Problem Statement

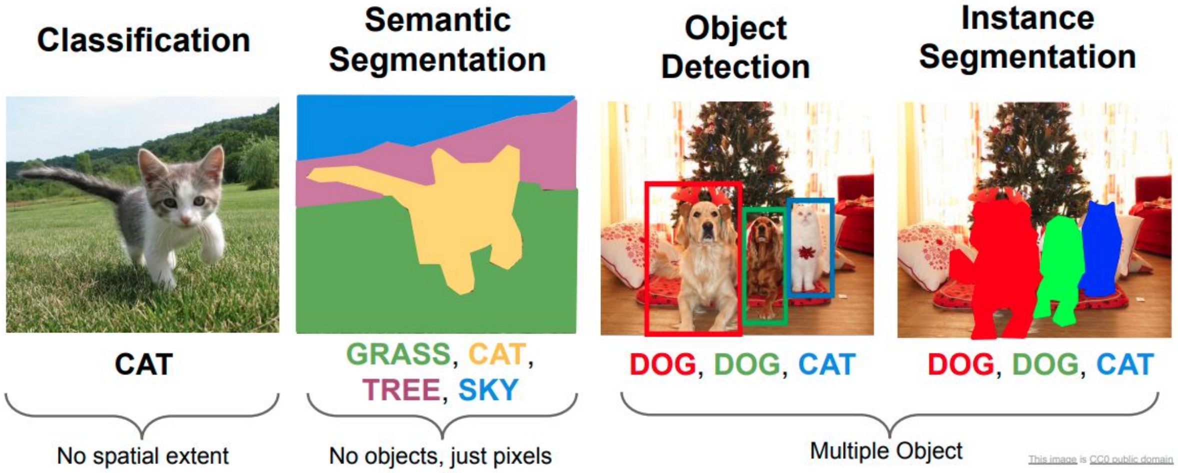


Image Source: http://cs231n.stanford.edu/

Object Localization

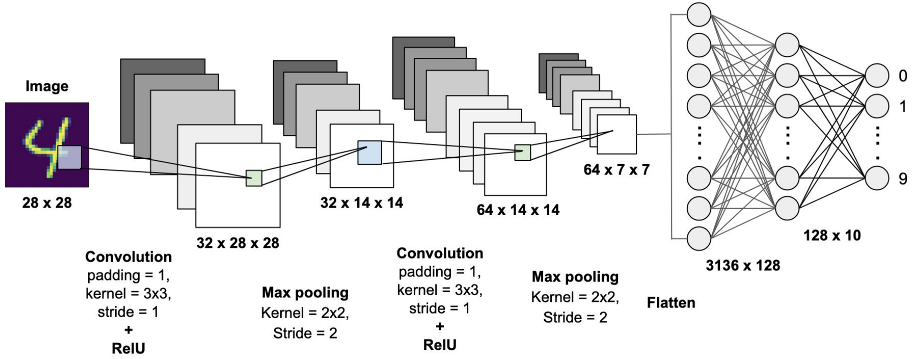
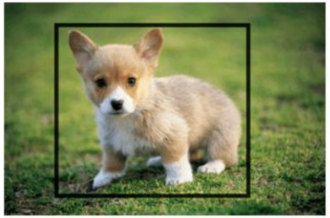


Image Source: [MNIST Handwritten Digits Classification using a Convolutional Neural Network (CNN)]()

Object Classification is the task of identifying that picture is a dog



Object Localization involves the class label as well as a bounding box to show where the object is located.

Image Source:
https://www.commonlounge.com/discussion/c9975025c9ff473c8f9ed2c4b1c3ea6a#image-localization-detection-segmentation

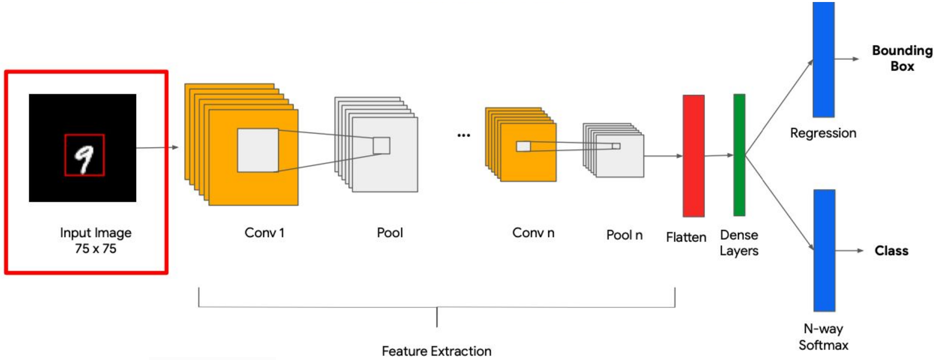


Image Source: deeplearning.ai

Idea: Sliding Window

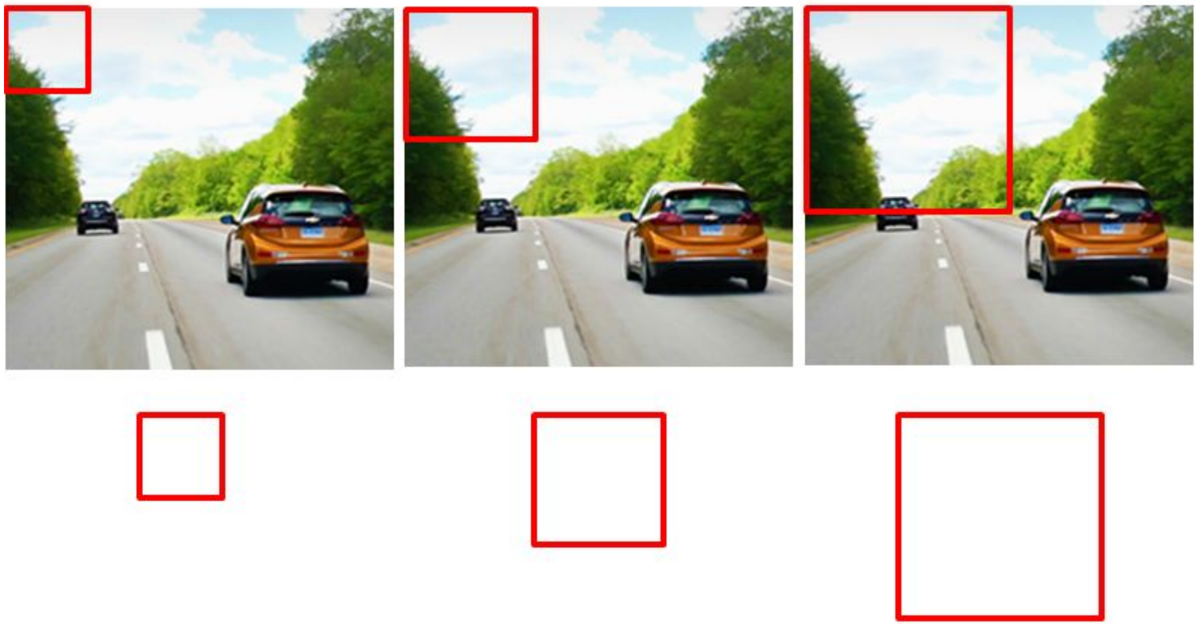


Image Source: https://datahacker.rs/deep-learning-object-detection/

Idea: Region Proposal

Felzenszwalb's Algorithm [2004] :

Efficient Graph-Based Image Segmentation

- Turn the image into an undirected graph **G = (V,E)**
  - Vertex $\mathbf{v_i} \in \mathbf{V}$ is a pixel
  - Edge $\mathbf{e} = (\mathbf{v_i}, \mathbf{v_j}) \in \mathbf{E}$ connects two vertices
  - Weight $\mathbf{w}(\mathbf{v_i}, \mathbf{v_j})$ measure dissimilarity between $\boldsymbol{v_i}$ and $\mathbf{v_j}$
- A segmentation solution **S** is a partition of **V** into connected components **{C}**

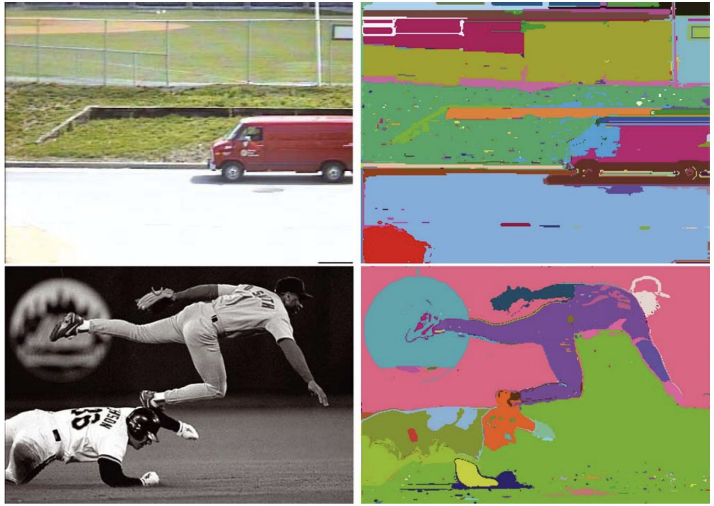


Image Source: Felzenszwalb, P.F., Huttenlocher, D.P. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision* 59, 167–181 (2004).

Idea: Region Proposal

Selective Search :

“Class Agnostic Object Detector”

- Start from Felzenszwalb’s Segmentations
- Recursively combine similar regions into larger ones
- Convert regions to boxes

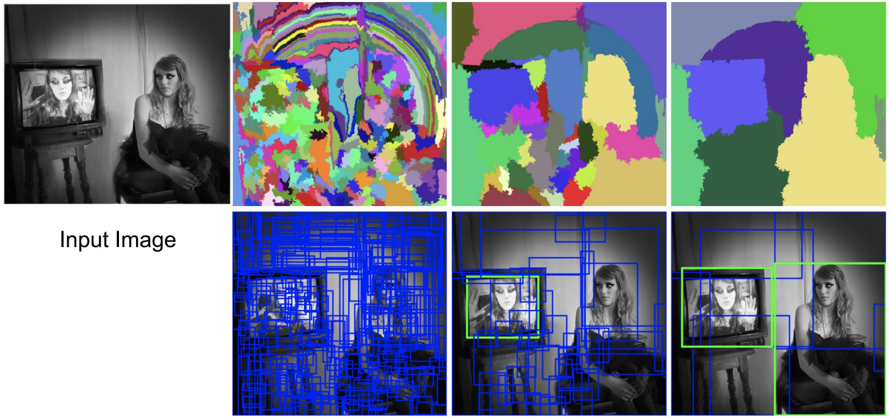
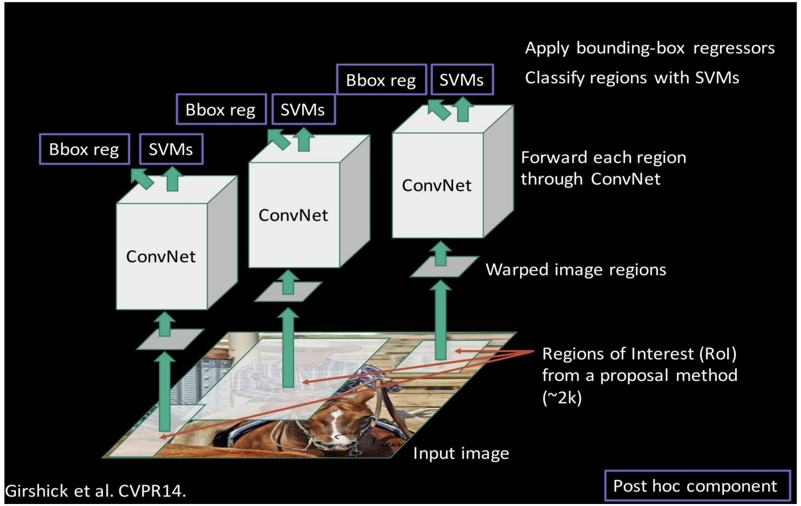


Image Source: http://vision.stanford.edu/teaching/cs231b_spring1415

Overview: R-CNNs

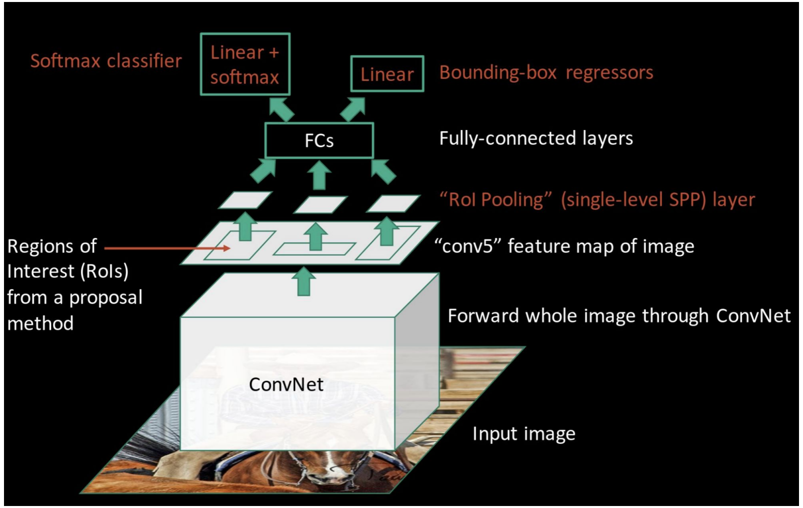
R-CNN



Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image
Girshick et al. CVPR14.
Post hoc component

Image: Girschick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014

Fast R-CNN



Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
“RoI Pooling” (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
“conv5” feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Image: Girschick, “Fast R-CNN”, ICCV 2015

Overview: R-CNNs

Faster R-CNN

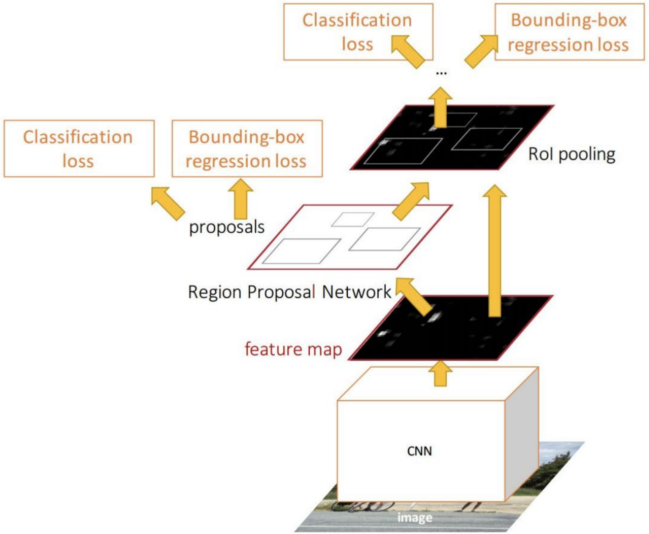


Image: Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Mask R-CNN

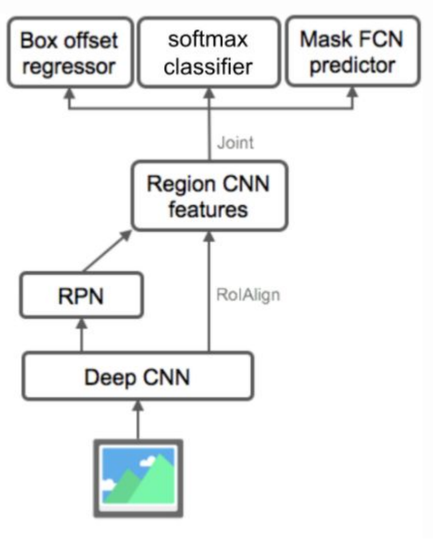


Image Source: Object Detection for Dummies Part 3: R-CNN Family

Model: Mask R-CNN

Transfer Learning:

- Start with a pretrained model
- Replace classifier and mask predictor
- Fine tune

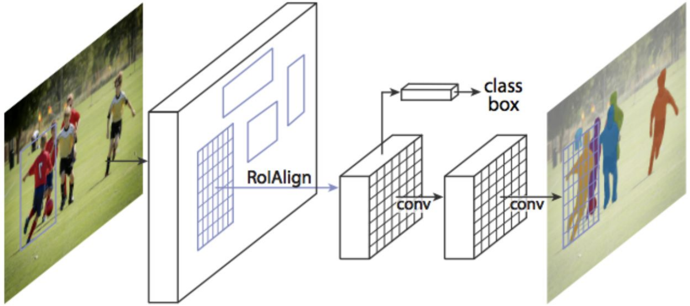


Image Source: Object Detection for Dummies Part 3: R-CNN Family

Loss Function: $\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{mask}}$

where;

$$\mathcal{L}(\{p_i\}, \{t_i\}) = \frac{1}{N_{\text{cls}}} \sum_i \mathcal{L}_{\text{cls}}(p_i, p_i^*) + \frac{\lambda}{N_{\text{box}}} \sum_i p_i^* \cdot L_1^{\text{smooth}}(t_i - t_i^*)$$

$$\mathcal{L}_{\text{cls}}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i)$$

$$\mathcal{L}_{\text{mask}} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij} \log \hat{y}_{ij}^k + (1 - y_{ij}) \log(1 - \hat{y}_{ij}^k) \right]$$

Model: Mask R-CNN

Transfer Learning:

- Start with a pretrained model
- Replace classifier and mask predictor
- Fine tune

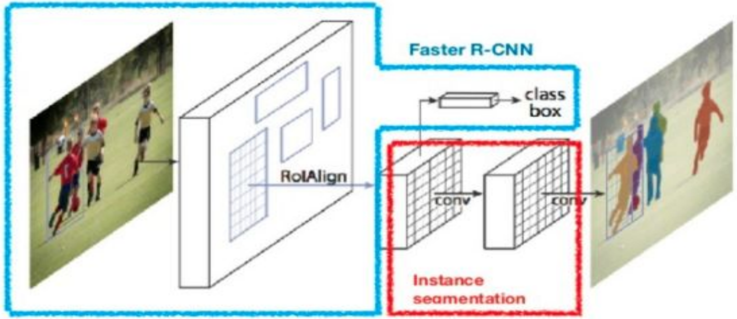


Image Source: [Object Detection for Dummies Part 3: R-CNN Family]

Loss Function: $\mathcal{L} = \boxed{\mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}}} + \boxed{\mathcal{L}_{\text{mask}}}$

where;

$$\boxed{\begin{aligned}
&\mathcal{L}(\{p_i\}, \{t_i\}) = \frac{1}{N_{\text{cls}}} \sum_i \mathcal{L}_{\text{cls}}(p_i, p_i^*) + \frac{\lambda}{N_{\text{box}}} \sum_i p_i^* \cdot L_1^{\text{smooth}}(t_i - t_i^*) \\
&\mathcal{L}_{\text{cls}}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i)
\end{aligned}}$$

$$\boxed{\mathcal{L}_{\text{mask}} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij} \log \hat{y}_{ij}^k + (1 - y_{ij}) \log(1 - \hat{y}_{ij}^k) \right]}$$

Tutorials:

PyTorch:

- TORCHVISION OBJECT DETECTION FINETUNING TUTORIAL
  https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html#torchvision-object-detection-finetuning-tutorial

Tensorflow:

- Object Detection API with Tensorflow 2
  https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2.md

Keras:

- Object Detection with RetinaNet
  https://keras.io/examples/vision/retinanet/

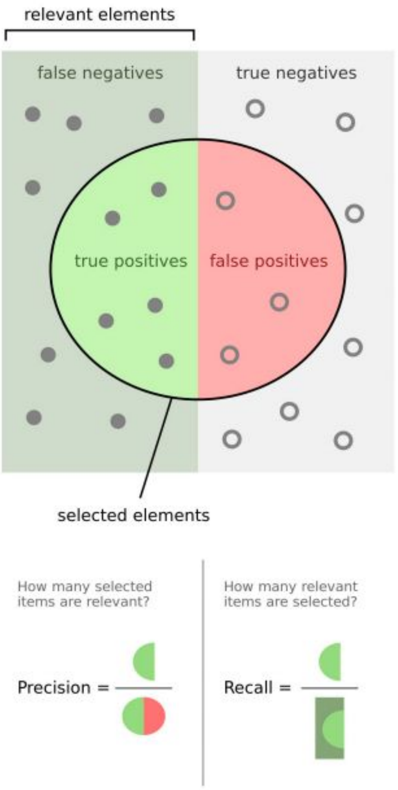
Results: Metrics

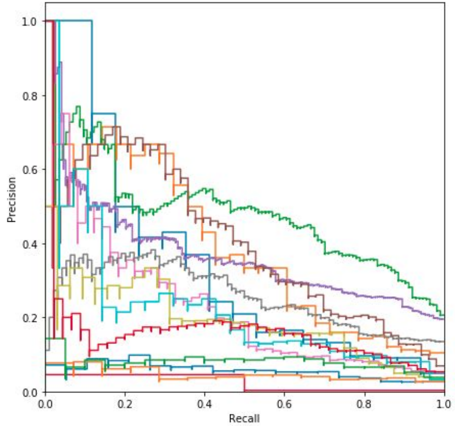
- **Precision** = TP / (TP + FP)
- **Recall** = TP / (TP + FN)

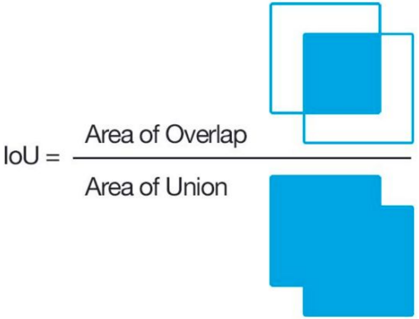
- **Average Precision (AP):**
  Area under Precision/Recall curve

- **Mean AP (mAP):**
  Average AP over all classes

  mAP@0.5 → TP if IoU > 0.5


relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many selected items are relevant?
Precision =
How many relevant items are selected?
Recall =


Precision
Recall


IoU =
Area of Overlap
Area of Union