**Problem Statement**

The area where we need a little help is to build a classifier. We need a model that can classify between an object image, text image and paragraph image. The three classes are given below with example.



|  |  |
| --- | --- |
| Image | Type |
| image.png | Object Image |
| image2.png | Paragraph Image |
| image3 | Text Image |

At the end of this project or exercise the model should be able to classify the Type of image it is.

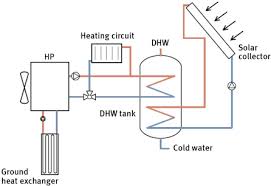
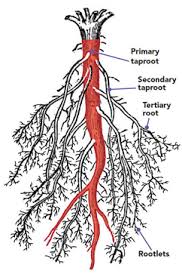
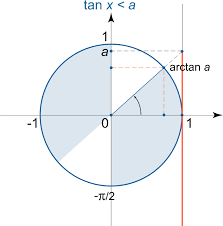
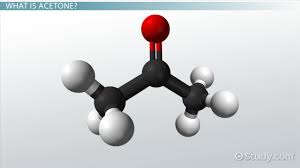
**Sample Details**

* The Problem Statement Describes About a Textbook Contains Paragraphs, Diagrams, Single line Questions
* So We took This Paragraph (Paragraph\_Image), Diagrams (Object\_Image), Single line Questions (Text\_Image) as a Samples

**Object Image**

In a Textbook We Consider Diagrams or Pictures are helpful to Classify (Object\_Image)

1)Diagrams from Physics, Chemistry



**Paragraph Image**

In a Textbook We See ‘Theory Part’ ,So I took Some Sample Images from Below

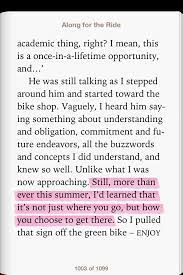
1)Letters

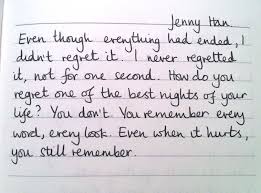
2)Newspapers

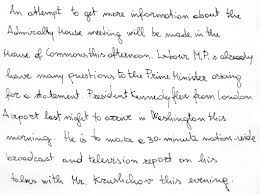
3)Certificate’s

4)Poem’s

We Can Identify different Patters Which are Helpful to Classify the Paragraph Image

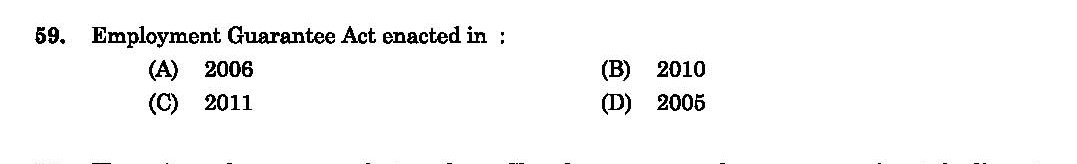


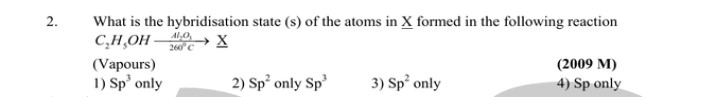


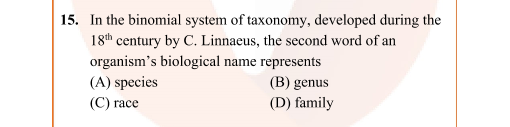


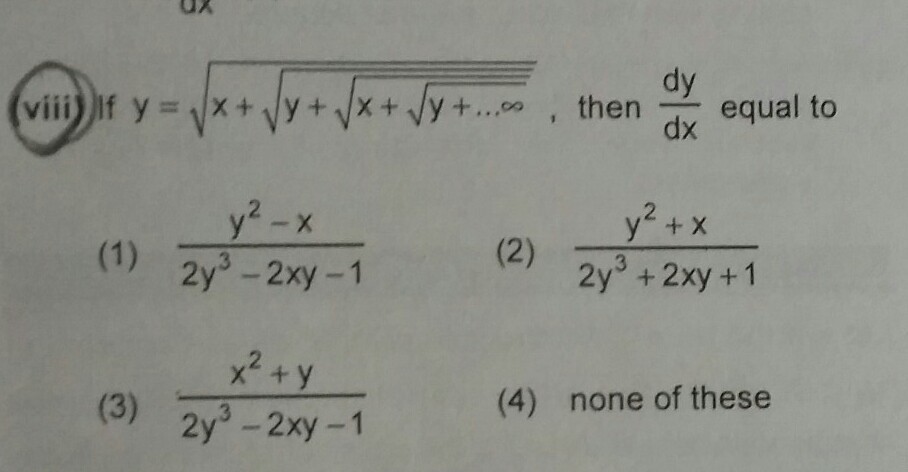
**Text Image**

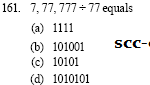
1)Single Type Questions With Options (Maths, Physics, Chemistry)

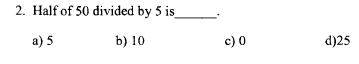












D:\Data Science\Obj_Text_Para\train\Text Image\t26.jpg

**Splitting The Samples(70:15:15)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Object Image | Paragraph Image | Text Image |
| Train | 353 | 353 | 353 |

Total=1059

|  |  |  |  |
| --- | --- | --- | --- |
|  | Object Image | Paragraph Image | Text Image |
| Valid | 73 | 73 | 73 |

Total=219

|  |  |  |  |
| --- | --- | --- | --- |
|  | Object Image | Paragraph Image | Text Image |
| Test | 73 | 73 | 73 |

Total=219

**Data Augmentation:**

I used Image Rotation as Data Augmentation Because, End User can take a picture from a Phone by landscape (or) Portrait Mode.

**Image Rotation**

**Syntax**

***Syntax:****cv2.cv.rotate( src, rotateCode[, dst] )*

***Parameters:******src:****It is the image whose color space is to be changed.****rotateCode:****It is an enum to specify how to rotate the array.****dst:****It is the output image of the same size and depth as src image. It is an optional parameter.*

img1 = cv2.rotate(img, cv2.cv2.ROTATE\_90\_COUNTERCLOCKWISE)

img1=cv2.resize(img1,(300,300))

**Sharpness**

***Syntax:***filter2D (src, dst, ddepth, kernel)

**Parameters**

**Src** – The source image to apply the filter on.

**Dst** – Name of the output image after applying the filter

**Ddepth** – Depth of the output image [ -1 will give the output image depth as same as the input image]

**Kernel** – The 2d matrix we want the image to convolve with.

**Example**

kernel = np.array([[0,-1,0], [-1,5,-1], [0,-1,-0]])<----Creating a Sharpness Matrix

g=cv2.imread('/content/02.jpg')<----Reading the Image

cv2.filter2D(g, -1, kernel)<-----Applying Kernel to a Image

**After Data Augmentation**

Train Sample Size=3177

Validation Sample Size=657

Test Sample Size=657

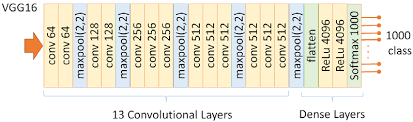
**Model 1 Evaluation:**

**VGG16**

1)VGG16 is **a** convolutional neural network trained on a subset of the ImageNet dataset, a collection of over 14 million images belonging to 22,000 categories.

2) The default input size for this model is 224x224

3**)**VGG16 Contains convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2**.**



tf.keras.applications.VGG16(

include\_top=False<---- Don’t include VGG16 Dense Layers

input\_shape=(300,300),<------Input Image Shape

classes=3<----Total Number of Classes

)

for layers in vgg.layers:#trainable parameters

layers.trainable=False<--- we are using non-trainable parameters so weights are not randomly initialized

**Model Summary**

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

Layer (type) Output Shape Param #

=================================================================

input\_1 (InputLayer) [(None, 300, 300, 3)] 0

block1\_conv1 (Conv2D) (None, 300, 300, 64) 1792

block1\_conv2 (Conv2D) (None, 300, 300, 64) 36928

block1\_pool (MaxPooling2D) (None, 150, 150, 64) 0

block2\_conv1 (Conv2D) (None, 150, 150, 128) 73856

block2\_conv2 (Conv2D) (None, 150, 150, 128) 147584

block2\_pool (MaxPooling2D) (None, 75, 75, 128) 0

block3\_conv1 (Conv2D) (None, 75, 75, 256) 295168

block3\_conv2 (Conv2D) (None, 75, 75, 256) 590080

block3\_conv3 (Conv2D) (None, 75, 75, 256) 590080

block3\_pool (MaxPooling2D) (None, 37, 37, 256) 0

block4\_conv1 (Conv2D) (None, 37, 37, 512) 1180160

block4\_conv2 (Conv2D) (None, 37, 37, 512) 2359808

block4\_conv3 (Conv2D) (None, 37, 37, 512) 2359808

block4\_pool (MaxPooling2D) (None, 18, 18, 512) 0

block5\_conv1 (Conv2D) (None, 18, 18, 512) 2359808

block5\_conv2 (Conv2D) (None, 18, 18, 512) 2359808

block5\_conv3 (Conv2D) (None, 18, 18, 512) 2359808

block5\_pool (MaxPooling2D) (None, 9, 9, 512) 0

flatten (Flatten) (None, 41472) 0

dense (Dense) (None, 3) 124419

=================================================================

Total params: 14,839,107

Trainable params: 124,419

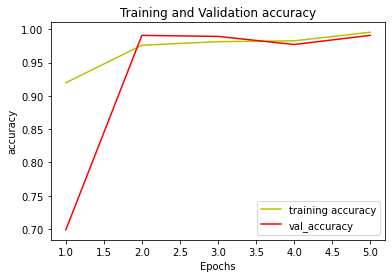
Non-trainable params: 14,714,688

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

I Used VGG Parameters because there are pre-trained on so many images ,So It may helpful to resolve the Problem Statement

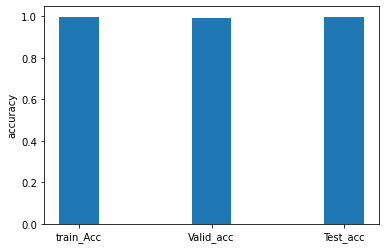
**Model Evaluation:**

Training and Validation Accuracy



I fit the Model for (5 Epochs,batch\_size=16) I got both Validation & Training accuracy approximately 99%

**Train & Validation & Test Accuracy**

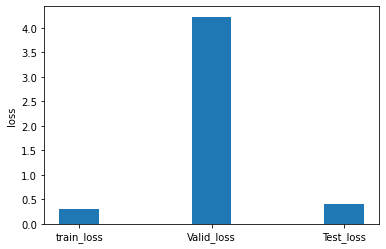


Train = 99.75%

Validation = 99.09%

Test = 99.54

**Train & Validation & Test Loss**



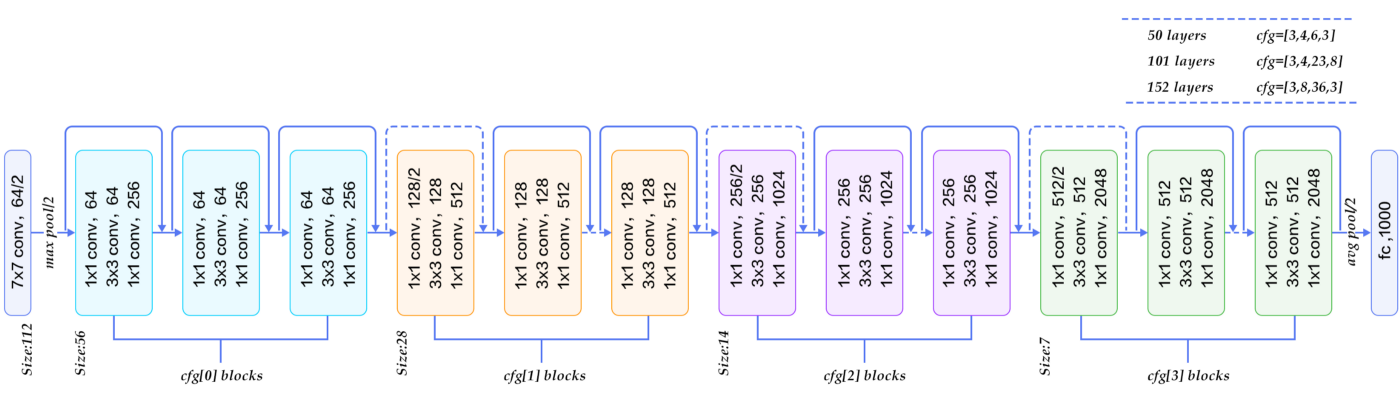
Train = 0.307

Validation = 4.226

Test = 0.41

**Model 2 Evaluation**

**RESNET50**



ResNet50=ResNet50(include\_top=False,classes=3,input\_shape=(300,300,3))#clone the VGG16 Model

for layers in ResNet50.layers:#trainable parameters

layers.trainable=False

ResNet50.summary()

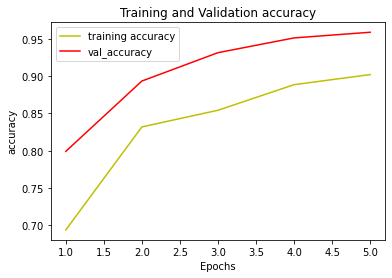
x=Flatten()(ResNet50.output)

Preduction=Dense(3,activation='softmax',)(x)

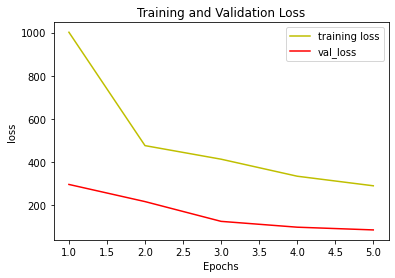
model=Model(inputs=ResNet50.input,outputs=Preduction)

I fit the Model for (5 Epochs,batch\_size=16) I got both Training accuracy 90% and Validation Accuracy =95% ,Training loss =291 and Validation loss=86

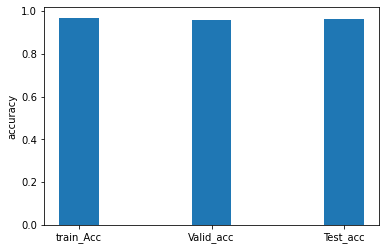
**Train & Validation & Test Accuracy**



**Train & Validation & Test Loss**



**Train & Validation & Test Accuracy**

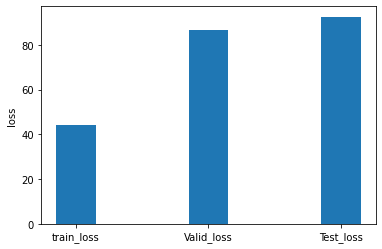


Train = 96.79%

Validation = 95.89

Test = 96.04

**Train & Validation & Test loss**



Train = 44.271

Validation = 86.76

Test = 92.54

From the Above Models, VGG16 Giving best accuracy both in Train and Test Samples