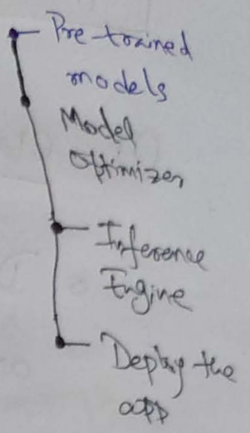
② LEVERAGING PRE-TRAINED MODELS



Open-VINO toolkit

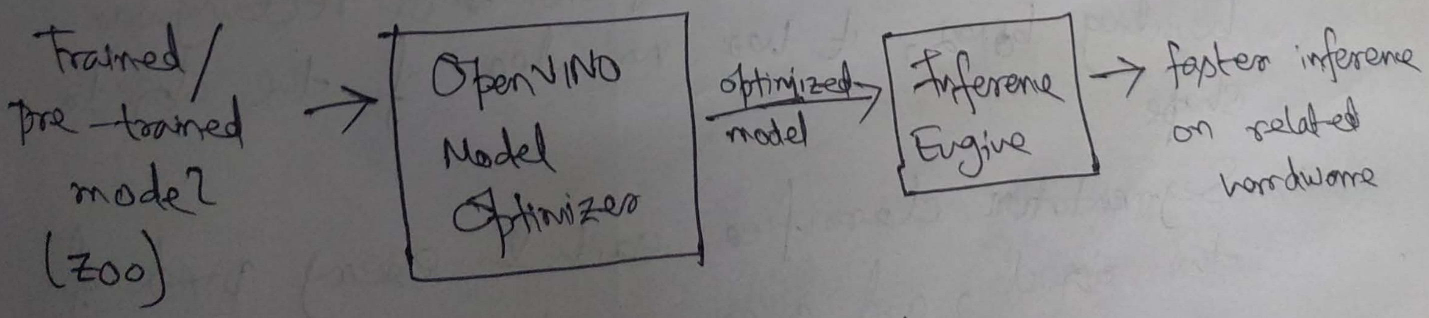
- Stands for

  Open Visual Inferencing & Neural Network Optimization

- It is an open source library useful for edge deployment due to its
  - performance maximizations
  - pre-trained models

- It ~~tel~~ helps support fast inference across ~~different~~ many Intel hardware w/ a common API.

- It enables running at the edge by optimizing for speed and size.


Trained/ pre-trained model (zoo)
OpenVINO Model Optimizer
optimized model
Inference Engine
faster inference on related hardware

Open VINO tool kit

Types of Computer Vision Models

① Classification

- Determines what class does ~~an~~ an object in an image belong to.
- These classifications can be associated w/ a probability to signify how confident the classificatio is.

② Detection

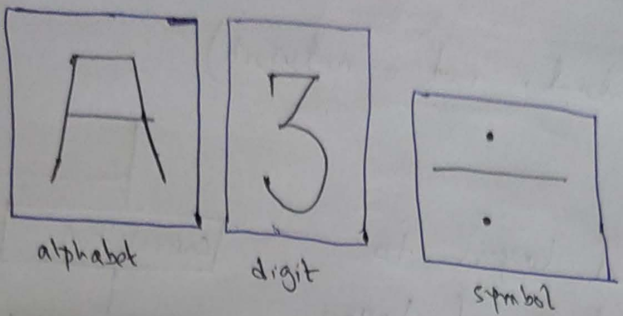
- Determines the presence and location of objects in an image.
- Often surround the objects w/ a bounding box

③ Segmentation

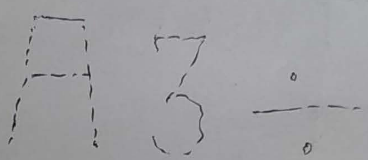
Segmentation < Semantic → all objects of same class are one
Instance → each object of a class is separate

- Since detection provides square ~~or~~ or rectangular bounding boxes, it has no idea of the object's shape.
- Segmentation classifies each & every pixel of the object, and gives granular understanding of the image.

Eg:



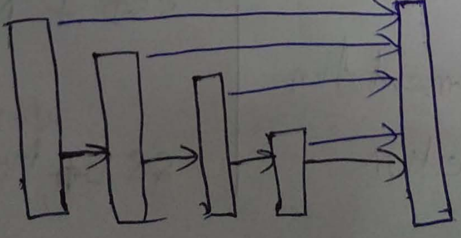
- ○ Classifications → alphabet, digit, symbol
- ○ Detections → shown in bounding boxes (□)
- ○ Segmentations →



Case Studies in CV

① Single Shot MultiBox Detector (SSD)

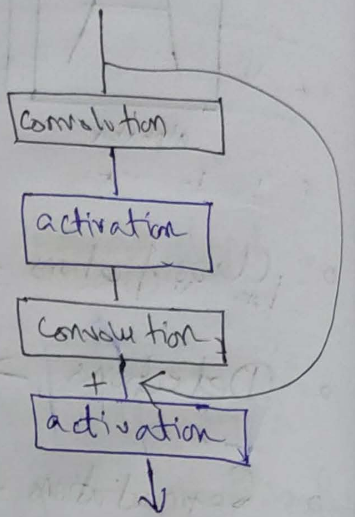
- Combines classifier and detector across images
- Uses default bounding boxes and features at different levels.
- It is basically an object detection network



② ResNet

(res = residual, net = network)

- Uses residual layers to "skip" over sections of layers.
- Avoids vanishing gradient problem w/ very deep neural networks.



③ MobileNet

- Uses layers like 1x1 convolutions.
- Cuts down computational complexity and network size.
- This leads to fast inference w/ out ~~sub~~ substantial decrease in accuracy.

Pre-trained Model Types

~~Age & gender~~	CV ~~rec~~ type	CV arch.
Age & gender	classification	
Pedestrian	detection	SSD + MobileNet
Advanced roadside	segmentation	
Face detection	detection	SSD, MobileNet,

Image Pre processing

Parameters

- Color channel (BGR order)
- Image resizing ~~(height~~ (width, height)
- Normalization

There may be more, like batch size, etc.

Pre-processing steps

(Needed for 2nd exercise)

1. Resize → cv2.resize (w, h)
2. Transpose → image.transpose ((ch3, ch1, ch2))

   where: ch3 = third channel (for BGR) = 2

   ch1 = first channel (for h) = 0

   ch2 = second channel (for w) = 1
3. Reshape → image.reshape $(\underset{\substack{\downarrow\\ \text{batch}\\ \text{size}}}{b}, \underset{\substack{\downarrow\\ \text{\# of}\\ \text{channels}}}{n}, \underset{\substack{\downarrow\\ \text{height}}}{h}, \underset{\substack{\searrow\\ \text{width}}}{w})$

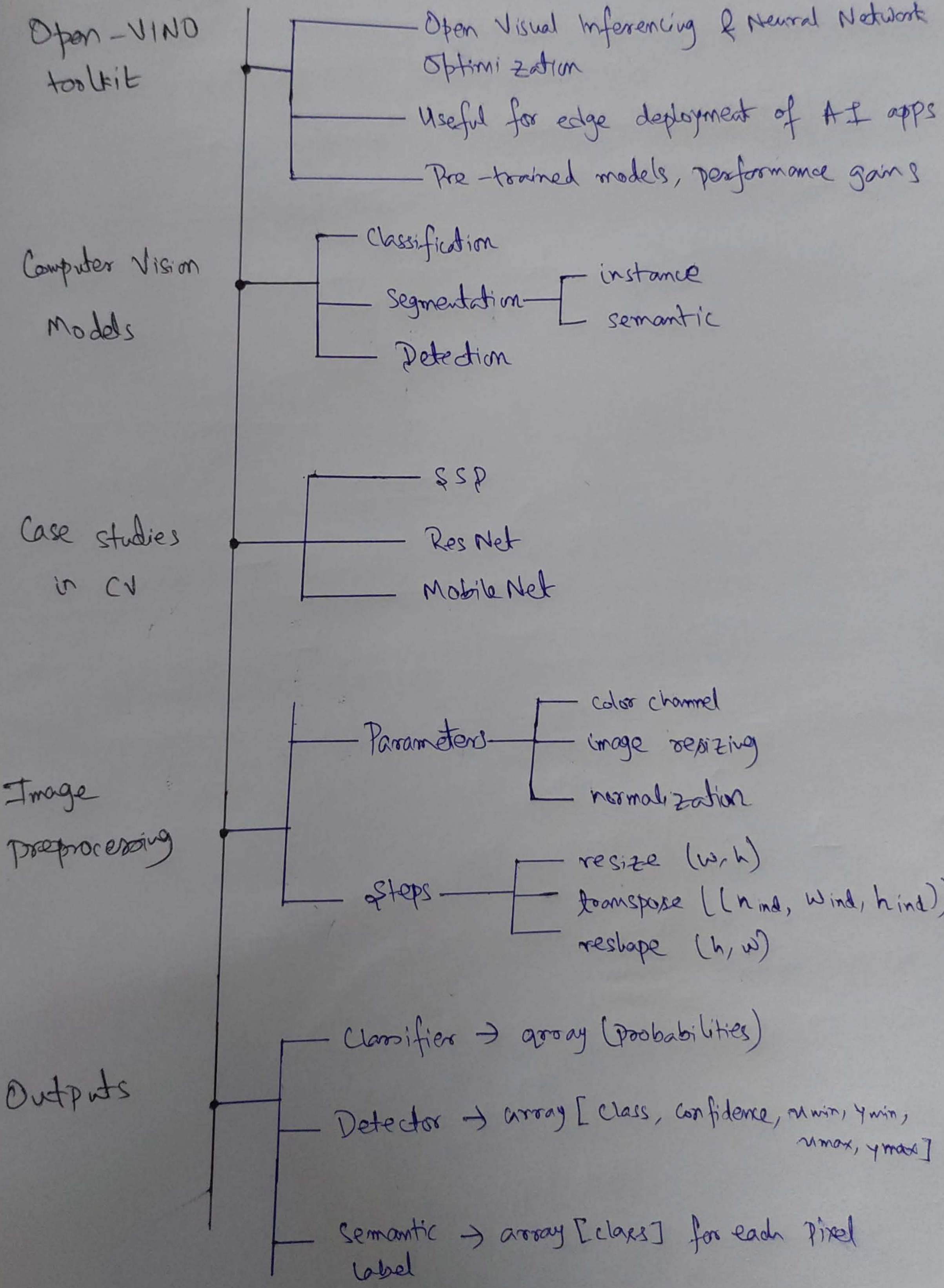
(Get (b, n, h, w) from docs)

$$\left.\begin{array}{ll} \text{i/p} & \text{Image shape: } (\overset{0}{\text{width}}, \overset{1}{\text{height}}, \overset{2}{3}) \\ \text{o/p} & \text{Image shape: } (\overset{2}{3}, \overset{0}{\text{width}}, \overset{1}{\text{height}}) \end{array}\right\} \text{for transpose}$$

Handling Network Outputs

CV Model	Output
i) Classification	• An array w/ softmax probabilities by class • argmax (probabilities) is used for prediction
ii) Bounding Box (Detection)	• Array: box [0..5], where: ◦ box [0] = class ◦ box [1] = threshold ◦ box [2], box [3] = x_{min}, y_{min} ◦ box [4], box [5] = x_{max}, y_{max}
iii) Semantic label	• An array w/ ~~class~~ class for each pixel

Summary



- Open-VINO toolkit
  - Open Visual Inferencing & Neural Network Optimization
  - Useful for edge deployment of AI apps
  - Pre-trained models, performance gains
- Computer Vision Models
  - Classification
  - Segmentation
    - instance
    - semantic
  - Detection
- Case studies in CV
  - SSP
  - ResNet
  - MobileNet
- Image preprocessing
  - Parameters
    - color channel
    - image resizing
    - normalization
  - Steps
    - resize (w, h)
    - transpose ((n ind, w ind, h ind))
    - reshape (h, w)
- Outputs
  - Classifier → array (probabilities)
  - Detector → array [class, confidence, xmin, ymin, xmax, ymax]
  - Semantic label → array [class] for each pixel