Introduction:

The project is an Image Caption Generator, designed to automatically generate descriptive text captions for images. Leveraging deep learning techniques and natural language processing, the system processes input images and produces human-readable captions that describe the visual content. This technology has diverse applications, including improving accessibility for visually impaired individuals, enhancing content indexing and search, aiding in educational materials creation, and automating content generation. The report explores the development process, implementation, performance evaluation, and potential use cases of this innovative image captioning solution.

Application of this Project:

An image caption generator has a wide range of potential use cases across various domains and industries. Here are some common applications:

- **Accessibility:**

  Image caption generators can make visual content more accessible to individuals with visual impairments. By providing textual descriptions of images, it enables them to understand and interact with the content.

- **Content Indexing and Search:**

  Captioned images are easier to index and search. This can be valuable in content management systems, e-commerce, and image databases. Users can search for images using keywords from the generated captions.

- **Social Media:**

  Social media platforms can use image caption generators to help users add captions to their images. This can improve engagement and make the content more informative.

- **Education:**

  In educational settings, image caption generators can help create learning materials, making images more meaningful. They can also be used for creating alternative text for educational videos and presentations.

- **Content Generation:**

  Image captions can be used to automatically generate content for blogs, websites, and other platforms. For example, in travel or real estate, images can be automatically captioned to describe places and properties.

- **Healthcare:**

In medical imaging, image captions can help doctors and researchers better understand the content of medical images. It can also assist in automating radiology reports.

- **Art and Culture:**

Image caption generators can provide context and descriptions for art collections in museums, making them more accessible to visitors.

- **News and Journalism:**

News outlets can use image caption generators to quickly add descriptive captions to images in their articles, especially for breaking news events.

- **Automatic Video Description:**

In video content, image caption generators can automatically generate descriptions of scenes, making videos accessible to a wider audience

Thus we can see there are plenty of use cases for this project.

Objectives:

In this project we have used Kaggle Flickr 8k dataset which comprises of 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events. One of such sample image along with the captions associated with this has been given below:

```
A child in a pink dress is climbing up a set of stairs in an entry way .
A girl going into a wooden building .
A little girl climbing into a wooden playhouse .
A little girl climbing the stairs to her playhouse .
A little girl in a pink dress going into a wooden cabin .
```

In our project we would try to generate the caption for such image automatically using deep learning model which can be built combining Convolutional Neural Network (for image feature extraction) and Recurrent Neural Network (for text feature extraction). We have used tensorflow.keras model for implementing all deep learning methods. OpenCV (cv2) and pillow have been used for the purpose of image data loading and displaying.

Methodology:

The entire process flow for this project has been described below in stepwise manner:

- **Step 1: Extraction of Image Features**

In this project pretrained VGG16 model has been used for image feature selection. Hence first all the images has been reshaped to (224,224,3) using keras load_img function and necessary preprocessing like mean cantering, scaling, Channel reordering etc. For the model preparation, we have taken only the convolution and fully connected portion of VGG16 model, not the softmax portion because here our sole objective is to extract features from image. So after running this model for each image in our dataset we have got a 1D feature vector of shape (4096,) for each image. Then we have stored the image ID (which is chosen to be image name without extension) and the corresponding feature vector in a dictionary named **'features'.** The VGG16 model used for the image feature extraction purpose has been shown below

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```

- **Step 2: Preprocessing of text data:**

First we have created another dictionary called ‘mapping’. This dictionary basically stores the image ID as its keys and a list which consists of all the captions of that image as the value. One of such example has been shown below:

```
mapping['1000268201_693b08cb0e']
```

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

Now we have done some preprocessing for the text data which involves:

- ✓ Making all characters in lower case
- ✓ Eliminate single letter words
- ✓ Eliminate extra space
- ✓ Adding ‘start ‘ and ‘ end’ at the starting and the ending of each phrase respectively

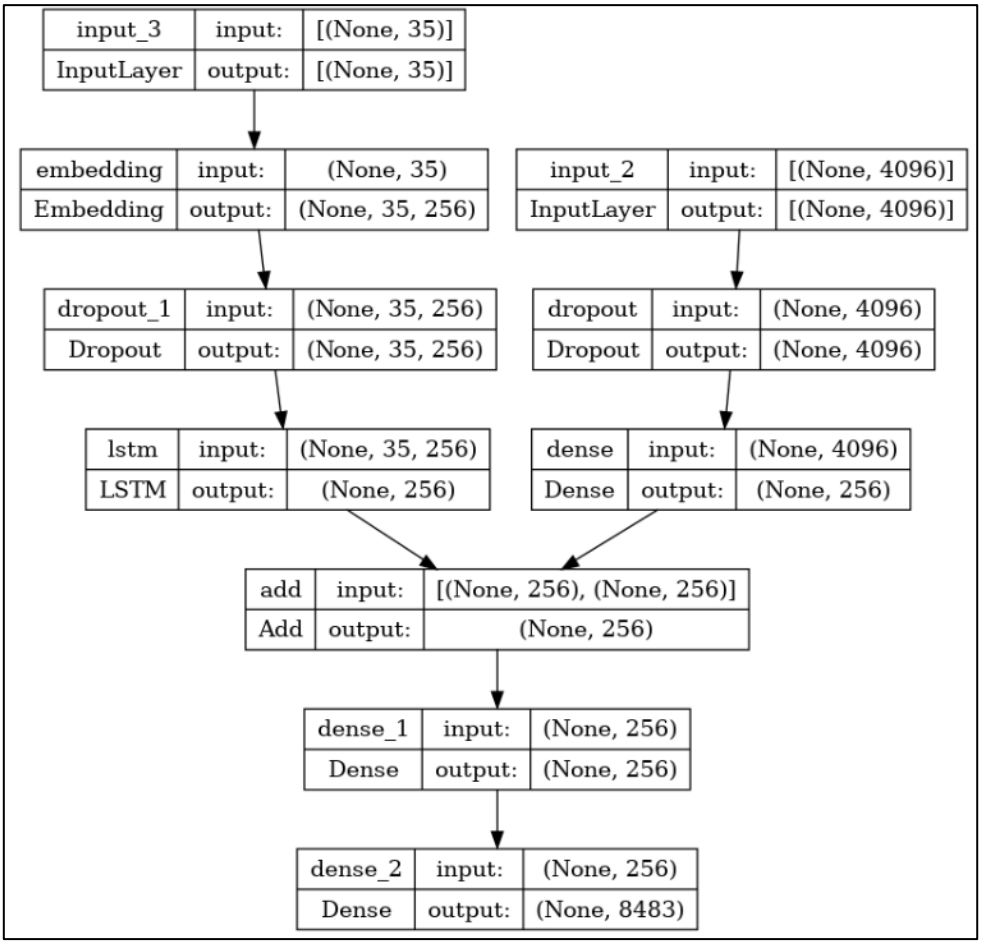
- **Step 3: Natural Language Processing**

Here at first we have gone through each sentences in the mapping dictionary and tokenize the words of the sentences. Along with we have calculated the size of our vocabulary (+1 as padding is used) and max size of sequence has been calculated. Then we have created a data generator function as our dataset is too large and its better to pass it to the model batchwise. The input data has two components, first is the image feature vector and the second is the text itself and the output is also a text sequence. Here, we have assumed each word of the sequence individually as output and all the words prior to that has been assumed to be the input. So say if the sequence is ‘start the dog is jumping over the fence end’ then the input and output sequences are as follows:

INPUT	OUTPUT
Start	The
Start the	dog
Start the dog	Is
Start the dog is	Jumping
Start the dog is jumping	Over
Start the dog is jumping over	The
Start the dog is jumping over the	Fence
Start the dog is jumping over the fence	End

- **Step 4: Finalize Encoder Decoder Model:**

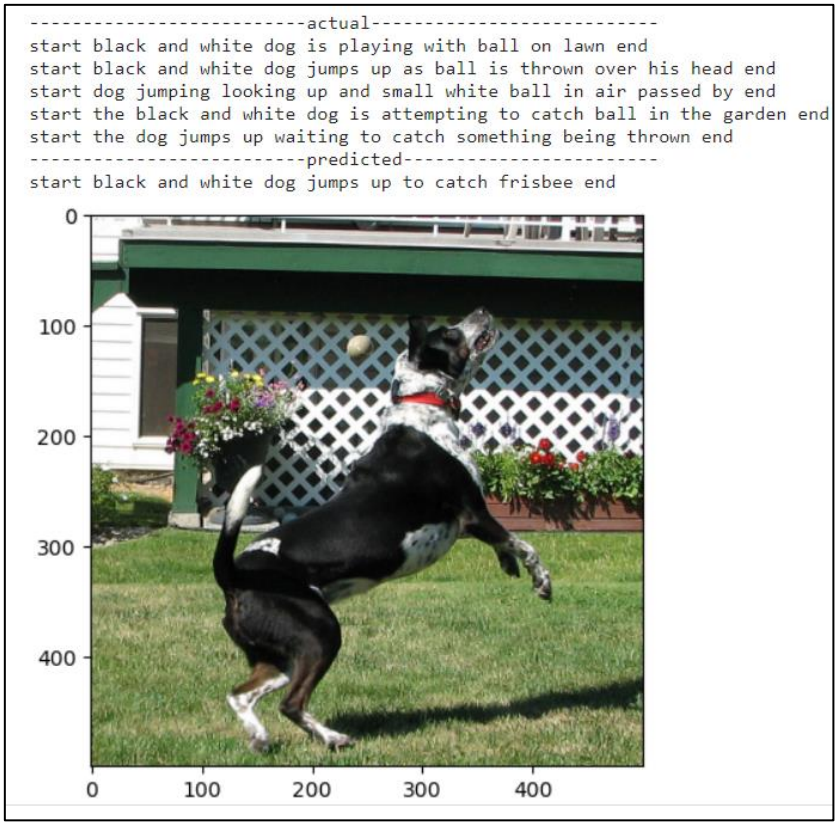
The encoder model for image is formed which takes image feature as input and give a 1D vector of dimension (256,) as output. On the other hand the text encoding model takes text features of size 'max_size' of sequence as input and after vector embedding LSTM model is applied on these vectors. Now, both of this encoded image and text has been taken together as input of our decoder model which converts the input to a 1D array of size (256,) and use softmax regression in next layer to evaluate what word is likely to be occurred next (based on its token value word can be known) and that predicted word is again added to the input to predict next word. So if we just pass image feature vector along with the word 'start ', it can predict the entire caption for the image. The combined model architecture has been shown below:

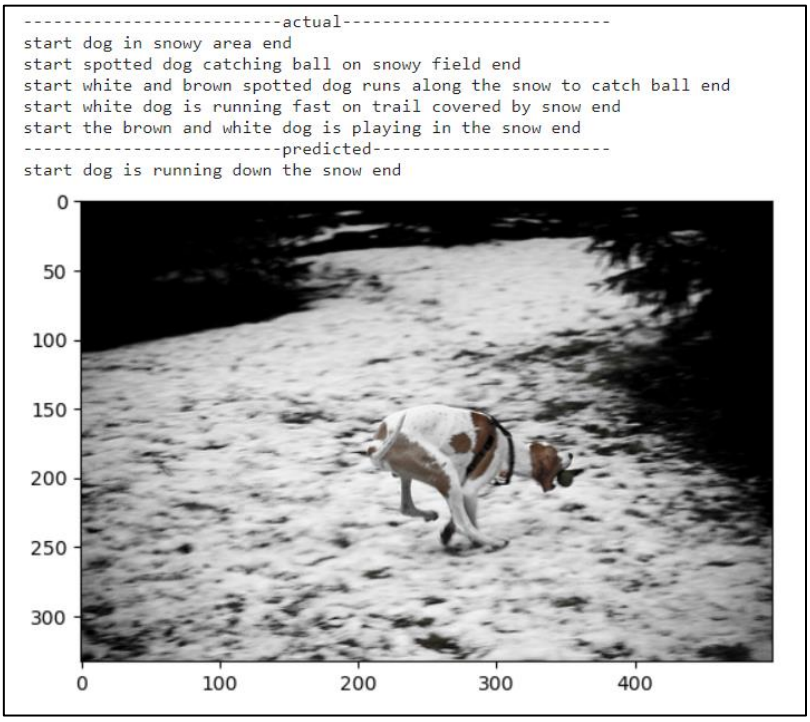


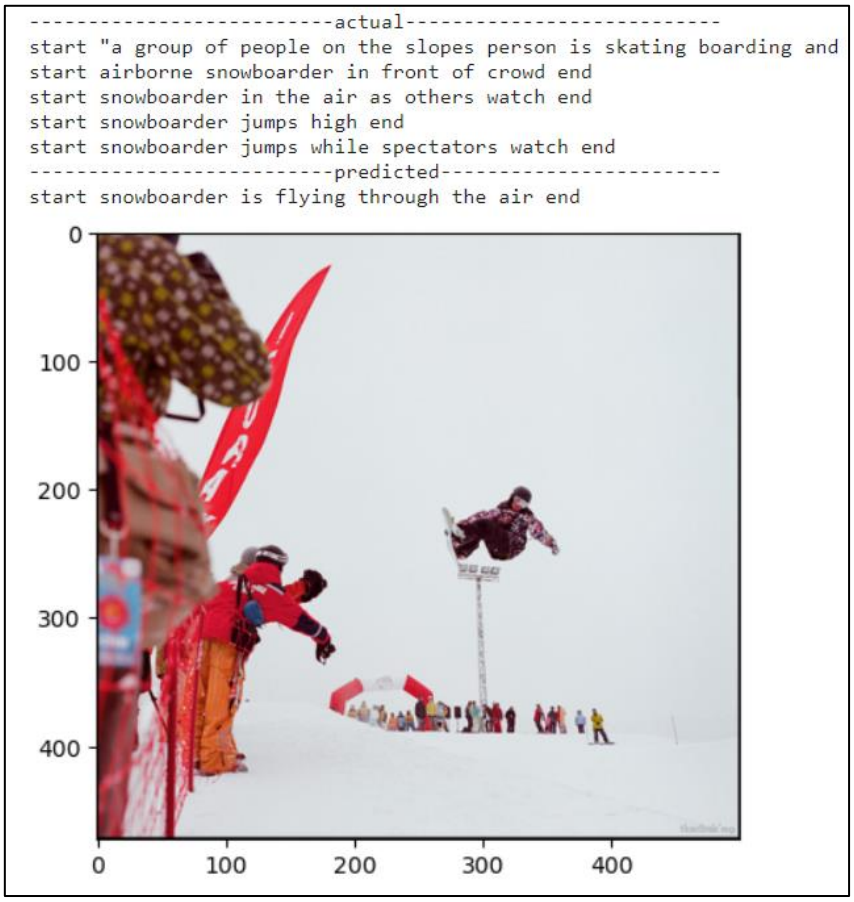
- **Step 5: Evaluation of model:**

For the evaluation of model we cant use standard technique lie confusion matrix or classification report as it's a text generation model. Here we have used BLEU-1 and BLEU-2 score for evaluation which basically calculate what fraction of predicted words exists in actual caption. BLEU-1 score is calculated based on unigram only where as BLEU-2 has been calculated based on bigram. BLEU-1 score obtained by our model is about 0.67 which ca be considered as quite good score in NLP. BLEU-2 score is lesser as it takes

consecutive two words together (0.44). It may be noted these scores can be further improved by increasing the training epoch value which is only 20 here as the training of the entire model is quite time consuming and needs lots of CPU memory (we have use GPU T4 in order to increase the training speed though). Some of the sample predictions along with the actual caption has been shown below:







So we can see our results are quite impressive !!

Conclusion:

In summary, the Image Caption Generator project showcases the successful fusion of computer vision and natural language processing. With its capacity to automatically generate descriptive image captions, the project has significant implications across multiple domains. While the project has met its initial goals, opportunities for improvement and expanded applications exist, making it a promising area for future work. In essence, this image caption generator effectively bridges the gap between visual and textual information, unlocking new possibilities for accessibility and content management.