**MODULES**

**1. matplotlib.pyplot**

**2. pyplot.imshow(X)**

Display an image on the axes.

X : array\_like, shape (n, m) or (n, m, 3) or (n, m, 4)

Display the image in X to current axes. X may be a float array, a uint8 array or a PIL image. If X is an array, it can have the following shapes:

MxN -- luminance (grayscale, float array only)MxNx3 -- RGB (float or uint8 array)MxNx4 -- RGBA (float or uint8 array)

**3. CV2**

**i. cv2.imread()**

This method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

Syntax: cv2.imread(path, flag)

Parameters:

path: A string representing the path of the image to be read.

flag: It specifies the way in which image should be read. It’s default value is cv2.IMREAD\_COLOR

Return Value: This method returns an image that is loaded from the specified file.

cv2.IMREAD\_COLOR: It specifies to load a color image. Any transparency of image will be neglected. It is the default flag. Alternatively, we can pass integer value 1 for this flag.

cv2.IMREAD\_GRAYSCALE: It specifies to load an image in grayscale mode. Alternatively, we can pass integer value 0 for this flag.

cv2.IMREAD\_UNCHANGED: It specifies to load an image as such including alpha channel. Alternatively, we can pass integer value -1 for this flag.

**ii. cv2.split()**

Is used to split an image into three different intensity arrays for each color channel

**iii. cv2.merge()**

Is used to merge different arrays into a single multi-channel array, that is, a color image

**iv. cv.resize()**

Scaling is just resizing of the image.

OpenCV comes with a function cv.resize() for this purpose. The size of the image can be specified manually, or you can specify the scaling factor. Different interpolation methods are used. Preferable interpolation methods are cv.INTER\_AREA for shrinking and cv.INTER\_CUBIC (slow) & cv.INTER\_LINEAR for zooming. By default, interpolation method used is cv.INTER\_LINEAR for all resizing purposes

**4. Numpy**

**i. numpy.array()**

numpy.array(data type, value list) function is used to create an array with data type and value list specified in its arguments.

Arrays are used to store multiple values in one single variable.Python does not have built-in support for Arrays, but Python lists can be used instead.

**ii. numpy.hstack()**

This function is used to stack the sequence of input arrays horizontally (i.e. column wise) to make a single array.

Syntax : numpy.hstack(tup)

Parameters :

tup : [sequence of ndarrays] Tuple containing arrays to be stacked. The arrays must have the same shape along all but the second axis.

Return : [stacked ndarray] The stacked array of the input arrays.

**iii. numpy.random.shuffle()**

Modify a sequence in-place by shuffling its contents.

This function only shuffles the array along the first axis of a multi-dimensional array. The order of sub-arrays is changed but their contents remains the same.

Parameters:

x : array\_like

The array or list to be shuffled.

Returns:None

**5. Tensorflow**

**i. tf.estimator.train\_and\_evaluate()**

This utility function trains, evaluates, and (optionally) exports the model by using the given estimator. All training related specification is held in train\_spec, including training input\_fn and training max steps, etc. All evaluation and export related specification is held in eval\_spec, including evaluation input\_fn, steps, etc.

**ii. tf.estimator.TrainSpec()**

TrainSpec determines the input data for the training, as well as the duration. Optional hooks run at various stages of training.

**iii. tf.estimator.EvalSpec()**

EvalSpec combines details of evaluation of the trained model as well as its export. Evaluation consists of computing metrics to judge the performance of the trained model. Export writes out the trained model on to external storage.

**6. Sklearn**

sklearn.model\_selection.train\_test\_split(\*arrays, \*\*options)

Split arrays or matrices into random train and test subsets

**FUNCTIONS**

**normalize(df)**

* Used to normalize data by using standard formula.
* It scales the column of large values within the range 0 to 1

**denormalize(df,norm\_data)**

* Once the prediction is done, the data is required to be scaled back to original form.
* Hence, Denormalization is done to reverse scale it back to original value.

**getdata(path)**

* Here, the path for images is provided to fetch all the data required to train and test the model.
* For all the images in given path, the file name contains keyword which specifies its class.
* Initially, Images are segregated into respective list of each class.
* Simultaneously, a label is appended into a second list.
* Therefore, it provides us with 8 lists: 4 separate lists of Class-wise image-paths and their respective classes in another 4 set of lists.
* Thereafter, the lists are stacked together to form 2 discrete list, one for image path and other for their respective class.
* To randomize the process, the list is zipped together and shuffled.
* The randomized list is then separated into ImagePath and Label List and returned back as output of function.

**process(path)**

* It reads and processes image file in 4D tensor format i.e. [batch\_size, IMG\_Width, IMG\_Height, channel] for grayscale channel = 1, for RGB, channel = 3
* It receives the ImagePath and Label List form getdata()
* Traversing through each image, follow operations take place.
  + The image is read as a color image
  + The image is split in 3 color components, as it is read as a BGR image, but computation require an RGB input.
  + The image is recombined as a RGB image.
  + The image is resized to a size of IMG\_W x IMG\_H resolution. (Here, 160x160). Interpolation is cv2.INTER\_CUBIC which implies the image is zoomed.
  + The image is appended into a global list of images.
* The function returns a numpy array of processed images as 3D matrices and an array of Labels.

**CELL-WISE DESCRIPTION**

**Cell 1**

The required modules are imported.

Required Global Variables are defined.

gs –

N-Classes –

ImgW

ImgH

Learning Rate

disease –

Definition of function

Normalize()

Denormalize()

**Cell 2**

Definition of function

Getdata()

Process()

**Cell 3**

Initalize path

Process and get data using path.

**Cell 4**

Reshape every image into a 1D array. (Flatten a 3D array into a 1D array)

**Cell 5**

Prints shape of current data array

**Cell 6**

Displays image and its current label

**Cell 7**

Divide the data into training and testing data. Provides us 70:30 ratio of train to test data.

**Cell 8**

The input images in test and train have integer values more than 1. Therefore, needs to be normalized.

**Cell 9**

Initialize a dummy feature column of size same as input image.

**Cell 10**

Developing a Deep model –

Using feature column developed in cell 9 as input.

Produces an output of 4 class.

As a deep model has 6 fully connected hidden layers with 100,100,100,100,100,100,50 nodes respectively.

ReLU is the activation function as it provides easier computation and works better compared to sigmoid or tanh

It uses GradientDescent optimizer to train the feedforward network with a predefined learning rate of 0.0001.

The model parameters are saved to /model directory

**Cell 11**

Create training input using estimator.input\_fn() that returns input function that would feed dict of numpy arrays into the model.

X\_train and Y\_train is used in function.

**Cell 12**

Create value input using estimator.input\_fn() that returns input function that would feed dict of numpy arrays into the model.

X\_val and Y\_val is used in function.

**Cell 13**

Loop to display Specifications of the training and evaluation phase of the model.

It displays the step by step description of the model working for each training input given.

TrainSpec and EvalSpec are functions from estimator that are used to retrieve specifications of the running model.

**Cell 14**

Create input for predictions from the model.

Predict the result of the deep model for the given input.

**Cell 15**

Store predictions in a dict p used in the model at index of ‘class\_id’

**Cell 16**

Convert prediction list into array

**Cell 17**

Run a loop

Display input image after reshaping.

Display predicted value by the model.

Display actual value in data set.

**Cell 18**

Calculate True values.

Calculate False values.

Calculate error by formula - F/(T+F) \* 100

Display Error.