**Problem Statement :- Build CNN Model for Classification Of Flowers**

**Perform Below Tasks to complete the assignment:-**

**● Image Augmentation**

**● Create Model**

**● Add Layers (Convolution,MaxPooling,Flatten,Dense-(Hidden**

**Layers),Output)**

**● Compile The Model**

**● Fit The Model**

**● Save The Model**

**● Test The Model**

**Image Augmentation:**

Image segmentation is a commonly used technique in digital image processing and **analysis to partition an image into multiple parts or regions,** oftenbased on the characteristics of the pixels in the image.

**Create Model:**

1. Clone Mask R-CNN.
2. Library Dependencies.
3. Pre Trained Weights.
4. Make a new Jupyter Notebook.
5. Importing the Necessary Libraries.
6. The path for pretrained weights.
7. Inference class to infer the Mask R-CNN Model.
8. Loading the Weights.

**Add Layers (Convolution,MaxPooling,Flatten,Dense-(Hidden Layers),Output):**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35** | **# example of adding layers**  **from numpy import asarray**  **from keras.models import Sequential**  **from keras.layers import Conv2D**  **from keras.layers import AveragePooling2D**  **# define input data**  **data = [[0, 0, 0, 1, 1, 0, 0, 0],**  **[0, 0, 0, 1, 1, 0, 0, 0],**  **[0, 0, 0, 1, 1, 0, 0, 0],**  **[0, 0, 0, 1, 1, 0, 0, 0],**  **[0, 0, 0, 1, 1, 0, 0, 0],**  **[0, 0, 0, 1, 1, 0, 0, 0],**  **[0, 0, 0, 1, 1, 0, 0, 0],**  **[0, 0, 0, 1, 1, 0, 0, 0]]**  **data = asarray(data)**  **data = data.reshape(1, 8, 8, 1)**  **# create model**  **model = Sequential()**  **model.add(Conv2D(1, (3,3), activation='relu', input\_shape=(8, 8, 1)))**  **model.add(AveragePooling2D())**  **# summarize model**  **model.summary()**  **# define a vertical line detector**  **detector = [[[[0]],[[1]],[[0]]],**  **[[[0]],[[1]],[[0]]],**  **[[[0]],[[1]],[[0]]]]**  **weights = [asarray(detector), asarray([0.0])]**  **# store the weights in the model**  **model.set\_weights(weights)**  **# apply filter to input data**  **yhat = model.predict(data)**  **# enumerate rows**  **for r in range(yhat.shape[1]):**  **# print each column in the row**  **print([yhat[0,r,c,0] for c in range(yhat.shape[2])])**  **Compile The Model:**  **from** **segmentation\_models** **import** Unet  **from** **segmentation\_models** **import** get\_preprocessing  **from** **segmentation\_models.losses** **import** bce\_jaccard\_loss  **from** **segmentation\_models.metrics** **import** iou\_score  BACKBONE = 'resnet34'  preprocess\_input = get\_preprocessing(BACKBONE)  *# load your data*  x\_train, y\_train, x\_val, y\_val = load\_data(...)  *# preprocess input*  x\_train = preprocess\_input(x\_train)  x\_val = preprocess\_input(x\_val)  *# define model*  model = Unet(BACKBONE, encoder\_weights='imagenet')  model.compile('Adam', loss=bce\_jaccard\_loss, metrics=[iou\_score])  *# fit model*  model.fit(  x=x\_train,  y=y\_train,  batch\_size=16,  epochs=100,  validation\_data=(x\_val, y\_val),  )  **Fit The Model:**  def unet\_model(output\_channels:int):   inputs = tf.keras.layers.Input(shape=[128, 128, 3])    # Downsampling through the model   skips = down\_stack(inputs)   x = skips[-1]   skips = reversed(skips[:-1])    # Upsampling and establishing the skip connections   for up, skip in zip(up\_stack, skips):     x = up(x)     concat = tf.keras.layers.Concatenate()     x = concat([x, skip])    # This is the last layer of the model   last = tf.keras.layers.Conv2DTranspose(       filters=output\_channels, kernel\_size=3, strides=2,       padding='same')  #64x64 -> 128x128    x = last(x)    return tf.keras.Model(inputs=inputs, outputs=x)  **Save The Model:**   |  | | --- | | **import pixellib** | |  | **from pixellib.semantic import semantic\_segmentation** | |  | **import cv2** | |  |  | |  | **segment\_image = semantic\_segmentation()** | |  | **segment\_image.load\_pascalvoc\_model("pascal.h5")** | |  | **output, segmap = segment\_image.segmentAsPascalvoc("sample1.jpg")** | |  | **cv2.imwrite("img.jpg", output)** | |  | **print(output.shape)** | |

**Test The Model:**

1. **Pixel accuracy:**



# Jaccard’s Index (Intersection over Union, IoU)

# ****3. Dice Coefficient****