**CS6301 MACHINE LEARNING – MINI PROJECT**

SRIHARI. S T.K.S. ARUNACHALAM

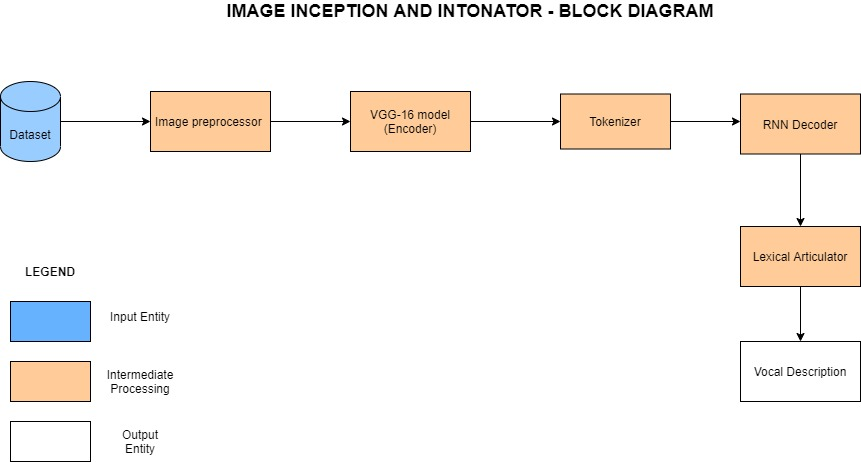
2018103601 2018103616

**IMAGE INSCRIPTION AND INTONATOR - A NEURAL NETWORK APPROACH**

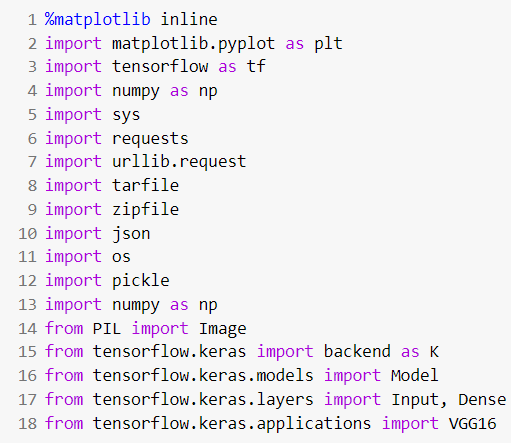
**40% IMPLEMENTATION UPDATE DOCUMENTATION**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DATASET USED:** MS-COCO Dataset (<https://cocodataset.org/#download>)



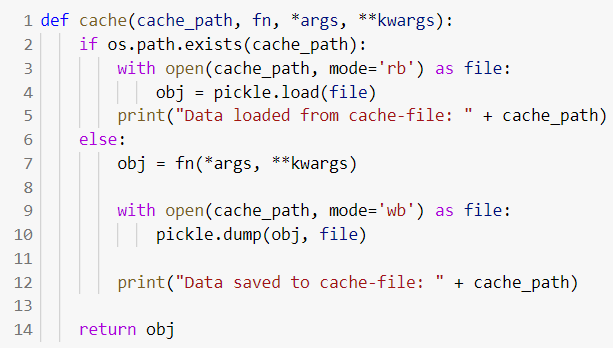
|  |  |
| --- | --- |
| **IMPLEMENTATION PROGRESS** | |
| **Completed Modules** | Image Pre-processor  VGG-16 Model (Encoder) |
| **Ongoing Modules** | Tokenizer  RNN Decoder  Lexical Articulator |



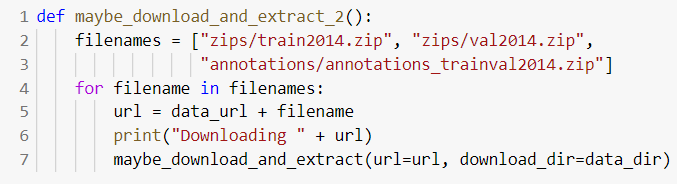


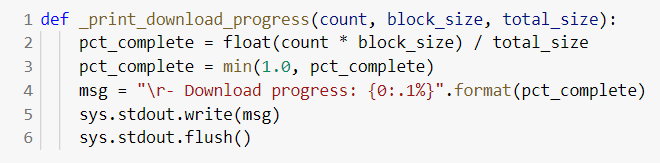
An iterative approach has been chosen for the implementation of the problem statement. The first goal was to understand the nature of the dataset and pre-process it. The input MS-COCO 2014 dataset is of size 25 GB. In-order to deal with this huge dataset and the constrained computing resources we make use of the dynamic programming paradigm by caching the values, the first time the dataset is downloaded, in-order to make access faster the subsequent times. The dataset consisting of both images and captions together is loaded. The images and the corresponding captions are then segregated and stored separately. The images then undergo normalization followed by scaling to finish the pre-processing. On the other hand, the captions are encoded in a dictionary and are thus pre-processed so that it could be used by the tokenizer.

The cached file is stored as a pickle object and the function to accomplish this is defined as below. This is used to persist the data so it can be reloaded very quickly and easily. If the cache-file exists then the data is reloaded and returned, otherwise the function is called and the result is saved to cache.



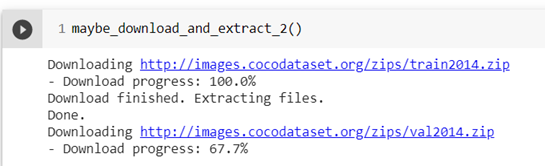
We facilitate the download of the dataset in the desirable format with the aid of the below functions.

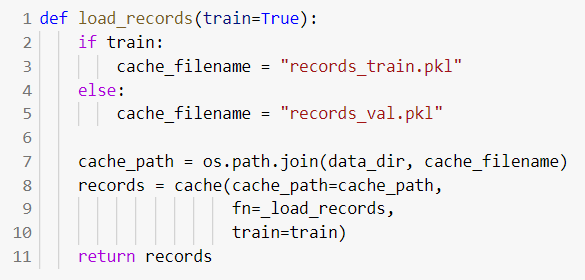


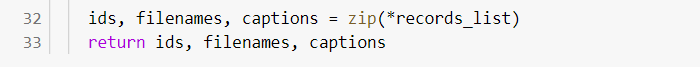


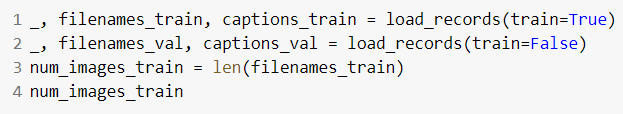


**Downloading the dataset:**

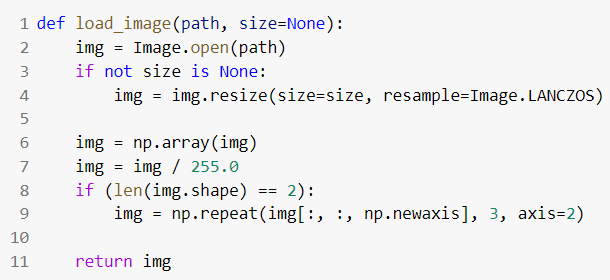


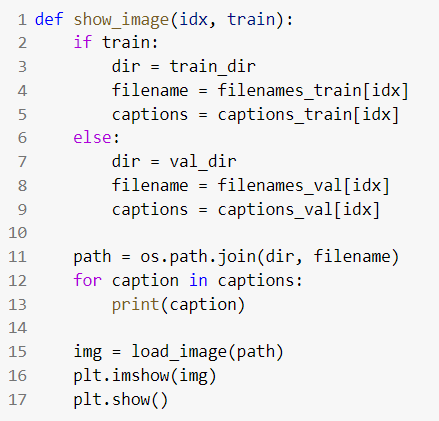
The COCO data-set contains a large number of images and various data for each image stored in a JSON-file. The load\_records function provides the functionality to get a list of image-filenames (but not actually loading the images) along with their associated data such as text-captions describing the contents of the images. 

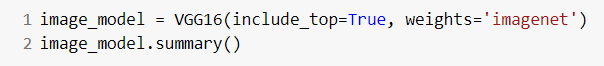


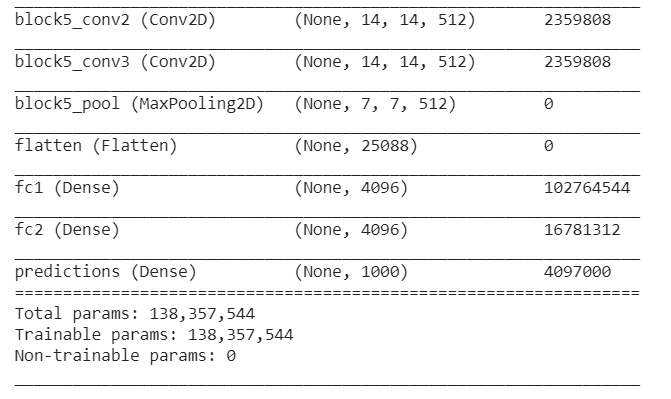
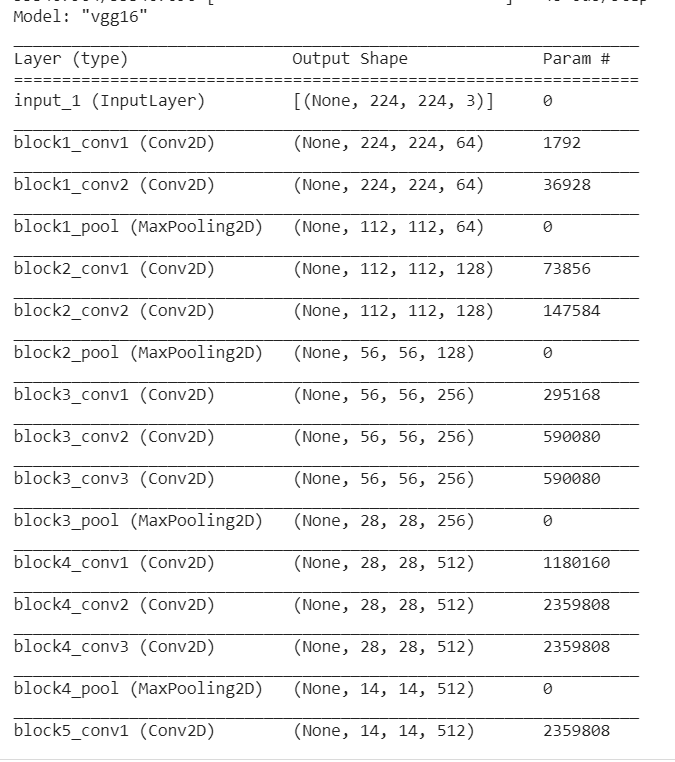


The below given load\_image function accomplishes the job of image pre-processing. It loads the image from the given file-path and resizes it to the given size. The images are scaled so that their pixels fall between 0.0 and 1.0. It is then plotted with the show\_image function.









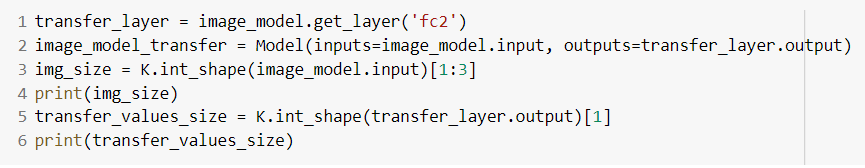
We instantiate the VGG16 architecture by importing it from tensorflow.keras.applications. It loads weights pre-trained on ImageNet. The default input size for this model is 224x224. We remove the last predictions layer and transfer the values of the second fully connected layer.



The get\_layer() method takes as parameters name of the specific layer which we want and retrieves the layer correspondingly.The transfer\_layer variable has the fc2 layer stored. We make use of the K.int\_shape() function which returns the shape of tensor or variable as a tuple of int or None entries.

1. K.int\_shape(image\_model.input) – Denotes shape of input vector to the model having value (None, 224, 224, 3).
2. K.int\_shape(transfer\_layer.output) – Denotes shape of output vector of fc2 layer having value (None, 4096).

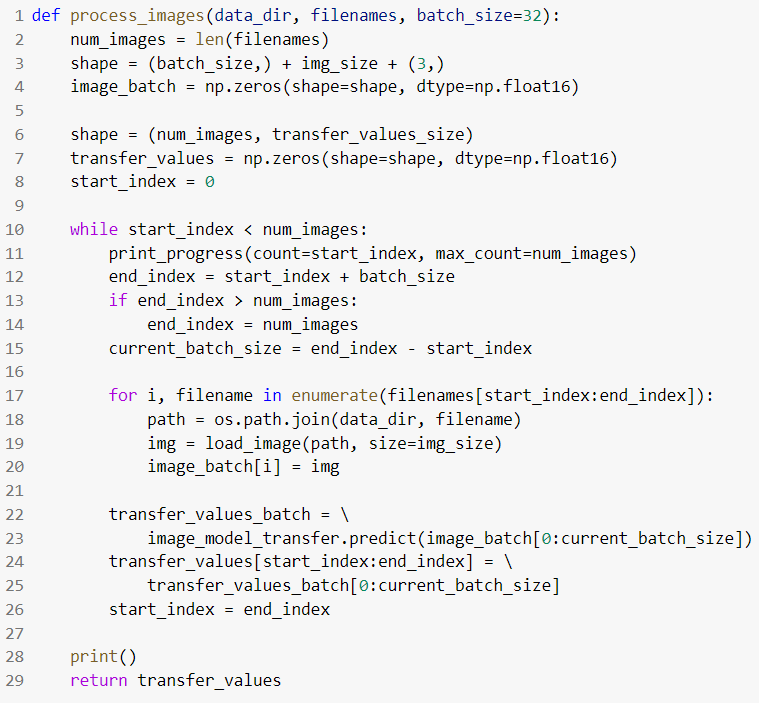
Thus, we assign img\_size and transfer\_values\_size the values (224,224) and 4096 respectively.

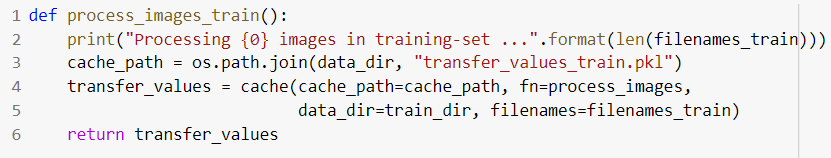


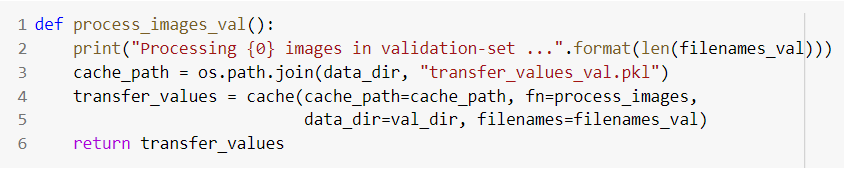
Next step is to process all the images with the vgg16 model and cache the values.

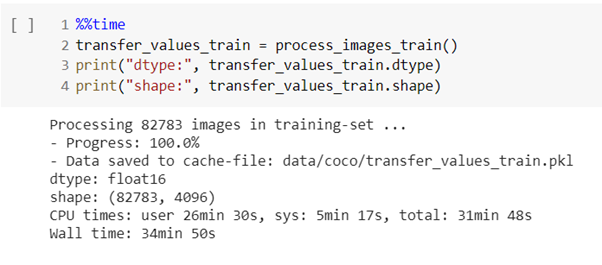
In-order to cache the transfer-values, the cache function is called upon with the path as data/coco/transfer\_values\_train.pkl .

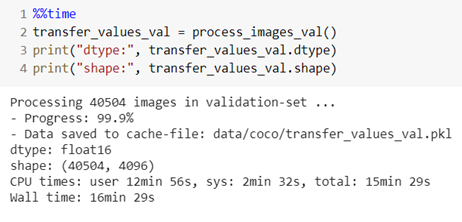
If the cache path i.e. the transfer\_values\_train.pkl file exists, we return the obj. If it doesn't exist the caching of the transfer-values occurs in batches of images. During this process, we load the image using load\_image() function defined previously and the images are resized to meet the expected format for the vgg16 architecture. Thus, we have cached the transfer values for the 82783 images in the training dataset by having the features extracted from the image from the output of the fc2 layer of vgg16 model.

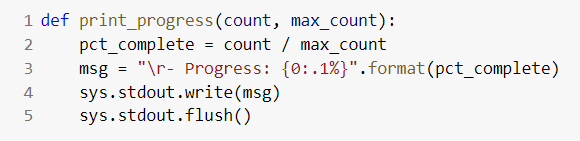












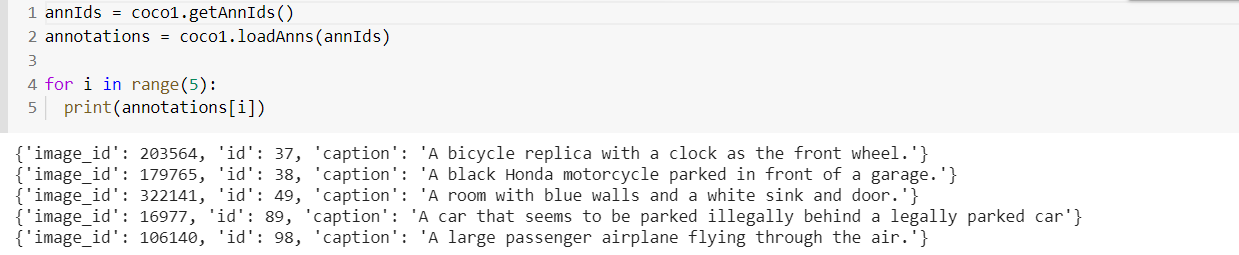
In-order to keep track of the progress, the percentage of completed downloads is constantly updated.

The pycocotools has been put into use. It is a Python API that assists in loading, parsing and visualizing the annotations in COCO. We instantiate the COCO class by passing the json file as an argument.



**Sample of the Annotations:**

Annotations is a list of dictionaries. The dictionary contains the image\_id, id(caption id), caption as the keys. Here id is the primary key and is used to retrieve a unique caption



img is a dictionary with the following keys. We use the coco\_url to load and display the image



**Using the load\_img method to display an image from the url:**

