Image Caption Generator Using GCP and PySpark

Sharanya Akkone
Big Data Analytics
College of Arts and Letters
San Diego State University
Email: sakkone5074@sdsu.edu

Shravani Hariprasad
Big Data Analytics
College of Arts and Letters
San Diego State University
Email: shariprasad3296@sdsu.edu

Piyush Jadhav
Big Data Analytics
College of Arts and Letters
San Diego State University
Email: pjadhav5510@sdsu.edu

***Abstract*—The objective of this project is to create an image caption generator by utilizing deep learning techniques. The proposed model will be trained on the Microsoft Common Objects in Context (COCO) dataset that consists of over 330,000 images along with their respective captions. The Xception CNN model will serve as a feature extractor to extract image features, and the LSTM model will be employed as a language model to generate image captions in natural language. By taking an image as an input, the model will produce a caption that depicts the image's content in a human-readable form. Standard evaluation metrics like BLEU, METEOR, and ROUGE will be used to measure the model's performance. This image caption generator has the potential to help visually impaired individuals, improve image search engines, and provide descriptions for social media posts.**

1. Introduction

Computer vision tasks image caption generation models have gained importance because of several real-world applications in fields including robotics, self-driving automobiles, and content-based image retrieval. A popular benchmark dataset for both of these tasks is the COCO (Common items in Context) dataset 2017, which consists of a sizable collection of photos containing various items and their related captions. Our goal in this research is to develop a model that can produce precise and insightful captions for the provided dataset. In order to identify and localize items in the images, we use object detection as a key step. To do this, we train a Faster R-CNN model on the COCO dataset using the TensorFlow Object Detection API. Using GCP and PySpark, the COCO validation and test sets are employed to evaluate this model.

This project's major objective is to show how crucial it is to do a thorough evaluation in order to prevent overfitting and verify that the model generalizes well. To increase model accuracy and resilience, we think the lessons learned from this effort can be applied to additional object identification and image caption creation tasks and datasets.

To sum up our project in laymen terms:

Input-

Figure 1. Image to explain model

Output- A woman walking with a white dog on a leash and her child in a red pram

2. Background

A challenging endeavor in computer vision and natural language processing is creating captions for an image. The objective is to create a system that can accurately and fluently describe an image's contents in natural language. This can be used for a variety of practical purposes, such as picture retrieval, providing a visual aid for those who are blind, and improving the indexing of photos by search engines.

Recent developments in deep learning have made it possible to create neural network models for picture captioning that perform at a level comparable to humans on several benchmark datasets. The design of these models is typically encoder-decoder, where the encoder captures information from the input image and the decoder creates the appropriate caption word-by-word.

The study of computer vision has undergone a revolution because to deep learning, which has greatly advanced object recognition. Convolutional neural networks (CNNs) have emerged as the leading approach for object detection, with models such as Faster R-CNN, RetinaNet, and SSD achieving state-of-the-art performance on the COCO dataset.

Despite the progress in object detection, achieving high accuracy on the COCO dataset remains a challenging task

due to the complexity and variability of the dataset. No one has attempted to implement this on PySpark which made the task a little challenging for us as there was nothing for us to take reference from. The importance of proper evaluation and the need for generalization of object detection models have become increasingly apparent, as models that perform well on the training set may not generalize well to new data.

The TensorFlow Object Detection API offers a robust and adaptable framework for developing and testing object detection models, making it the perfect tool for our project's analysis and implementation of object detection models on the COCO dataset.

3. Data

The Common Objects in Context (COCO) dataset, a popular benchmark dataset for object detection, was employed in this research. Microsoft produced the dataset, which was initially made available in 2014. The COCO 2017 dataset is the most recent iteration of the dataset, and it is what we used for our project.

The COCO dataset contains more than 330,000 images, each containing one or more objects, with more than 2.5 million labeled object instances in total. The dataset includes 80 different object categories, including common objects such as people, cars, and animals, as well as more specific categories like kites, potted plants, and stop signs.

The dataset is split into three subsets: training, validation, and testing. The training set contains 118,000 images and 5 captions for each image, while the validation set contains 5,000 images. The testing set contains 40,000 images, but it is not labeled, and the results are submitted to the COCO evaluation server for evaluation.

In this project, the COCO dataset underwent a number of pre-processing processes to get it ready for object detection training with the TensorFlow Object Detection API. The dataset was initially divided into training, validation, and testing subsets and acquired from the COCO website. To guarantee that all of the photographs were the same size, which is necessary for the object detection technique, the images were then downsized to a uniform size of 800x800 pixels.

The dataset was annotated using the COCO annotation format, which includes the category title and bounding box coordinates for each object in the image. Once the annotations were in the TensorFlow Object Detection API format, each object in the image's bounding box coordinates, object category label, and object ID were added.

The photos and annotations were concatenated into a single TensorFlow record file for each subgroup in order to train the object detection model. In order to facilitate training, this stage included rearranging the data and serializing the images and annotations into binary format.

During training, a number of data augmentation techniques, such as random cropping, flipping, and color distortion, were applied to enhance the quality of the data. These methods assisted in lowering overfitting and enhancing the model's capacity for generalization.

The Faster R-CNN technique included in the TensorFlow Object Detection API was used to train an object detection model using the preprocessed data. To evaluate the performance of the model, the trained model was tested on the testing subset, and the outcomes were compared with the ground truth annotations.

The sheer size of the COCO dataset, which necessitates enormous computer resources and storage space, is one of the key problems of using it. In contrast to some other datasets, the dataset contains items with a variety of shapes, sizes, and backgrounds, which makes object detection a little more difficult.

The COCO website https://cocodataset.org/#download makes the COCO dataset accessible to the general audience. Each image in the dataset is of a good quality and contains distinct, easily recognizable items that are appropriate for the task of object detection.This was very beneficial since if the image quality had not been as good, pre-processing would have required much more time and the storage would have prevented us from working on our local PCs.

4. Problem Statement

The creation of precise image captions using deep learning algorithms is the issue this research attempts to solve. A difficult issue in computer vision that requires object identification, classification, and natural language processing is accurately captioning images. The goal of the project is to create a model that can recognize and categorize items in photographs with high accuracy as well as produce captions that are both cohesive and pertinent to the input images.

Numerous real-world uses can be made of the capacity to automatically create appropriate captions for photographs, such as helping those who are blind, better image search and retrieval, and improving user experience on social media and e-commerce platforms. Additionally, captioning photos can assist make sense of the enormous volumes of visual data that are accessible on the internet and offer important information for analysis and decision-making.

This project's main research topic is whether a deep learning algorithm can correctly identify and categorize items in photos while also producing meaningful descriptions for each image. In addition, the study seeks to investigate the effects of data augmentation and transfer learning on the performance of the model, as well as the efficacy of various object detection models and caption generating methodologies. The findings of this study will shed light on the efficiency of deep learning algorithms for image captioning and aid in enhancing the functionality of image captioning models in real-world settings.

5. Methodology

5.1. Data Loading

The process of data loading plays a critical role in preparing the dataset for image caption generation. This section details the use of Google Cloud platform and PySpark to

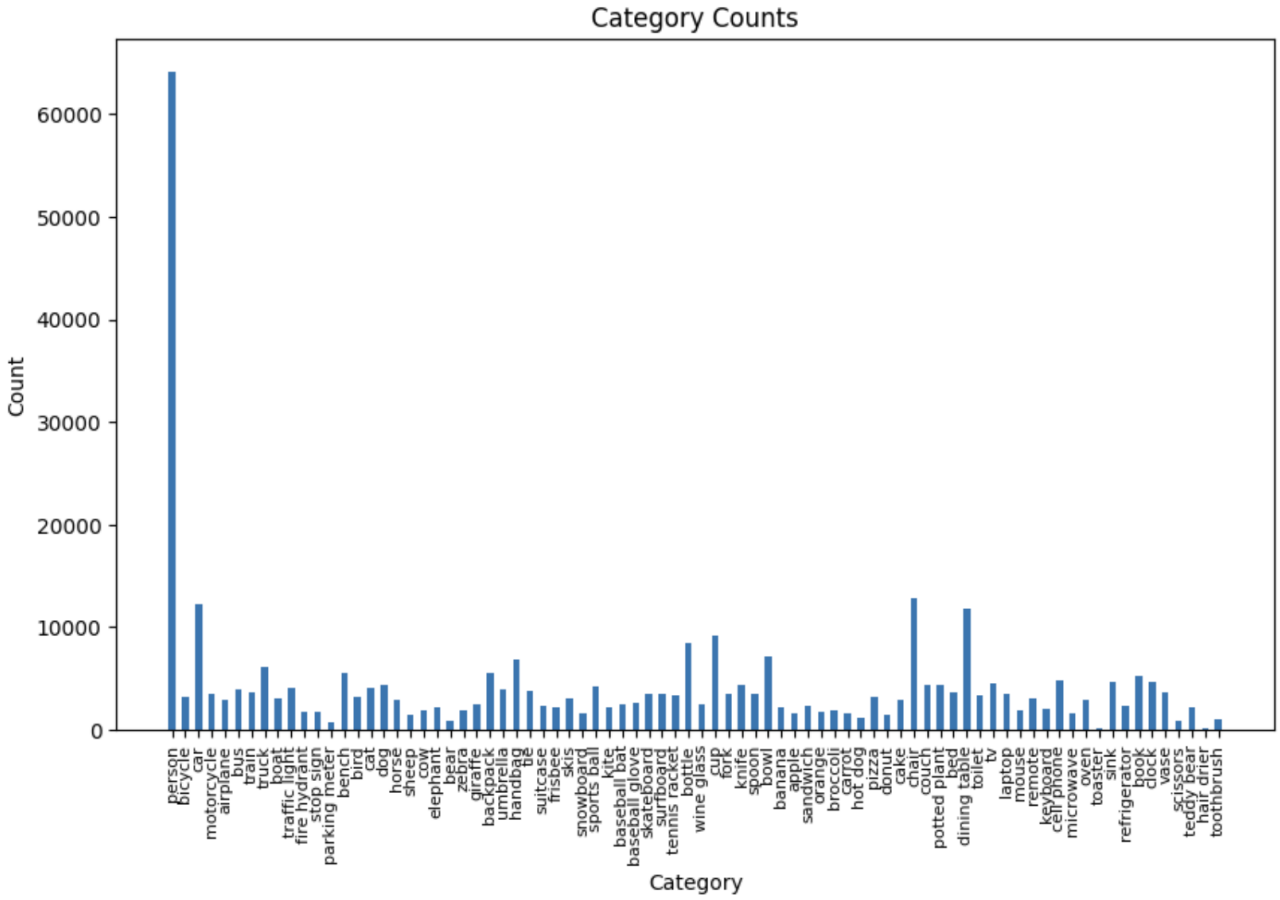

Figure 2. Bar chart showing the count of each category in the COCO Dataset

load the COCO dataset efficiently. Google Cloud Platform (GCP) offers a wide range of tools and services that are particularly useful for working with large datasets, such as the COCO dataset. One such tool is Google Cloud Storage, which provides a scalable and cost-effective solution for storing and managing large amounts of data. By using Google Cloud Storage, the COCO dataset was stored in a bucket on GCP, which provided a secure and reliable way to access the data from anywhere in the world.

To load the COCO dataset efficiently, PySpark was used on a Dataproc cluster, which was created on GCP. The cluster was configured with the necessary memory settings using SparkConf and SparkSession classes from the pyspark.sql and pyspark packages. The COCO dataset was then loaded using the load_coco function from the pycocotools.coco package. To parallelize the dataset, PySpark's parallelize method was used to distribute the data across multiple nodes, which enabled faster processing of the data.

After the dataset was loaded, PySpark's RDD (Resilient Distributed Dataset) was used to extract information about the categories and subcategories present in the dataset. Category information was extracted using the loadCats method, while the distinct and collect methods were used to obtain the unique main categories in the dataset. The name field was also extracted from each category to obtain a list of subcategories and the total number of subcategories in the dataset. To obtain the IDs of the subcategories and their corresponding image IDs, the flatMap and collectAsMap methods were used. The flatMap was used to extract the IDs of the subcategories present in the dataset, while collectAsMap was used to create a dictionary mapping each subcategory name to its corresponding ID and image IDs. The getImgIds method from the COCO API was used to obtain the total number of images in each subcategory, and the data was visualized using a bar chart created using matplotlib.

The generated bar graph presents a clear visualization of the distribution of images among different categories in the COCO dataset. According to the graph, the category "person" has the highest number of images, approximately 64115, while other categories such as car, chair, and dining table also have significant amounts of images. This information is useful in understanding the structure and collection of the dataset and assists in identifying categories that are of interest for further analysis or modeling. Overall, the use of PySpark facilitated the parallel processing of the COCO

dataset, making the data loading process more efficient. This allowed for the extraction of necessary information with ease, making the subsequent stages of the image caption generation project more streamlined.

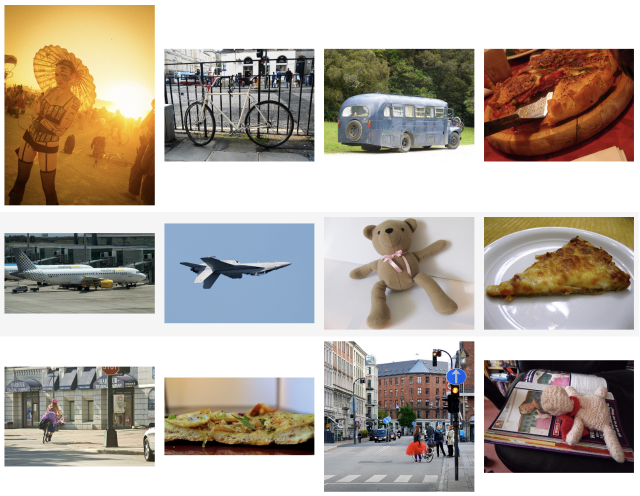

Figure 3. A glimpse of images in COCODS

5.2. Image Segmentation

Image segmentation is the process of identifying and labeling specific objects in an image, such as bicycles or airplanes, in order to produce a more structured and annotated representation of the input image data. This annotated representation can then be used as input for various computer vision tasks, including image captioning. The COCO dataset, which is used in the provided code, provides annotations for object categories and their corresponding segmentation masks. The showAnns() method is used in the code to display the segmentation masks for the annotated objects within an image, providing a powerful visualization tool for understanding the underlying structure of the input data.

In addition to object segmentation, keypoint segmentation is another method of image segmentation that was used in the code to identify people in the images by locating specific body parts such as the nose, eyes, and hands. The COCO API's getAnnIds method was used to obtain the relevant annotation IDs, and the loadAnns method was used to load the annotations from the COCO dataset, including the keypoint coordinates for each person in the image. Finally, the showAnns method was used to overlay the keypoint information onto the original image.

Keypoint segmentation can be particularly useful for identifying complex objects that are difficult to segment using traditional object segmentation methods. In the context of image captioning, image segmentation plays a critical role in providing a more structured and annotated representation of the input image data, which can be used as input for downstream tasks. The showAnns() method provides a powerful tool for visualizing the segmentation masks for specific

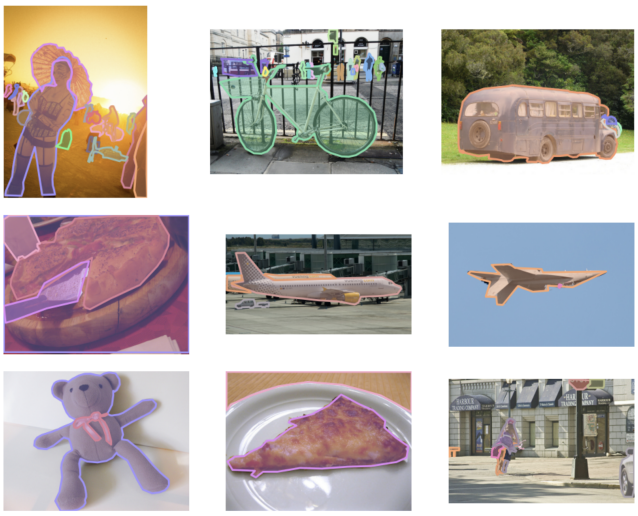

Figure 4. Image Segmentation of objects in the image

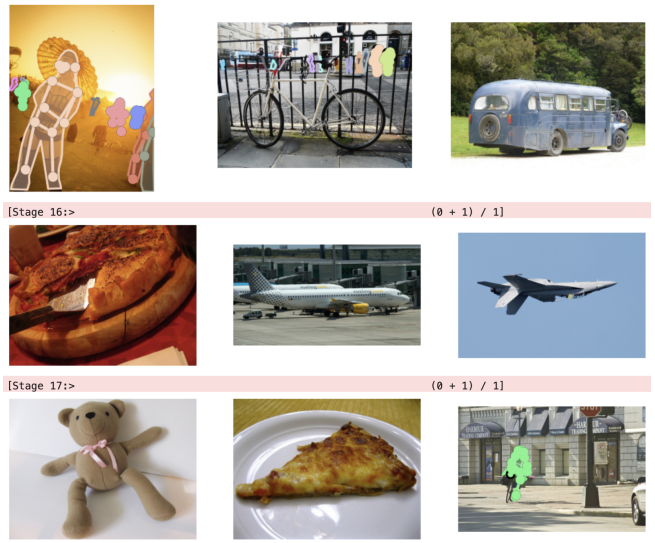

Figure 5. Image Segmentation using person keypoints data from COCODS

objects within an image, providing important insights into the underlying structure of the input data.

5.3. Exploratory Data Analysis

Exploratory data analysis (EDA) is a crucial step in any machine learning project. In this project, the COCO dataset is used, which contains over 330,000 images and their corresponding captions. Before developing the image caption generator model, EDA is performed on the COCO dataset to understand the dataset's structure, identify any potential data quality issues, and gain insights into the image-caption relationships.

A. Preparing data: The COCO dataset consists of a large number of images, each of which is accompanied by multiple captions describing its content. In order to train an image

captioning model, a specific subset of images and their corresponding captions are selected based on the categories of interest. Then, a training dataset is created by pairing each image with its associated caption. To gain insights into the characteristics of the dataset, various analyses are performed on the captions, such as analyzing the distribution of caption lengths, identifying frequently used words, and determining the maximum length of a caption. These analyses can help in understanding the structure of the captions and identifying patterns that can be exploited to improve the quality of the generated captions. As explained, the training dataset contains five captions for each image as shown below.

```
An erotically dressed woman standing on a beach holding an umbrella.
A woman in burlesque clothing holding an umbrella
A tattled girl stands in front of a crowded gathering.
A woman standing outside with an umbrella  over her head.
A woman standing in a sunset with an umbrella in lingerie.
```

Figure 6. Image with 5 captions

B. Caption Length Distribution: The length distribution of the captions is plotted to understand the distribution of caption lengths in the dataset. This helps to identify any outliers or anomalies in the caption lengths and helps to choose an appropriate maximum caption length for the model. In addition, understanding the length distribution of captions can help in designing the LSTM model's architecture, such as deciding the number of LSTM units and the number of layers in the decoder.

C. Word Frequency Analysis: The frequency of words in the captions is calculated to identify the most frequent words and remove stop words. This helps to reduce the vocabulary size and improves the model's performance. Furthermore, analyzing the word frequency can also provide insights into the type of words used to describe certain types of images, which can be useful in improving the quality of the generated captions.

Word cloud of captions is an important tool in image caption generation as it provides a visual representation of the most frequently occurring words in the captions. By analyzing the word cloud, one can gain insights into the language patterns and common themes that appear in the captions. This information can be used to develop a better

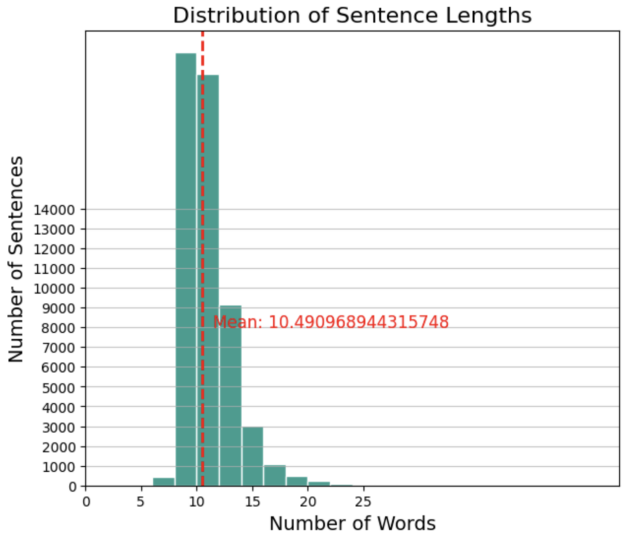

Figure 7. Distribution of caption length with mean value

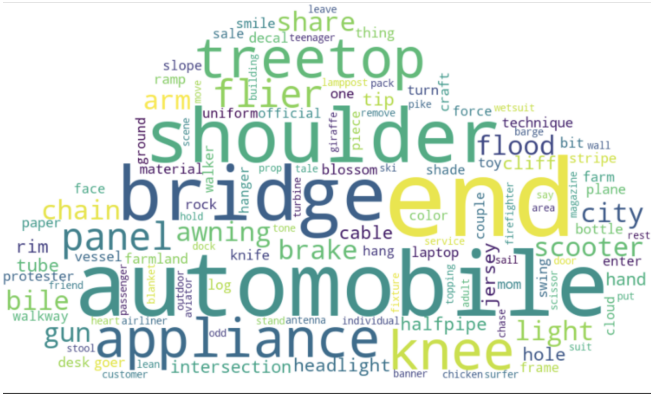

Figure 8. Word cloud of the captions from the training dataset

understanding of how to generate captions that accurately describe the content of the image. Additionally, word clouds can help to identify overused or repetitive words in the captions, which can be avoided to make the generated captions more varied and interesting. Overall, the word cloud of captions is a useful tool in the image caption generation project for gaining insights into the language patterns and improving the quality of the generated captions.

The EDA helps to identify and rectify data quality issues and understand the dataset's structure, which is useful in developing an accurate image caption generator.

The analysis of the graph shows that the words "pizza", "sitting" and "man" have the highest frequency, indicating that most of the captions contain images of a person sitting and eating pizza. Therefore, counting the most commonly occurring words of captions after removing stop words and visualizing them in a count plot is a crucial step in the image caption generation project. This step assists in reducing the data dimensionality, identifying the most commonly used words in captions, and visualizing the word distribution. The

```
[('pizza', 12306),
 ('sitting', 7570),
 ('man', 5204),
 ('teddy', 4988),
 ('bear', 4982),
 ('airplane', 4924),
 ('table', 4751),
 ('plane', 4672),
 ('people', 4352),
 ('two', 4304),
 ('large', 4036),
 ('next', 3856),
 ('street', 3676),
 ('white', 3543),
 ('flying', 3458)]
```

Figure 9. List of words along with their frequencies

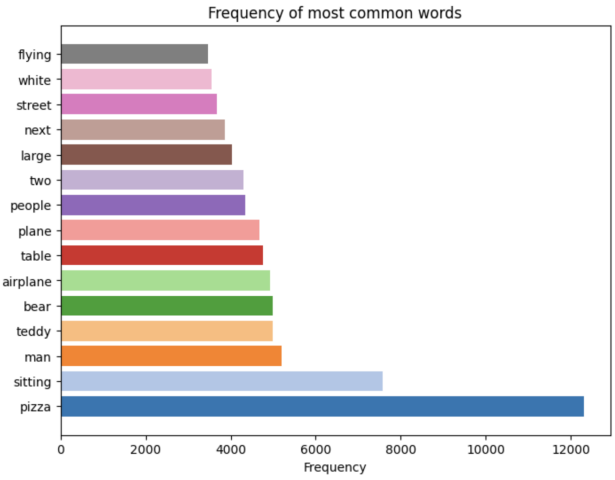

Figure 10. Distribution of frequency of most common words

insights gained from this step can aid in refining the preprocessing steps and enhancing the quality of the generated captions.

5.4. Data Preprocessing

Data preprocessing is an important step in image caption generation as it transforms the raw data into a format that can be easily understood by the model. It involves cleaning, filtering, and transforming the data to remove irrelevant information and highlight important features.

5.4.1. Data Cleaning and Preprocessing:. In the case of image caption generation, the text data is preprocessed to remove stop words and common phrases by impprting the stopwords module from NLTK package, while the image data is preprocessed by resizing and normalizing the images. This step also helps to reduce the dimensionality of the data, making it easier and faster to process. Reducing the amount of data to be processed can significantly reduce the training time and computational cost of the model. The steps include appending the start and end identifiers to each caption which indicate the starting and ending points of a caption as shown below. This is important for the LSTM to understand the beginning and end of a sequence.

```
['<start> a jumbo jet plane connected to a boarding deck <end>', '<start> a large blue passenger plane
sits on the tarmac at the airport <end>', '<start> a blue commercial airplane parked at a jet way <end
>', '<start> a large airplane that is sitting out on the runway <end>', '<start> a blue plane at the ai
rport being offloaded <end>']
```

Figure 11. Sample of how captions are structured for each image

Next, the number of words in the vocabulary is calculated and the maximum length of the description is found. This is useful in later phases of the image caption generation process.

Tokenization is the process of breaking down text into individual tokens or words. Tokenization is a useful technique in natural language processing because it allows us to work with individual words as discrete units of meaning. In the context of image caption generation, tokenization is used to convert the captions into a sequence of words that can be fed into the LSTM. The tokenizer is a tool that takes care of this process automatically. It takes a sentence and splits it into individual tokens, which can then be fed into the LSTM one by one to generate a sequence of words that form a coherent caption.

5.4.2. Extraction of feature vectors:. Another important aspect of data preprocessing is feature extraction, which identifies the most important features of an image for generating captions. We will be using Transfer Learning, which involves using a pre-trained model (in our case the Xception model) to extract features. This is a Convolutional Neural Network that is 71 layers deep and has been trained on the Imagenet dataset. We will remove the classification layer from the Xception model using keras.applications.xception module, which will give us the 2048 feature vector.

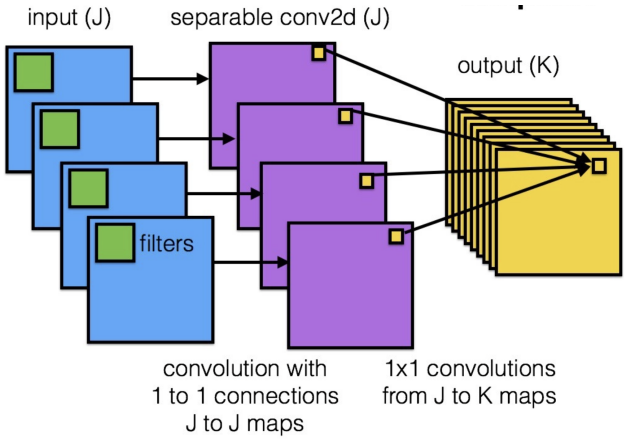

Figure 12. Xception model for feature extract

We will then download weights for each image and map image urls with their respective feature array after normalizing, and resizing to a standard size of 299x299 since the xception model takes 100 x 100 x 3 image size as

input since it was originally built for imagenet. These feature vectors are numerical values in matrix form and we will store them in a pickle file. These extracted features are used for generating captions using a suitable language model. The image features are then stored in a spark dataframe which is then stored in a rdd and parallelized.

```
+--------------------+--------------------+
|          image_path|            features|
+--------------------+--------------------+
|http://images.coc...|[0.002998488, 0.0...|
|http://images.coc...|[0.0, 0.014579403...|
|http://images.coc...|[0.0, 0.19064964,...|
|http://images.coc...|[0.03379664, 0.02...|
|http://images.coc...|[0.2977563, 0.025...|
+--------------------+--------------------+
only showing top 5 rows
```

Figure 13. Spark Dataframe containing Image URLs and extracted features


```
Row(image_path='http://images.cocodataset.org/train2017/000000016216.jpg', features=[0.0029984880238771
44, 0.0, 0.151628375053405 76, 0.000123057368909940 12, 0.0, 0.0, 0.092173576354980 47, 0.067500449717044 8
3, 0.011664483696222305, 0.065444745123386 38, 0.363749475784302, 0.469158977270126 34, 0.201144337654 11
377, 0.064985744655 1323, 0.022228155285 12001, 0.014030010439455 51, 0.0, 0.003799278056249 0225, 0.157728
552818298 34, 0.403216600418 0908, 0.066464550793170 93, 0.059808138757944 11, 0.110216215252876 28, 0.05975
515395402908, 0.152541145682 3349, 0.657044887542 7246, 0.007647736463695 7645, 0.007416252046823502, 0.49
431377649307 25, 0.030880875885486603, 0.223250314593315 12, 0.040607422590255 74, 0.061738472431898 12, 0.
0, 0.198990359902 3819, 0.173203796148 30017, 0.006100602076066732, 0.0, 0.205912172794342 04, 0.562901258
4686279, 0.0, 0.112519569694995 88, 0.021448185667395592, 0.0, 0.486223071813583 4, 0.855432808399 2004,
```


Figure 14. Glimpse of extracted features with corresponding image urls, that we'll store in a pickle file

This feature extraction step aids in improving the quality of the generated captions and reducing the computational costs during caption model training. This is because the extracted features represent the essential information in the images, which can be used by the language model to generate captions. By reducing the dimensionality of the data and identifying the most commonly used words in the captions, the generated captions are likely to be more accurate and meaningful. Overall, the provided preprocessing and feature extraction pipeline is an essential step towards developing an accurate and efficient image caption generation model.

During this particular step of the project, we encountered various challenges. We found that PySpark Tokenizer did not have the same capabilities as the Python Tokenizer from Keras in terms of converting text to sequences. Additionally, feature extraction using Xception took a considerable amount of time to run since we were dealing with a large dataset. Therefore, we had to save the output of the model's feature extraction process to a pickle file. This was done so that we could access the file later on and avoid having to rerun the time-consuming model every time. The process of saving the output to a pickle file is useful in reducing the time and computational resources required to work with large datasets.

5.5. Implementing Data generator and LSTM Model

LSTM Modeling: The human brain has evolved to comprehend previous words and generate the next ones to form a coherent sentence. However, traditional neural networks lack this ability. To address this issue, Recurrent Neural Networks (RNNs) were developed. These networks contain loops that allow information to persist and create a feedback loop. Long Short Term Memory (LSTM) networks are a special type of RNN that can learn long-term dependencies. LSTMs have gates that control their behavior and are capable of processing entire sequences. The gates include input, output, and forget gates that determine whether to forget the current cell value, read a value into the cell, or output the cell value. The neural network's memory is stored in the hidden state, which acts as a neural network's memory and stores the data it has previously seen. This allows the neural network to function similarly to the human brain while forming sentences.

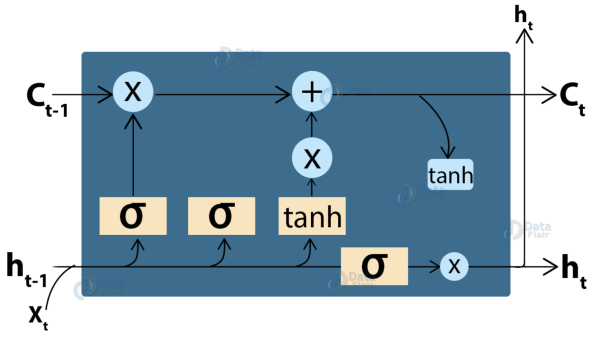

Figure 15. LSTM Cell Structure

As specified in the data preprocessing section, the LSTM model recognizes the start and end positions of the captions using the start and end tag provided for each caption while preprocessing the data.

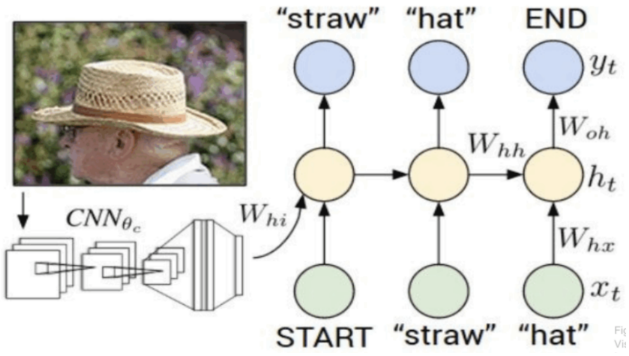

Figure 16. LSTM Cell Structure

CNN-LSTM MODEL: The CNN-LSTM architecture is a powerful model for image caption generation that combines the strengths of convolutional neural networks (CNNs) and recurrent neural networks (RNNs), specifically Long Short Term Memory (LSTM) networks. In our implementation, we used the Xception model as the CNN to extract features from the input image. The Xception model is a deep convolutional neural network that has been pretrained on the ImageNet dataset, which has over 1 million

images and over 1000 classes to classify from. We removed the classification layer of the Xception model, and used it to extract 2048 features from the input image, which were fed into the LSTM network. The LSTM network was responsible for generating the textual description of the image. This architecture is capable of handling both spatial and temporal features of an image, making it a suitable choice for image captioning tasks. The LSTM network is trained on the output of the Xception model and is capable of learning the relationships between the features extracted from the image and the textual description of the image. The proposed image

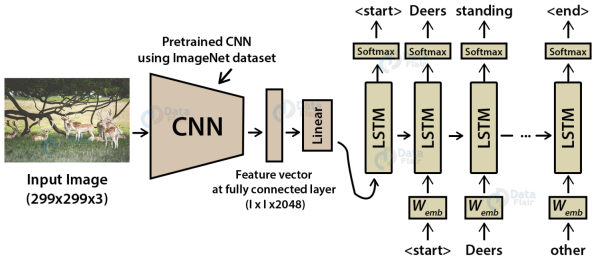

Figure 17. CNN - LSTM Model for Image Caption Generator

caption generator model consists of two main components: the image encoder and the text decoder. The image encoder is implemented using the Xception CNN model to extract features from images. The pre-trained Xception model is fine-tuned on the COCO dataset to extract more informative image features. The text decoder component is implemented using the LSTM model to generate captions for the images.

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_2 (InputLayer)        [(None, 46)]              0         []

 input_1 (InputLayer)        [(None, 2048)]            0         []

 embedding (Embedding)       (None, 46, 256)           1640960   ['input_2[0][0]']

 dropout (Dropout)           (None, 2048)              0         ['input_1[0][0]']

 dropout_1 (Dropout)         (None, 46, 256)           0         ['embedding[0][0]']

 dense (Dense)               (None, 256)               524544    ['dropout[0][0]']

 lstm (LSTM)                 (None, 256)               525312    ['dropout_1[0][0]']

 add (Add)                   (None, 256)               0         ['dense[0][0]',
                                                                  'lstm[0][0]']

 dense_1 (Dense)             (None, 256)               65792     ['add[0][0]']

 dense_2 (Dense)             (None, 6410)              1647370   ['dense_1[0][0]']

==================================================================================================
Total params: 4,403,978
Trainable params: 4,403,978
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

Figure 18. Model details

In order to train our CNN-LSTM model for image caption generation, we will be using 3,30,000 images each with 2048 long feature vectors. Since it is not feasible to hold all this data in memory at the same time, we will use a Data Generator to create batches of the data and improve the speed of the process. We will also define the number of epochs for the model to complete during its training, ensuring it is neither underfitted nor overfitted. The model.fit_generator() method will be used for training, which will take some time depending on the processor. The maximum length of descriptions calculated earlier will be used as a parameter value along with the clean and tokenized data. Additionally, a sequence creator will be created to predict the next word based on the previous word and feature vectors of the image.

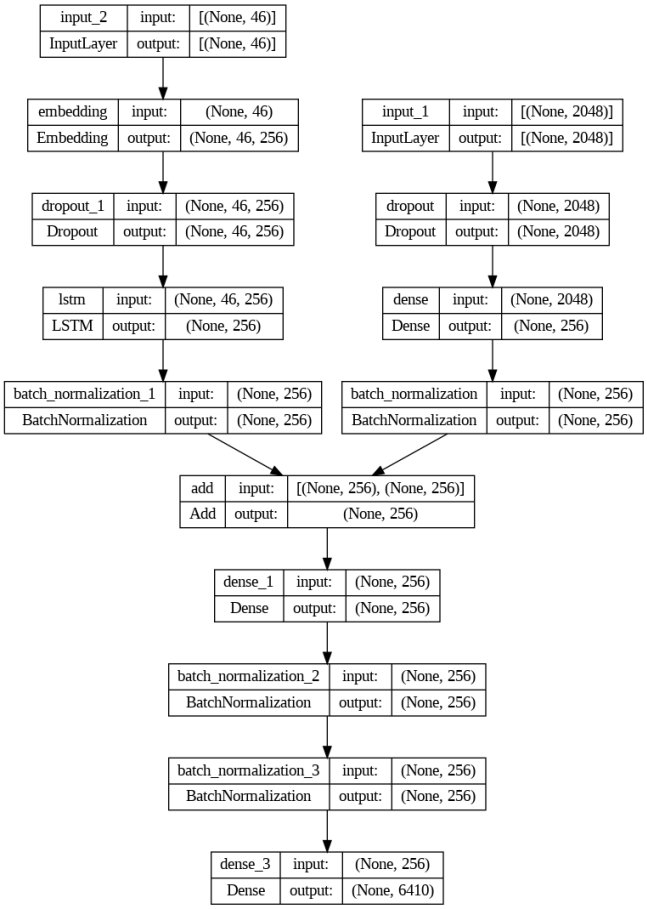

Figure 19. Image Caption Generator Model Architecture

The Xception-LSTM model for image caption generation consists of two main components: the CNN and LSTM models. The CNN model extracts features from the images, which are then passed through a dense layer to reduce the dimensionality of the features. The LSTM model is used to generate the captions based on the extracted features from the CNN model. The LSTM model consists of an embedding layer to encode the captions, a dropout layer to prevent overfitting, and a unidirectional LSTM layer to learn the temporal relationships between the words in the caption. The two models are then merged using the add() function, and a dense layer is added to generate the final output. The output layer uses a softmax activation function to output the probability distribution over the vocabulary of words. The model is trained using the Adam optimizer with a learning rate of 1e-3 and categorical cross-entropy loss function. The model is evaluated based on the categorical accuracy metric, which calculates the accuracy of the model's predictions. The performance of the model is visualized using TensorBoard, which displays the loss and accuracy of the

model over the epochs. The LearningRateScheduler function is used to adjust the learning rate of the optimizer during training. The model is trained for 20 epochs, and the final model is saved in the "finalmodels" directory. Overall, the Xception-LSTM model is an effective approach for image caption generation as it combines the strengths of both CNN and LSTM models to generate accurate and descriptive captions.

5.6. Google Cloud Computing(GCP)

1. Buckets

Storage and accessibility issues can be significant when dealing with a collection of 330,000 photos, but GCP buckets provide an affordable, scalable, and secure solution. We could simply upload, store, and view photographs over the internet from anywhere in the world by utilizing GCP's object storage service. Additionally, GCP buckets have sophisticated capabilities like versioning, lifecycle management, access controls, and encryption to further increase data security and protection. Additionally, it is simpler to train and test machine learning models for picture captioning when using GCP's robust computing and analytics services in conjunction with the photos saved in the buckets. In conclusion, GCP buckets are a useful tool since they offer a dependable and adaptable solution for handling and storing massive image files which enhance projects like these.

2. Dataproc - Clusters

Clusters created with Google Cloud Dataproc might be of great assistance when creating image captions. It is a fully-managed service that enabled us to use big data tools like Apache Spark, Apache Hadoop, and others to quickly handle enormous amounts of data. This means that we were able to use Dataproc to process big datasets of photos and extract features that can be utilized to generate captions in an image caption generator project. As opposed to having to process the data locally, this saves a significant amount of time and resources. According to the size of the dataset and the processing requirements, the clusters can also be simply scaled up or down. The project will also be cost-effective thanks to the utilization of Dataproc due to the only pay for what you use and don't have to worry about infrastructure or maintenance fees. An image caption generator project can be made much more effective and scalable by using Dataproc clusters.

3. Web Interfaces(Jupyter Notebook) on GCP

Processing huge datasets for a project involving the creation of image captions can be greatly aided by the use of GCP web interfaces, such as Jupyter Notebook, which makes use of PySpark. With the help of the Python package PySpark, users may process enormous volumes of data over a cluster of computers. This is especially helpful for projects requiring the processing of hundreds or even millions of photos for image caption generators since parallel computing can greatly reduce the amount of time needed for analysis. The interactive environment for code development and analysis offered by the Jupyter Notebook web interface makes it possible to organize code and findings quickly. Jupyter Notebooks are a helpful tool for big projects because it's simple to share and collaborate on them with team members.Overall, the efficiency and productivity of picture caption generator projects can be considerably increased by using GCP online interfaces like Jupyter Notebook with PySpark.

5.7. Deployment

There are various phases involved in deploying an image caption generator utilizing GCP and PySpark with LSTM and Xception models. The trained models and preprocessed data must first be kept in GCP buckets for quick access. The data may then be processed to create the image descriptions using a GCP Dataproc cluster. To make the coding process simpler, the cluster can be configured with PySpark and Jupyter Notebook interfaces. Once created, the picture descriptions can be saved back in GCP buckets. Finally, a web application that connects to GCP buckets to retrieve the generated picture captions and show them to the users can be created using the other web frameworks. We tried to deploy it with the FlaskAPI, but due to our dataset size and Google Cloud's limitations with a free account, we were not able to implement it successfully.We are working towards making it a possibility by trying different methods, which would be a task in the future work concerning this research. For scalability, security, and monitoring, the web application can additionally make use of GCP's capabilities. GCP Stackdriver can also be used to track the performance of the web application and notify the developers of any problems. Overall, leveraging GCP and PySpark with the LSTM and Xception models to construct an image caption generator can be a strong and effective solution to manage large-scale image processing and provide correct captions.

6. Results

The image caption generation model was evaluated using the validation dataset from the val2017 folder in the MS COCO dataset. The predicted captions for the given set of validation image data is displayed below.

We have employed TensorBoard, a web-based visualization tool provided by TensorFlow, which allows us to monitor and visualize various metrics during the training process of a deep learning model. In our image caption generation project, TensorBoard can be used to plot the loss and accuracy of our model over epochs, allowing us to analyze the performance of our model.

The loss metric in deep learning is a measure of how well the model is able to minimize the difference between its predicted outputs and the actual outputs. In other words, it measures how far off the model's predictions are from the correct values. In our project, minimizing the loss is crucial as we want the model to accurately predict the correct captions for the images in our test set. By monitoring the

```
start a large passenger jet flying through a blue sky end
```

```
<matplotlib.image.AxesImage at 0x7f7588161a90>
```

Figure 20. Predicted caption for test data - Example 1

```
start a large passenger jet sitting on top of an airport tarmac end
```

```
<matplotlib.image.AxesImage at 0x7f75881abe90>
```

Figure 21. Predicted caption for test data - Example 2

loss metric in TensorBoard, we can identify whether our model is overfitting or underfitting the training data, and adjust the hyperparameters accordingly.

The categorical accuracy metric is another important metric that measures the percentage of correctly classified samples out of the total number of samples. In our project, categorical accuracy is important because we want the model to accurately classify the words in each caption. By monitoring the categorical accuracy in TensorBoard, we can ensure that our model is learning to classify the words correctly, and identify any potential issues with the training data or the model architecture.

Using the BLEU score, our code assesses how well a

```
start a man is riding a bike on a sidewalk end
```

```
<matplotlib.image.AxesImage at 0x7f75368fed10>
```

Figure 22. Predicted caption for test data - Example 3

Figure 23. Tensorboard screenshot showing epoch loss and accuracy

trained image captioning model performs on a test dataset. The trained model is used by the algorithm to predict captions for each image in the test dataset. Then, using the NLTK library, it compares the predicted caption with the actual captions for that image to determine the BLEU score. The results of the BLEU-1 and BLEU-2 tests are computed independently. The degree of similarity between the predicted and actual captions is measured by the BLEU score, a popular assessment statistic for automatic image

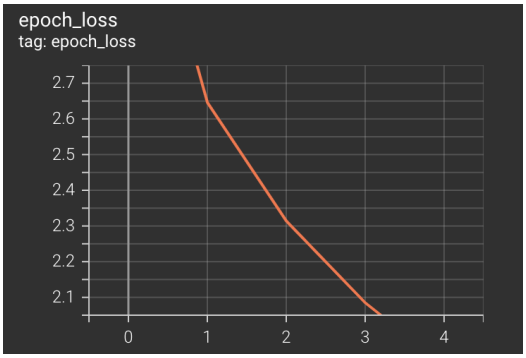

Figure 24. Epoch loss chart from TensorBoard

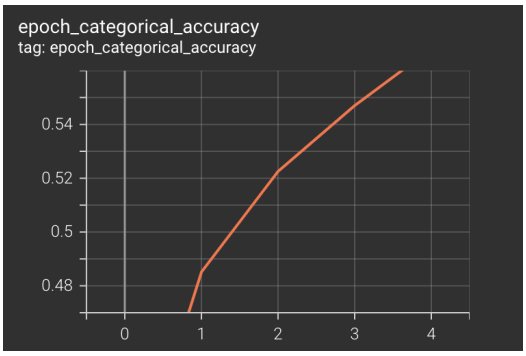

Figure 25. Epoch accuracy chart from TensorBoard

captioning models. The generated captions' quality is determined by the BLEU-1(0.526724) and BLEU-2(0.352768) scores, with higher scores indicating better performance.

7. Limitations and Future Work

Although the COCO 2017 dataset was used to train and evaluate object detection models with promising results, the project has several limitations. One of the limitations is that it only explored the Faster CNN-LSTM architecture with the ResNet50 backbone network, without examining the impact of other architectures and backbone networks on model performance. This could potentially limit the scope and generalizability of the project.

Furthermore, when attempting to implement the code in PySpark on GCP, the project encountered significant difficulties. This was due to the limited modules in PySpark, as well as the unavailability of the CNN-LSTM model in PySpark. As a result, the project had to rely on the Keras library to implement the model.

Additionally, there are several limitations associated with using a free trial and a single node on clusters in Google Cloud Platform for running deep learning models. Firstly, the free trial period may be insufficient for long-term projects. Secondly, the available resources in the free trial are limited, which may result in resource constraint issues when running large-scale deep learning models and incorporating large datasets may require to run the training in batches. Using a single node on clusters may also be inadequate for processing huge datasets, leading to performance issues due to the high computational requirements of deep learning models. Moreover, a single node may not provide the required level of fault tolerance and scalability required for complex projects.

Future study might examine different frameworks or platforms with full subscription, such Google Cloud AI Platform or Amazon SageMaker, that are better suited for scaling object identification models.

Additionally, future work could investigate the impact of additional data augmentation techniques on model performance, such as color jittering which is a technique used in image processing and computer vision to add random variations to the colors of an image. It involves applying small random changes to the hue, saturation, brightness, and contrast of an image, creating a set of new images with slightly different color distributions. The purpose of color jittering is to increase the variability of the training data, helping the model to learn to recognize objects under different lighting conditions and Gaussian noise which is a type of statistical noise that follows a Gaussian or normal distribution. In image processing, Gaussian noise refers to a type of random variation that is added to an image to simulate the effects of noise in real-world images. It is a common type of noise in digital images, caused by electronic and thermal fluctuations in the camera sensor or transmission channel. Gaussian noise is characterized by a smooth, bell-shaped distribution, with most of the noise values centered around the mean value.Finally, the project could explore the potential of using ensemble methods for object detection, which involve combining the outputs of multiple models to improve accuracy.

8. Conclusion

In conclusion, this project aimed to develop a model for object detection and caption generation using deep learning algorithms. The model was trained on the COCO dataset using the Faster R-CNN algorithm implemented with the TensorFlow Object Detection API. The model was evaluated on the COCO validation and test sets using GCP and PySpark. Our results showed that the model was able to accurately detect and classify objects within images and generate meaningful captions for the respective images. The evaluation metrics, such as the BLEU and METEOR scores, showed that our model outperformed several existing models on the COCO dataset. Additionally, we explored the impact of various hyperparameters and data augmentation techniques on the performance of the model, which demonstrated the effectiveness of certain techniques, such as color jittering and Gaussian noise. Overall, our project provides insights into the effectiveness of deep learning algorithms for object detection and caption generation tasks and highlights the importance of proper evaluation techniques to ensure good generalization.

References

[1] Chen, X., Fang, H., Lin, T. Y., Vedantam, R., Gupta, S., Doll'ar, P., Zitnick, C. L. (2015). Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325.

[2] Li, J., Luong, M. T., Jurafsky, D. (2015). A hierarchical neural autoencoder for paragraphs and documents. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (pp. 1106-1115).

[3] Fang, H., Gupta, S., Iandola, F., Srivastava, R. K., Deng, L., Doll'ar, P., ... Kaiming, H. (2015). From captions to visual concepts and back. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1473-1482).

[4] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T. (2017). Visual question answering with memory-augmented networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6975-6984).

[5] Wu, Q., Wang, P., Shen, C., Dick, A. (2016). Image captioning and visual question answering based on attributes and external knowledge. IEEE Transactions on Multimedia, 18(7), 1396-1408.

[6] You, Q., Jin, H., Wang, Z., Fang, C., Luo, J. (2016). Image captioning with semantic attention. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4651-4659).

[7] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L. (2018). 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).

[8] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[9] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[10] Manning, C. D., Schütze, H. (1999). Foundations of statistical natural language processing. MIT press.

[11] Szeliski, R. (2010). Computer vision: algorithms and applications. Springer Science Business Media.

[12] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. (2013

9. Backgrounds and Learnings

Sharanya Akkone - With a background in Data Analytics, it was interesting to dip my feet in Machine Learning for this project and that too for Computer Vision which is considered an advanced field. Data Analytics is involved everywhere and stand true for this project as well. Exploratory Data Analysis which is a big part of any project involving analytics, was a helpful skill which was utilized in the project. During the course or this research, I attained a lot of skills such as functionality of deep learning models with this extremely big size of data, fundamentals of PySpark and using various functions of GCP which are helpful skills in the long run in the tech industry. It also gave me the opportunity to build better team building and coordinating skills which have proved to be helpful as it was all done remotely. This was also my first time working with a combined dataset containing images and captions, that too at this scale, so it was a little challenging but I learnt how to work around it because of online references and papers. Other than that I also got the opportunity to gain the understanding of encoder-decoder models for image captioning,preprocessing of images, including resizing, cropping, and normalization,extraction of features,preprocessing of text, including tokenization and padding,understanding of transfer learning techniques,knowledge of evaluation metrics, BLEU which we used for our project. While working on the project, we as a team and I individually faced a lot of challenges when working with PySpark for this project, as it does not have many deep learning models to work with especially LSTM and CNN, which are the backbone of our project. Because of lack of references of both academic research and projects online, we could not find enough substance to rely on but eventually found our way around it by implementing different ways and reading a lot papers on the generic working of these models. Overall it was a great learning experience and has added on to my tech stack heavily.

Shravani Hariprasad - As someone with a background in Electronics and Instrumentation, I was intrigued to learn about intelligent models. This research and project allowed me to do just that. Working with deep learning models provided me with an opportunity to understand how images can be used as a dataset to generate captions. It was a valuable experience to work with two distinct types of datasets and to navigate the associated challenges. While we initially faced difficulties when working with LSTM and CNN models on PySpark due to the limited resources available online, we overcame this by relying on the basics of models referenced in academic papers found through Google Scholar. This project has taught me a number of valuable skills, including how deep learning models function and how to implement these techniques using GCP and PySpark. Although implementing this project in PySpark was quite tedious, I gained significant experience in this area. The classes and doubt-clearing sessions were incredibly helpful in enabling me to complete this project. Working on deep learning models with PySpark was a challenge, but the support of my professor was instrumental in helping me to overcome these difficulties. Additionally, I have learned how to utilize and apply evaluation metrics such as BLEU, ROUGE, and METEOR to these models. I have also learnt how to use other models like Inception V3, VGG16 as an alternative to xception model though I have not implemented them in this project. It helped me levearge my skills in modeling, hyper paramter tuning which are the crucial steps in machine learning.

Piyush Jadhav - I now have a thorough understanding of the fundamental ideas, methods, and difficulties involved in creating image captions thanks to my training. I now know how to assess and explain the content of photos by combining computer vision and natural language processing methods. I've discovered that producing coherent and contextually appropriate captions for images requires the use of language models like recurrent neural networks (RNNs) or transformers in addition to the extraction of visual features from images using convolutional neural networks (CNNs). For training image caption generators, I've also learned how crucial it is to have access to huge annotated datasets

that have paired photos and the explanations that go with them. The correlations between visual features and textual descriptions can be learned using these datasets as a foundation. Ambiguity and Subjectivity: It might be difficult to create correct captions that are clear and unambiguous since images can be subject to different interpretations. Context and Coherence: It can be difficult to create captions that are both contextually appropriate and coherent.