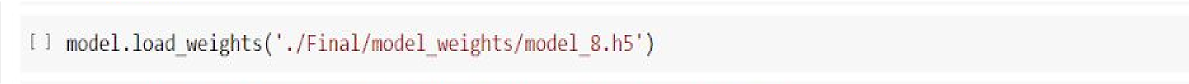
README

Please note that glove vectors have been used for word embedding and “glove.6B.200d” will be required for running the model (to obtain the embedding matrix).

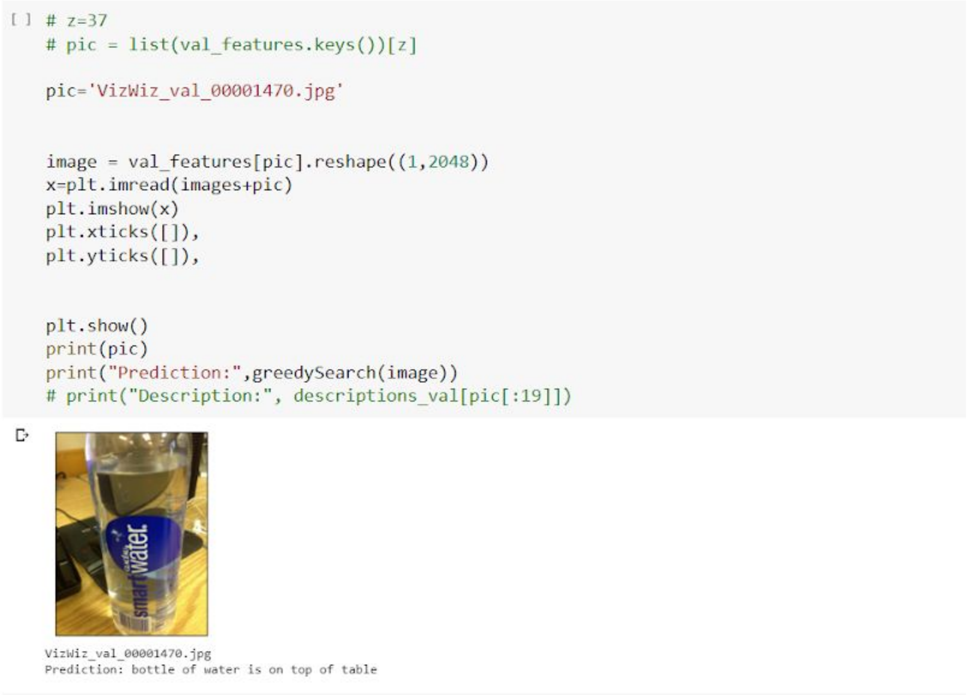
Running the Pre-trained model:

The model_weights subfolder contains saved model weights named model_epoch_number.h5
The best working model is “model_8.h5” (obtained after 9 epochs).
After importing all the libraries required and running the functions, compile the model as in the code cells and then load the model: “model_8.h5”.



```
[ ] model.load_weights('./Final/model_weights/model_8.h5')
```

Now to test on any image given in the sample folder or any other image of the validation set , enter the name as the value of ‘pic’ and run the cell:



```
[ ] # z=37
    # pic = list(val_features.keys())[z]

    pic='VizWiz_val_00001470.jpg'


    image = val_features[pic].reshape((1,2048))
    x=plt.imread(images+pic)
    plt.imshow(x)
    plt.xticks([]),
    plt.yticks([]),


    plt.show()
    print(pic)
    print("Prediction:",greedySearch(image))
    # print("Description:", descriptions_val[pic[:19]])
```

```
VizWiz_val_00001470.jpg
Prediction: bottle of water is on top of table
```

If you have the whole validation set available, you can use any value of z in the range
(0, len(validation set)) to try out a result.

For highly blurred, at inappropriate angles, “Quality issues too severe to predict an output” is printed as the result.

Training the model:

To train on the images for the final captioning, first the data is preprocessed using the **Inception V3 mode**l for feature extraction on **ImageNet** dataset.

The **Bottleneck Features** are extracted by removing the last layer of the model and processing all images on it. The results are temporarily stored in **encoding_train** (and corresponding encoding_val) and then written in a file **"encoded_train_images_3.pkl"** (and corresponding "encoded_val_images_3.pkl")

After compiling the model, we use an **'adam'** optimizer for learning with the learning rate set as default i.e. 0.001 in the beginning.

We train with a batch size of 6 and learning rate = 0.001 for the first 2 epochs.

For the subsequent training, **for every 2 epochs**, keep increasing the **batch size twofold** and decreasing the **learning rate to half**.

Decreasing the learning rate is important as the model is towards convergence in the later stages of training and a high learning rate can lead to **overfitting**.

Increasing the batch size is also important as it makes the **gradient updates more powerful**.

References:

https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8

https://arxiv.org/abs/1411.4555