

INSTITUTE FOR ADVANCED COMPUTING AND

SOFTWARE DEVELOPMENT AKURDI, PUNE

## Documentation On

**“Image Caption Generation using Deep Learning”**

PG-DBDA SEP 2022

*Submitted By:*

**Group No: 03**

**Ajit Yadav**

**Roll no. 229302**

# Mr. Rohit Puranik Dr. Shantanu Pathak

**Centre Coordinator Project Guide**

**Contents:**

1. [**Introduction**](#_bookmark0)[**1**](#_bookmark0)
   1. [PROBLEM STATEMENT 1](#_bookmark1)
   2. [Abstract 1](#_bookmark2)
   3. [Use case 2](#_bookmark4)
2. [**Overall Description**](#_bookmark5)[**3**](#_bookmark5)
   1. [Workflow of Project: 3](#_bookmark6)
   2. [Data Preprocessing and Cleaning 3](#_bookmark8)
   3. [VGG16](#_bookmark11) 5

2.3.1 [Import Modules](#_bookmark18) 7

2.3.2 [Extract Image Features](#_bookmark19) 9

2.3.3 [Load the Captions Data](#_bookmark20) 11

2.3.4 [Preprocess Text Data 1](#_bookmark24)2

2.3.5 [Train Test Split 1](#_bookmark28)4

2.3.6 [Model Creation](#_bookmark32) 15

2.3.7 Visualize the Results………………………………………………………………………………………………………………..19

2.3.8 Test with Real Image………………………………………………………………………………………….…………………….21

3. Flow of Project………………………………………………………………………………………………………………………………..24

4. Applications and Future scope ………………………………………………….………….……………………………………….25

5. Conclusion …………………………………………………………………………………………..………………………………………….26

6. References ………………………………………………………………………………………………………………………………………27

### INTRODUCTION

### 1.1Problem Statement:

### To develop a system for users, that can automatically generate a textual description of an image. This involves teaching a machine learning model to understand the visual content of an image and use that understanding to produce a coherent sentence that describes the objects, people, and actions depicted in the image.

### 1.2 Abstract:

### Image captioning is an interesting and challenging task with applications in diverse domains such as image retrieval, organizing and locating images of users’ interest etc. It has huge potential for replacing manual caption generation for images and is especially suitable for large scale image data. Recently, deep neural network based methods have achieved great success in the field of computer vision, machine translation and language generation. In this project, we propose an encoder-decoder based model that is capable of generating grammatically correct captions for images. This model makes use of VGG16 (Pre-Trained Model) as encoder and LSTM as decoder. To ensure the complete ground truth accuracy, the model is trained on the labelled Flickr8k dataset.

IACSD-PG-DBDA-SEP-22

1

**1.3** **Use Case:**

An image caption generator has several use cases across various fields.

**Social Media:** Social media platforms such as Facebook, Instagram, and Twitter can use image caption generators to help users with visual impairments access the content shared on the platform. The image caption generator can automatically generate a description of the images that are shared on the platform.

**E-commerce:** E-commerce websites can use image caption generators to improve the search ability and accessibility of their product catalogs. The generator can automatically generate descriptions of the products and help users find what they are looking for.

**Healthcare:** Medical imaging such as X-rays, CT scans, and MRIs can be described using an image caption generator. This can help doctors and healthcare professionals better understand and diagnose medical conditions.

**Entertainment:** Image caption generators can be used to enhance the user experience of online media platforms such as YouTube and Netflix. The generator can automatically generate captions for the videos and make them more accessible to a wider audience.

**Education:** Image caption generators can be used in educational settings to help students with visual impairments access visual content such as charts, graphs, and diagrams. The generator can automatically generate descriptions of the visual content and help students understand the material.

**Autonomous Vehicles:** Image caption generators can be used in autonomous vehicles to help them better understand and navigate their surroundings. The generator can automatically generate descriptions of the objects and scenes that the vehicle encounters and help it make better decisions.

**2.** [**Overall Description**](#_bookmark5)

**2.1 Workflow of Project:**

The diagram below shows the workflow of this project.

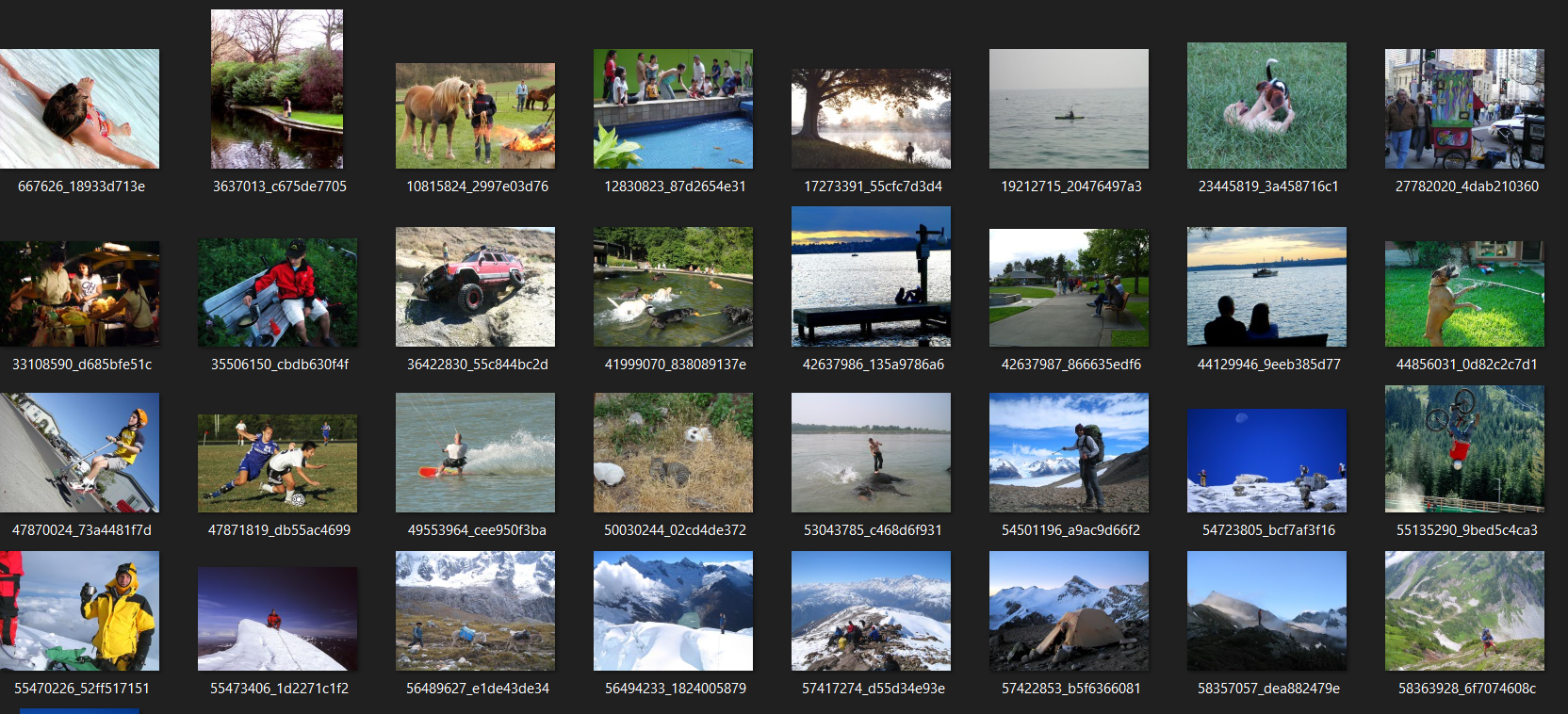
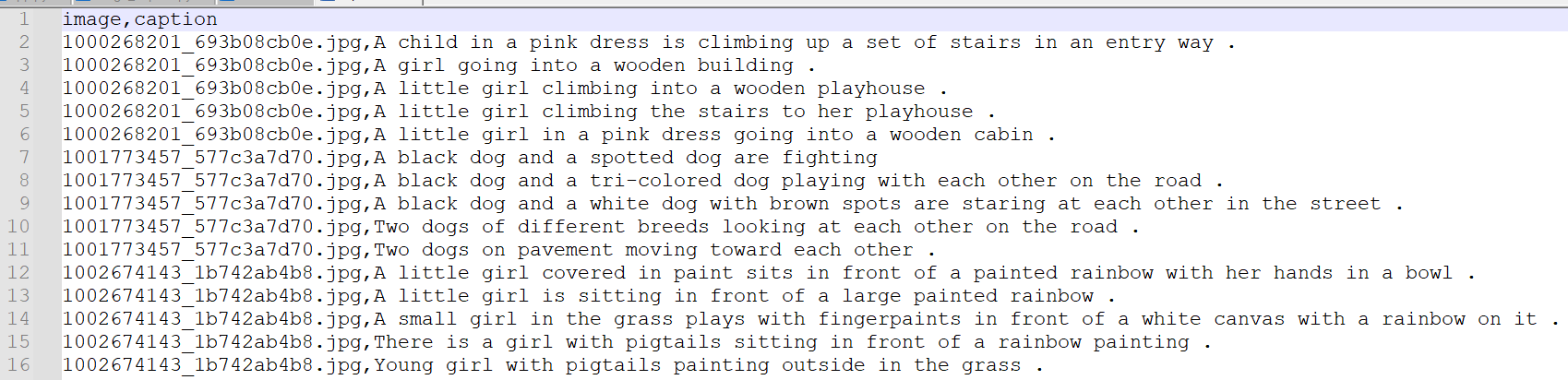
*Figure 2.1 Workflow Diagram*

### 

**2.2 About Datasets:**

The Flickr8K dataset is a publicly available dataset consisting of 8,091 images that have been annotated with textual descriptions. The dataset was created by gathering images from the photo-sharing website Flickr and then annotating each image with five different captions. The captions were written by different people to capture a range of perspectives and interpretations of the same image. The dataset contains a wide variety of images, including landscapes, animals, people, and objects. The images have been resized to a resolution of 500 pixels on the longest side, and each image is accompanied by five captions in plain text format.

The Flickr8K dataset has been widely used for research in computer vision and natural language processing, particularly for tasks such as image captioning and multimodal machine learning. Text file contains 5 different caption for each image.



*Figure 2.2 About Dataset*

**2.2.1 VGG16 Model:**

The VGG16 model is a convolutional neural network (CNN) architecture that was proposed by the Visual Geometry Group (VGG) at the University of Oxford in 2014. It achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014, and has since been widely used as a pre-trained model for various computer vision tasks.

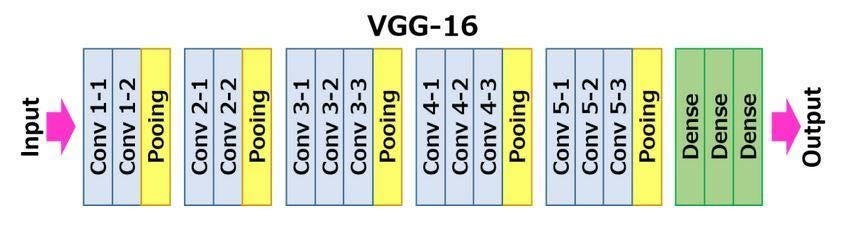
The VGG16 architecture consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. It has a fixed input size of 224x224 RGB images and produces a 1000-dimensional output vector that represents the probabilities of the input image belonging to each of the 1000 classes in the ImageNet dataset.

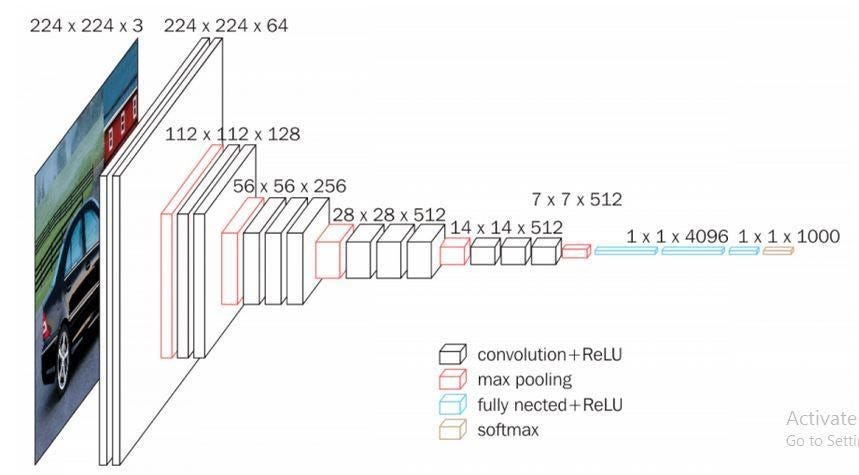
The key innovation of the VGG16 architecture is the use of small 3x3 convolutional filters throughout the network, which allows the model to learn more complex and non-linear features while keeping the number of parameters manageable. VGG16 also uses max pooling and dropout layers to prevent overfitting.

Because of its strong performance on the ILSVRC, the pre-trained VGG16 model has been used as a feature extractor for various computer vision tasks, such as image classification, object detection, and image segmentation. The model is available in the TensorFlow and Keras libraries, and can be easily fine-tuned on a new dataset for a specific task.

**VGG16 Architecture:**

The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer. VGG16 takes input tensor size as 224, 244 with 3 RGB channel.





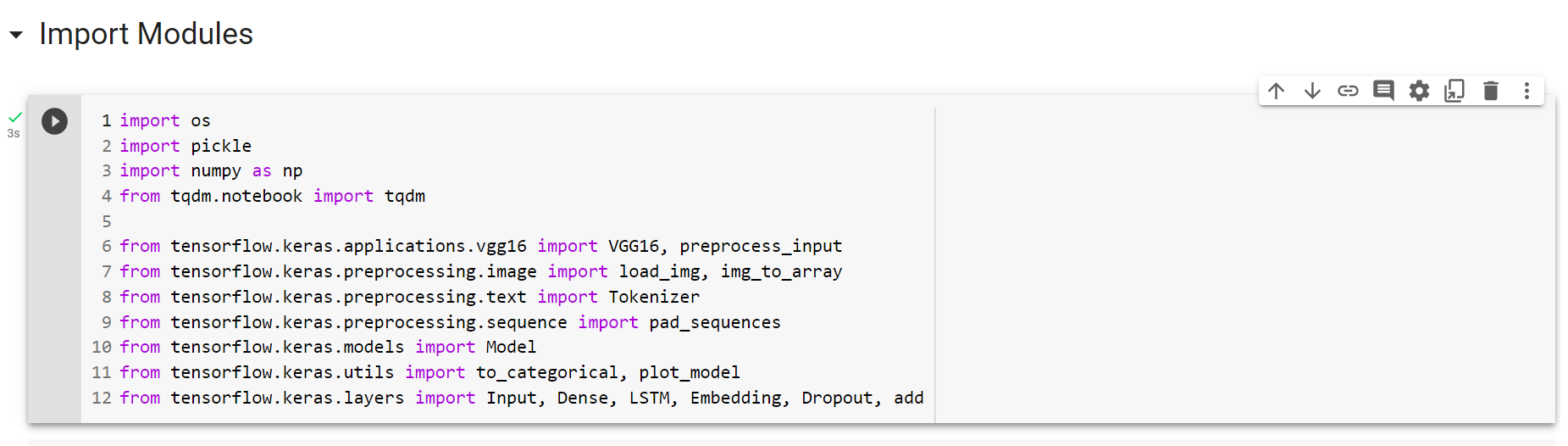
*Figure 2.3 Architecture of VGG16 Model*

Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2.

The convolution and max pool layers are consistently arranged throughout the whole architecture. Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters.

Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

**2.3.1 Import Modules:**

****

**OS Module:**

OS provides a way of using operating system dependent functionality like reading or writing to the file system, working with environment variables, and executing system commands.

os.path.join(path1, path2): Joins two paths together to form a complete path

**Pickle Module:**

The pickle module is used for object serialization and deserialization. It allows you to convert a Python object hierarchy into a byte stream that can be stored or transmitted, and then reconstruct the object hierarchy from the byte stream. This process is known as "pickling" and "unpickling".

The pickle module is commonly used for tasks such as:

Saving and loading machine learning models, so that they can be reused later Caching expensive computations, so that they can be reused instead of recalculated every time Saving and loading program state, so that a program can be resumed from where it left off after a crash or restart. The pickle module provides two main functions: **dump and load.**

with open('/content/drive/MyDrive/Colab Notebooks/Project/working/features.pkl', 'rb') as f:

    features = pickle.load(f)

pickle.dump(features, open(os.path.join(WORKING\_DIR, '/content/drive/MyDrive/Colab Notebooks/Project/working/features\_ver3.pkl'), 'wb'))

**Numpy module:**

NumPy is a powerful library for numerical computing in Python and is widely used for tasks such as scientific computing, data analysis, and machine learning.

**Tensorflow.keras module:**

TensorFlow is a popular open-source library for numerical computation and machine learning, which provides a flexible platform for building and deploying machine learning models. TensorFlow includes a wide range of tools and libraries for building and training deep learning models, including the Keras API. Keras is a high-level deep learning API written in Python, which provides a user-friendly interface for building, training, and evaluating deep learning models. Once VGG16 and the preprocess\_input function are imported, you can use them to classify images or fine-tune the pre-trained model on a new dataset.

**img\_to\_array, load\_img :**

img\_to\_array is used to import two functions from the Keras preprocessing module in TensorFlow: load\_img and img\_to\_array.

load\_img is a function that loads an image file from disk and returns a PIL (Python Imaging Library) image object. It takes two arguments: path is the path to the image file, and target\_size is a tuple specifying the size to which the image should be resized. If target\_size is not specified, the function returns the original size of the image.

img\_to\_array is a function that converts a PIL image object into a NumPy array. It takes a single argument: img is the PIL image object to be converted.

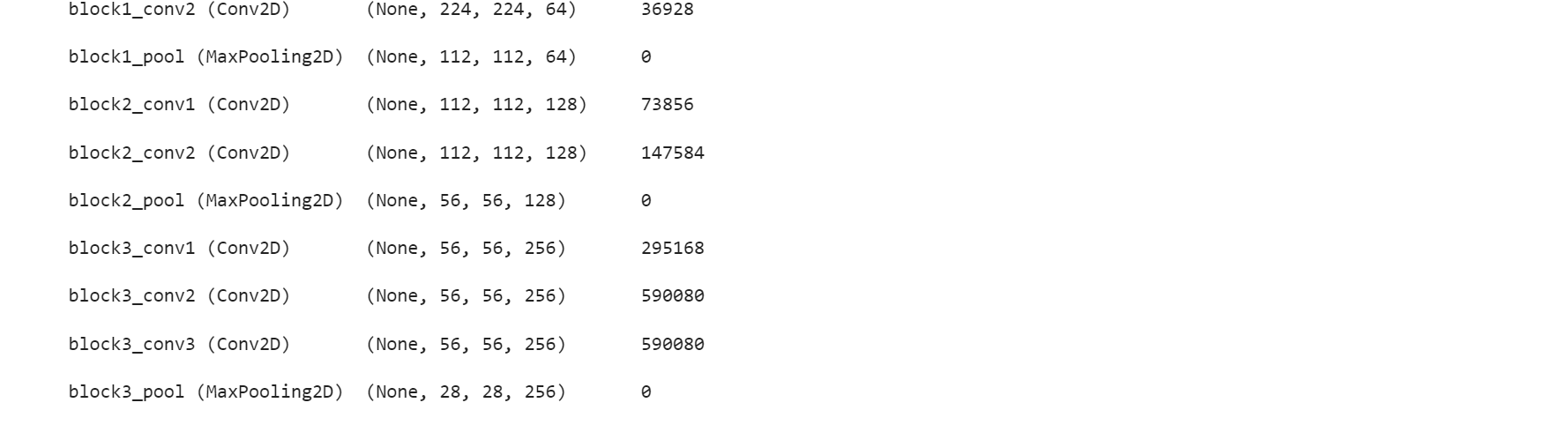
These two functions are often used together to load and pre-process image data for deep learning models.

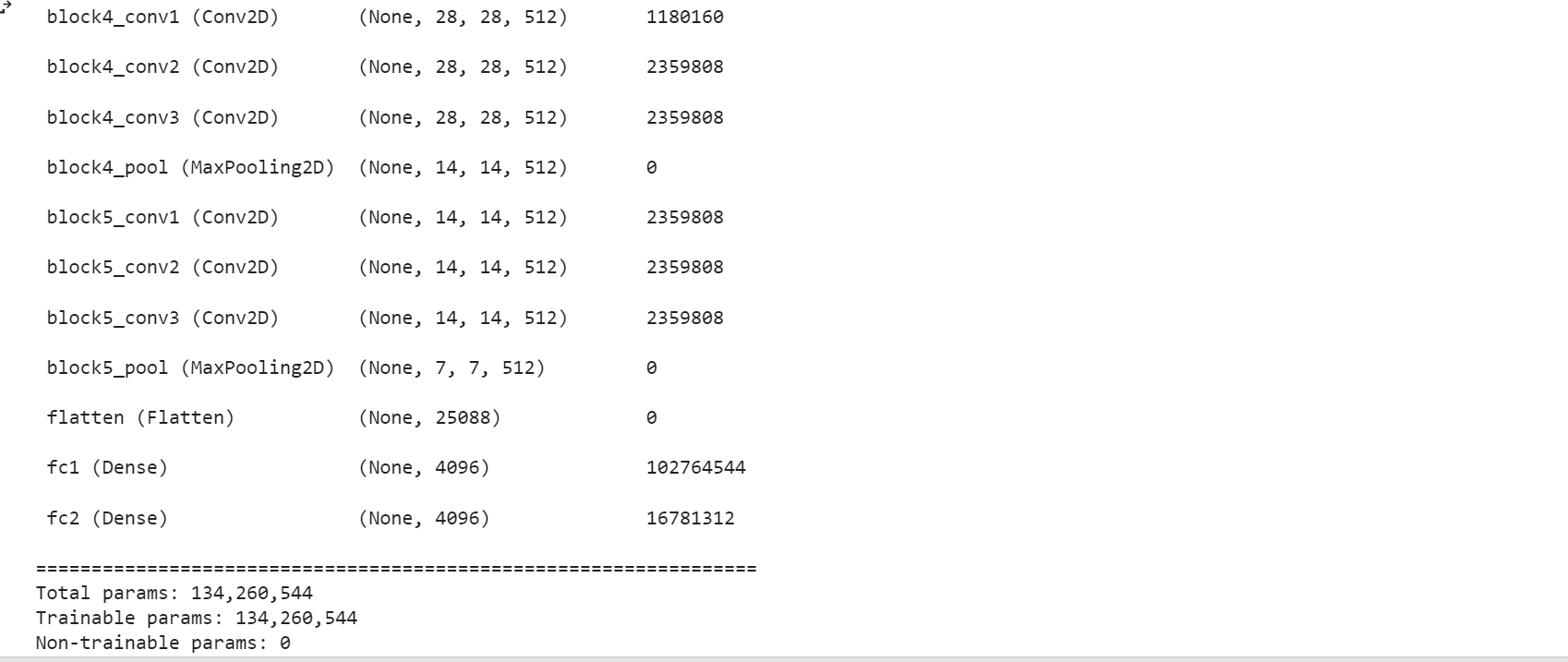
**2.3.2 Extract Image Features:**

In image caption generation, one common approach for extracting image features is to use a pre-trained convolutional neural network (CNN) such as VGG16. These CNNs are trained on large datasets (such as ImageNet) to classify images into different categories.

To extract image features, the pre-trained CNN is typically used as a feature extractor. The input image is passed through the CNN, and the output of one of the intermediate layers (before the fully connected layers) is extracted as the image feature representation. This feature representation can then be used as input to a separate natural language processing (NLP) model, such as a recurrent neural network (RNN), to generate captions





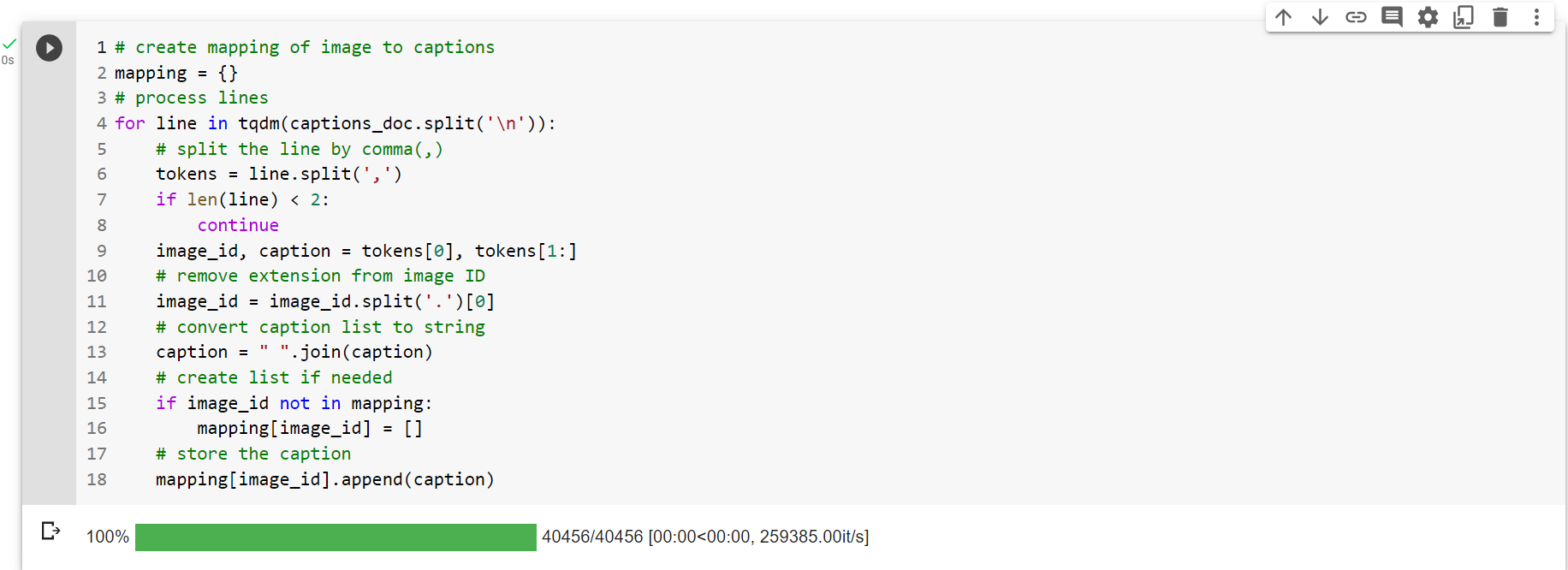




**2.3.3 Load the Captions Data**

We reads the contents of a file named 'captions.txt' located in the directory specified by the variable BASE\_DIR. It then skips the first line of the file using the next(f) function call, and reads the rest of the file into a string variable named captions\_doc. And we are Mapping each captions to it’s image ID.





**2.3.4 Preprocess Text Data:**

We defined a function named clean that takes a single argument mapping, which is assumed to be a dictionary where each key corresponds to an image and the associated value is a list of captions describing that image. The function cleans and preprocesses each caption in the mapping by performing the following steps:

1. Converts each caption to lowercase using the .lower() method.

2. Removes any non-alphabetic characters (e.g. digits, special characters) from the caption using the .replace() method with a regular expression pattern [^\w].

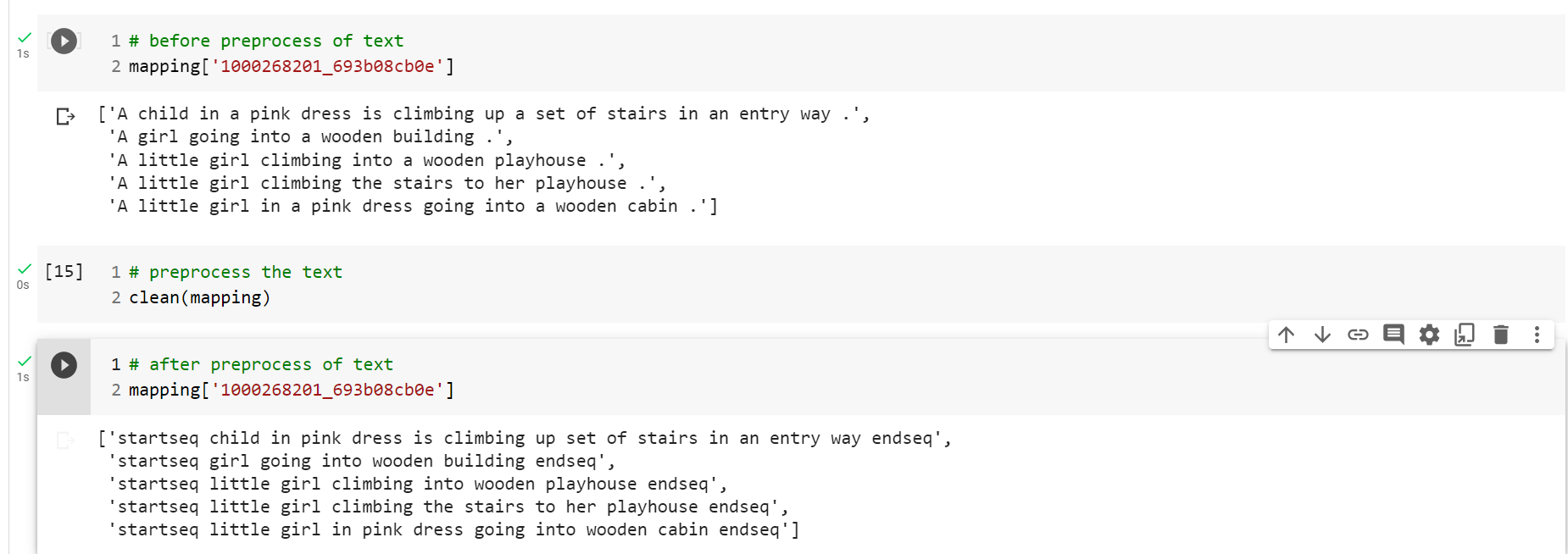
3. Removes any extra whitespace characters (e.g. multiple spaces in a row) from the caption using the .replace() method with the pattern \s+.

4. Adds special "start" and "end" tags to the caption to indicate the beginning and end of the caption using the str.join() method with a list comprehension.

The cleaned captions are stored back in the original mapping dictionary, overwriting the original values for each key.



We have displayed before and after Preprocessing of text data:



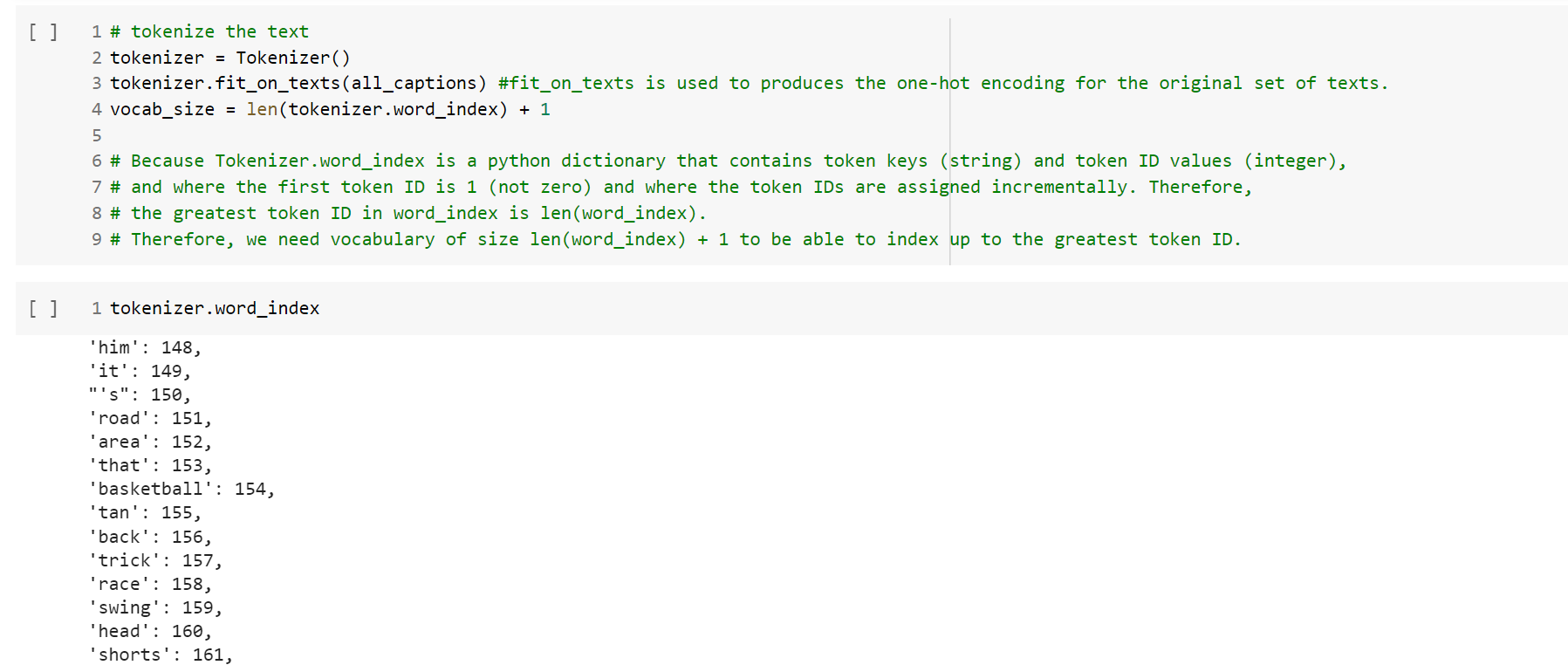
**Tokenize the text data:**

We are creating Keras Tokenizer class by calling the constructor with no arguments. This creates a new Tokenizer object that can be used to convert sequences of text into sequences of integers suitable for use as input to a neural network.

The fit\_on\_texts method is then called on the Tokenizer object, passing in a list of all the captions in the dataset as its argument. This method updates the internal state of the Tokenizer object to create a vocabulary of all the unique words in the text, and assigns a unique integer index to each word.

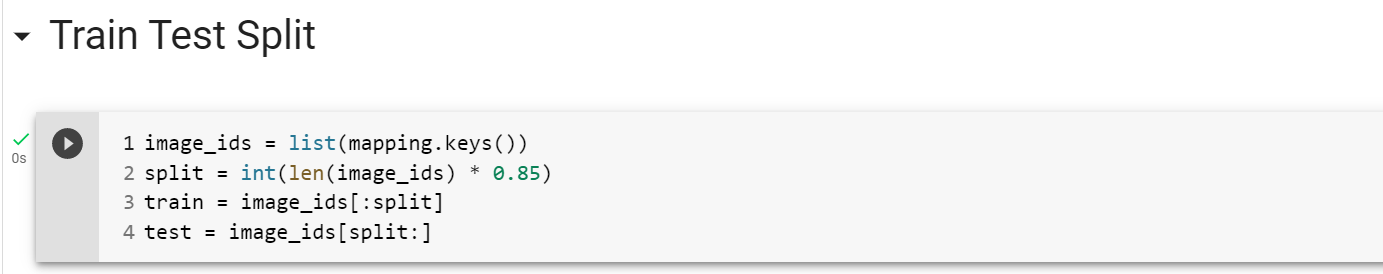
Finally, the vocab\_size variable is set to the size of the vocabulary created by the Tokenizer, which is the total number of unique words plus one (to account for the special "out of vocabulary" token).

To prepare the vocabulary for the captions in the dataset, which will be used to represent each caption as a sequence of integers for input into a neural network. The Tokenizer object provides a convenient way to perform this transformation, and the resulting vocabulary size is used to set the size of the embedding layer in the neural network.



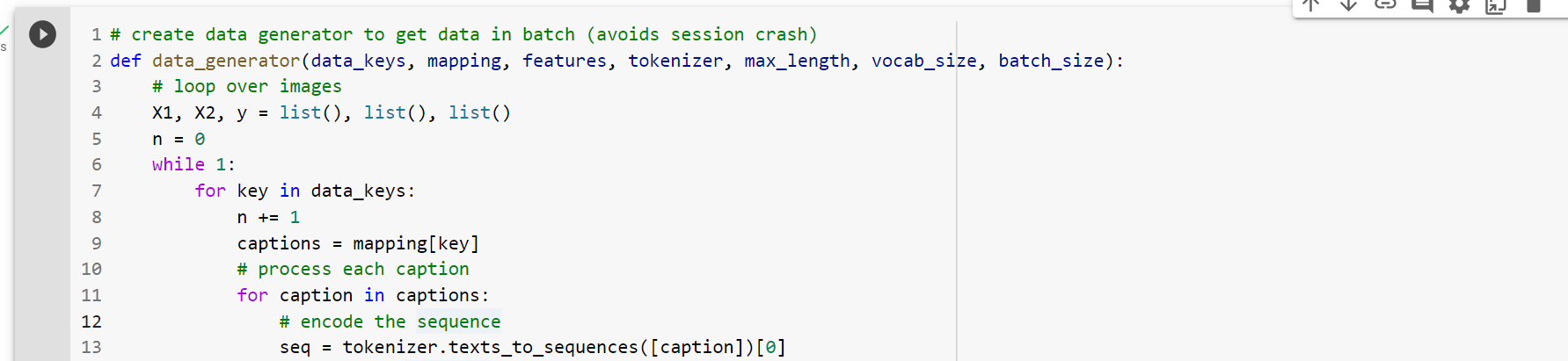
**2.3.5 Train Test Split:**

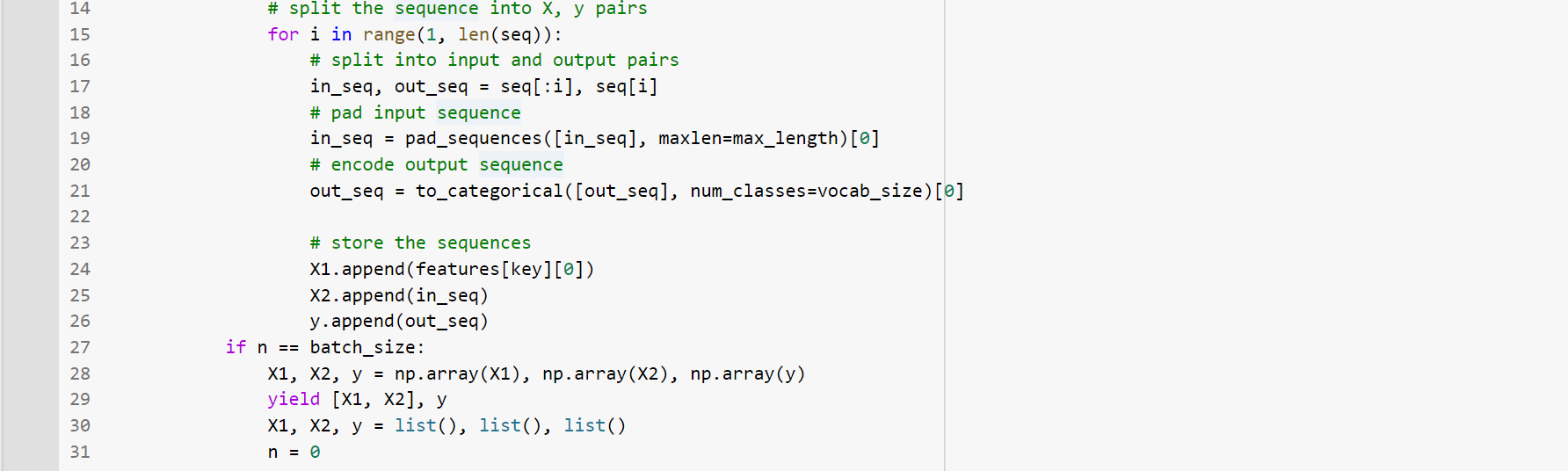
We are splitting the image ids into training and testing sets for use in a machine learning model. The split is based on a percentage of the total number of image ids, with 85% of the image ids assigned to the training set and 15% assigned to the testing set.



**data\_generator():**

We have created data generator function to get data in batch (avoids session crash).

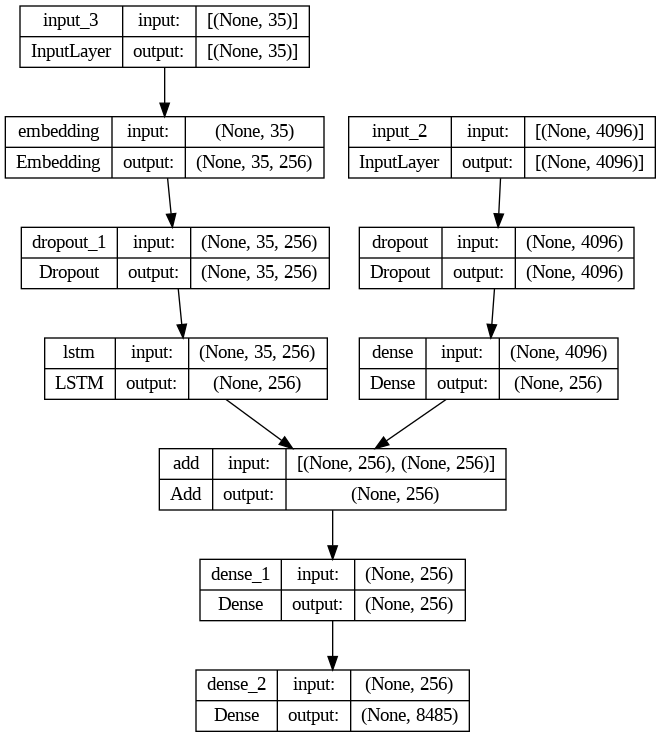


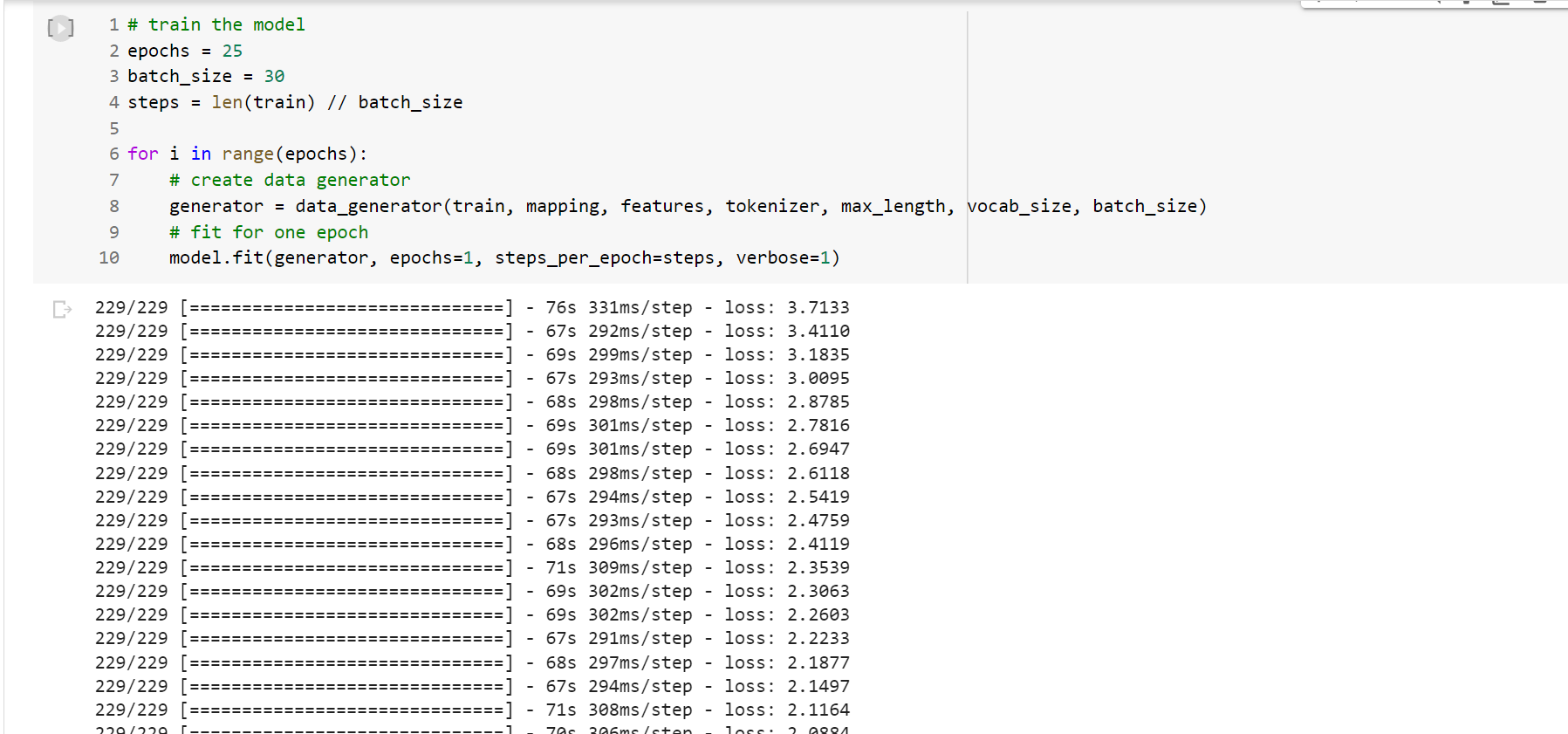


**2.3.6 Model Creation:**

We have defined an image captioning model with an encoder-decoder architecture. The encoder takes in the image features as input and passes them through a series of layers, including dropout and a fully connected layer, to extract a feature vector. The decoder takes in the captions as input and passes them through an embedding layer and an LSTM layer to extract a sequence of feature vectors. The decoder then combines the image feature vector with the sequence of feature vectors from the LSTM layer using an element-wise addition and passes the resulting vector through a series of fully connected layers to generate the final output. The model is compiled using the Adam optimizer and categorical cross-entropy loss function.

****

****

****

We defined the maximum number of epochs to 25 and the batch size to 30. We calculated the number of steps needed in each epoch for training and validation data.

And we use a data generator to generate arrays of these sequence of the size of the batch progressively. We used this approach to avoid reaching RAM and GPU limits. Otherwise, we weren’t able to train the model with the RAM available in Google Collaboratory. The data generator receives the training data shuffled and works with image ids to take the images that correspond to the captions.

Rnn Model1: It has two inputs. In the text submodel it has an Embedding, Dropout and LSTM layers. In the image submodel, it has a Dropout and a Dense layer. Then it adds these two submodels and finally there are two Dense layers, the last one having vocabulary size with softmax to train the model with the RAM available in Google Colaboratoty. The data generator receives the training data shuffled and works with image ids to take the images that correspond to the captions.

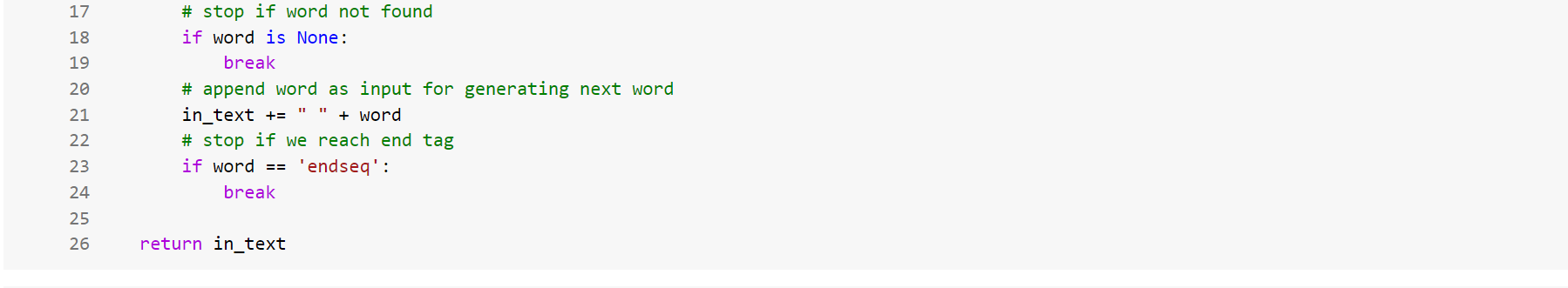
**2.3.7 Generate Captions for the Image:**

The **idx\_to\_word** function takes an integer representing a word index and a tokenizer object as inputs, and returns the corresponding word. It iterates over the word\_index dictionary of the tokenizer, which maps words to integer indices, and returns the word that has the given integer index. If the integer index is not found in the word\_index, it returns None. This function can be used to convert the output of the model, which is a sequence of integer indices representing the predicted words, back into a sequence of words.

The function **predict\_caption()** takes as input the trained model, an image, a tokenizer, and the maximum length of the sequence, and returns the predicted caption for the image.

The function starts by adding a special start tag startseq to the input sequence. It then iterates over the maximum length of the sequence, encoding the input sequence, padding it, and using the model to predict the next word. The predicted word is then converted to its corresponding string form using the idx\_to\_word() function. If the predicted word is not found, the loop is stopped. Otherwise, the predicted word is appended to the input sequence for generating the next word. If the predicted word is the end tag endseq, the loop is stopped.

****

****

The BLEU score is a metric used to evaluate the quality of generated text, such as machine translation or image captioning. It measures how similar the generated text is to a set of reference texts. The score ranges between 0 and 1, where a higher score indicates a better match with the reference texts.The BLEU score is computed using n-grams, which are sequences of n words. The score is a weighted average of the n-gram precision, which is the percentage of n-grams in the generated text that appear in the reference texts. The weights give more importance to longer n-grams.

In this code, we are computing the BLEU-1 and BLEU-2 scores using the corpus\_bleu function from the nltk.translate.bleu\_score module. The actual list contains the reference texts, which are the captions associated with the test images. The predicted list contains the generated texts, which are the captions predicted by the model for the test images.

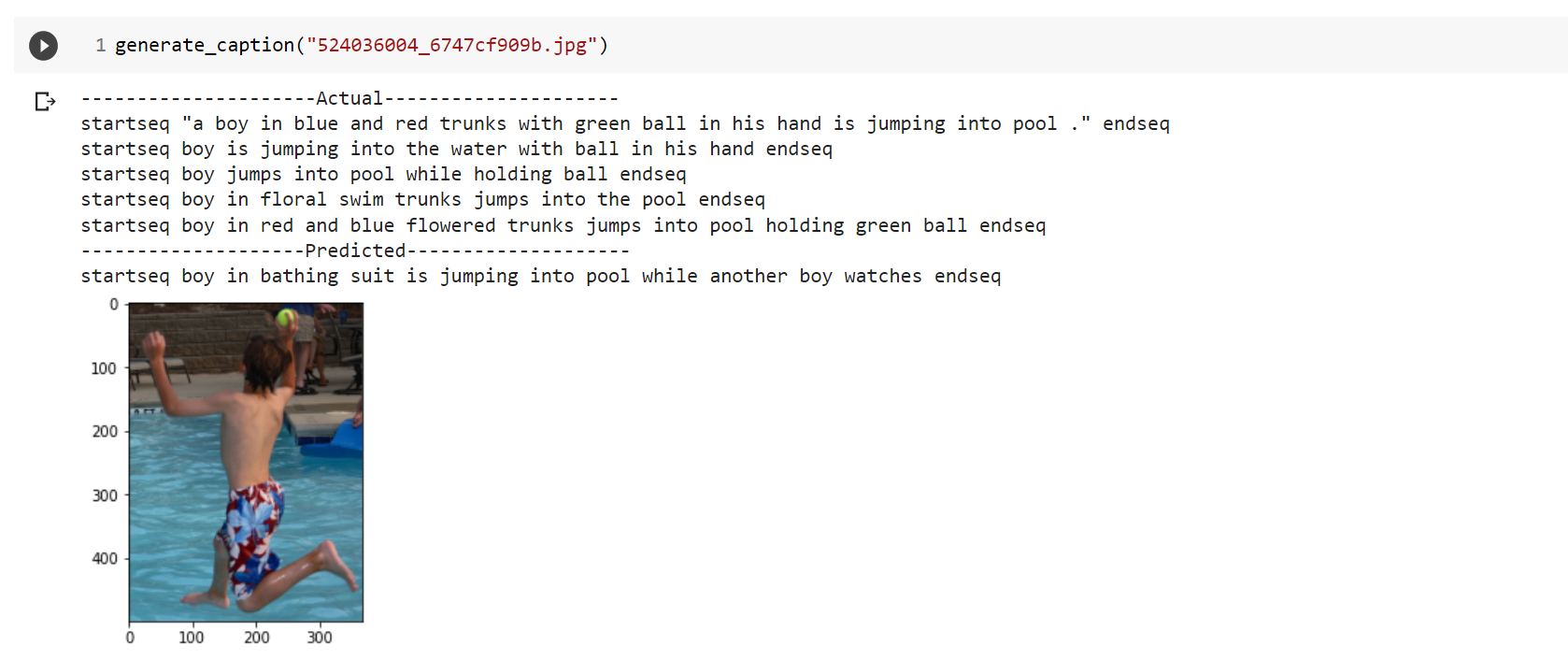
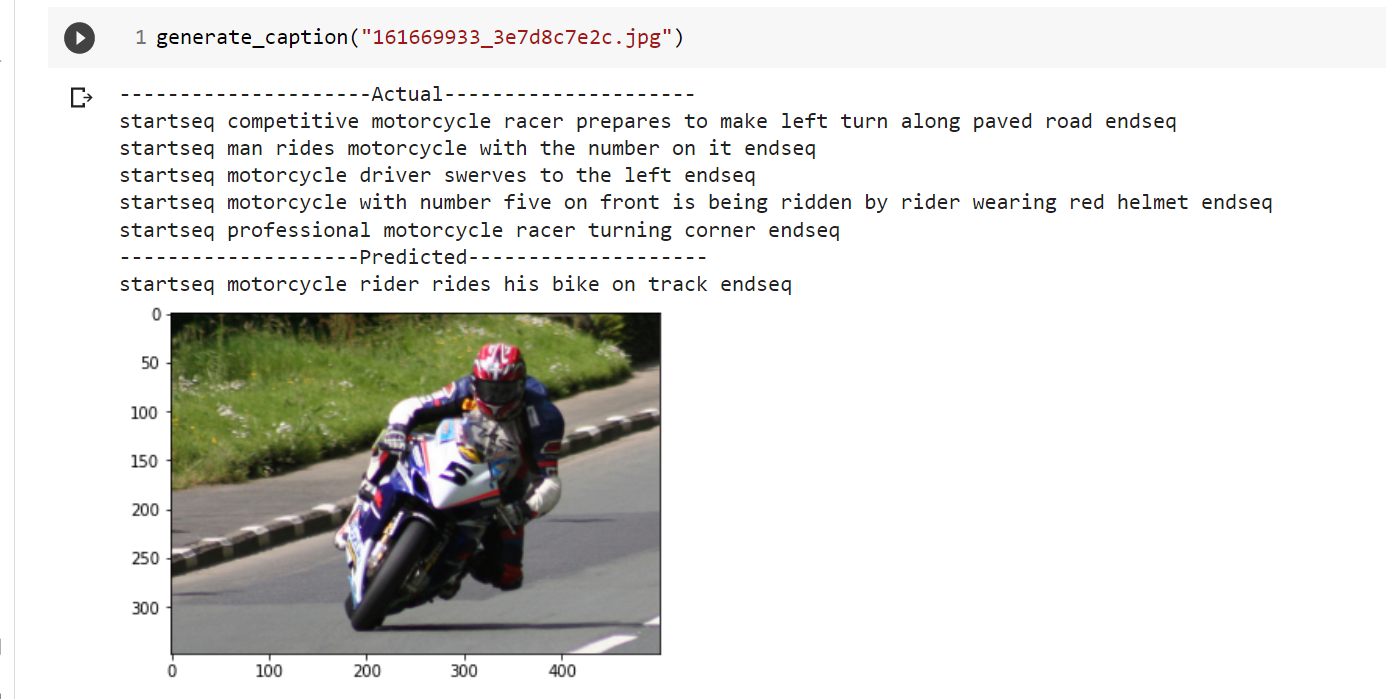
The BLEU-1 score is computed using only unigrams (n=1), while the BLEU-2 score is computed using both unigrams and bigrams (n=2). The weights used in the computation are (1,0,0,0) for BLEU-1 and (0.5,0.5,0,0) for BLEU-2, which give equal importance to unigrams and bigrams.

****

**2.3.8 Visualize the Results:**

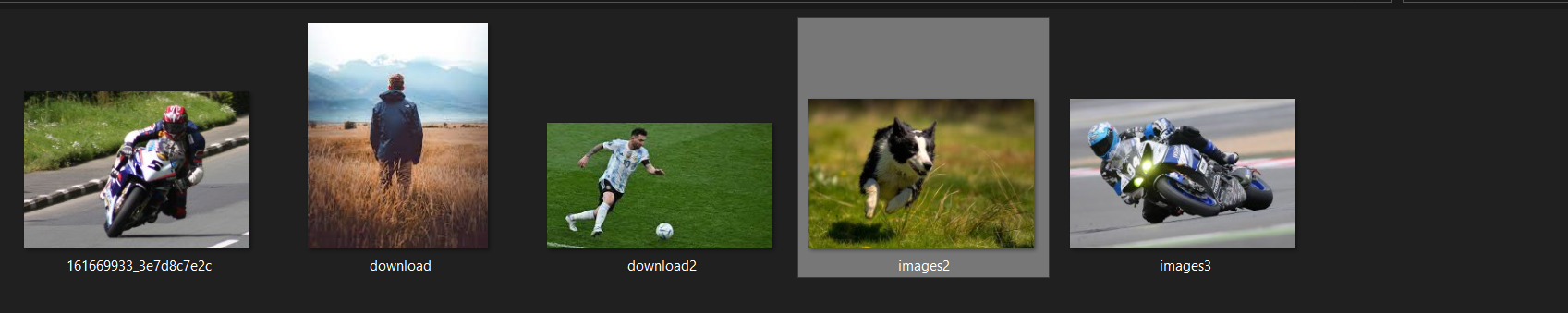
We are using generate\_caption() function for caption generation , and we are displaying actual vs predicted caption .

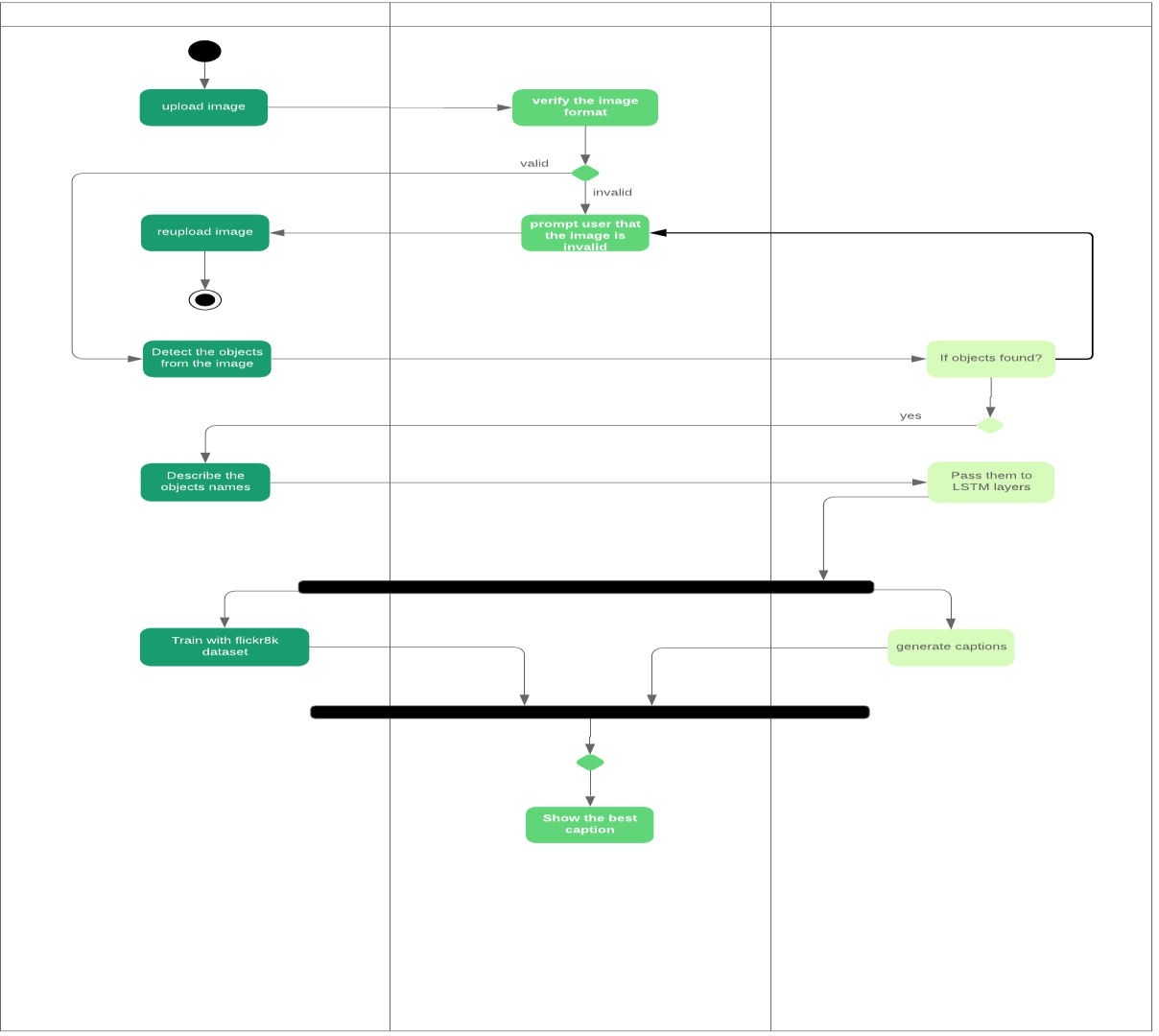
****

****

**2.3.9 Test with Real Image:**

****

****



**Fig : Flow of the project**

**Applications and Future scope of Image Caption Generator:**

* It can be used to assist the blind using text-to-speech.
* It can be used to convert captions for images in social feed as well as messages to speech which will enhance social medial experience of users.
* It can also be used for educational purposes as young children can be assisted about recognition of objects and learning the English language.
* It is also helpful in field of robotics as environmental insights can be provided through natural language representation.
* It can also be used for image searches and indexing purposes on internet if images present on the internet have captions

**Conclusion**

In conclusion, an image caption generator is a powerful tool that uses deep learning algorithms to analyze an image and generate a descriptive and accurate caption for it. This technology has various applications, including improving accessibility for the visually impaired, enhancing social media engagement, and supporting image retrieval in large databases.

Image caption generators have been developed using various deep learning architectures, including Convolutional Neural Networks (CNNs) and Long Short Term Memory (LSTM), and have been trained on large datasets of images with corresponding captions.

While image caption generators have shown impressive results, there is still room for improvement in terms of the accuracy and diversity of generated captions. Researchers are continuing to work on developing more advanced models that can capture more nuanced and contextual information from images, as well as incorporating more knowledge about language and grammar.

Overall, image caption generators have the potential to revolutionize the way we interact with and understand visual content, and will likely continue to evolve and improve in the years to come.

**References**

<https://www.kaggle.com/code/aswintechguy/image-caption-generator-tutorial-flickr-dataset>

<https://www.youtube.com/watch?v=fUSTbGrL1tc>

<https://www.kaggle.com/datasets/shadabhussain/flickr8k>

<https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/>

<https://www.analyticsvidhya.com/blog/2021/12/step-by-step-guide-to-build-image-caption-generator-using-deep-learning/>