Exp 14

Implement a Pre-trained CNN model as a Feature Extractor using transfer learning.

Aim:

To use a pre-trained CNN model for extracting features and training a new classifier.

Objective

1. To load a pre-trained model like ResNet50 or VGG16
2. To freeze convolutional layers for feature extraction
3. To add custom dense layers for classification
4. To fine-tune the model for new dataset
5. To evaluate transfer learning performance

Pseudo code:

1. Load pre-trained CNN without top layers.
2. Freeze base model weights.
3. Add custom dense and output layers.
4. Compile and train model on new dataset
5. Evaluate and test classification accuracy

Result

VGG16 Architecture

Input 224x224

↓

Conv 3x3 + Relu 64
Conv 3x3 + Relu 64
pooling + Max 64

224 x 224

↓

3x3 Conv + Relu 128
3x3 Conv + Relu 128
2x2 Max Pool 128

112 x 112

↓

3x3 Conv + Relu 256
3x3 Conv + Relu 256
3x3 Conv + Relu 256
2x2 Max Pool 256

~~56x56~~ 56x56

↓

3x3 Conv + ~~+~~ Relu 512
3x3 Conv + Relu 512
3x3 Conv + Relu 512
2x2 Max Pool 512

~~14x14~~ 28x28

↓

3x3 Conv + Relu 512
3x3 Conv + Relu 512
3x3 Conv + Relu 512
2x2 - Max pool 512

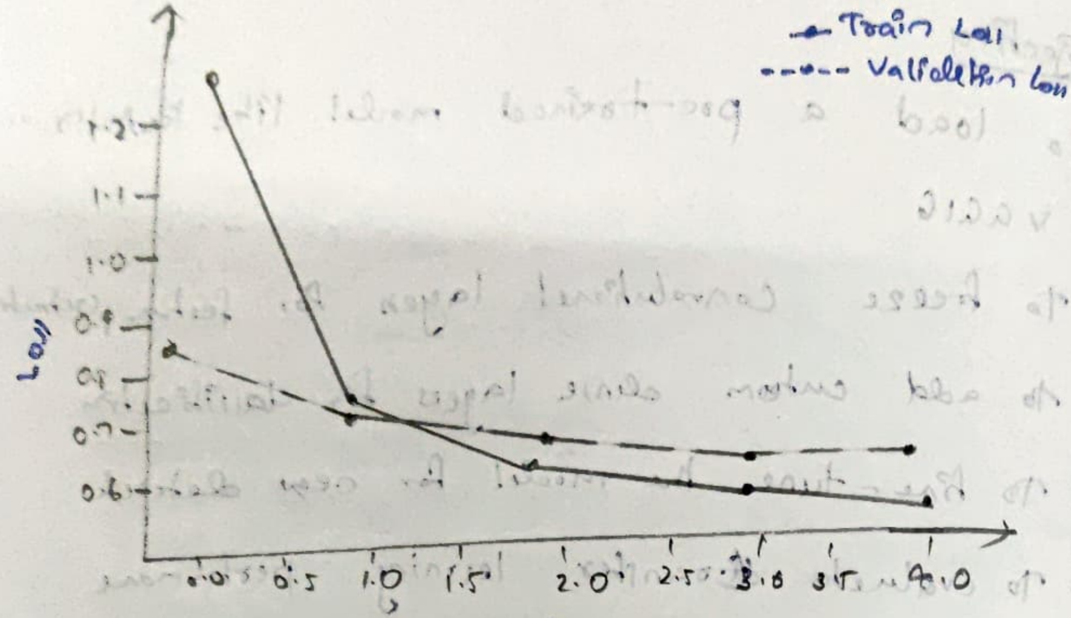
14 x 14

↓

FC 4096
FC 4096
FC Output layer

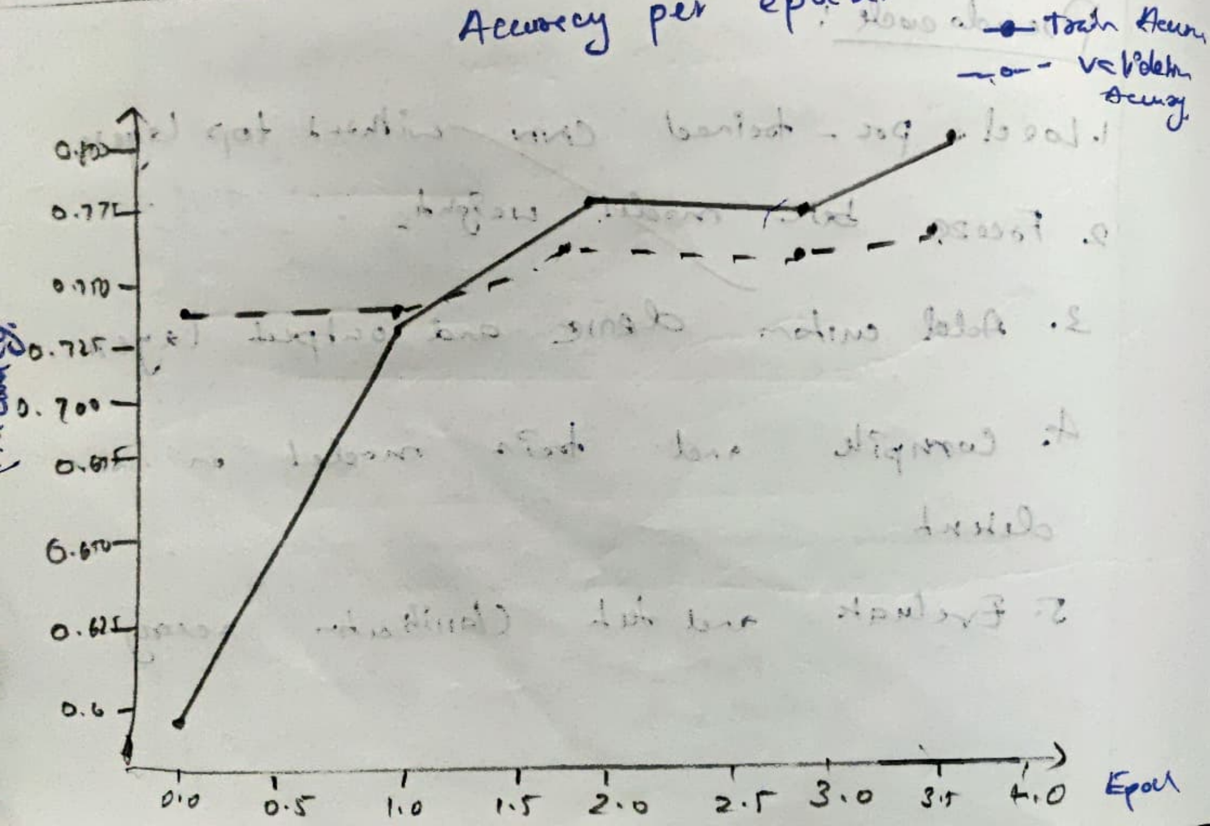
7x7
→ output

Epoch	Training Accuracy	Validation Accuracy
1	92.3%	70.9%
5	85.5%	82.8%
10	90.6%	88.9%
15	92.4%	90.5%



Loss per epoch

Accuracy per epoch.



Observation

1. Transfer learning reduces computation time
2. Pre-trained layers extract robust features
3. Model perform well with limited data.
4. Fine-tuning improves specific class recognition
5. High accuracy achieved with fewer epochs

Result :

Transfer learning model achieved strong performance as a feature extractor and classifier

Exp II:

Implement a YOLO Model to Detect Object.

Aim:

To Implement YOLO model for real time object detection

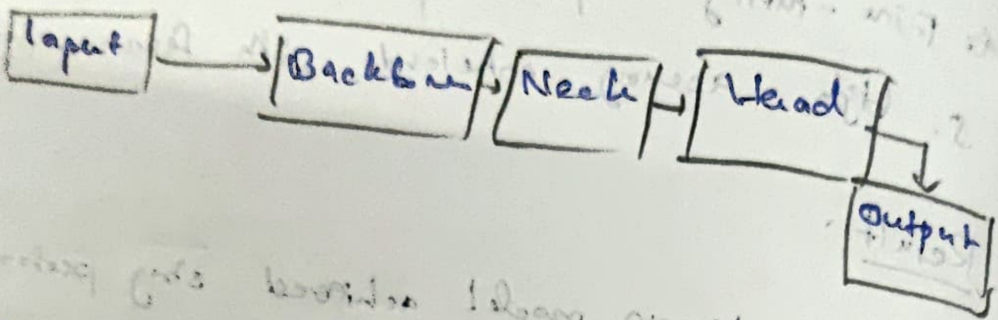
Objectives:

1. To understand YOLO architecture and working mechanism.
2. To load pre-trained YOLO weights and configuration.
3. To preprocess input image and perform detection
4. To extract bounding boxes and class labels.
5. To visualize detected object with confidence score

Pseudocode:

1. Load YOLO config, weights, and class labels.
2. Read input image and create image blob.
3. Perform forward pass through YOLO network
4. Extract and filter detections based on confidence
5. Draw bounding boxes and display detected objects.

YOLO v8 :



Backbone : extracts essential visual features from input using convolutional and CSP layers

Neck : Connect backbone and head, combining multi scale features using structure like PAN or FPN

Head : Predict bounding box, object class, and confidence score from the processed fea

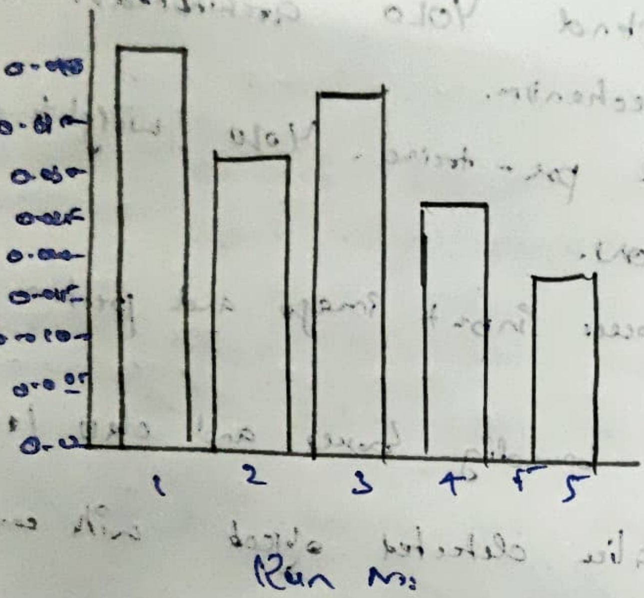
Observation :

1. YOLO detects multiple objects in real-time
2. Accuracy depends on chosen confidence threshold
3. Detection is faster with GPU support
4. Bounding boxes accurately locate objects.
5. Works effectively ~~across~~ across different object classes

Result

YOLO model ~~successfully~~ to successfully detected and labelled multiple objects with high accuracy in images and video streams

Object	Confidence(%)	Bounding Box (x,y,
Person	94	(320,185, 100,220)
Car	88	(320,150,150,110)
Dog.	91	(480,200,120,11



Detected image Confidence level