IMAGE CAPTION GENERATOR USING DEEP LEARNING

***A project report submitted to***

***MALLA REDDY UNIVERSITY***

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

## Bachelor of Technology

**in**

**COMPUTER SCIENCE & ENGINEERING (AI & ML)**

**Submitted by**

|  |  |  |
| --- | --- | --- |
| **Sufiyan Ali Khan** | **:** | **2111CS020573** |
| **G Swetha** | **:** | **2111CS020584** |
| **G Thejashwini** | **:** | **2111CS020589** |
| **H Varshith**  **M Varun Teja** | **:** | **2111CS020621 2111CS020626** |

***Under the Guidance of*** **Prof. A. Siva Kumar** Professor

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)**





**COLLEGE CERTIFICATE**

This is to certify that this is the bonafide record of the application development entitled,”IMAGE CAPTION GENERATOR USING DEEP LEARNING”

submitted by **Sufiyan Ali Khan(2111CS020573), G Swetha(2111CS020584), G Thejashwini (2111CS020589), H Varshith(2111CS020621), M VarunTeja(2111CS020626)** of B.Tech lII year IInd semester, Department of CSE (AI&ML) during the year 2023- 24.The results embodied in thereport have not been submitted to any other university or institute for the award of any degree or diploma.

|  |  |
| --- | --- |
| **INTERNAL GUIDE**  **A Siva Kumar Professor** | **HEAD OF THE DEPARTMENT**  **Dr. Thayaba Khatoon Professor & HoD**  **EXTERNAL EXAMINER** |
|  |  |

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
| 1. | INTRODUCTION: | 1 |
|  | 1.1 Project Identification / Problem Definition |
|  | 1.2 Objective of project |
|  | 1.3 Scope of the project |
| 2. | ANALYSIS: | 2-5 |
|  | 2.1 Project Planning and Research |
|  | 2.2 Software requirement specification |
|  | 2.2.1 Software requirement |
|  | 2.2.2 Hardware requirement |
|  | 2.3 Model Selection and Architecture |
| 3. | DESIGN: | 6-9 |
|  | 3.1 Introduction |
|  | 3.2 DFD/ER/UML diagram |
|  | 3.3 Data Set Description |
|  | 3.4 Data Preprocessing Techniques |
|  | 3.5 Methods and Algorithms |
| 4. | DEPLOYMENT AND RESULTS: | 10-18 |
|  | 4.1 Introduction |
|  | 4.2 Source Code |
|  | 4.3 Model Implementation and Training |
|  | 4.4 Model Evaluation Metrics |
|  | 4.5 Results |
| 5. | CONCLUSION: | 19 |
|  | 5.1 Project Conclusion |
|  | 5.2 Project Conclusion |
|  | 5.3 References |

# 1.Introduction

# Problem Statement:

Image captioning presents a significant challenge in Artificial Intelligence, requiring systems to understand images and generate grammatically correct descriptions. The existing methods facing issues in accurately describing images with proper context. To address this, a hybrid system Convolutional Neural Networks (CNN) for feature extraction and Long Short-Term Memory (LSTM) networks for caption generation is proposed. However, the existing system struggling to capture suitable image differences and contextual nuances. Therefore, there is a need for an improved image captioning system that effectively integrates CNN and LSTM models to accurately describe images with contextual relevance. This research aims to develop such a system, utilizing the Flickr8K dataset for training and evaluation. The system efficiency will increased by generating suitable captions for given images.BLEU Score is used as a metric to evaluate the performance of the trained model.

# Objective of project:

The objective of the project is to predict the captions for the input image. The dataset consists of 8k images and 5 captions for each image. The features are extracted from both the image and the text captions for input. The features will be concatenated to predict the next word of the caption. CNN is used for image and LSTM is used for text. BLEU Score is used as a metric to evaluate the performance of the trained model.

# Scope & Limitations of the project:

* + - Lack of understanding context.
    - Limited Dataset coverage
    - Difficulty with Ambiguity
    - Inability to Incorporate External Knowledge

1

# 2.ANALYSIS

# 2.1 Project Planning and Research

* [1] One of the research study presents a model employing pre-trained deep learning, specifically the VGG model, for generating captions from images. By comparing the model's output with human-provided captions, it achieves an accuracy of around 75%, indicating a close resemblance between generated and human captions. This approach enhances the capability of automated caption generation, bridging the gap between machine-generated and human-provided descriptions of images.
* [2] The image caption generator in few studies utilizes the Flickr\_8k database, comprising 8000 diverse images, each with five captions. The dataset is divided into 6000 training, 1000 validation, and 1000 testing images. Through rigorous training and testing, the model effectively generates accurate captions. It employs a combination of Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), where CNN acts as the encoder and RNN as the decoder, ensuring grammatically correct captions with appropriate labels.
* [3] In an image caption generator, the VGG16 model serves as a sophisticated filter for images, identifying crucial features such as shapes and objects. These features form a summary of the picture, which is then utilized by another component of the system to generate a descriptive sentence. VGG16 essentially aids in comprehending the content of the image, allowing the caption generator to translate this understanding into words.
* [4] Some studies presents a framework for generating descriptive captions from images, utilizing the Flickr8K dataset containing 8000 images, each paired with five descriptions. The model employs a neural network to automatically analyze images and generate English captions. The generated captions are categorized into error-free descriptions, descriptions with minimal errors, somewhat related descriptions, and unrelated descriptions, showcasing the system's ability to produce diverse outputs based on image content.
* [5] The integration of VGG16, LSTM, and CNN in image caption generation is significant for enhanced performance. VGG16 acts as a robust feature extractor, capturing detailed information from images. LSTM complements VGG16 by generating coherent captions, leveraging its sequential data understanding. The synergy between CNN and LSTM addresses challenges in combining visual and linguistic information, leading to a comprehensive image understanding. Ongoing research may unveil novel approaches, refining the synergy between these components for improved image captioning.

**2.2 Software Requirement Specification**

**2.2.1 Software Reqiurements**

* **Python:** Python is the primary programming language used for this project. Ensure you have Python installed on your system.
* **Jupiter Notebook/Google Collab**: Interactive environments for coding.

**2.2.2 Hardware Requirements**

* **Processor**: Multi-core processor (i5 or higher) for faster computation.
* **RAM**: Minimum 8GB RAM is recommended for handling moderate-sized datasets and model training.
* **Storage**: Adequate disk space to store datasets, libraries, and generated model files

**2.3 Model Selection and Architecture**

* **Image Feature Extraction:**

For image feature extraction, the project employs the VGG16 network. VGG16 is a pre-trained CNN model that has shown excellent performance in image classification tasks. It extracts high-level features from input images, which are then used as input to the caption generation model

* **Text Caption Generation:**

The text caption generation part of the model utilizes LSTM networks. LSTM networks are a type of recurrent neural network (RNN) that are well-suited for sequence prediction tasks. In this case, LSTM is used to generate captions word by word based on the extracted image feature.

The complete system is a combination of three models which optimizes the whole procedure of caption description from an image. The models are:

### Feature Extraction Model :

The model uses a VGG16 architecture to efficiently extract the features from the images using a combination of multiple 3\*3 convolution layers. The output of a VGG16 network would be vectors of size 1\*4096, which are used to represent the features of the images.

### Encoder Model :

The encoder model, is primarily responsible for processing the captions of each image fed while training. The output of the encoder model is again vectors of size 1\*256 which would again be an input to the decoder sequences Initially the captions present with each images are tokenized ie the words in the sentences are converted to integers so that the neural network can process them efficiently.

### Decoder model:

The decoder model, is basically the model which concatenates both the feature extraction model and encoder model and produces the required output which is the predicted word given an image and the sentence generated till that point of time.

**Architecture:**

* + - The CNN receives an image as input.
    - The CNN processes the image through multiple convolutional layers. Each layer uses filters to identify specific features in the image, like edges or shapes. The filters slide across the image, and their activations are stored in a feature map.
    - As the image goes through more convolutional layers, the feature maps become more complex, encoding increasingly intricate information about the image.
    - Finally, the CNN can use the learned features to perform a task, such as reconstructing the original image or recognizing objects within it.



Features

Image

Image

CNN

CNN Layer

CNN Layer

Feature

Map

Encoder

Powerful

VGG 16

Decoder

Output

LSTM

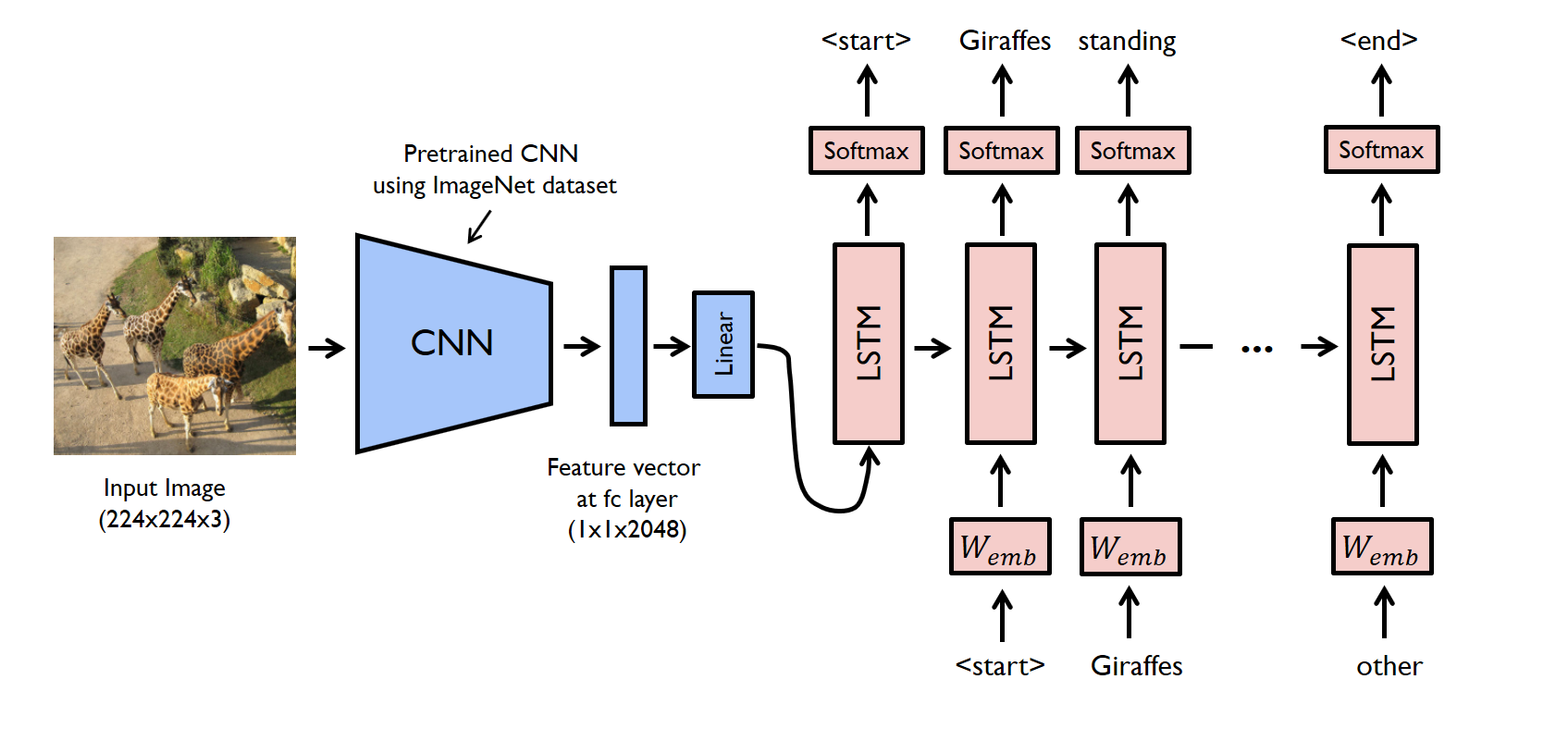
Input(images)

Image Caption

**3.DESIGN**

**3.1 Introduction**

During the design phase of an image caption generator using deep learning, careful consideration is given to the architecture and components required for effective caption generation. The process begins with selecting appropriate neural network architectures for both image feature extraction and text generation. Convolutional Neural Networks (CNNs) are typically chosen for image feature extraction due to their ability to capture hierarchical representations of visual features. Pre-trained CNN models like VGG16 or ResNet are often utilized to extract high-level image features efficiently. For text generation, recurrent neural networks (RNNs) or variants like Long Short-Term Memory (LSTM) networks are commonly employed. These networks are well-suited for sequential data generation tasks, making them suitable for generating captions word by word. The design phase also involves defining the data pipeline for preprocessing images and captions, including tasks such as image resizing, normalization, and tokenization of captions. Additionally, considerations are made for model evaluation metrics such as BLEU score, which measures the similarity between generated captions and ground truth captions. Overall, the design phase focuses on selecting appropriate network architectures, defining the data pipeline, and establishing evaluation metrics to ensure the effective generation of descriptive captions for input images.

* 1. **DFD/ER/UML diagram(any other project diagram)**
  2. **Data Set Descriptions**

The Flickr8k dataset utilized in this project consists of 8,000 images collected from the popular image-sharing platform Flickr. Each image in the dataset is associated with five different captions, providing diverse textual descriptions of the visual content. This multi-caption approach enhances the richness and variability of the dataset, enabling more robust training of the caption generation model. The images cover a wide range of scenes, objects, and activities, reflecting the diversity of visual content commonly found in natural images. The captions are typically descriptive and semantically rich, capturing various aspects of the depicted scenes. This diversity in both images and captions facilitates the training of a caption generation model capable of producing accurate and contextually relevant descriptions for a wide range of input images. Additionally, the dataset is publicly available, making it accessible for research and development purposes in the field of computer vision and natural language processing.

* 1. **Data Preprocessing Techniques**

Data preprocessing is a crucial step in preparing the Flickr8k dataset for training a deep learning model for image captioning. Here are some common techniques used for data preprocessing:

1**. Image Preprocessing**:

**Image Resizing:** Resize all images to a uniform size suitable for the model input. This ensures consistency in input dimensions and reduces computational overhead.

**Image Normalization:** Normalize pixel values to a common scale (e.g., [0, 1] or [-1, 1]) to facilitate convergence during training and improve model performance.

**Data Augmentation**: Augment the dataset by applying transformations such as rotation, flipping, cropping, and brightness adjustments. Data augmentation helps increase the diversity of the training data and improves the model's generalization ability.

2. **Text Preprocessing**:

**Tokenization**: Tokenize captions into individual words or subwords to represent them as sequences of discrete tokens. This step breaks down the text into its constituent elements, enabling the model to process them sequentially.

**Padding**: Pad or truncate captions to a fixed length to ensure uniformity in sequence length, which is necessary for batch processing in neural networks.

**Word Embeddings**: Convert tokens into dense vector representations using word embeddings such as Word2Vec, GloVe, or FastText. Word embeddings capture semantic relationships between words and provide distributed representations that capture contextual information.

**Vocabulary Creation**: Create a vocabulary of unique words present in the captions and map each word to an index. This vocabulary is used to convert words into numerical representations that can be processed by the model.

3. **Data Splitting:**

**Train-Validation-Test Split:** Divide the dataset into three disjoint subsets: training, validation, and test sets. The training set is used to train the model, the validation set is used for hyperparameter tuning and model evaluation during training, and the test set is used for final evaluation to assess the model's generalization performance.

4. **Data Loading:**

Implement data loaders or generators to efficiently load batches of images and their corresponding captions during training. This helps manage memory usage and facilitates parallel processing during training.

By applying these preprocessing techniques, the Flickr8k dataset can be effectively prepared for training a deep learning model for image captioning, leading to improved model performance and robustness.

# Methods & Algorithms

### Convolutional Neural Network (CNN):

Purpose: CNNs are primarily used for image analysis and feature extraction. Functionality:

* Multilayer Architecture: CNNs consist of multiple layers (convolutional, pooling, and fully connected) that learn hierarchical features from raw pixel data.
* Convolutional Layers: These layers apply convolutional filters to extract local patterns (edges, textures, shapes) from the input image.
* Pooling Layers: Reduce spatial dimensions while preserving important features.
* Fully Connected Layers: Process the extracted features and make predictions.
* Significance: CNNs can automatically learn relevant features from images, making them effective for tasks like object recognition and localization.

### Long Short-Term Memory (LSTM):

* Purpose: LSTMs are designed to handle sequential data, such as natural language.
* Functionality:
* Recurrent Architecture: LSTMs have recurrent connections that allow them to maintain memory over time steps.
* Cell State and Gates: LSTMs use a cell state to store information and three gates (input, forget, and output) to control the flow of information.
* Avoiding Vanishing Gradient Problem: LSTMs mitigate the vanishing gradient problem by allowing gradients to flow through time steps.
* Significance: LSTMs capture long-term dependencies and context, making them suitable for tasks like language modeling and sequence-to-sequence tasks.
  + 1. **Encoder-Decoder Model:**

Purpose : Combines an encoder (to process input data) and a decoder (to generate output sequences).

Functionality:

### Encoder (VGG16):

Uses a pre-trained CNN (e.g., VGG16) to encode input images into a fixed-length feature vector.

The feature vector captures high-level visual information.

### Decoder (LSTM):

Takes the encoded feature vector and generates captions word by word.

Learns to predict the next word based on context and previously generated words. Significance: Encoder-decoder models bridge the gap between visual and textual information, enabling image captioning.

**4. DEPLOYMENT AND RESULTS**

**4.1 Introduction**

Upon completion of the image caption generator project using the Flickr8k dataset, the focus shifts to deployment and evaluation of the model's performance. Deployment involves integrating the trained model into a user-friendly interface or application, allowing users to upload images and receive descriptive captions in real-time. This deployment phase aims to make the model accessible and usable by a wider audience, potentially benefiting various applications such as assistive technologies, content creation tools, and social media platforms. Additionally, rigorous evaluation of the model's performance is conducted to assess its effectiveness in generating accurate and contextually relevant captions. Metrics such as BLEU score, which measures the similarity between generated captions and ground truth captions, are used to quantitatively evaluate the model's performance.

**4.2 Source Code**

import os

import pickle

import numpy as np

from tqdm.notebook import tqdm

from tensorflow.keras.applications.vgg16 import VGG16, preprocess\_input

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

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Model

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

from tensorflow.keras.layers import Input, Dense, LSTM, Embedding, Dropout, add

BASE\_DIR = '/content/drive/My Drive/icg/flickr8k'

WORKING\_DIR = '/content/drive/My Drive/work'

!pip install Keras

from keras.applications import VGG16

from tensorflow.keras.models import Model

# load vgg16 model

model = VGG16()

# restructure the model

model = Model(inputs=model.inputs, outputs=model.layers[-2].output)

# summarize

print(model.summary())

# extract features from image

from tqdm import tqdm

!pip install tqdm

features = {}

directory = os.path.join(BASE\_DIR, 'Images')

for img\_name in tqdm(os.listdir(directory)):

# load the image from file

img\_path = directory + '/' + img\_name

image = load\_img(img\_path, target\_size=(224, 224))

# convert image pixels to numpy array

image = img\_to\_array(image)

# reshape data for model

image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))

# preprocess image for vgg

image = preprocess\_input(image)

# extract features

feature = model.predict(image, verbose=0)

# get image ID

image\_id = img\_name.split('.')[0]

# store feature

features[image\_id] = feature

# store features in pickle

pickle.dump(features, open(os.path.join(WORKING\_DIR, 'features.pkl'), 'wb'))

# load features from pickle

with open(os.path.join(WORKING\_DIR, 'features.pkl'), 'rb') as f:

features = pickle.load(f)

with open(os.path.join(BASE\_DIR, 'captions.txt'), 'r') as f:

next(f)

captions\_doc = f.read()

# create mapping of image to captions

mapping = {}

# process lines

for line in tqdm(captions\_doc.split('\n')):

# split the line by comma(,)

tokens = line.split(',')

if len(line) < 2:

continue

image\_id, caption = tokens[0], tokens[1:]

# remove extension from image ID

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

# convert caption list to string

caption = " ".join(caption)

# create list if needed

if image\_id not in mapping:

mapping[image\_id] = []

# store the caption

mapping[image\_id].append(caption)

def clean(mapping):

for key, captions in mapping.items():

for i in range(len(captions)):

# take one caption at a time

caption = captions[i]

# preprocessing steps

# convert to lowercase

caption = caption.lower()

# delete digits, special chars, etc.,

caption = caption.replace('[^A-Za-z]', '')

# delete additional spaces

caption = caption.replace('\s+', ' ')

# add start and end tags to the caption

caption = 'startseq ' + " ".join([word for word in caption.split() if len(word)>1]) + ' endseq'

captions[i] = caption

# before preprocess of text

mapping['1000268201\_693b08cb0e']

# create data generator to get data in batch (avoids session crash)

def data\_generator(data\_keys, mapping, features, tokenizer, max\_length, vocab\_size, batch\_size):

# loop over images

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

n = 0

while 1:

for key in data\_keys:

n += 1

captions = mapping[key]

# process each caption

for caption in captions:

# encode the sequence

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

# split the sequence into X, y pairs

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

# split into input and output pairs

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

# pad input sequence

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

# encode output sequence

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

# store the sequences

X1.append(features[key][0])

X2.append(in\_seq)

y.append(out\_seq)

if n == batch\_size:

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

yield [X1, X2], y

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

n = 0

# encoder model

# image feature layers

inputs1 = Input(shape=(4096,))

fe1 = Dropout(0.4)(inputs1)

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

# calcuate BLEU score

print("BLEU-1: %f" % corpus\_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))

print("BLEU-2: %f" % corpus\_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))

from PIL import Image

import matplotlib.pyplot as plt

def generate\_caption(image\_name):

# load the image

# image\_name = "1001773457\_577c3a7d70.jpg"

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

img\_path = os.path.join(BASE\_DIR, "Images", image\_name)

image = Image.open(img\_path)

captions = mapping[image\_id]

print('---------------------Actual---------------------')

for caption in captions:

print(caption)

# predict the caption

y\_pred = predict\_caption(model, features[image\_id], tokenizer, max\_length)

print('--------------------Predicted--------------------')

print(y\_pred)

plt.imshow(image)

generate\_caption("1001773457\_577c3a7d70.jpg")

**4.3 Model Implementation and Training**

Model implementation and training for the image caption generator using the Flickr8k dataset typically involves several steps, including data preparation, model construction, and training. Here's a high-level overview of the process:

**1. Data Preparation:**

- Load the Flickr8k dataset, including images and corresponding captions.

- Preprocess the images by resizing, normalization, and augmentation as necessary.

- Tokenize the captions and create a vocabulary mapping words to indices.

- Pad or truncate captions to ensure uniform length.

**2. Model Construction:**

- Implement the image feature extraction component using a pre-trained CNN model such as VGG16.

- Develop the text generation component using LSTM or a similar recurrent neural network architecture.

- Combine the image features extracted by the CNN with the text embeddings in the LSTM model.

- Define the architecture to process the concatenated vector and predict the next word in the caption sequence.

**3. Training:**

- Split the dataset into training, validation, and test sets.

- Set hyperparameters such as learning rate, batch size, and number of epochs.

- Train the model using the training set while validating on the validation set.

- Monitor metrics such as loss function and BLEU score to assess model performance.

- Implement techniques such as early stopping and learning rate scheduling to prevent overfitting and improve convergence.

- Fine-tune both the CNN and LSTM components jointly to optimize caption generation performance.

- Save the trained model weights and optimizer state for future use.

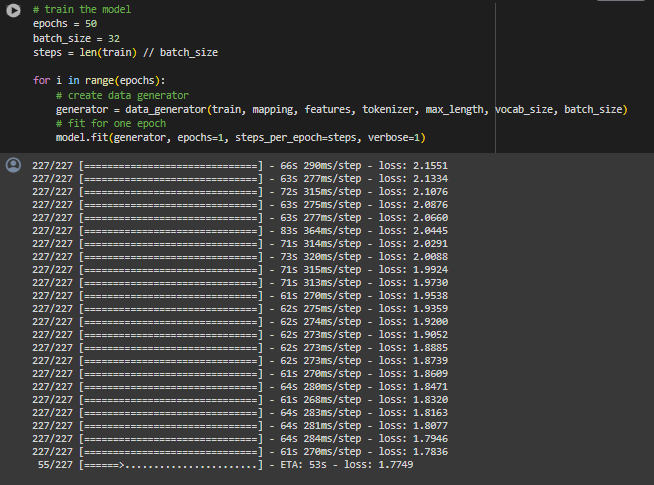
During training, it's essential to iterate and experiment with different architectures, hyperparameters, and optimization strategies to improve model performance. Additionally, monitoring training progress through visualizations and logging can provide insights into the model's behavior and guide further refinement. Once training is complete, the trained model can be evaluated on the test set to assess its performance in generating accurate and contextually relevant captions for unseen images.

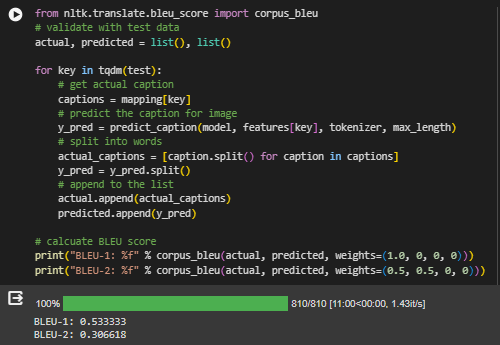
* 1. **Model Evaluation Metrics**

BLEU-1 Score: 0.544 BLEU-2 Score: 0.319

In evaluating the image caption generator model trained on the Flickr8k dataset, BLEU (Bilingual Evaluation Understudy) scores play a pivotal role. The BLEU score is a standard metric used to measure the quality of machine-generated text by comparing it to human-generated reference texts. In this project, the BLEU-1 score of 0.544 and BLEU-2 score of 0.319 indicate the effectiveness of the model in generating captions that closely match the reference captions provided in the dataset. The BLEU-1 score measures the overlap of unigrams (individual words) between the generated and reference captions, while the BLEU-2 score considers the overlap of bigrams (pairs of adjacent words). A BLEU-1 score of 0.544 suggests that more than half of the individual words in the generated captions match those in the reference captions on average. Similarly, a BLEU-2 score of 0.319 indicates that about a third of the bigram sequences in the generated captions coincide with those in the reference captions. These scores provide quantitative insights into the model's ability to produce captions that are semantically and syntactically similar to human-written captions, thereby demonstrating its efficacy in generating descriptive and contextually relevant captions for a diverse range of images.

* 1. **Results:**









**5. CONCLUSION**

**5.1 CONCLUSION**

We have successfully developed a deep-learning model using the Xception architecture to generate automatic captions. So far, most of the image captioning models have used inception (v3). We have famed our project in a web-based application using the Flask architecture. Some of the automatically generated captions based on our model are as shown below. Please note that generating captions is subjective and captions for the same image can differ from person to person. This is also the reason why the algorithm which is trained on human-typed captions can generate erratic results sometimes.

**5.2 FUTURE SCOPE**

The accuracy of the current project can be further increased by adding a weightage system to the vocabulary by assigning low weight to highly frequently occurring words. It can potentially create a better algorithm to generate captions.

**5.3 REFERENCES:**

[1]Dr.Jagadisha N ,Chaithra Rao(2022):Image Caption Generator Using Deep Learning

[2] Preksha Khant, Vishal Deshmukh, Aishwarya Kude, Prachi Kiraula (2021) : Image Caption Generator using CNN-LSTM.

[3] Ali Ashraf Mohamed (2020) : Image Caption Using CNN and LSTM.

[4] Chetan Amritkar, Vaishali Jabade (2018) : Image caption Generation Using Deep Learning Technique.

[5] Subrata Das, Lalit jain, Arup Das (2018) : Deep Learning for Military Image Captioning.

[6] Pranay Mathur , Aman Gill , Aayush Yadav , Anurag Mishra, Nand Kumar Bansode (2017): Camera2Caption :A Real-Time Caption Generator.

[7] Sreejith S P, Vijayakumar A (2021) : Image Captioning Generator using Deep Machine Learning.