
IMAGE CAPTION GENERATOR

PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree

Of

-BACHELOR OF TECHNOLOGY-

In

-CSAI-

By

GROUP 15

Guided by

**-Miss Shweta Singhal-
-Assistant Professor-
-CSAI and IGDTUW**



**INDIRA GANDHI DELHI TECHNICAL UNIVERSITY
FOR WOMEN
NEW DELHI – 110006**

- JULY 2021 -

CERTIFICATE

This is to certify that the project entitled “**Image Caption Generator**” is being submitted at IGDTUW, Delhi for the award of **Bachelor of Technology in CSAI** degree. It contains the record of bonafide work carried out by **Team number 15** comprising of students having enrollment numbers **02101172020, 02501172020, 03401172020 and 04301172020** under my supervision and guidance. It is further certified that the work presented here has reached the standard of B.Tech and to the best of our knowledge has not been submitted anywhere else for the award of any other degree or diploma.

Miss SHWETA SINGHAL
(Assistant Professor)
CSAI
IGDTUW

Date: 7th April, 2021 – 6th August, 2021
Place: IGDTUW

UNDERTAKING REGARDING ANTI-PLAGIARISM

We, the members of **GROUP 15** hereby, declare that the material/ content presented in the report are free from plagiarism and is properly cited and written in my own words. In case, plagiarism is detected at any stage, we shall be solely responsible for it.

Shrujal Bansal-02101172020

Sohali Baisla-02501172020

Mehak Aggarwal-03401172020

Rishika Pal-04301172020

ACKNOWLEDGEMENT

In the accomplishment of completion of our project on Image Caption Generator, We would like to convey our special gratitude to Mrs. Shweta Singhal of Computer Science and Artificial Intelligence (CSAI).

Your valuable guidance and suggestions helped us in various phases of the completion of this project. We will always be thankful to you in this regard.

Shrujal Bansal-02101172020

Sohali Baisla-02501172020

Mehak Aggarwal-03401172020

Rishika Pal-04301172020

DECLARATION

We, the members of Group 15, solemnly declare that the project report, **IMAGE CAPTION GENERATOR**, is based on our own work carried out during the course of our study under the supervision of Ms Shweta Singhal, CSAI. We assert the statements made and conclusions drawn are an outcome of our research work. We further certify that:

- I. The work contained in the report is original and has been done by us under the supervision of our supervisor.
- II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.
- III. We have followed the guidelines provided by the university in writing the report.
- IV. Whenever we have used materials (text, data, theoretical analysis/equations, codes/program, figures, tables, pictures, text etc.) from other sources, we have given due credit to them in the report and have also given their details in the references.

Shrujal Bansal-02101172020

Sohali Baisla-02501172020

Mehak Aggarwal-03401172020

Rishika Pal-04301172020

LIST OF ABBREVIATIONS

Abbreviation	Description
VGG16	Very Deep Convolutional Networks for Large-Scale Image Recognition
LSTM	Long Short-Term Memory
BLEU	Bilingual Evaluation Memory
CNN	Convolutional Neural Network

TABLE OF FIGURES

S.NO	FIGURE NAME	PAGE NO.
1.	Figure 1	12
2.	Figure 2	13
3.	Figure 3	14
4.	Figure 4	14
5.	Figure 5	16
6.	Figure 6	16
7.	Figure 7	18

ABSTRACT

In the past few years, the problem of generating descriptive sentences automatically for images has garnered a rising interest in natural language processing and computer vision research. Image captioning is a fundamental task which requires semantic understanding of images and the ability of generating description sentences with proper and correct structure. In this study, the authors propose a hybrid system employing the use of multilayer Convolutional Neural Network (CNN) to generate vocabulary describing the images and a Long Short-Term Memory (LSTM) to accurately structure meaningful sentences using the generated keywords. The convolutional neural network compares the target image to a large dataset of training images, then generates an accurate description using the trained captions.

In this project, we have presented a deep learning model to describe images and generate captions using computer vision and machine translation. This project aims to detect different objects found in an image, recognize the relationships between those objects and generate captions. The dataset used is Flickr8k and the programming language used was Python3, and an ML technique called Transfer Learning will be implemented with the help of the CNN model, to demonstrate the proposed experiment.

INDEX

Certificate	1
Undertaking regarding anti plagiarism	2
Acknowledgement	3
Declaration	4
List of Abbreviations	5
Table of Figures	6
Abstract.....	7
1. Introduction	
1.1. Scope of project.....	11
1.2. Problem statement	11
1.3. Work contribution	11
2. Other Chapters	
2.1. Evaluation.....	12
2.2. Result.....	16
3. Conclusion	17
4. Future Scope	18
5. Bibliography	19
6. Program/ Appendix	21

CHAPTER 1: INTRODUCTION

Caption generation is an interesting artificial intelligence problem where a descriptive sentence is generated for a given image. It involves the dual techniques from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. Image captioning has various applications such as recommendations in editing applications, usage in virtual assistants, for image indexing, for visually impaired persons, for social media, and several other natural language processing applications. Recently, deep learning methods have achieved state-of-the-art results on examples of this problem. It has been demonstrated that deep learning models are able to achieve optimum results in the field of caption generation problems. Instead of requiring complex data preparation or a pipeline of specifically designed models, a single end-to-end model can be defined to predict a caption, given a photo. In order to evaluate our model, we measure its performance on the Flickr8K dataset using the BLEU standard metric. These results show that our proposed model performs better than standard models regarding image captioning in performance evaluation. The limitations of neural networks are determined mostly by the amount of memory available on the GPUs used to train the network as well as the duration of training time it is allowed, we have designed our model in such a way that it uses 95% of the memory storage and can be processed within a short span of time.

We have used two strategies specially CNN and LSTM for image classification;

CNN

Convolutional Neural systems are specific important neural systems that can produce information that has an information shape, for example, a 2D lattice and CNN is valuable for working with pictures. It examines pictures from left corner to the right corner and through to extricate significant highlights from the picture, and consolidates the element to characterize pictures. It can deal with interpreted, pivoted, scaled, and modified pictures. The Convolutional neural system is a profound learning calculation that takes in the info picture, allocates significance to various components/protests in the picture, and recognizes it from each other.

LSTM

It is used to store the input data, as well as supply predictions about the subsequent datasets through its own. The LSTM network retains the stored data for a particular time period and on that basis predicts or gives the future values to the data.

LSTM is used in predicting texts. The long-term memory, understanding of LSTM makes it capable enough to predict the next words in the sentences. This is the result of the LSTM network in predicting the next words by its own. The LSTM first stores the data, the feel of the words, the styling of the words, the use of the words in a particular situation, etc and on that basis predicts the next words. The stored data, i.e. input data is further used for future use.

1.1: SCOPE OF PROJECT

With the advancement in Deep learning techniques and availability of huge dataset ,yes we can build models that can generate captions for an image. Automatically generating captions to an image shows the understanding of the image by computers, which is a fundamental task of intelligence.

1.2: PROBLEM STATEMENT

The main problem in the development of image description started with object detection using static object class libraries in the image and modelled using statistical language models.

1 Making use of CNN: It's a Deep Learning algorithm that will intake in a 2D matrix input image, assign importance (learnable weights and biases) to different aspects/objects in the image, and be intelligent enough to be able to differentiate one from the other.

2. This model was advantageous in naming the objects in an image but it could not tell us the relationship among them (that's plain image classification).

3. In this paper, we present a generative model built on a deep recurrent architecture that unites recent advances in computer vision and machine translation and that can effectively generate meaningful sentence.

4. Making use of an RNN: They are networks with loops in them, allowing information to persist. LSTMs are a particular kind of RNN, capable of learning long-term dependencies.

1.3) WORK CONTRIBUTION

- 1) SHRUJAL BANSAL- BLEU
- 2) SOHALI BAISLA- DATA PREPROCESSING (Image and Text)
- 3) MEHAK AGGARWAL- MODEL (Building, training and testing the model)
- 4) RISHIKA PAL- IMAGE PREPROCESSING

CHAPTER 2: OTHER CHAPTERS

2.1: EVALUATION

Execution of the entire program takes place in 5 major steps. The implementation of the five Major modules is as follows:

A. Data Cleaning and Pre-processing:

1. For a comfortable and fast work experience, we use Visual Studio Code: Visual Studio Code is a source-code editor made by Microsoft, it is a tool which provides free GPU/TPU processing power, over our local machines, which can take several hours to do the task that a GPU will take few minutes to do.
2. Our program starts with loading both, the text file, and the image file into separate variables.
3. This string is used and manipulated in such a way so as to create a dictionary that maps each image with a list of 5 descriptions.
4. The main task of data cleaning involves removing punctuation marks, converting the whole text to lowercase, removing stop words and removing words that contain numbers.
5. Further, a vocabulary of all unique words from all the descriptions is created, which in the future will be used to generate captions to test images.
6. Another aspect of Pre-processing the data involves tokenizing our vocabulary with a unique index value. This is because a computer won't understand regular English words, hence they need to be represented using numbers. The tokens are then stored in a pickle file. i.e. in the form of character stream, but with all the information necessary to reconstruct it into the original object type.
7. The above two pre-processing tasks can be achieved manually or by using the Keras, Pre-processing module for ease of writing the code.
8. We proceed to append the <startseq> and <endseq> identifier for each caption since these will act as indicators for our LSTM to understand where a caption is starting and where it's ending.
9. We will proceed with calculating the number of words in our vocabulary and finding the maximum length of the description, which will be used in later phases.



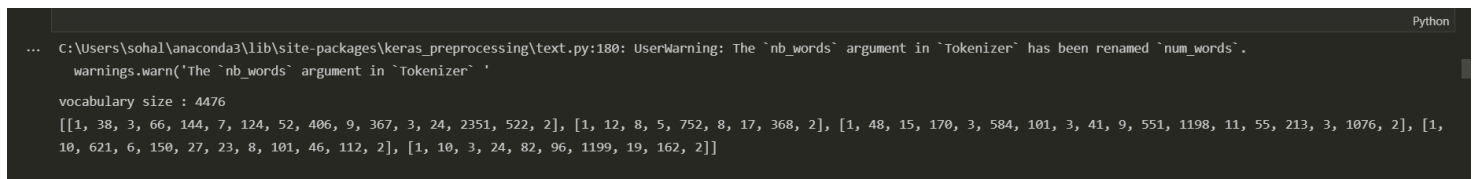
```
df_txt0[:5]
```

	filename	index	caption
0	1000268201_693b08cb0e.jpg	0	startseq child in pink dress is climbing up s...
1	1000268201_693b08cb0e.jpg	1	startseq girl going into wooden building endseq
2	1000268201_693b08cb0e.jpg	2	startseq little girl climbing into wooden pla...
3	1000268201_693b08cb0e.jpg	3	startseq little girl climbing the stairs to h...
4	1000268201_693b08cb0e.jpg	4	startseq little girl in pink dress going into...

Figure 1

B. Extraction of feature vectors:

1. A feature vector(or simply feature) is a numerical value in the matrix form, containing information about an object's important characteristics, eg. intensity value of each pixel of the image in our case. These vectors, we'll ultimately store in a pickle file.
2. In our model we'll be using Transfer Learning, which simply means, using a pretrained model (in our case the VGG16 model) to extract features from it.
3. The VGG16 model is a Convolutional Neural Network that is 16 layers deep. It is trained on the Flickr8k dataset which has 8091 images.
4. Python makes using this model in our code extremely easy with `keras.applications.VGG16` module. To use it in our code, we'll drop the classification layer from it, and hence obtain the 2048 feature vector, having a vocabulary size of 4476.
5. Hence weights will be downloaded for each image, and then image names will be mapped with their respective feature array.



```
... C:\Users\sohal\anaconda3\lib\site-packages\keras_preprocessing\txt.py:180: UserWarning: The `nb_words` argument in `Tokenizer` has been renamed `num_words`.
warnings.warn('The `nb_words` argument in `Tokenizer` '
vocabulary size : 4476
[[1, 38, 3, 66, 144, 7, 124, 52, 406, 9, 367, 3, 24, 2351, 522, 2], [1, 12, 8, 5, 752, 8, 17, 368, 2], [1, 48, 15, 170, 3, 584, 101, 3, 41, 9, 551, 1198, 11, 55, 213, 3, 1076, 2], [1, 10, 621, 6, 150, 27, 23, 8, 101, 46, 112, 2], [1, 10, 3, 24, 82, 96, 1199, 19, 162, 2]]
```

Figure 2

C. Layering the CNN-RNN model:

1. Feature Extractor: It will be used to reduce the dimensions from 2048 to 256. We'll make use of a Dropout Layer. One of these will be added in the CNN and the LSTM each. We have pre-processed the photos with the CNN model (without the output layer) and will use the extracted features predicted by this model as input.
2. Sequence Processor: This Embedding layer will handle the textual input, followed by the LSTM layer.
3. Decoder: We will merge the output from above two layers, and use a Dense layer to make the final predictions. Both the feature extractor and sequence processor output a fixed-length vector. These are merged together and processed by a Dense layer. The number of nodes in the final layer will be the same as the size of our vocabulary.

D. Training the model:

1. We'll be training our model on 6400 images each having 2048 long feature vectors.
2. Since it is not possible to hold all this data in the memory at the same time, we'll make use of a Data Generator. This will help us create batches of the data, and will improve the speed.
3. Along with this we'll be defining the number of epochs(i.e. iterations of the training dataset) the model has to complete during its training. This number has to be selected in such a way that our model is neither underfitted nor overfitted.

4. model.fit() method will be used. And this whole process will take some time depending on the processor.
5. The maximum length of descriptions calculated earlier will be used as parameter value here. It will also take as input the clean and tokenized data.
6. We'll also create a sequence creator, which will play the role of predicting the next word based on the previous word and feature vectors of the image.
7. While training our model, we can use the development dataset, to monitor the performance of the model, to decide when to save the model version to the file.

```

C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > from keras import layers
+ Code + Markdown | Run All | Clear Outputs | Restart | Interrupt | Variables ...

... 4476
Model: "model_1"

Layer (type)                 Output Shape          Param #          Connected to
-----
input_2 (InputLayer)         [(None, 30)]          0                (None, 30, 64)
embedding (Embedding)        (None, 30, 64)        286464           input_2[0][0]
CaptionFeature (LSTM)        (None, 30, 256)       328704           embedding[0][0]
dropout (Dropout)           (None, 30, 256)       0                CaptionFeature[0][0]
input_1 (InputLayer)         [(None, 1000)]        0                (None, 256)
CaptionFeature2 (LSTM)       (None, 256)           525312           dropout[0][0]
ImageFeature (Dense)         (None, 256)           256256           input_1[0][0]
add (Add)                    (None, 256)           0                CaptionFeature2[0][0]
                           ImageFeature[0][0]
dense (Dense)                (None, 256)           65792            add[0][0]
dense_1 (Dense)              (None, 4476)          1150332          dense[0][0]

Note with layer: the output here is a list without ...
Total params: 2,612,860
Trainable params: 2,612,860
Non-trainable params: 0

None

```

Figure 3

```

... C:\Users\sohal\anaconda3\lib\site-packages\keras\utils\generic_utils.py:494: CustomMaskWarning: Custom mask layers require a config and must override get_config. When loading, the
custom mask layer must be passed to the custom_objects argument.
warnings.warn('Custom mask layers require a config and must override '

Epoch 1/6
1551/1551 - 533s - loss: 5.3539 - val_loss: 4.8180
Epoch 2/6
1551/1551 - 446s - loss: 4.5164 - val_loss: 4.5363
Epoch 3/6
1551/1551 - 457s - loss: 4.1198 - val_loss: 4.4681
Epoch 4/6
1551/1551 - 476s - loss: 3.8487 - val_loss: 4.4542
Epoch 5/6
1551/1551 - 536s - loss: 3.6249 - val_loss: 4.4700
Epoch 6/6
1551/1551 - 694s - loss: 3.4247 - val_loss: 4.5524

```

Figure 4

E. Testing the model:

1. The same python notebook is used to perform testing. We loaded the trained model that we had saved in the previous step and generate predictions.
2. The sequence generator will come into play at this stage, besides the tokenizer file we created.
3. The primary step of feature extraction for the particular image under observation will be performed.
4. The path of one of the images from the remaining 1600 test images is passed to the function manually. You can also iterate through the test data set, and store the prediction for each image in a dictionary or a list.
5. The actual functioning behind image generation involves using the start sequence and the end

2.2: RESULT

The image captioning model was implemented and we were able to generate moderately comparable captions with compared to human generated captions. The VGG net model first assigns probabilities to all the objects that are possibly present in the image. The model converts the image into word vector. This word vector is provided as input to LSTM cells which will then form sentence from this word vector.

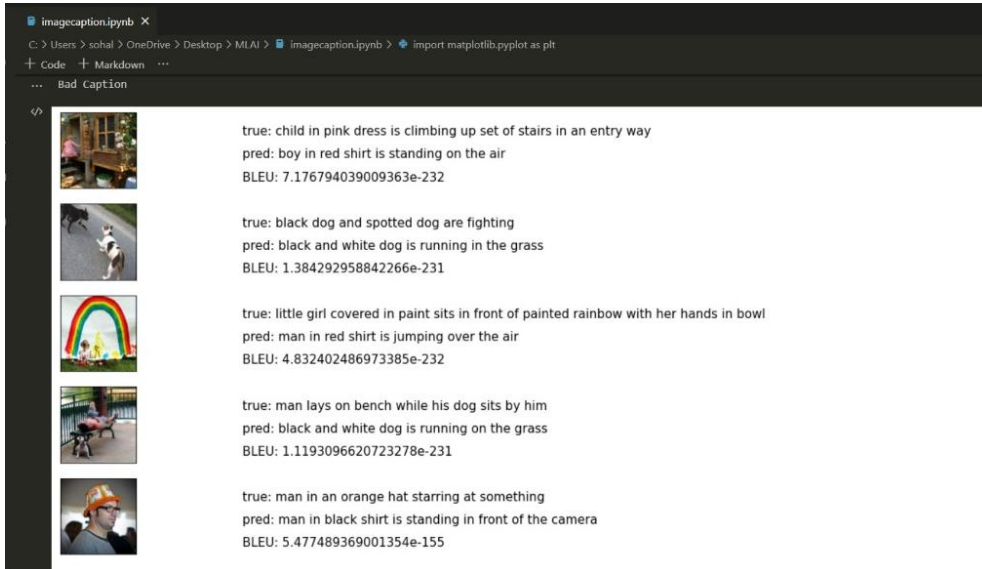


Figure 5

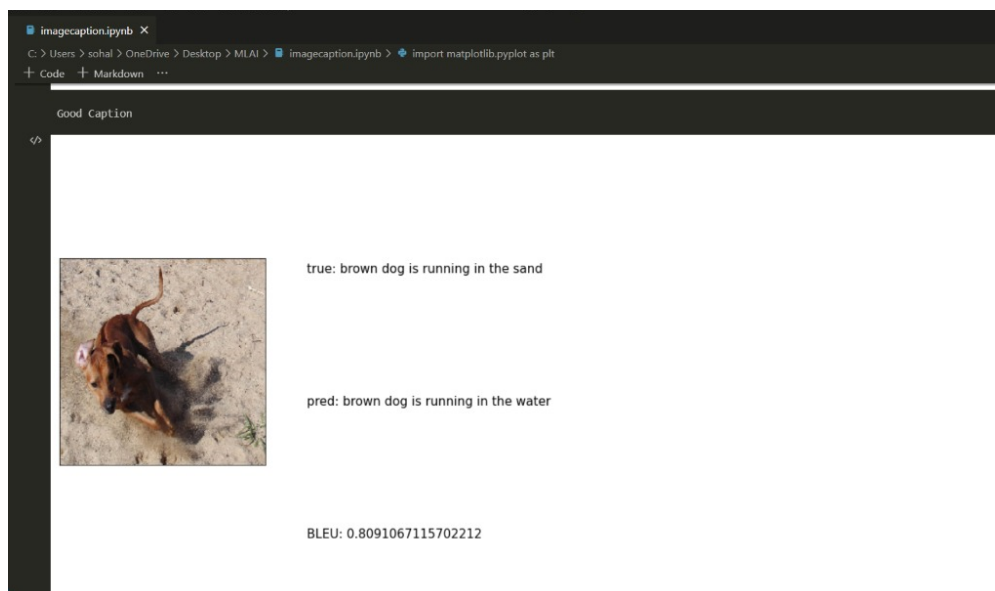


Figure 6

CHAPTER 3: CONCLUSION

Based on the results obtained we can see that the deep learning methodology used here bore successful results. The CNN and the LSTM worked together in proper synchronization, they were able to find the relation between objects in images.

In this advanced Python project, we have implemented a CNN-RNN model by building an image caption generator. Some key points to note are that our model depends on the data, so, it cannot predict the words that are out of its vocabulary. We used a small dataset consisting of 8000 images. For production-level models, we need to train on datasets larger than 100,000 images which can produce better accuracy models.

To compare the accuracy of the predicted caption, we can compare them with target captions in our Flickr8k test dataset, using BLEU(Bilingual Evaluation Understudy) score. BLEU scores are used in text translation for evaluating translated text against one or more reference translations. Over the years several other neural network technologies have been used to create hybrid image caption generators, similar to the one proposed here.

The model has been successfully trained to generate the captions as expected for the images. The caption generation has constantly been improved by fine tuning the model with different hyper parameter. Higher BLEU score indicates that the generated captions are very similar to those of the actual caption present on the images. Below you will find a table displaying different BLEU scores obtained by tuning the parameters.

CHAPTER 4: FUTURE SCOPE

The limitation faced by this image caption generator is that as the validation loss curve is greater than the training loss curve thus, the model is overfitting. It can be avoided by training a larger dataset therefore, getting a more precise result.

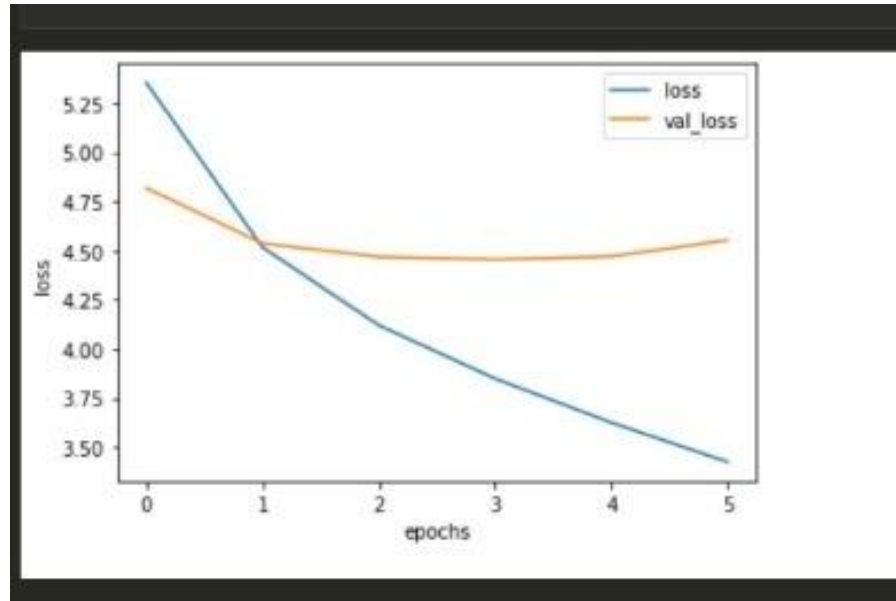


Figure 7

BIBLIOGRAPHY

- <https://towardsdatascience.com/image-captions-with-attention-in-tensorflow-step-by-step-927dad3569fa>
- <https://www.math.ucla.edu/~minchen/doc/ImgCapGen.pdf>
- https://data-flair.training/blogs/python-based-project-image-caption-generator-cnn/#google_vignette
- <https://medium.com/swlh/automatic-image-captioning-using-deep-learning-5e899c127387>
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Conference on Learning Representations (ICLR)
- https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Vinyals_Show_and_Tel_2015_CVPR_paper.html
- <https://arxiv.org/pdf/1411.4555.pdf>
- <https://arxiv.org/pdf/1711.09151.pdf>
- <https://ieeexplore.ieee.org/abstract/document/8272660>
- Shuang Bai and Shan An. 2018. A Survey on Automatic Image Caption Generation. Neurocomputing. ACM Computing Surveys, Vol. 0, No. 0, Article 0. Acceptance Date: October 2018. 0:30 Hossain et al.
- Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, Vol. 29. 65–72.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, [Online] Available: <https://papers.nips.cc/paper/4824-imagenetclassificationwith-deep-convolutionalneural-networks.pdf>

- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database Andrej Karpathy, Li Fei-Fei, Deep VisualSemantic Alignments for Generating Image Descriptions,
Available: <https://cs.stanford.edu/people/karpathy/cvpr2015.pdf>
- BLEU: a Method for Automatic Evaluation of Machine Translation Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu IBM T. J. Watson Research Center Yorktown Heights, NY 10598, USA
<https://arxiv.org/pdf/1711.09151.pdf>
- Neural Image Caption Generation with Visual Attention,
Available: <https://arxiv.org/pdf/1502.03044.pdf>
- Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics”, Journal of Artificial Intelligence Research, Volume 47, pages 853-899
https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Vinyals_Show_and_Tel_1_2015_CVPR_paper.html
- https://ijcrt.org/papers/IJCRT_196552.pdf

APPENDIX/PROGRAM

```
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > config1 = tf.compat.v1.ConfigProto()
+ Code + Markdown | ▶ Run All ☰ Clear Outputs ⌂ Restart ☐ Interrupt | 📄 Variables ...
```

```
import matplotlib.pyplot as plt
import tensorflow as tf
import keras
import sys, time, os, warnings
import numpy as np
import pandas as pd
from collections import Counter
```

```
config1 = tf.compat.v1.ConfigProto()
config1.gpu_options.per_process_gpu_memory_fraction = 0.95
config1.gpu_options.visible_device_list = "0"
tf.compat.v1.Session(config=config1)
```

```
... <tensorflow.python.client.session.Session at 0x245acb838b0>
```

```
def set_seed(sd=123):
    from numpy.random import seed
    from tensorflow import set_random_seed
    import random as rn
    seed(sd)
    rn.seed(sd)
    set_random_seed(sd)
```

```
image_dataset = 'D:\python-project-image-caption-generator\Flickr8k_Dataset\Flickr8k_Dataset'
text_dataset = 'D:\python-project-image-caption-generator\Flickr8k_text\Flickr8k.token.txt'
```

```

imagecaption.ipynb
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > imgs = os.listdir(image_dataset)
+ Code + Markdown | Run All Clear Outputs Restart Interrupt Variables ...
>
imgs = os.listdir(image_dataset)
print("The number of jpg files in Flicker8k: {}".format(len(imgs)))

... The number of jpg files in Flicker8k: 8091

file = open(text_dataset, 'r', encoding='utf8')
text = file.read()
file.close()
datatext = []
for line in text.split('\n'):
    column = line.split('\t')
    if len(column) == 1:
        continue
    w = column[0].split("#")
    datatext.append(w + [column[1].lower()])
df_txt = pd.DataFrame(datatext, columns=["filename", "index", "caption"])
uni_filenames = np.unique(df_txt.filename.values)
print("The number of file names: {}".format(len(uni_filenames)))
print("The distribution of captions for each image: ")
Counter(Counter(df_txt.filename.values).values())
print(df_txt[:5])

... The number of file names: 8092
The distribution of captions for each image:
      filename index \
0  1000268201_693b08cb0e.jpg  0
1  1000268201_693b08cb0e.jpg  1
2  1000268201_693b08cb0e.jpg  2
3  1000268201_693b08cb0e.jpg  3
4  1000268201_693b08cb0e.jpg  4

```

```

imagecaption.ipynb
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > file = open(text_dataset, 'r', encoding='utf8')
+ Code + Markdown | Run All Clear Outputs Restart Interrupt Variables ...
>
1000268201_693b08cb0e.jpg
4  1000268201_693b08cb0e.jpg  4

caption
0  a child in a pink dress is climbing up a set o...
1           a girl going into a wooden building .
2  a little girl climbing into a wooden playhouse .
3  a little girl climbing the stairs to her playh...
4  a little girl in a pink dress going into a woo...

```

Indira Gandhi Delhi Technical University for Women

```
imagecaption.ipynb •
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > file = open(text_dataset,'r', encoding='ut
+ Code + Markdown | ▶ Run All ≡ Clear Outputs ↻ Restart □ Interrupt | 📄 Variables ...

▶
from keras.preprocessing.image import load_img, img_to_array
from IPython.display import display
from PIL import Image

npic1 = 5
npic2 = 224
target_size = (npic2,npic2,3)

count = 1
fig = plt.figure(figsize=(10,20))
for jpg in uni_filenames[-5:]:
    filename = image_dataset + '/' + jpg
    captions = list(df_txt["caption"].loc[df_txt["filename"]==jpg].values)
    image_load = load_img(filename, target_size=target_size)




    a = fig.add_subplot(npic1,2,count,xticks=[],yticks=[])
    a.imshow(image_load)
    count += 1

    a = fig.add_subplot(npic1,2,count)
    plt.axis('off')
    a.plot()
    a.set_xlim(0,1)
    a.set_ylim(0,len(captions))
    for i, caption in enumerate(captions):
        a.text(0,i,caption,fontsize=20)
    count += 1
plt.show()
```

imagecaption.ipynb • imagecaption.ipynb - Visual Studio Code

C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > file = open(text_dataset,'r', encoding='utf8')

+ Code + Markdown | ▶ Run All ≡ Clear Outputs ↻ Restart □ Interrupt | 📄 Variables ...

	<p>man on a bicycle riding on only one wheel .</p> <p>asian man in orange hat is popping a wheelie on his bike .</p> <p>a man on a bicycle is on only the back wheel .</p> <p>a man is doing a wheelie on a mountain bike .</p> <p>a man does a wheelie on his bicycle on the sidewalk .</p>
	<p>five people are sitting together in the snow .</p> <p>five children getting ready to sled .</p> <p>a group of people sit in the snow overlooking a mountain scene .</p> <p>a group of people sit atop a snowy mountain .</p> <p>a group is sitting around a snowy crevasse .</p>
	<p>a white crane stands tall as it looks out upon the ocean .</p> <p>a water bird standing at the ocean 's edge .</p> <p>a tall bird is standing on the sand beside the ocean .</p> <p>a large bird stands in the water on the beach .</p> <p>a grey bird stands majestically on a beach while waves roll in .</p>


```
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > def df_word(df_txt):
+ Code + Markdown | Run All Clear Outputs Restart Interrupt Variables ...
def df_word(df_txt):
    vocabulary = []
    for txt in df_txt.caption.values:
        vocabulary.extend(txt.split())
    print('Vocabulary Size: %d' % len(set(vocabulary)))
    ct = Counter(vocabulary)
    dfword = pd.DataFrame({"word":List(ct.keys()),"count":List(ct.values())})
    dfword = dfword.sort_values("count",ascending=False)
    dfword = dfword.reset_index()[["word","count"]]
    return(dfword)
dfword = df_word(df_txt)
dfword.head(3)

... Vocabulary Size: 8918

</>
  word  count
0    a  62989
1    .  36581
2   in  18975

import string
text_original = "I ate 1000 marshmallows and a pie. What's the time?"

print(text_original)
print("\nRemove punctuations..")
def remove_punctuation(text_original):
    text_no_punctuation = text_original.translate(str.maketrans('', '', string.punctuation))
    return(text_no_punctuation)
text_no_punctuation = remove_punctuation(text_original)
print(text_no_punctuation)
```

```
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > def df_word(df_txt):
+ Code + Markdown | Run All Clear Outputs Restart Interrupt Variables ...

print("\nRemove a single character word..")
def remove_single_character(text):
    text_len_more_than1 = ""
    for word in text.split():
        if len(word) > 1:
            text_len_more_than1 += " " + word
    return(text_len_more_than1)
text_len_more_than1 = remove_single_character(text_no_punctuation)
print(text_len_more_than1)

print("\nRemove words with numeric values..")
def remove_numeric(text,printTF=False):
    text_no_numeric = ""
    for word in text.split():
        isalpha = word.isalpha()
        if printTF:
            print("  {:10} : {}".format(word,isalpha))
        if isalpha:
            text_no_numeric += " " + word
    return(text_no_numeric)
text_no_numeric = remove_numeric(text_len_more_than1,printTF=True)
print(text_no_numeric)
```

```
... I ate 1000 marshmallows and a pie. What's the time?
```

```
Remove punctuations..
```

```
I ate 1000 marshmallows and a pie Whats the time
```

```
Remove a single character word..
```

```
ate 1000 marshmallows and pie Whats the time
```

```
Remove words with numeric values..
```

```
ate      : True
1000     : False
marshmallows : True
and      : True
pie      : True
Whats    : True
the      : True
time     : True
```

```
ate marshmallows and pie Whats the time
```

```
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > def df_word(df_txt):
+ Code + Markdown | ▶ Run All | Clear Outputs | Restart | Interrupt | Variables ...

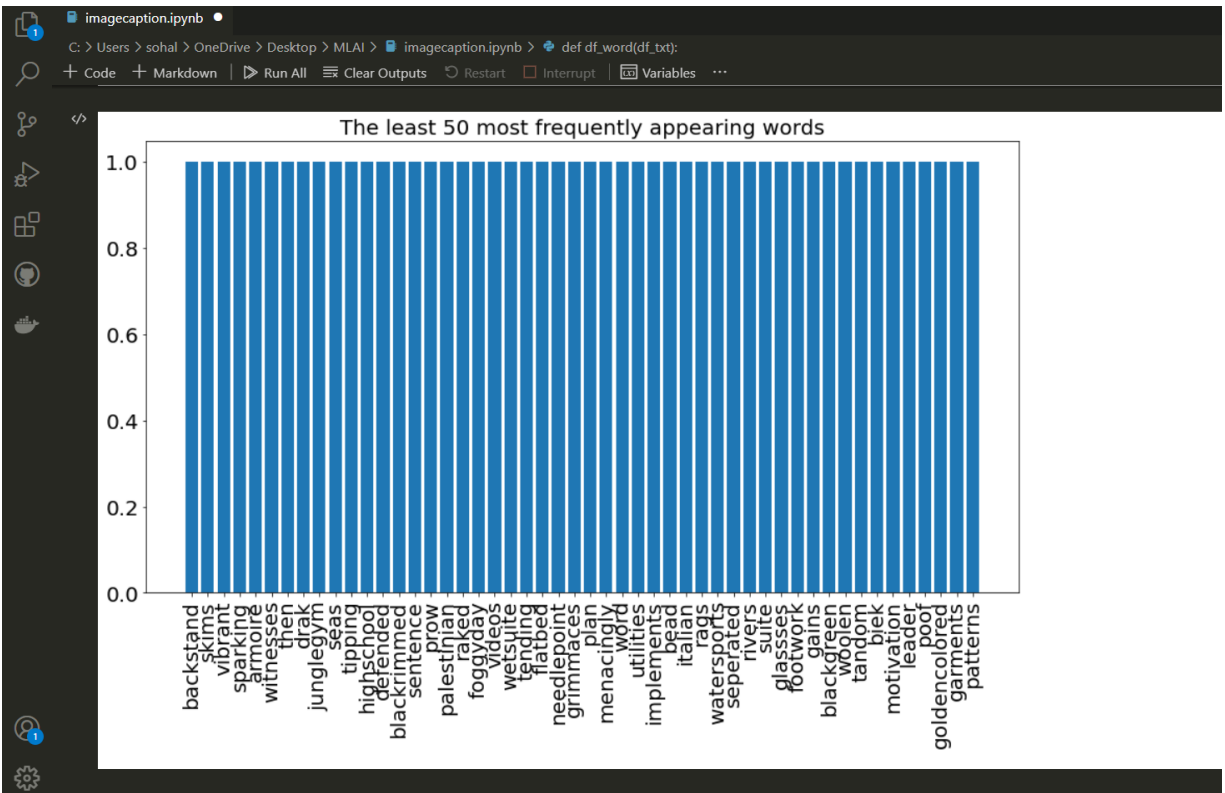
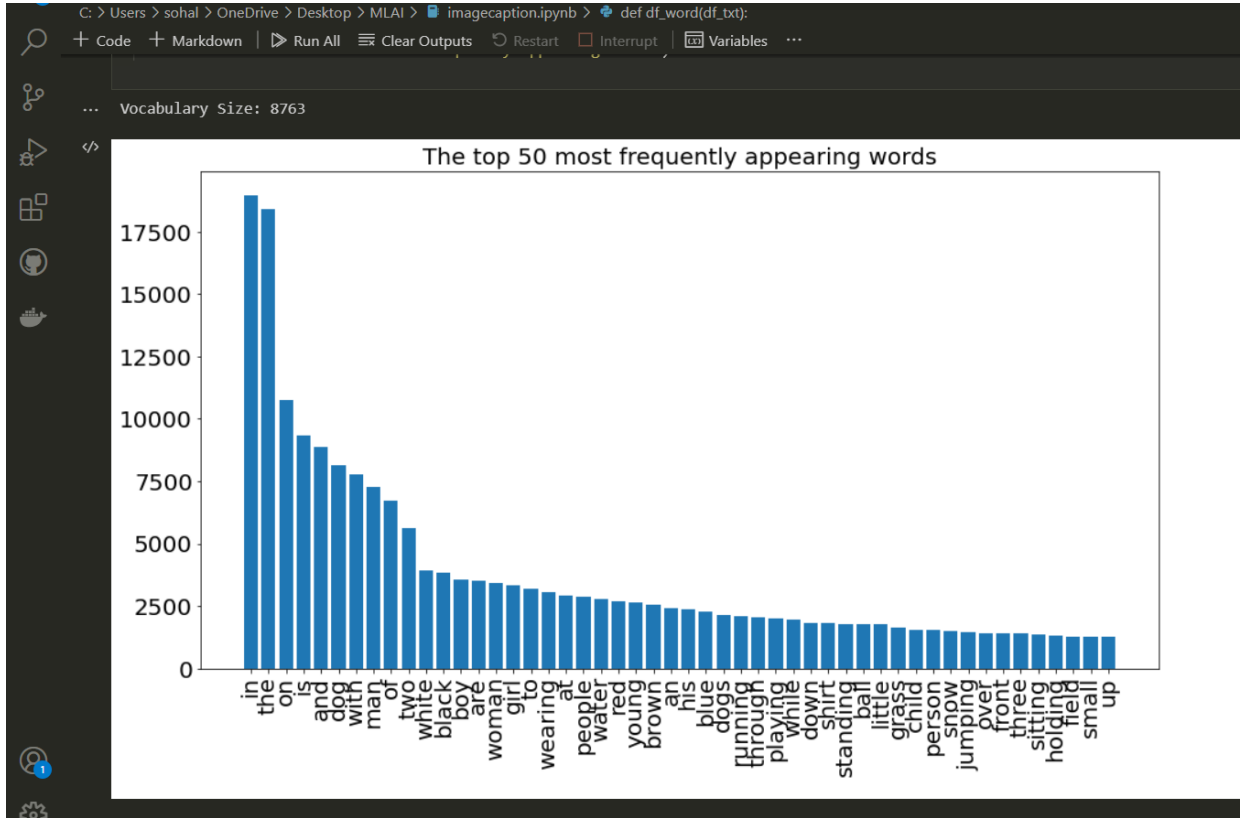
def text_clean(text_original):
    text = remove_punctuation(text_original)
    text = remove_single_character(text)
    text = remove_numeric(text)
    return(text)

for i, caption in enumerate(df_txt.caption.values):
    newcaption = text_clean(caption)
    df_txt["caption"].iloc[i] = newcaption

top = 50

def plthist(dfsub, title="The top 50 most frequently appearing words"):
    plt.figure(figsize=(15,8))
    plt.bar(dfsub.index,dfsub["count"])
    plt.yticks(fontsize=20)
    plt.xticks(dfsub.index,dfsub["word"],rotation=90,fontsize=20)
    plt.title(title,fontsize=20)
    plt.show()

dfword = df_word(df_txt)
plthist(dfword.iloc[:top,:],
        title="The top 50 most frequently appearing words")
plthist(dfword.iloc[-top:,:],
        title="The least 50 most frequently appearing words")
```



Indira Gandhi Delhi Technical University for Women

```
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > def df_word(df_txt):
+ Code + Markdown | Run All | Clear Outputs | Restart | Interrupt | Variables ...

from copy import copy
def add_start_end_seq_token(captions):
    caps = []
    for txt in captions:
        txt = 'startseq ' + txt + ' endseq'
        caps.append(txt)
    return(caps)
df_txt0 = copy(df_txt)
df_txt0["caption"] = add_start_end_seq_token(df_txt["caption"])
df_txt0.head(5)
del df_txt

df_txt0[:5]
```

	filename	index	caption
0	1000268201_693b08cb0e.jpg	0	startseq child in pink dress is climbing up s...
1	1000268201_693b08cb0e.jpg	1	startseq girl going into wooden building endseq
2	1000268201_693b08cb0e.jpg	2	startseq little girl climbing into wooden pla...
3	1000268201_693b08cb0e.jpg	3	startseq little girl climbing the stairs to h...
4	1000268201_693b08cb0e.jpg	4	startseq little girl in pink dress going into...

```
imagecaption.ipynb
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > def df_word(df_txt):
+ Code + Markdown | Run All | Clear Outputs | Restart | Interrupt | Variables ...

from tensorflow.keras.applications import VGG16

modelvgg = VGG16(include_top=True,weights=None)
## load the locally saved weights
modelvgg.load_weights('F:/vgg16_weights_tf_dim_ordering_tf_kernels.h5')
modelvgg.summary()
```

Model: "vgg16"

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

Indira Gandhi Delhi Technical University for Women

```
imagecaption.ipynb
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > def df_word(df_txt):
+ Code + Markdown | Run All Clear Outputs Restart Interrupt Variables ...

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

show more (open the raw output data in a text editor) ...

=====
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0

-----

from keras import models
modelvgg.layers.pop()
modelvgg = models.Model(inputs=modelvgg.inputs, outputs=modelvgg.layers[-1].output)
modelvgg.summary()
```

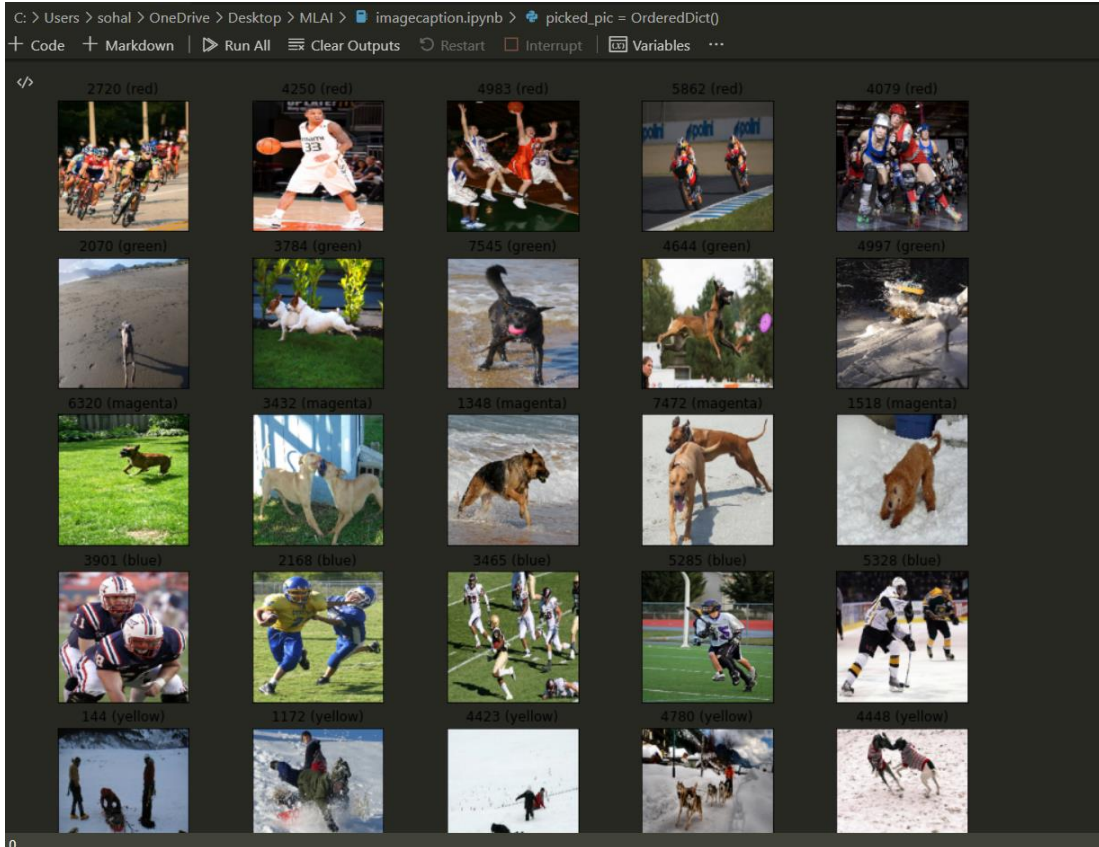
```
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > from keras.preprocessing.image import load_i
+ Code + Markdown | Run All Clear Outputs Restart Interrupt Variables ...

from keras.preprocessing.image import load_img, img_to_array
from keras.applications.vgg16 import preprocess_input
from collections import OrderedDict
images = OrderedDict()
npix = 224
target_size = (npix,npix,3)
data = np.zeros((len(imgs),npix,npix,3), dtype='uint8')
for i,name in enumerate(imgs):
    filename = image_dataset + '/' + name
    image = load_img(filename, target_size=target_size)
    image = img_to_array(image)
    nimage = preprocess_input(image)

    y_pred = modelvgg.predict(nimage.reshape( (1,) + nimage.shape[:3]))
    images[name] = y_pred.flatten()

from sklearn.decomposition import PCA
encoder = np.array(list(images.values()))
pca = PCA(n_components=2)
y_pca = pca.fit_transform(encoder)

picked_pic = OrderedDict()
picked_pic["red"] = [2720,4250,4983,5862,4079]
picked_pic["green"] = [2070,3784,7545,4644, 4997]
picked_pic["magenta"] = [6320,3432,1348,7472, 1518]
picked_pic["blue"] = [3901,2168,3465,5285,5328]
picked_pic["yellow"] = [144,1172,4423,4780,4448]
picked_pic["purple"] = [5087]
```

```
C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > from keras.preprocessing.text import Tokenizer
+ Code + Markdown | Run All Clear Outputs Restart Interrupt Variables ...
```

```

dimages, keepindex = [], []
df_txt0 = df_txt0.loc[df_txt0["index"].values == "0", :]
for i, fnm in enumerate(df_txt0.filename):
    if fnm in images.keys():
        dimages.append(images[fnm])
        keepindex.append(i)
fnames = df_txt0["filename"].iloc[keepindex].values
dcaptions = df_txt0["caption"].iloc[keepindex].values
dimages = np.array(dimages)

```

```
df_txt0[:5]
```

	filename	index	caption
0	1000268201_693b08cb0e.jpg	0	startseq child in pink dress is climbing up s...
5	1001773457_577c3a7d70.jpg	0	startseq black dog and spotted dog are fighti...
10	1002674143_1b742ab4b8.jpg	0	startseq little girl covered in paint sits in...
15	1003163366_44323f5815.jpg	0	startseq man lays on bench while his dog sits...
20	1007129816_e794419615.jpg	0	startseq man in an orange hat starring at som...

```

+ Code + Markdown ▶ Run All ☰ Clear Outputs ⌂ Restart ☐ Interrupt 📄 Variables ... Select Kernel
▶
from keras.preprocessing.text import Tokenizer
nb_words = 6000
tokenizer = Tokenizer(nb_words=nb_words)
tokenizer.fit_on_texts(dcaptions)
vocab_size = len(tokenizer.word_index) + 1
print("vocabulary size : {}".format(vocab_size))
dtexts = tokenizer.texts_to_sequences(dcaptions)
print(dtexts[:5])

... C:\Users\sohal\anaconda3\lib\site-packages\keras_preprocessing\text.py:180: UserWarning: The `nb_words` argument in `Tokenizer` has been renamed `num_words`.
  warnings.warn('The `nb_words` argument in `Tokenizer` '

vocabulary size : 4476
[[1, 38, 3, 66, 144, 7, 124, 52, 406, 9, 367, 3, 24, 2351, 522, 2], [1, 12, 8, 5, 752, 8, 17, 368, 2], [1, 48, 15, 170, 3, 584, 101, 3, 41, 9, 551, 1198, 11, 55, 213, 3, 1076, 2], [1,
10, 621, 6, 150, 27, 23, 8, 101, 46, 112, 2], [1, 10, 3, 24, 82, 96, 1199, 19, 162, 2]]

prop_test, prop_val = 0.2, 0.2
N = len(dtexts)
Ntest, Nval = int(N*prop_test), int(N*prop_val)
def split_test_val_train(dtexts, Ntest, Nval):
    return (dtexts[:Ntest],
            dtexts[Ntest:Ntest+Nval],
            dtexts[Ntest+Nval:])
dt_test, dt_val, dt_train = split_test_val_train(dtexts, Ntest, Nval)
di_test, di_val, di_train = split_test_val_train(dimages, Ntest, Nval)
fnn_test, fnn_val, fnn_train = split_test_val_train(fnames, Ntest, Nval)
    
```

```

C:\Users\sohal> sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > maxlen = np.max([len(text) for text in dtexts])
+ Code + Markdown ▶ Run All ☰ Clear Outputs ⌂ Restart ☐ Interrupt 📄 Variables ...
▶
maxlen = np.max([len(text) for text in dtexts])
print(maxlen)

... 30

from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
def preprocessing(dtexts, dimages):
    N = len(dtexts)
    print("# captions/images = {}".format(N))
    assert(N==len(dimages))
    Xtext, Ximage, ytext = [], [], []
    for text, image in zip(dtexts, dimages):
        for i in range(1, len(text)):
            in_text, out_text = text[:i], text[i]
            in_text = pad_sequences([in_text], maxlen=maxlen).flatten()
            out_text = to_categorical(out_text, num_classes = vocab_size)
            Xtext.append(in_text)
            Ximage.append(image)
            ytext.append(out_text)
    Xtext = np.array(Xtext)
    Ximage = np.array(Ximage)
    ytext = np.array(ytext)
    print("{} {} {}".format(Xtext.shape, Ximage.shape, ytext.shape))
    return(Xtext, Ximage, ytext)
Xtext_train, Ximage_train, ytext_train = preprocessing(dt_train, di_train)
Xtext_val, Ximage_val, ytext_val = preprocessing(dt_val, di_val)

... # captions/images = 4855
(49631, 30) (49631, 1000) (49631, 4476)
# captions/images = 1618
(16353, 30) (16353, 1000) (16353, 4476)
    
```



```

C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > maxlen = np.max([len(text) for text in dtexts])
+ Code + Markdown | Run All Clear Outputs Restart Interrupt Variables ...

from keras import layers
from keras.layers import Input, Flatten, Dropout, Activation
from keras.layers.advanced_activations import LeakyReLU, PReLU
print(vocab_size)
## image feature

dim_embedding = 64

input_image = layers.Input(shape=(Ximage_train.shape[1],))
fimage = layers.Dense(256,activation='relu',name="ImageFeature")(input_image)
## sequence model
input_txt = layers.Input(shape=(maxlen,))
ftxt = layers.Embedding(vocab_size,dim_embedding, mask_zero=True)(input_txt)
ftxt = layers.LSTM(256,name="CaptionFeature",return_sequences=True)(ftxt)
#,return_sequences=True
#,activation='relu'
se2 = Dropout(0.04)(ftxt)
ftxt = layers.LSTM(256,name="CaptionFeature2")(se2)
## combined model for decoder
decoder = layers.add([ftxt,fimage])
decoder = layers.Dense(256,activation='relu')(decoder)
output = layers.Dense(vocab_size,activation='softmax')(decoder)
model = models.Model(inputs=[input_image, input_txt],outputs=output)

model.compile(loss='categorical_crossentropy', optimizer='adam')

print(model.summary())
... 4476

```

```

C: > Users > sohal > OneDrive > Desktop > MLAI > imagecaption.ipynb > from keras import layers
+ Code + Markdown | Run All Clear Outputs Restart Interrupt Variables ...

... 4476
Model: "model_1"

Layer (type)                 Output Shape         Param #         Connected to
-----
input_2 (InputLayer)         [(None, 30)]        0               input_2[0][0]
embedding (Embedding)        (None, 30, 64)      286464          input_2[0][0]
CaptionFeature (LSTM)         (None, 30, 256)     328704          embedding[0][0]
dropout (Dropout)            (None, 30, 256)     0               CaptionFeature[0][0]
input_1 (InputLayer)         [(None, 1000)]      0               input_1[0][0]
CaptionFeature2 (LSTM)        (None, 256)         525312          dropout[0][0]
ImageFeature (Dense)          (None, 256)         256256          input_1[0][0]
add (Add)                     (None, 256)         0               CaptionFeature2[0][0]
                             ImageFeature[0][0]
dense (Dense)                 (None, 256)         65792           add[0][0]
dense_1 (Dense)               (None, 4476)        1150332         dense[0][0]

show more (open the raw output data in a text editor) ...

Total params: 2,612,860
Trainable params: 2,612,860
Non-trainable params: 0

None

```

```
from time import time
from keras.callbacks import TensorBoard
tensorboard = TensorBoard(log_dir="logs/{}".format(time()))
hist = model.fit([Ximage_train, Xtext_train], ytext_train, epochs=6, verbose=2, batch_size=32, validation_data=([Ximage_val, Xtext_val], ytext_val), callbacks=[tensorboard])
```

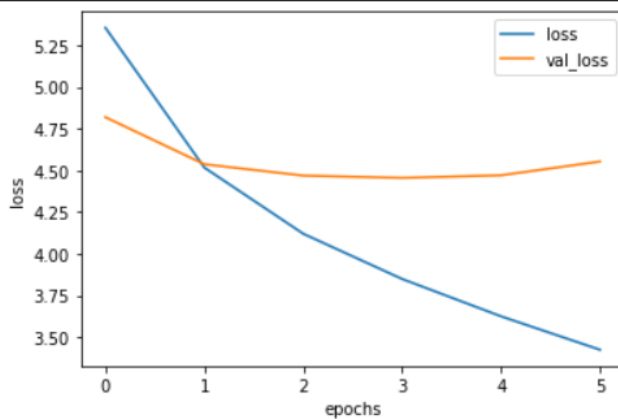
Python

... C:\Users\sohal\anaconda3\lib\site-packages\keras\utils\generic_utils.py:494: CustomMaskWarning: Custom mask layers require a config and must override get_config. When loading, the custom mask layer must be passed to the custom_objects argument.

warnings.warn('Custom mask layers require a config and must override ')

```
Epoch 1/6
1551/1551 - 533s - loss: 5.3539 - val_loss: 4.8180
Epoch 2/6
1551/1551 - 446s - loss: 4.5164 - val_loss: 4.5363
Epoch 3/6
1551/1551 - 457s - loss: 4.1198 - val_loss: 4.4681
Epoch 4/6
1551/1551 - 476s - loss: 3.8487 - val_loss: 4.4542
Epoch 5/6
1551/1551 - 536s - loss: 3.6249 - val_loss: 4.4700
Epoch 6/6
1551/1551 - 694s - loss: 3.4247 - val_loss: 4.5524
```

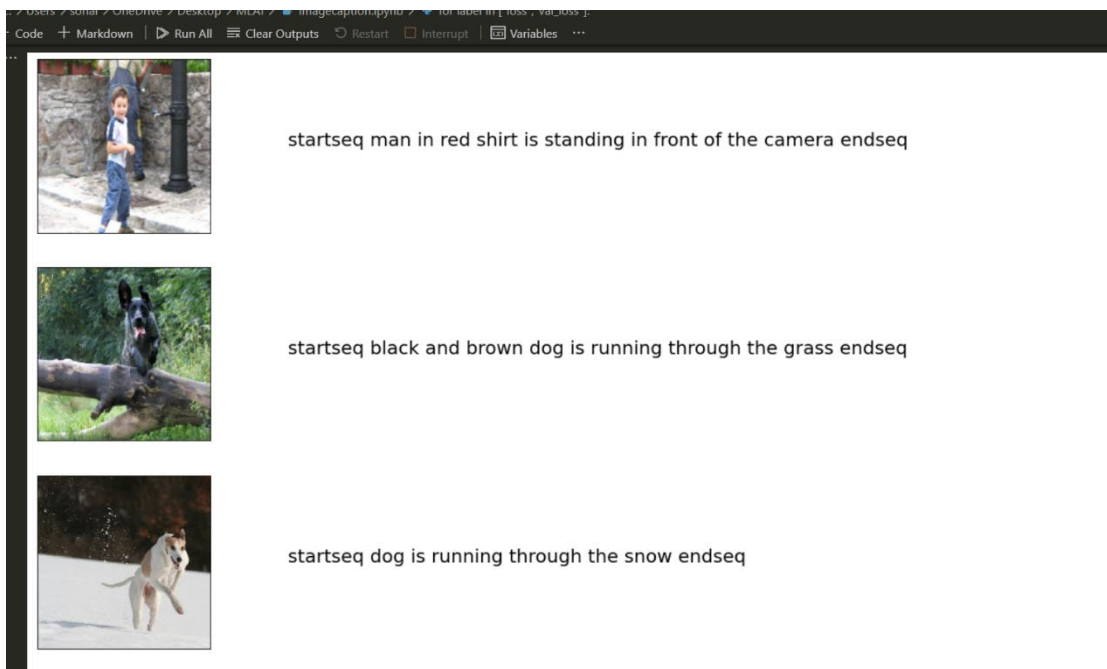
```
for label in ["loss", "val_loss"]:
    plt.plot(hist.history[label], label=label)
plt.legend()
plt.xlabel("epochs")
plt.ylabel("loss")
plt.show()
```



```

+ Code + Markdown | ▶ Run All | ☰ Clear Outputs | ↺ Restart | ☐ Interrupt | 📄 Variables | ...
▶
index_word = dict([(index,word) for word, index in tokenizer.word_index.items()])
def predict_caption(image):
    in_text = 'startseq'
    for iword in range(maxlen):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence],maxlen)
        yhat = model.predict([image,sequence],verbose=0)
        yhat = np.argmax(yhat)
        newword = index_word[yhat]
        in_text += " " + newword
        if newword == "endseq":
            break
    return(in_text)
npic1 = 5
npic2 = 224
target_size = (npic2,npic2,3)
count = 1
fig = plt.figure(figsize=(10,20))
for jpg, image_feature in zip(fnm_test[8:13],di_test[8:13]):
    ## images
    filename = image_dataset + '/' + jpg
    image_load = load_img(filename, target_size=target_size)
    a = fig.add_subplot(npic1,2,count,xticks=[],yticks=[])
    a.imshow(image_load)
    count += 1
    caption = predict_caption(image_feature.reshape(1,len(image_feature)))
    a = fig.add_subplot(npic1,2,count)
    plt.axis('off')
    a.plot()
    a.set_xlim(0,1)
    a.set_ylim(0,1)
    a.text(0,0.5,caption,fontsize=20)
    count += 1
plt.show()

```



```

+ Code + Markdown | ▶ Run All | ☰ Clear Outputs | ⌂ Restart | ☐ Interrupt | 📄 Variables | ⋮
▶
hypothesis = "I like dog"
hypothesis = hypothesis.split()
reference = "I do like dog"
references = [reference.split()]

from nltk.translate.bleu_score import sentence_bleu
print("BLEU={:4.3f}".format(sentence_bleu(references,hypothesis)))

... BLEU=0.000

C:\Users\sohal\anaconda3\lib\site-packages\nltk\translate\bleu_score.py:516: UserWarning:
The hypothesis contains 0 counts of 3-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
C:\Users\sohal\anaconda3\lib\site-packages\nltk\translate\bleu_score.py:516: UserWarning:
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)

```

```

Code + Markdown | ▶ Run All | ☰ Clear Outputs | ⌂ Restart | ☐ Interrupt | 📄 Variables | ⋮
▶
index_word = dict([(index,word) for word, index in tokenizer.word_index.items()])
nkeep = 5
pred_good, pred_bad, bleus = [], [], []
count = 0
for jpg, image_feature, tokenized_text in zip(fnm_test,di_test,dt_test):
    count += 1
    if count % 200 == 0:
        print("  {:4.2f}% is done..".format(100*count/float(len(fnm_test))))
    caption_true = [ index_word[i] for i in tokenized_text ]
    caption_true = caption_true[1:-1]
    caption = predict_caption(image_feature.reshape(1,len(image_feature)))
    caption = caption.split()
    caption = caption[1:-1]
    bleu = sentence_bleu([caption_true],caption)
    bleus.append(bleu)
    if bleu > 0.7 and len(pred_good) < nkeep:
        pred_good.append((bleu,jpg,caption_true,caption))
    elif bleu < 0.3 and len(pred_bad) < nkeep:
        pred_bad.append((bleu,jpg,caption_true,caption))

12.36% is done..
24.72% is done..
37.08% is done..
49.44% is done..
61.80% is done..
74.17% is done..
86.53% is done..
98.89% is done..

```

```
Code | Markdown | Run All | Clear Outputs | Restart | Interrupt | Variables
```

```
print("Mean BLEU {:.3f}".format(np.mean(bleus)))
```

... Mean BLEU 0.013

```
def plot_images(pred_bad):
    def create_str(caption_true):
        strue = ""
        for s in caption_true:
            strue += " " + s
        return(strue)
    npix = 224
    target_size = (npix,npix,3)
    count = 1
    fig = plt.figure(figsize=(10,10))
    npic = len(pred_bad)
    for pb in pred_bad:
        bleu,jpgfnm,caption_true,caption = pb
        ## images
        filename = image_dataset + '/' + jpgfnm
        image_load = load_img(filename, target_size=target_size)
        ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
        ax.imshow(image_load)
        count += 1

        caption_true = create_str(caption_true)
        caption = create_str(caption)

        ax = fig.add_subplot(npic,2,count)
        plt.axis('off')
        ax.plot()
        ax.set_xlim(0,1)
        ax.set_ylim(0,1)
        ax.text(0,0.7,"true:" + caption_true,fontsize=15)
```

```
Code | Markdown | Run All | Clear Outputs | Restart | Interrupt | Variables
```

```
ax.set_xlim(0,1)
ax.set_ylim(0,1)
ax.text(0,0.7,"true:" + caption_true,fontsize=15)
ax.text(0,0.4,"pred:" + caption,fontsize=15)
ax.text(0,0.1,"BLEU: {}".format(bleu),fontsize=15)
count += 1
plt.show()

def plot_images(pred_good):
    def create_str(caption_true):
        strue = ""
        for s in caption_true:
            strue += " " + s
        return(strue)
    npix = 224
    target_size = (npix,npix,3)
    count = 1
    fig = plt.figure(figsize=(10,10))
    npic = len(pred_good)
    for pg in pred_good:
        bleu,jpgfnm,caption_true,caption = pg
        ## images
        filename = image_dataset + '/' + jpgfnm
        image_load = load_img(filename, target_size=target_size)
        ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
        ax.imshow(image_load)
        count += 1

        caption_true = create_str(caption_true)
        caption = create_str(caption)

        ax = fig.add_subplot(npic,2,count)
        plt.axis('off')
        ax.plot()
        ax.set_xlim(0,1)
        ax.set_ylim(0,1)
```

```

code + Markdown | ▶ Run All | ☰ Clear Outputs | ⏪ Restart | ☐ Interrupt | 📄 Variables | ⋮
filename = image_dataset + "/" + jpg_filename
image_load = load_img(filename, target_size=target_size)
ax = fig.add_subplot(npic,2,count,xticks=[],yticks=[])
ax.imshow(image_load)
count += 1

caption_true = create_str(caption_true)
caption = create_str(caption)






ax = fig.add_subplot(npic,2,count)
plt.axis('off')
ax.plot()
ax.set_xlim(0,1)
ax.set_ylim(0,1)
ax.text(0,0.7,"true:" + caption_true,fontsize=15)
ax.text(0,0.4,"pred:" + caption,fontsize=15)
ax.text(0,0.1,"BLEU: {}".format(bleu),fontsize=15)
count += 1

plt.show()

print("Bad Caption")
plot_images(pred_bad)
print("Good Caption")
plot_images(pred_good)

```


... Bad Caption

	<p>true: child in pink dress is climbing up set of stairs in an entry way pred: boy in red shirt is standing on the air BLEU: 7.176794039009363e-232</p>
	<p>true: black dog and spotted dog are fighting pred: black and white dog is running in the grass BLEU: 1.384292958842266e-231</p>
	<p>true: little girl covered in paint sits in front of painted rainbow with her hands in bowl pred: man in red shirt is jumping over the air BLEU: 4.832402486973385e-232</p>
	<p>true: man lays on bench while his dog sits by him pred: black and white dog is running on the grass BLEU: 1.1193096620723278e-231</p>
	<p>true: man in an orange hat starring at something pred: man in black shirt is standing in front of the camera BLEU: 5.477489369001354e-155</p>

+ Code + Markdown ...

Good Caption

<>



true: brown dog is running in the sand

pred: brown dog is running in the water

BLEU: 0.8091067115702212

The image shows a brown dog running on a sandy surface. The dog is captured in motion, with its front legs extended forward and its back legs pushing off. The sand is light-colored and appears to be a beach or a similar outdoor setting. The dog's shadow is cast on the sand to its right, indicating a light source from the upper left. The overall scene is a simple, naturalistic depiction of a dog in its environment.