Week-13

Aim: To analyze and understand the architecture and working Principle of Pretrained deep learning model

Algorithm:

* import a Pretrained model from keras applications.
* Load model with Pretrained weights.
* Display model summary
* Analyze layer Types
* Identifying trainable vs non trainable Parameters
* Optionally visualize feature maps

Pseudocode:

Import Tensorflow

Load a PreTrained model

→ Include weights = "imagenet"

→ exclude top layer of using for feature extrac-tion

Display model summary

→ Print layer names, types, output shapes and Parameter counts

For each layer in model:

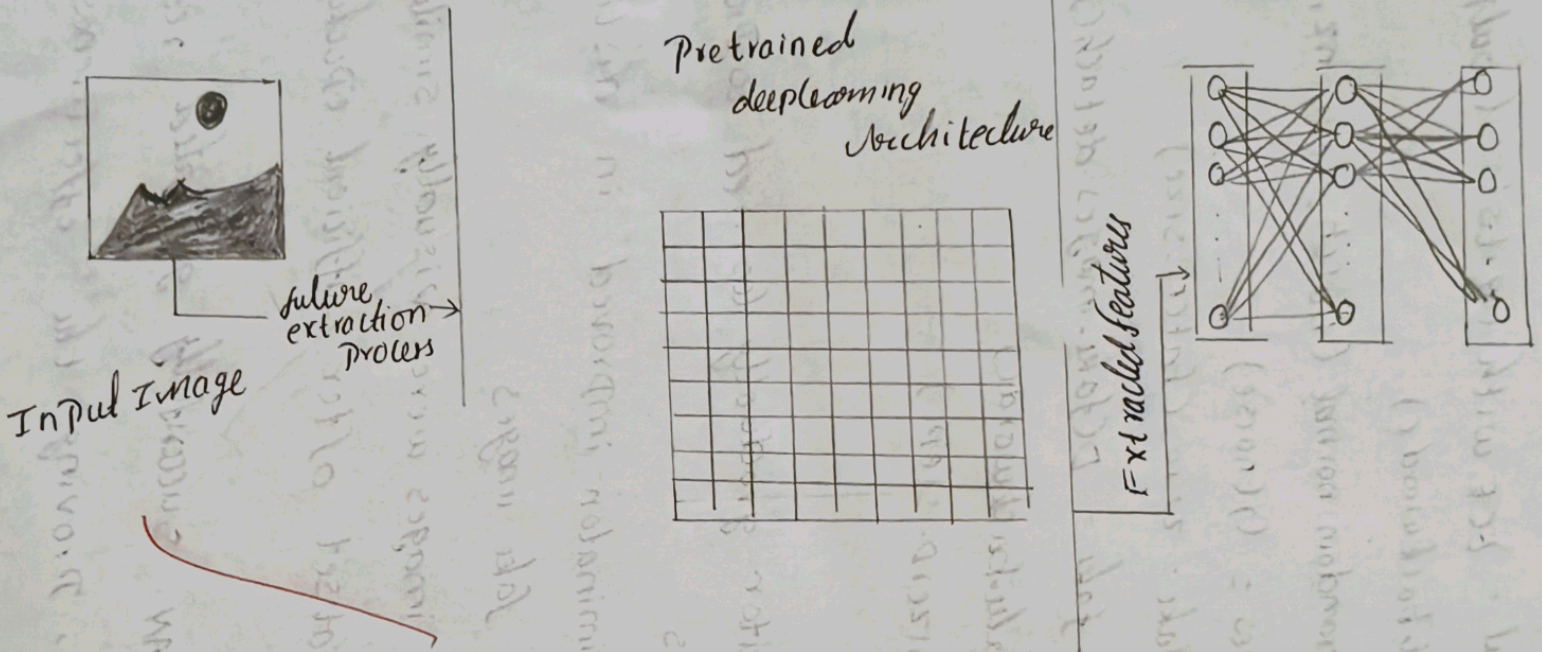
→ Identify type

→ Note Activation

Train the model

Freeze lower layers if needed

Usage of Pre Trained Architecture



observation:

* The Pretrained model has convolutional, Pooling and fully connected layer.
* conv layers extract failures, Pooling layers reduce size
* Pre trained weights allow faster traing
* sample images are correctly classified

Result:

The Architecture of Pretrained CNN's was successfully analyzed and visualized.

Output:

Total Parameters: 138357544

Trainable Parameters: 138357544

Non Trainable Parameters: 0

conv 01 : conv02D

relu : ReLU

maxpool : MaxPool 2D

input layer : (None, 224, 224, 3)

conv2D (None, 224, 224, 64)

MaxPooling2D (None, 56, 56, 128)

Flatten (None, 25088)

Dense (None, 4096)

```
# Experiment 13: Understanding a Pre-trained CNN model architecture
from tensorflow.keras.applications import VGG16

# Load VGG16 with pre-trained weights
model = VGG16(weights='imagenet')

# Display architecture summary
model.summary()
```

```python
# Experiment 13: Understanding a Pre-trained CNN model architecture
from tensorflow.keras.applications import VGG16

# Load VGG16 with pre-trained weights
model = VGG16(weights='imagenet')

# Display architecture summary
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/v
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
Model: "vgg16"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808

block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102,764,544
fc2 (Dense)	(None, 4096)	16,781,312
predictions (Dense)	(None, 1000)	4,097,000

Total params: 138,357,544 (527.79 MB)
Trainable params: 138,357,544 (527.79 MB)
Non-trainable params: 0 (0.00 B)