NOTE FOR DESCRIPTION OF ORGANIZATION OF PYTHON CODE

* Some Basic Version of libraries used in training and testing  
  Tensorflow Version – 1.14.0 used (Any version from 1.1.0 to 1.14.0 can be used)  
  Keras Version – 2.2.4-tf

Python Version – 2.7 and 3.6 both installed

* For image capturing – ‘bgapi2\_gige.cti’ file needs to be initialised at same address as original code

All code and trained modules are present at ‘C:\Users\itc.DESKTOP-RF2G1DL\\_\_\_\_\_\_’ this path ( Default path for any Jupyter Notebook code )

To open a code in Jupyter Notebook ( file should be in .ipynb format) and open command prompt – type jupyter notebook – press enter and upload the file to open it  
  
To run the full code go to Cells column and press run all   
Otherwise, if need to execute each block individually Shift+Enter through each block

There are essentially four tasks:

* Training:-

1. Capture defected and non\_defected images
2. Make a dataset of images for training and testing
3. Training Code for saving finalised model

* Deploying online for real time detection:-

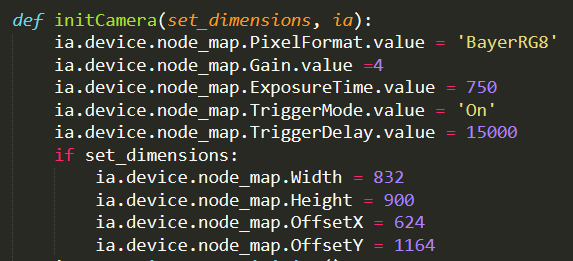
1. Saving captured images and checking accuracy

All 4 tasks have their independent codes

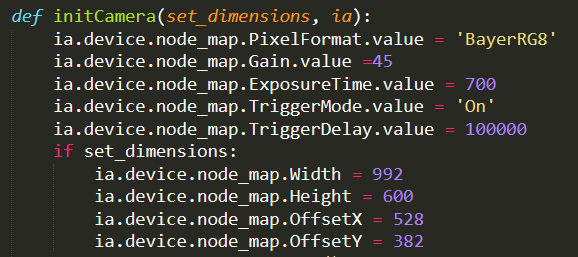
1. First task is capturing good quality images where defect is easily visible.  
   For this task Cam3\_image\_save.py is used

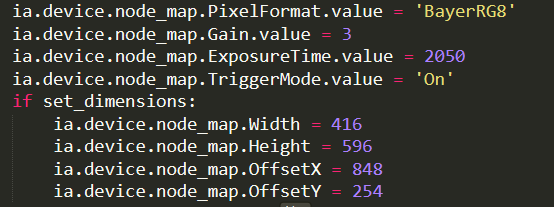
Values to be changed:-

* Line 12 – Camera ID
  + Cam 1 – 700004583388
  + Cam 2 – 700004618338
  + Cam 3 – 700004978993
  + Cam 4 – 700004979003
* Line 15 – File path where images will be saved
* Line 37-47 – Function to define initialization value of cameras

Cam 2 : 

Note : (Cam 2 cannot have gain value more than 4)

Cam 3: 

Cam 4: 

These are just reference values and you might need to set according to your own lighting conditions and packet colour and other conditions ( Trial and error )

* Line 54 ( bgapi2\_gige.cti) this file needs to be at same location as python code as it is accessed for image capturing
* Line 91 – Change dimension of image captured to whatever desired (Reference Value -992\*600 pixels)
* Line 100 – Comment this line if don’t want to see the images being captured

1. To make dataset of images Training\_dataset\_maker.py is used

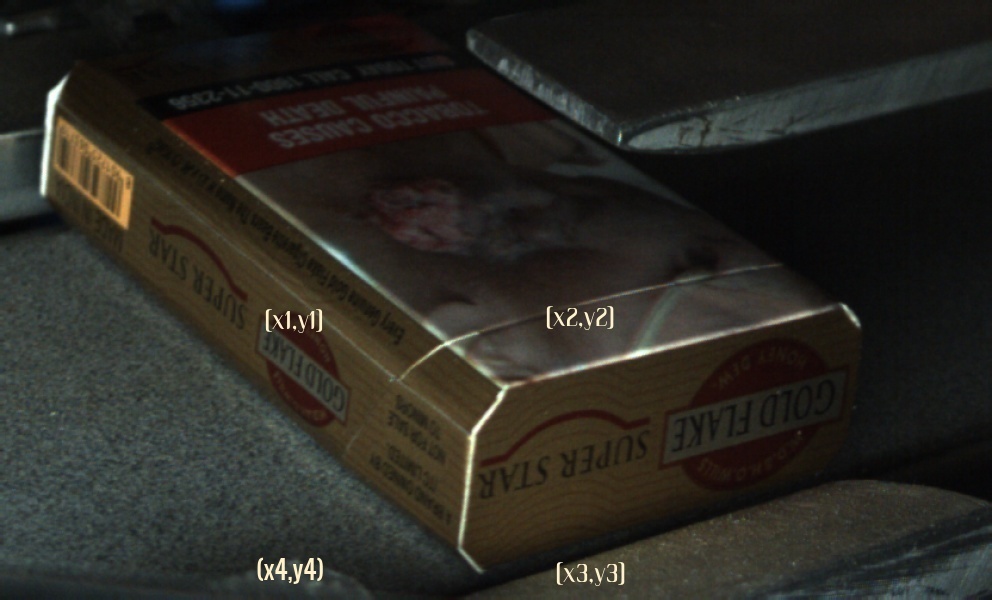
This code essentially has 2 sections:

* Cropping an image based on coordinates
* Augmentation of cropped images

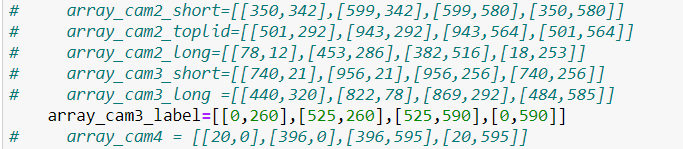
This code makes bounding boxes around coordinates and make a training dataset of each image of size 224\*224 pixels

Generally, Training needs around 2000 non\_defective and 2000 defective images and there is ample amount of data available for good images so try to avoid augmenting them. But can only capture around 400 defective images which need to be augmented 4 times per image to make a dataset of 2000 defective images

Values to be changed:-

* Line 19 – Input image directory
* Line 20 – Output image directory ( If path not available will automatically create a folder)
* Line 34 – Coordinates from which image needs to be cropped   
  

Coordinates\_to be cropped = [[x1,y1], [x2,y2], [x3,y3], [x4,y4]]

* 
* Different array coordinates used in Cam2, Cam 3 and Cam 4 images
* Comment code after line 44 if not using augmentation technique on images
* Line 44 – Input path of augmenting images
* Line 45 – Output path where augmented images are saved
* Line 46- No of copies to be made per image
* Line 50

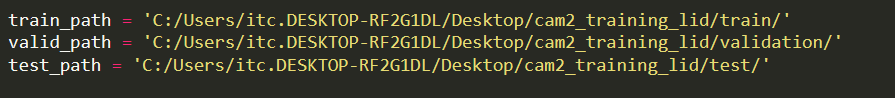

Rotation range specifies in degrees rotation range of augmented images  
 Height\_shift\_range specifies in ratio of how much vertical shift can be incurred  
 Channel\_shift\_range changes contrast  
 Horizontal\_flip is not used as it will create mirror image of original image  
 Brightness range is specified  
  
Many more augmentation options are available which I will specify in developer options whereas for certain task this much augmentation options are good enough

1. Training code for saving finalised model

Initially all the training, testing and validation data needs to be put in a folder in this format

* Cam2\_training\_lid
  + Test
    - Bad
      * Img1
      * Img2
    - Good
      * Img1
      * Img2
  + Train
    - Bad
      * Img1
      * Img2
    - Good
      * Img1
      * Img2
  + Validation
    - Bad
      * Img1
      * Img2
    - Good
      * Img1
      * Img2

Now, use file named Train\_VGG16.py  
  
Values to change:-

* 
* Line 24 – Set Batch Size (don’t increase too much)
* Line 25 – Save\_model\_after 20 epochs (Total epoch to train is 100-200)
* Line 46 – Learning rate( Use from 0.01 to 0.05), don’t change loss and metrics
* Line 48 – Change the name of saved file from cam2\_lid{epoch:06d}.h5 to xxxxxxx{epoch:06d}.h5 (otherwise previous file will be overwritten)
* Line 50 – Set steps\_per\_epoch to floor value of (total no. of training images / Batch\_size)
  + For eg. Let us say there are 2000 good images and 1594 bad images and batch size is 32 so (2000+1594)/32= 112.31  
    So, set it to 112
  + Similarly set validation\_steps by calculating with respect to validation images
* Set no. of epochs to 100-150-200.

Only use model i.e giving loss value less than 0.001 and accuracy more than 0.98  
Even, if loss is zero after 40 epochs, try to run it and only used saved model after 50 epochs

1. Running Model Online and Saving Images

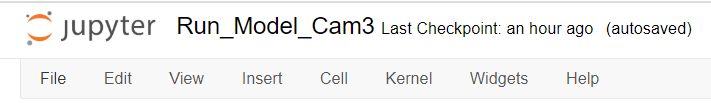
* Unzip contents of CAM3\_DATA or CAM4\_DATA on the desktop, images will be saved to it automatically, after running code file named Run\_Model\_Cam3.ipynb or Run\_Model\_Cam4.ipynb



All the files are available at ‘C:\Users\itc.DESKTOP-RF2G1DL\\_\_\_\_\_\_’

No need to import these as preimported in jupyter notebook

All models are imported and if need to initialize Mint Capsule Model than uncomment that section and comment previous one

Otherwise, just unzip the contents of zip on desktop and   
Press Cell and Run-All   
  
Check if camera is connected or not by going to the end of code to check if image saving has started and it is printing ‘Working’