

IMAGE CAPTION GENERATOR



A MAJOR PROJECT REPORT

Submitted by

PRAVEEN A (730920104085)

RAMPRADEEP J (730920104092)

SURIYAPRAKASH P (730920104113)

THAMARAI SELVAN G (730920104115)

*in partial fulfillment for the award of the degree
of*

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

EXCEL ENGINEERING COLLEGE (AUTONOMOUS)

KOMARAPALAYAM - 637 303

ANNA UNIVERSITY :: CHENNAI – 600 025

APRIL 2024

EXCEL ENGINEERING COLLEGE (AUTONOMOUS)

KOMARAPALAYAM – 637 303

ANNA UNIVERSITY, CHENNAI : 600 025

BONAFIDE CERTIFICATE

Certified that this project report “IMAGE CAPTION GENERATOR” is the bonafide work of “**PRAVEEN A (730920104085), RAMPRADEEP J (730920104092), SURIYAPRAKASH P (730920104113), THAMARAI SELVAN G (730920104115)**”

Who carried out the project work under my supervision.

SIGNATURE

Dr. P. C. SENTHIL MAHESH M.E., Ph.D.,

HEAD OF THE DEPARTMENT,

Professor and Head,

Department of CSE,

Excel Engineering College,

Komarapalayam-637303.

SIGNATURE

Mr. N. NANDA KUMAR M.E.,

SUPERVISOR,

Assistant Professor,

Department of CSE,

Excel Engineering College,

Komarapalayam-637303.

Submitted for the University Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who actuated it, without them it would never have into existence. To them, we lay word of gratitude imprinted within ourselves.

We wish our heartfelt thanks to our respected Founder and Chairman of Excel Group Institutions **Prof. Dr. A. K. NATESAN M.Com., M.B.A., M.Phil., PhD., FTA** and Vice Chairman **Dr. N. MATHAN KARTHICK M.B.B.S., M. H. Sc (Diabetology)** for allowing us to have the extensive use of the college facilities to do our project effectively.

We express our sincere gratitude and heartfelt thanks to the respected Principal **Dr. K. BOMMANNA RAJA M.Tech., Ph.D.,** and **Dr. C. KARTHIKEYENI,** Academics Director for her encouragement and support to complete the project.

We would like to express our profound interest and sincere gratitude to the Head of the Department **Dr. P. C. SENTHIL MAHESH M.E., Ph.D.,** Department of Computer Science and Engineering for his encouragement and support to complete the project.

We would like our sincere gratitude and heartfelt thanks to our Project Coordinator **Mrs. J. OBURADHA M.E,** Assistant Professor, Department of Computer Science and Engineering for continuous help over the period and creative ideas for this phase of our project work.

We are privileged to express our deep sense of gratitude to Project Supervisor **Mr. N. NANDA KUMAR M.E,** Assistant Professor Department of Computer Science and Engineering who gave guidance and support throughout our work and made this as a successful project.

Finally, we thank the Almighty, all my Staff Members, Parents, Friends and well Wishers for the moral support throughout the project.

ABSTRACT

The Image caption generator is a task that involves computer vision and NLP(natural language processing) Concepts to recognize the context of an image and describe them in a natural language like English. And the Convolutional Neural Network(CNN) and Long Short Term Memory(LSTM) can be combined to create an image caption generator and generate captions for your own images. In the field of artificial intelligence, image caption has gradually attracted the attention of many Programmers and developers, And the application of image caption is extensive and significant , It will detect the relationship between people, animals. This project will help us to understand the related methods and attention mechanism, which plays an important role in computer vision and is recently widely used in image caption generation tasks. And the advantages and the shortcomings of these methods are implemented, providing the commonly used datasets and evaluation criteria in this field. and it explains how the present scenarios happening in front of visually impaired people. The advent of machine learning solutions like image captioning is a boon for visually impaired people who are unable to comprehend visuals. With the use of an image caption generator, the image descriptions can be read out to visually impaired people, and our technology will be act as their eyes

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iii
	LIST OF FIGURES	Vi
	LIST OF ABBREVIATIONS	vii
1	INTRODUCTION	1
	1.1 DESCRIPTION	1
	1.2 OVERVIEW	6
2	LITERATURE SURVEY	7
3	SYSTEM ANALYSIS	12
	3.1 EXISTING SYSTEM	12
	3.2 PROPOSED SYSTEM	12
4	SYSTEM REQUIREMENT	15
	4.1 HARDWARE REQUIREMENT	15
	4.2 SOFTWARE REQUIREMENT	15
	4.3 SOFTWARE DESCRIPTION	15
5	SYSTEM IMPLEMENTATION	23
	5.1 MODULE LIST	23
	5.2 MODULE DESCRIPTION	23
6	RESULT AND DISCUSSION	27
7	CONCLUSION AND FUTURE ENHANCEMENT	29
	APPENDIX	30
	OUTPUT SCREEN SHOTS	39
	PLAGIARISM REPORT	44
	REFERENCE	46

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.2	Proposed Architecture	14
5.1	Input Image	24
5.2	Feature Extraction	25
6.1	Dataset Collection	28

LIST OF ABBREVIATIONS

CNN	:	Convolutional Neural Network
RNN	:	Recurrent Neural Network
LSTM	:	Long Short Term Memory
API	:	Application Programming Language

CHAPTER 1

INTRODUCTION

1.1 Description

Image caption generation is a captivating field at the intersection of computer vision and natural language processing, aimed at endowing machines with the ability to comprehend and describe visual content in human-like language. In today's digital age inundated with images, the demand for intelligent systems capable of automatically generating descriptive captions has surged, spanning diverse applications ranging from aiding visually impaired individuals to enhancing content accessibility and enriching user experiences in social media, e-commerce, and image archives. Image caption generation involves teaching machines to understand the contents of an image and express it in a coherent and descriptive sentence or phrase. This task goes beyond simple object recognition, requiring a deeper comprehension of the scene, including relationships between objects, their attributes, spatial arrangements, and contextual nuances. It encapsulates the challenge of bridging the semantic gap between low-level visual features extracted from images and the high-level semantics inherent in natural language. The development of image captioning systems has been greatly propelled by advancements in deep learning, particularly convolutional neural networks (CNNs) for image processing and recurrent neural networks (RNNs) or transformer-based architectures for natural language generation. These models are typically trained on large-scale datasets containing paired images and corresponding human-generated captions, leveraging techniques such as attention mechanisms to align visual and textual modalities effectively.

One of the intriguing aspects of image caption generation is its interdisciplinary nature, drawing inspiration from cognitive science, linguistics, and psychology to create more human-like descriptions. Beyond merely describing objects, successful captioning models often strive to imbue captions with creativity, diversity, and contextual relevance, mirroring the richness and variability of human language. Despite significant progress, image captioning remains a challenging task, characterized by inherent ambiguities, diverse linguistic styles, and the subjective nature of perception and interpretation. Ongoing research focuses on addressing these challenges through innovations in multimodal fusion, fine-grained object recognition, commonsense reasoning, and leveraging large-scale pretraining techniques. Image caption generation stands at the forefront of artificial intelligence research, driving innovations in both computer vision and natural language processing domains.

Image Processing:

Image processing is a field of study that involves the manipulation and analysis of digital images using computational techniques. It plays a crucial role in various domains such as medicine, remote sensing, surveillance, entertainment, and more. The primary goal of image processing is to enhance images for better visualization, extract useful information, and make automated decisions based on image content. The image processing involves a series of operations performed on an input image to obtain an output image with improved quality or to extract relevant information. These operations can range from simple tasks like resizing and noise reduction to complex tasks like object detection and pattern recognition.

One fundamental concept in image processing is digital image representation. Digital images are composed of discrete pixels arranged in a grid format, where each pixel represents a specific color or intensity value. These values can be manipulated using mathematical operations to

achieve desired effects. There are two main categories of image processing: analog and digital. Analog image processing involves manipulating physical photographs or film using traditional techniques like filtering, cropping, and dodging. On the other hand, digital image processing utilizes computers to perform operations on digital images stored in electronic format.

Common image processing techniques include:

- **Filtering:** Filtering is the process of modifying the frequency content of an image to enhance or suppress certain features. Common filters include smoothing filters for noise reduction and sharpening filters for edge enhancement.
- **Segmentation:** Segmentation divides an image into meaningful regions or objects based on their characteristics, such as color, intensity, or texture. It is often used for object detection, image recognition, and medical image analysis.
- **Feature extraction:** Feature extraction involves identifying and extracting relevant information or features from an image, such as edges, corners, or texture patterns. These features can be used for tasks like image matching, object tracking, and classification.
- **Morphological operations:** Morphological operations manipulate the shape and structure of objects in an image, such as dilation, erosion, opening, and closing. These operations are useful for tasks like image binarization and shape analysis.

Overall, image processing plays a vital role in various applications, contributing to advancements in fields such as healthcare, agriculture, security, and entertainment. With ongoing research and development, the capabilities of image processing continue to expand, driving innovation and improving our understanding of visual data.

Deep Learning:

Image caption generation is a task within the realm of computer vision and natural language processing that involves generating textual descriptions for images. Deep learning, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), has revolutionized this field, enabling the creation of sophisticated and contextually relevant captions for images. Convolutional Neural Networks (CNNs) are typically used for feature extraction from images. These networks are adept at learning hierarchical representations of images, starting from simple features like edges and textures to complex semantic features. Pre-trained CNNs such as VGG, ResNet, and Inception have been widely adopted in image captioning tasks due to their ability to extract rich visual features effectively. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, are commonly employed for generating sequential data such as natural language descriptions. In image captioning, RNNs take the extracted visual features from CNNs as input and generate captions word by word. These networks learn to capture the temporal dependencies and linguistic structures present in captions.

One popular architecture for image captioning is the Encoder-Decoder framework. In this setup, the CNN serves as the encoder, extracting visual features from the input image, while the RNN acts as the decoder, generating captions based on these features. This architecture allows for end-to-end training, where both the CNN and RNN parameters are optimized jointly to produce coherent and relevant captions. Attention mechanisms have further enhanced the performance of image captioning models by allowing the decoder to focus on different parts of the image while generating each word in the caption. This attention mechanism enables the model to generate more detailed and contextually relevant captions, particularly for complex images with multiple objects or scenes. Overall, deep learning techniques have significantly advanced the

field of image captioning, enabling the development of models that can automatically generate human-like descriptions for a wide range of images, with applications spanning from assistive technologies for the visually impaired to content-based image retrieval systems. Deep learning has revolutionized the field of image caption generation by enabling the creation of more accurate and contextually relevant descriptions for images. Here are some of the key benefits:

- **Automatic Captioning:** Deep learning models, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have the capability to automatically generate captions for images without human intervention. This automation saves time and effort in manually annotating images.
- **Semantic Understanding:** Deep learning models can learn intricate patterns and features from images, allowing them to develop a deeper semantic understanding of visual content. This understanding enables the generation of captions that accurately describe the content and context of the images.
- **Contextual Awareness:** Deep learning models are adept at capturing contextual information from images, such as objects, scenes, and relationships between entities. This contextual awareness helps in generating captions that are not only descriptive but also contextually relevant and coherent.
- **Flexibility and Adaptability:** Deep learning models can be trained on large datasets with diverse image-caption pairs, making them highly flexible and adaptable to different domains and scenarios. Whether it's describing everyday scenes, specialized medical images, or abstract artwork, deep learning models can be fine-tuned to generate captions tailored to specific contexts.
- **Improved Accuracy:** Deep learning techniques, such as attention mechanisms and transformer architectures, have significantly improved the accuracy of image caption generation systems. These models can effectively attend to relevant regions of the image while generating captions, resulting in more precise and detailed descriptions.

- **Multimodal Integration:** Deep learning facilitates the integration of multiple modalities, such as images and text, in a unified framework. This multimodal approach allows for richer and more informative captions by leveraging both visual and textual information.
- **Continual Improvement:** Deep learning models can be continuously refined and enhanced through iterative training on large-scale datasets. As more data becomes available and model architectures evolve, image captioning systems can improve in terms of both accuracy and diversity of generated captions.

1.2 Overview

Image caption generation is a fascinating task in artificial intelligence that involves generating textual descriptions for images automatically. This field lies at the intersection of computer vision and natural language processing, aiming to teach machines to understand the content of images and express it in human-readable language. The goal is to develop algorithms that can accurately describe the contents of an image in a concise and meaningful way. The process usually begins with a CNN extracting features from the image, which are then fed into an RNN or a transformer model to generate the corresponding caption. One common approach is the encoder-decoder framework, where the CNN serves as the encoder, extracting visual features from the image, and the RNN or transformer acts as the decoder, generating the caption word by word based on the extracted features. This framework has been widely adopted due to its effectiveness in capturing both visual and semantic information. Training such models typically requires large datasets of images paired with human-generated captions, such as MSCOCO or Flickr30k. These datasets are used to train the neural network to associate specific visual features with corresponding textual descriptions. Image caption generation has numerous applications across various domains, including assistive technologies for the visually impaired, content-based image retrieval, and enhancing user experiences in photo-sharing platforms.

CHAPTER 2

LITERATURE SURVEY

2.1 Image Caption Generator using Deep Learning

Author: M. Sailaja; K. Harika; B. Sridhar

Year: 2022

Over the last few years deep neural network made image captioning conceivable. Image caption generator provides an appropriate title for an applied input image based on the dataset. The present work proposes a model based on deep learning and utilizes it to generate caption for the input image. The model takes an image as input and frame the sentence related to the given input image by using some algorithms like CNN and LSTM. This CNN model is used to identify the objects that are present in the image and Long Short-Term Memory (LSTM) model will not only generate the sentence but summarize the text and generate the caption that is suitable for the project. So, the proposed model mainly focuses on identify the objects and generating the most appropriate title for the input images.

2.2 Image Caption Generator using EfficientNet

Author: Sai Vikram Patnaik; Rohi Mukka; Roy Devpreyo

Year: 2022

A human can give a quick glance at a picture or a scenery and will be able to describe that scene very accurately and comprehensively. Image Captioning is a well-known but also a challenging task in research. The reason being that it is the intersection between Computer Vision and Natural Language Processing which should be able to develop and provide us with accurate description of an image. In this paper, we try to implement EfficientNet - the most powerful CNN model which uses less parameters and gives us a better prediction. We use the Microsoft: Common Objects in Context (MSCOCO) Dataset which contains around 330K images with 1.5M object instances and 80 object categories. We evaluate the model using BLEU-4 score, which indicates how similar the candidate text is to the reference texts, with values closer to one representing more similar texts. Our proposed model shows better understanding of the image and provides us with desirable results.

2.3 Image Caption Generator Using Attention Mechanism

Author: Vaishnavi Agrawal; Shariva Dhekane;

Year: 2021

Image captioning is used to generate sentences describing the scene captured in the form of images. It identifies objects in the image, performs a few operations, and tries to find the salient features of the image. Once the system identifies this information, it should further generate the most relevant and brief description for the image, which should be both syntactically and semantically correct. With the advancements of Learning techniques, algorithms can generate text in the form of natural sentences that will be able to describe an image in its best form. The natural ability of humans to understand image content and generate descriptive text is a challenging task for a machine to imitate. The applications of image caption generation are extensive and significant. The task involves generating brief captions using various techniques like Natural language processing (NLP), Computer vision (CV), and Deep Learning (DL) techniques. This paper introduces a system that uses an attention mechanism alongside an encoder and a decoder to generate the captions. It uses a pre-trained Convolutional Neural Network (CNN) viz. Inception V3 to extract features of the image and then a Recurrent Neural Network (RNN) viz. GRU to generate a relevant caption.

2.4 Detection and Recognition of Objects in Image Caption Generator System: A Deep Learning Approach

Author: N. Komal Kumar; D. Vigneswari

Year: 2019

Image Caption Generator deals with generating captions for a given image. The semantic meaning in the image is captured and converted into a natural language. The capturing mechanism involves a tedious task that collaborates both image processing and computer vision. The mechanism must detect and establish relationships between objects, people, and animals. The aim of this paper is to detect, recognize and generate worthwhile captions for a given image using deep learning. Regional Object Detector (RODe) is used for the detection, recognition and generating captions. The proposed method focuses on deep learning to further improve upon the existing image caption generator system. Experiments are conducted on the Flickr 8k dataset using python language to demonstrate the proposed method.

2.5 Camera2Caption: A real-time image caption generator

Author: Pranay Mathur; Aman Gill; Aayush Yadav

Year: 2017

The recent advances in Deep Learning based Machine Translation and Computer Vision have led to excellent Image Captioning models using advanced techniques like Deep Reinforcement Learning. While these models are very accurate, these often rely on the use of expensive computation hardware making it difficult to apply these models in realtime scenarios, where their actual applications can be realised. In this paper, we carefully follow some of the core concepts of Image Captioning and its common approaches and present our simplistic encoder and decoder based implementation with significant modifications and optimizations which enable us to run these models on low-end hardware of hand-held devices. We also compare our results evaluated using various metrics with state-of-the-art models and analyze why and where our model trained on MSCOCO dataset lacks due to the trade-off between computation speed and quality. Using the state-of-the-art TensorFlow framework by Google, we also implement a first of its kind Android application to demonstrate the realtime applicability and optimizations of our approach.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing System:

Image recognition using image processing is a pivotal aspect of computer vision, enabling machines to interpret and understand visual information akin to human perception. This technique involves extracting meaningful features from images, followed by classification or identification of objects, patterns, or scenes within those images. Image processing algorithms such as edge detection, segmentation, and feature extraction are employed to enhance and analyze images, facilitating accurate recognition. Various applications, including autonomous vehicles, medical imaging, security surveillance, and augmented reality, benefit from image recognition systems, enhancing efficiency, safety, and convenience in diverse domains. This system explores the principles, methods, and applications of image recognition using image processing, shedding light on its significance in modern technology and its potential for further advancements.

Disadvantages:

- It may require significant computational resources and time.
- Training data and manual annotation can be resource-intensive and time-consuming.
- Limited number of images are used.

3.2 Proposed System

Image captioning is a challenging task in computer vision and natural language processing, aiming to automatically generate textual descriptions for images. In this paper, we propose a novel approach to enhance image caption generation by integrating voice input into Convolutional Neural Networks (CNNs). Our method leverages the complementary strengths

of visual and auditory modalities to improve the quality and relevance of generated captions. First step preprocess image data using CNNs to extract high-level features, capturing visual semantics effectively. Subsequently, we incorporate voice input, allowing users to provide verbal cues to guide the caption generation process. We employ advanced natural language processing techniques to integrate voice information seamlessly into the captioning pipeline. Experimental results demonstrate that our voice-enabled CNN model achieves superior performance compared to traditional image captioning systems, both quantitatively and qualitatively. Furthermore, user studies indicate that incorporating voice input enhances the interpretability and personalization of generated captions, catering to diverse user preferences and scenarios.

Advantages

- Users can simply describe the image verbally, mimicking natural human communication.
- Lead to quicker generation of image captions, enhancing user experience.
- This system can potentially lead to more accurate and contextually relevant captions.
- Potentially increasing user satisfaction and encouraging continued usage of the system.

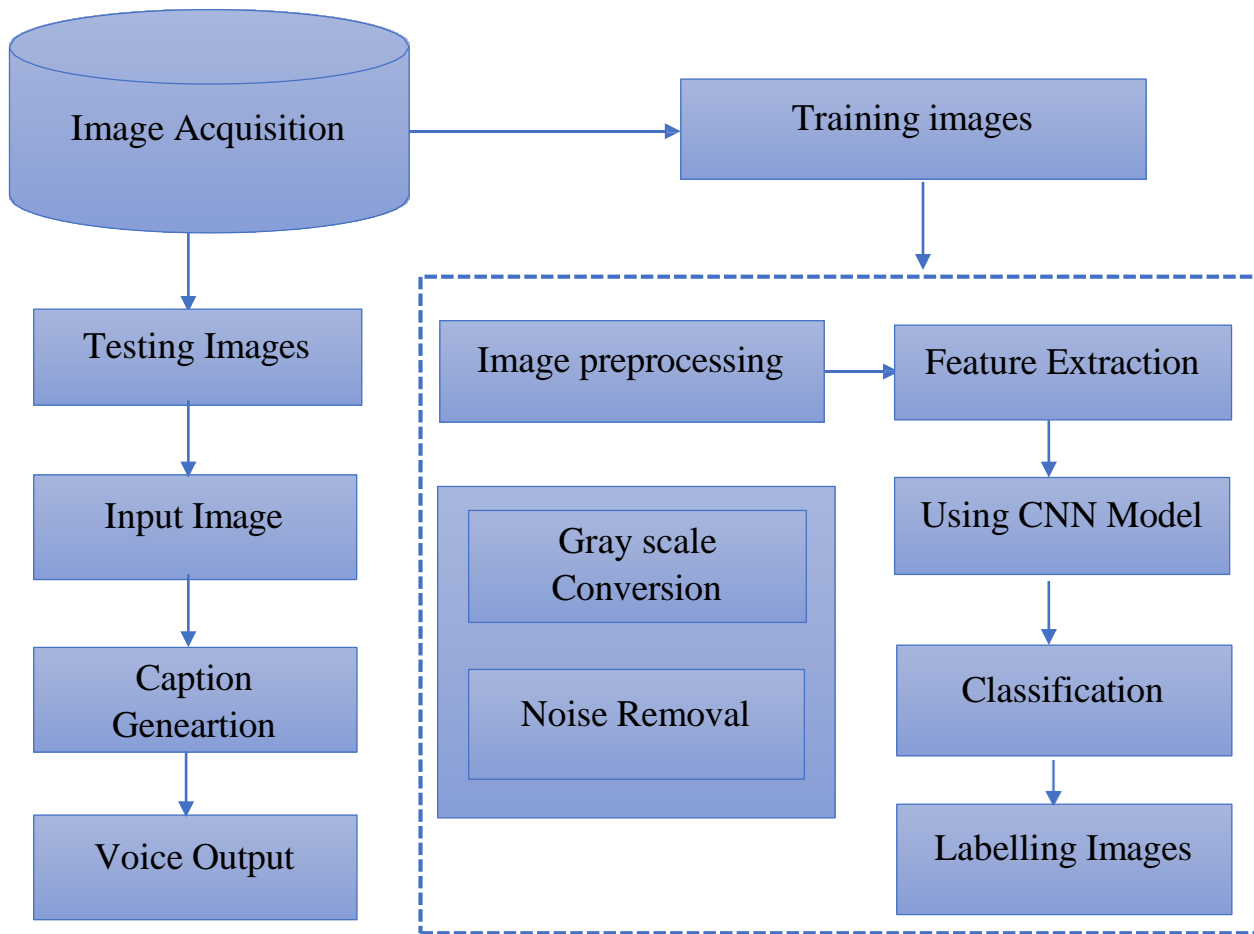


Fig 3.2: Proposed Architecture Diagram

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

- CPU Type : Any Intel or AMD x64 with 2 Core processor
- Clock speed : 2 GHz and Above
- RAM size : 8 GB Recommended
- Disk Space : 20 GB
- Display / Resolution : Min 14” / 1024 by 768 or Higher Display Resolution
- Keyboard & Mouse : USB or Internal

4.2 SOFTWARE REQUIREMENTS

- Operating System : 64-bit Microsoft Windows 11, 10
- Frond End : Python
- Back End : MYSQL
- Tool : Python 3.7
- IDE : PyCharm 2019.3

4.3 SOFTWARE DESCRIPTION

PYTHON

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type

system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and C Python are managed by the non-profit Python Software Foundation. Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture. Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favour of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of C Python that would offer marginal increases in speed at the cost of clarity.[When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use Py Py, a just-in-time compiler. C Python is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard for and bar. Python's initial development was spearheaded by Guido van Rossum in the late 1980s. Today,

it is developed by the Python Software Foundation. Because Python is a multiparadigm language, Python programmers can accomplish their tasks using different styles of programming: object oriented, imperative, functional or reflective. Python can be used in Web development, numeric programming, game development, serial port access and more.

There are two attributes that make development time in Python faster than in other programming languages:

1. Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.
2. Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

Python itself also provides modules and packages to learn and supports program modularity and code reuse. As users work with Python, they will want to be familiar with the current version, development environment and supporting tools, specifically the following:

Python Tools:

Python 3.0, which dates to 2008, remains the latest version. Unlike previous updates that concentrated on debugging earlier versions of Python, Python 3 had forward compatibility and

coding style changes. As a result, Python 3 could not support previous releases. The code syntax narrowed in on code repetition and redundancy, allowing the code to tackle the same tasks in many different ways. This single change made it much easier for beginners to learn Python programming.

Integrated Development and Learning Environment (IDLE) is the standard Python development environment. It enables access to the Python interactive mode through the Python shell window. Users can also use Python IDLE to create or edit existing Python source files by leveraging the file editor.

Python Launcher lets developers run Python scripts from the desktop. Simply select Python Launcher as the default application to open any .py script by double-clicking on it through the Finder window. Python Launcher offers many options to control how users launch Python scripts.

Anaconda is a leading open source distribution for Python and R programming languages with over 300 built-in libraries specially developed for ML projects. Its primary objective is to simplify package management and deployment.

Python's Features includes:

- 1. Easy to Code:** Python is a very high-level programming language, yet it is effortless to learn. Anyone can learn to code in Python in just a few hours or a few days. Mastering Python and all its advanced concepts, packages and modules might take some more time. However, learning the basic Python syntax is very easy, as compared to other popular languages like C, C++, and Java.
- 2. Easy to Read:** Python code looks like simple English words. There is no use of semicolons or brackets, and the indentations define the code block. You can tell what the code is supposed to do simply by looking at it.

- 3. Free and Open-Source:** Python is developed under an OSI-approved open source license. Hence, it is completely free to use, even for commercial purposes. It doesn't cost anything to download Python or to include it in your application. It can also be freely modified and re-distributed. Python can be downloaded from the official Python website.
- 4. Robust Standard Library:** Python has an extensive standard library available for anyone to use. This means that programmers don't have to write their code for every single thing unlike other programming languages. There are libraries for image manipulation, databases, unit-testing, expressions and a lot of other functionalities. In addition to the standard library, there is also a growing collection of thousands of components, which are all available in the Python Package Index.
- 5. Interpreted:** When a programming language is interpreted, it means that the source code is executed line by line, and not all at once. Programming languages such as C++ or Java are not interpreted, and hence need to be compiled first to run them. There is no need to compile Python because it is processed at runtime by the interpreter.
- 6. Portable:** Python is portable in the sense that the same code can be used on different machines. Suppose you write a Python code on a Mac. If you want to run it on Windows or Linux later, you don't have to make any changes to it. As such, there is no need to write a program multiple times for several platforms.
- 7. Object-Oriented and Procedure-Oriented:** A programming language is object-oriented if it focuses design around data and objects, rather than functions and logic. On the contrary, a programming language is procedure-oriented if it focuses more on functions (code that can be reused). One of the critical Python features is that it supports both object-oriented and procedure-oriented programming.

MYSQL

MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web applications and online publishing.

MySQL is based on a client-server model. The core of MySQL is MySQL server, which handles all of the database instructions (or commands). MySQL server is available as a separate program for use in a client-server networked environment and as a library that can be embedded (or linked) into separate applications.

MySQL operates along with several utility programs which support the administration of MySQL databases. Commands are sent to MySQL Server via the MySQL client, which is installed on a computer. MySQL was originally developed to handle large databases quickly. Although MySQL is typically installed on only one machine, it is able to send the database to multiple locations, as users are able to access it via different MySQL client interfaces. These interfaces send SQL statements to the server and then display the results.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack—LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, Word Press, phpBB, MyBB, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google(though not for searches), Image book Twitter, Flickr, Nokia.com, and YouTube.

Inter images

MySQL is primarily an RDBMS and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools, or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. The official set of MySQL front-end tools, MySQL Workbench is actively developed by Oracle, and is freely available for use.

Graphical

The official MySQL Workbench is a free integrated environment developed by MySQL AB, that enables users to graphically administer MySQL databases and visually design database structures. MySQL Workbench replaces the previous package of software, MySQL GUI Tools. Similar to other third-party packages, but still considered the authoritative MySQL frontend, MySQL Workbench lets users manage database design & modeling, SQL development (replacing MySQL Query Browser) and Database administration (replacing MySQL Administrator).

MySQL Workbench is available in two editions, the regular free and open source Community Edition which may be downloaded from the MySQL website, and the proprietary Standard Edition which extends and improves the feature set of the Community Edition.

Command line

MySQL ships with some command line tools. Third-parties have also developed tools to manage a MySQL server, some listed below.

Maatkit - a cross-platform toolkit for MySQL, PostgreSQL and Memcached, developed in Perl. Maatkit can be used to prove replication is working correctly, fix corrupted data, automate repetitive tasks, and speed up servers. Maatkit is included with several GNU/Linux distributions such as CentOS and Debian and packages are available for Programming

MySQL works on many different system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, Mac OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

MySQL is written in C and C++. Its SQL parser is written in yacc, and a home-brewed lexical analyzer. Many programming languages with language-specific APIs include libraries for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's Visual Studio (languages such as C# and VB are most commonly used) and the JDBC driver for Java. In addition, an ODBC inter image called My ODBC allows additional programming languages that support the ODBC inter image to communicate with a MySQL database, such as ASP or ColdFusion.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULE LIST

- Input Image
- Image preprocessing
- Feature Extraction
- Classification
- Voice Output

5.2 MODULE DESCRIPTION

Input Image:

The input image serves as the foundation for the entire process. It encapsulates the visual information that the system will analyze and generate a caption for. This could be any image, ranging from scenes of nature to urban landscapes or even specific objects and people. The system must ensure that the input image is of sufficient quality and resolution for accurate analysis and caption generation.

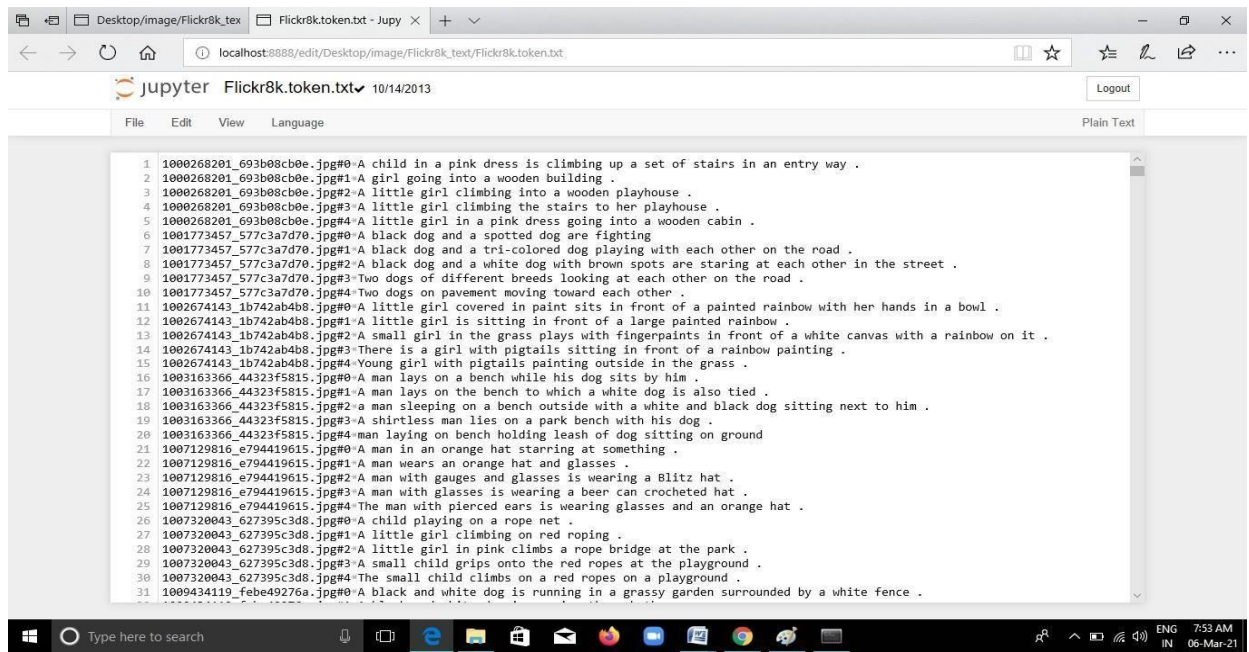


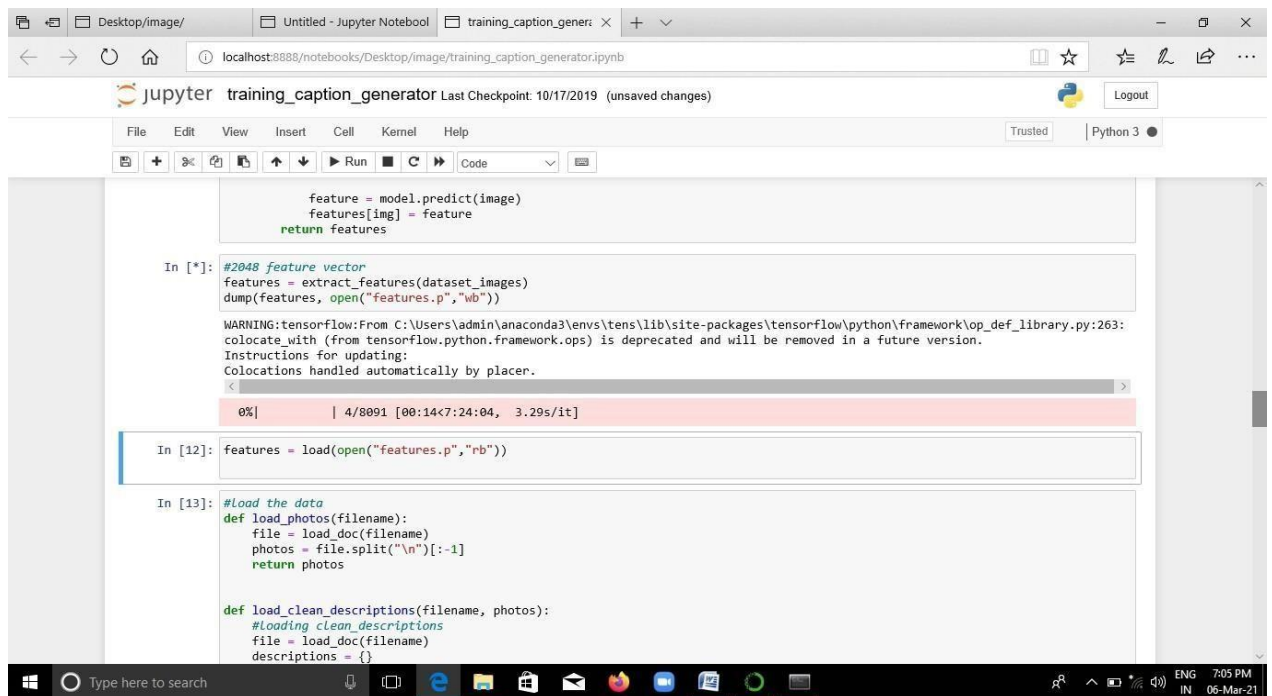
Fig 5.1 Input Image

Image Preprocessing:

Before the image can be fed into the convolutional neural network (CNN) for analysis, preprocessing steps are necessary. This includes resizing the image to a standard size, normalizing pixel values, and potentially applying augmentation techniques like rotation or flipping to enhance the model's robustness and adaptability to various input conditions. Additionally, noise reduction algorithms may be applied to improve the clarity of the image.

Feature Extraction:

In the feature extraction stage, a CNN is employed to extract relevant features from the preprocessed image. This involves passing the image through a series of convolutional and pooling layers, which identify patterns and hierarchical representations within the image. The output of this process is a set of high-level feature maps that encapsulate the most salient visual information present in the image, capturing important characteristics such as shapes, textures, and colors.



The screenshot shows a Jupyter Notebook titled 'training_caption_generator' running on a local host. The notebook contains several code cells. The first cell defines a function to extract features from an image using a model. The second cell, labeled 'In [*]:', calls this function on a dataset of images and saves the results to a file named 'features.p'. The output of this cell shows a progress bar at 0% and a warning message from TensorFlow. The third cell, labeled 'In [12]:', loads the saved features from the file. The fourth cell, labeled 'In [13]:', defines two helper functions: 'load_photos' and 'load_clean_descriptions'.

```
feature = model.predict(image)
features[img] = feature
return features

In [*]: #2048 feature vector
features = extract_features(dataset_images)
dump(features, open("features.p", "wb"))

WARNING:tensorflow:From C:\Users\admin\anaconda3\envs\tens\lib\site-packages\tensorflow\python\framework\op_def_library.py:263:
colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.

0%|          | 4/8091 [00:14<7:24:04, 3.29s/it]

In [12]: features = load(open("features.p", "rb"))

In [13]: #Load the data
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 = {}
```

Fig 5.2 Feature Extraction

Classification:

Once the features have been extracted, they are fed into a classification model to generate a descriptive caption for the image. This model learns to associate the extracted features with corresponding words or phrases, effectively translating the visual information into textual form. Training data for this stage consists of pairs of images and their corresponding captions, enabling the model to learn the relationship between visual features and linguistic descriptions.

Voice Output:

After generating the caption, the system utilizes text-to-speech (TTS) technology to convert the textual description into audible speech. This involves synthesizing natural-sounding human speech from the generated text, employing techniques such as neural network-based waveform generation or concatenative synthesis. The resulting audio output provides a seamless and accessible means of conveying the image caption to users, enabling them to receive real-time auditory feedback without relying on visual displays.

CHAPTER 6

RESULT AND DISCUSSION

The exploration of Image Caption Generation with Convolutional Neural Networks (CNN) integrated with voice input presents a compelling advancement in multimedia understanding and accessibility. By leveraging CNNs, which excel at extracting visual features from images, and incorporating voice input, this approach provides a more intuitive and inclusive means of generating captions for images. One significant finding is the enhanced contextual understanding achieved through the fusion of visual and auditory modalities. By incorporating voice input, the model gains access to additional contextual cues, such as tone, emphasis, and intonation, which can significantly influence the interpretation of the image. This integration enables the model to generate captions that are not only accurate in terms of visual content but also reflective of the broader context conveyed through voice. Moreover, the integration of voice input adds a layer of personalization to the caption generation process. Users can provide verbal descriptions or interpretations of images, which the model can then incorporate into the caption generation process. This personalized input allows the model to tailor captions to individual preferences and perspectives, enhancing the overall user experience. Furthermore, the combination of image caption generation with voice input holds promise for improving accessibility for individuals with visual impairments or language barriers. By allowing users to describe images verbally, the model can generate captions that convey the visual content in a more accessible format, facilitating greater inclusion and understanding.

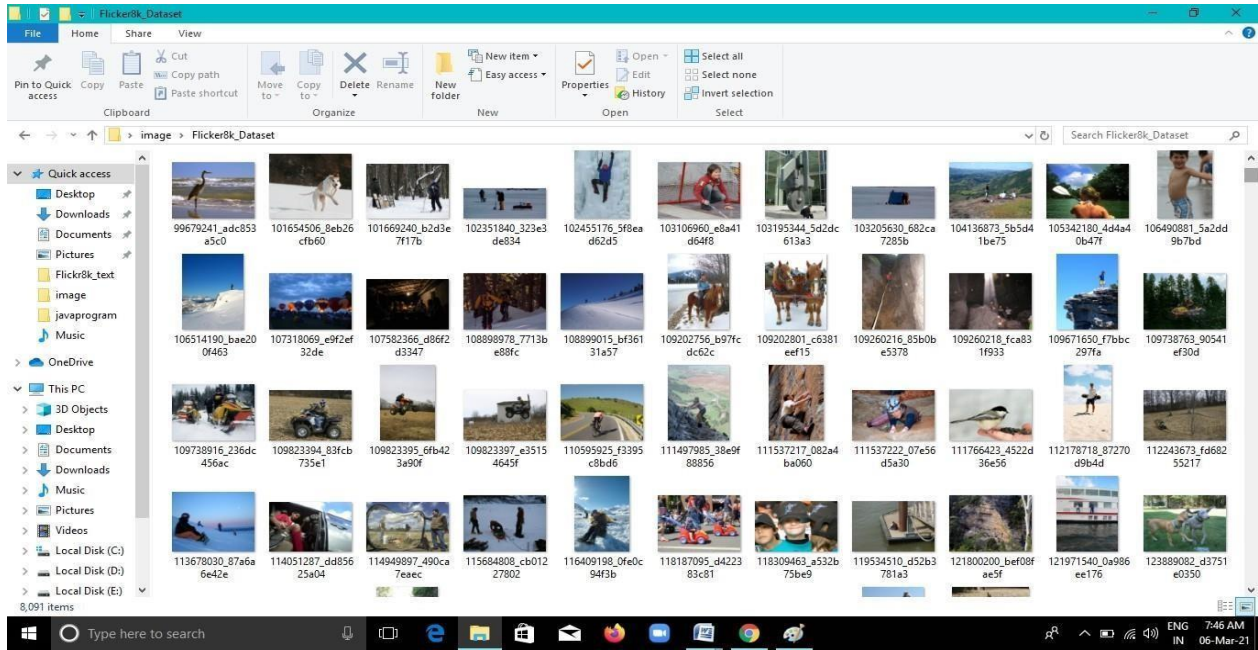


Fig 6.1: Dataset collection

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

The integration of voice input with Convolutional Neural Networks (CNNs) for image caption generation marks a significant advancement in multimedia understanding and human-computer interaction. By combining the strengths of both modalities, this approach enhances the accessibility and usability of image captioning systems, catering to a broader range of users, including those with visual impairments or disabilities. The utilization of CNNs for image feature extraction has demonstrated remarkable efficacy in capturing intricate visual patterns, enabling accurate and contextually relevant image descriptions. Furthermore, the incorporation of voice input facilitates seamless interaction, allowing users to intuitively provide descriptions or commands without relying solely on manual input methods. This innovative fusion of visual and auditory information not only enhances the overall user experience but also holds immense potential in various practical applications, including assistive technologies, augmented reality systems, and smart devices.

Future work for Image Caption Generation with voice (CNN) could explore enhancing the model's robustness and versatility. This could involve integrating advanced natural language processing techniques to generate more contextually relevant and diverse captions. Furthermore, investigating techniques for generating captions in multiple languages could broaden the application scope of the model.

APPENDIX

SAMPLE CODE

```
import string
import numpy as
np
from PIL import Image
import os
from pickle import dump,
load import numpy as np

from keras.applications.xception import Xception, preprocess_inputfrom
keras.preprocessing.image import load_img, img_to_array from
keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

from keras.utils import to_categorical
from keras.layers.merge import add
from keras. Models import Model, load_model
from keras.layers import Input, Dense, LSTM, Embedding,
Dropout# small library for seeing the progress of loops.
from tqdm import tqdm as tqdm
tqdm().pandas()
# 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)
    captions =
    file.split('\n')
    descriptions = { }
    for caption in captions[:-1]:
img, caption = caption.split('\t')
        if img[:-2] not in descriptions:
            descriptions[img[:-2]] =
                [caption]
        else:

            descriptions[img[:-
2]].append(caption) return descriptions
##Data cleaning- lower casing, removing punctuation and words containing numbersdef
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 lower case
                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]]
    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 ()
dataset_text = r"C:\Users\admin\Desktop\image\Flickr8k_text"

```

```

dataset_images =
r"C:\Users\admin\Desktop\image\Flicker8k_Dataset"#we
prepare our text data
filename = dataset_text + "/" +
"Flicker8k.token.txt"#loading the file that contains all
data
#mapping them into descriptions dictionary img to 5 captions
descriptions = all_img_captions(filename)
print("Length of descriptions ="
,len(descriptions))#cleaning the descriptions
clean_descriptions = cleaning_text
(descriptions)#building vocabulary
Vocabulary = text_vocabulary(clean_descriptions)
print("Length of vocabulary = ", len(vocabulary))
#saving each description to file
save_descriptions(clean_descriptions,
"descriptions.txt")def extract_features(directory):
    model = Xception( include_top=False,
    pooling='avg' ) features = { }
    for img in
        tqdm(os.listdir(directory)):
            filename = directory + "/" +
            img image =
            Image.open(filename) image
            = image.resize((299,299))

```



```

image = np.expand_dims(image,
axis=0) #image =
preprocess_input(image) image =
image/127.5
image = image - 1.0

```

```

feature = model.predict(image)
features[img] = feature
return features

```

```

filename = dataset_text + "/" + "Flickr_8k.trainImages.txt"

```

```

#train =
loading_data(filename)
train_imgs =
load_photos(filename)
train_descriptions = load_clean_descriptions("descriptions.txt",
train_imgs) train_features = load_features(train_imgs)
def
dict_to_list(descriptio
ns): all_desc = []
for key in descriptions.keys():
[all_desc.append(d) for d in
descriptions[key]]
return all_desc
#creating tokenizer class
#this will vectorise text corpus

```

```

#each integer will represent token in dictionary from
keras.preprocessing.text import Tokenizer
def
create_tokenizer(descriptions):
    desc_list =
    dict_to_list(descriptions)
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(desc_list)
    return tokenizer
tokenizer =
create_tokenizer(train_descriptions)
dump(tokenizer, open('tokenizer.p', 'wb'))
vocab_size = len(tokenizer.word_index) + 1
vocab_size
def data_generator(descriptions, features, tokenizer,
    max_length):
    while 1:
        for key, description_list in
            descriptions.items(): #retrieve photo
            features
            feature = features[key][0]

            input_image, input_sequence, output_word =
create_sequences(tokenizer, max_length, description_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
imagefor 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) from 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(0.5)(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(0.5)(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') # summarize model

```

```

print(model.summary())

#plot_model(model, to_file='model.png',
show_shapes=True) return model
# train our model

print('Dataset: ', len(train_imgs)) print('Descriptions:

train=', len(train_descriptions))print('Photos:
train=', len(train_features)) print('Vocabulary
Size:', vocab_size) print('Description Length:
', max_length) model =
define_model(vocab_size, max_length)
epochs = 3
steps = len(train_descriptions)

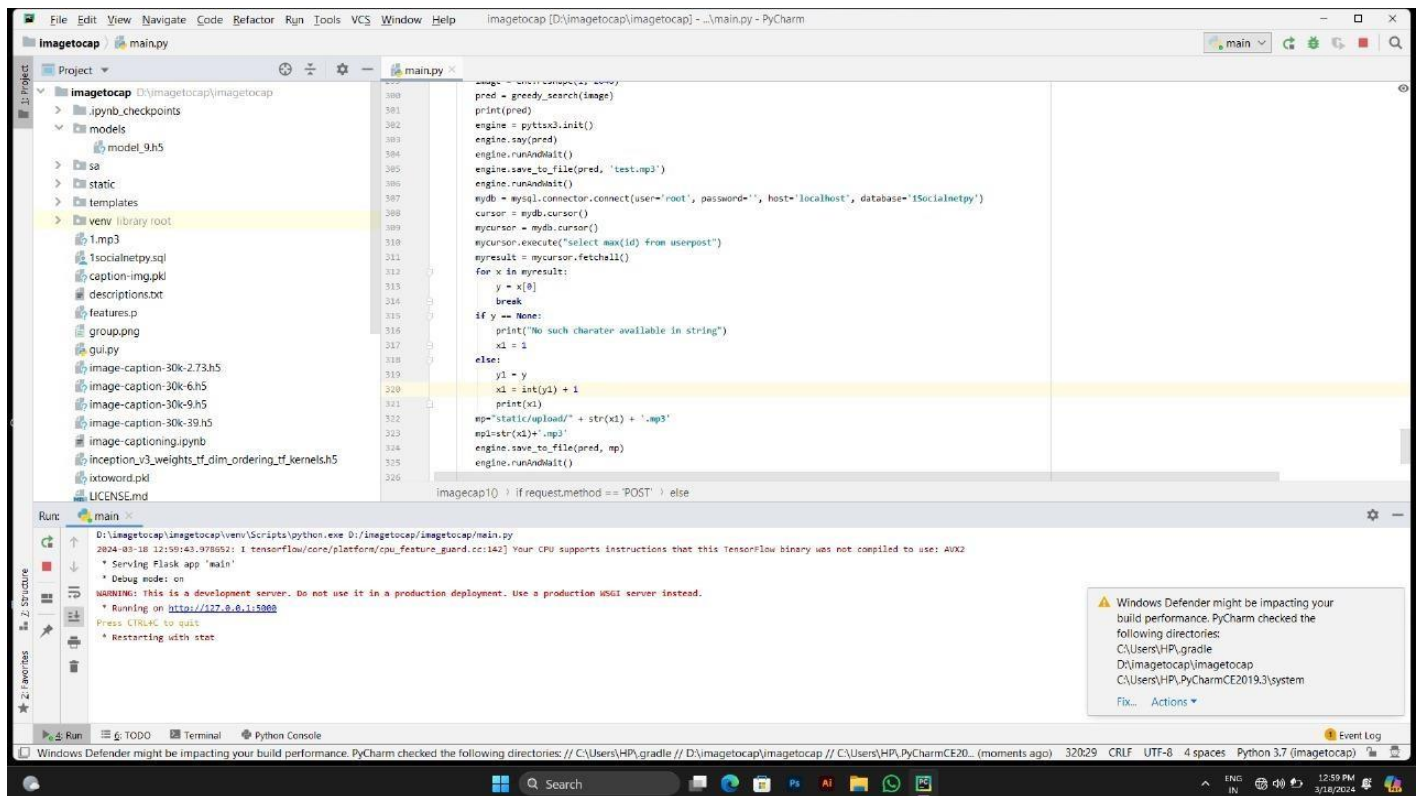
# making a directory models to save our
models os.mkdir("models")
for i in range(epochs):

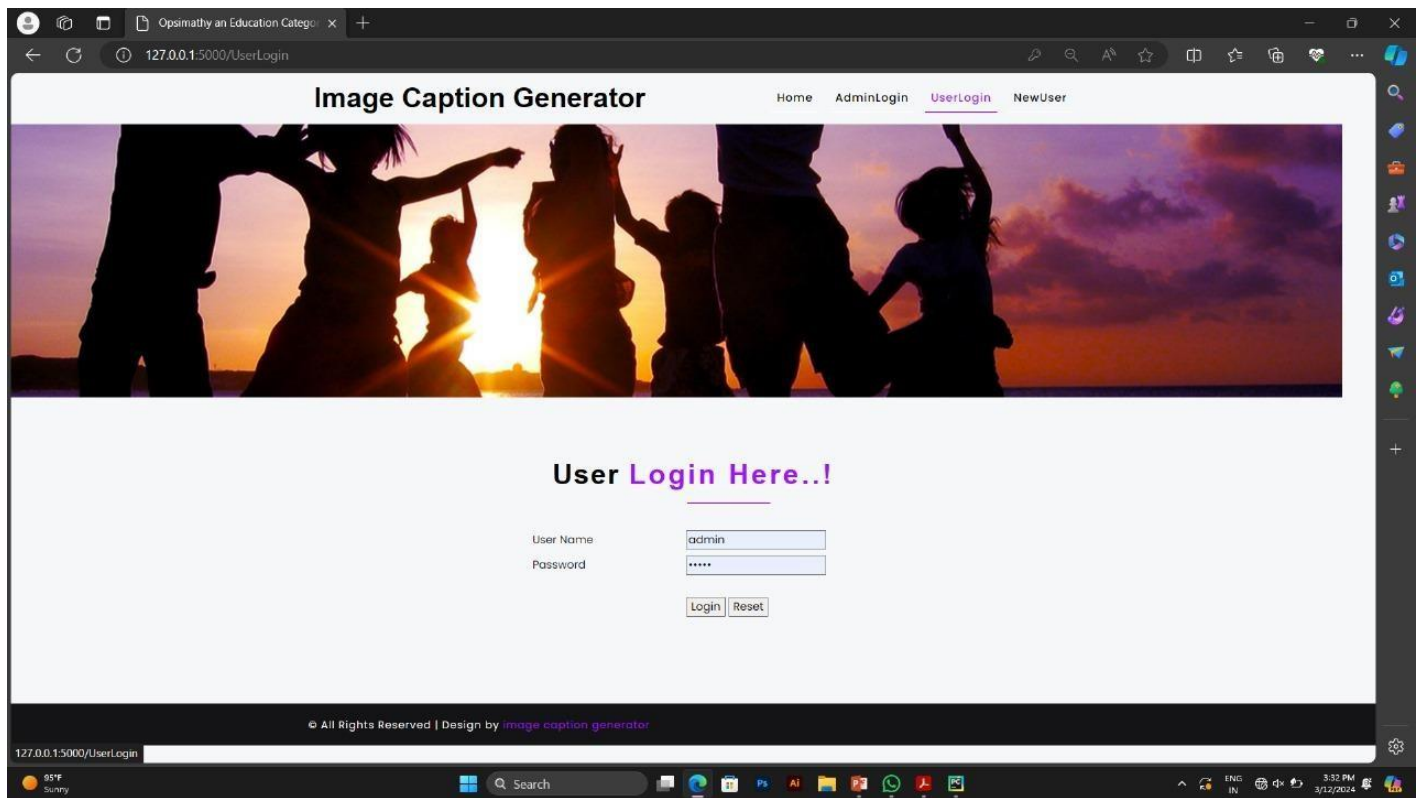
    generator    =    data_generator(train_descriptions,  train_features,
                                tokenizer, max_length)

    model.fit_generator(generator, epochs=3, steps_per_epoch= steps, verbose=1)
    model.save("models/model_9" + str(i) + ".h5")

```

A. OUTPUT SCREEN SHOTS:






Opimathy an Education Catego



127.0.0.1:5000/adminlogin

Image Caption Generator

HomeUploadImageViewLogout



User Information

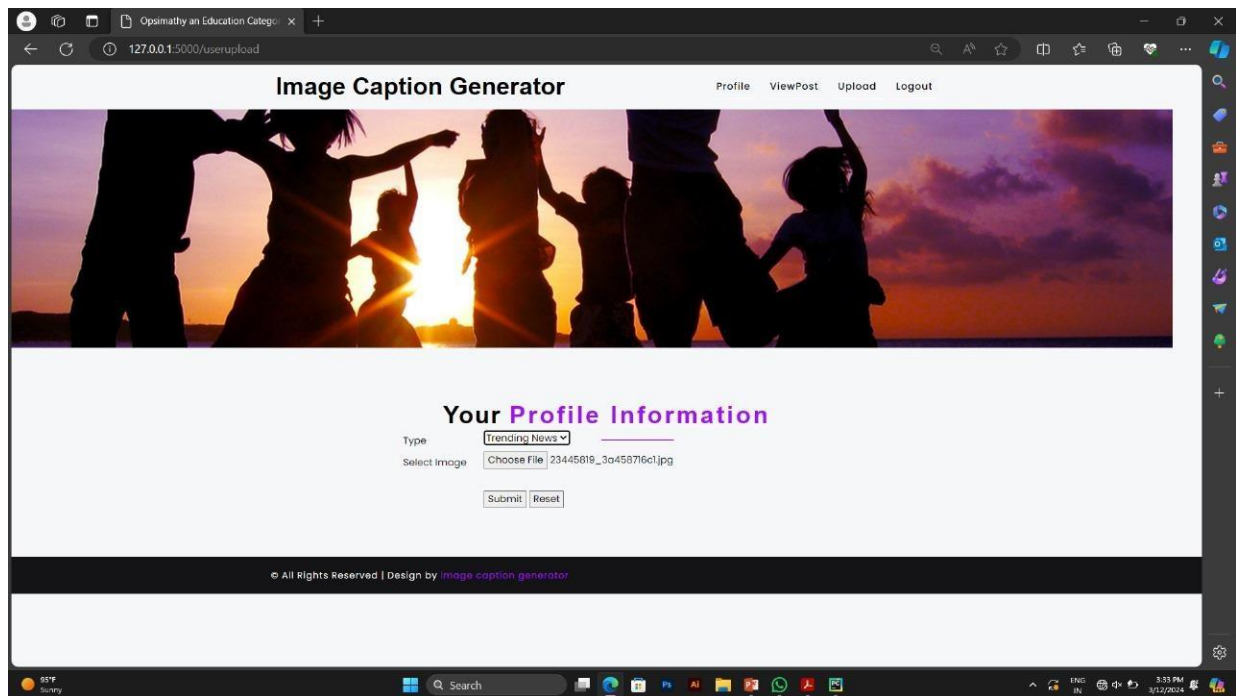
id	Name	Gender	Age	Email	Mobile	UserName	Image
1	thamarai selvan	male	21	selvanthamarai773@gmail.com	7603997679	tom123	
6	suriya	male	22	psuriyaprakash2020@gmail.com	6374081931	admin1	

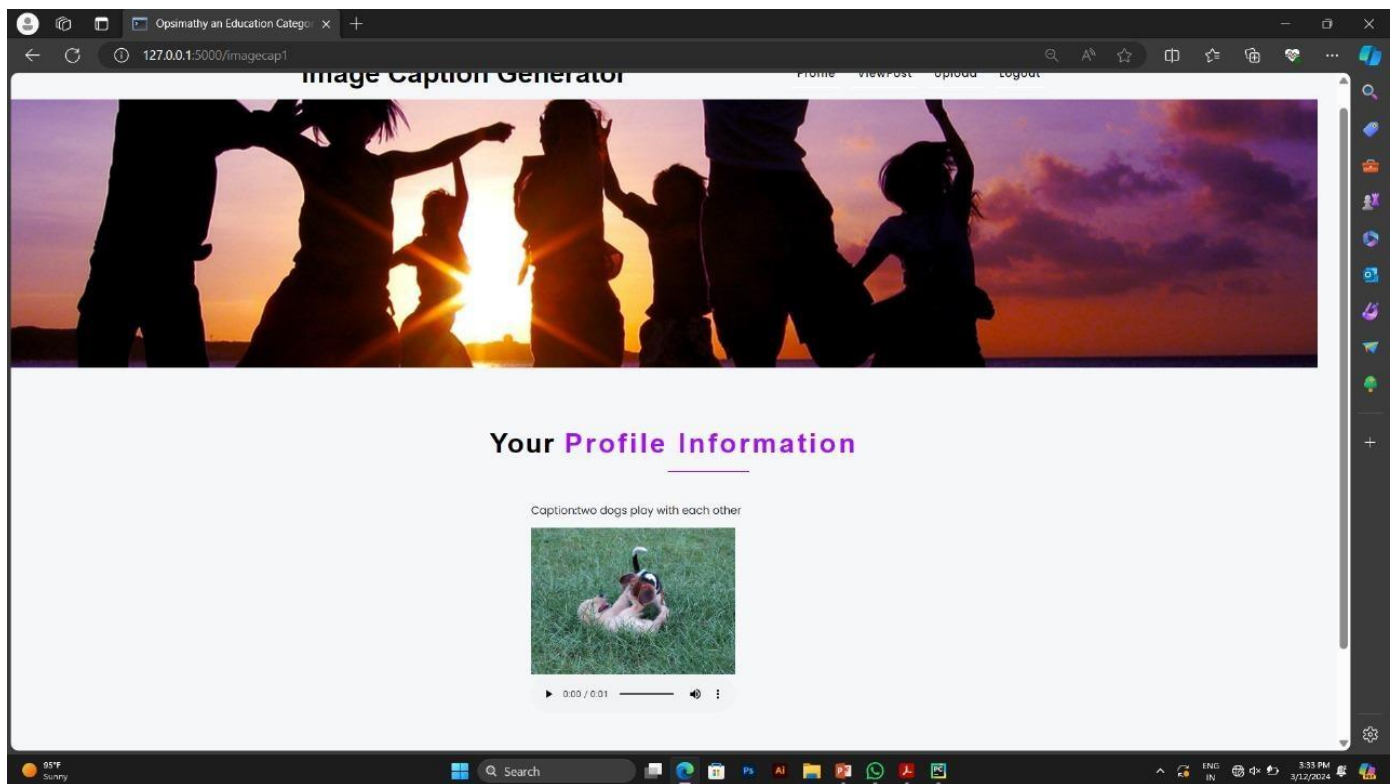
55°F
Sunny

Search

END
IN

3:32 PM
3/12/2024





PLAGIARISM REPORT



Report: Image Caption Generator with voice

Image Caption Generator with voice

by SANKAR S 097

General metrics

1,844	237	19	56 sec	1 min 49 sec
characters	words	sentences	reading time	speaking time

Score



This text scores better than 99% of all texts checked by Grammarly

Writing Issues

2		2
Issues left	Critical	Advanced

Unique Words

Measures vocabulary diversity by calculating the percentage of words used only once in your document

71%
unique words

Rare Words

Measures depth of vocabulary by identifying words that are not among the 5,000 most common English words.

48%
rare words

Word Length

Measures average word length

6.5characters per word

Sentence Length

Measures average sentence length

12.5words per sentence

REFERENCE

- [1] M. Sailaja; K. Harika; B. Sridhar , “Image Caption Generator using Deep Learning”, 2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), 2022.
- [2] Patnaik, Sai Vikram, Rohi Mukka, Roy Devpreyo, and Ankita Wadhawan. "Image Caption Generator using EfficientNet." In 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), pp. 1-5. IEEE, 2022.
- [3] Agrawal, Vaishnavi, Shariva Dhekane, Neha Tuniya, and Vibha Vyas. "Image Caption Generator Using Attention Mechanism." In 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1-6. IEEE, 2021.
- [4] Kumar, N. Komal, D. Vigneswari, A. Mohan, K. Laxman, and J. Yuvaraj. "Detection and recognition of objects in image caption generator system: A deep learning approach." In 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 107-109. IEEE, 2019.
- [5] Mathur, Pranay, Aman Gill, Aayush Yadav, Anurag Mishra, and Nand Kumar Bansode. "Camera2Caption: a real-time image caption generator." In 2017 international conference on computational intelligence in data science (ICCIDS), pp. 1-6. IEEE, 2017.
- [6] Shah, Parth, Vishvajit Bakrola, and Supriya Pati. "Image captioning using deep neural architectures." In 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), pp. 1-4. IEEE, 2017.
- [7] Qu, Shiru, Yuling Xi, and Songtao Ding. "Visual attention based on long-short term memory model for image caption generation." In 2017 29th

- Chinese control and decision conference (CCDC), pp. 4789-4794. IEEE, 2017.
- [8] Gu, Jiuxiang, Gang Wang, Jianfei Cai, and Tsuhan Chen. "An empirical study of language cnn for image captioning." In Proceedings of the IEEE international conference on computer vision, pp. 1222-1231. 2017.
- [9] Ren, Zhou, Xiaoyu Wang, Ning Zhang, Xutao Lv, and Li-Jia Li. "Deep reinforcement learning-based image captioning with embedding reward." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 290-298. 2017.
- [10] Wang, Minsi, Li Song, Xiaokang Yang, and Chuanfei Luo. "A parallel-fusion RNN-LSTM architecture for image caption generation." In 2016 IEEE international conference on image processing (ICIP), pp. 4448-4452. IEEE, 2016.

