Computer Vision cheat sheet

Pixel:

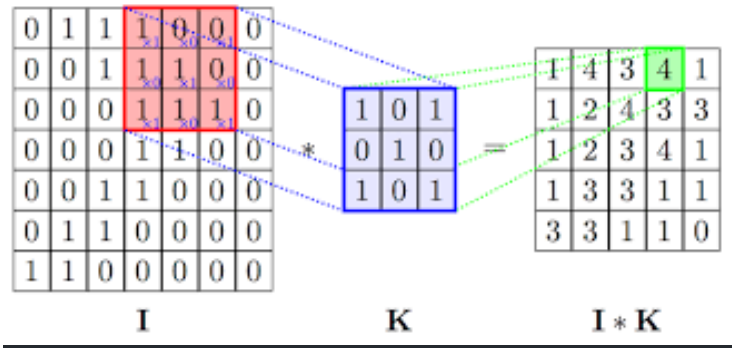
An image can be broken into pixels as its smallest fragment.

Filters:

filters are mainly used to suppress either the high frequencies in the image, i.e. smoothing the image, or the low frequencies, i.e. enhancing or detecting edges in the image.

Convolution:

Convolution helps to reduce computation by reducing the dimension of the image which can be given as an input to simple neural networks to perform image classification and helps to extract unique features/characteristics from an image by producing a unique number as per the image pixels.



Padding:

The purpose of padding is to preserve the original size of an image when applying a convolutional filter and enable the filter to perform full convolutions on the edge pixels.

Stride:

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on.

Pooling:

The main purpose of pooling layer is to progressively reduce the spatial size of the input image, so that number of computations in the network are reduced. Pooling performs downsampling by reducing the size and sends only the important data to next layers in CNN.

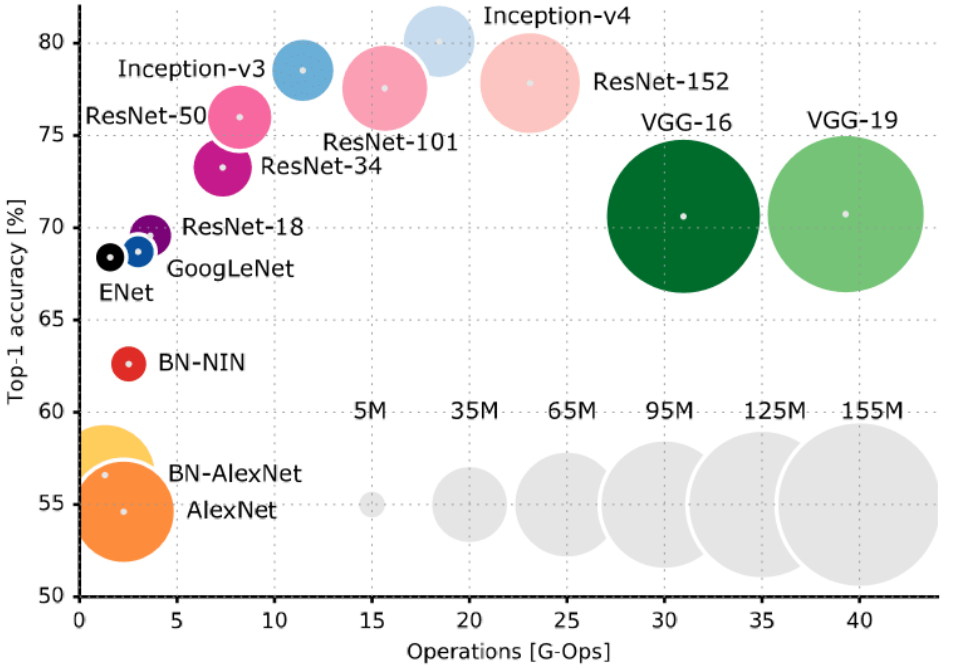
- Input: $(N, C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

Transfer Learning:

Transfer learning aims to leverage the learned knowledge from a previous task to help learning a new similar task. In CNN, instead of learning new weights from scratch, we try to utilize the learned weights from the previously trained architectures.



Object Detection:

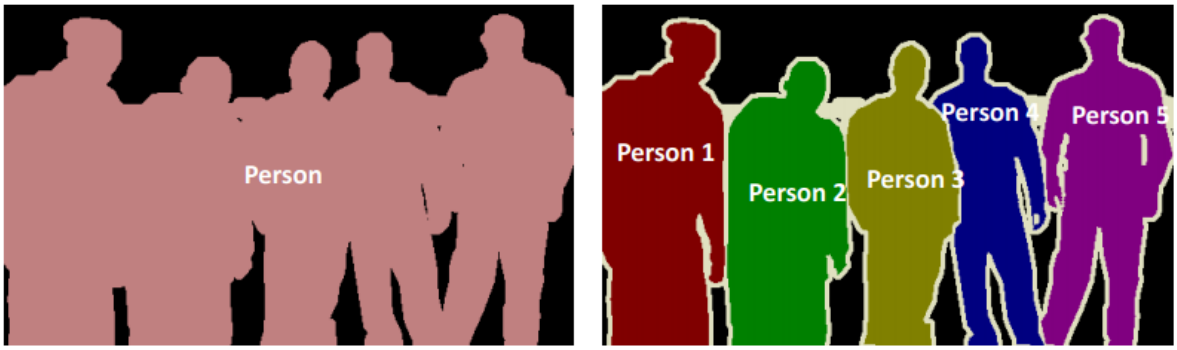
Given an image we want to learn the classes of the image and where they are located. We need to detect an object and a rectangle of where that object is located.

Semantic Segmentation:

associates every pixel of an image with a class label. Gives a fine boundary of objects. Multiple objects of the same class are considered as the same entity.

Instance Segmentation:

multiple objects of the same class are considered as a single entity. Gives the instances of each class.



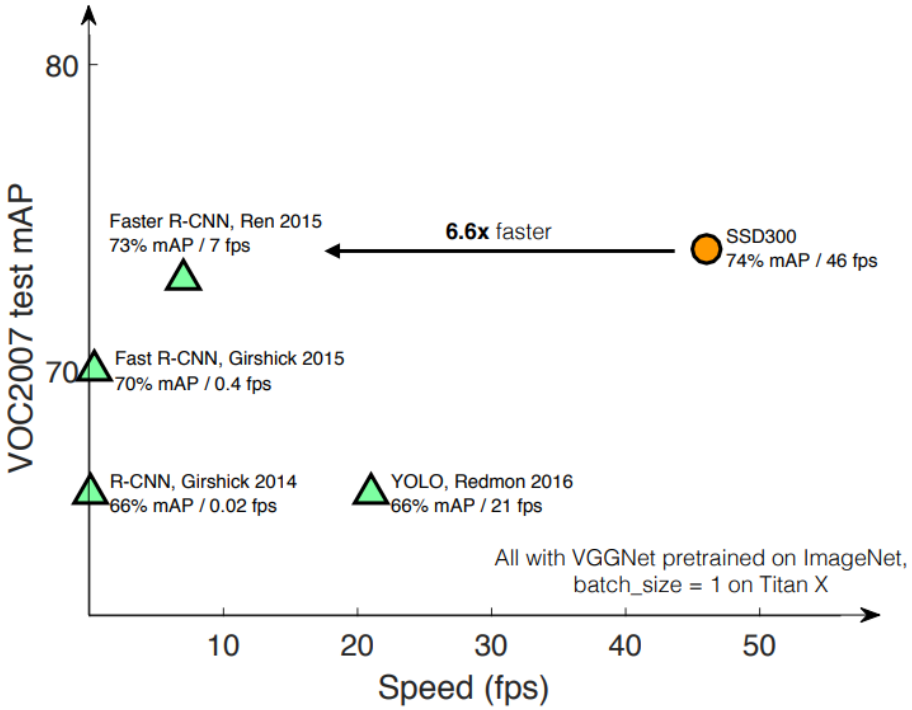
Semantic Segmentation **Instance Segmentation**

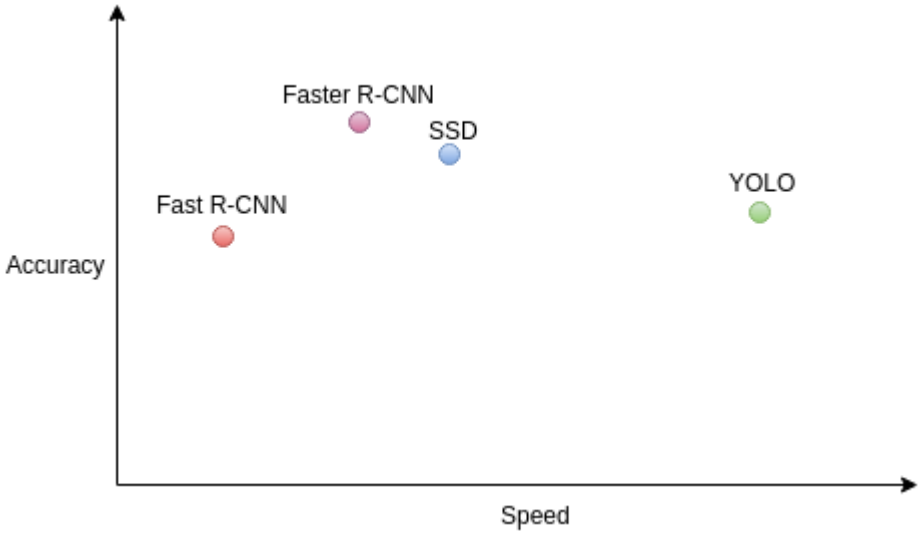
Approaches & Algorithms:

Approaches:- ● Object detection as regression ● Object detection as classification

Region Proposal Based Algorithms:- ● R-CNNs ● Fast R-CNNs ● Faster R-CNNs

Algorithms without Region Proposal:- ● Yolo ● SSD.



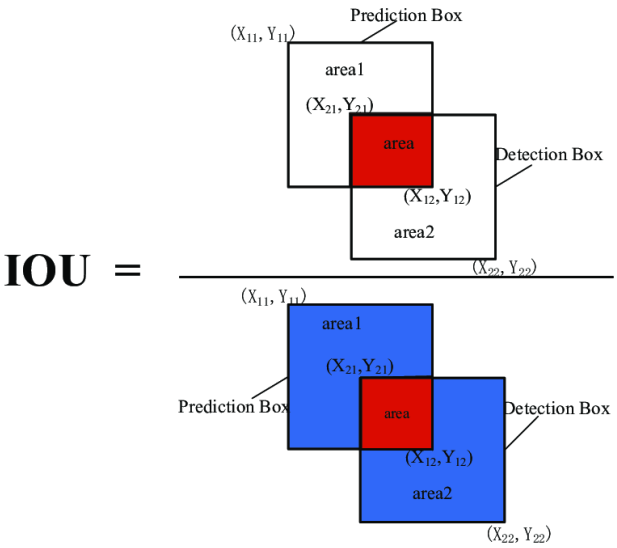


Evaluation Metrics:

- **Mean of Average Precision (mAP):**

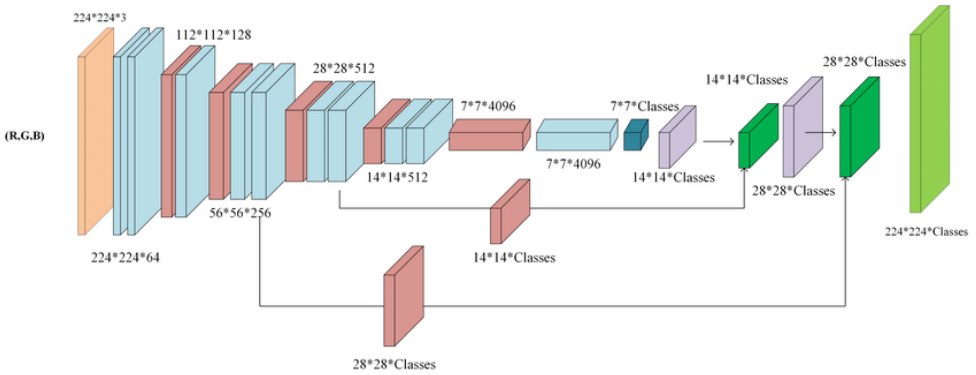
$$\text{mAP} = \frac{1}{|\text{classes}|} \sum_{c \in \text{classes}} \frac{|TP_c|}{|FP_c| + |FP_c|}$$

- **Intersection Over Union (IOU):**

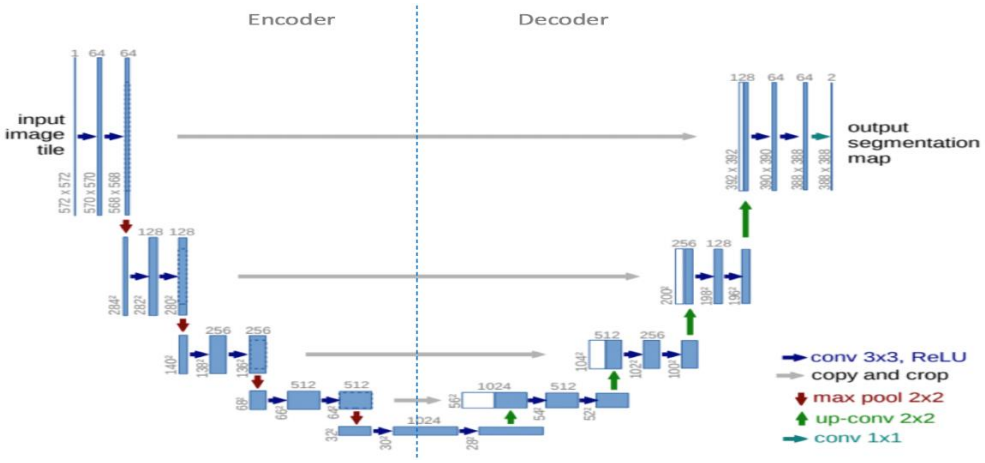


Approaches to solve Semantic Segmentation:

- **Fully Convolutional Network:**



- **Encoder-decoder (UNet):**



Comparison between FCN & U-Net:

FCN:

• It has only one layer in the decoder (Up-Samples only once)
• For upsampling it uses bilinear interpolation technique which has no learnable filter.
• Variants of FCN-[FCN 16s and FCN 8s] add the skip connections from lower layers to make the output robust to scale changes

U-Net:

• It has multiple upsampling layers
• It concatenates using skip connections instead of adding up
• It uses learnable weight filters instead of fixed interpolation technique

U-Net gives better performance because it has multiple up-sampling layers along with more skip connections which theoretically make it more robust to scale variations as compared to FCN

Upsampling Techniques:

- **Nearest Neighbours**
- **Bed of Nails**
- **Max Un-pooling**

Mask R-CNN for Instance Segmentation:

- It is Built on top of Faster R-CNN.
- It has an additional branch to predict segmentation mask on ROI in a pixel-pixel manner.
- It has 3 outputs, one for classification, other for regression and the third for object mask.
- It is divided into two parts
  - Region Proposal Network (RPN) – Candidate object bounding boxes
  - Binary Mask Classifier – Mask generation for every class

Loss Functions:

- **Binary Cross-Entropy:**

$$\text{Loss} = -\frac{1}{\substack{\text{output}\\\text{size}}} \sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

- **Dice Loss:**

  Dice Loss = 1 - Dice Coefficient

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

D:- Dice Coefficient

- **Triplet loss function:**

– Anchor and Positive: Minimum distance
– Anchor and Negative: Maximum distance

$$\mathcal{L} = max(d(a,p) - d(a,n) + margin, 0)$$

Image credits:

- 1st Image:- ResearchGate, https://www.researchgate.net/figure/An-example-of-convolution-operation-in-2D-2_fig3_324165524
- 2nd Image:- Medium, https://tsakunelsonz.medium.com/loading-and-training-a-neural-network-with-custom-dataset-via-transfer-learning-in-pytorch-8e672933469
- 3rd Image:- Analytics Vidhya, https://www.analyticsvidhya.com/blog/2019/02/tutorial-semantic-segmentation-google-deeplab/
- 4th Image:- Manal El Aidouni, https://manalelaidouni.github.io/Single%20shot%20object%20detection.html
- 5th Image:- O'Reilly, https://www.oreilly.com/library/view/reinforcement-learning-with/9781788835725/786aac81-77a7-437e-9a75-64925d7940ca.xhtml
- 6th Image:- ResearchGate, https://www.researchgate.net/figure/Intersection-over-Union-IOU-calculation-diagram_fig2_335876570

- 7th Image:- ResearchGate, https://www.researchgate.net/figure/Fully-convolutional-neural-network-architecture-FCN-8_fig1_327521314
- 8th Image:- Towards Data Science, https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5