

13	27/10/25	Architecture of pre-trained models	
14	27/10/25	pre-trained CNN model as a feature Extractor using transfer learning	Eq. 1
15	27/10/25	Yolo model to Detect the objects.	Eq. 2



27/10/25

# Understanding the Architecture of Lab 13: Pre-trained Models.

Aim:

To study and analyse the architecture of pre-trained CNN models such as VGG16, ResNet101, and InceptionV3.

Objectives:

- To understand layer configuration of pre-trained CNN models.
- To visualize input, hidden and output layers.
- To interpret the feature extraction process of pre-trained models.

Observation:

- ~~pre-trained~~ models are trained on large dataset like ImageNet.
- Layers closer to input extract general features (edges, textures) while deeper layers capture more abstract features (shape, objects).
- Model Summary provides the total parameters and layer structure.



output.

```
model = Sequential(  
    (1) Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1),  
        padding=(1, 1))
```

```
    (2) ReLU(inplace=True)
```

```
    (3) Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),  
        padding=(1, 1))
```

```
    (4) ReLU(inplace=True)
```

```
    (5) MaxPool2d(kernel_size=2, stride=2, padding=0,  
        dilation=1, ceil_mode=False)
```

```
    (6) MaxPool2d(kernel_size=2, stride=2,  
        padding=0, dilation=1, ceil_mode=False)
```

```
    (7) AdaptiveAvgPool2d(output_size=(1, 1))
```

```
    (8) Linear
```

```
    (9) ReLU(inplace=True)
```

```
    (10) Dropout(p=0.5, inplace=False)
```

```
    (11)
```

Pseudocode:

1) Import library

2) Load a pre-trained model

3) Display model summary using  
model.summary()

4) Visualize architecture using  
plot\_model()

5) Print layers names and output shapes,  
output

Feature Extractor layer

Sequential(  
 (0) Conv2d

(1) Conv2d

(2) MaxPool2d

(3) MaxPool2d

Classification layer

Sequential(  
 (4) Linear

(5) Linear (1, features=256, activation=softmax, bias=True)

(6) ReLU(inplace=True)

(7) Dropout(p=0.5, inplace=False)

(8)

Total trainable parameters: 136352544

Result:

Successfully implemented the  
Architecture of pre-trained model.

!pip install torch torchvision matplotlib

```
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.6.0+cu126)
Requirement already satisfied: torchvision in /usr/local/lib/python3.12/dist-packages (0.23.0+cu126)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from torch) (3.20.0)
Requirement already satisfied: typing-extensions<=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (78.2.0)
Requirement already satisfied: sympy<=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-toolkit-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.5.7.77 in /usr/local/lib/python3.12/dist-packages (from torch) (10.5.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch) (11.7.1.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.20.3 in /usr/local/lib/python3.12/dist-packages (from torch) (2.20.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-mitl-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.85)
Requirement already satisfied: nvidia-oxrt-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packages (from torch) (3.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from torchvision) (2.0.2)
Requirement already satisfied: pillow==10.4.0 in /usr/local/lib/python3.12/dist-packages (from torchvision) (10.4.0)
Requirement already satisfied: torchvision==0.23.0 in /usr/local/lib/python3.12/dist-packages (from torchvision) (0.23.0)
Requirement already satisfied: cytoolz==0.12.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools==4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.22.0)
Requirement already satisfied: kiwisolver==1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: packaging==20.9 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (20.9)
Requirement already satisfied: pyparsing==3.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.0.1)
Requirement already satisfied: python-dateutil==2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.7)
Requirement already satisfied: six==1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil==2.7->matplotlib) (1.17.0)
Requirement already satisfied: sphinx==4.7.1 in /usr/local/lib/python3.12/dist-packages (from sphinx==4.7.1->matplotlib) (4.7.1)
Requirement already satisfied: MarkupSafe==2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch) (2.0.2)
```

```
# lab13_pretrained_model_architecture.py
import torch
import torchvision.models as models

# Load a pretrained model (VGG16)
model = models.vgg16(pretrained=True)

# Print model architecture
print("===== VGG16 Pre-trained Model Architecture =====")
print(model)

# Print only feature extractor part
print("===== Feature Extractor Layers =====")
print(model.features)

# Print classifier part
print("===== Classifier Layers =====")
print(model.classifier)

# Print total trainable parameters
total_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print(f"Total trainable parameters: {total_params}")
```

/usr/local/lib/python3.12/dist-packages/torchvision/models/\_utils.py:188: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.

/usr/local/lib/python3.12/dist-packages/torchvision/models/\_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is equivalent to passing 'weights=VGG16\_Weights.DEFAULT'. You can also use 'weights=VGG16\_Weights.DEFAULT' to get the most up-to-date weights.

Downloading: "https://download.pytorch.org/models/vgg16-397923af.pth" to /root/.cache/torch/hub/checkpoints/vgg16-397923af.pth

200% |#####| 520M/520M [00:03<00:00, 149MB/s]

===== VGG16 Pre-trained Model Architecture =====

VGG16

Features: Sequential(

(0): Conv2d(3, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(1): ReLU(inplace=True)

(2): Conv2d(64, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))

(3): ReLU(inplace=True)



```
# Print total trainable parameters
total_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print(f"Total Trainable Parameters: {total_params}")

#####
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is equivalent to passing 'weights=WGS16_Weights.DEFAULT'. You can also use 'weights=WGS16_Weights.DEFAULT' to get the most up-to-date weights.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/wgs16-307020af.pth" to /root/.cache/torch/hub/checkpoints/wgs16-307020af.pth
0%|          | 528M/528M [00:00<00:00, 140MB/s]
##### WGS16 Pre-trained Model Architecture #####
VGG
(Features): Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (8): ReLU(inplace=True)
  (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (11): ReLU(inplace=True)
  (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (13): ReLU(inplace=True)
  (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (15): ReLU(inplace=True)
  (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (18): ReLU(inplace=True)
  (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (20): ReLU(inplace=True)
  (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (22): ReLU(inplace=True)
  (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (25): ReLU(inplace=True)
  (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (27): ReLU(inplace=True)
  (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (29): ReLU(inplace=True)
  (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(pool): AdaptiveAvgPool2d(output_size=(7, 7))
(classifier): Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace=True)
  (5): Dropout(p=0.5, inplace=False)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
)

##### Feature Extractor Layers #####
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
```

```
11 # print total trainable parameters
total_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print(f"Total Trainable Parameters: {total_params}")

(Classifier): Sequential(
  (0): Linear(in_features=25600, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace=True)
  (5): Dropout(p=0.5, inplace=False)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
)

==== Feature Extractor Layers ====
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (8): ReLU(inplace=True)
  (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (11): ReLU(inplace=True)
  (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (13): ReLU(inplace=True)
  (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (15): ReLU(inplace=True)
  (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (18): ReLU(inplace=True)
  (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (20): ReLU(inplace=True)
  (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (22): ReLU(inplace=True)
  (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (25): ReLU(inplace=True)
  (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (27): ReLU(inplace=True)
  (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (29): ReLU(inplace=True)
  (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)

==== Classifier Layers ====
Sequential(
  (0): Linear(in_features=25600, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace=True)
  (5): Dropout(p=0.5, inplace=False)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
)

Total Trainable Parameters: 138257544
```