others. The neural networks compress large amounts of features extracted from the original image into smaller and RNN-compatible feature vectors. It is the reason why CNN is also referred to as Encoder.



**Figure 5.2.4: Feature Extraction and Classification**

**Tokenization**

The second phase brings RNN into the picture for ‘decoding’ the process vector inputs generated by the CNN module. For initiating the task for captions, the RNN model needs to be trained with a relevant dataset. It is essential to train the RNN model for predicting the next word in the

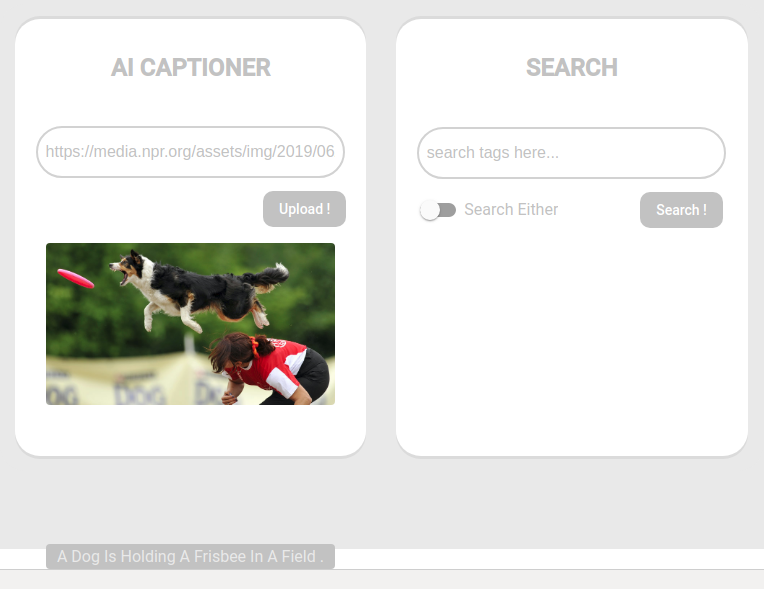
sentence. However, training the model with strings is ineffective without definite numerical alphas values. For this purpose, it is required to convert the image captions into a list of tokenized words, as shown below:

**Figure 5.2.5: Conversion of captions into a list of tokenized words.**

**Text Prediction**

Post tokenization, the last phase of the model is triggered using LSTM. This step requires an embedding layer for transforming each word into the desired vector and eventually pushed for decoding. With LSTM, the RNN model must be able to remember spatial information from the input feature vector and predict the next word. Now with LSTM performing its tasks, the final output is generated by calling the (get\_prediction) function.

Recently, a company named Oodles built an image captioning model powered by deep neural networks. Here’s how it processes the images to generate near accurate outputs-

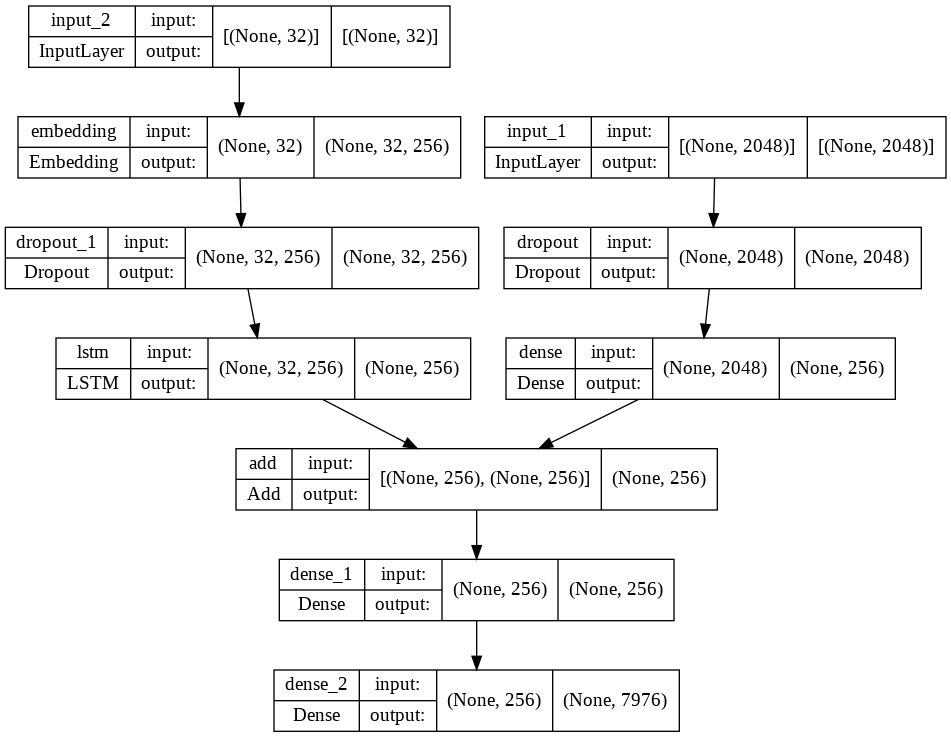


**Figure 5.2.6: Image Captioning Model by Team Oodles**

From figure 5.2.6 an experiential AI Development Company, Oodles AI decodes the underlying layers of CNN and how businesses can deploy CNN for computer vision applications.

In addition to image captioning, the model can be used to search for relevant images with input in the form of tags such as “cars”, “books”, etc.

We also create a plot to visualize the structure of the network that better helps understand the two streams of input.



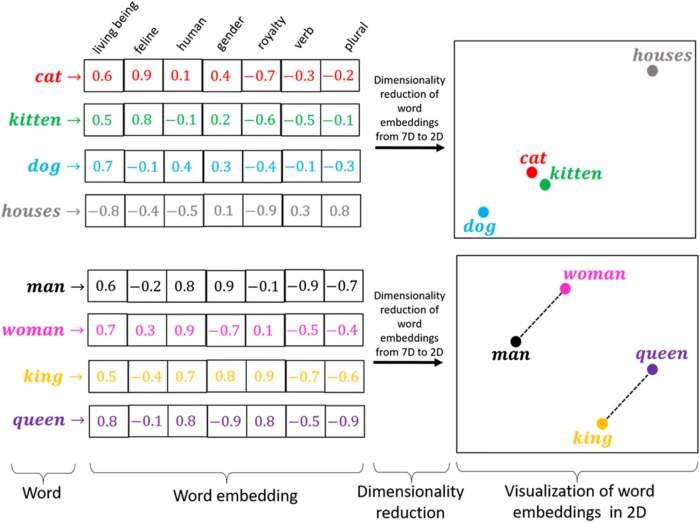
**Figure 5.2.7: Plot of the Caption Generation Deep Learning Model**

**WORD EMBEDDINGS**

A word embedding is a learned representation for text where words that have the same meaning have a similar representation.

It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.

Word embeddings are in fact a class of techniques where individual words are represented as real- valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning.



**Figure 5.2.8: In-depth Working of Word Embedding**

**5.3 LIBRARIES DESCRIPTION**

**NUMPY**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

**Numeric**, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created the NumPy package by incorporating the features of Numarray into the Numeric package. There are many contributors to this open-source project. NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib**(plotting library). This combination is widely used as a replacement for MATLAB, a popular platform for technical computing.

However, the Python alternative to MATLAB is now seen as a more modern and complete programming language. It is open-source, which is an added advantage of NumPy.

**PANDAS**

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. Pandas can clean messy data sets and make them readable and relevant. Relevant data is very important in data science.

Pandas is a software library written for a python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating tables.

* Data Frame object for data manipulation with integrated indexing.
* Tools for reading and writing data between in-memory data structures and different file formats
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of data sets.
* Label-based slicing, fancy indexing, and sub-setting of large data sets.

Pandas support the integration with many file formats or data sources out of the box (CSV, Excel, SQL, JSON, parquet,).

Importing data from each of these data sources is provided by a function with the prefix read\_\*. Similarly, the to\_\* methods are used to store data.

Change the structure of your data table in multiple ways. You can melt () your data table from wide to long/tidy form or pivot () from long to wide format. With aggregations built-in, a pivot table is created with a single command.

Pandas have great support for time series and have an extensive set of tools for working with dates, times, and time-indexed data. Data sets do not only contain numerical data. Pandas provide a wide range of functions to clean textual data and extract useful information about it.

**KERAS**

While deep neural networks are all the rage, the complexity of the major frameworks has been a barrier to their use for developers new to machine learning. There have been several proposals for improved and simplified high-level APIs for building neural network models, all of which tend to look similar from a distance but show differences upon closer examination. Keras is one of the leading high-level neural network APIs. It is written in Python and supports multiple back-end neural network computation engines.

Keras was created to be user-friendly, modular, easy to extend and to work with Python. The API was “designed for human beings, not machines,” and “follows best practices for reducing cognitive load.”

Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add as new classes and functions. Models are defined in Python code, not separate model configuration files.

**PICKLE**

The pickle (Python object serialization) module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” 1, or “flattening”; however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

To serialize an object hierarchy, you simply call the dumps () function. Similarly, to de-serialize a data stream, you call the loads () function. However, if you want more control over serialization and de-serialization, you can create a Pickler or an Unpickler object, respectively.

**Advantages of using Pickle Module:**

1. Recursive objects (objects containing references to themselves): Pickle keeps track of the objects it has already serialized, so later references to the same object won’t be serialized again. (The marshal module breaks for this.)
2. Object sharing (references to the same object in different places): This is similar to self-referencing objects; the pickle stores the object once, and ensures that all other references point to the master copy. Shared objects remain shared, which can be very important for mutable objects.
3. User-defined classes and their instances: Marshal does not support these at all, but pickle can save and restore class instances transparently. The class definition must be importable and live in the same module as when the object was stored.

**TENSORFLOW**

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on the training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many sectors.

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

**Features**

1. **AutoDifferentiation:**

AutoDifferentiation is the process of automatically calculating the gradient vector of a model concerning each of its parameters.

1. **Eager execution:**

TensorFlow includes an “eager execution” mode, which means that operations are evaluated immediately as opposed to being added to a computational graph which is executed later.

1. **Distribute:**

In both eager and graph executions, TensorFlow provides an API for distributing computation across multiple devices with various distribution strategies.

1. **Losses:**

To train and assess models, TensorFlow provides a set of loss functions (also known as cost functions).

1. **Metrics:**

To assess the performance of machine learning models, TensorFlow gives API access to commonly used metrics.

1. **TF.nn:**

TensorFlow.nn is a module for executing primitive neural network operations in models.

1. **Optimizers:**

TensorFlow offers a set of optimizers for training neural networks, including ADAM, ADAGRAD, and Stochastic Gradient Descent (SGD).

**Applications**

Medical, Social Media, Search Engine, Education, Retail, Research

**CHAPTER- 6**

**SYSTEM DESIGN**

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from problem domain to the solution domain. The design of a system is perhaps the most critical factor affecting the quality of the software, and has a major impact on the later phases, particularly testing and maintenance. The output of this phase is the design document. This document is similar to a blue print or plan for the solution, and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phase-system design and detailed design. System design, which is sometimes also called top-level design, aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of system design all the major data structures, file formats, output formats, as well as the major modules in the system and their specifications are decided. A design methodology is a systematic approach to creating a design by application of set of techniques and guidelines. Most methodologies focus on system design. The two basic principles used in any design methodology are problem partitioning and abstraction. A large system cannot be handled as a whole, and so for design it is partitioned into smaller systems. Abstraction is a concept related to problem partitioning. When partitioning is used during design, the design activity focuses on one part of the system at a time. Since the part being designed interacts with other parts of the system, a clear understanding of the interaction is essential for properly designing the part. For this, abstraction is used.

**6.1 INPUT AND OUTPUT DESIGN**

**6.1.1 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document 21 or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**6.1.2 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

**6.2 UML DIAGRAMS**

A diagram is a graphical presentation of a set of elements, most often is rendered as a graph of vertices and arcs. A UML diagram can contain nine types in it where vertices are denoted as things and arcs are denoted as relationships. The Unified Modelling Language diagram is that is, the unified modelling language is probably the most widely known and used notation for object-oriented analysis and design. The Unified Modelling Language is used to visualize, specify, construct and document the artifacts. A Modelling language is a language whose vocabulary and rules focus on the conceptual and physical representation of a system. Modelling is the designing of software applications before coding. According to uml no one diagram can capture the different elements of a system in its entirety.

We have represented them through diagrams:

* Use case diagram
* State diagram
* Activity diagram
* Sequence Diagram
* Class Diagram
* Collaboration Diagram
* Component Diagram
* Deployment diagram

**USECASE DIAGRAM**

**Use cases:**

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actors:**

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

**System boundary boxes (optional**):

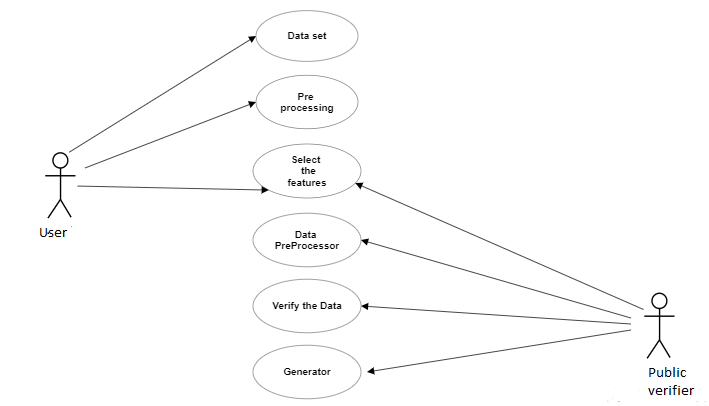
A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not. Four relationships among use cases are used often in practice.

**Include:**

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behaviour of the included use case is inserted into the behaviour of the including use case. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviours from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»". There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behaviour of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.

**Extend:**

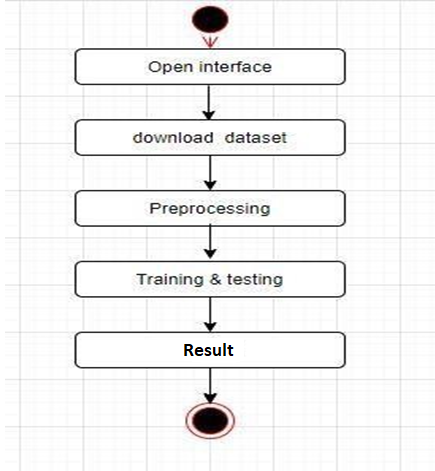
In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behaviour of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modeller’s use the «extend» relationship to indicate use cases that are "optional" to the base use case.



**Fig 6.2.1: Use case diagram**

**ACTIVITY DIAGRAM**

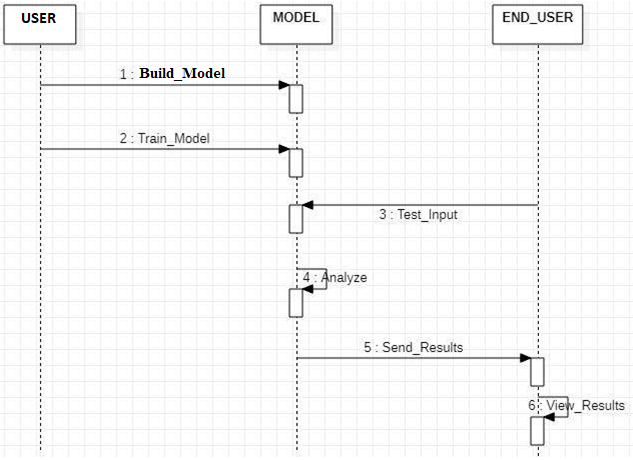
An activity diagram is a behavioural diagram i.e., it depicts the behaviour of a system. This is to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens.



**Fig 6.2.2: Activity Diagram**

**SEQUENCE DIAGRAM**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. The important aspect of the sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing the “messages”.



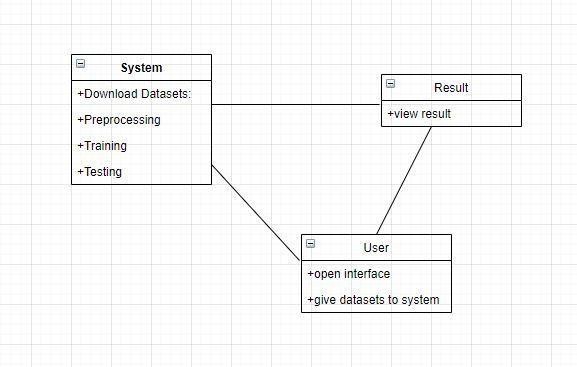
**Fig 6.2.3: Sequence Diagram**

**CLASS DIAGRAM**

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

The class diagram is the main building block in object-oriented modelling. It is used both for general conceptual modelling of the semantics of the application, and for detailed modelling translating the models into programming code. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain the three parts:

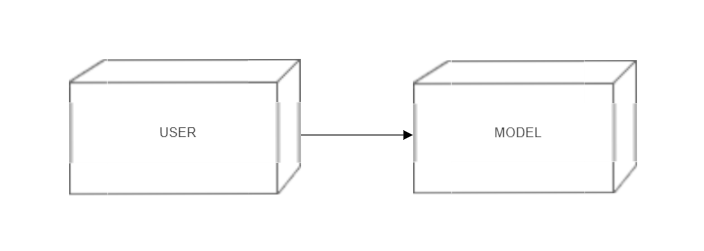
* The upper part holds the name of the class.
* The middle part contains the attributes of the class.
* The lower part contains the operations of the class.



**Fig 6.2.4: Class Diagram**

**DEPLOYMENT DIAGRAM**

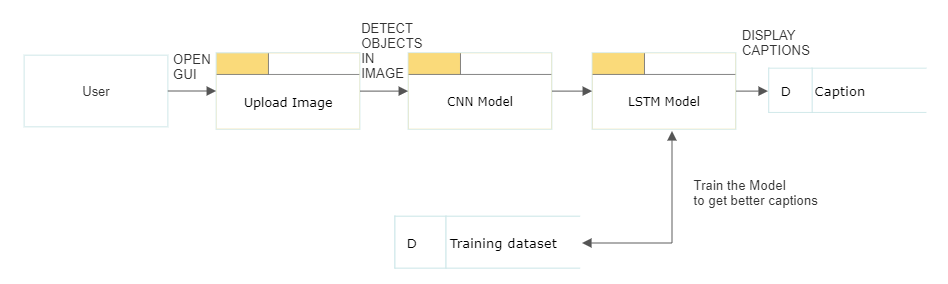
Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So, deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.



**Fig 6.2.5: Deployment Diagram**

**DATA FLOW DIAGRAM**

A data flow diagram is graphical tool used to describe and analyse movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labelled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD’S is done in several levels. Each process in lower-level diagrams can be broken down into a more detailed DFD in the next level. The lop- level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.



**Fig 6.2.5: Data Flow Diagram**

**CHAPTER- 7**

**IMPLEMENTATION**

**7.1 INTRODUCTION**

**The model requires:**

1. Methods from Computer Vision (to understand the content of the image)
2. Language model from NLP (to turn the understanding of image into Words)
3. After training the model, its learning will be tested using Deep Learning.

**Python 3**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum from 1985 to 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is named after a TV Show called ëMonty Pythonís Flying Circusí and not after Python-the snake.

Python 3.0 was released in 2008. Although this version is supposed to be backward-incompatible, later on, many of its important features have been backported to be compatible with version 2.7.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is an open-source, and it is automatically included with Linux distributions, Macintosh computers, and a wide range of products and hardware, further clouding the user base picture. Few points about the importance of Python:

* Python is a great open-source, interpreted, object-oriented programming language. There is no concept of structure like C/C++ & Go language.
* Decorators, Generators provide much functionality.
* Great web frameworks like Django, Flask, etc. that encourage rapid web development.
* With Flask, we can use loops, conditional statements, etc. inside the HTML document that adds dynamic nature to web pages.
* Python has a rich set of libraries/packages that we can use to create excellent applications, games, etc.
* We can easily install libraries using pip You can create great projects using different Python libraries for web scrapping, calculators, text editor & other innovative apps.
* For aspiring Data Scientists, Python is probably the most important language to learn because of its rich ecosystem. Python's major advantage is its breadth.

For example, R can run Machine Learning algorithms on a pre-processed dataset, but Python is much better at processing the data.

Python can be used in many ways, like

* **App development**: Python is one of the few languages that are not native to the Java world and yet can be used to develop apps easily for the android platform. Kivy is a framework available at Kivy: Cross-platform Python Framework for NUI that lets a developer design and test touch-friendly applications for android. Python support extension is available in Eclipse IDE for this purpose.
* **Web Development**: There are several python frameworks, most notable among them Django, a flask that is optimized to maximize the potential of the web via the use of python.
* **Backend development**: Python is a very powerful language with a lot of libraries built into it. These can be used to achieve nearly any activity that is needed via python, from cryptanalysis to GUI applications.

**7.2 SAMPLE CODE**

**Data Cleaning:**

# Loading a text file into memory

def load\_doc(filename):

    # Opening the file as read only

    file = open(filename, 'r')

    text = file.read()

    file.close()

    return text

# get all imgs with their captions

def all\_img\_captions(filename):

    file = load\_doc(filename)

    descriptions = {}

    for line in file.split('\n'):

        tokens = line.split()

        if len(line) > 2:

            image\_id = tokens[0].split(',')[0]

            image\_desc = ' '.join(tokens[1:])

        if image\_id not in descriptions:

              descriptions[image\_id] = []

        descriptions[image\_id].append(image\_desc)

    return descriptions

#Data cleaning- lower casing, removing puntuations and words containing numbers

def cleaning\_text(captions):

    table = str.maketrans('','',string.punctuation)

    for img,caps in captions.items():

        for i,img\_caption in enumerate(caps):

            img\_caption.replace("-"," ")

            desc = img\_caption.split()

            #converts to lowercase

            desc = [word.lower() for word in desc]

            #remove punctuation from each token

            desc = [word.translate(table) for word in desc]

            #remove hanging 's and a

            desc = [word for word in desc if(len(word)>1)]

            #remove tokens with numbers in them

            desc = [word for word in desc if(word.isalpha())]

            #convert back to string

            img\_caption = ' '.join(desc)

            captions[img][i]= img\_caption

    return captions

def text\_vocabulary(descriptions):

    # build vocabulary of all unique words

    vocab = set()

    for key in descriptions.keys():

        [vocab.update(d.split()) for d in descriptions[key]]

    #sort vocabulary

    vocab=list(vocab)

    vocab.sort()

    return vocab

#All descriptions in one file

def save\_descriptions(descriptions, filename):

    lines = list()

    for key, desc\_list in descriptions.items():

        for desc in desc\_list:

            lines.append(key + '\t' + desc )

    data = "\n".join(lines)

    file = open(filename,"w")

    file.write(data)

    file.close()

**Loading dataset for Training the model:**

def load\_photos(filename):

    file = load\_doc(filename)

    photos = file.split('\n')[:-1]

    return photos

def load\_clean\_descriptions(filename, photos):

    #loading clean\_descriptions

    file = load\_doc(filename)

    descriptions = {}

    for line in file.split('\n'):

        words = line.split()

        if len(words)<1 :

            continue

        image, image\_caption = words[0], words[1:]

        if image in photos:

            if image not in descriptions:

                descriptions[image] = []

            desc = '<start> ' + " ".join(image\_caption) + ' <end>'

            descriptions[image].append(desc)

    return descriptions

def load\_features(photos):

    #loading all features

    all\_features = load(open(os.path.join(save\_data,"flickr8k\_features.p"),"rb"))

    #selecting only needed features

    features = {k:all\_features[k] for k in photos}

    return features

**Creation of Data generator:**

#create input-output sequence pairs from the image description.

#data generator, used by model.fit\_generator()

def data\_generator(train, features, tokenizer, max\_length):

    while 1:

        for key, train\_list in train.items():

            #retrieve photo features

            feature = features[key][0]

            input\_image, input\_sequence, output\_word = create\_sequences(tokenizer, max\_length, train\_list, feature)

            yield [[input\_image, input\_sequence], output\_word]

def create\_sequences(tokenizer, max\_length, desc\_list, feature):

    X1, X2, y = list(), list(), list()

    # walk through each description for the image

    for desc in desc\_list:

        # encode the sequence

        seq = tokenizer.texts\_to\_sequences([desc])[0]

        # split one sequence into multiple X,y pairs

        for i in range(1, len(seq)):

            # split into input and output pair

            in\_seq, out\_seq = seq[:i], seq[i]

            # pad input sequence

            in\_seq = pad\_sequences([in\_seq], maxlen=max\_length)[0]

            # encode output sequence

            out\_seq = to\_categorical([out\_seq], num\_classes=vocab\_size)[0]

            # store

            X1.append(feature)

            X2.append(in\_seq)

            y.append(out\_seq)

    return np.array(X1), np.array(X2), np.array(y)

#You can check the shape of the input and output for your model

[a,b],c = next(data\_generator(train\_descriptions, features, tokenizer, max\_length))

a.shape, b.shape, c.shape

**Creation of Model:**

from tensorflow.keras.utils import plot\_model

# define the captioning model

def define\_model(vocab\_size, max\_length):

    # features from the CNN model squeezed from 2048 to 256 nodes

    inputs1 = Input(shape=(2048,))

    fe1 = Dropout(0.4)(inputs1)

    fe2 = Dense(256, activation='relu')(fe1)

    # LSTM sequence model

    inputs2 = Input(shape=(max\_length,))

    se1 = Embedding(vocab\_size, 256, mask\_zero=True)(inputs2)

    se2 = Dropout(0.4)(se1)

    se3 = LSTM(256)(se2)

    # Merging both models

    decoder1 = add([fe2, se3])

    decoder2 = Dense(256, activation='relu')(decoder1)

    outputs = Dense(vocab\_size, activation='softmax')(decoder2)

    # tie it together [image, seq] [word]

    model = Model(inputs=[inputs1, inputs2], outputs=outputs)

    model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics = ['accuracy'])

    # summarize model

    print(model.summary())

    return model

**CHAPTER- 8**

**TESTING**

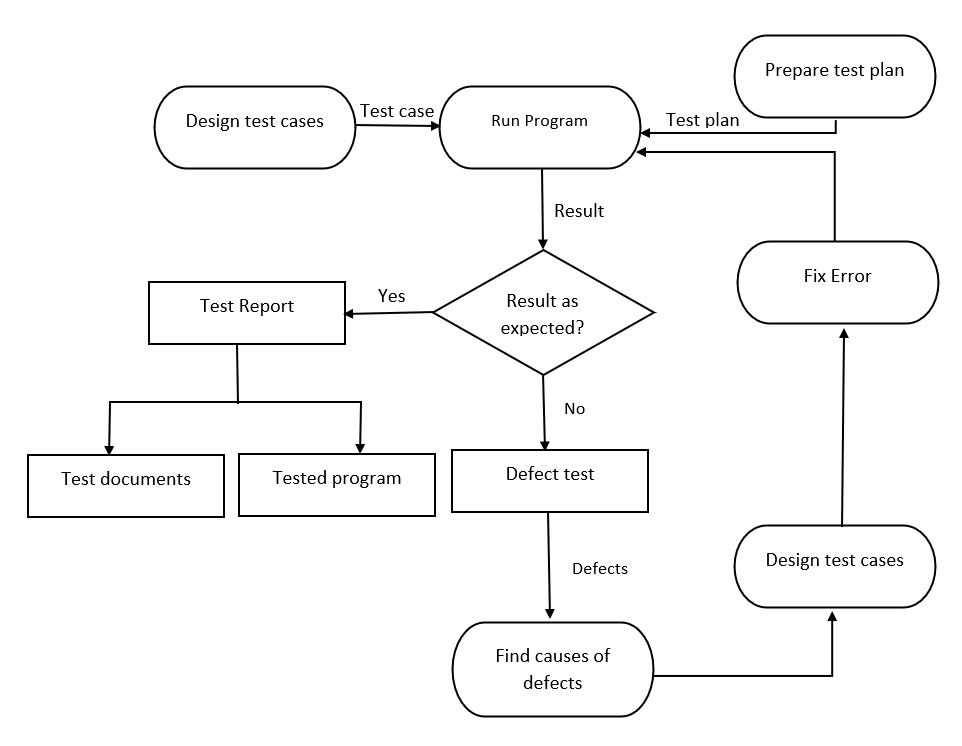
**8.1 TESTING INTRODUCTION**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement. Software testing is one of the important phases in software development. Effective testing reduces maintenance cost an provides reliable outcomes.

**8.2 TESTING PROCESS**

Testing is a disciplined process of finding and debugging defects to produce defect-free software. During testing, a test plan is prepared that specifies the name of the module to be tested, reference modules, date and time, location, name of the tester, testing tools, etc. Software engineers design test cases while writing the source codes, Testers may also involve in test case design.

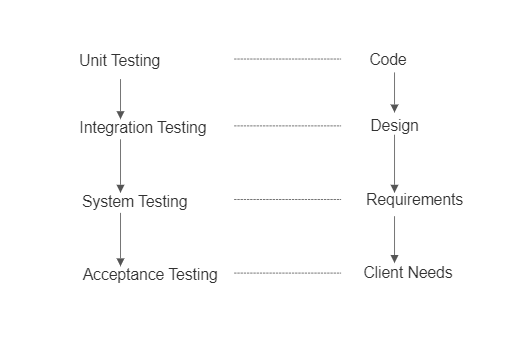
Test cases include the input, output, and the conditions. Software tester runs the program using test cases according to the test plan and observes the test results. We can validate the test rest cases also



**Figure 8.2: Testing Process**

**8.3 LEVEL OF TESTING**

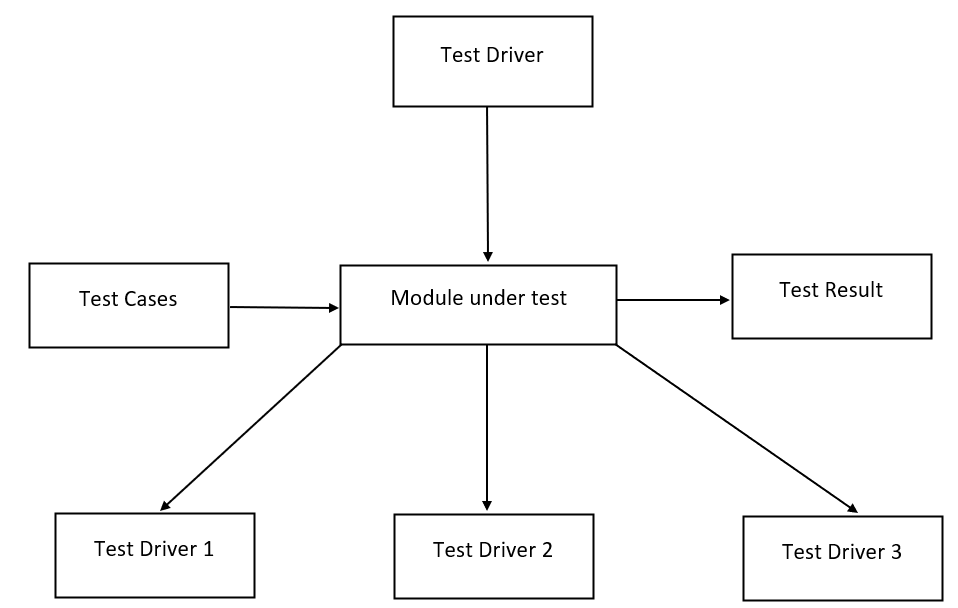
Testing is a defect detection technique that is performed at various levels. Testing begins once a module is fully constructed. Although software engineers test source codes after it is written, but it is not an appealing way that can satisfy customer’s needs and expectations. Software is developed through a series of activities, i.e., customer needs, specification, design, and coding. Each of these activities has different aims. Therefore, testing is performed at various levels of development phases to achieve their purpose.



**Figure 8.3: Testing Levels**

**UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.



**Figure 8.3.1: Unit Testing**

**INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

There are various approaches in which the modules are combined together for integration testing.

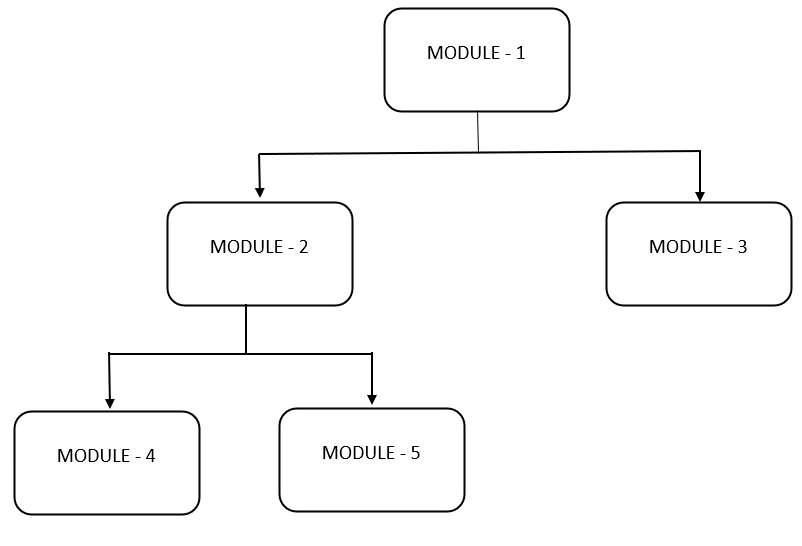
* Big-bang approach
* Top-down approach
* Bottom-up approach
* Sandwich approach

**BIG-BANG APPROACH**

The big-bang is a simple and straightforward integration testing. In this approach, all the modules are first tested individually and then these are combined together and tested as a single system. This approach works well where there is a smaller number of modules in a system. As all modules are integrated to form a whole system, the chaos may occur. If there is any defect found, it becomes difficult to identify where the defect has occurred. Therefore, big-bang approach is generally avoided for large and complex systems.

**TOP-DOWN APPROACH**

Top-down integration testing begins with the main module and move downwards integrating and testing its lower-level modules. Again, the next lower-level modules are integrated and tested. Thus, this incremental integration and testing is continued until all modules up to the concrete level are integrated and tested. Thus, writing test stubs and simulating to act as actual modules may be complicated and time-consuming task.



**Figure 8.3.2: Top-down Integration**

**BOTTOM-UP APPROACH**

As the name implies, bottom-up approach begins with the individual testing of bottom-level modules in the software hierarchy. Then lower-level modules are merged function wise together to form a subsystem and then all subsystems are integrated to test the main module covering all modules of the system.

The approach of bottom-up integration is as follows:

Concrete level modules -> subsystem –> main module.

The bottom-up approach works opposite to the top-down integration approach.

**SANDWICH APPROACH**

The sandwich testing combines both top-down and bottom-up integration approaches. During sandwich testing, top-down approach force to the lower-level modules to be available and bottom-up approach requires upper-level modules. Thus, testing a module requires its top and bottom level modules. It is the most preferred approach in testing because the modules are tested as and when these are available for testing.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before

functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

The unit and integration testing are applied to detect defects in the modules and the system as a whole. Once all the modules have been tested, system testing is performed to check whether the system satisfies the requirements (both functional and non-functional). To test the functional requirements of the system, functional or black-box testing methods are used with appropriate test cases. System testing is performed keeping in view the system requirements and system objectives. The non-functional requirements are tested with a series of tests whose purpose is to check the computer-based system. A single test case cannot ensure all the system non-functional requirements. For specific non-functional requirements, special tests are conducted to ensure the system functionality. Some of the non-functional system tests that are being used to test various features are:

Some of the non-functional system tests that are being used to test various features are:

* Performance testing
* Volume testing
* Stress testing
* Security testing
* Recovery testing
* Compatibility testing
* Configuration testing
* Installation testing
* Documentation testing

**ACCEPTANCE TESTING**

Acceptance testing is a kind of system testing, which is performed before the system is released into the market. It is performed with the customer to ensure that the system is acceptable for delivery. Once all system testing has been exercised, the system is now tested from the customer’s point of view. Acceptance testing is conducted because there is a difference between the actual user and the simulated users considered by the development organization. The user involvement is important during acceptance testing of the software as it is developed for the end-users.

Acceptance testing is performed at two levels, i.e.

* Alpha testing
* Beta testing

**BLACK-BOX TESTING**

Black-Box testing is performed on the basis of functions or features of the software. In black-box testing, only the input values are considered for the design of test cases. The output values that the software provides on execution of test cases are observed. The internal logic or program structures are not considered during black box testing. It is also known as behavioural or functional testing.

There are a number of black-box test case design methods:

* Equivalence class partitioning
* Boundary value analysis
* Cause-effect graphing
* Error guessing

**WHITE-BOX TESTING**

White box testing is a method of testing software that tests internal structures or working of an application. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. It is also known as Clear Box testing, Glass Box testing, Transparent Box testing, and Structural testing.

The following white-box testing methods are widely used for testing the software:

* Control flow-based testing
* Path testing
* Data flow-based testing
* Mutation testing

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.

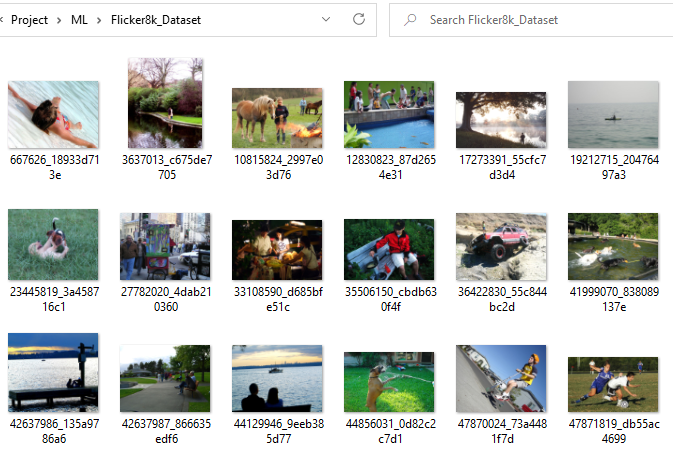
**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

**CHAPTER- 9**

**RESULTS AND DISCUSSION**

**9.1 DATA COLLECTION:**



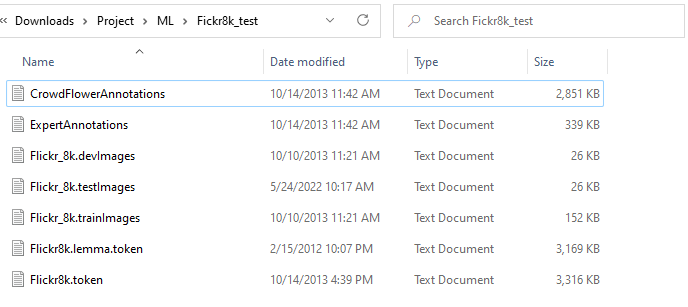
**Figure 9.1.1: IMAGE DATASET**

Data pre-processing and cleaning is an important part of the whole model building process. Understanding the data helps us to build more accurate models.

After extracting zip files, you will find the above folders…

**Flickr8k\_Dataset:** This contains a total of 8092 images in JPEG format, with different shapes and sizes. Of which 6000 are used for training, 1000 for test and 1000 for development.

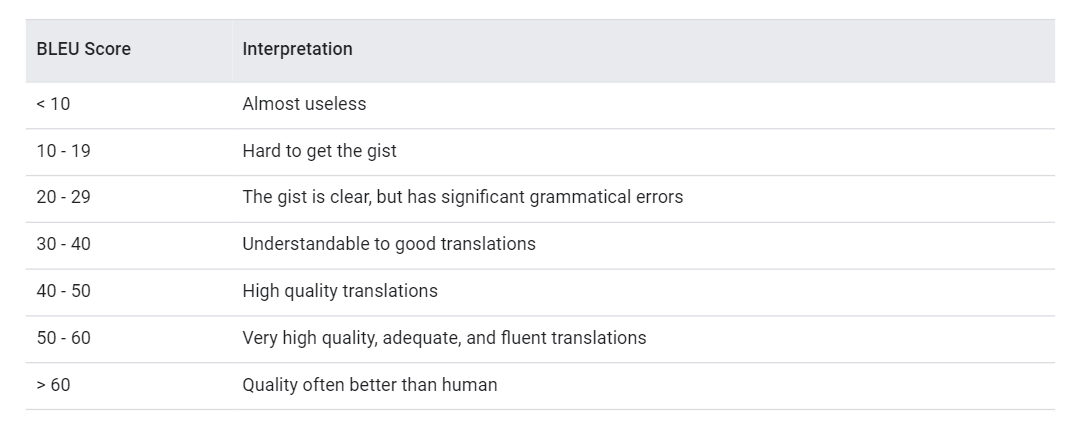
**Flickr8k\_text:** Contains text files describing train\_set, test\_set. Flickr8k.token.txt contains 5 captions for each image, i.e., total 40460 captions.



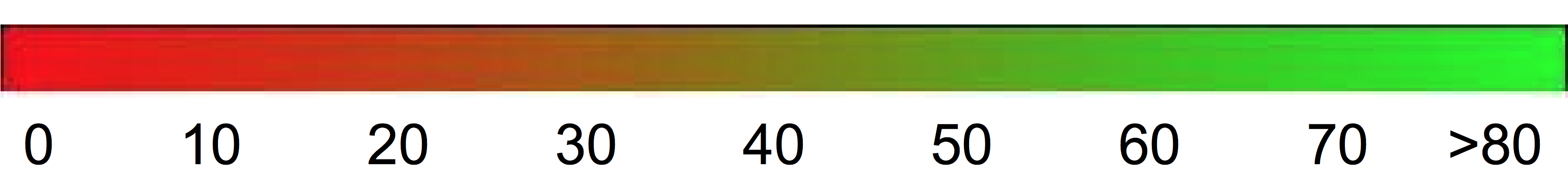
**Figure 9.1.2: TEXTUAL DATASET**

**9.2 BILINGUAL EVALUATION UNDERSTUDY:**

**BLEU** is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the correspondence between a machine’s output and that of a human. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high-quality reference translations. A value of 0 means that the machine-translated output has no overlap with the reference translation (low quality) while a value of 1 means there is perfect overlap with the reference translations (high quality).

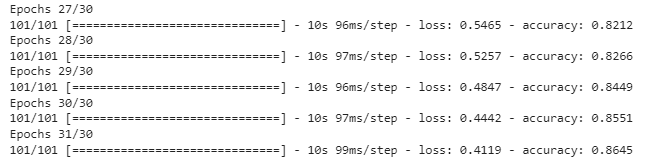


**Figure-9.2.1: Interpretation of BLEU scores**



**Figure-9.2.2: Color Gradient of BLEU Scores**

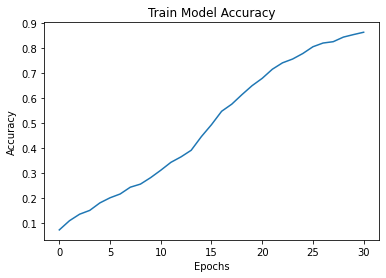
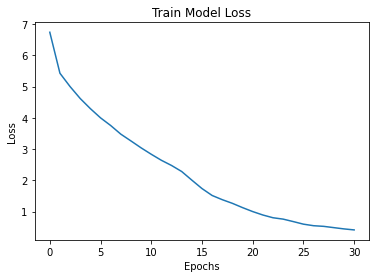
**9.3 TRAINING AND TESTING THE MODEL:**

**Figure 9.3.1: Accuracy of the Model**

To make sure the model can predict well, there are steps in testing the model. The first step s making predictions on the testing set. The result for 30 iterations in checking the loss and accuracy when training the model is shown in the above figure.

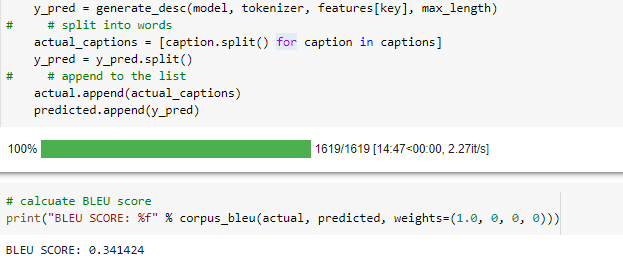
The Above is the result of Accuracy of the Model and here we are getting the percentage of ~**86%.**

** **

**Figure 9.3.2: Train Model Accuracy Figure 9.3.3: Train Model Loss**

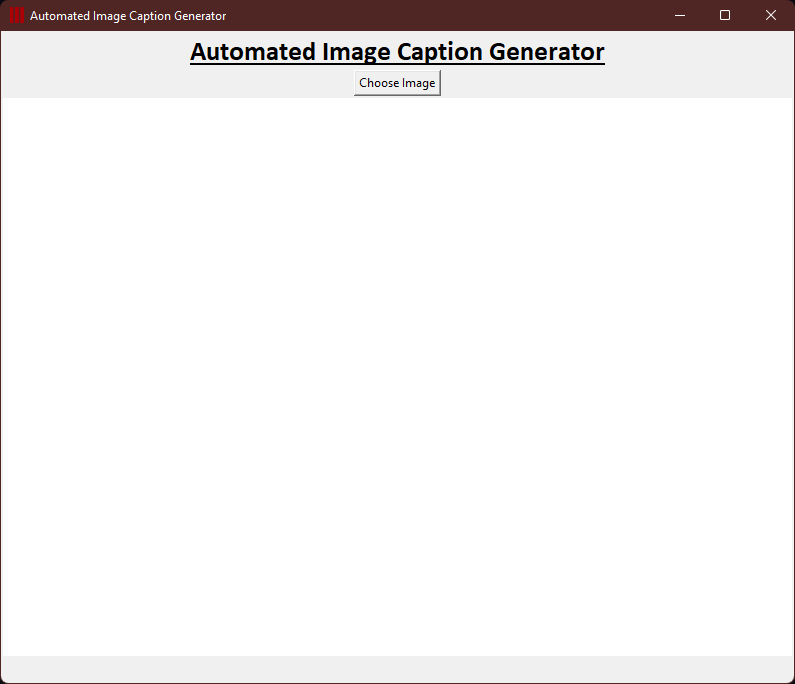
Here is the **Figure 9.1.4,** Representation of Train Model Accuracy and we observe the result is gradually **increasing**.

Here is the **Figure 9.1.5,** Representation of Train Model Loss and we observe the result is gradually **decreasing**.

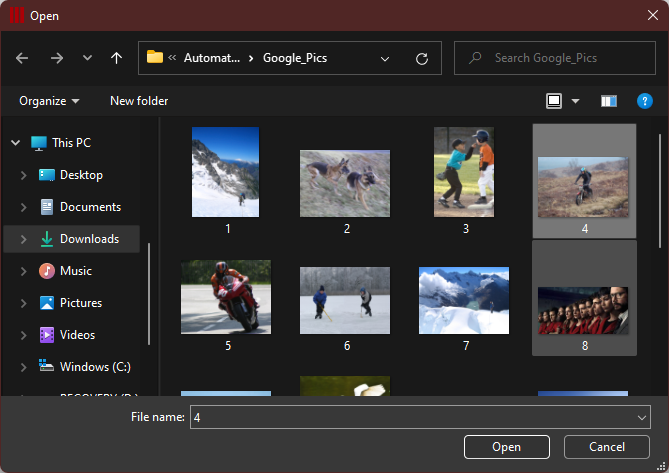


**Figure 9.3.4: BLEU SCORE of the Model**

**9.4 USER INTERFACE:**

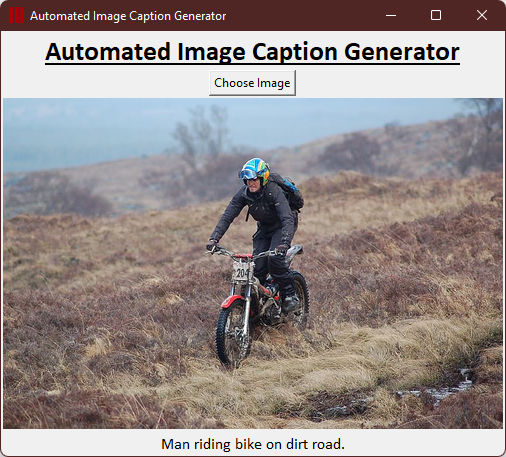
****

**Figure- 9.4.1: UI APPLICATION**

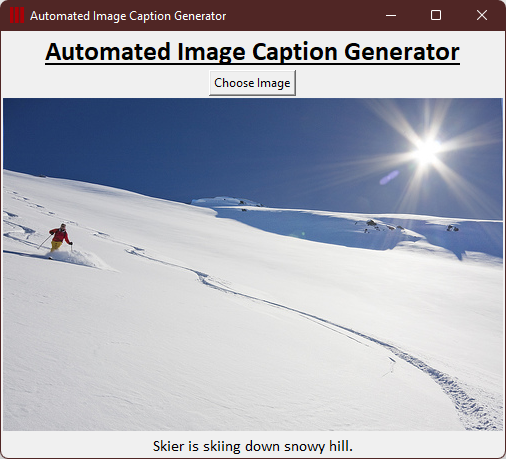
****

**Figure 9.4.2: Choosing the Image file**

**9.5 RESULTS:**

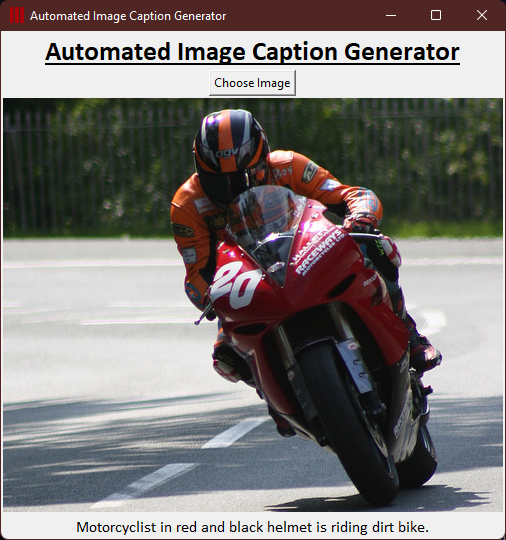


**Figure 9.5.1: Testcase- 1**

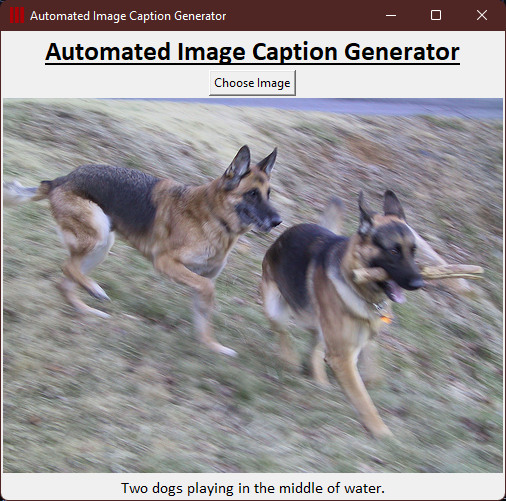


**Figure 9.5.2: Testcase- 2**

**DISCUSSION:**

****

**Figure 9.6.1: Testcase- 3**

****

**Figure 9.6.2: Testcase- 4**

From the above **Figure 9.6.1**, **Figure 9.6.2,** the limitations of retrieval-based image captioning methods are obvious. Such methods transfer well-formed human-written sentences or phrases to generate descriptions for query images. Although the yielded outputs are usually grammatically correct and ﬂuent, constraining image descriptions to sentences that have already existed cannot adapt to new combinations of objects or novel scenes.

Under certain conditions, generated descriptions may even be irrelevant to image contents. Retrieval-based methods have large limitations in their capability to describe images. Our model will depend on the data, so it cannot predict the words that are out of the scope of its vocabulary.

**CONCLUSION**

Automatically image captioning is far from mature, and there are a lot of ongoing research projects aiming for more accurate image feature extraction and semantically better sentence generation. We completed what we mentioned in the project proposal but used a smaller dataset (Flickr8k) due to limited computational power. There can be potential improvements if given more time. It worked quite well when tested on several images. The captions it generated for the images were quite accurate. But the source of the input image also played an important role in feature extraction and, hence, caption generation. Certain images are not well recognized, and we found out that there is still some scope for improvement. First of all, we directly used the pre-trained CNN network as part of our pipeline without fine-tuning, so the network does not adapt to this specific training dataset. Thus, by experimenting with different CNN pre-trained networks.

**FUTURE WORK**

* We can deploy it as a web app.
* Deep Learning model to road accidents using this as pre-model.
* Improvisations: We can take a more generalized dataset which may produce more accurate results.
* Another potential improvement is by training on a combination of Flickr8k, Flickr30k, and MS COCO. In general, the more diverse the training dataset the network has seen, the more accurate the output will be. We all agree this project ignites our interest in the application of Machine Learning knowledge in Computer Vision and expects to explore more in the future.
* Advanced loss function: The original SoftMax loss function can cause problems. It can produce force negatively. For example, if we input the test picture with the caption A man is riding a horse, the produced caption A horse is carrying a horse will produce high loss, but actually, these two captions all correctly describe the picture. On the other hand, the model can also produce force negatively.

**REFERENCES**

1. MM. Z. Hossain et al. "Text to Image Synthesis for Improved Image Captioning", 2021 IEEE Journal on IEEE, 2021.
2. Jyoti Aneja, Aditya Deshpande, Alexander Schwing, "Convolutional Image Captioning", University of Illinois, 2017, [Online] Available: https://arxiv.org/pdf/1711.09151.pdf.
3. Chetan Amritkar, Vaishali Jabade, "Image Caption Generation using Deep Learning Technique" 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018.
4. Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on. IEEE, 2015.
5. A Gentle Introduction to Transfer Learning for Deep Learning, https://bit.ly/3Ej5he1
6. Brett Grossfeld(2020, 23 January). Deep learning vs machine learning: a simple way to understand the difference: https://www.zendesk.com/blog/machine-learning-and-deep-learning/#:~:text=To%20recap%20the%20differences%20between,intelligent%20decisions% z0on%20its%20own
7. Moses Soh. Research Article. Department of Computer Science. Stanford University. Learning CNN-LSTM Architectures for Image Caption Generation: https://cs224d.stanford.edu/reports/msoh.pdf
8. Sanam Malhotra. Oodles AI (2020, 8 April). Building and Deploying an AI-powered Image Caption Generator: https://artificialintelligence.oodles.io/blogs/ai-powered-image- captiongenerator/#:~:text=The%20advent%20of%20machine%20learning,better%20sense%2 0of%20their%20surroundings.
9. Haoran Wang. Research Article. An Overview of Image Caption Generation Methods: <https://www.hindawi.com/journals/cin/2020/3062706/>
10. Dr. Vinayak D.Shinde, Mahiman P.Dave, Anuj M.Singh, Amit C.Dubey. Research Journal. International Research Journal of Engineering and Technology (IRJET) (2020, April). Image Caption Generator using Big Data and Machine Learning: <https://www.irjet.net/archives/V7/i4/IRJET-V7I41167.pdf>
11. Flickr8k Dataset link from Kaggle Website: <https://www.kaggle.com/shadabhussain/flickr8k>
12. Evaluating Models | AutoML, Understanding the BLEU Score <https://cloud.google.com/translate/automl/docs/evaluate>
13. Calculating the BLEU Score for Text in Python

https://machinelearningmastery.com/calculate-bleu-score-for-text-python/