

1 Summary of the project

Click on the Github link or go copy the following url: https://github.com/Atharv13/Image_Captioning_Project

An image caption generator will analyze the content of an image and generate a descriptive text caption that describes the objects, people, or scenes depicted in the image.

2 Data Pre-processing

2.1 what data have you collected so far and what preprocessing have you done?

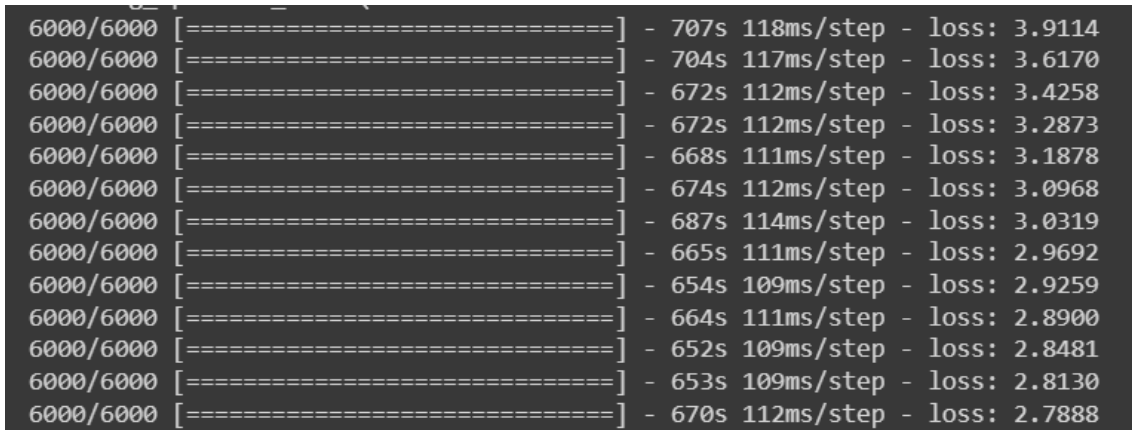
1. We use flicker8k dataset from kaggle. It consists of 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events.
2. First we did the feature extraction of the given images using VGG16 model. We resize the given image in 224*224 pixels. It preprocess the data using the function preprocess input to match the preprocessing steps applied during the training of the VGG16 model. Then, it uses the modified VGG16 model to extract features from the preprocessed image. And lastly it stores the features in the dictionary named features using image ID as the key.
3. In text preprocessing we made a function called clean descriptions which will remove the punctuation marks. It will convert all the text in the lower case. Also it will split each word from the text. We will also remove single character words as it will have no effect on the meaning of the sentence. We will also remove the words which are not alphabetic.
4. We will find the vocabulary of the text by splitting and appending the text into a set as set contains only unique words.
5. We created a save descriptions function is used to save a collection of image descriptions to a text file. It takes a dictionary of image descriptions and a filename as input and writes the descriptions to the specified file.

3 Training with the basic model, validation, and completion of the data pipeline

3.1 Which models did you use, what training/validation accuracy have you achieved?

1. we split the dataset of 8000 images and their captions. For training we used 6000 images, for validation we used 1000 images and for testing we used 1000 images.
2. We will use BLEU score to evaluate the model.
3. BLEU stands for Bilingual Evaluation Understudy score. Which is a metric commonly used to evaluate the quality of machine-generated text, including machine translation and image captioning.
4. The BLEU score measures the similarity between a generated text and one or more reference texts. It calculates a precision score based on the n-grams present in both the generated and reference texts.

5. We achieved a 56 percent similarity between the generated text and actual text of $n = 1$ that is of unigram but it was just 1 epoch we also run it for 20 epochs but it crashed in between , we'll be saving the models intermediately at each step for better recovery expecting better BLEU score

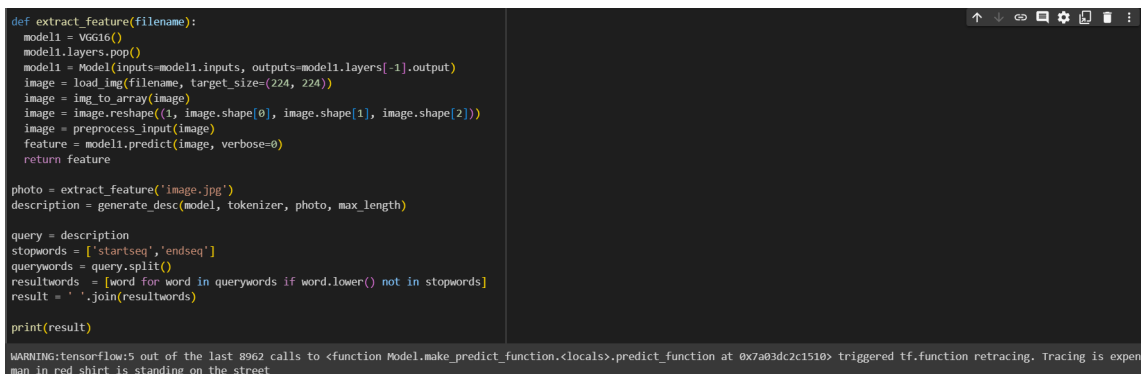


6000/6000	[=====]	- 707s	118ms/step	- loss: 3.9114
6000/6000	[=====]	- 704s	117ms/step	- loss: 3.6170
6000/6000	[=====]	- 672s	112ms/step	- loss: 3.4258
6000/6000	[=====]	- 672s	112ms/step	- loss: 3.2873
6000/6000	[=====]	- 668s	111ms/step	- loss: 3.1878
6000/6000	[=====]	- 674s	112ms/step	- loss: 3.0968
6000/6000	[=====]	- 687s	114ms/step	- loss: 3.0319
6000/6000	[=====]	- 665s	111ms/step	- loss: 2.9692
6000/6000	[=====]	- 654s	109ms/step	- loss: 2.9259
6000/6000	[=====]	- 664s	111ms/step	- loss: 2.8900
6000/6000	[=====]	- 652s	109ms/step	- loss: 2.8481
6000/6000	[=====]	- 653s	109ms/step	- loss: 2.8130
6000/6000	[=====]	- 670s	112ms/step	- loss: 2.7888

Figure 1: Loss in different epochs

3.2 Is your data pipeline completed?

1. It's ongoing but it's not completed yet. We will complete it when we will find the best models suitable for our data i.e where we will get the most accurate results.



```
def extract_feature(filename):
    model1 = VGG16()
    model1.layers.pop()
    model1 = Model(inputs=model1.inputs, outputs=model1.layers[-1].output)
    image = load_img(filename, target_size=(224, 224))
    image = img_to_array(image)
    image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
    image = preprocess_input(image)
    feature = model1.predict(image, verbose=0)
    return feature

photo = extract_feature('image.jpg')
description = generate_desc(model, tokenizer, photo, max_length)

query = description
stopwords = ['startseq', 'endseq']
querywords = query.split()
resultwords = [word for word in querywords if word.lower() not in stopwords]
result = ' '.join(resultwords)

print(result)
```

WARNING:tensorflow:5 out of the last 8962 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7a03dc2c1510> triggered tf.function retracing. Tracing is expensive and will repeatedly recompile the traced function. It is suggested to either pass the function to tf.function directly or to use tf.function.experimental_decorator.

Figure 2: Final predicted caption

4 Identification of the exact tasks you want to complete for the final submission.

4.1 What challenges you are facing and how you plan to address them?

1. Insufficient training data. So, to address this issue we are planning to use a different dataset which contains 30k images called Flickr30k.
2. We will try to implement the transformer-based models instead of CNN-LSTM models to increase the accuracy. We will try to implement BERT.

-
3. Image captioning models can overfit to the training data. We will try to implement various techniques such as dropout, regularization or early stopping to decrease the overfitting of the models.

4.2 What will be your final deliverables?

1. We will use the 30k flicker dataset for better results to tackle the insufficient training data problem.
2. We will also try to implement the transformer based models.
3. We will try to evaluate the quality of generated captions on different evaluation metrics such as CIDEr, ROUGE etc to assess the quality of captions.
4. We will also create a data pipeline.