

Image Caption Generator

A Project Report

submitted in partial fulfillment of the requirements

of

Industrial Artificial Intelligence with Cloud Computing

by

Himanshu Kartela,

Roshan Jha,

Jaivik Rathod,

Rahil Mandalia

Under the Esteemed Guidance of

Jay Rathod

ACKNOWLEDGEMENT

The journey of building this image caption generator has been enriching, and I'm indebted to several individuals and resources for their invaluable contributions. First and foremost, I'd like to express my deepest gratitude to the vibrant deep learning community. The groundbreaking research and advancements in computer vision and natural language processing laid the groundwork for this project.

Furthermore, I extend my heartfelt appreciation to the creators of the image captioning datasets used in this project. These datasets were instrumental in training and refining the model's capabilities.

A special thanks goes to **Jay Rathod** sir for their insightful guidance and unwavering support throughout this project. Their expertise was invaluable in shaping the direction of this research. I am incredibly grateful to my fellow team members for their collaborative spirit and dedication. Their contributions were essential in bringing this project to fruition.

TABLE OF CONTENTS

Chapter 1. Introduction	1
1.1 Problem Statement	6
1.2 Problem Definition	6
1.3 Expected Outcomes	6
1.4 Organization of the Report	6
Chapter 2. Proposed Methodology	7
2.1 System Design	8
2.2 Modules Used	9
2.3 Data Flow Diagrams	10
2.4 Advantages	13
2.5 Requirement Specification	14
Chapter 3. Implementation and Results	15
3.1. Results of Image Detection	16
3.2. Results of Caption Generation	17
Chapter 4. Conclusion	20
Github Link	
Video Link	
References	23

LIST OF FIGURES

		Page No.
Figure 1	System Design	8
Figure 2	Data Flow diagram level 0	10
Figure 3	Data Flow diagram level 1 -(1)	10
Figure 4	Data Flow diagram level 1 -(2)	11
Figure 5	Result of Image Detection – (1)	15
Figure 6	Result of Image Detection – (2)	15
Figure 7	Result of Image Detection – (3)	16
Figure 8	Result of Caption Generation – (1)	17
Figure 9	Result of Caption Generation – (2)	18

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1. Problem Statement:

Training data limitations leads to bias and stereotyping in image caption generation, potentially producing inaccurate or culturally insensitive descriptions. Addressing these issues requires diverse and representative datasets and careful model training to mitigate biases.

1.2. Problem Definition:

Developing an image caption generator that accurately describes visual content while mitigating biases and stereotyping induced by training data limitations, ensuring inclusivity and cultural sensitivity in generated captions.

1.3. Expected Outcomes:

- Enhanced accuracy in describing visual content across diverse images.
- Reduction in biases and stereotyping reflected in generated captions.
- Improved cultural sensitivity and inclusivity in the language used for image descriptions.
- Increased user trust and satisfaction with the generated captions.
- Potential applications in various fields including accessibility, education, and content generation with minimized risk of inadvertently perpetuating harmful stereotypes.

1.4. Organization of the Report

The remaining report is organized as follows:

Chapter 2

Chapter 3

Chapter 4

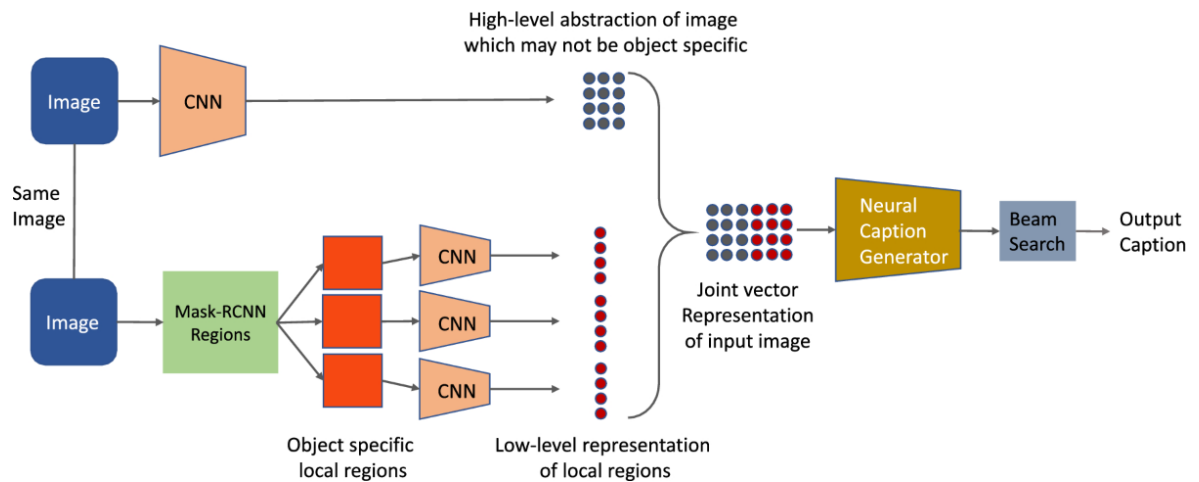
CHAPTER 2

PROPOSED METHODOLOGY

CHAPTER 2

PROPOSED METHODOLOGY

2.1 System Design



This system design ensures a robust, scalable, and efficient image caption generation service that meets user needs and maintains high performance.

2.2 Modules Used

2.2.1 Image Detection: -

The Image Detection module is responsible for analyzing input images and extracting relevant features that will be used by the caption generation module. It employs deep learning techniques to detect objects, scenes, and other visual features within the images.

- **Object Detection:** Utilizes pre-trained object detection models (e.g., Faster R-CNN, YOLO, SSD) to identify and localize objects within images.
- **Scene Recognition:** Determines the overall scene or context of the image (e.g., indoor, outdoor, nature, urban) using scene recognition models.
- **Feature Extraction:** Extracts high-level visual features from the detected objects and scenes, which serve as input to the caption generation module.
- **Preprocessing:** Preprocesses input images by resizing, normalization, and any other necessary transformations before feeding them into the detection models.

The Image Detection module acts as the initial processing step in the image caption generation pipeline, providing essential visual information that guides the caption generation process.

2.2.2 Caption Generation: -

The Caption Generation module is responsible for generating descriptive captions based on the features extracted from the input images. It employs deep learning techniques, specifically sequence-to-sequence models, to generate coherent and contextually relevant captions.

- **Sequence-to-Sequence Models:** Utilizes recurrent neural networks (RNNs) or transformer-based architectures to generate captions by mapping input visual features to sequences of words.
- **Attention Mechanisms:** Implements attention mechanisms to dynamically focus on different parts of the input image while generating each word of the caption, improving the relevance and quality of the generated captions.
- **Language Modeling:** Trains the caption generation model using large-scale datasets of images paired with human-generated captions, learning the statistical properties of natural language and image-caption relationships.
- **Fine-tuning:** Fine-tunes pre-trained caption generation models on domain-specific datasets to adapt them to specific application scenarios and improve performance.

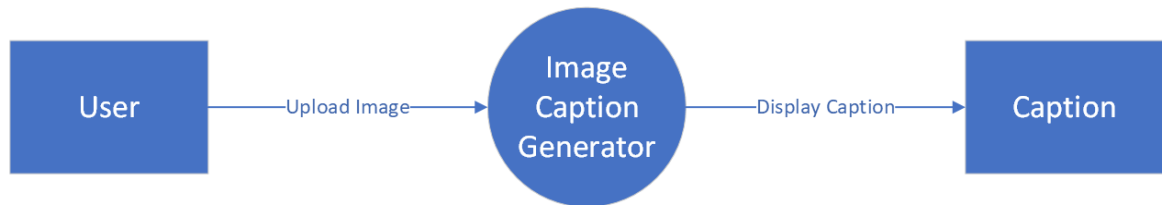
The Caption Generation module forms the core component of the image caption generation system, responsible for synthesizing textual descriptions that accurately represent the content and context of the input images. It utilizes advanced deep learning techniques to produce coherent and contextually relevant captions.

2.3 Data Flow Diagram

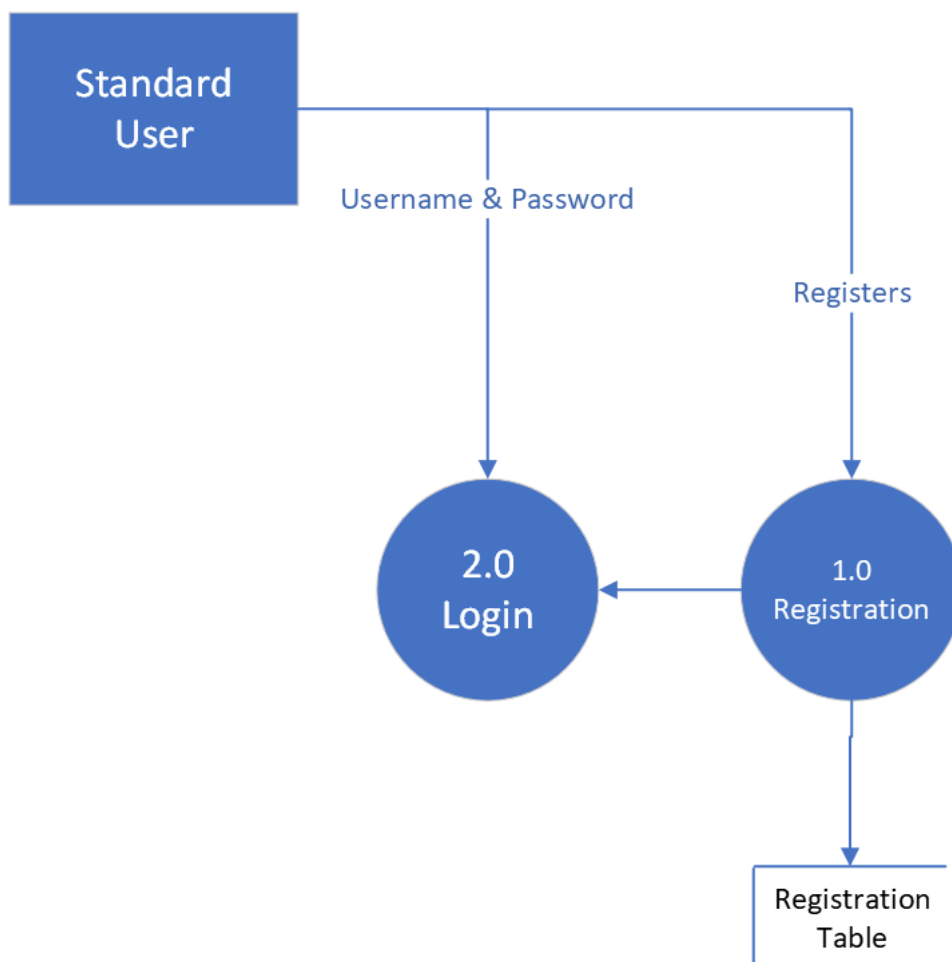
A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

Image Caption Generator

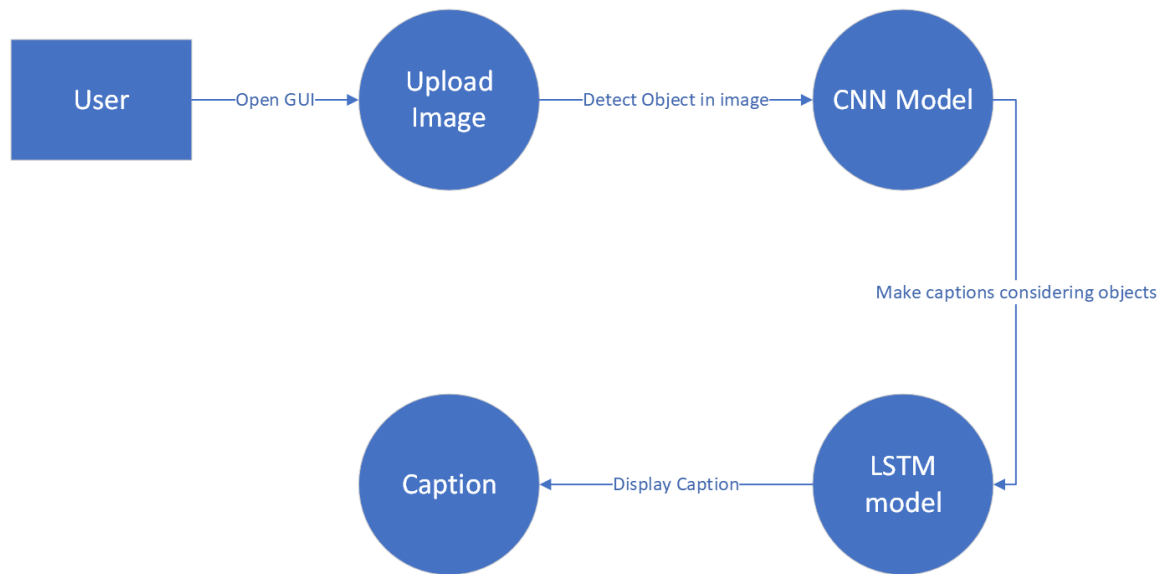
DFD Level 0 :-



DFD Level 1 – Image Caption Generator Registration Module:-



DFD Level 1 – Image Caption Generator image caption Module:-



2.4 Advantages: -

- **Improved Accessibility:** Enhances access to visual content for visually impaired individuals through detailed and accurate captions.
- **Content Moderation:** Aids in identifying and managing inappropriate or sensitive content automatically.
- **Enhanced User Experience:** Provides a more engaging and informative experience on social media, e-commerce, and other platforms.
- **Cultural Sensitivity:** Promotes inclusivity and cultural awareness by generating respectful and appropriate captions.
- **Bias Mitigation:** Helps in developing fairer AI systems by reducing biases present in training data.
- **Scalability:** Efficiently generates captions for large volumes of images, beneficial for businesses handling extensive visual data.
- **Educational Applications:** Assists in language learning and comprehension through visual aids and detailed descriptions.
- **Improved Searchability:** Enhances image search and retrieval by providing more relevant metadata for images.

2.5 Requirement Specification

2.5.1 Hardware Requirements:-

- **Development and Training:**
 - **CPU:** Multi-core processor (e.g., Intel Core i7 or AMD Ryzen 7)
 - **GPU:** NVIDIA GPU with CUDA support (e.g., NVIDIA GeForce RTX 3080)
 - **RAM:** Minimum 32 GB
 - **Storage:** SSD with at least 1 TB of storage for datasets and models
 - **Other:** High-speed internet connection for downloading datasets and libraries
- **Deployment:**
 - **Server:**
 - **CPU:** Multi-core server-grade processor (e.g., Intel Xeon)
 - **GPU:** High-performance GPU (e.g., NVIDIA Tesla or A100)
 - **RAM:** Minimum 64 GB
 - **Storage:** SSD with at least 2 TB of storage

- **Cloud Services:** Optional use of cloud-based services like AWS, Google Cloud, or Azure for scalable deployment

2.5.2 Software Requirements: -

- **Development Environment:**

- **Operating System:** Linux (Ubuntu 20.04 LTS or later), macOS, or Windows 10
- **Programming Languages:** Python 3.7 or later
- **Development Tools:** Jupyter Notebook, IDEs like PyCharm or VSCode
- **Libraries and Frameworks:**
 - **Deep Learning:** TensorFlow 2.x or PyTorch
 - **Data Processing:** NumPy, Pandas, OpenCV
 - **Natural Language Processing:** NLTK, SpaCy, Transformers
 - **Model Training and Evaluation:** Scikit-learn, Matplotlib, Seaborn
- **Version Control:** Git and GitHub or GitLab

- **Deployment Environment:**

- **Web Framework:** Flask, Django, or FastAPI for building APIs
- **Containerization:** Docker for containerizing the application
- **Orchestration:** Kubernetes for managing containerized applications (optional)
- **Cloud Services:** AWS, Google Cloud, or Azure for scalable deployment (if applicable)
- **Database:** MongoDB, PostgreSQL, or MySQL for storing image metadata and captions

- **Additional Tools:**

- **Monitoring:** Prometheus and Grafana for monitoring the application performance
- **Logging:** ELK stack (Elasticsearch, Logstash, Kibana) for logging and analyzing application logs

CHAPTER 3

Implementation and Result

CHAPTER 3

IMPLEMENTATION and RESULT

Results of Image Detection: -

Image Caption Generator - Flickr Dataset - CNN-LSTM.ipynb X

C > Users > dhyy > Downloads > Image Caption Generator - Flickr Dataset - CNN-LSTM.ipynb > Import Modules

+ Code + Markdown ...

Extract Image Features

```
# load vgg16 model
model = VGG16()
# restructure the model
model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
# summarize
print(model.summary())
```

[3]

...

Model: "functional_1"

...

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,880
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,880
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102,764,544
fc2 (Dense)	(None, 4096)	16,781,312

Load the Captions Data

```
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(tokens) < 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)
```

```
0% | 0/40456 [00:00<?, ?it/s]
```

```
len(mapping)
```

```
8091
```

Preprocess Text Data

```
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
```


Model Creation

```

#!pip install pydot

#!pip install graphviz

#!pip install pydot

#!pip install pydotplus

# encoder model
# image feature layers
inputs1 = Input(shape=(4096,), name="image")
fe1 = Dropout(0.4)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)
# sequence feature layers
inputs2 = Input(shape=(max_length,), name="text")
se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
se2 = Dropout(0.4)(se1)
se3 = LSTM(256)(se2)

# decoder model
decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)

model = Model(inputs=[inputs1, inputs2], outputs=outputs)
model.compile(loss='categorical_crossentropy', optimizer='adam')

# plot the model
plot_model(model, show_shapes=True)

```

You must install graphviz (see instructions at <https://graphviz.gitlab.io/download/>) for `plot_model` to work.

Train Test Split

```

image_ids = list(mapping.keys())
split = int(len(image_ids) * 0.90)
train = image_ids[:split]
test = image_ids[split:]

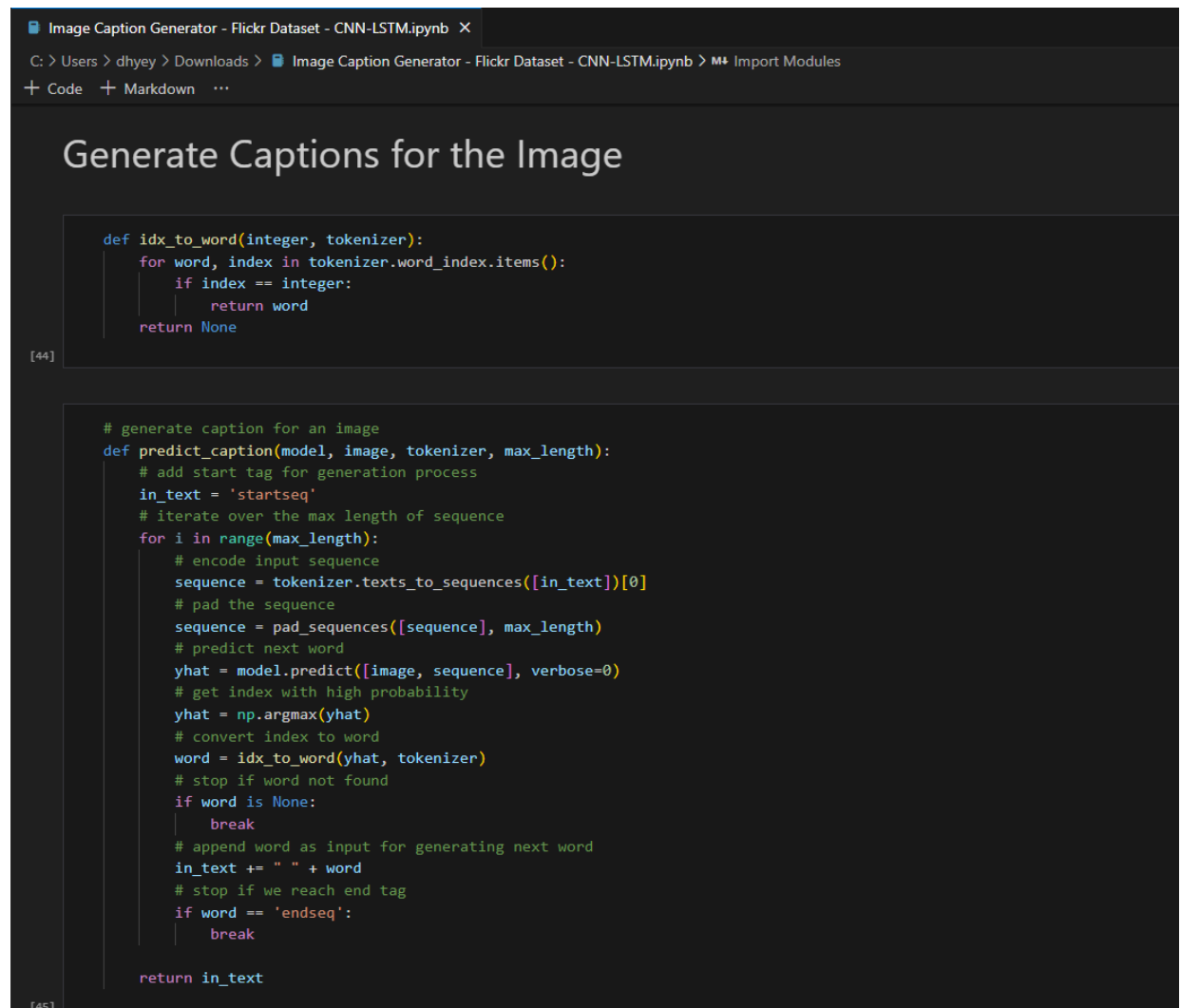
# startseq girl going into wooden building endseq
# x y
# startseq girl
# startseq girl going
# startseq girl going into
# .....
# startseq girl going into wooden building endseq

# 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 x1, 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 ("image": x1, "text": x2), y
                x1, x2, y = list(), list(), list()
                n = 0

```

Results of Caption Generation: -



```
def idx_to_word(integer, tokenizer):
    for word, index in tokenizer.word_index.items():
        if index == integer:
            return word
    return None

[44]
```

```
# generate caption for an image
def predict_caption(model, image, tokenizer, max_length):
    # add start tag for generation process
    in_text = 'startseq'
    # iterate over the max length of sequence
    for i in range(max_length):
        # encode input sequence
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        # pad the sequence
        sequence = pad_sequences([sequence], max_length)
        # predict next word
        yhat = model.predict([image, sequence], verbose=0)
        # get index with high probability
        yhat = np.argmax(yhat)
        # convert index to word
        word = idx_to_word(yhat, tokenizer)
        # stop if word not found
        if word is None:
            break
        # append word as input for generating next word
        in_text += " " + word
        # stop if we reach end tag
        if word == 'endseq':
            break
    return in_text

[45]
```

Image Caption Generator

```
from nltk.translate.bleu_score import corpus_bleu
# validate with test data
actual, predicted = list(), list()

for key in tqdm(test):
    # get actual caption
    captions = mapping[key]
    # predict the caption for image
    y_pred = predict_caption(model, features[key], tokenizer, max_length)
    # split into words
    actual_captions = [caption.split() for caption in captions]
    y_pred = y_pred.split()
    # append to the list
    actual.append(actual_captions)
    predicted.append(y_pred)

# calculate 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)))
```

0%| | 0/810 [00:00<?, ?it/s]

BLEU-1: 0.522882
BLEU-2: 0.299050

Image Caption Generator - Flickr Dataset - CNN-LSTM.ipynb X
C: > Users > dhruv > Downloads > Image Caption Generator - Flickr Dataset - CNN-LSTM.ipynb > Import Modules
+ Code + Markdown ...

Visualize the Results

```
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)
```

[47]

```
generate_caption("1001773457_577c3a7d70.jpg")
```

[48]

```
...
-----Actual-----
startseq black dog and spotted dog are fighting endseq
startseq black dog and tri-colored dog playing with each other on the road endseq
startseq black dog and white dog with brown spots are staring at each other in the street endseq
startseq two dogs of different breeds looking at each other on the road endseq
startseq two dogs on pavement moving toward each other endseq
-----Predicted-----
startseq two dogs are playing on the floor endseq
...
```



CHAPTER 4

CONCLUSION

CHAPTER 4

CONCLUSION

ADVANTAGES:

- **Improved Accessibility:** Generates accurate and descriptive captions, enhancing accessibility for visually impaired users by providing them with better understanding of visual content.
- **Content Moderation:** Assists in automated content moderation by identifying and describing inappropriate or sensitive content.
- **Enhanced User Experience:** Offers a more engaging and informative user experience in applications like social media, online shopping, and digital storytelling.
- **Cultural Sensitivity:** Promotes cultural sensitivity and inclusivity in automated content generation, reducing the risk of offending users.
- **Bias Mitigation:** Contributes to the development of fairer AI systems by addressing and reducing biases present in training data.
- **Scalability:** Enables efficient and scalable captioning for large volumes of images, beneficial for businesses and platforms that handle extensive visual data.
- **Educational Use:** Provides valuable tools for educational purposes, aiding in language learning and comprehension through visual aids and descriptions.
- **Enhanced Searchability:** Improves image search and retrieval by providing more detailed and relevant metadata for images.

SCOPE:

- **Data Collection and Preprocessing:**

- Gather a diverse and representative dataset of images with corresponding captions.
- Preprocess data to ensure quality, consistency, and remove any potential biases.

- **Model Development:**

- Develop and train a neural network model capable of generating descriptive and contextually accurate captions for images.
- Implement techniques to mitigate biases and ensure inclusivity in generated captions.

- **Evaluation and Testing:**

- Evaluate the model's performance using standard metrics such as BLEU, METEOR, and CIDEr.
- Conduct bias and fairness assessments to identify and address any stereotyping or cultural insensitivity in the outputs.
- Perform user testing to gather feedback on the accuracy and cultural appropriateness of the captions.

- **Optimization and Refinement:**

- Optimize the model for efficiency and scalability to handle large volumes of images.
- Refine the model based on feedback and evaluation results to improve performance and reduce biases further.

- **Integration and Deployment:**

- Integrate the image caption generator into target applications such as social media platforms, e-commerce websites, and accessibility tools.
- Ensure seamless deployment and provide ongoing monitoring to maintain the quality and fairness of generated captions.

- **Documentation and Reporting:**

- Document the development process, methodologies, and findings.
- Produce comprehensive reports on model performance, bias mitigation strategies, and user feedback.

- **Future Enhancements:**

- Plan for future enhancements such as incorporating additional languages, expanding the dataset, and continually improving bias mitigation techniques.
- Explore potential extensions of the project into related areas such as video captioning or real-time caption generation.

Text to speech: -

The captions generated by the Image Caption Generator can be used as Text To Speech for the blind people who can not see the text. So this will provide the opportunity for them to explore the world easily.

We can add Translators also for translating into various languages which enrich the accessibility of the project.

REFERENCES

- "Very Deep Convolutional Networks for Large-Scale Image Recognition" by Karen Simonyan and Andrew Zisserman. [Link](#)
- "Show and Tell: A Neural Image Caption Generator" by Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. [Link](#)
- "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering" by Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. [Link](#)
- "Attention Is All You Need" by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. [Link](#)
- "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. [Link](#)

Links

Video link: -

https://drive.google.com/drive/folders/1hDuB_9SNobSUVbnrxz12TKo88EGwUGkW?usp=sharing

GitHub link: -

<https://github.com/Roshan0412/Image-Caption-Generator>

APPENDIX

In the appendix, we present supplementary details crucial to understanding and contextualizing the image caption generation project. It begins with sample code snippets offering insight into the practical implementation of key functionalities, such as image preprocessing and caption generation. These snippets serve as reference points for developers and enthusiasts, aiding comprehension of the project's technical intricacies. Alongside, a visual depiction of the model architecture elucidates the intricate components and data flow within the system, enhancing comprehension of its underlying mechanisms.

Furthermore, comprehensive information regarding the dataset utilized for training and evaluation is provided, encompassing its name, scale, and covered categories. This inclusion offers transparency regarding the foundational data driving the project's development and evaluation. Moreover, insight into the evaluation metrics employed, such as BLEU, METEOR, and CIDEr scores, sheds light on the rigorous standards used to assess the quality and efficacy of generated captions.

The appendix also delineates the hardware specifications and software stack utilized throughout the project's lifecycle, providing crucial context regarding the computational infrastructure and technological environment. Lastly, expressions of acknowledgment are extended to the authors of referenced papers, developers of open-source libraries, and contributors to datasets leveraged in the project, recognizing their invaluable contributions to the advancement of image captioning technology.