**CS6301 MACHINE LEARNING – MINI PROJECT**

SRIHARI. S T.K.S. ARUNACHALAM

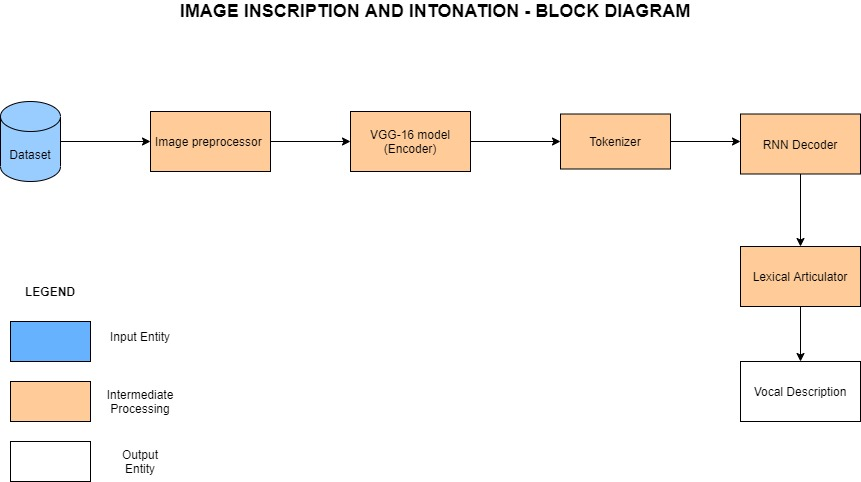
2018103601 2018103616

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

**50% IMPLEMENTATION UPDATE DOCUMENTATION**

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

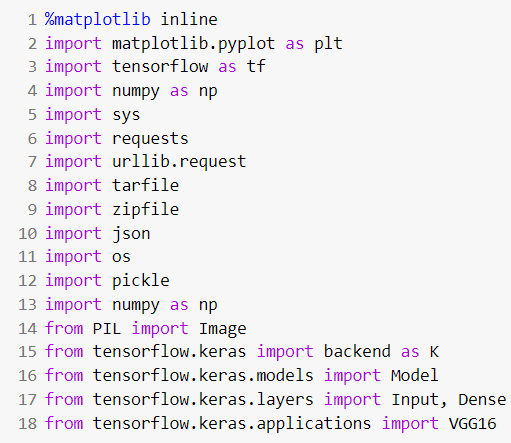
**DATASET USED:** MS-COCO Dataset (<https://cocodataset.org/#download>)

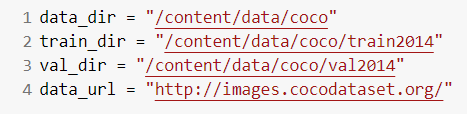


|  |  |
| --- | --- |
| **IMPLEMENTATION PROGRESS** | |
| **Completed Modules** | Image Pre-processor  VGG-16 Model (Encoder)  Tokenizer |
| **Ongoing Modules** | RNN Decoder  Lexical Articulator |

|  |  |
| --- | --- |
| **INDIVIDUAL CONTRIBUTION** | |
| **SRIHARI.S** | **ARUNACHALAM.T.K.S.** |
| Caption Pre-processing | Image Pre-processing |
| Transfer Learning on the dataset using VGG16 | Incorporating PyCoCo Tools for dealing with the data |
| Vocabulary construction | Embeddings productions |

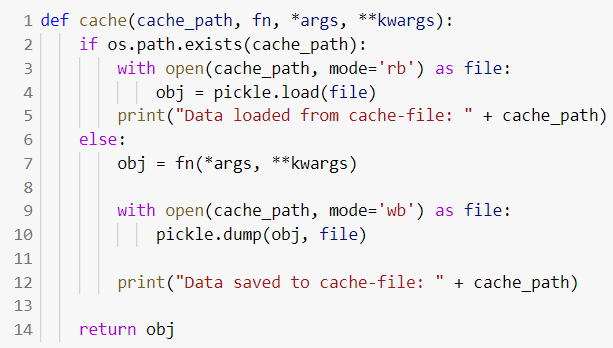
**MODULE - 1 PRE-PROCESSOR**



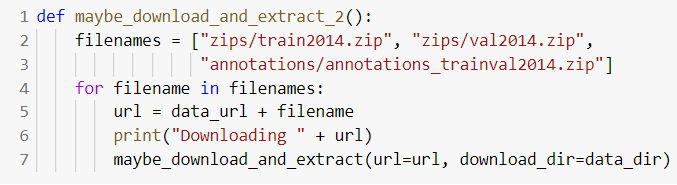


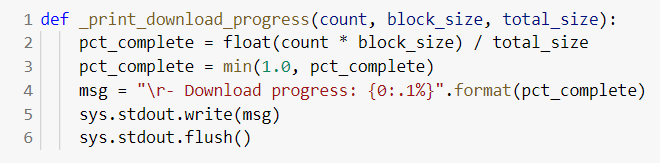
An iterative approach has been chosen for the implementation of the problem statement. The first goal was to understand the nature of the dataset and pre-process it. The input MS-COCO 2014 dataset is of size 25 GB. In-order to deal with this huge dataset and the constrained computing resources we make 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.



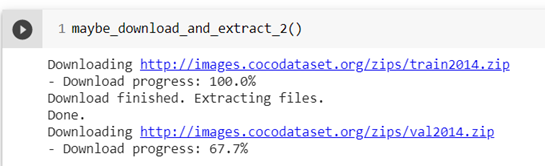
We facilitate the download of the dataset in the desirable format with the aid of the below functions.

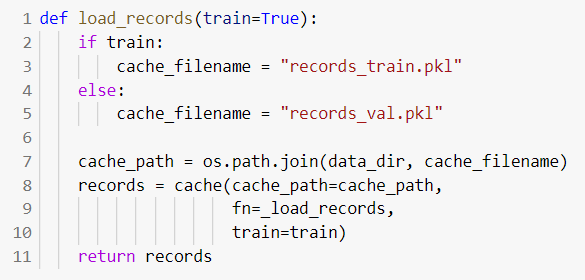


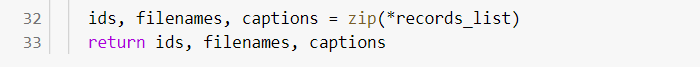


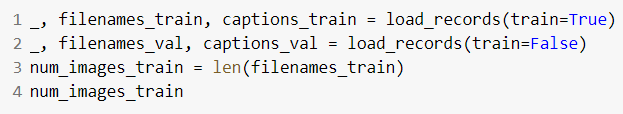


**Downloading the dataset:**



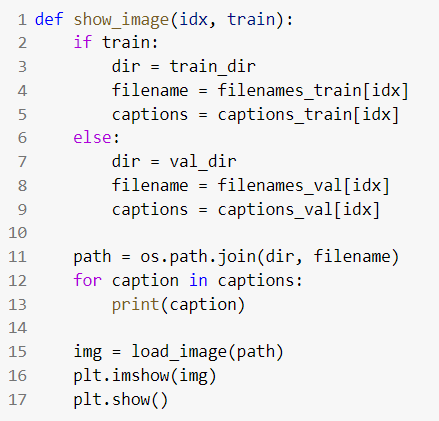
The COCO data-set contains a large number of images and various data for each image stored in a JSON-file. The load\_records function provides the functionality to get a list of image-filenames (but not actually loading the images) along with their associated data such as text-captions describing the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



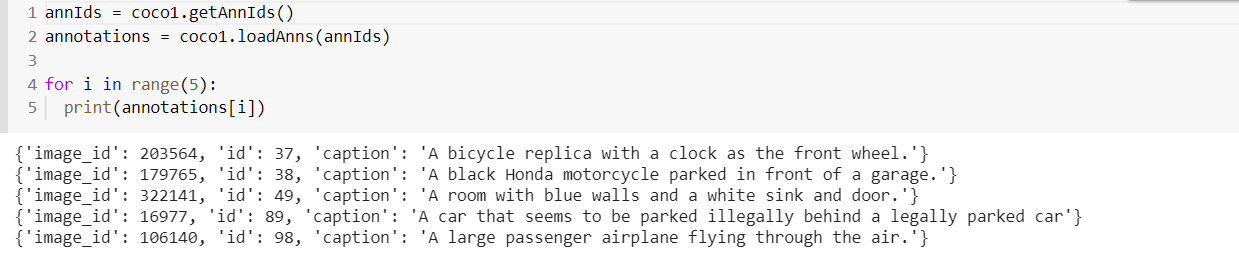


The pycocotools has been put into use. It is a Python API that assists in loading, parsing and visualizing the annotations in COCO. We instantiate the COCO class by passing the json file as an argument.



**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



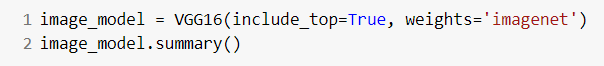
img is a dictionary with the following keys. We use the coco\_url to load and display the image

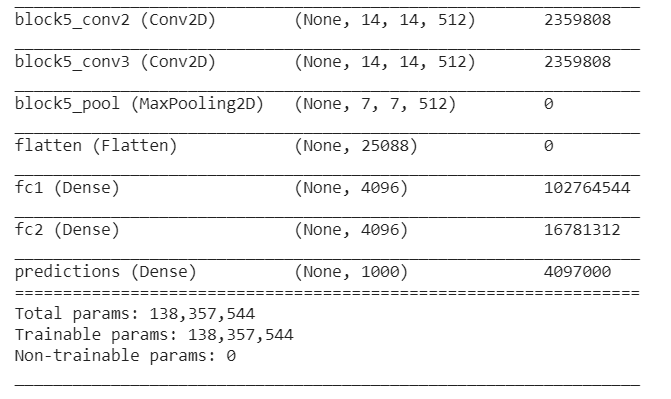
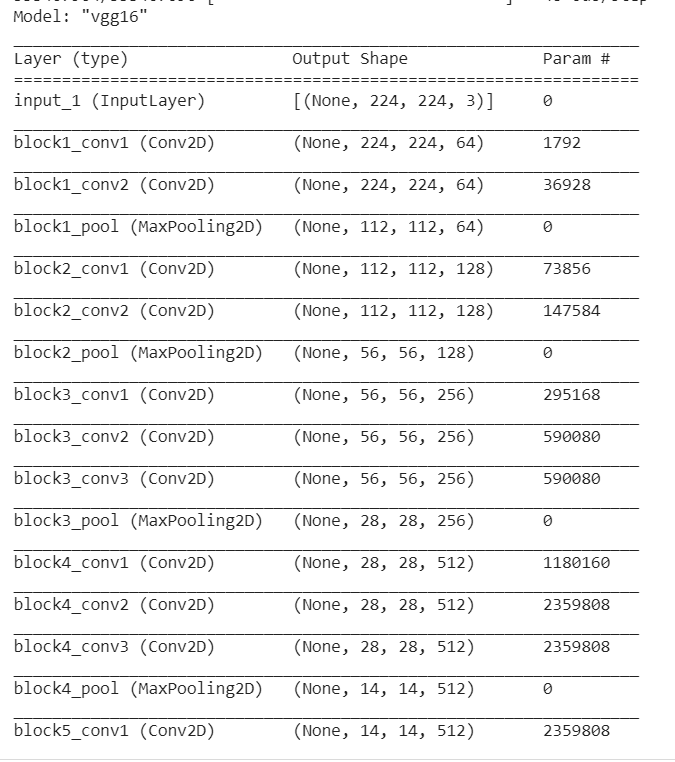


**Using the load\_img method to display an image from the url:**



**MODULE – 2 VGG 16 MODEL ENCODER**





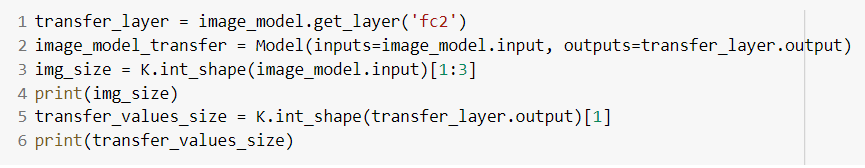
We instantiate the VGG16 architecture by importing it from tensorflow.keras.applications. It loads weights pre-trained on ImageNet. The default input size for this model is 224x224. We remove the last predictions layer and transfer the values of the second fully connected layer.



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).

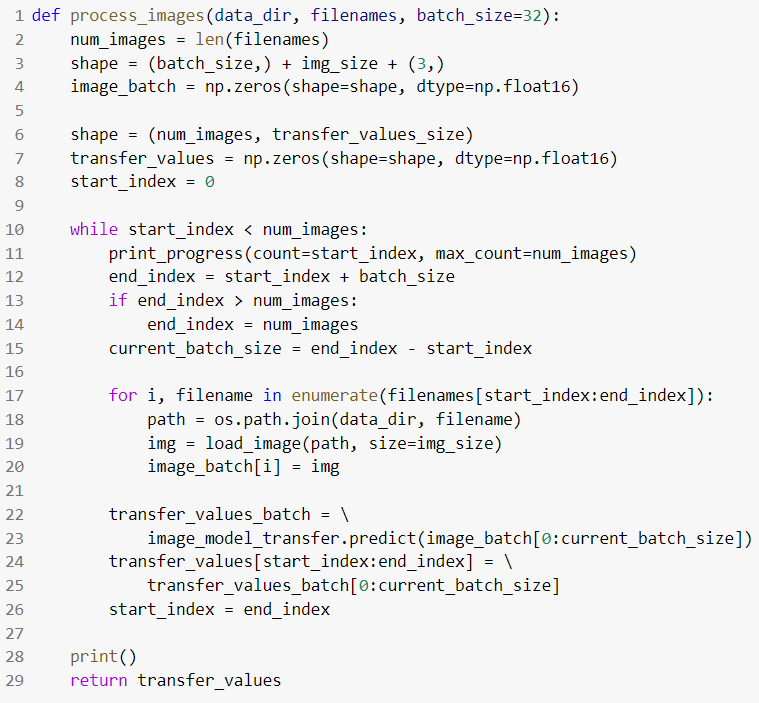
Thus, we assign img\_size and transfer\_values\_size the values (224,224) and 4096 respectively.

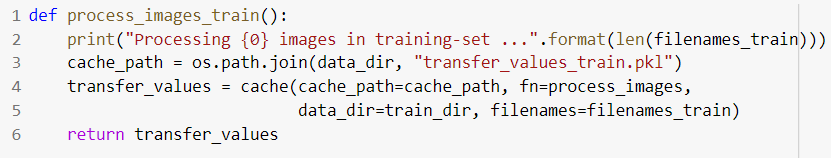


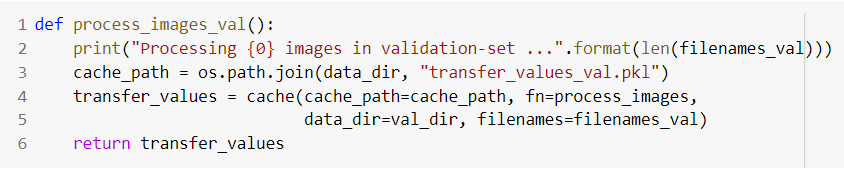
Next step is to process all the images with the vgg16 model and cache the values.

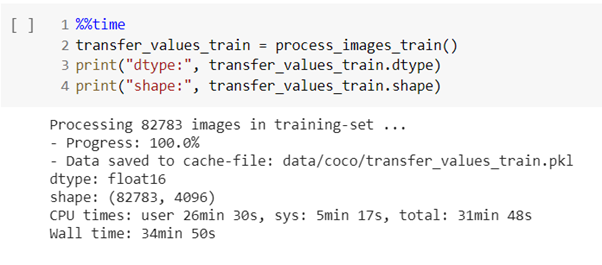
In-order to cache the transfer-values, the cache function is called upon with the path as data/coco/transfer\_values\_train.pkl .

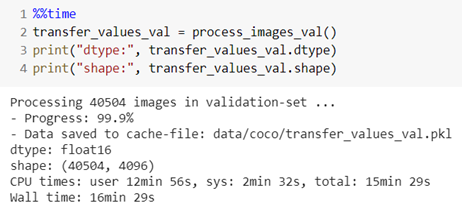
If the cache path i.e. the transfer\_values\_train.pkl file exists, we return the obj. If it doesn't exist the caching of the transfer-values occurs in batches of images. During this process, we load the image using load\_image() function defined previously and the images are resized to meet the expected 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.

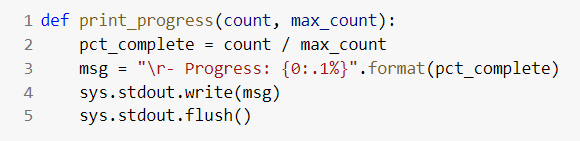








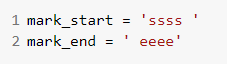




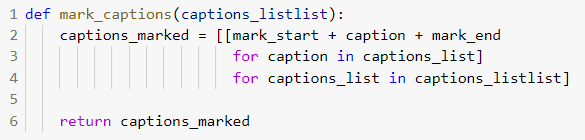
In-order to keep track of the progress, the percentage of completed downloads is constantly updated.

**MODULE - 3 TOKENIZER**

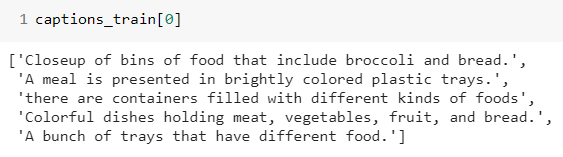
Neural Networks cannot work directly on text-data. We use a two-step process to convert text into numbers that can be used in a neural network. The first step is to convert text-words into so-called integer-tokens. The second step is to convert integer-tokens into vectors of floating-point numbers using a so-called embedding-layer.



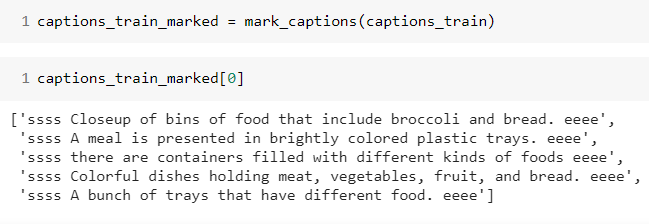
Before we can start processing the text, we first need to mark the beginning and end of each text-sequence with unique words that most likely aren't present in the data.



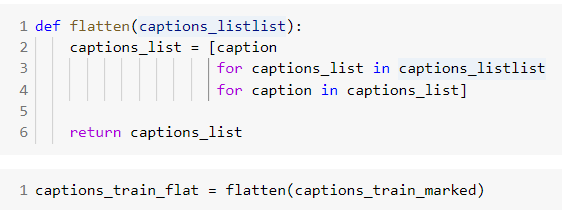
The mark\_captions function wraps all text-strings in the above markers. Since the captions are a list of lists, we use a nested for-loop to process using list-comprehension in python. In the inner loop we iterate over all the captions as we have four to six captions for each image. For each caption we append the start and end marker. Thus, captions\_marked is a list of lists which is returned by this function to the variable captions\_train\_marked.



captions\_train is obtained from the pre-processor module. It’s a tuple of lists. Thus, each element of the tuple captions\_train is as shown above.

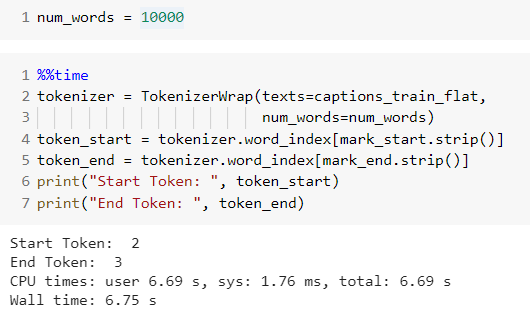


captions\_train\_marked is a list of lists where each caption has the start and end markers added. Thus, each element of the tuple captions\_train\_marked is like shown above.



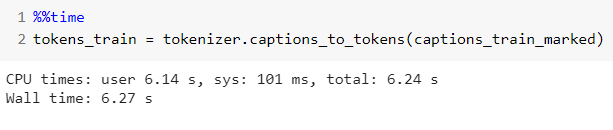
Now process all the captions in the training-set. Next we make a call to flatten() function to which we pass as parameters the marked captions. The functionality achieved here is pretty straightforward and simple.

Basically, we’re converting list of lists into a single list. But now we unwrap the inner lists to provide a single flattened list. Thus, captions\_list is a python list having all the captions in the training dataset. This is returned to captions\_train\_flat.



The TokenizerWrap is inherited from the Tokenizer class in tensorflow.keras.preprocessing.text as we need more functionality than provided by this Tokenizer class so we wrap it.

We instantiate a TokenizerWrap object as below by passing the flat list of captions and num\_words as parameters. TokenizerWrap is used to convert a text into sequence of integers. The maximum number of words in the vocabulary is set using the num\_words variable to 10000. This means that we will only use the 10000 most frequent words in the captions from the training-data.



Next we pass the flattened caption to the fit\_on\_texts() function of Tokenizer class. This function updates internal vocabulary based on a list of texts. Then we iterate over the items of the list.

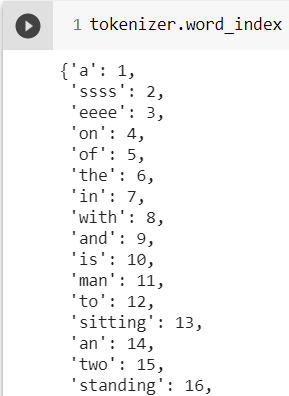
Now create a tokenizer using all the captions in the training-data. The flattened list of captions is used to create the tokenizer because it cannot take a list-of-lists.





This is the sample execution of text\_to\_word\_sequence(). We only pass the text parameter which is each string in the flattened list. Initially we convert it to lowercase. translate\_dict is a dictionary where keys are the punctuations and values are space. The maketrans() method returns a mapping table that can be used with the translate() method to replace specified characters.

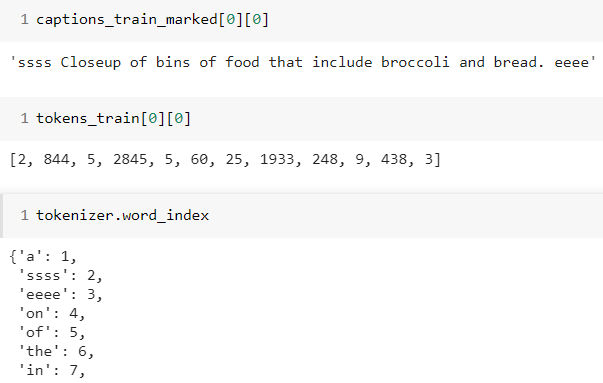
In the mapping table we replace the keys in translate\_dict dictionary with their corresponding ASCII values. Using translate function we replace the instances of the key in the text with space. i.e We turn all the punctuators into spaces. Now we split the text having space as the delimiter. In the end we return a list from the split text consisting of words which are of size at least 1.

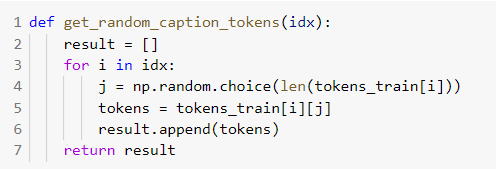


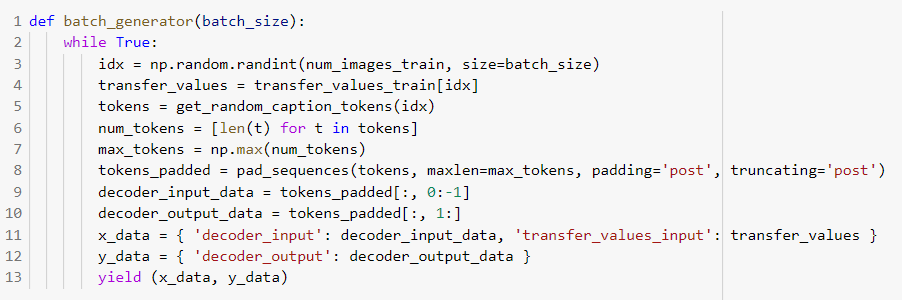
Next passing captions\_train\_marked which is a list of lists of marked captions to the function captions\_to\_tokens, we call the function texts\_to\_sequences() of Tokenizer class for each list in the variable caption\_listlist.

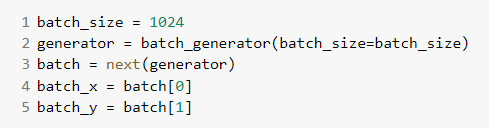


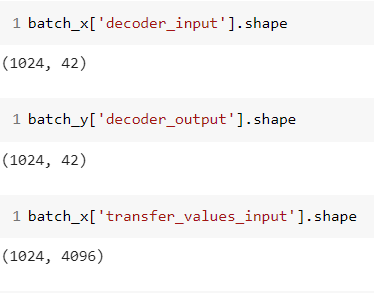
tokens\_train displays the output of the tokenizer (sequence of integers) .

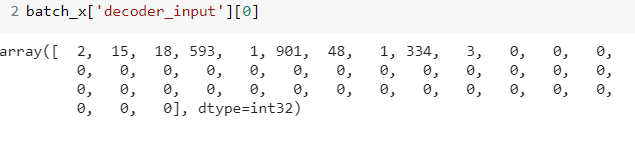


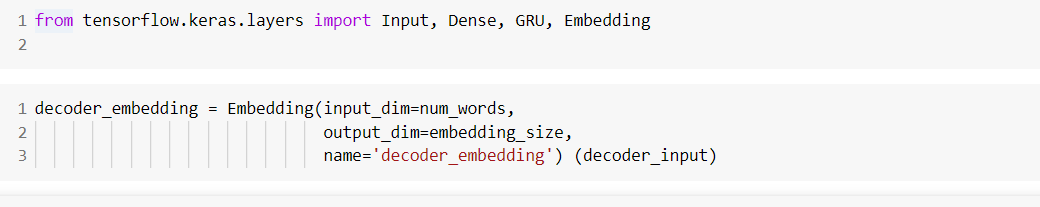












**MODULE WISE TEST CASES**

|  |  |  |
| --- | --- | --- |
| **MODULE 1: PRE-PROCESSOR** | | |
| **INPUT** | **SEPARATED IMAGE** | **SEPARATED CAPTIONS** |
| {"license": 5,"file\_name": "COCO\_train2014\_000000057870.jpg","coco\_url": "http://images.cocodataset.org/train2014/COCO\_train2014\_000000000025.jpg","height": 480,"width": 640,"date\_captured": "2013-11-14 16:28:13","id": 57870} |  | ['A giraffe eating food from the top of the tree.', 'A giraffe standing up nearby a tree ', 'A giraffe mother with its baby in the forest.', 'Two giraffes standing in a tree filled area.', 'A giraffe standing next to a forest filled with trees.'] |
| {"license": 5,"file\_name": "COCO\_train2014\_000000384029.jpg","coco\_url": "http://images.cocodataset.org/val2014/COCO\_val2014\_000000003014.jpg","height": 429,"width": 640,"date\_captured": "2013-11-14 16:29:45","id": 384029} |  | ['Three-quarters of a meat-lovers pizza with mushrooms with drink', 'A partially eaten pizza is sitting beside a soda.', 'A table with a partially eaten pizza and blue canned beverage on it.', 'A close up of a pizza and a drink on a table.', 'A pizza with cherry tomatoes has a piece taken out.'] |
| {"license": 1,"file\_name": "COCO\_train2014\_000000222016.jpg","coco\_url": "http://images.cocodataset.org/train2014/COCO\_train2014\_000000000404.jpg","height": 640,"width": 480,"date\_captured": "2013-11-14 16:37:59”,"id": 222016} |  | ['a couple of boats that are in some water', 'A pair of boats docked at a pier is shown.', 'Three boats docked in still water with clouds in the sky. ', 'Three boats are docked together on the cloudy day.', 'some colorful boats sitting next to a dock '] |
| {"license": 3,"file\_name": "COCO\_val2014\_000000391895.jpg","coco\_url": "http://images.cocodataset.org/train2014/COCO\_train2014\_000000000089.jpg","height": 360,"width": 640,"date\_captured": "2013-11-14 11:18:45","id": 391895} |  | ['An oven with a stove on top of it in a kitchen.', 'A stove with a lighted hood in the kitchen.', 'A small light is on above the polished stove top.', 'Smooth top stove with exhaust fan that has light turned on.', 'A stove top is cleaned with a set of knives on the wall.'] |
| {"license": 4,"file\_name": "COCO\_val2014\_000000522418.jpg","coco\_url": "http://images.cocodataset.org/val2014/COCO\_val2014\_000000000073.jpg","height": 480,"width": 640,"date\_captured": "2013-11-14 11:38:44","id": 522418} |  | ['A motorcycle parked in a parking space next to another motorcycle.', 'An old motorcycle parked beside other motorcycles with a brown leather seat.', 'Motorcycle parked in the parking lot of asphalt.', 'A close up view of a motorized bicycle, sitting in a rack. ', 'The back tire of an old style motorcycle is resting in a metal stand. '] |

|  |  |
| --- | --- |
| **MODULE 2: ENCODER** | |
| **INPUT** | **OUTPUT – TRANSFER VALUES** |
|  |  |
|  |  |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **MODULE 3: TOKENIZER** | | | |
| **INPUT** | **INTERMEDIATE OUTPUT** | **OUTPUT SEQUENCE** | **EMBEDDINGS** |
| 'Closeup of bins of food that include broccoli and bread.' | 'ssss Closeup of bins of food that include broccoli and bread. eeee' | [2, 844, 5, 2845, 5, 60, 25, 1933, 248, 9, 438, 3] | [[0.41177 , -2.223 , -1.0756 , -1.0783 ,]...[0.15155 , 0.78321 , -0.91241]] |
| 'A giraffe eating food from the top of the tree.' | 'ssss A giraffe eating food from the top of the tree. eeee' | [2, 1, 117, 108, 60, 96, 6, 32, 5, 6, 133, 3] | [[ 0.50451 , 0.68607 , -0.59517,]...[0.60046,-0.13498,.. -0.022801]] |
| 'White vase with different colored flowers sitting inside of it. ' | 'ssss White vase with different colored flowers sitting inside of it. eeee' | [2, 21, 202, 8, 191, 395, 200, 13, 159, 5, 30, 3] | [[ 0.32157 , 0.45894 , -0.32014,]...[0.32015,-0.23156,.. -0.30215]] |
| ‘A cat is lying on its back in a man's lap’ | ssss A cat is lying on its back in a man's lap. eeee | [2, 1, 50, 10, 370, 4, 154, 163, 7, 1, 1248, 584, 3] | [[ 0.09981 , 0.45477 , -0.87890,]...[0.12344,-0.98757,.. -0.00854]] |
| ‘a giraffe standing in tall grass with trees in the background .’ | ssss a giraffe standing in tall grass with trees in the background eeee | [2, 6, 117, 10, 16, 683, 7, 6, 1613, 3] | [[ 0.54687 , 0.63547 , -0.77894,]...[0.30210,-0.65489,.. -0.32154]] |