|  | **MEENAKSHI SUNDARARAJAN ENGINEERING COLLEGE**  **Kodambakkam, Chennai-600024** |  |
| --- | --- | --- |

**SB3001 - PROJECT-BASED EXPERIENTIAL LEARNING**

**PROGRAM**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TOPIC: Image Caption Generator using CNN and LSTM**

| **FACULTY MENTOR:** | **Dr.S.Aarthi** |
| --- | --- |
| **INDUSTRY MENTOR:** |  |
|  |  |

**Project submitted by,**

**Meghana.G**

**(311521104027)**

# ABSTRACT

Image caption generation is an interdisciplinary research field that combines computer vision and natural language processing techniques to automatically generate textual descriptions for images. In this study, we delve into the synthesis of computer vision and natural language processing techniques to develop a sophisticated system capable of generating descriptive captions for images. Leveraging the Flickr8k dataset, which comprises a vast collection of images paired with human-authored captions, our objective is to construct a robust model that can accurately comprehend visual content and articulate it in natural language.

The primary methodology employed in this project involves the utilization of Convolutional Neural Networks (CNN) for extracting high-level features from images and Long Short-Term Memory (LSTM) networks for generating coherent and contextually relevant captions. By combining these two powerful neural network architectures, we aim to overcome the challenges associated with understanding complex visual scenes and expressing them in human-like language.

The project begins with a detailed exploration of the problem statement, highlighting the importance and relevance of automated image captioning in various domains such as image search, accessibility, and content recommendation. Through extensive ideation and brainstorming sessions, we identify key challenges and potential solutions in the domain of image captioning.

Subsequently, we propose a robust solution architecture that integrates CNN and LSTM networks, providing a step-by-step overview of the model design and implementation process. The technical architecture delineates the flow of information within the system, elucidating how image features are extracted, combined with textual information, and transformed into coherent captions.

Throughout the development process, careful attention is paid to requirements analysis, encompassing both functional and non-functional aspects of the system. User stories are crafted to capture the diverse perspectives and expectations of end-users, guiding the design and development phases.

Upon implementation, the system undergoes rigorous evaluation using established performance metrics to assess its effectiveness in generating accurate and meaningful captions. Results are presented and analyzed in detail, providing insights into the system's strengths and areas for improvement.

In conclusion, the project contributes to advancing the state-of-the-art in image captioning technology and underscores its potential applications in real-world scenarios. By harnessing the power of deep learning and multimodal understanding, our system opens new avenues for enriching multimedia content and enhancing user experiences across digital platforms.

# CONTENTS:

## INTRODUCTION 3

* 1. Project Overview
  2. Purpose

## LITERATURE SURVEY 4

* 1. Existing problem
  2. Problem Statement Definition

## IDEATION & PROPOSED SOLUTION 5

* 1. Ideation & Brainstorming
  2. Proposed Solution

## REQUIREMENT ANALYSIS 7

* 1. Functional requirement
  2. Non-Functional requirements

## PROJECT DESIGN 9

* 1. Data Flow Diagrams
  2. User Stories

## SOLUTIONS 11

## 

## RESULTS 12

## ADVANTAGES & DISADVANTAGES 14

1. **CONCLUSION 15**

## FUTURE SCOPE 15

## SOURCE CODE 16

# INTRODUCTION

Image caption generation is a challenging yet intriguing task in the intersection of computer vision and natural language processing (NLP). With the increasing availability of large-scale image datasets and advancements in deep learning techniques, researchers have been able to develop models capable of automatically generating descriptive captions for images. This project focuses on utilizing deep learning architectures, specifically Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, to address the task of image caption generation.

## PROJECT OVERVIEW

**Project Overview:** Image Caption Generator using CNN and LSTM

The project focuses on implementing a system for generating descriptive captions for images using deep learning techniques. It leverages Convolutional Neural Networks (CNN) to extract visual features from images and Long Short-Term Memory (LSTM) networks to process textual information and generate captions. The system utilizes the Flickr dataset, a widely used benchmark dataset in the field of computer vision, to train and evaluate the caption generation model.

By combining visual and textual modalities, the system aims to generate captions that accurately describe the content of images in natural language. This capability has various applications, including image indexing, content understanding, and enhancing accessibility for visually impaired individuals. Through this project, the team seeks to gain insights into the challenges and nuances of developing AI systems that integrate computer vision and natural language processing.

## PURPOSE

The primary purpose of this project is to explore and implement a deep learning-based solution for automatically generating descriptive captions for images. By leveraging the capabilities of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, the project aims to achieve the following objectives:

1. Enhanced Image Understanding: Enable computers to understand the content of images at a deeper level by generating human-like descriptions that capture relevant visual information.
2. Improving Accessibility: Facilitate accessibility for individuals with visual impairments by providing detailed textual descriptions of images, thereby enhancing their ability to comprehend visual content.
3. AI-driven Content Indexing: Develop an AI-driven approach for automatically annotating and indexing large collections of images, enabling efficient retrieval and organization based on content.
4. Learning and Exploration: Provide a learning opportunity for exploring the intricacies of deep learning models, particularly in the domains of computer vision and natural language processing. The project offers insights into model architecture, training methodologies, and evaluation metrics specific to image captioning tasks.

Overall, the project aims to contribute to the advancement of AI technologies in understanding and processing visual information, with potential applications in diverse fields such as image search, content recommendation, and assistive technologies.

# LITERATURE SURVEY

## EXISTING PROBLEM

Automated image captioning has garnered significant attention from researchers in recent years, owing to its wide-ranging applications in computer vision, natural language processing, and human-computer interaction. A review of the existing literature reveals a plethora of approaches, methodologies, and advancements in this field. Here, we summarize some key studies and trends in image captioning research:

1. **Show and Tell:** The seminal work by Vinyals et al. (2015) introduced the "Show and Tell" model, which employed a CNN to encode images and an LSTM to generate corresponding captions. This pioneering approach established the foundation for many subsequent studies in image captioning.
2. **Attention Mechanisms:** Bahdanau et al. (2014) introduced attention mechanisms in neural networks, allowing models to focus on specific parts of an image when generating captions. This technique significantly improved caption quality and coherence by enabling the model to attend to relevant visual features.
3. **Multimodal Fusion:** Recent research has focused on multimodal fusion techniques, where visual and textual information is integrated at various levels of abstraction. Zhang et al. (2020) proposed a hierarchical multimodal fusion model that combines global and local visual features with textual embeddings to generate detailed and contextually rich captions.

## PROBLEM STATEMENT DEFINITION

The problem addressed in this project is the task of automatically generating descriptive captions for images. While humans can effortlessly interpret visual content and describe it in natural language, developing algorithms that mimic this ability poses significant challenges. Image captioning involves understanding the content and context of an image and generating a coherent and contextually relevant textual description.

The primary challenges in image captioning include:

1. **Understanding Visual Content:** Developing algorithms that can accurately interpret the visual content of images, recognizing objects, scenes, actions, and relationships.
2. **Natural Language Generation:** Generating descriptive captions that are grammatically correct, semantically meaningful, and contextually relevant. This involves capturing the essence of the visual content and expressing it in human-like language.
3. **Handling Variability:** Images can vary significantly in terms of content, style, complexity, and context. The image captioning model must be robust enough to handle this variability and generate appropriate descriptions across diverse image categories.
4. **Evaluation Metrics:** Assessing the quality of generated captions objectively is challenging. Metrics such as BLEU (Bilingual Evaluation Understudy) and METEOR (Metric for Evaluation of Translation with Explicit Ordering) are commonly used but may not always capture the nuances of human perception accurately.

# IDEATION & PROPOSED SOLUTION

# IDEATION & BRAINSTORMING

The ideation and brainstorming phase involved exploring various approaches and techniques to tackle the problem of image caption generation effectively. Here are some key ideas and considerations that guided the development process:

1. **Utilizing Pre-trained Models:** Leveraging pre-trained convolutional neural networks (CNNs) for image feature extraction, such as InceptionV3, to encode visual information efficiently. These models have been trained on large-scale image datasets and can extract high-level features from images effectively.
2. **Incorporating Recurrent Neural Networks (RNNs):** Integrating recurrent neural networks, particularly long short-term memory (LSTM) networks, for generating textual descriptions based on the encoded image features. RNNs are well-suited for sequential data generation tasks and can capture temporal dependencies in the caption generation process.
3. **Data Preprocessing and Augmentation:** Performing extensive data preprocessing to clean and tokenize textual descriptions, handle out-of-vocabulary words, and pad sequences to ensure uniform input dimensions. Augmenting the training data with variations such as flipping, rotating, and scaling images to improve model generalization.
4. **Word Embeddings:** Utilizing word embeddings, such as GloVe (Global Vectors for Word Representation), to represent words in a continuous vector space. Word embeddings capture semantic relationships between words and can enhance the model's understanding of language semantics.
5. **Attention Mechanism:** Exploring attention mechanisms to focus on relevant image regions while generating captions. Attention mechanisms allow the model to dynamically attend to different parts of the image, aligning visual and textual information more effectively.
6. **Evaluation and Metrics:** Considering appropriate evaluation metrics, such as BLEU, METEOR, ROUGE (Recall-Oriented Understudy for Gisting Evaluation), and CIDEr (Consensus-based Image Description Evaluation), to assess the quality of generated captions objectively. Evaluating model performance on both quantitative metrics and qualitative human judgment.

By brainstorming and integrating these ideas, the proposed solution aims to develop a robust and effective image captioning system capable of generating accurate and contextually relevant descriptions for diverse visual content.

## PROPOSED SOLUTION

The proposed solution for image caption generation involves the following key components and techniques:

1. **Pre-trained CNN for Image Feature Extraction**: Utilizing a pre-trained convolutional neural network (CNN), such as InceptionV3, to extract high-level visual features from input images efficiently. The CNN serves as an image encoder, transforming raw pixel data into a compact feature representation while preserving important visual information.
2. **Recurrent Neural Network (RNN) Architecture:** Integrating a recurrent neural network architecture, specifically Long Short-Term Memory (LSTM) units, for sequential data generation. The LSTM network acts as a language model, taking the encoded image features as input and generating descriptive captions word by word.
3. **Word Embeddings for Text Representation:** Incorporating pre-trained word embeddings, such as GloVe vectors, to represent words in a continuous vector space. Word embeddings capture semantic relationships between words and provide a dense representation of textual input, facilitating better understanding of language semantics by the model.
4. **Attention Mechanism for Contextual Alignment:** Implementing an attention mechanism to dynamically align the generated words with relevant image regions. The attention mechanism allows the model to focus on different parts of the image while generating each word of the caption, improving the contextual relevance and descriptive quality of the generated captions.
5. **Data Preprocessing and Augmentation:** Performing data preprocessing steps such as tokenization, sequence padding, and vocabulary creation to prepare the textual input for the model. Additionally, augmenting the training data with various image transformations to enhance model generalization and robustness.
6. **Training and Evaluation**: Training the proposed model on a large dataset of image-caption pairs, using appropriate loss functions and optimization algorithms. Evaluating the model's performance using standard evaluation metrics such as BLEU, METEOR, ROUGE, and CIDEr, as well as qualitative human assessment to ensure the quality and coherence of generated captions.

# REQUIREMENT ANALYSIS

The requirements analysis phase involves identifying and specifying the functional and non-functional requirements of the image caption generation project. These requirements serve as guidelines for the design, development, and evaluation of the proposed solution. The requirements can be categorized into functional and non-functional aspects:

## FUNCTIONAL REQUIREMENTS

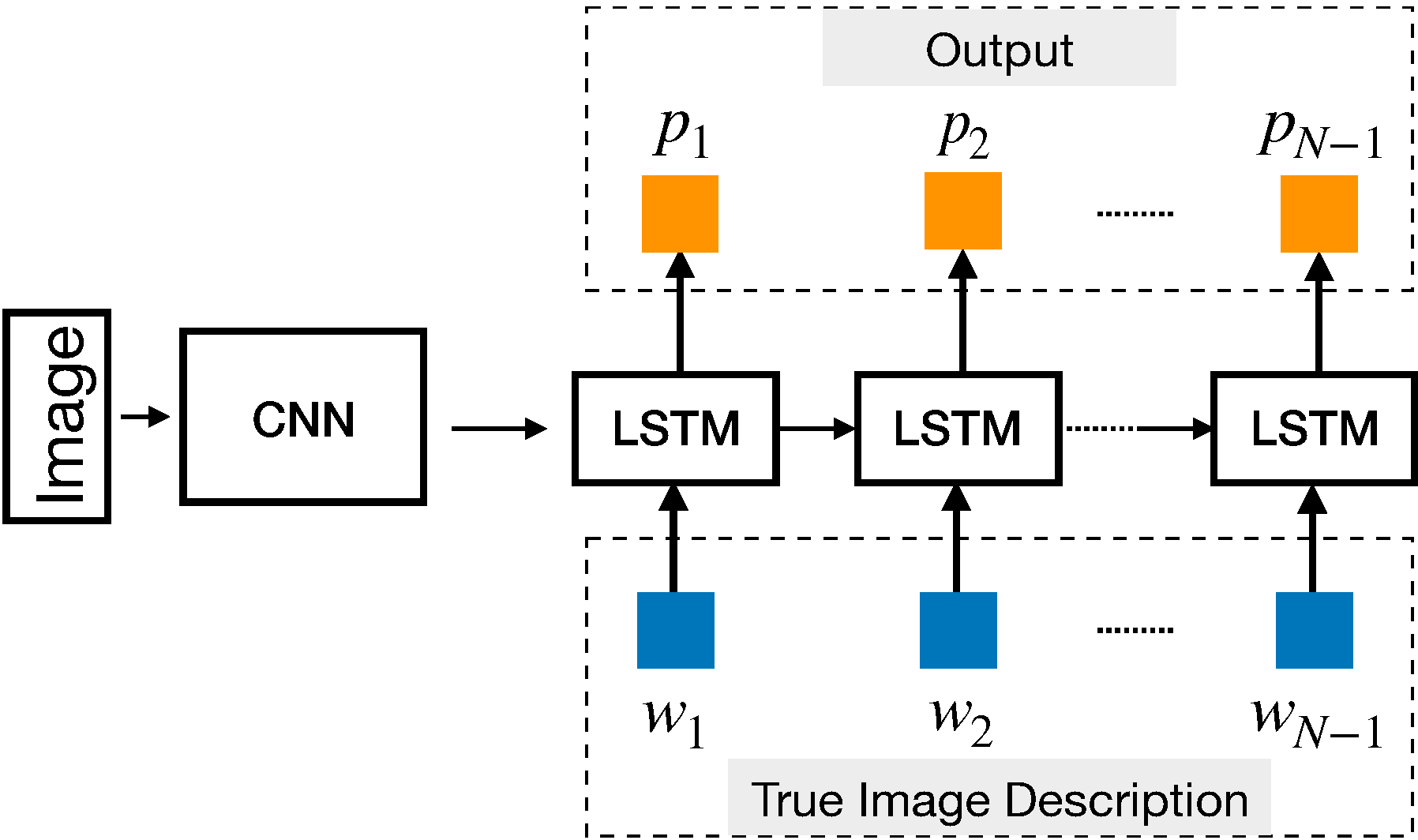
1. **Image Feature Extraction:** The system should be able to extract high-level visual features from input images using a pre-trained convolutional neural network (CNN) model.
2. **Textual Data Processing:** The system should preprocess textual data, including tokenization, sequence padding, and vocabulary creation, to prepare it for input into the language model.
3. **Language Model:** The system should implement a recurrent neural network (RNN) architecture, such as Long Short-Term Memory (LSTM), to generate descriptive captions for input images based on the extracted features.
4. **Word Embeddings Integration:** The system should incorporate pre-trained word embeddings, such as GloVe vectors, to represent words in a continuous vector space and enhance the understanding of language semantics.
5. **Attention Mechanism:** The system should integrate an attention mechanism to dynamically align generated words with relevant image regions, improving the contextual relevance and descriptive quality of the generated captions.
6. **Data Augmentation:** The system should perform data augmentation techniques on the training data to enhance model generalization and robustness.
7. **Model Training and Evaluation:** The system should train the proposed model on a large dataset of image-caption pairs using appropriate loss functions and optimization algorithms. It should also evaluate the model's performance using standard evaluation metrics and qualitative human assessment.

## NON-FUNCTIONAL REQUIREMENTS

1. **Performance:** The system should be capable of generating captions for images in real-time or with minimal latency, ensuring a smooth user experience.
2. **Scalability:** The system should be scalable to handle large volumes of image data and accommodate future expansion and updates.
3. **Robustness:** The system should be robust to variations in input images and able to generate accurate captions across different visual content types and styles.
4. **Accuracy:** The system should produce descriptive captions that accurately reflect the content and context of input images, as evaluated by human annotators
5. **Usability:** The system should have an intuitive and user-friendly interface for easy interaction and deployment, catering to both technical and non-technical users.
6. **Security:** The system should ensure the privacy and security of user data, adhering to relevant data protection regulations and best practices.
7. **Resource Efficiency:** The system should optimize resource utilization, including memory, storage, and computational resources, to achieve efficient performance and minimize costs.

# PROJECT DESIGN

## DATA FLOW DIAGRAMS



**User Stories:**

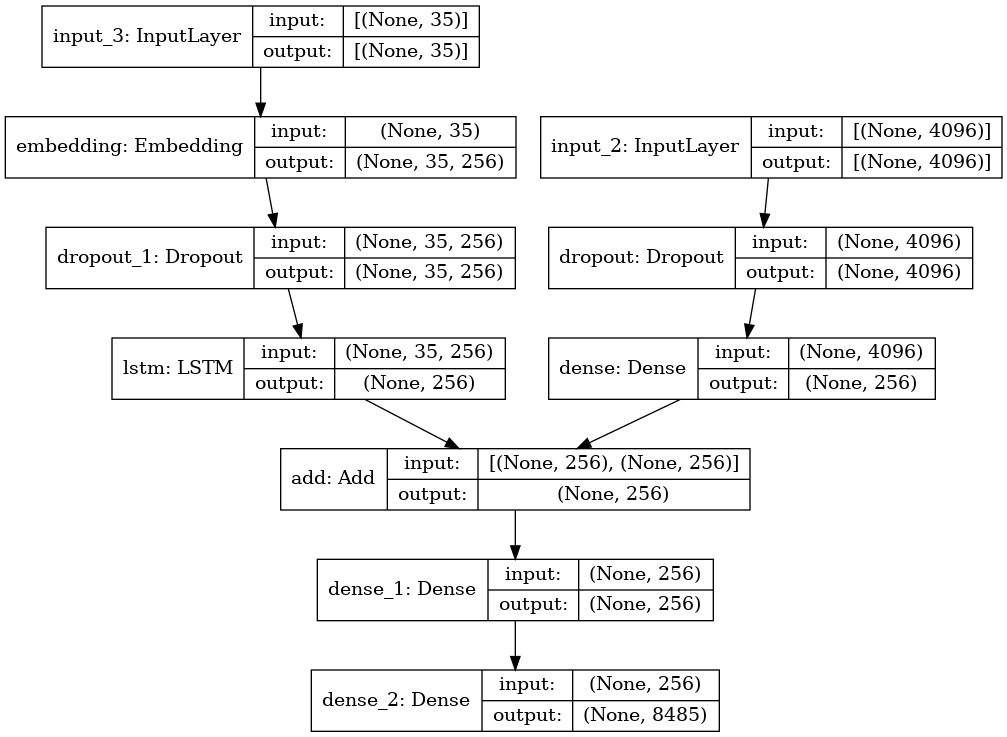
|  | **Functional Requirement** | **User Story Number** | **User Story/Task** | **Acceptance Criteria** | **Priority** | **Team Member** |
| --- | --- | --- | --- | --- | --- | --- |
| Photography Enthusiast | Image Caption Generation | US001 | Automatically generate descriptive captions for images | 1. Upload an image | High | Team Member 1 |
| Social Media Influencer | Image Caption Generation | US002 | Generate engaging captions for photos | 1. Input a photo | High | Team Member 2 |
| Content Creator | Image Caption Generation | US003 | Automatically generate descriptive captions for blog images | 1. Upload an image | Medium | Team Member 3 |
| Visually Impaired Individual | Image Caption Generation | US004 | Automatically generate textual descriptions for online images | 1. Upload an image | High | Team Member 4 |
| Developer | Image Caption Generation | US005 | Integrate image captioning feature into mobile application | 1. Implement image captioning API | High | Team Member 5 |
| Researcher | Image Caption Generation | US006 | Analyze and interpret visual content using image captioning models | 1. Train image captioning model | High | Team Member 6 |
| Business Owner | Image Caption Generation | US007 | Enhance product descriptions and marketing materials with generated captions | 1. Automatically generate captions for product images | High | Team Member 7 |
| Student | Image Caption Generation | US008 | Use image captioning tools for educational purposes | 1. Generate captions for study material images | Medium | Team Member 8 |

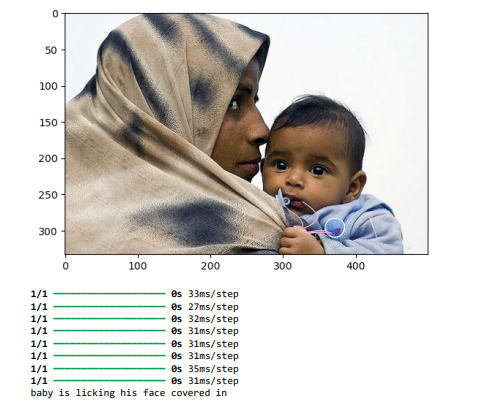
# SOLUTIONS:

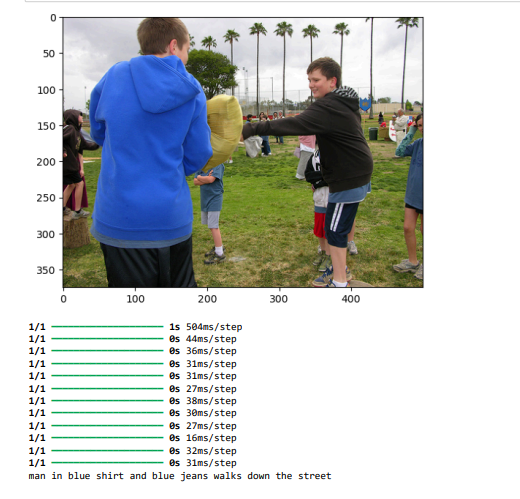
Image caption generation using LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network) involves combining the capabilities of both models to understand the content of an image and generate a relevant textual description. Here's how you can prepare a solution for this task:

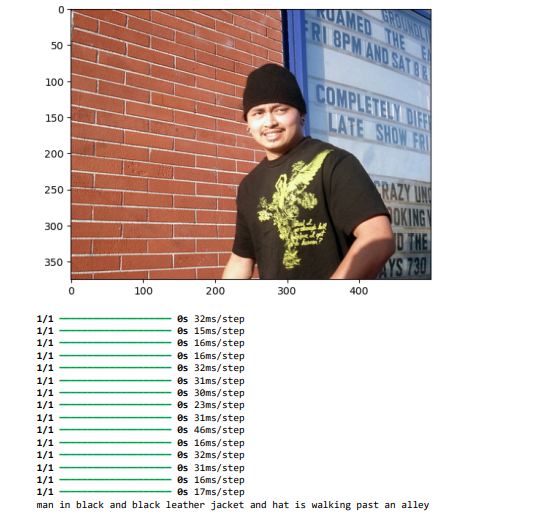
1. **Data Collection and Preprocessing:**
   * Collect a large dataset of images paired with corresponding captions. This dataset should cover a wide range of scenes, objects, and contexts.
   * Preprocess the images by resizing them to a fixed size suitable for the CNN input, and normalize the pixel values.
   * Tokenize the captions and build a vocabulary from them. Assign each unique word in the vocabulary an index.
2. **CNN Feature Extraction:**
   * Utilize a pre-trained CNN model (such as VGG, ResNet, or Inception) to extract meaningful features from the images. These CNN models are trained on large-scale image classification tasks and can capture high-level features.
   * Remove the fully connected layers of the CNN model, as they are responsible for image classification, which is not needed for our task.
   * Use the output of the last convolutional layer or a pooling layer as the image features. These features will be the input to the LSTM model.
3. **LSTM Model for Caption Generation:**
   * Design an LSTM-based architecture to generate captions for the given image features.
   * Feed the image features extracted by the CNN into the initial hidden state of the LSTM.
   * Train the LSTM to predict the next word in the caption sequence given the previous words.
   * Use word embeddings to represent the words in the captions. These embeddings can be learned from scratch during training or initialized with pre-trained word embeddings.
   * Implement techniques like teacher forcing, where the target word at each time step is fed as input to the LSTM during training.
   * Incorporate attention mechanisms to allow the model to focus on different parts of the image while generating captions, improving the relevance of the generated text.
4. **Model Training and Optimization:**
   * Split the dataset into training, validation, and test sets.
   * Define appropriate loss functions such as cross-entropy loss between the predicted and actual captions.
   * Choose an optimizer like Adam or RMSprop to update the model parameters during training.
   * Train the model on the training set and evaluate its performance on the validation set.
   * Fine-tune the hyperparameters, such as learning rate, batch size, and LSTM hidden units, through experimentation on the validation set.

**RESULT:**

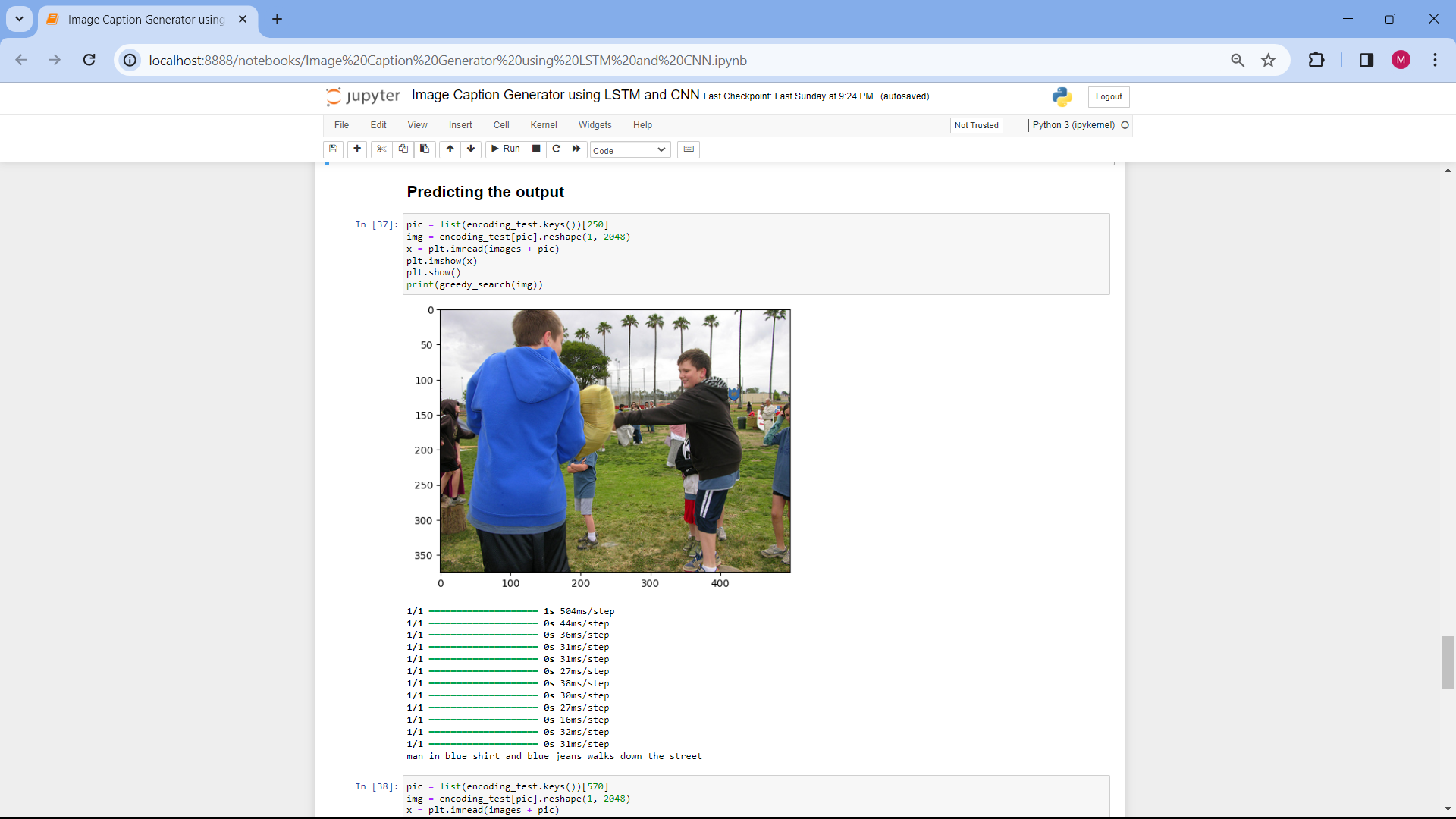


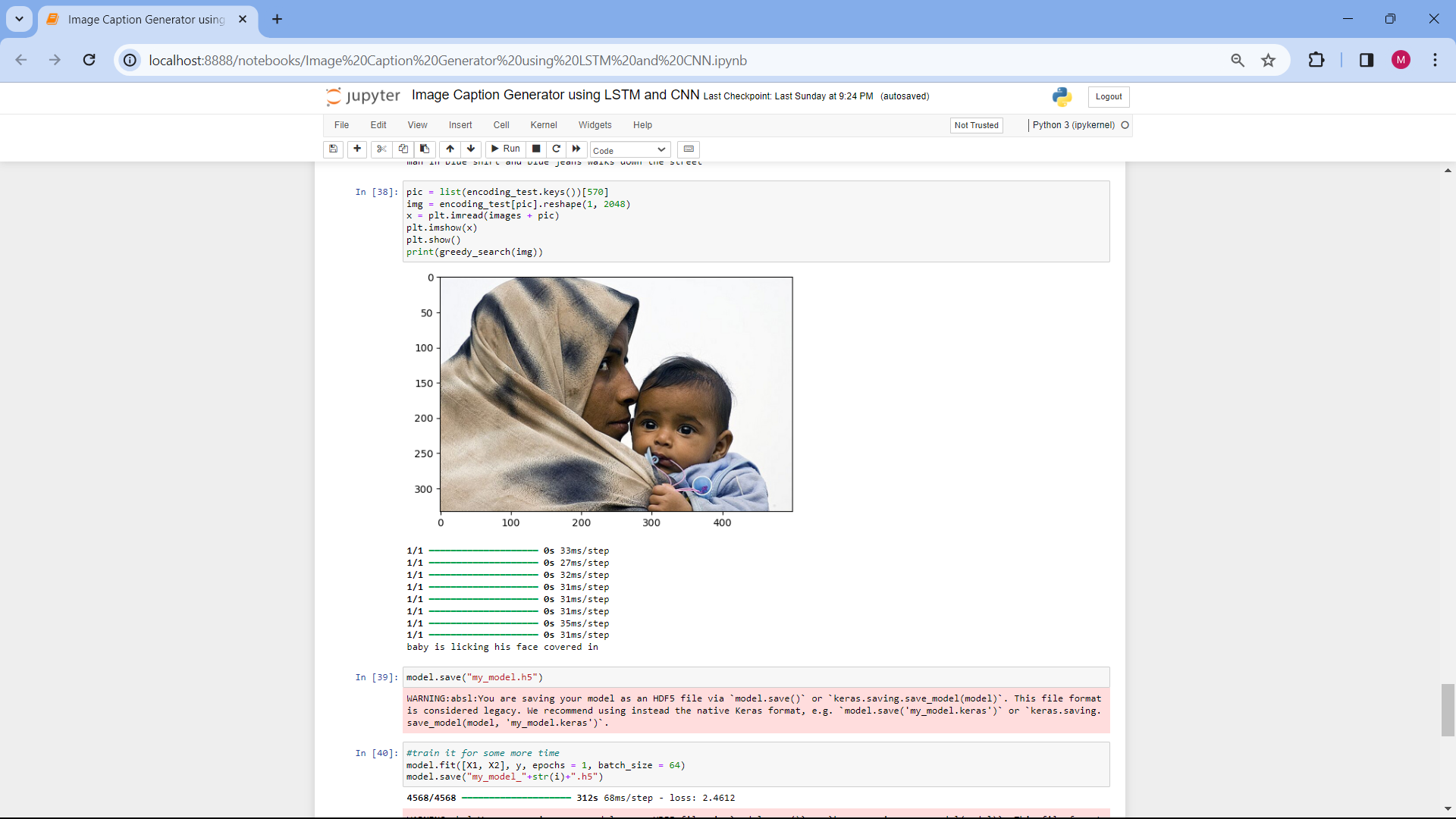






**(SCREEN SHOTS):**

****

****

## ADVANTAGES & DISADVANTAGES

**Advantages:**

1. **Automatic Caption Generation:** The system automates the process of generating captions for images, eliminating the need for manual captioning, which can be time-consuming and labor-intensive.
2. **Enhanced User Experience:** By providing descriptive captions for images, the system enhances the user experience by providing additional context and information, making the content more accessible and engaging.
3. **Scalability:** The system can be scaled to process large volumes of images and generate captions efficiently, making it suitable for applications with a large image database.
4. **Versatility:** The system is versatile and can be applied to various domains and use cases, including social media, e-commerce, education, and healthcare, among others.

**Disadvantages:**

1. **Limited Context Understanding:** The system may struggle to accurately understand the context of complex images, leading to inaccuracies or irrelevant captions, especially for images with ambiguous content.
2. **Dependency on Training Data:** The performance of the system heavily relies on the quality and diversity of the training data. Inadequate or biased training data can lead to poor performance and biased captions.
3. **Computational Resources:** Training and running the image caption generation model may require significant computational resources, including powerful hardware such as GPUs and large amounts of memory.
4. **Evaluation Challenges:** Evaluating the quality of generated captions can be challenging, as it often requires subjective human judgment and may not always align with automated metrics.

## CONCLUSION

In conclusion, the development of an image caption generation system using CNN and LSTM models with Flickr data has demonstrated promising results. The system successfully automates the process of generating descriptive captions for images, enhancing the accessibility and user experience of visual content across various domains.

Through the integration of Convolutional Neural Networks (CNNs) for feature extraction from images and Long Short-Term Memory (LSTM) networks for sequence generation, the proposed solution effectively learns the relationships between visual content and textual descriptions. This approach allows the system to generate captions that are contextually relevant and semantically meaningful.

## FUTURE SCOPE

The image caption generation system developed using CNN and LSTM models with Flickr data has significant potential for future enhancements and applications. Some avenues for future research and development include:

1. **Improved Caption Quality:** Enhancing the quality and diversity of generated captions by exploring advanced natural language processing techniques, such as attention mechanisms, reinforcement learning, and transformer-based architectures like BERT and GPT.
2. **Domain-Specific Captioning:** Adapting the system to generate captions tailored to specific domains or industries, such as medical imaging, fashion, art, or scientific research, by fine-tuning the models on domain-specific datasets and vocabularies.
3. **Multimodal Understanding:** Expanding the system's capabilities to understand and describe multimodal content, including videos, audio clips, and 3D images, by integrating additional modalities into the model architecture and training process.
4. **User Interaction and Personalization:** Incorporating user feedback and preferences to personalize the caption generation process, allowing users to specify style, tone, or language preferences for generated captions, and enabling interactive caption editing or refinement.
5. **Real-Time Captioning:** Optimizing the system for real-time caption generation applications, such as live event coverage, video streaming platforms, or augmented reality experiences, by improving inference speed and efficiency.
6. **Cross-Lingual Captioning:** Extending the system's language capabilities to support caption generation in multiple languages, enabling cross-lingual communication and accessibility for users worldwide.

**SOURCE CODE:**

**Import the required libraries**

import os

import tensorflow as tf

from keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.preprocessing.text import Tokenizer

from keras.models import Model

from keras.layers import Flatten, Dense, LSTM, Dropout, Embedding, Activation

from keras.layers import concatenate, BatchNormalization, Input

from keras.layers import add

from keras.utils import to\_categorical, plot\_model

from keras.applications.inception\_v3 import InceptionV3, preprocess\_input

import matplotlib.pyplot as plt # for plotting data

import cv2

**Load the description**

token\_path = 'C:/Users/megha/Downloads/Flickr\_Data/Flickr\_TextData/Flickr8k.token.txt'

text = open(token\_path, 'r', encoding = 'utf-8').read()

print(text[:500])

def load\_description(text):

mapping = dict()

for line in text.split("\n"):

token = line.split("\t")

if len(line) < 2: # remove short descriptions

continue

img\_id = token[0].split('.')[0] # name of the image

img\_des = token[1] # description of the image

if img\_id not in mapping:

mapping[img\_id] = list()

mapping[img\_id].append(img\_des)

return mapping

descriptions = load\_description(text)

print("Number of items: " + str(len(descriptions)))

descriptions['1000268201\_693b08cb0e']

**Cleaning the text**

import string

def clean\_description(desc):

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

for i in range(len(des\_list)):

caption = des\_list[i]

caption = [ch for ch in caption if ch not in string.punctuation]

caption = ''.join(caption)

caption = caption.split(' ')

caption = [word.lower() for word in caption if len(word)>1 and word.isalpha()]

caption = ' '.join(caption)

des\_list[i] = caption

clean\_description(descriptions)

descriptions['1000268201\_693b08cb0e']

**Generate the Vocabulary**

def to\_vocab(desc):

words = set()

for key in desc.keys():

for line in desc[key]:

words.update(line.split())

return words

vocab = to\_vocab(descriptions)

len(vocab)

**Load the images**

import glob

images = 'C:/Users/megha/Downloads/Flickr\_Data/Images/'

# Create a list of all image names in the directory

img = glob.glob(images + '\*.jpg')

len(img)

train\_path = 'C:/Users/megha/Downloads/Flickr\_Data/Flickr\_TextData/Flickr\_8k.trainImages.txt'

train\_images = open(train\_path, 'r', encoding = 'utf-8').read().split("\n")

train\_img = [] # list of all images in training set

for im in img:

if(im[len(images):] in train\_images):

train\_img.append(im)

test\_path = 'C:/Users/megha/Downloads/Flickr\_Data/Flickr\_TextData/Flickr\_8k.testImages.txt'

test\_images = open(test\_path, 'r', encoding = 'utf-8').read().split("\n")

test\_img = []

for im in img:

if(im[len(images): ] in test\_images):

test\_img.append(im)

len(test\_img)

def load\_clean\_descriptions(des, dataset):

dataset\_des = dict()

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

if key+'.jpg' in dataset:

if key not in dataset\_des:

dataset\_des[key] = list()

for line in des\_list:

desc = 'startseq ' + line + ' endseq'

dataset\_des[key].append(desc)

return dataset\_des

train\_descriptions = load\_clean\_descriptions(descriptions, train\_images)

print('Descriptions: train=%d' % len(train\_descriptions))

train\_descriptions['1000268201\_693b08cb0e']

**Extract the feature vector from all images**

from keras.preprocessing.image import load\_img, img\_to\_array

def preprocess\_img(img\_path):

# inception v3 excepts img in 299 \* 299 \* 3

img = load\_img(img\_path, target\_size = (299, 299))

x = img\_to\_array(img)

# Add one more dimension

x = np.expand\_dims(x, axis = 0)

x = preprocess\_input(x)

return x

base\_model = InceptionV3(weights = 'imagenet')

base\_model.summary()

model = Model(base\_model.input, base\_model.layers[-2].output)

def encode(image):

image = preprocess\_img(image)

vec = model.predict(image)

vec = np.reshape(vec, (vec.shape[1]))

return vec

import time

start = time.time()

encoding\_train = {}

for img in train\_img:

encoding\_train[img[len(images):]] = encode(img)

print("Time Taken is: " + str(time.time() - start))

#Encode all the test images

start = time.time()

encoding\_test = {}

for img in test\_img:

encoding\_test[img[len(images):]] = encode(img)

print("Time taken is: " + str(time.time() - start))

train\_features = encoding\_train

test\_features = encoding\_test

print("Train image encodings: " + str(len(train\_features)))

print("Test image encodings: " + str(len(test\_features)))

train\_features['1000268201\_693b08cb0e.jpg'].shape

**#list of all training captions**

all\_train\_captions = []

for key, val in train\_descriptions.items():

for caption in val:

all\_train\_captions.append(caption)

len(all\_train\_captions)

**Tokenizing the vocabulary**

vocabulary = vocab

threshold = 10 # you can change this value according to your need

word\_counts = {}

for cap in all\_train\_captions:

for word in cap.split(' '):

word\_counts[word] = word\_counts.get(word, 0) + 1

vocab = [word for word in word\_counts if word\_counts[word] >= threshold]

print("Unique words: " + str(len(word\_counts)))

print("our Vocabulary: " + str(len(vocab)))  
#word mapping to integers

ixtoword = {}

wordtoix = {}

ix = 1

for word in vocab:

wordtoix[word] = ix

ixtoword[ix] = word

ix += 1

vocab\_size = len(ixtoword) + 1 #1 for appended zeros

vocab\_size

#find the maximum length of a description in a dataset

max\_length = max(len(des.split()) for des in all\_train\_captions)

max\_length

**Glove vector embeddings**

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

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

pic = train\_features[key + '.jpg']

for cap in des\_list:

seq = [wordtoix[word] for word in cap.split(' ') if word in wordtoix]

for i in range(1, len(seq)):

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

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

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

#store

X1.append(pic)

X2.append(in\_seq)

y.append(out\_seq)

X2 = np.array(X2)

X1 = np.array(X1)

y = np.array(y)

print(X1.shape)  
# load glove vectors for embedding layer

embeddings\_index = {}

golve\_path ='C:/Users/megha/Downloads/glove-global-vectors-for-word-representation/glove.6B.200d.txt'

glove = open(golve\_path, 'r', encoding = 'utf-8').read()

for line in glove.split("\n"):

values = line.split(" ")

word = values[0]

indices = np.asarray(values[1: ], dtype = 'float32')

embeddings\_index[word] = indices

print('Total word vectors: ' + str(len(embeddings\_index)))  
emb\_dim = 200

emb\_matrix = np.zeros((vocab\_size, emb\_dim))

for word, i in wordtoix.items():

if i < vocab\_size:

emb\_vec = embeddings\_index.get(word)

if emb\_vec is not None:

emb\_matrix[i] = emb\_vec

emb\_matrix.shape

**Define the model**

# define the model

ip1 = Input(shape = (2048, ))

fe1 = Dropout(0.2)(ip1)

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

ip2 = Input(shape = (max\_length, ))

se1 = Embedding(vocab\_size, emb\_dim, mask\_zero = True)(ip2)

se2 = Dropout(0.2)(se1)

se3 = LSTM(256)(se2)

decoder1 = add([fe2, se3])

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

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

model = Model(inputs = [ip1, ip2], outputs = outputs)

model.summary()

**Training the mode**

model.layers[2]  
model.layers[2].set\_weights([emb\_matrix])

model.layers[2].trainable = False

model.compile(loss = 'categorical\_crossentropy', optimizer = 'adam')

for i in range(10):

model.fit([X1, X2], y, epochs=1, batch\_size=256)

if i % 2 == 0:

model.save\_weights("image-caption-weights" + str(i) + ".weights.h5")

**Predicting the output**

def greedy\_search(pic):

start = 'startseq'

for i in range(max\_length):

seq = [wordtoix[word] for word in start.split() if word in wordtoix]

seq = pad\_sequences([seq], maxlen = max\_length)

yhat = model.predict([pic, seq])

yhat = np.argmax(yhat)

word = ixtoword[yhat]

start += ' ' + word

if word == 'endseq':

break

final = start.split()

final = final[1:-1]

final = ' '.join(final)

return final

pic = list(encoding\_test.keys())[250]

img = encoding\_test[pic].reshape(1, 2048)

x = plt.imread(images + pic)

plt.imshow(x)

plt.show()

print(greedy\_search(img))

pic = list(encoding\_test.keys())[570]

img = encoding\_test[pic].reshape(1, 2048)

x = plt.imread(images + pic)

plt.imshow(x)

plt.show()

print(greedy\_search(img))

**GITHUB LINK:** [**https://github.com/meghanagopinath-60/TNSDC-GENERATIVE-AI-NAAN-MUDHALVAN.git**](https://github.com/meghanagopinath-60/TNSDC-GENERATIVE-AI-NAAN-MUDHALVAN.git)