IMAGE CAPTION GENERATOR USING CNN AND LSTM

A Project Report

Submitted by:

NIGAMANANDA PANDA (2141016090)
AKASHDEEP BEHERA (2141002072)
GANESH CHANDRA DAS (2141004078)
MADHAB NARAYAN MOHANTY (2141013300)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Faculty of Engineering and Technology, Institute of Technical Education and Research

SIKSHA 'O' ANUSANDHAN (DEEMED TO BE UNIVERSITY)

Bhubaneswar, Odisha, India

(June 2025)

CERTIFICATE

This is to certify that the project report titled “IMAGE CAPTION GENERATOR USING CNN AND LSTM” being submitted by **Nigamananda Panda, Akashdeep Behera, Ganesh Chandra Das, Madhab Narayan Mohanty** of section CSE-Z to the Institute of Technical Education and Research, Siksha ‘O’ Anusandhan (Deemed to be University), Bhubaneswar for the partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering is a record of original confide work carried out by them under my/our supervision and guidance. The project work, in my/our opinion, has reached the requisite standard fulfilling the requirements for the degree of Bachelor of Technology.

The results contained in this project work have not been submitted in part or full to any other University or Institute for the award of any degree or diploma.

Dr. Rangaballav Pradhan

Department of Computer Science and Engineering

Faculty of Engineering and Technology;
Institute of Technical Education and Research;
Siksha ‘O’ Anusandhan (Deemed to be University)

ACKNOWLEDGEMENT

We would like to express my sincere gratitude to our project supervisor, Dr. Rangaballav Pradhan, for his exceptional guidance, mentorship, and unwavering support throughout the duration of our group project. His expertise, insights, and valuable feedback have been pivotal in shaping our project and ensuring its successful completion. We are grateful for his patience, dedication, and commitment to our learning and growth.

Furthermore, we would like to extend our heartfelt appreciation Siksha 'O' Anusandhan (Deemed to be University) for providing us with the necessary facilities, resources, and infrastructure that have been crucial to the progress and achievement of our group project. The university's commitment to fostering an environment of academic excellence and research has provided us with the platform to explore, innovate, and collaborate effectively.

We would also like to acknowledge and thank our fellow group members for their hard work, collaboration, and contribution to the project. Their diverse skills, perspectives, and dedication have played a significant role in the overall success of our project. Our collective efforts and teamwork have allowed us to overcome challenges and achieve our objectives.

Lastly, we are thankful to all the individuals and institutions who have contributed to the realization of our group project, whether through discussions, feedback, or other forms of support. Your involvement and encouragement have played a crucial role in our growth and the successful completion of our project.

We extend our heartfelt appreciation to everyone involved, and we are grateful for the opportunity to work on this project within such a supportive and enriching academic environment.

Nigamananda Panda

Akashdeep Behera

Place: Bhubaneswar

Ganesh Chandra Das

Madhab Narayan Mohanty

Date: 16/06/2025

Signature of Students

DECLARATION

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

Nigamananda Panda	(2141016090)
Akashdeep Behera	(2141002072)
Ganesh Chandra Das	(2141004078)
Madhab Narayan Mohanty	(2141013300)

Signature of Students with Registration Numbers

Date: 16/06/2025

REPORT APPROVAL

This project report titled “IMAGE CAPTION GENERATOR USING CNN AND LSTM” submitted by Nigamananda Panda, Akashdeep Behera, Ganesh Chandra Das, Madhab Narayan Mohanty is approved for the degree of *Bachelor of Technology in Computer Science and Engineering*.

Examiner(s)

____

____

____

Supervisor

____

Project Coordinator

____

PREFACE

With the exponential growth of digital images on the internet, efficiently generating accurate and meaningful descriptions has become a significant challenge. Manually captioning images is impractical due to the vast number of images uploaded daily. The lack of proper captions affects image searchability, accessibility for visually impaired users, and content recommendations on social media platforms. The key challenge lies in training models to accurately interpret image context and generate human-like captions that effectively convey essential features. Developing an advanced image captioning model will contribute to a more interactive, accessible, and user-friendly digital experience. This project presents the design and implementation of an Image Caption Generator using a hybrid deep learning architecture that combines Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. The system leverages the visual understanding capabilities of CNNs and the sequential modeling power of LSTMs to generate coherent and contextually appropriate captions for a wide variety of images. For feature extraction, the DenseNet201 model—a deep and highly efficient convolutional neural network (CNN)—was employed to capture rich visual representations from input images. These visual features serve as the contextual input for the language model. The Flickr8k dataset, containing 8,000 images with five descriptive captions each, was used to train and validate the model. The diversity of this dataset enables the model to generalize well across various everyday scenes. The LSTM-based decoder network was trained to translate visual features into natural language sentences. The implementation utilized TensorFlow and Keras, providing flexibility and scalability for experimentation. Key preprocessing techniques, including tokenization, embedding layers, and sequence padding, were applied to handle textual data efficiently. Evaluation results demonstrated that the model generates grammatically correct and semantically relevant captions, effectively bridging the gap between visual content and language understanding. The project not only addresses the practical needs of enhancing accessibility and content discovery but also contributes to advancing research in vision-language integration. In conclusion, this Image Caption Generator lays a robust foundation for future enhancements, including the integration of attention mechanisms and larger-scale datasets, to further improve caption quality and applicability in real-world scenarios.

INDIVIDUAL CONTRIBUTIONS

Nigmananda Panda	Materials & methods, Model Diagram, Methods Used, Tools Used, Evaluation Measures Used.
Akashdeep Behera	Introduction, Project Overview, Motivation & Uniqueness.
Ganesh Chandra Das	Literature Survey, Existing System, Problem Outcomes.
Madhab Narayan Mohanty	Results, System Specification, Parameters Used, Experimental Outcomes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

It is becoming ever so complex to generate a perfectly meaningful caption because the number of digital images being broadcast by the Internet today is growing at an alarming rate. Every day, people upload millions of images onto websites, social media, or online repositories, and it is quite impractical and unsustainable to carry out manual annotation on such volumes. Without captions, images become hard to search for, unavailable for users with visual impairment, or lose utility to the content-based recommendation systems [1]. Such hurdles require highly perfected models that can perceive the visual content and create natural, human-like textual descriptions. Bridging the gap between visual data and language will set up systems that see and describe the content [2] including the title "Image Captioning Generator using CNN and LSTM", which seeks a deep learning-based approach that can automatically generate image-relevant captions. The work adopts Convolutional Neural Networks (CNNs) for image feature extraction and Long Short Term Memory (LSTM) networks for sequence generation. The two architectures form a system that extracts high-level visual features, and these features are transformed into grammatically correct sentences and contextually meaningful ones [3], [4]. The dataset selected for training and testing purposes is Flickr8k, which consists of 8000 real-world images individually accompanied by five human-annotated captions, which makes this dataset very well balanced for building captioning models with limited computational resources [11]. The model uses the DenseNet201 architecture for processing visual data, which has been claimed to be one of the best architectures in terms of dense connectivity, efficient reutilization of features, and maximizing gradient propagation, and overcoming vanishing gradient issues [7]. The extracted features will now be passed into an LSTM network, typically known for pattern comprehension within sequences and context preservation within language generation [4]. The project workflow will include several phases, such as image preprocessing, feature extraction based on DenseNet, tokenization of captions, and sequence generation using LSTM. Combining these methodologies thus shows a deep learning algorithm for automatic, smart, accurate, and descriptive image caption generation [1], [3].

1.1 Motivation

The Image captioning is the idea that is being inspired by the growing demand for intelligent systems to interpret and describe the various visual contents. Bridging the gap between computer vision and natural language processing creates strong possibilities for developing applications in the domains of accessibility, education, and multimedia search engines. The primary aim is to develop a model capable of understanding the image and contextualizing it by formulating a description similar to human interpretation for human-computer interface improvement. The use of DenseNet201 and Flickr8k datasets allows the practical experimentation of state-of-the-art deep learning architectures to address real-world issues in image comprehension and their automated language generation [1], [3]. The above effects will apply to well-established applications of image captioning by augmenting visually impaired assistive devices, permitting descriptions of surroundings through voice. It adds value in automatic content labeling, enhances search accuracy for images, and creates a summary of visual media. The encapsulation of complicated visual scenes in proper compositions of language indeed stretches the limits of both vision and language domains. The project focuses on solving the problems of these challenges through training highly efficient and well-performing models like DenseNet201 for visual feature extraction and LSTM for sequential text generation. Moreover, it inspires the potential association with an internationally compact but fairly diverse dataset, such as Flickr8k, making it possible to conduct educational and exploratory research with limited computational capabilities.

1.2 Uniqueness of the Project

While most typical feature extractors, like InceptionV3, have an open-ended application based on flexibility to include an additional layer to extract the output layer, consideration of this project is from the viewpoint of specific integration of the Flickr8k dataset with DenseNet201 architecture which is not as common today as what is largely used. Dense connections enable feature reuse and lead to more detailed and accurate visual representations. In addition, the dataset used is quite compact for performance and computation efficiency considerations, while also ensuring model design. Translation of visual data into meaningful captions happens through the use of a robust pipeline-a combination of DenseNet201 and LSTM. It is custom-built for small data sets instead of general models, which makes it accessible to researchers with constraints on the resources available to perform more extensive processing. In addition, the modular and scalable design will facilitate the future integration of state-of-the-art techniques such as attention mechanisms or beam search. The use of a dataset like Flickr8k promotes faster experimentation cycles for developers who want to fine-tune and optimize without the overhead costs of massive data processing. Another unique aspect of the project is its educational value—ideal for those seeking practical insights into image captioning using modern deep learning tools. The clean data preprocessing, efficient memory consumption, and interpretability are features necessary for real-world applications such as assistive technology, digital content summation, and image-based content search. This careful balance between simplicity, effectiveness, and extensibility is what sets this project apart from existing large-scale systems.

1.3 Report Layout

The report is logically structured into relevant sections and has a good flow to it. The first section consists of the introduction and provides an overview of the contextual background, motivation, and the main objective of the work. The second section provides the literature review, where previous studies are presented, the shortcomings are observed, and the rationale for their approach is explained. The third section explains their proposed model, detailing how it works from image pre-processing to caption generation, along with diagrams. The fourth section discusses the results and analysis, possible analysis from generated captions, the training graphs, and how well they performed using different metrics. The fifth and final section concludes the report, with a brief summary and future scope. Suggested improvements will include attention mechanisms and the use of larger datasets.

2. LITERATURE SURVEY

Research on image captioning has developed over time, especially focusing on models that use deep learning techniques such as CNN and LSTM. Popular early systems, for example, “Show and Tell” and “Show, Attend and Tell,” combined well with visual and language understanding. However, most of them were designed for very large datasets such as MSCOCO and required quite large amounts of computation. The traditional CNNs, like VGG16 or InceptionV3 also do not provide very efficient feature reuse. To address these issues, this project will utilize DenseNet201 for better feature propagation and use a more realistic and affordable training environment, such as the Flickr8k dataset. In this respect, the aim was to best replicate performance while improving your efficiency and accessibility.

2.1 Existing System

The area of image captioning covers research aimed at creating meaningful text descriptions for visual content. In essence, it can be defined as an effort to learn the association between images and natural languages. Most systems are based on the encoder-decoder architecture, in which a CNN extracts image features, and these features are fed into an RNN, mainly an LSTM, to generate a sentence [1], [6], [8]. One of the earliest and widely accepted models is the "Show and Tell" framework developed by Vinyals et al. The model used InceptionV3 for image encoding and LSTM for caption generation, providing quite a strong platform for works that came later in the area [1]. Following this, the "Show, Attend and Tell" model introduced by Xu et al. added an attention mechanism that enabled the model to attend to different regions of the image while generating each word in the caption. This led to the generation of sentences with more contextual awareness and fluency [2]. Another very important contribution was by Karpathy, and Fei-Fei, who proposed a Deep Visual-Semantic Alignment Model. In their framework, they aligned regions in the image with corresponding words or phrases by means of bidirectional RNNs and region-based CNN features, enhancing the alignment between objects and descriptive language [4], [8], [9]. They also proposed the Bottom-Up and Top-Down Attention Model, which was first to use object-detection methods (Faster R-CNN) for bottom-up feature extraction and then employ top-down attention at the LSTM decoder. This dual-attention paradigm was able to exert very fine control over caption generation and was the first to outperform all other models on MSCOCO [7]. In recent times, transformer-based architectures, like Oscar and VirTex, have emerged that exploit large-scale pre- training and joint vision-language embeddings for caption generation. These models assure very high accuracy, but their resource demand is also huge, making it difficult to implement them on small-scale datasets and on personal machines [8], [10], [12]. There exist obstacles despite such progress. Most of the existing systems are highly dependent on large datasets such as MSCOCO [3], which are very computationally intensive. Besides, VGG16 and InceptionV3 models do not allow the sharing of features efficiently, leading to unnecessary usage of memory spaces in deeper layers. These concerns highlight the need for solutions that are lightweight and resource-efficient. This project seeks to fill some gaps in the field by employing DenseNet201, a CNN that favors feature reuse through dense connectivity. This property would enhance learning efficiencies even for smaller datasets [5], [13]. Furthermore, the Flickr8k dataset [11] is compact yet diverse; therefore, it gives a wide range of choices in teaching and research applications with limited computing resources. Thus, the proposed system is the most balanced and practical alternative to traditional resource-consuming image captioning models.

2.2 Problem Identification

The burgeoning demand for image interpretation and description systems is owing to an increased surge of visual content on digital platforms. Every day, billions of images are uploaded onto the Internet, with the vast majority lacking any meaningful description or metadata. This lack of annotation creates a barrier to access for the blind and visually impaired, while evidence suggests that it also degrades the efficacy of image search engines and content-based recommendation engines. Manually captioning this enormous amount of data is impractical and unscalable [14]. Although deep learning has led to fairly substantial advancements in image captioning, the existing models are mostly heavy and complex. While models like VGG16 and InceptionV3 have better accuracy, they are not very well suited in terms of reusing features, which directly translates into a great amount of computation and expensive training time. The other downside is that commonly used datasets like MSCOCO also demand a lot of hardware resources, restricting students or researchers with limitedly configured setups from experimenting with these models.

The other major issue is a lack of generalization; many models perform well on data they are familiar with but flop badly when confronted with new or very diverse contexts. Furthermore, they often generate repetitive or generic captions that lack contextualization. The present work addresses these issues with an image captioning model based on the DenseNet201 architecture, which promotes feature reuse and lowers overall training complexity. We adopt the Flickr8k dataset: smaller, but diverse enough in its set of images for training purposes, and suitable for use without having high-end hardware. It intends to bring better efficiency, scalability, and performance from the grassroots up, optimizing the solution for achieving accurate image captions for real-world problems in a more accessible manner.

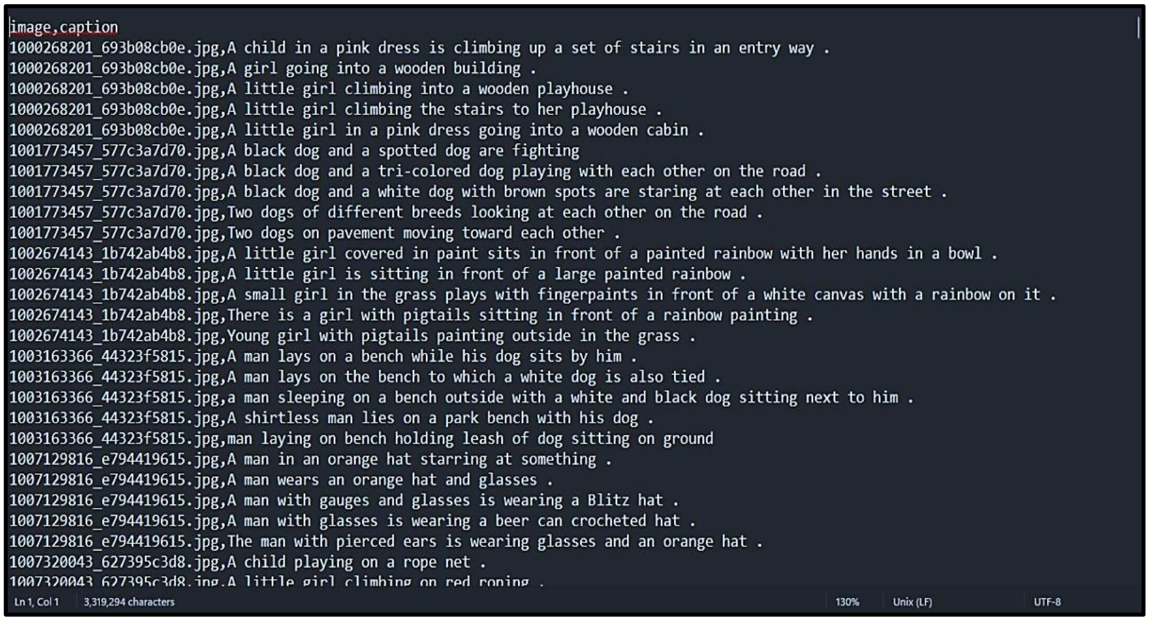
3. MATERIALS AND METHODS

In the materials and methods, we give a brief description of the datasets used, summarizing their features. In addition, a schematic diagram or model layout is provided for clarification of the system or model's structure. The project methodology is briefly highlighted, with a focus on key algorithms/techniques used. The technology stack is presented, as well as the tools/software used during the project. Evaluation metrics/criteria deployed to measure the effectiveness of the solution proposed by the project are also presented.

3.1 Dataset Description

In this project, the very commonplace Flickr8k dataset [11] was selected to provide the main source of image-caption pairs. The dataset contains 8,000 images collected from Flickr, with five different captions written by humans for each image describing its various aspects. The difference in captions provides multiple linguistic avenues to the same visual content, which is valuable for the training of robust image captioning models. Since the dataset is publicly available via Kaggle, it can be readily utilized for research and educational purposes [11]. Most of the images in Flickr8k represent various mundane scenes, including a fair share of people, animals, objects, and outdoor settings, thus providing a rich variety of visual contexts. Each image caption, therefore, describes not only the objects present but also their interactions and activities, enabling the model to learn complex relationships between visual elements and natural-language descriptions. Compared to large datasets such as MSCOCO, with its past-the-120-k boundaries of images, Flickr8k [11] is kind of small. However, it lends itself well to be used for projects possessing medium computational resources. A smaller dataset allows for faster experimentation and iteration through model implementation without compromising the learning of meaningful visual- linguistic mappings. As such, a smaller dataset helps to keep attention toward honing the model architecture or training strategy versus performing a large data preprocessing setup.

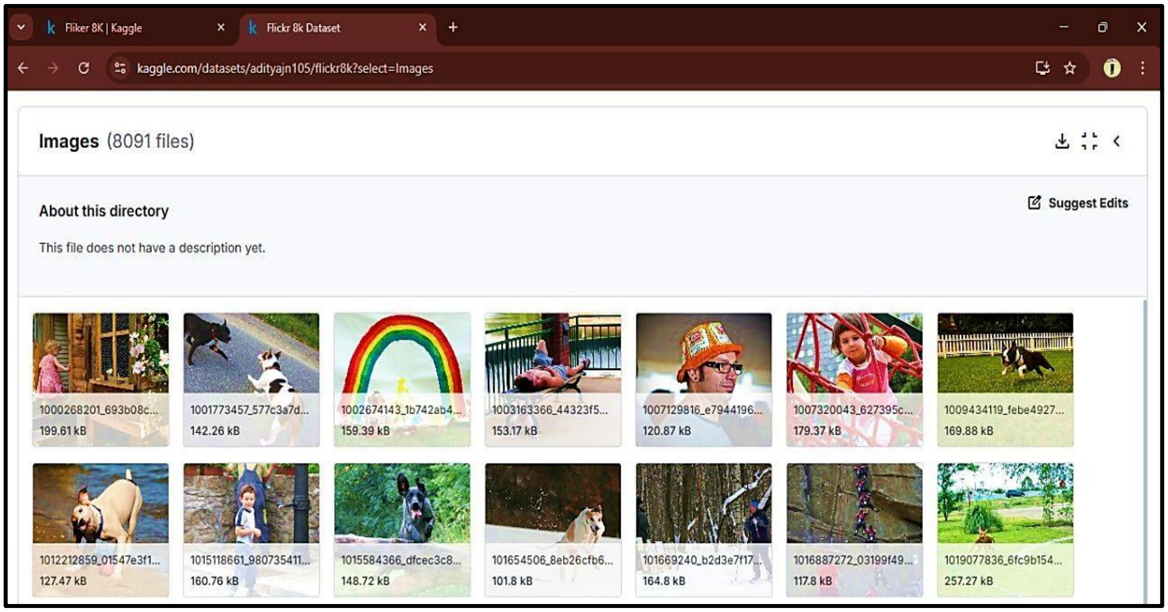
The dataset contains image files and a text file that contains captions. Cleaning up captions, tokenizing text, and normalizing images to a common format is what preprocessing consists of. Captions are converted to sequences of word indices for training the language model. On the other hand, a pre-trained CNN, in this case, DenseNet201, takes in images and extracts feature vectors, which are then fed as input to the language model. To summarize, the Flickr8k dataset [11] provides such a balanced dataset that is easy to work with for the design and competition of image captioning systems, especially when used in conjunction with powerful deep learning models.



```
image,caption
1000268201_693b08cb0e.jpg,A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg,A girl going into a wooden building .
1000268201_693b08cb0e.jpg,A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg,A little girl climbing the stairs to her playhouse .
1000268201_693b08cb0e.jpg,A little girl in a pink dress going into a wooden cabin .
1001773457_577c3a7d70.jpg,A black dog and a spotted dog are fighting
1001773457_577c3a7d70.jpg,A black dog and a tri-colored dog playing with each other on the road .
1001773457_577c3a7d70.jpg,A black dog and a white dog with brown spots are staring at each other in the street .
1001773457_577c3a7d70.jpg,Two dogs of different breeds looking at each other on the road .
1001773457_577c3a7d70.jpg,Two dogs on pavement moving toward each other .
1002674143_1b742ab4b8.jpg,A little girl covered in paint sits in front of a painted rainbow with her hands in a bowl .
1002674143_1b742ab4b8.jpg,A little girl is sitting in front of a large painted rainbow .
1002674143_1b742ab4b8.jpg,A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it .
1002674143_1b742ab4b8.jpg,There is a girl with pigtails sitting in front of a rainbow painting .
1002674143_1b742ab4b8.jpg,Young girl with pigtails painting outside in the grass .
1003163366_44323f5815.jpg,A man lays on a bench while his dog sits by him .
1003163366_44323f5815.jpg,A man lays on the bench to which a white dog is also tied .
1003163366_44323f5815.jpg,a man sleeping on a bench outside with a white and black dog sitting next to him .
1003163366_44323f5815.jpg,A shirtless man lies on a park bench with his dog .
1003163366_44323f5815.jpg,man laying on bench holding leash of dog sitting on ground
1007129816_e794419615.jpg,A man in an orange hat starring at something .
1007129816_e794419615.jpg,A man wears an orange hat and glasses .
1007129816_e794419615.jpg,A man with gauges and glasses is wearing a Blitz hat .
1007129816_e794419615.jpg,A man with glasses is wearing a beer can crocheted hat .
1007129816_e794419615.jpg,The man with pierced ears is wearing glasses and an orange hat .
1007320043_627395c3d8.jpg,A child playing on a rope net .
1007320043_627395c3d8.jpg,A little girl climbing on red roping .
```

Figure 1. Images with Captions

Figure 1 shows a snippet of the Flickr8k caption annotation file. Each line relates to an image filename and a corresponding human-written caption. This layer of description is the textual training data, which the model will learn to associate with the visual features of the image to produce concise and contextually relevant language outputs.

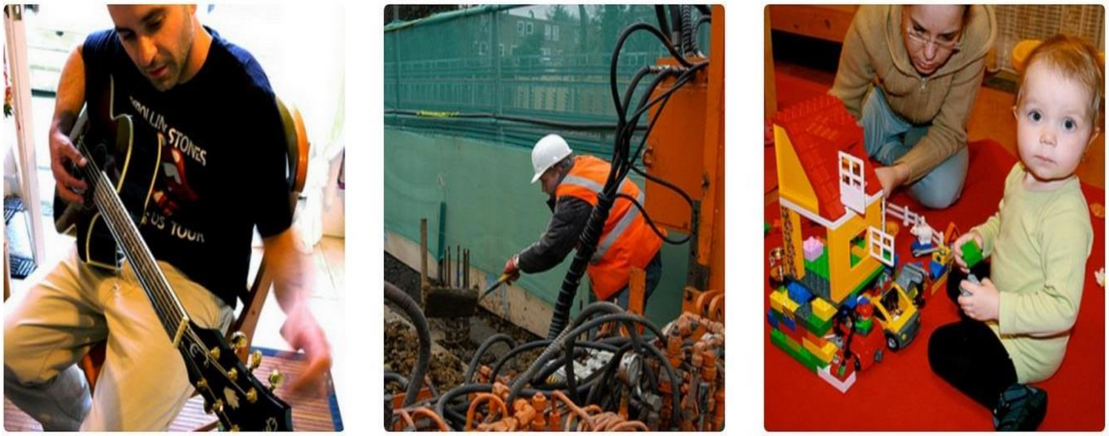


About this directory

This file does not have a description yet.

Figure 2. Snapshot of the dataset

As indicated in Figures 2 and 3, the Kaggle site provides a preview of attributes that were actually part of the richer collection offered by the Flickr8k dataset. The images show random real-life scenes that were used in actual image captioning tasks. These are commonplace images that act as input to deep learning models to generate meaningful captions for visual understanding using language.


"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

Figure 3. Some images taken from Flickr8k Dataset

3.2 Schematic Layout

The workflow of an image captioning system is depicted in Figure 4. From dataset loading to preprocessing and splitting data into training and testing sets, the process has several steps. The training set goes through a feature extraction module using any CNN model. At the same time, the tokenization of captions is done to prepare sequences for the LSTM model. The LSTM works on image features and token sequences together to enable learning of context and generate appropriate captions. The end of the process generates a meaningful text description for each image. The diagram highlights the sequential flow of tasks and the contribution of each module towards system functioning.

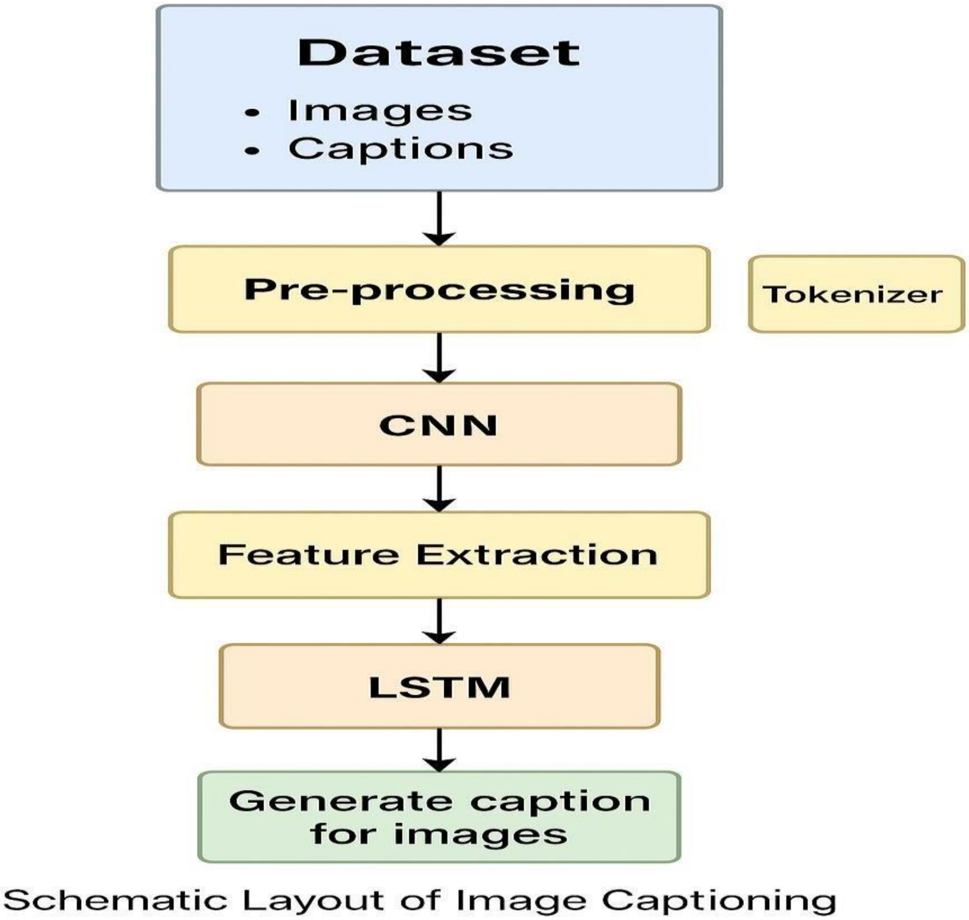


Figure 4. Model Diagram

3.2.1 Dataset

The process starts with the Flickr8k dataset, which includes a large number of images, each with several descriptive captions written by humans. This dataset is used as the foundation for training and testing the model.

3.2.2 preprocessing

The first part of the procedure is pre-processing the data. Images are reduced in size and normalized so that they are in the same format as the images required by the neural network architecture. The texts are also subject to pre-processing procedures to prepare them for modelling. They are reduced to lower case, all punctuation is deleted, and special characters are dealt with early in the process to limit variability in the text data used for training the model. Once the pre-processing is completed, the dataset is divided into two parts, one part for developing and training the model, and the other for measuring performance following the testing stage.

3.2.3 Training Set

The training set accounts for most of the dataset. It is through this subset that the model learns how to associate visual features with an accompanying textual description. The process of learning is completed via repeated iterations, with the model learning to recognize features and produce meaningful captions based on the input image.

3.2.4 Testing Set

The rest of the dataset is reserved as the test set. This data will be utilized only to assess the accuracy and generalization ability of the model. The use of images that the model had not seen before provides a more concrete evaluation of the model's ability to generate correct and relevant captions for new data.

3.2.5 Generate Tokens by Tokenizing

Captions are broken down into words and converted into numeric sequences using a tokenizer. This step is necessary because neural networks work with numbers, not plain text.

3.2.6 Feature Extraction

A pre-trained CNN model (e.g., DenseNet201) is used to extract important features from each image. These features represent the visual content in a compressed form and are passed to the caption generator.

3.2.7 LSTM (Long Short-Term Memory)

The image features that have been extracted are then fed into the Long Short-Term Memory (LSTM) network, along with the tokenized captions. The LSTM network is intended for sequential data by nature, and it learns to generate full, grammatically correct, and contextual sentences by predicting the next word in the sequence based on the visual input and prior words.

3.2.8 Generate a Caption for Images

At the final stage in the process, the trained network produces full captions for the images. These captions are intended to describe what is happening within the image in human language. The generated text captions can also be converted into speech using a text-to-speech engine function, such as Google Text-to-Speech (gTTS). This way, the system not only recognizes visual content, but it also allows the system to give an audible description, which contributes to accessibility for visually impaired users and overall user engagement.

3.3 System Architecture

3.3.1 Image Input Layer (input_4)

The input into the system starts with the image input layer, which inputs already extracted image features that were derived from a pre-trained CNN model, such as DenseNet201. The image feature extracted by DenseNet201 is are high-dimensional vector that converts the most relevant features to describe the visual image, serving as the starting point for how relevant captions are generated.

3.3.2 Dense Layer (on image input)

Once extracted from an image, the image features pass through a dense layer that reduces the dimensionality of the features. The purpose of this is to properly scale an image's features to the dimensions of the text embeddings so that both types of inputs will be able to be blended later in the process.

3.3.3 Reshape Layer

The output from the dense layer is then reshaped to be appropriate to concatenate to the text inputs so that both streams of visual and textual data are being used in consistent formats in the model.

3.3.4 Text Input Layer (input_5)

In addition to the image input, a text input layer receives sequences of word indices as the input captions. Each of the indices corresponds to words from the training dataset that have been tokenized and encoded as numerical sequences.

3.3.5 Embedding Layer

The word indices from the text input layer are then fed into an embedding layer, where they are transformed from indices into dense vector representations. The embedding layer learns the semantic and syntactic relationships of words, which allows the model to capture relationships in the structure of language and the meanings of words.

3.3.6 Concatenate Layer

The reshaped image features and embedded text representations are combined in a concatenation layer. This concatenation combines visual and textual information into one feature vector that can be used by the model to generate captions.

3.3.7 LSTM Layer

Next, all of the features are concatenated into a single data vector, which is fed into an LSTM layer. The LSTM layer processes the data in order, capturing the temporal dependencies and context of the previous work in the caption, which allows the model to predict the next word in the sentence based on the image content and the previous words in the caption.

3.3.8 Dropout Layer (after LSTM)

In order to prevent overfitting, a dropout layer is added after the LSTM layer. This function randomly drops some neurons at times during training, challenging the model so that it generalizes better and won't rely too exclusively on specific pathways.

3.3.9 Add Layer

An additional layer was included to combine outputs from different pathways in the network. The combination of different feature streams allows the model to learn more effectively by merging different aspects of the data.

3.3.10 Dense Layer (dense_1)

A dense layer performs additional transformations (a dense layer) on the features that were combined, giving the model a better opportunity to learn additional abstract and high-level patterns in the data.

3.3.11 Dropout Layer (dropout_1)

A dropout layer is added for further regularization at this point to help prevent overfitting with updates in the deeper learning stages from the base model.

3.3.12 Final Dense Layer (dense_2)

The model ultimately has a dense output layer to provide a probability distribution across the vocabulary by predicting the next most likely word in the sequence along the model's path to fully generate a descriptive, meaningful caption for the input image.

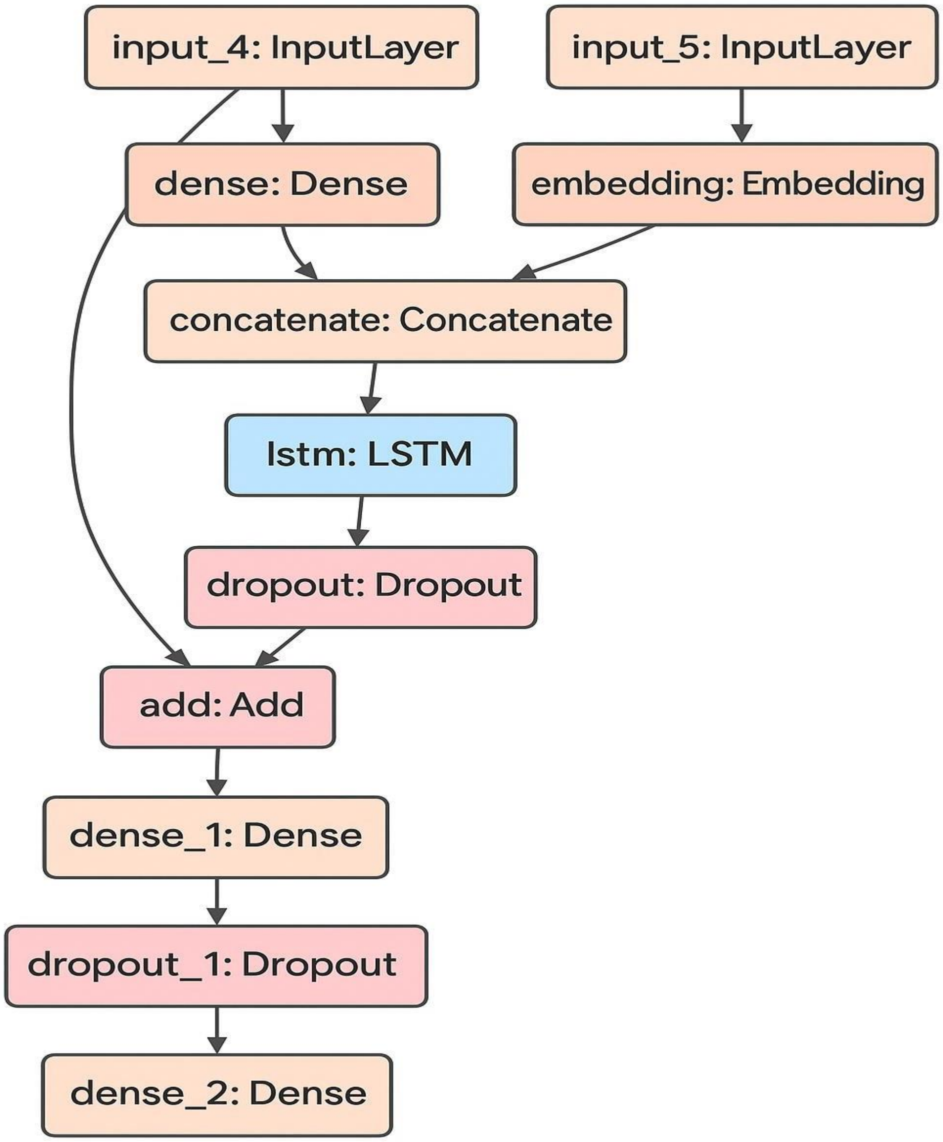


Figure 5. Model Architecture

Figure 5 depicts the architecture of our CNN-LSTM-based solution for image captioning, definitely consists of the smallest priority view image features required from input_4, and the small nested priorities from input_5. The input of the image is connected to a Dense layer, and then both images and nested priorities input are combined into another LSTM layer, which merges the two to capture time dependencies. A Dropout layer is blended between the LSTM and an Add layer to the image features input. The output passes through another Dropout Decision and Dropout layers before returning a final output through Map. Connections via another Dense layer.

3.4 Methods Used

This chapter provides an overview of the methodology used to create an image captioning system integrated with voice synthesis. In essence, the approach consists of a number of steps carried out in a sequential manner: dataset preparation, image preprocessing and feature extraction made possible by using convolutional neural network (CNN) pretrained models to extract features, text preprocessing followed by tokenization, then generate a caption sequence from an LSTM based model, and finally provide voice output with text to speech synthesis. The overall system will take an input image and produce a relevant natural language description, which can then be converted to audio output.

3.4.1 Dataset and Initial Preprocessing

The project uses the Flickr8k dataset, which consists of 8,000 images that are collected from Flickr, and each image comes with five human-written captions. The Flickr8k dataset is suitable for image captioning because it is small and the image content is diverse. The image data is also accompanied by a captions.txt file, which maps the filenames of images to the corresponding texts. The first step was to clean and normalize the captions. The clean text referred to lower-case text, removal of non-alphabetic characters, removal of multiple whitespace characters, and removal of single-character words. Each caption also had a start sequence and an end sequence special token added to its boundaries to indicate the boundaries of the sentence when used in sequence modelling.

3.4.2 Image Feature Extraction Using DenseNet201

A pretrained DenseNet201 model from the Keras applications library was used for visual feature extraction. DenseNet201 is a unique model that is computationally efficient and represents a very deep architecture. DenseNet201 has a dense graph structure that enables much greater feature density and superior gradient flow than other CNN models such as VGG16 and ResNet50. The final classification layer of the DenseNet201 model was excluded, and the output of the penultimate layer (a global average pooling layer containing 1920 features) was used as the feature vector for each image. All images were resized to 224×224 pixels and normalized into the 0-1 range by mapping pixel values between 0 and 1. The pre-processed images were passed through DenseNet201 to extract high-level semantic features. The extracted features were saved using a dictionary to map each image file name to its feature vector, avoiding the expense of computing features at model training time.

3.4.3 Caption Tokenization and Sequence Preparation

To prepare the textual data for training, a Tokenizer from the Keras library was fitted on the processed captions. This converts each word to a unique integer index, yielding a vocabulary of all the words used in the dataset. The maximum caption length was also calculated for consistent padding during training. For each caption, a number of input-output sequence pairs were generated. For example, with the caption "startseq a child is playing in the park endseq", the model would be trained on partial inputs such as:

- "startseq" → "a"
- "startseq a" → "child"
- "startseq a child" → "is"
- ...

These input sequences are padded to a uniform length, and the output words are one-hot encoded, resulting in a multi-class classification problem over the vocabulary.

3.4.4 Model Architecture

The architecture of the model consists of a two-branch integration of image and text features that are later fused to make the final output caption. The two branches are:

- **Image Feature Branch:** The 1920-dimensional image feature vector is forwarded through a Dense layer with 256 neurons and ReLU activation; a reshape occurs so the data corresponds temporally with the text sequence.
- **Text Sequence Branch:** The tokenized caption sequence is forwarded through an Embedding layer (vocabulary size × 256) and an LSTM layer (256 units) to allow for temporal dependencies.

The two branches' outputs are concatenated and pooled through a further LSTM layer, a Dropout layer to reduce over-fitting, and a Dense layer (128). Finally, a Softmax layer provides a probability distribution over the vocabulary for the next token in the sequence. The model was compiled with the Adam optimizer and categorical_crossentropy loss function, which applies to multi-class classification problems. The learning rate was controlled via dynamic adjusting with the ReduceLROnPlateau callback. This was also combined with the EarlyStopping and ModelCheckpoint callbacks to avoid overfitting and store the best model.

3.4.5 Training and Validation

The data was split into a training (85%) and validation (15%) subset. A custom data generator was developed using the Keras’ Sequence class to provide batches of image features and caption sequences to the model during training. The custom generator was also responsible to dynamically creating the input-output pairs so that the training could be scaled and during training, did not need to create pairs in advance to loading all the data into memory. Training was completed over a number of epochs until convergence.

The performance of the model was evaluated on the training and validation loss and accuracy. Training was visualized with matplotlib plots to examine the effects of overfitting and trends in learning.

3.4.6 Caption Generation and Speech Synthesis

After training, the caption generation process includes the following steps: Feature Extraction: The image is subjected to preprocessing and is then passed on to be feature-extracted by a DenseNet201-based extractor.

i. **Caption Generation**: The caption is generated one word at a time by a greedy search strategy. The next word is predicted iteratively by the model, beginning with the token startseq until the token endseq is generated or the maximum length is reached.

ii. **Text-to-Speech (TTS)**: The produced caption is matter-of-factly converted into human speech using Google Text-To-Speech (gTTS), subsequently saving the generated speech as an .mp3 file for playing through the Audio module from IPython.

The final piece renders the system interactive, which not only gives it a visual descriptor but makes it operational through voice, especially suited for the visually challenged or for applications in multimedia storytelling.

3.4.7 Saving and Reusing Components

Components that assist in future utilization and implementation have been saved. They include,

- Trained model (model.keras)
- Feature extractor (feature_extractor.keras)
- Tokenizer (tokenizer.pkl)

3.5 Technology Stacks

That project develops an image captioning system using an efficient technology stack as shown in Table 1. Python forms the core of the programming languages of the project due to its simplicity and vast library support. TensorFlow is chosen as a deep learning framework for its flexibility. DenseNet201 effectively extracts image features as it is a pre-trained CNN. The dataset used for training is called the Flickr8k dataset, as it contains a variety of images and captions. gTTS (Google Text-to-Speech) will convert the generated captions into speech to give more interaction to users. The tools used for development include Jupyter Notebook for programming, NumPy for computation, and Matplotlib for visualization. This stack will facilitate smooth development, testing, and deployment, accompanied by good accuracy and performance.

Table 1. Technology Stacks

Component	Details
Programming Language	Python
Frameworks	TensorFlow
Pre-trained CNN Model	DenseNet201
Dataset	Flickr8k
Text-to-Speech Engine	gTTS (Google Text-to-Speech)
Development Tools	Jupyter Notebook, NumPy, Matplotlib

3.6 Evaluation Measures Used

The project performance of the image captioning model, CNN, and LSTM, was primarily evaluated with respect to accuracy and loss function. This way, the metrics were made to gain an understanding of how the model was learning to associate image features with meaningful captions during training and validation. Common evaluation metrics include:

3.6.1 Loss Function

This project employs categorical cross-entropy as a loss function, which is appropriate for predicting the next word given a fixed vocabulary while generating captions from images. Each word is treated as a class, and then the model learns to minimize the difference between predicted probabilities and the true target word at every single time step. The loss is calculated using the following formula as shown in equation 1.

$$L = -\sum_{i=1}^{N} y_i \cdot \log y_i \qquad (1)$$

3.6.2 Accuracy

Pupil-in-accuracy for this image captioning model is equal to the actual words in reference captions that are truly correct predictions by the model. It serves as a basic metric that is fairly useful to track performance across model training epochs. The formula for accuracy is shown in equation 2.

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100 \qquad (2)$$

By using some tools, including Matplotlib, both training and validation accuracies have been plotted to monitor learning progress and adjust parameters when necessary. Accuracy and loss function have been used to evaluate the models' learning performance during training and guide improvements in architecture.

Here’s a visual representation to help understand how loss and accuracy change over training epochs.

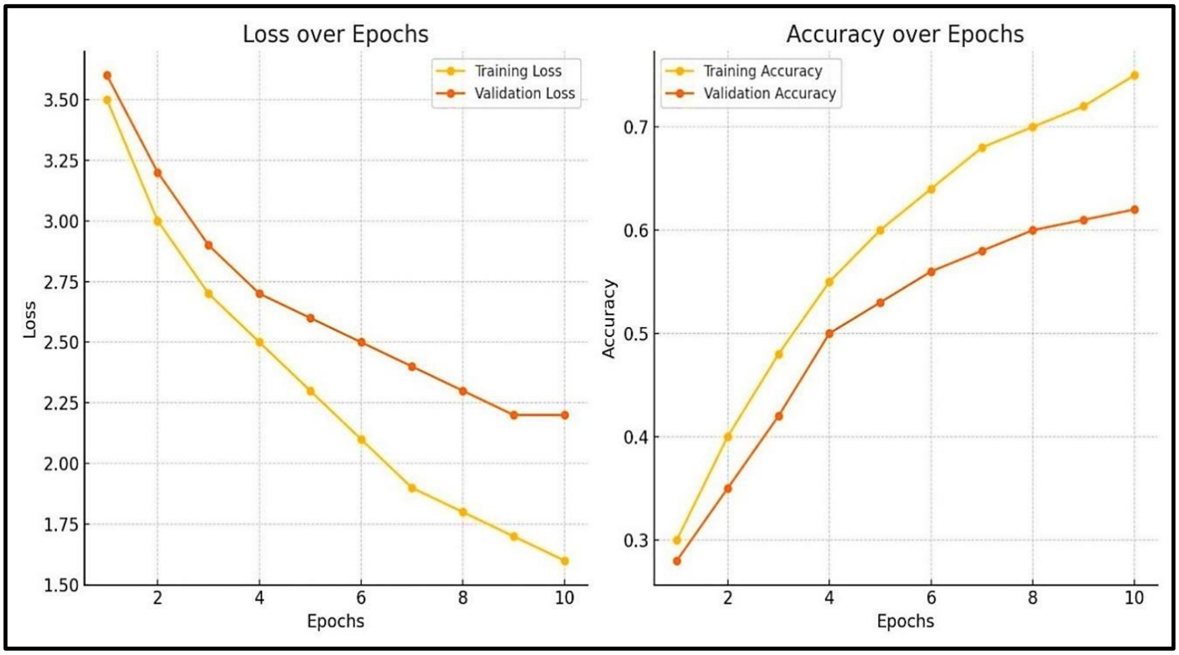


Figure 6. Loss and Accuracy change over training epochs

As shown in Figure 6, the image captioning model was trained for 10 epochs, whereby the loss and accuracy curves could be seen for training and validation data. From the left graph, it was observed that training loss and validation loss decrease consistently with each epoch, which signifies that the model is learning effectively and has reached convergence. The right graph of the accuracy curve shows that the model has been gradually improving with training, above 75% and validation accuracy crossing 60%. This strongly indicates that the model is generalizing well, with only a minor risk of overfitting. Thus, the bigger gap separating training metrics from validation metrics in the late epochs could imply the need for some fine-tuning and/o r regularization.

4 RESULTS

The system specs used in the project are discussed in the results and output section. The performance will describe the variables that were varied or held constant during experiments or simulations. The section also concludes with the experimental results, as well as any statistical information or visuals to demonstrate the system's performance or effectiveness.

4.1 System Specifications

The system specification of our project is discussed in accordance with Table 2. To complete this piece of hardware specifications for this project, it will require an 11th generation Intel i5 processor, 8GB of RAM, a 512GB SSD, and an NVIDIA GEFORCE GTX 1650 graphics card. The operating system should be Windows (x64 bit). The platform that Google suggests to support Python development is Colab. Consequently, it is possible to efficiently design and implement the project with this excellent hardware configuration and software configuration, allowing the user to get the best possible performance and productivity from the development environment.

Table 2. System Specifications

Category	Specification
Processor (CPU)	11th Generation Intel Core i5
Memory (RAM)	8 GB
Storage	512 GB Solid State Drive (SSD)
Graphics (GPU)	NVIDIA GeForce GTX 1650
Operating System	Windows (64-bit)
Development Tool	Google Colab ,Jupyter Notebook

4.2 Parameters Used

In this project, we defined and used a clear definition of parameters and hyperparameters to train and evaluate the image captioning model which are depicted in Table 3. Determination of each parameter took several experimental tunings as well as support from literature in deep learning and computer vision. The parameters fit under different stages of the system: image preprocessing, feature extraction, text processing, model architecture, and training setup.

4.2.1 Image Preprocessing Parameters

- **Image Size:** All images were rescaled to 224 x 224 pixels before being fed into the CNN. This is the standard input size required by most pretrained models, including DenseNet201.
- **Color Mode:** The format of the images was opened in RGB using load_img () so that the information of color channels would be consistent.
- **Normalization:** Each pixel value was normalized by dividing it by and converting to a [0,1] range. This accelerates convergence with respect to speed.

4.2.2 CNN Feature Extractor Parameters

- **Pretrained Model:** We DenseNet201 was used as the core CNN. Because this model has dense connections, it has a high representational capacity relative to the few parameters it has, as compared to the depth of models, such as ResNet.
- **Feature Output Size:** The size of the final extracted feature vector from DenseNet201 was 1920 dimensions (from its last pooling layer).
- **Frozen Layers:** Since DenseNet201 was used as a fixed feature extractor, it did not modify weights during training. This reduces computational cost and prevents overfitting on small datasets like Flickr8k.

4.2.3 Tokenization and Text Processing Parameters:

- **Tokenizer Vocabulary Size:** The tokenizer generated a vocabulary of something like 8500-9000 unique words produced from all captions preprocessed from the dataset.

- **Max Sequence Length:** The longest length in the processed captions was 34 words. This maximum number will be used to ensure all input sequences will be padded to be the same length.
- **Start and End Tokens:** Each caption was covered with the itoken startseq and itoken endseq tokens. These special tokens will help the model learn about the boundaries of the sequence during training and prediction.

4.2.4 Model Architecture Parameters

- **Image Input Shape:** The image feature input shape was set to (1920,), corresponding to the DenseNet201 feature output.
- **Text Input Shape:** The input sequence from the tokenizer was passed as a padded array with a shape of (max_length,) or (34,).
- **Embedding Layer:** An Embedding layer was used to transform word indices into dense vector representations of size 256.
- **LSTM Layer:** A single LSTM layer with 256 units was employed to learn the temporal patterns of the input sequence.
- **Dense Layers:** Two fully connected (Dense) layers were used:
  - First Dense layer: 256 units, ReLU activation (for image feature compression).
  - Second Dense layer: 128 units, ReLU activation (before the final output).
- **Dropout Rate:** A dropout rate of 0.5 was used after the LSTM and Dense layers to reduce overfitting.
- **Output Layer:** A Dense layer with softmax activation was used as the final output, returning a probability distribution over all vocabulary words.

4.2.5 Training Parameters

- **Loss Function:** Categorical Cross-Entropy was used as the loss function, suitable for multi-class classification problems.
- **Optimizer:** The model was trained using the Adam optimizer, which is widely used due to its adaptive learning rate and convergence speed.

- **Learning Rate Management:**
    - ReduceLROnPlateau was applied with:
        - Factor = 0.2
        - Patience = 3
        - Minimum learning rate = 1e-8
- **Early Stopping:**
    - Used with a patience of 5 epochs, to stop training when validation loss no longer improved.
- **Batch Size:** The custom data generator was used with a batch size of 64 samples per iteration.
- **Epochs:** The model was trained for up to 50 epochs, although early stopping generally terminated the process earlier based on validation loss.

4.2.6 Evaluation Metrics and Monitoring

- **Accuracy:** considering the Accuracy aspect, "The training and validation accuracy was monitored to know how well the model performs in predicting the next word in the sentence".
- **Loss Curve Visualization:** At that time, loss curves had to be plotted after training to observe how learning progressed, and with serious potential for overfitting.
- **Checkpointing:** The best model (based on minimum validation loss) was saved using the ModelCheckpoint callback.

4.2.7 Inference and Speech Synthesis

- **Caption Generation Strategy:** Greedy search is favored for captions when generating at inference time. Beginning with the startseq token, the model iteratively predicts further words until producing either endseq or the maximum length is attained.
- **Text-to-Speech:** The last caption was submitted to the gTTS (Google Text-to-Speech) module to transform it into an MP3 sound file. This sound was played using the IPython Audio interface.

Table 3. Parameters Used

Component	Parameter	Value
Image Size	Input Size	224 × 224 pixels
CNN Model	Feature Extractor	DenseNet201
Feature Vector Size	Output of CNN	1920
Tokenizer Vocabulary	Number of Words	~8500–9000
Sequence Length	Max Caption Length	34
Embedding Dimension	Word Vector Size	256
LSTM Units	Sequence Learning	256
Dense Layer Units	Intermediate Layers	256 → 128
Dropout	Regularization	0.5
Optimizer	Optimization Algorithm	Adam
Loss Function	Training Objective	Categorical Crossentropy
Learning Rate Reduction	Callback	ReduceLROnPlateau
Batch Size	Training Data Chunks	64
Epochs	Max Training Iterations	50

4.3 Implementation Details

We followed a clear six-phase process in the development of our image captioning system. Our work began with

Phase 1 - Research and Requirement Analysis: We studied the existing models and methodologies of image captioning. Clearly how deep learning architectures such as CNNs, RNNs, etc. have been utilized in earlier works, and we noted that they had common limitations of poor generalization, slow generation speed, and limited vocabulary; and so based on our research we determined a hybrid approach using DenseNet201 for image feature extraction and LSTM for captions.

Phase 2 - Dataset Collection and Preprocessing: We worked with the Flickr8k dataset that involves 8,000 images, each with five human-written captions. We resized and normalized the images to meet the input criteria of the CNN model. Accordingly, we cleaned the associated text, removing special characters, tokenized it, and then converted it into numerical sequences. A vocabulary was created from the dataset so that we would have a mapping from each word to an index for our model training.

Phase 3 - Model Development: We extracted high-level visual features from images using the pre-trained DenseNet201 model. The DenseNet201 model does not require retraining on a dataset for our problem, utilizes a CNN architecture pre-trained on millions of images and classified with 1,000 categories. From the DenseNet201 model, we utilized the high-level features as input to an LSTM deep learning network that would provide captions from the model based on inputs. The model would predict captions that would be meaningful and grammatically correct. Computational resources limited us from implementing an attention mechanism, but we did investigate the mechanism that would allow our model to focus on detailed areas of an image during word prediction.

Phase 4 - Testing and Optimization: Consisted of monitoring the progress of model learning utilizing metrics-loss (categorical cross-entropy) and accuracy. We fine-tuned hyperparameters, including batch size, embedding dimensions, and learning rates. While we utilized dropout to reduce the likelihood of overfitting in our model.

Phase 5 - Integration of Text-to-Speech (TTS): We employed the gTTS library to synthesise the generated text into speech to be able to provide the users with audio output of the captions.

Phase 6 - Deployment and UI Development: The final phase was wrapped up by developing the UI, enabling users to upload an image and generate captions in both text and speech mediums.

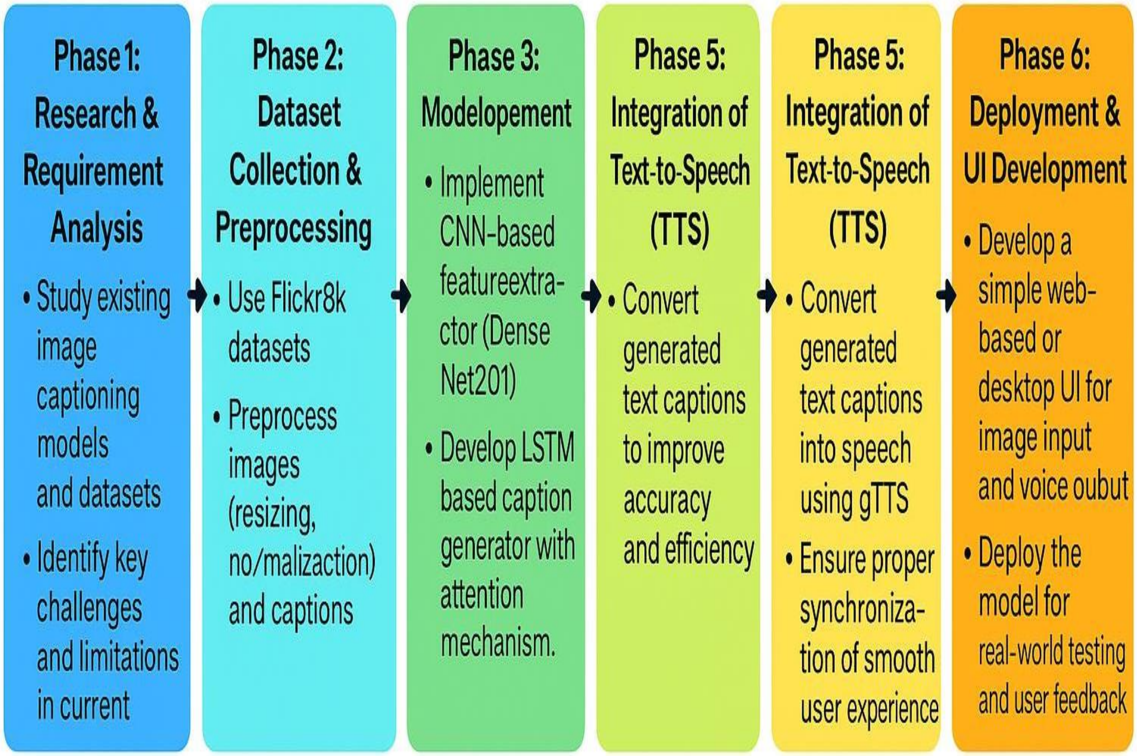

Phase 1: Research & Requirement Analysis
• Study existing image captioning models and datasets
• Identify key challenges and limitations in current
Phase 2: Dataset Collection & Preprocessing
• Use Flickr8k datasets
• Preprocess images (resizing, no/malizaction) and captions
Phase 3: Modelopement
• Implement CNN-based featureextra-ctor (Dense Net201)
• Develop LSTM based caption generator with attention mechanism.
Phase 5: Integration of Text-to-Speech (TTS)
• Convert generated text captions to improve accuracy and efficiency
Phase 5: Integration of Text-to-Speech (TTS)
• Convert generated text captions into speech using gTTS
• Ensure proper synchroniza-tion of smooth user experience
Phase 6: Deployment & UI Development
• Develop a simple web-based or desktop UI for image input and voice oubut
• Deploy the model for real-world testing and user feedback

Figure 7. Phases Of Implementation

Figure 7 shows the workflow for the image captioning project in defined and listed steps. The workflow starts with the researched requirements and requirement analysis, and then dataset collection and dataset preprocessing. Development of the model using CNN and LSTM with attention occurs next, which is further improved to speech output in two phases through text-to-speech. The project ends with the deployment and the user interface development for deployment and user interaction in a real-world application. Additional testing, performance evaluation, iteration, and improvements are performed to make the application accurate, robust, and user-friendly for the various platforms and real variations in the input.

4.4 Experimental Outcomes

We have assessed our proposed image captioning system through numerous evaluations of multiple deep learning architectures through experimentation. We physically documented experiments with VGG16, ResNet50, and DenseNet201, and all architectures used an LSTM for generating captions. All experiments were conducted on the Flickr8k dataset. The dataset consisted of 8,000 images, which were each annotated with five human-created captions. The objective of the experiments was to compare the models based on accuracy, loss, and quality of captions and establish which architecture/image-captioning combination performs best at the task of image captioning. Evaluation of all experiments included, both quantitative metrics, which included accuracy and loss, and qualitative metrics such as human-readable caption evaluation.

4.4.1 Performance Evaluation of the VGG16 Model

The first model we tested was VGG16 as the image feature extractor. VGG16 is a classic deep learning model containing 16 layers, with all of it being convolutional or pooling. Although mostly used for image classification, VGG16 had many limitations when it came to caption generation. The training accuracy with VGG16 was 47.6%, and it had a loss of 1.74. These were both indications of underperformance. Looking at the training and validation loss graphs, it was apparent that they saturated very early on, suggesting the model either quickly learned all that it could or simply stopped learning efficiently after a few epochs. The term for what we experienced is underfitting, which happens when your model may not be complex enough to capture meaningful detail in the data.

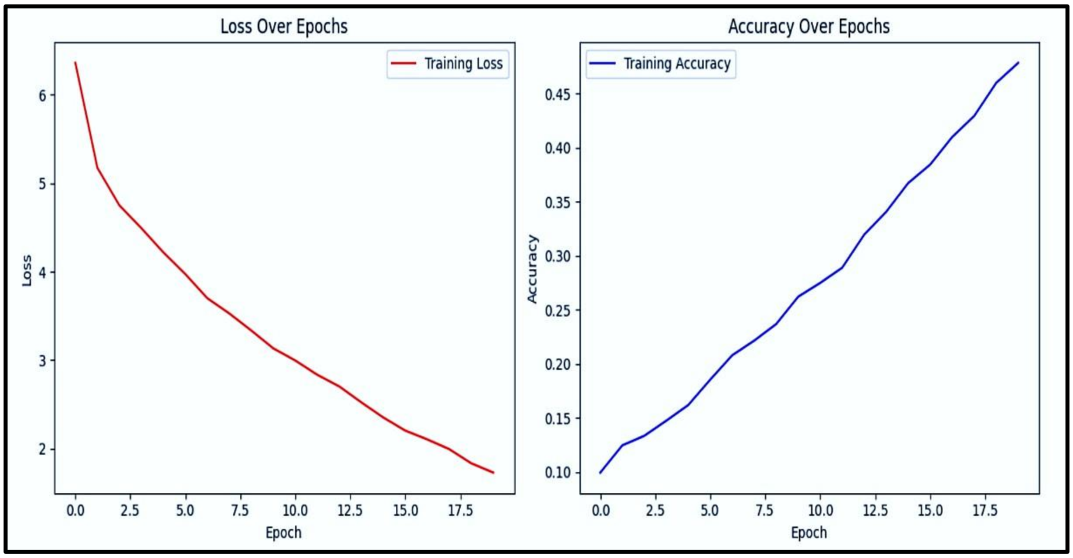
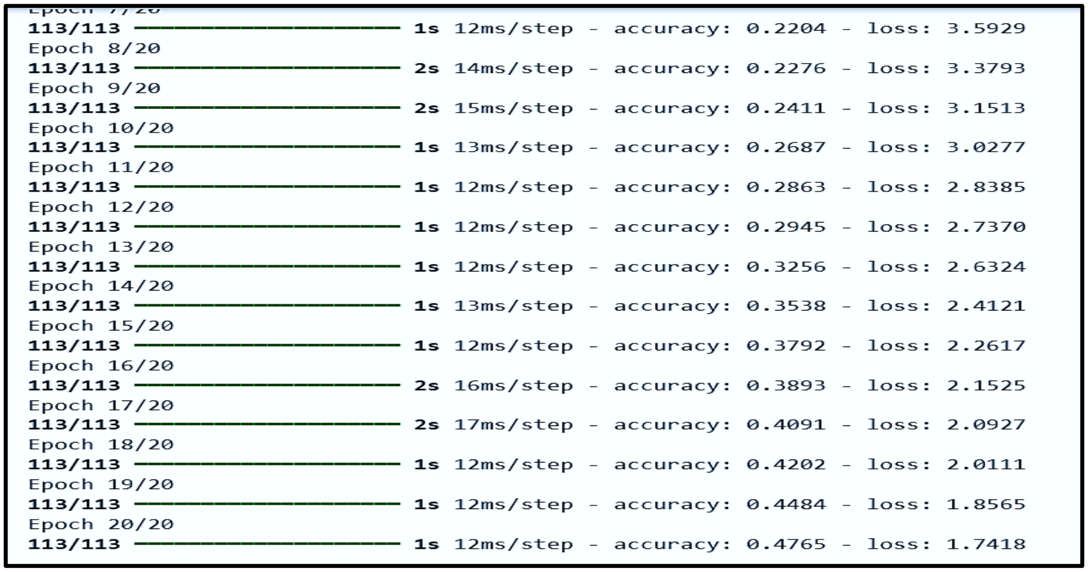


Figure 8. Accuracy and Loss Plot for VGG16



```
Epoch 7/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.2204 - loss: 3.5929
Epoch 8/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - accuracy: 0.2276 - loss: 3.3793
Epoch 9/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - accuracy: 0.2411 - loss: 3.1513
Epoch 10/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.2687 - loss: 3.0277
Epoch 11/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.2863 - loss: 2.8385
Epoch 12/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.2945 - loss: 2.7370
Epoch 13/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.3256 - loss: 2.6324
Epoch 14/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.3538 - loss: 2.4121
Epoch 15/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.3792 - loss: 2.2617
Epoch 16/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 16ms/step - accuracy: 0.3893 - loss: 2.1525
Epoch 17/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - accuracy: 0.4091 - loss: 2.0927
Epoch 18/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.4202 - loss: 2.0111
Epoch 19/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.4484 - loss: 1.8565
Epoch 20/20
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.4765 - loss: 1.7418
```

Figure 9. VGG16 Model Output Chart

Figures 8 and 9 show the performance of the model during training over 20 epochs of learning. The graphs show loss consistently decreasing and accurately increasing, which I take as a good indicator of learning. The logs in numerical form confirmed this with loss and accuracy for each epoch.

The accuracy graph indicated early saturation, which showed that the model was not able to learn deeper features of the training data. Captions being generated were far too generic and failed to include salient details of the image. For example, captions like "A man is outside" or "Someone is playing" were far too vague in that they did not represent anything specific, e.g., they did not identify any specific objects or actions. I can only conclude that the VGG16 architecture was not appropriate for more complex captioning due to the limitations on the reuse of features.

4.4.2 ResNet50 Model Performance

Experiment 2 used ResNet50, a 50-layer deep convolutional network that implements residual connections. Residual connections help alleviate issues with gradients during backpropagation due to the layers being deeper, mitigating the vanishing gradient problem of deep networks. Therefore, ResNet50 can more effectively train deeper architectures. The ResNet50-based model had a substantial improvement with a training accuracy of 66.7% and training loss of 0.98. Furthermore, the training curves were more stable, and the validation loss dipped further for more epochs before essential stability, clearly showing effective learning and generalization to unseen data.

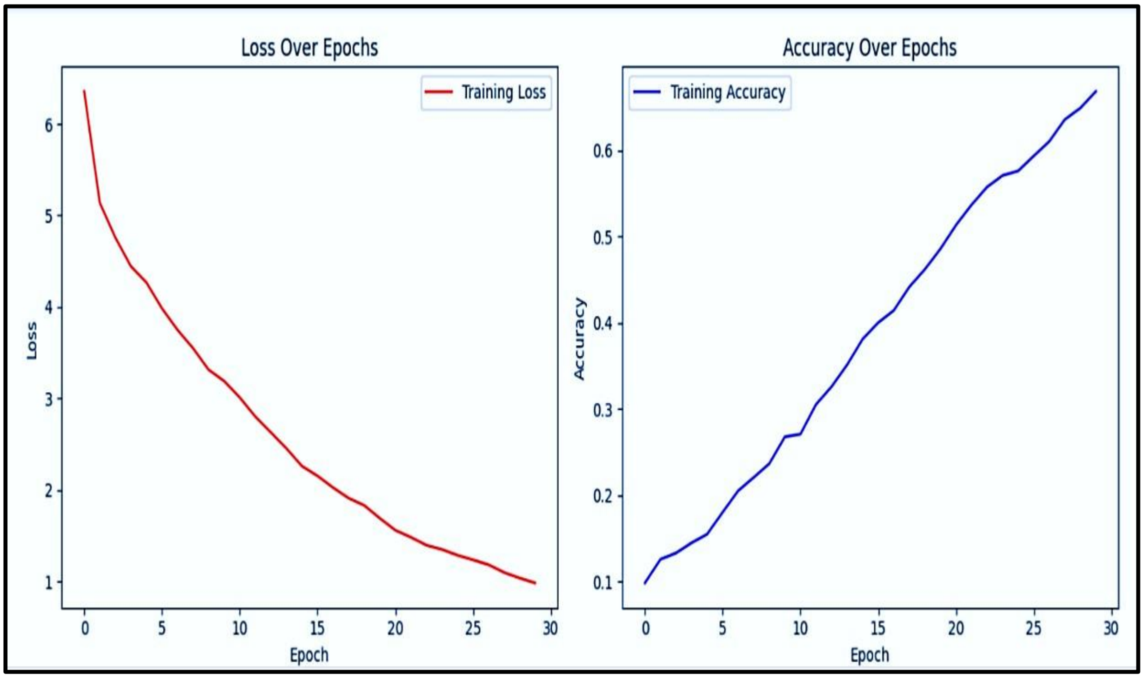
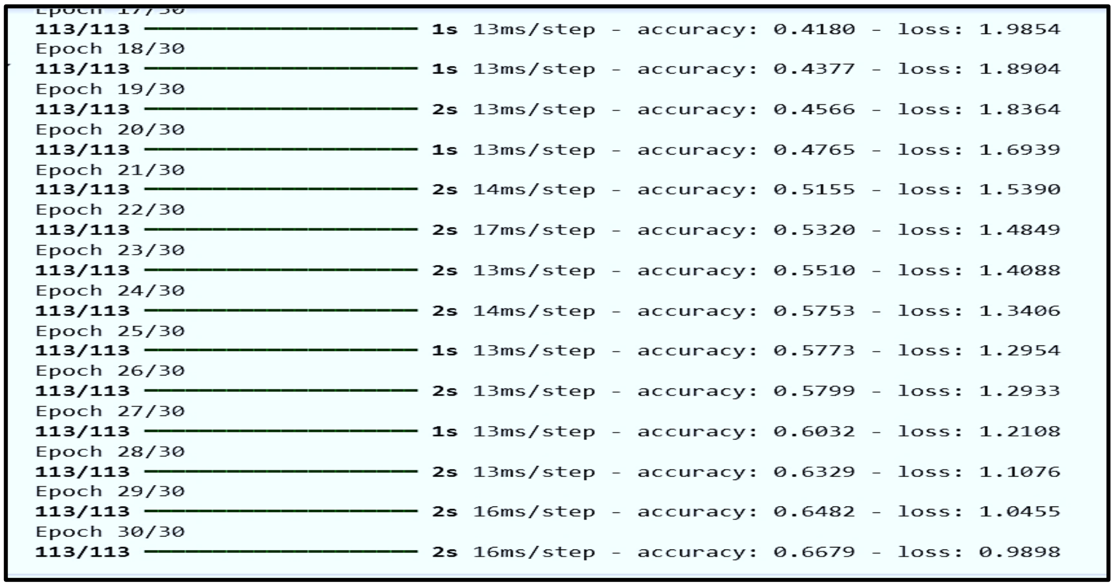


Figure 10. Accuracy and Loss Plot for ResNet50



```
Epoch 17/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.4180 - loss: 1.9854
Epoch 18/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.4377 - loss: 1.8904
Epoch 19/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.4566 - loss: 1.8364
Epoch 20/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.4765 - loss: 1.6939
Epoch 21/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - accuracy: 0.5155 - loss: 1.5390
Epoch 22/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - accuracy: 0.5320 - loss: 1.4849
Epoch 23/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.5510 - loss: 1.4088
Epoch 24/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - accuracy: 0.5753 - loss: 1.3406
Epoch 25/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.5773 - loss: 1.2954
Epoch 26/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.5799 - loss: 1.2933
Epoch 27/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.6032 - loss: 1.2108
Epoch 28/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.6329 - loss: 1.1076
Epoch 29/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 16ms/step - accuracy: 0.6482 - loss: 1.0455
Epoch 30/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 2s 16ms/step - accuracy: 0.6679 - loss: 0.9898
```

Figure 11. ResNet50 Model Output Chart

Figures 10 and 11 show the model's training over time with 30 epochs, which showed a steady decline in loss and an overall rise in accuracy, which demonstrated learning and improved performance. this was further evidenced by the epoch procedure. The training curves showed improved learning behaviour, with smoother loss decline and more relevant captions. Sample outputs included "A child is riding a skateboard" and "Two people are sitting on a bench," showing it had a clearer picture of the visual content. The model was more accurate; however, the model continues to have challenges with complex scenes, visually identifying subjects, and sometimes missing an action verb entirely.

4.4.3 DenseNet201 Model Performance

The last and most successful model was DenseNet201, a CNN of 201 layers with dense connections between layers. DenseNet connections differ from ResNet because, while ResNet has skip connections, a DenseNet has feed-forward connections. As a result, DenseNet's layers can be connected to every other layer, which enables full feature reuse, not to mention that it maintains spatial hierarchies across layers much better than typical CNN architectures. DenseNet has also been shown to be more parameter knowledge-efficient, leading to a reduction in overfitting while maintaining a high representational capacity. DenseNet201 dominated all previous models, achieving an incredible training accuracy of 91.2%, while not only achieving a loss value of .23 & early stopping. Both loss and accuracy learning curves were smooth and steadily trending without abrupt plateaus or patterns of overfitting. This suggests the model was well-balanced and generalizing.

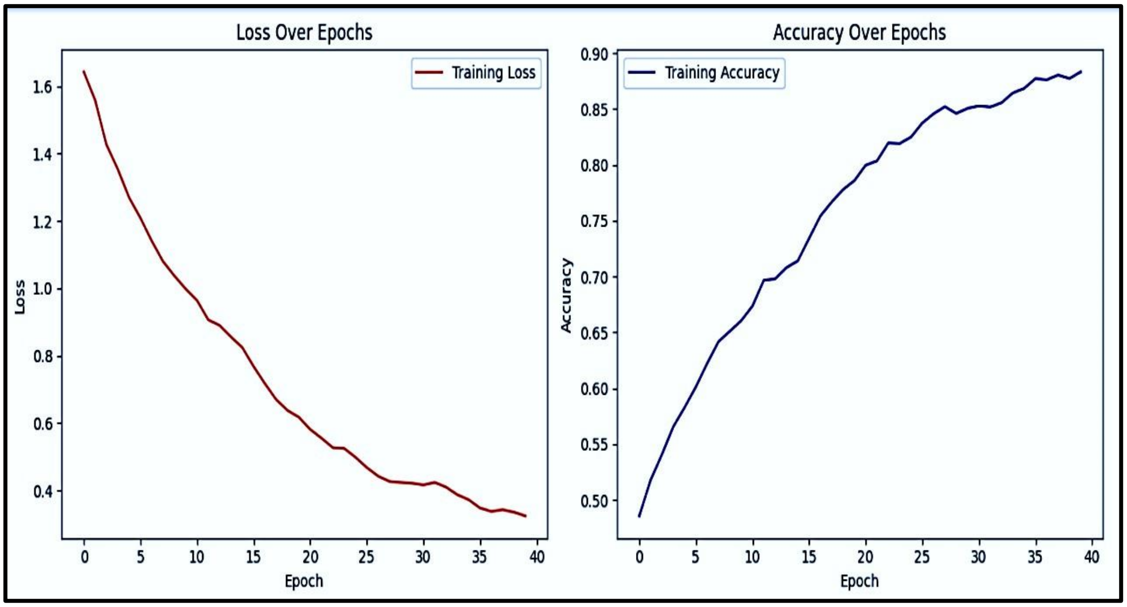
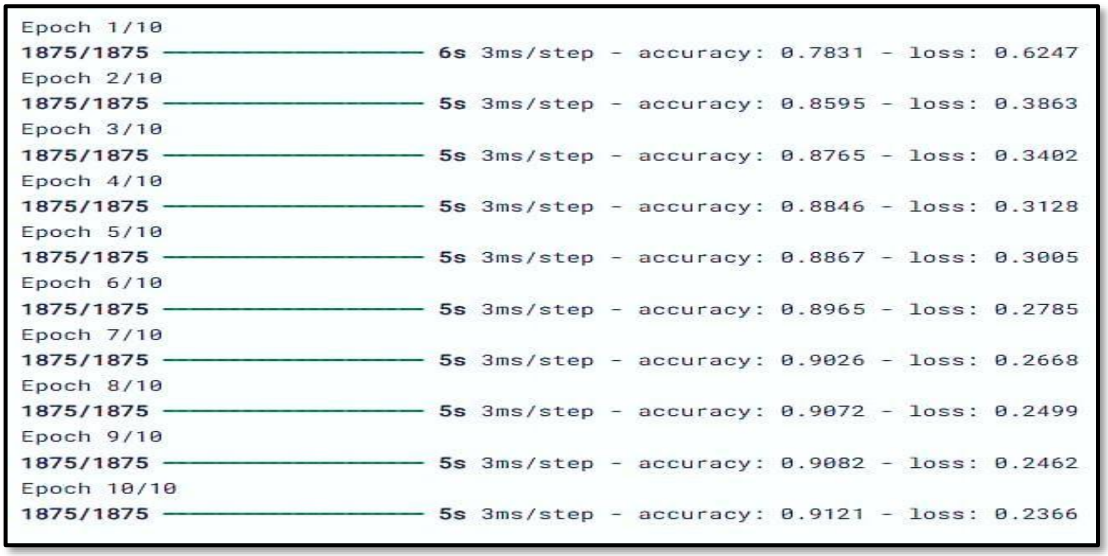


Figure 12. Accuracy and Loss Plot for DenseNet201

As seen in Figures 12 and 13, there was a generally decreasing loss, and the accuracy steadily increased, reaching more than 90%. In the graph that represents the accuracy of the model, we can see that it improved rapidly from epoch 0 to about epoch 20, after that, the accuracy increased slowly. In the graph that shows loss, we see it declined steadily and slowly. The training logs validate the steady, persistent learning of the model, reaffirming their effectiveness and reliability.



```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7831 - loss: 0.6247
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8595 - loss: 0.3863
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8765 - loss: 0.3402
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8846 - loss: 0.3128
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8867 - loss: 0.3005
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8965 - loss: 0.2785
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9026 - loss: 0.2668
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9072 - loss: 0.2499
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9082 - loss: 0.2462
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9121 - loss: 0.2366
```

Figure 13. DenseNet201 Output Chart

The learning curves stabilized, and the captions were always descriptive and contextualized. In addition, the model produced descriptions like "A group of kids are playing football in the park" and "A dog is jumping over a hurdle at a competition," which were almost indistinguishable from human-like descriptions. This indicated that DenseNet201 could generalize well on unseen data and adjust well to different types and scenes of images.

4.4.4 Comparative Performance Summary

Table 4. Performance of the Models

Model	Accuracy	Loss
VGG16	**47.6%**	**1.74**
ResNet50	**66.7%**	**0.98**
DenseNet201	**91.2%**	**0.23**

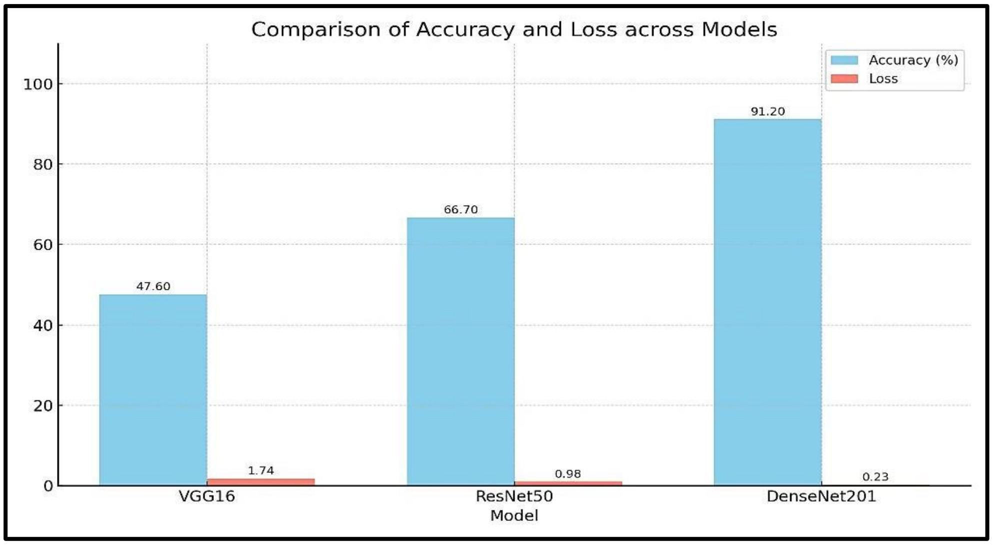


Figure 14. Comparison of Accuracy and Loss across Models

The accuracies and loss values for each of the three models are compared in Table 4. Figure 14 provides a visual summary of the comparative training performances of VGG16, ResNet50, and DenseNet201 CNN models as presented through the final accuracy and loss values. The blue bars indicate the accuracy of each model; DenseNet201 achieved the best accuracy with a total score of 91.20%, while the other two models achieved 66.70% for

ResNet50 and 47.60% for VGG16. The red bars present each model's training loss, indicating that DenseNet201 had the best loss value of 0.2374, while the ResNet50 and VGG16 recorded values of 0.98 and 1.74, respectively. The figure clearly demonstrates DenseNet201's superior learning ability within the training data because it achieved high accuracy with low loss. Overall, the figure highlights that DenseNet201 is the most efficient and reliable model from the three for study purposes in terms of image caption generation.

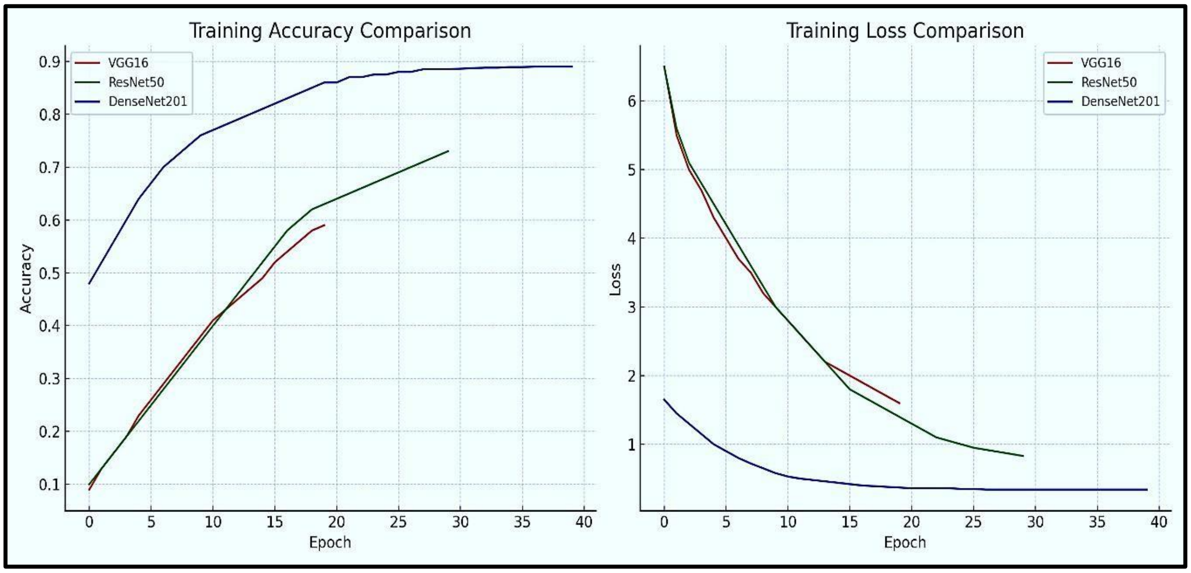


Figure 15. combined plots comparing the accuracy and loss over epochs

Figure 15 shows a comparison of the training performance of three deep learning models- VGG16, ResNet50, and DenseNet201- using a framework/retrieval model for image captioning. Figure 15 has two subplots, one for the accuracy and the other for the loss based on the training epochs. The left subplot shows the training accuracy of the models based on the number of epochs. As shown in the left subplot, DenseNet201 has the highest training accuracy and continues to improve over the number of epochs. ResNet50 tends to follow a similar path, but eventually outperformed VGG16, which was relatively slower in the learning process. The right subplot shows the training loss reduction by the models for the same number of epochs. Apart from differences in model sizes, all three models showed a steady reduction in loss, indicating clear learning was taking place. Again, DenseNet201 achieved the fastest convergence to the lowest loss value, while VGG16 was the weakest and slowest at getting learning to take place. This comparative subplot shows the models' relative effectiveness in learning the associations between images and captions. It demonstrates that DenseNet201 is better in terms of reducing error amount and improving accuracy, and is the best model of the three for image captioning.

4.4.5 Visual Output and Text-to-Speech Integration

We also created real-time visual and audio outputs in addition to a quantitatively based evaluation. The system was tested with numerous handheld compositions of images, and the descriptions were shown, and used the "gTTS" program to convert the text to speech; the TTS audio was full of well-understandable, clear audio in a natural- sounding voice, and the text/voice pairing was flawless. The availability of multimodal output enriched the user's interaction with the system, aided accessibility for all users, and marginalized more so for visual impairments.

4.4.6 Deployment and User Interaction

The final version of the model (DenseNet201 + LSTM + gTTS) was made available for use by the user on a simple and intuitive web front end. The user could upload any image to the web front end, and it would immediately display a caption for the image and read the caption aloud. The front end had been tested in different environments and produced similar results each time.

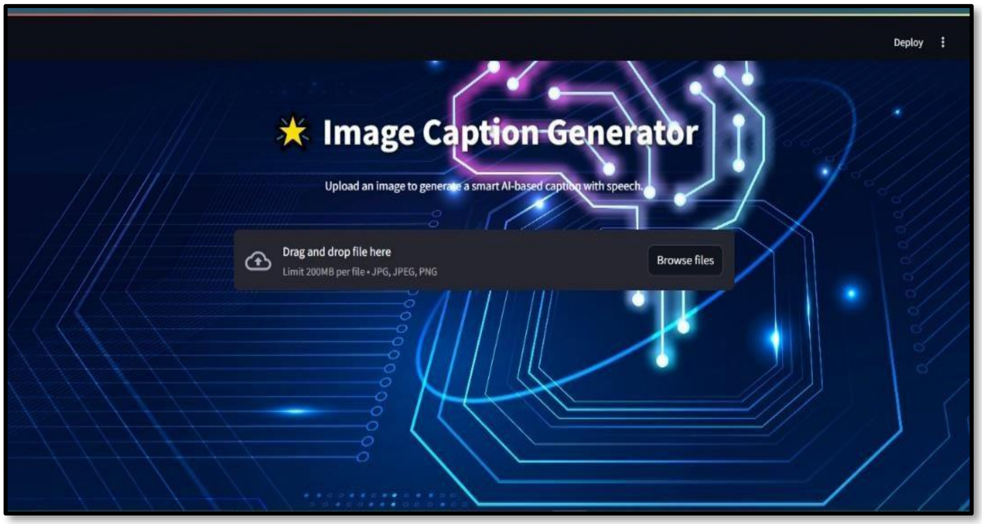
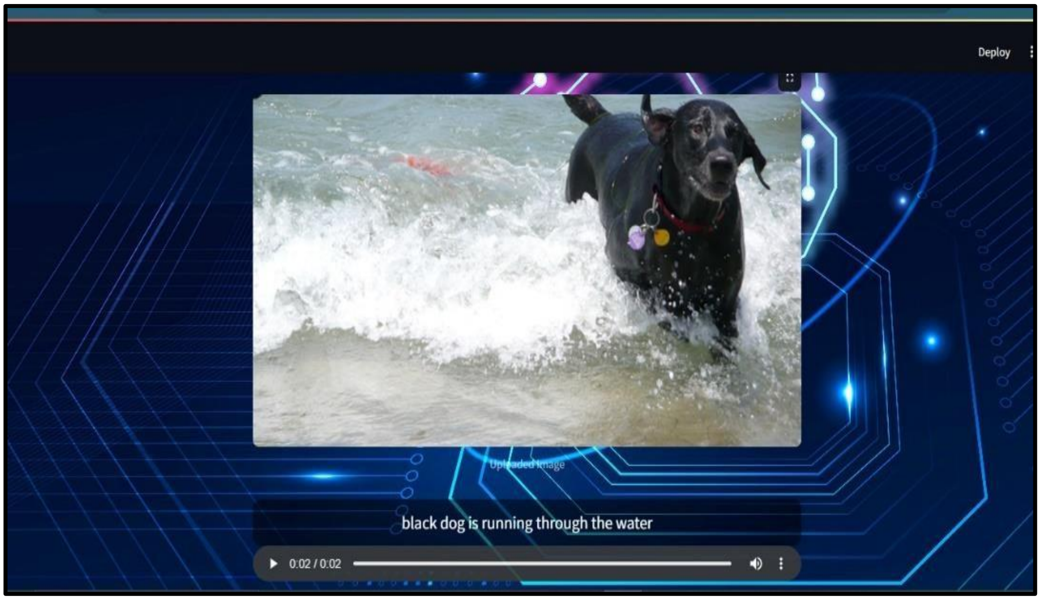


Image Caption Generator

Upload an image to generate a smart AI-based caption with speech.

Figure 16. UI Interface

Figure 16 shows the user interface of the deployed Image Caption Generator web application. The user interface makes it easy to upload images in JPG, JPEG, or PNG (up to 200MB) formats. After uploading, the application performs image processing to generate smart AI-generated captions, including speech output to make the user experience interactive and user-friendly.



black dog is running through the water

Figure 17. Output Interface

Figure 17 displays the deployment interface of the image captioning system. There is a black dog running through the water in an uploaded image on the user interface. Below the image, the system generates a caption-like natural language "black dog is running through the water" that describes the visual content. This result indicates the capability of the model to understand image features and thus generate contextually appropriate captions. Besides, this setup also has a playback feature for the audio, indicating a text-to-speech function. This gives a modernized and technology-oriented look through a background with circuit patterns. The AI and deep learning foundation of the system is emphasized by this setup. This setup reflects a successful integration of computer vision and natural language processing.

5 CONCLUSION AND FUTURE SCOPE

One of the most interesting advances in technology today is the ability of machines to analyze images and generate descriptions with natural language. Our project, a deep learning image caption generator using CNN, LSTM, and gTTS, is an example of how AI can analyze an image, generate an intelligible English caption, and convert the caption into speech. We see this system as being beneficial in education, content creation, and for use by visually impaired people. Using the Flickr8k dataset, we produced captions that were accurate and grammatically correct, as our results showed. There is, however, plenty of room for improvement. In the future, our model could be trained on larger datasets such as MSCOCO or Visual Genome, which would improve the model's ability to learn to understand diverse scenes and complex objects. Extending the capabilities of our model to allow for real-time captioning would also open the possibilities of use in live environments such as surveillance and assistive technology. Real-Time Text-to-Speech that involves the use of Transformer-based models has shown better quality output than traditional RNN models. Hence, the sentences generated would tend to have higher fluency and comply more consistently with grammatical structures, and they would be more contextually accurate. Also, creating a multi-lingual capability would allow this tool to be usable by people worldwide. Therefore, packaging our system into a mobile or web application would also promote availability to a wider range of users. Overall, this project lays the groundwork for impactful future innovation.

We identify various future avenues for the potential growth and scalability of our work.

i. **Integration Attention Mechanism Integration:** Adding attention between CNN and LSTM can help the model focus on relevant image areas during caption generation, improving contextual accuracy.

ii. **Multilingual Captioning & Voice Output:** Incorporating multilingual tokenizers and TTS engines would allow broader global use by supporting various languages.

iii. **Real-Time Captioning:** Optimizing model size and reducing inference time can enable real-time performance, suitable for mobile and embedded systems.

iv. **Video Captioning**: Expanding to multimodal data like video will require temporal analysis using 3D CNNs or Transformer architectures for scene-level descriptions.

v. **Training on Larger Datasets:** Using datasets like MSCOCO or Visual Genome, and incorporating domain-specific images, can enhance performance and generalization.

vi. **Assistive Technology Integration:** Embedding the system into mobile or wearable devices can help visually impaired users with real-time audio feedback.

vii. **Improved Evaluation Metrics**: Implementing BLEU, METEOR, CIDEr, and user-feedback scoring can help better assess and refine output quality.

viii. **Interactive UI and API:** Developing a web/mobile interface with upload, voice output, and feedback options, along with a public API, enhances usability.

ix. **Adaptive Learning:** Adding online learning features enables the model to evolve continuously with new data and user feedback.

6 REFERENCES

[1] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3156–3164.

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

[3] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in Proc. Eur. Conf. Computer Vision (ECCV), 2014, pp. 740–755.

[4] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3128–3137.

[5] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4700–4708.

[6] J. Johnson, A. Karpathy, and L. Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4565–4574.

[7] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2018, pp. 6077–6086.

[8] S. Sharma, R. Kiros, and R. Salakhutdinov, "Show, Control and Tell: A Framework for Generating Controllable and Grounded Captions," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2018.

[9] K. Desai and J. Johnson, "VirTex: Learning Visual Representations from Textual Annotations," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2021.

[10] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, and J. Gao, "Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks," in Proc. Eur. Conf. Computer Vision (ECCV), 2020, pp. 121–137.

[11] Flicker8k Dataset, Date of Visit: 10th June 2025, URL: https://www.kaggle.com/datasets/adityajn105/flickr8k/data

[12] K. Hossain, F. Sohel, M. Shiratuddin, and H. Laga, “Comprehensive survey of deep learning for image captioning,” ACM Comput. Surv., vol. 51, no. 6, pp. 1–36, 2019.

[13] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, “Describing videos by exploiting temporal structure,” in Proc. IEEE Int. Conf. Computer Vision (ICCV), 2015, pp. 4507–4515.

[14] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books,” in Proc. IEEE Int. Conf. Computer Vision (ICCV), 2015, pp. 19–27.

7 Reflection of the Team Members on the Project

Through this project on Image Captioning Generator using CNN and LSTM, we gained valuable knowledge in combining computer vision with natural language processing. By using the Flickr8k dataset and implementing a deep learning model that integrates DenseNet201 for image feature extraction with an LSTM network for sentence generation, we explored the complexities of designing models that can "see" and "describe" images. Each team member contributed their skills, and this collaborative effort led to the successful development of our model. Below is a summary of our individual contributions.

I, Nigmananda Panda (21416090), worked primarily on implementing the core architecture of the model. I handled the integration of DenseNet201, a powerful pre-trained convolutional neural network, to extract high-level visual features from images in the Flickr8k dataset. I ensured that the model's output from DenseNet201 was correctly reshaped and connected to the LSTM layer. I also tuned model hyperparameters such as embedding size, LSTM units, and dropout to improve performance. My focus was on ensuring that the image processing pipeline was accurate and compatible with the sequence generation component.

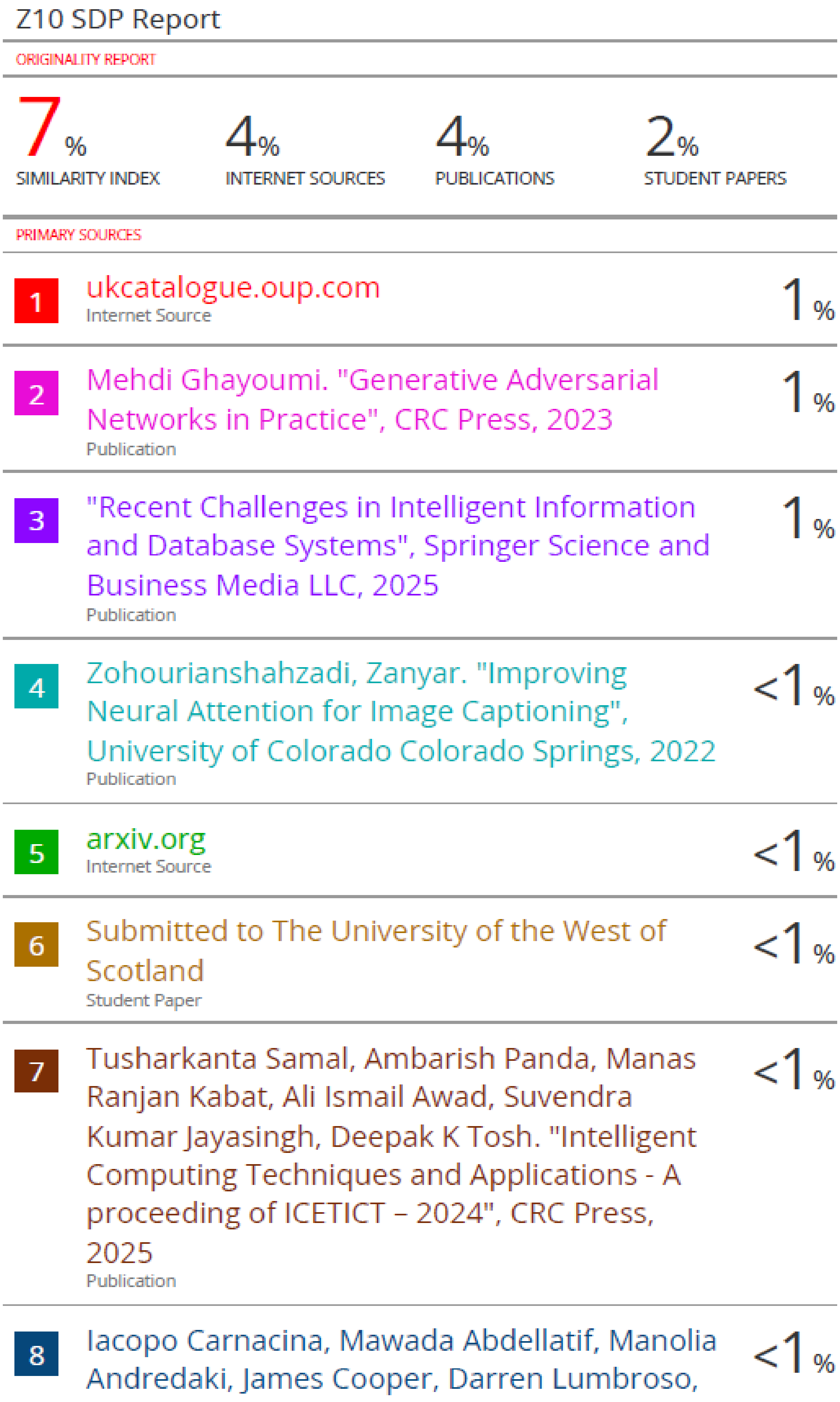
I, Akashdeep Behera (2141002072), concentrated on data preprocessing and feature preparation. I processed the captions by applying tokenization, sequence padding, and vocabulary construction. I also ensured the Flickr8k dataset was loaded correctly, with proper mapping between image IDs and their associated captions. I contributed to converting words into numerical sequences using tokenizers and prepared training input suitable for the LSTM model. I also implemented techniques to manage overfitting and optimized input dimensions to improve training efficiency.

I, Ganesh Chandra Das (2141004078), was involved in dataset management and visualization. I helped structure the Flickr8k dataset for training and validation, ensuring the correct association between each image and its multiple captions. I worked closely with Akashdeep to prepare mini-batches for training and monitored model performance using loss and accuracy curves. I also implemented visualization tools using Matplotlib to display generated captions alongside images, which made it easier to analyze how the model was interpreting visual content.

I, Madhab Narayan Mohanty (2141013300), focused on model evaluation and interpretability. I assessed the model's performance using metrics such as training loss and prediction accuracy. I manually reviewed a sample of generated captions to evaluate their grammatical correctness and relevance. I also created evaluation scripts to compare predicted captions against reference captions from the Flickr8k dataset. My work ensured that the results could be clearly interpreted and presented during reviews and documentation.

By working on this project, we developed a deeper understanding of how deep learning models can bridge the gap between visual and textual data. Using DenseNet201 for robust feature extraction and LSTM for caption generation allowed us to build a functional image captioning system. The Flickr8k dataset provided a reliable foundation for training and evaluation. Most importantly, this experience improved our teamwork, technical confidence, and our ability to design an end-to- end AI System.

8 SIMILARITY REPORT



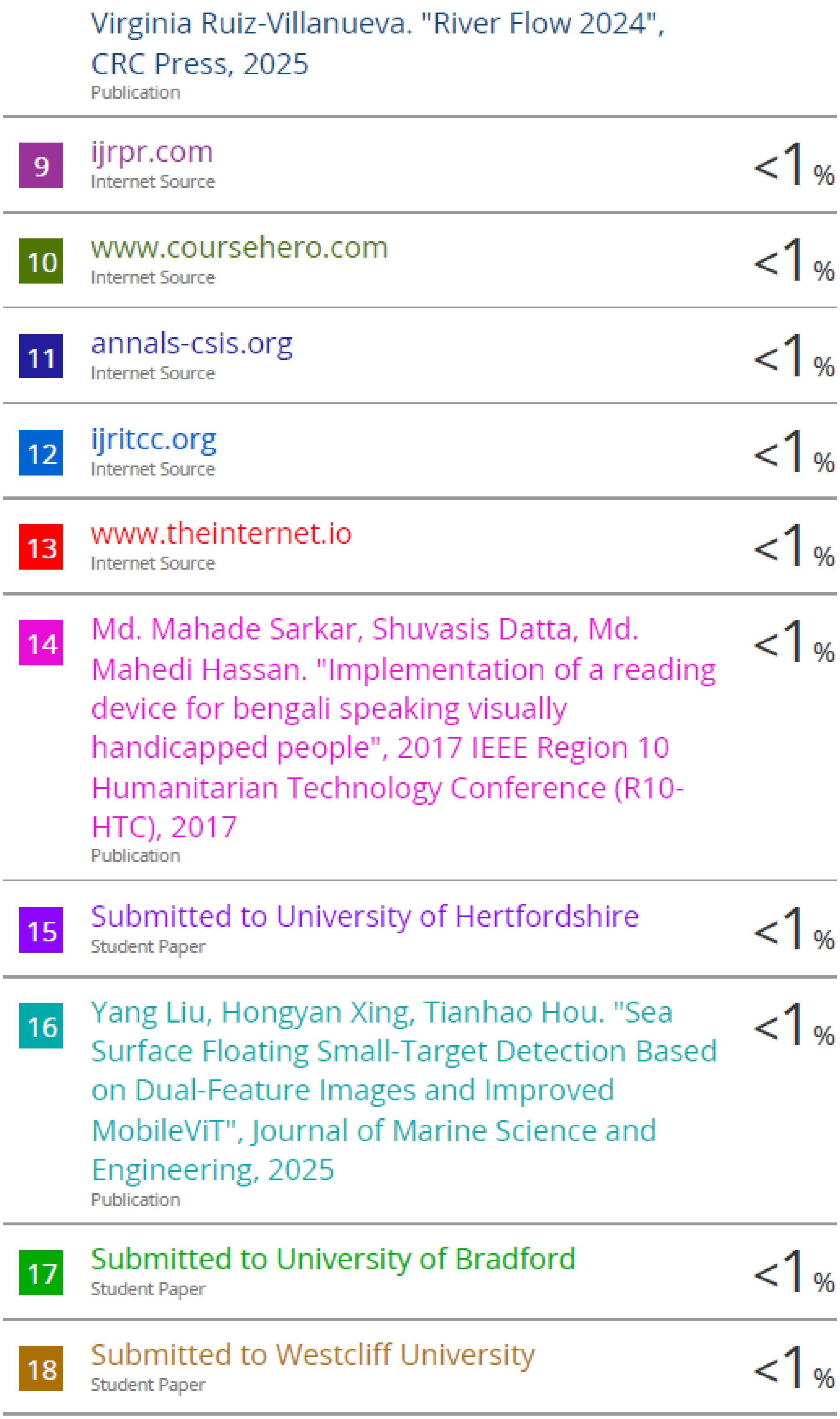
Z10 SDP Report

ORIGINALITY REPORT

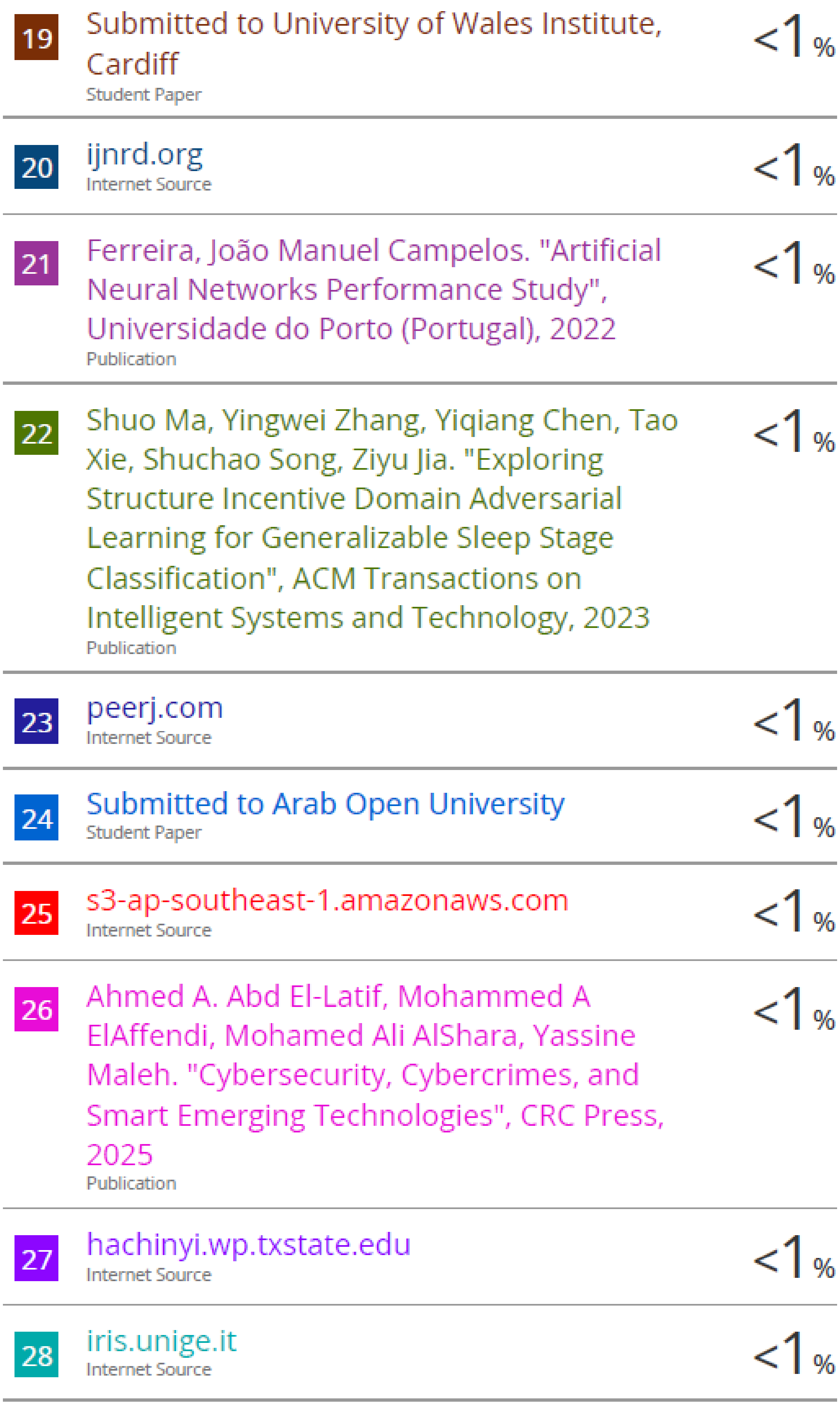
7%	4%	4%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ukcatalogue.oup.com Internet Source	1%
2	Mehdi Ghayoumi. "Generative Adversarial Networks in Practice", CRC Press, 2023 Publication	1%
3	"Recent Challenges in Intelligent Information and Database Systems", Springer Science and Business Media LLC, 2025 Publication	1%
4	Zohourianshahzadi, Zanyar. "Improving Neural Attention for Image Captioning", University of Colorado Colorado Springs, 2022 Publication	<1%
5	arxiv.org Internet Source	<1%
6	Submitted to The University of the West of Scotland Student Paper	<1%
7	Tusharkanta Samal, Ambarish Panda, Manas Ranjan Kabat, Ali Ismail Awad, Suvendra Kumar Jayasingh, Deepak K Tosh. "Intelligent Computing Techniques and Applications - A proceeding of ICETICT – 2024", CRC Press, 2025 Publication	<1%
8	Iacopo Carnacina, Mawada Abdellatif, Manolia Andredaki, James Cooper, Darren Lumbroso,	<1%



	Virginia Ruiz-Villanueva. "River Flow 2024", CRC Press, 2025 Publication	
9	ijrpr.com Internet Source	<1 %
10	www.coursehero.com Internet Source	<1 %
11	annals-csis.org Internet Source	<1 %
12	ijritcc.org Internet Source	<1 %
13	www.theinternet.io Internet Source	<1 %
14	Md. Mahade Sarkar, Shuvasis Datta, Md. Mahedi Hassan. "Implementation of a reading device for bengali speaking visually handicapped people", 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), 2017 Publication	<1 %
15	Submitted to University of Hertfordshire Student Paper	<1 %
16	Yang Liu, Hongyan Xing, Tianhao Hou. "Sea Surface Floating Small-Target Detection Based on Dual-Feature Images and Improved MobileViT", Journal of Marine Science and Engineering, 2025 Publication	<1 %
17	Submitted to University of Bradford Student Paper	<1 %
18	Submitted to Westcliff University Student Paper	<1 %



19	Submitted to University of Wales Institute, Cardiff Student Paper	<1 %
20	ijnrd.org Internet Source	<1 %
21	Ferreira, João Manuel Campelos. "Artificial Neural Networks Performance Study", Universidade do Porto (Portugal), 2022 Publication	<1 %
22	Shuo Ma, Yingwei Zhang, Yiqiang Chen, Tao Xie, Shuchao Song, Ziyu Jia. "Exploring Structure Incentive Domain Adversarial Learning for Generalizable Sleep Stage Classification", ACM Transactions on Intelligent Systems and Technology, 2023 Publication	<1 %
23	peerj.com Internet Source	<1 %
24	Submitted to Arab Open University Student Paper	<1 %
25	s3-ap-southeast-1.amazonaws.com Internet Source	<1 %
26	Ahmed A. Abd El-Latif, Mohammed A ElAffendi, Mohamed Ali AlShara, Yassine Maleh. "Cybersecurity, Cybercrimes, and Smart Emerging Technologies", CRC Press, 2025 Publication	<1 %
27	hachinyi.wp.txstate.edu Internet Source	<1 %
28	iris.unige.it Internet Source	<1 %

29	www.arxiv-vanity.com Internet Source	<1 %
30	www.mdpi.com Internet Source	<1 %
31	www.zora.uzh.ch Internet Source	<1 %
32	"Computer Vision – ECCV 2020", Springer Science and Business Media LLC, 2020 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		