

Report On

Image Caption Generator

Submitted in partial fulfillment of the requirements of the Course project in
Semester VII of Fourth Year Artificial Intelligence & Data Science Engineering

by
Devashree Pawar (Roll No. 21)
Naina Roy (Roll No. 24)
Khushi Upadhyay (Roll No. 30)

Mentor
Prof. T.P. Nagarhalli



University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science



(A.Y. 2023-24)

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled “Image Caption Generator” is a bonafide work of "Devashree Pawar (Roll No. 21), Naina Roy (Roll No. 24), Khushi Upadhyay (Roll No. 30)” submitted to the University of Mumbai in partial fulfillment of the requirement for the **Course project in semester VII of Fourth Year** Artificial Intelligence and Data Science engineering.

Supervisor

Dr. Tatwadarshi P. N.

Dr. Tatwadarshi P. N.
Head of Department

Abstract

An image caption generator is an artificial intelligence (AI) system that uses computer vision and natural language processing (NLP) techniques to generate captions or descriptions for images. The system typically analyses the visual content of an image and generates a corresponding textual description that captures the key objects, attributes, and relationships in the scene. The development of image caption generators has been driven by the growing demand for automated image understanding and annotation in various applications, such as image retrieval, multimedia indexing, and assistive technologies for the visually impaired. The underlying technology relies on deep learning models, such as convolutional neural networks (CNNs) for image analysis and recurrent neural networks (RNNs) for language generation. The image caption generation process typically involves several steps, including image preprocessing, feature extraction, sequence modeling, and decoding. The system first pre-processes the input image to normalize its size and color, and then extracts high-level features using a pre-trained CNN. These features are then fed into an RNN to generate a sequence of words that forms the caption. The RNN can be trained using various techniques, such as maximum likelihood estimation or reinforcement learning, to optimize the caption quality and coherence.

Table of Contents

Pg. No

Chapter No		Title	Page No.
1		Chapter # 1	1
	1.1	Problem Statement	3
2		Chapter # 2	4
	2.1	Block diagram, Description and Working	5
	2.2	Details of Hardware & Software	6
3		Chapter # 3	7
	3.1	Code	
	3.2	Result	
	3.3	Conclusion	
4		Chapter # 4	
		References	

Chapter # 1

1.1 Problem Statement:

The rapid growth of digital images and videos has resulted in a vast amount of visual data that needs to be analyzed and understood. However, analyzing and interpreting visual data is a complex and challenging task that requires human-level intelligence and expertise.

Therefore, there is a need for automated solutions that can extract meaningful information from visual data and facilitate their understanding and utilization.

Chapter # 2

2.1 BLOCK DIAGRAM

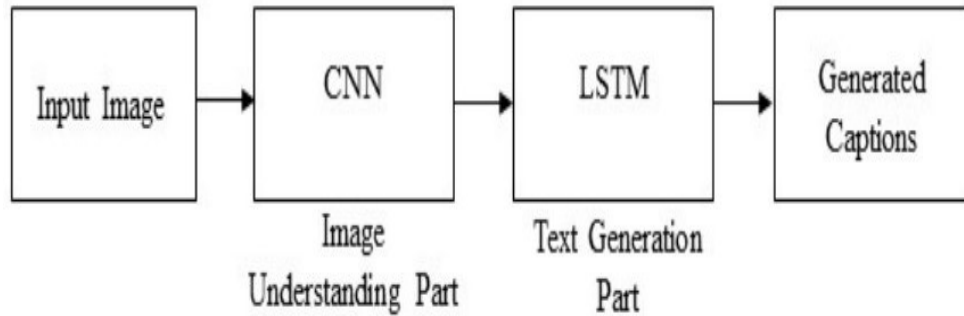


Fig 1: Block Diagram of basic image caption generation

The above figure shows a basic block diagram of our project image caption generator. The project typically consists of two parts: encoder and decoder

The encoder component uses a pre-trained CNN model to extract features from the input image. The CNN model is typically trained on a large dataset such as ImageNet and is able to extract high-level features from the image such as objects, shapes, and textures. The output of the CNN model is a fixed-length vector of feature values, which captures the visual content of the image.

The decoder component uses an LSTM model to generate the caption for the input image. The LSTM model takes in the feature vector generated by the encoder as its initial input, and then generates the caption word by word, based on the previous work in the sequence. The LSTM model learns the relationships between the words in the caption and generates a sequence of words that are coherent and relevant to the input image.

The CNN-LSTM model is a widely used approach for image caption generation as it can effectively learn the visual features of the image and can give more natural-sounding captions.

Module Description:

The process design for the image caption generator with GUI project can be divided into several stages:

1. **Planning and Requirements Gathering:** This stage involves understanding the project requirements, identifying the user needs, and planning the overall architecture of the system.
2. **Data Collection and Preprocessing:** This stage involves collecting and preprocessing the data needed to train the model. This can include finding and downloading image datasets, annotating the images with captions, and pre-processing the images to ensure they are of consistent quality. The dataset used by us is Flickr30k data. It is of size 9GB and has around 16,000 images and nearly 5 captions for each image. The preprocessing step also involves cleaning the caption data and preprocessing all the captions and tokenizing the caption dataset.
3. **Model Training:** This stage involves training the model using the pre-processed data. The model will typically consist of a combination of the Inception V3 model, an architecture for convolutional neural networks (CNNs) for image processing, and LSTM, a technique of recurrent neural networks (RNNs) for caption generation. The model can be trained using Keras, a deep-learning library that simplifies the process of building and training neural networks.
4. **Model Evaluation:** This stage involves evaluating the performance of the trained model using various metrics such as accuracy, perplexity, and BLEU score. This is important to ensure that the model is generating accurate captions.
5. **GUI Development:** This stage involves designing and developing the graphical user interface (GUI) that will allow users to interact with the image caption generator. The GUI should be intuitive, user-friendly, and responsive, allowing users to upload an image, and adjust input image settings, and caption settings. Our GUI is simple and is built by using the Tkinter library provided by Python. In the GUI, we can upload an image by browsing through our laptop and then click on generate caption to generate the caption. The caption generated is through two searching algorithms namely Beam search and greedy search which are a part of Artificial Intelligence.

6. System Integration: This stage involves integrating the various components of the system, such as the pre-processing module, feature extraction module, caption generation module, and GUI. This involves programming, testing, and optimizing the system to ensure that it works seamlessly.
7. Documentation: This stage involves creating user guides, technical documents, and other documentation to facilitate the use and development of the system. This documentation should be comprehensive and accessible to all stakeholders, including developers, users, and other interested parties.

2.2 DETAILS OF HARDWARE AND SOFTWARE

❖ **Hardware:**

- A personal computer
- Windows 10/11
- CPU 8 GB or more
- GPU 2 GB or more

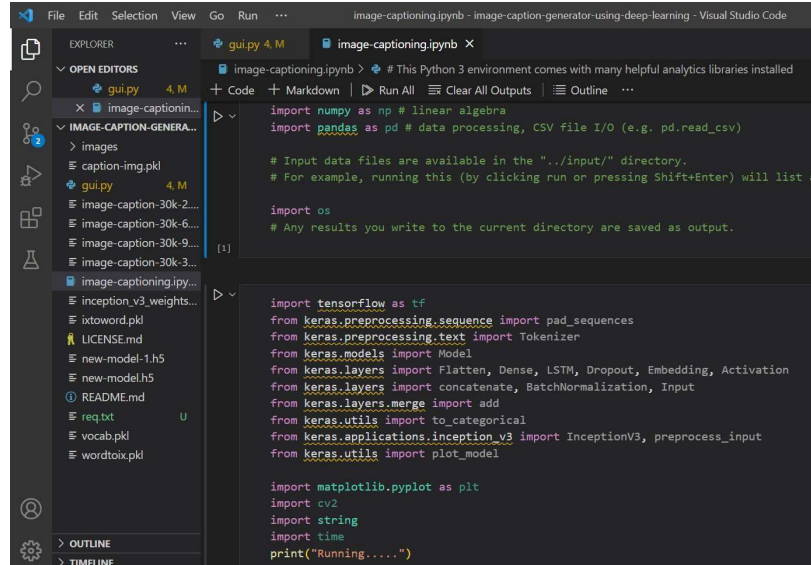


Software:

- VS code
- Jupyter Notebook
- Python

Chapter # 3

3.1 Code:



```
# This Python 3 environment comes with many helpful analytics libraries installed
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

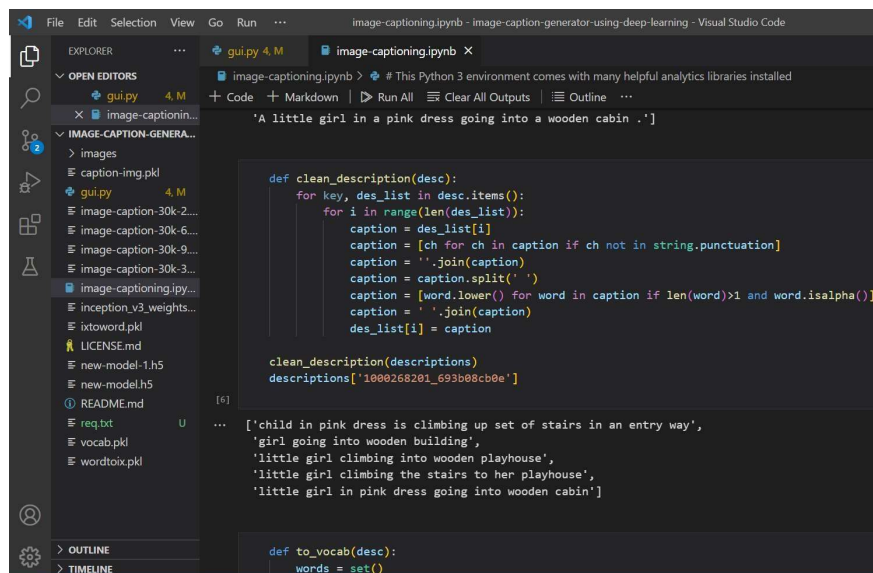
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list a
# Any results you write to the current directory are saved as output.

import os

import tensorflow as tf
from keras.preprocessing.sequence import pad_sequences
from 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.merge import add
from keras.utils import to_categorical
from keras.applications.inception_v3 import InceptionV3, preprocess_input
from keras.utils import plot_model

import matplotlib.pyplot as plt
import cv2
import string
import time
print("Running.....")
```

Fig 2. Importing packages



```
'A little girl in a pink dress going into a wooden cabin .']

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']

['child in pink dress is climbing up set of stairs in an entry way',
'girl going into wooden building',
'little girl climbing into wooden playhouse',
'little girl climbing the stairs to her playhouse',
'little girl in pink dress going into wooden cabin']

def to_vocab(desc):
    words = set()
```

Fig 3. Preprocessing data

The screenshot shows the Visual Studio Code interface with the file explorer on the left and the code editor in the center. The file explorer shows a project named 'image-caption-generator-using-deep-learning' with a subdirectory 'IMAGE-CAPTION-GENERATOR' containing files like 'caption-img.pkl', 'gui.py', 'image-caption-30k-2...', 'image-caption-30k-6...', 'image-caption-30k-9...', 'image-caption-30k-3...', 'inception_v3_weights...', 'ixtoword.pkl', 'LICENSE.md', 'new-model-1.h5', 'new-model.h5', 'README.md', 'req.txt', 'vocab.pkl', and 'wordtoix.pkl'. The code editor shows the 'image-captioning.ipynb' file with the following Python code:

```
# 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()
```

Below the code, the output of the model summary is displayed:

```
Model: "model_2"
```

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 34)	0	
input_2 (InputLayer)	(None, 2048)	0	
embedding_1 (Embedding)	(None, 34, 200)	338400	input_3[0][0]
dropout_1 (Dropout)	(None, 2048)	0	input_2[0][0]

Fig 4. Define model

The screenshot shows the Visual Studio Code interface with the file explorer on the left and the code editor in the center. The file explorer shows the same project as in Fig 4. The code editor shows the 'image-captioning.ipynb' file with the following Python code:

```
for i in range(30):
    model.fit([X1, X2], y, epochs = 1, batch_size = 256)
    if(i%2 == 0):
        model.save_weights("image-caption-weights" + str(i) + ".h5")
```

Below the code, the output of the training process is displayed:

```
Output exceeds the size limit. Open the full output data in a text editor
```

```
Epoch 1/1
292328/292328 [=====] - 55s 189us/step - loss: 3.8895
Epoch 1/1
292328/292328 [=====] - 55s 187us/step - loss: 3.1549
Epoch 1/1
292328/292328 [=====] - 54s 186us/step - loss: 2.9185
Epoch 1/1
292328/292328 [=====] - 54s 186us/step - loss: 2.7652
Epoch 1/1
292328/292328 [=====] - 54s 184us/step - loss: 2.6496
Epoch 1/1
292328/292328 [=====] - 54s 183us/step - loss: 2.5559
Epoch 1/1
292328/292328 [=====] - 54s 184us/step - loss: 2.4748
Epoch 1/1
292328/292328 [=====] - 53s 183us/step - loss: 2.4086
Epoch 1/1
292328/292328 [=====] - 54s 184us/step - loss: 2.3477
Epoch 1/1
292328/292328 [=====] - 53s 182us/step - loss: 2.2957
```

Fig 5. Training model

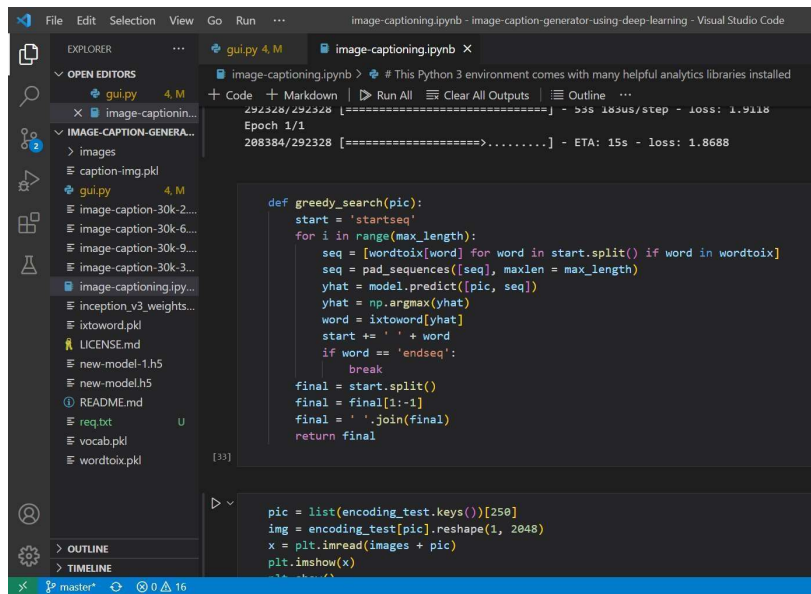


Fig 6. Training model with search algorithm

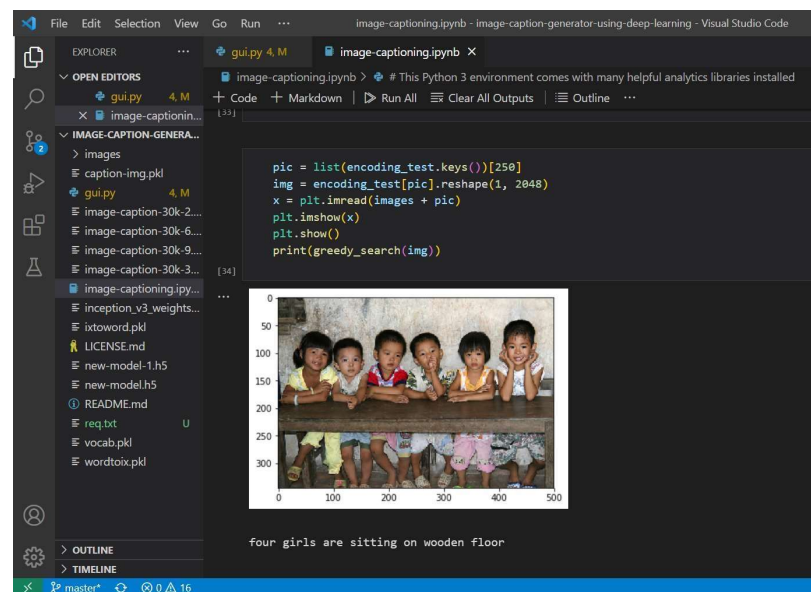
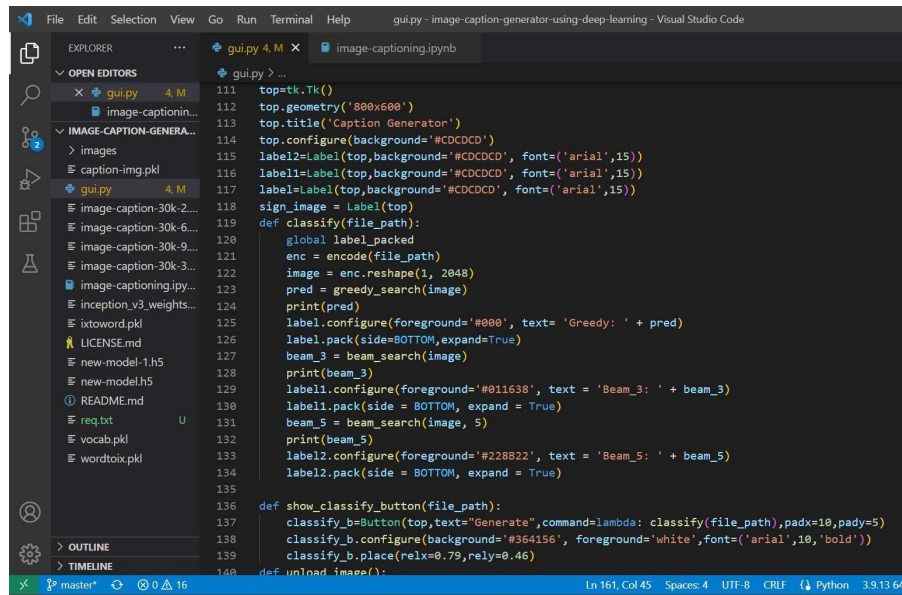


Fig 7. Testing the model



The screenshot displays the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'IMAGE-CAPTION-GENERA...' with files like 'images', 'caption-img.pkl', 'gui.py', and various model files. The code editor shows the 'gui.py' file with the following Python code:

```
111 top=tk.Tk()
112 top.geometry('800x600')
113 top.title('Caption Generator')
114 top.configure(background='#CDCDCD')
115 label2=Label(top,background='#CDCDCD', font=('arial',15))
116 label1=Label(top,background='#CDCDCD', font=('arial',15))
117 label1=Label(top,background='#CDCDCD', font=('arial',15))
118 sign_image = Label(top)
119 def classify(file_path):
120     global label_packed
121     enc = encode(file_path)
122     image = enc.reshape(1, 2048)
123     pred = greedy_search(image)
124     print(pred)
125     label.configure(background='#000', text= 'Greedy: ' + pred)
126     label.pack(side=BOTTOM,expand=True)
127     beam_3 = beam_search(image)
128     print(beam_3)
129     label1.configure(background='#011638', text = 'Beam_3: ' + beam_3)
130     label1.pack(side = BOTTOM, expand = True)
131     beam_5 = beam_search(image, 5)
132     print(beam_5)
133     label2.configure(background='#228B22', text = 'Beam_5: ' + beam_5)
134     label2.pack(side = BOTTOM, expand = True)
135
136 def show_classify_button(file_path):
137     classify_b=Button(top,text="Generate",command=lambda: classify(file_path),padx=10,pady=5)
138     classify_b.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
139     classify_b.place(relx=0.79,relx=0.46)
140 def upload_image():
```

The status bar at the bottom indicates the file is on the 'master' branch, at line 161, column 45, with 4 spaces, UTF-8 encoding, CRLF line endings, and Python 3.9.13.64.

Fig 8. GUI creation

3.2 Results:

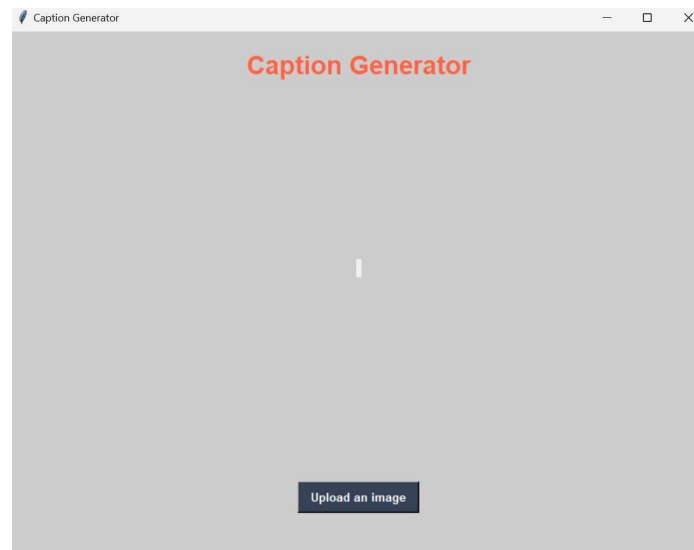


Fig 9. Caption generators first look

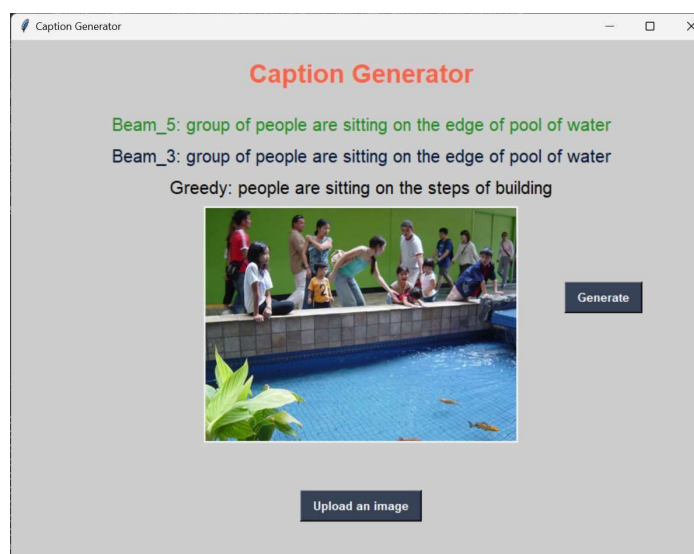


Fig 10. Output 1.

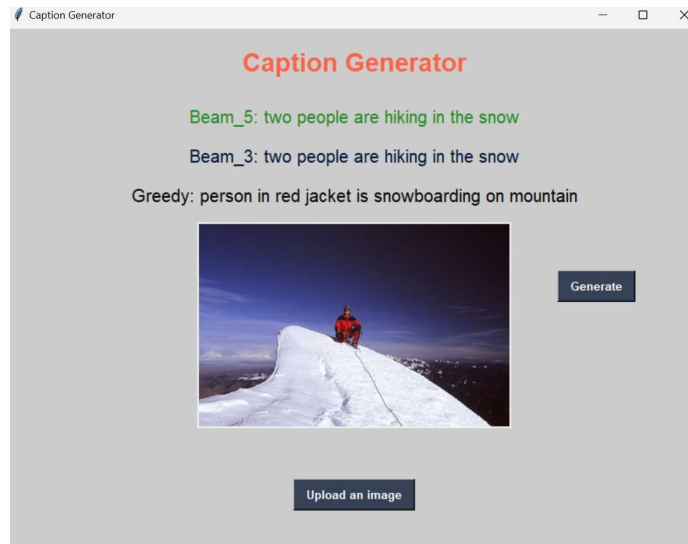


Fig 11. Output 2.

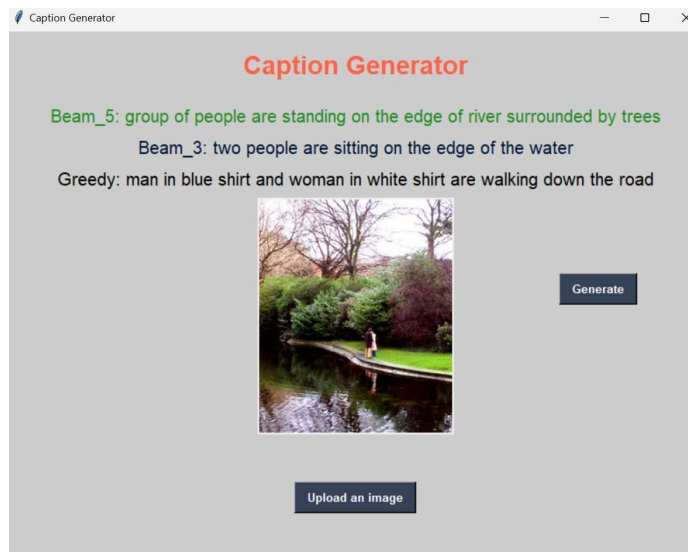


Fig 12. Output 3.

3.3 CONCLUSION

In conclusion, the image caption generator made with Keras, Inception V3 model, and LSTM is an advanced tool that uses deep learning techniques to generate accurate and natural-sounding captions for images. The GUI allows users to easily upload images and view the generated captions. In this project, a deep learning method for image caption generation using neural networks is presented, the proposed method was applied to a Flickr 30k dataset.

The encoder-decoder architecture is a strong method for creating picture captions, to sum up. The model can learn meaningful representations of images and produce natural language captions thanks to the usage of a convolutional neural network (CNN) as the encoder and an LSTM network as the decoder.

The model is effective in producing captions that are relevant and diverse, as evidenced by the evaluation of the generated captions using metrics like BLEU, METEOR, ROUGE, and CIDEr. Further enhancing the quality of the captions is the use of beam search and other methods for caption generation.

Chapter # 4

REFERENCE

- [1] Mr. Jadhav Chaitanya Chandrakant, Mr. Pandey Shiva Jitendra, Mr. Khade Nitin Narayan, Ms. Harshada Sonkamble, “IMAGE CAPTION GENERATOR Using Convolutional Neural Networks And Long Short Term Memory” , in *International journal of creative research thoughts(IJCRT)*, ISSN: 2320-2882, Volume 9, 2021.
- [2] Ch. Sneha, B. Premanvitha, B. Shanmukh , Kavitha Chaduv, “IMAGE CAPTION GENERATOR USING DEEP LEARNING”, in *The International Journal of Analytical and experimental modal analysis*, ISSN NO:0886-9367, Volume XIII, 2021.
- [3] Megha J Panicker, Vikas Upadhayay, Gunjan Sethi, Vrinda Mathur, “Image Caption Generator”, in *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, ISSN: 2278-3075 (Online), Volume-10 ,2021.
- [4] Seung-Ho Han, Ho-Jin Choi, “Domain-Specific Image Caption Generator with Semantic Ontology”, in *IEEE International Conference on Big Data and Smart Computing(BigComp)*, 2020.
- [5] N. Komal Kumar, D. Vigneswari , A. Mohan , K. Laxman , J. Yuvaraj, “Detection and Recognition of Objects in Image Caption Generator System: A Deep Learning Approach”, in *5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, 2019.
- [6] P. Hede , P. Moellic , J. Bourgeois , M. Joint , C. Thomas ,”Automatic generation of natural language descriptions for images”, in *Proceedings of the Recherche Dinformation Assistee Par Ordinateur*, 2004.
- [7] A. Farhadi , M. Hejrati , M.A. Sadeghi , P. Young , C. Rashtchian , J. Hocken- maier , D. Forsyth , “Every picture tells a story: Generating sentences from images”, in *Proceedings of the European Conference on Computer Vision*, pp. 15-29, 2010.

[8] O. Vinyals , A. Toshev , S. Bengio , D. Erhan , “Show and tell: a neural image caption generator”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164 ,2015.

[9] M. Hodosh, P. Young and J. Hockenmaier "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics", in *Journal of Artificial Intelligence Research*, Volume 47,2013.

[10] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6077–6086, 2018.

Free Online Plagiarism Checker

the above figure shows a basic block diagram of our project image caption generator the project typically consists of two parts encoder and decoder the encoder component uses a pre-trained cnn model to extract features from the input image the cnn model is typically trained on a large dataset such as imagenet and is able to extract high-level features from the image such as objects shapes and textures the output of the cnn model is a fixed-length vector of feature values which captures the visual content of the image the decoder component uses an lstm model to generate the caption for the input image the lstm model takes in the feature vector generated by the encoder as its initial input and then generates the caption word by word based on the previous work in the sequence the lstm model learns the relationships between the words in the caption

SIMILAR

17.6%

ORIGINAL

82.4%

MAKE IT UNIQUE

Text matches these sources

Sources:

Similarity:

1. [https://medium.com/@shibsankar/...](https://medium.com/@shibsankar/) 34.8%
[Include source](#) [View source](#)
2. <https://www.projectpro.io/article/lst...> 17.6%
3. <https://www.hackersrealm.net/post...> 13.8%
4. <https://github.com/jiteshgupta17/L...> 12.4%

33.1k
Shares



Let's Chat

Recheck this text after changes

Check another text