Introduction to Deep Learning

Object Detection and Image Segmentation

Problem of Image Classification

- Mostly on centered images
- Only a single object per image

→ Cannot solve many real life vision tasks

Beyond Image Classification

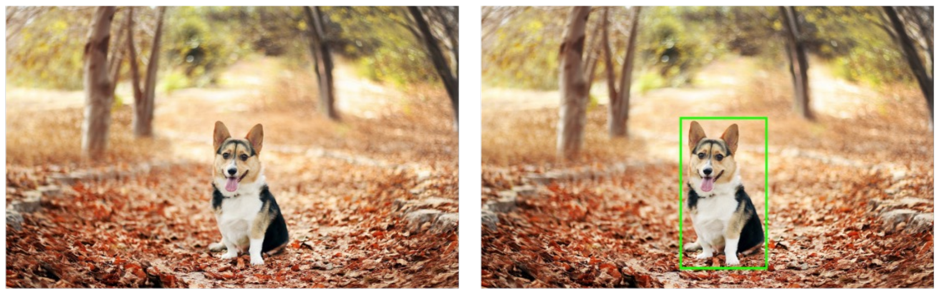
Classification

single object

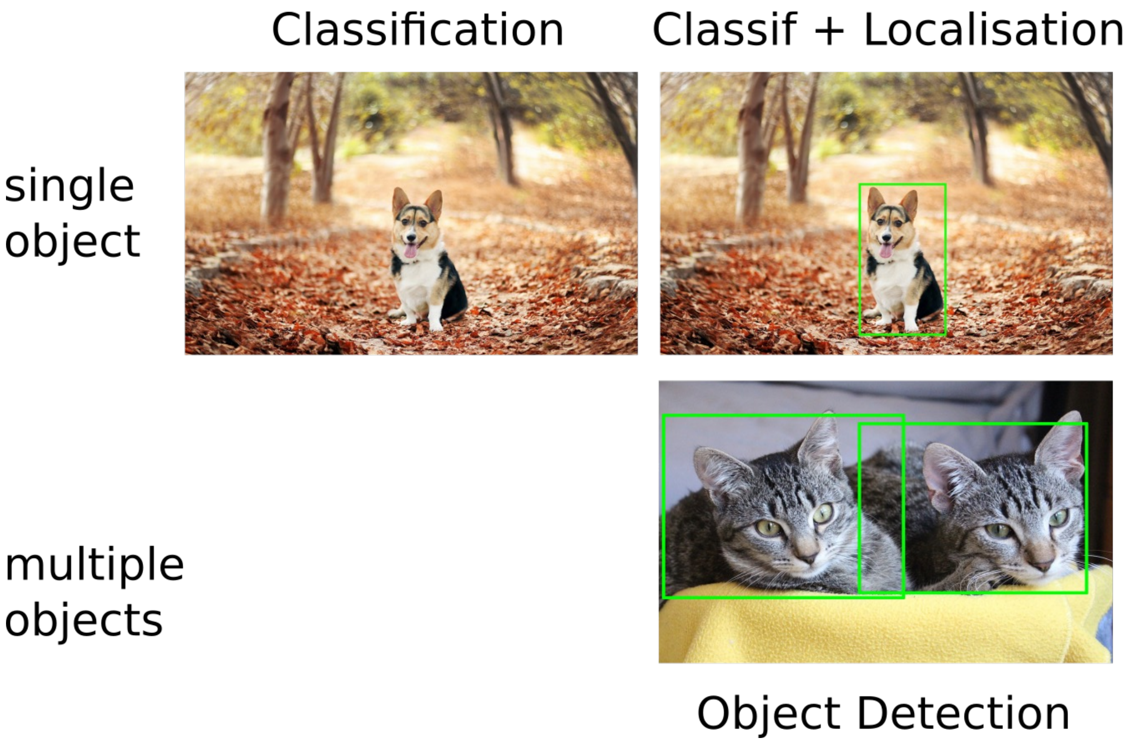
Beyond Image Classification

Classification | Classif + Localisation

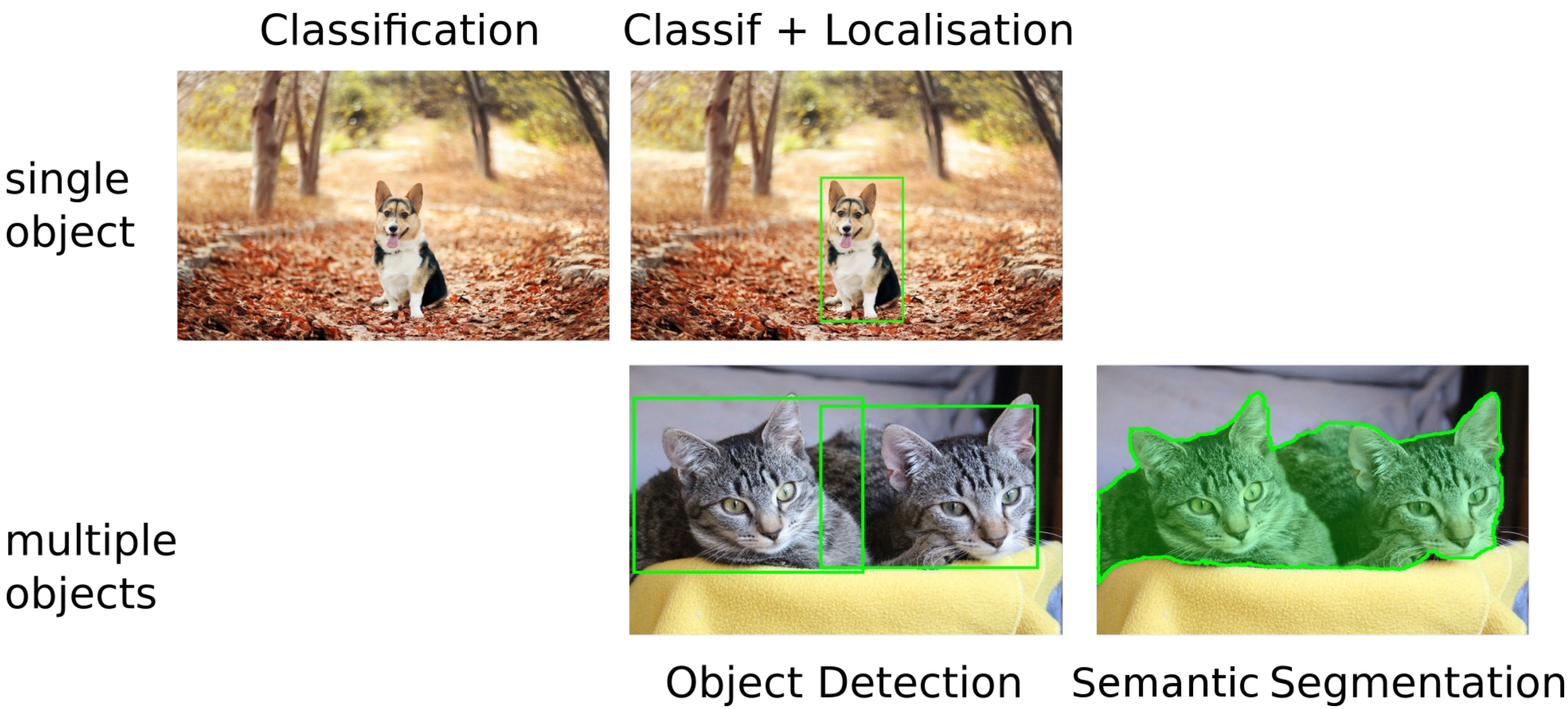
single object



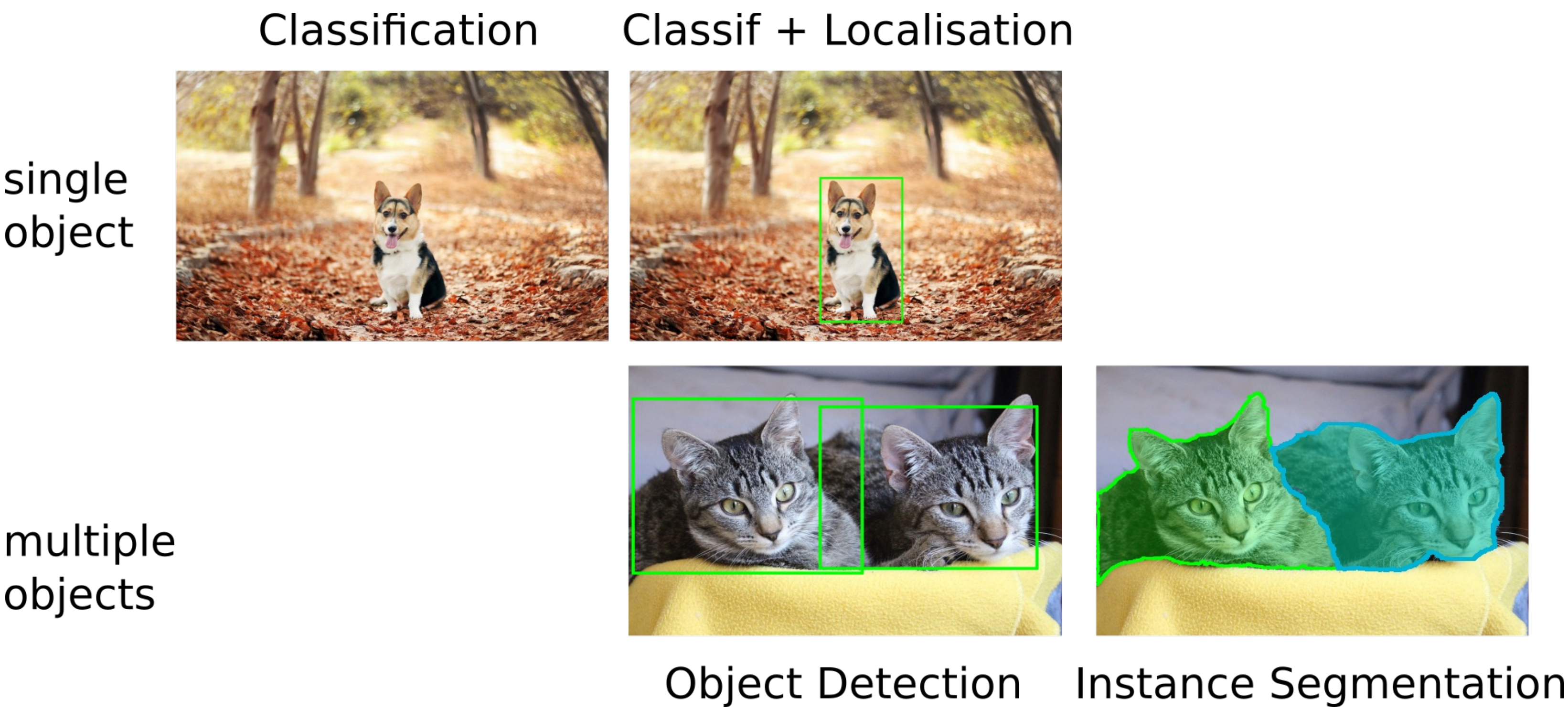
Beyond Image Classification



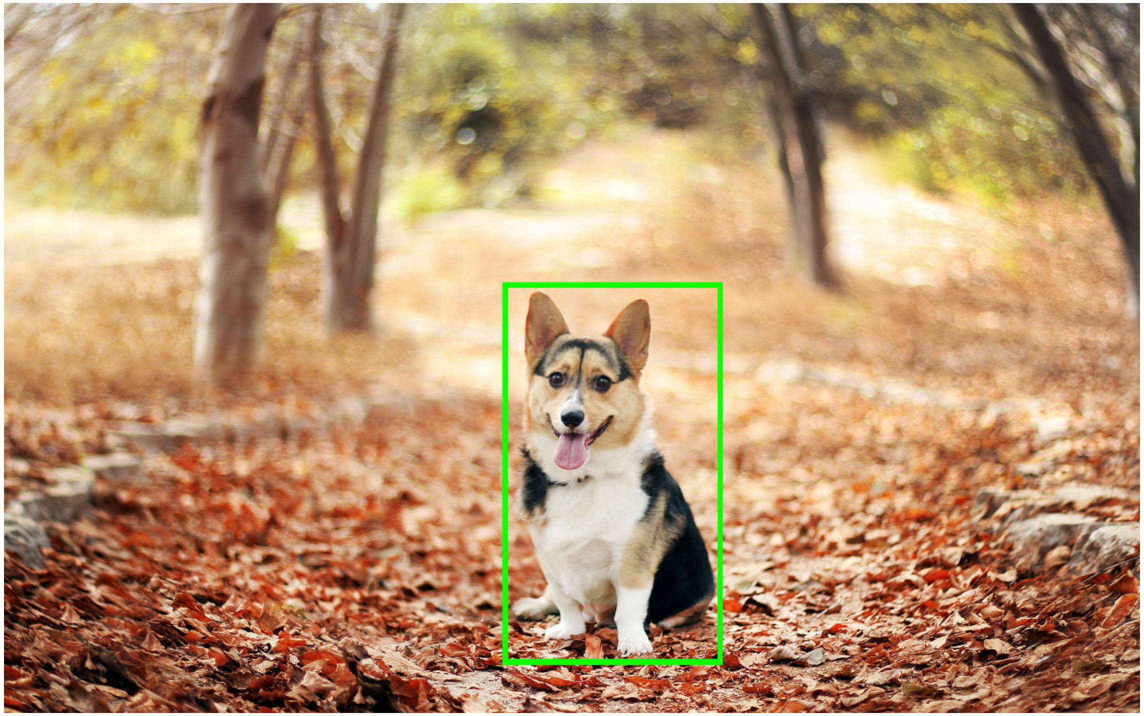
Beyond Image Classification



Beyond Image Classification

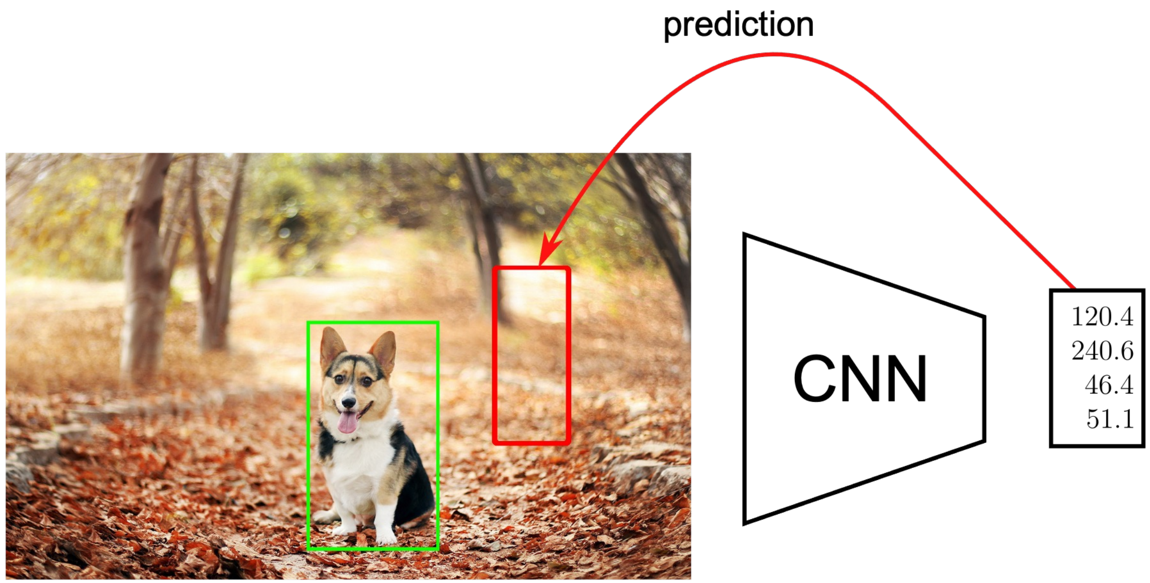


Localization

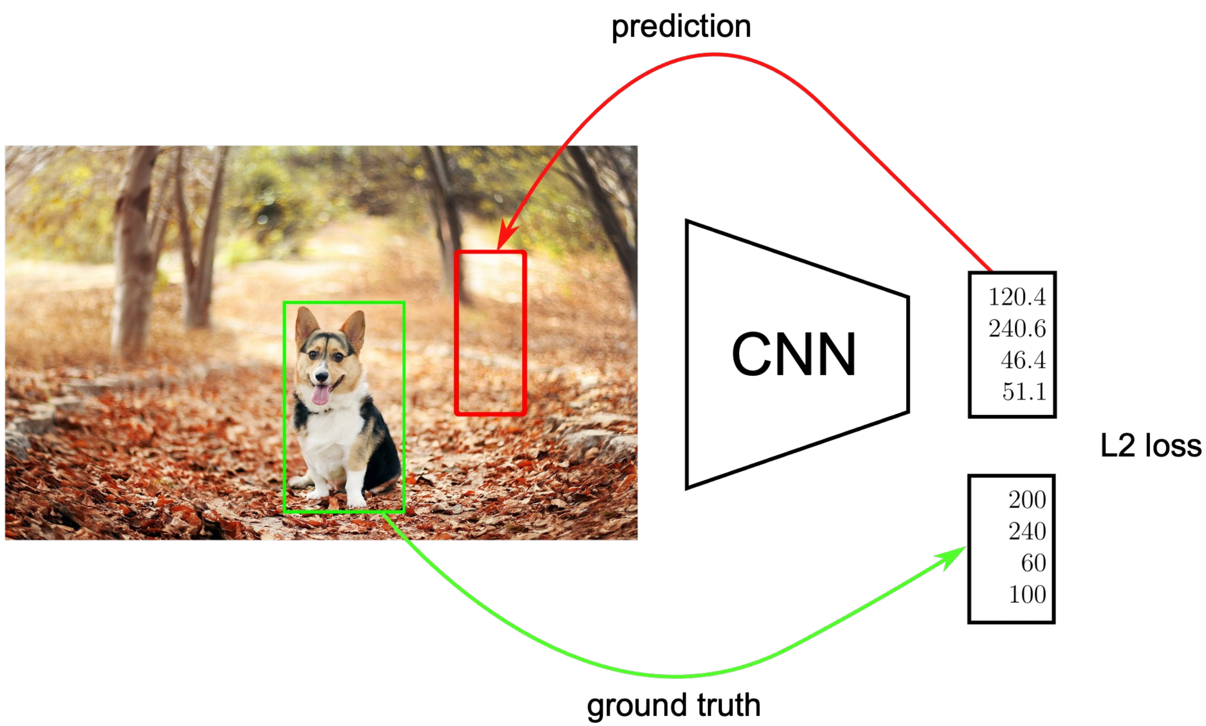


- Single object per image
- Predict coordinates of a bounding box (x, y, w, h)
- Evaluate via the metric Intersection over Union (IoU)

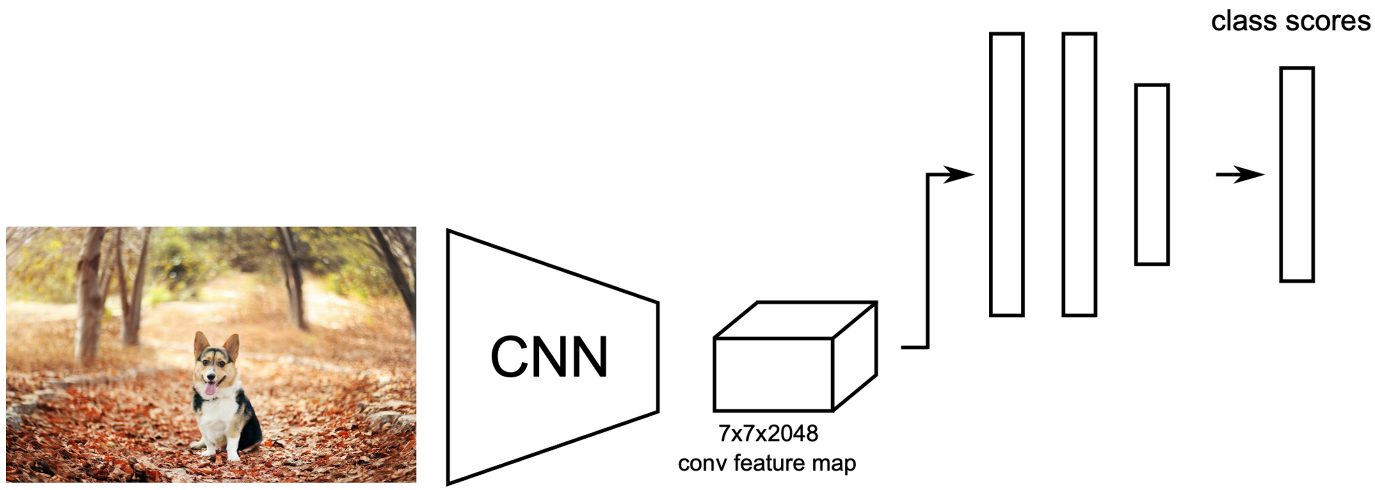
Localization as Regression



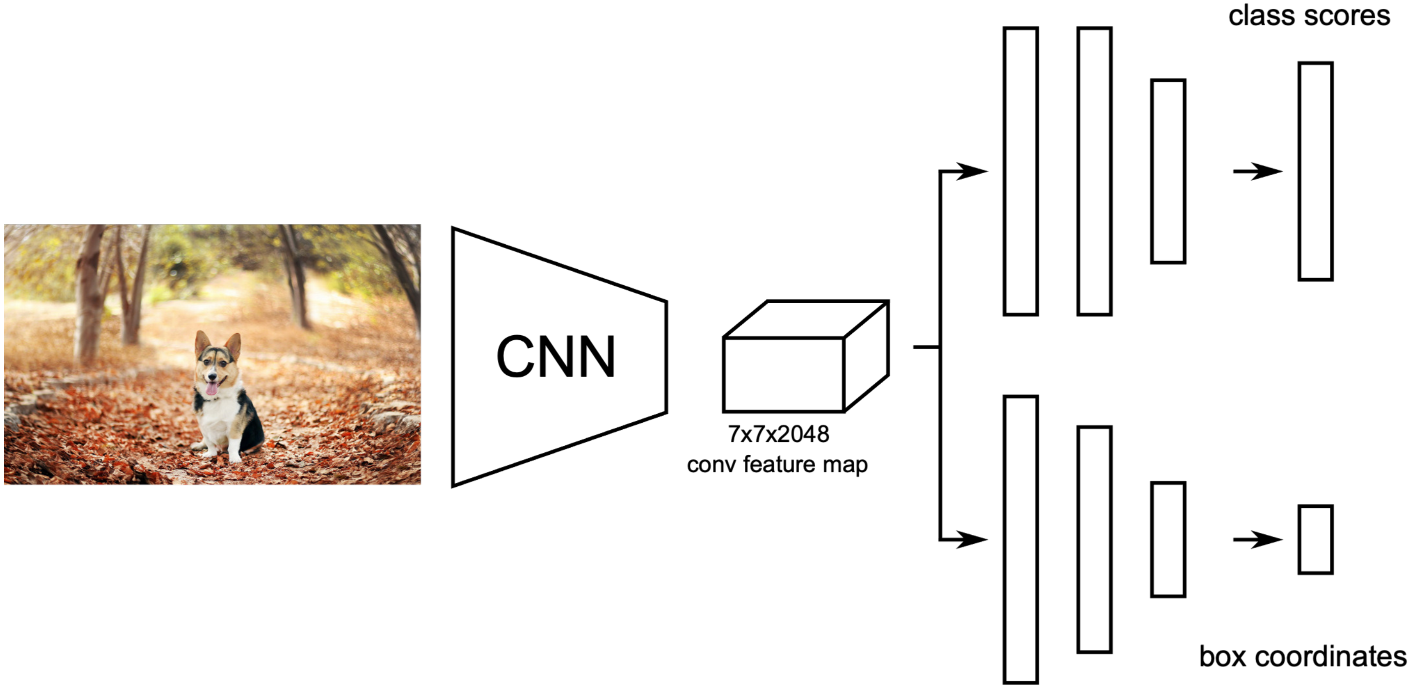
Localization as Regression



Classification + Localization

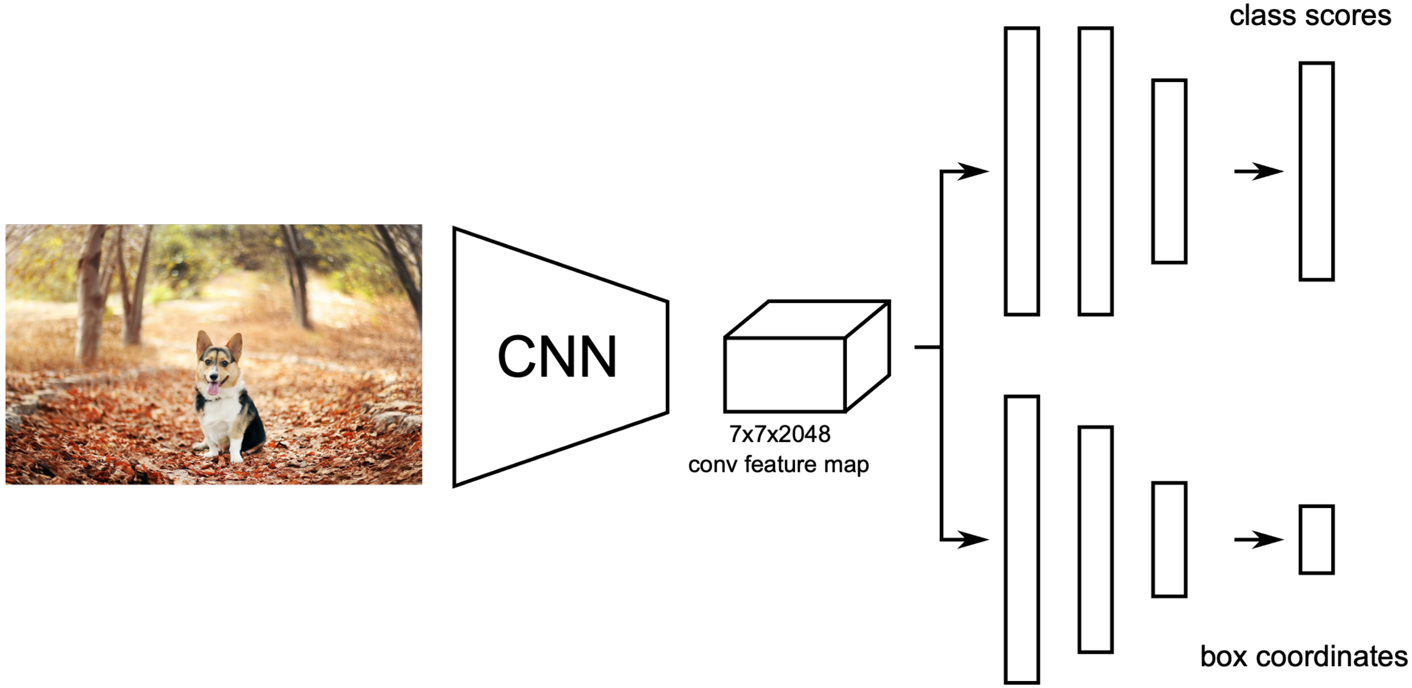


Classification + Localization



- Use a pre-trained CNN on ImageNet (ex. ResNet)
- The “localization head” is trained separately with regression
- Possible end-to-end fine-tuning of both tasks
- At test time, use both heads

Classification + Localization



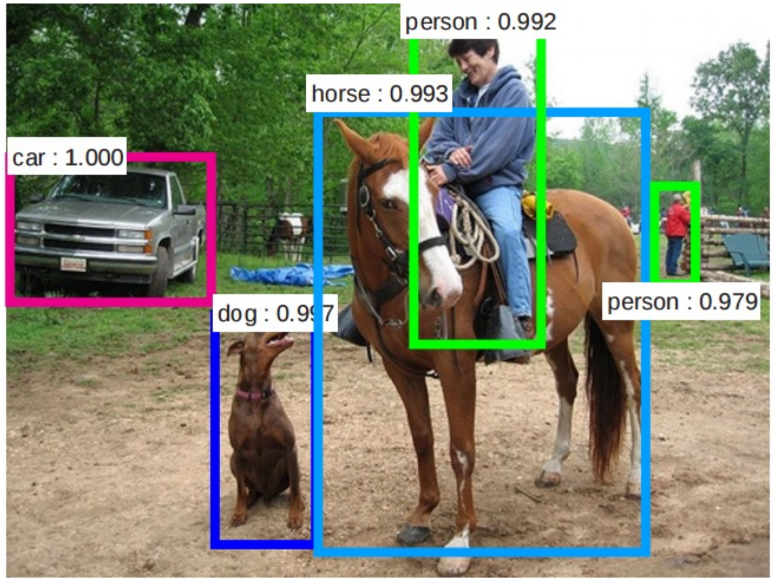
- *C* classes, 4 output dimensions (for 1 bounding box)
- Predict exactly *N* objects: predict (*N x 4*) coordinates and *(N x K)* class scores

Object Detection

- Problem: we don't know in advance the number of objects in the image

- Object detection performs two main tasks:
  - Object proposal: find regions of interest (ROIs) in the image
  - Object classification: classify the object in these regions

Object Detection



Two main tasks in object detection can be expressed as follows:

- Define bounding boxes of the objects
- For each bounding box, classify the object inside it to some classes (e.g. dog, horse, person, car, etc…) with % of confidence

Object Detection

- Two main families to perform object detection:
  - Single-Stage: A grid in the image where each cell is a proposal
  - Two-Stage: Region proposal then classification (Faster R-CNN)

Object Detection with Deep Learning

- Faster R-CNN
- YOLO
- RetinaNet

Object Detection with Deep Learning

- Faster R-CNN
- YOLO
- RetinaNet

Faster R-CNN

- R-CNN
- Fast R-CNN
- Faster R-CNN

Instead of having a predefined set of box proposals, find them on the image by:

- Selective Search – from pixels (not learnt, not used any more)

- Faster R-CNN – Region Proposal Network (RPN)

Faster R-CNN

Crop-and-resize operator (RoI-Pooling)

- Input: Convolutional map + N regions of interest
- Output: tensor of N x 7 x7 x depth boxes
- Allow to propagate gradient only on interesting regions, and efficient computation

R-CNN

- R-CNN (Region with CNN feature) algorithm:
  - Step 1: use **Selective Search** algorithm to get around 2000 bounding box in the input image which can contain the object
  - Step 2: with each bounding box, identify its class (e.g. person, car, etc…)

Selective Search Algorithm

- Input: color image
- Output: around 2000 region proposal (bounding box) which can contain the objects

Selective Search Algorithm

Input Image

Output Image

Image is segmented using the Graph Based Image Segmentation algorithm

Selective Search Algorithm

Input Image

Output Image

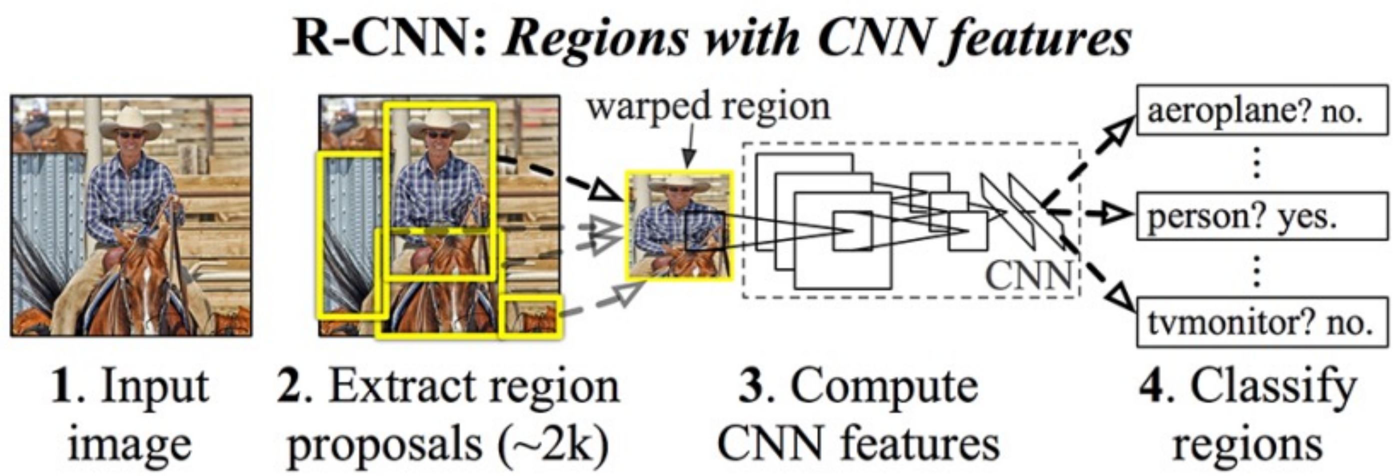
- Cannot use 1 color to present 1 region proposal
  - Each object can contain several colors
  - Some parts of an object can be hidden by others
- 1 region proposal is presented by a group of colors, each having color similarity, gradient direction, size, etc.

Classify Region Proposal

- Problem becomes Region Proposal Classification
- Issue: in 2000 region proposals output from selective search algorithm, there exists region proposal **without any objects**

- → one background class is added to solve the issue

Classify Region Proposal

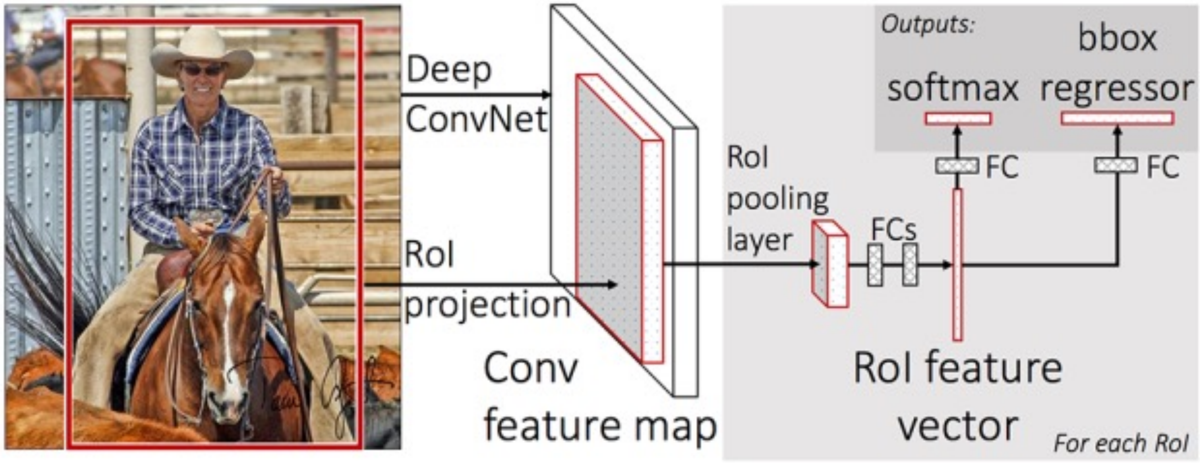


Input image (1) → Extract region proposals (2) → resize region proposals to the same size → transfer learning with feature extractor (3) → use SVM to classify the regions as person, or horse or background, etc. (4)

Problem of R-CNN

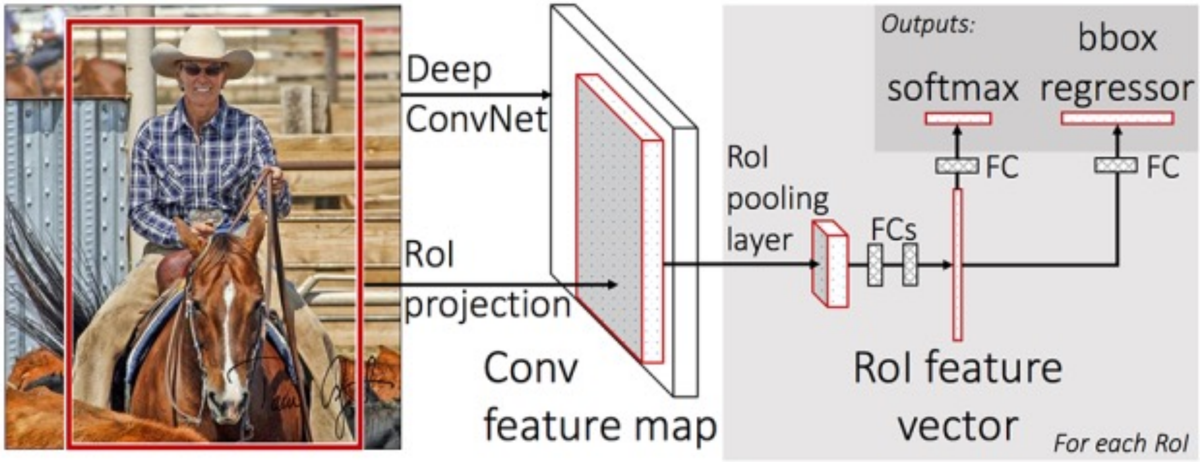
- For each image, need to classify 2000 region proposals -> very long training time
- Cannot apply real-time object detection → each image in test set uses 47s for processing

Fast R-CNN



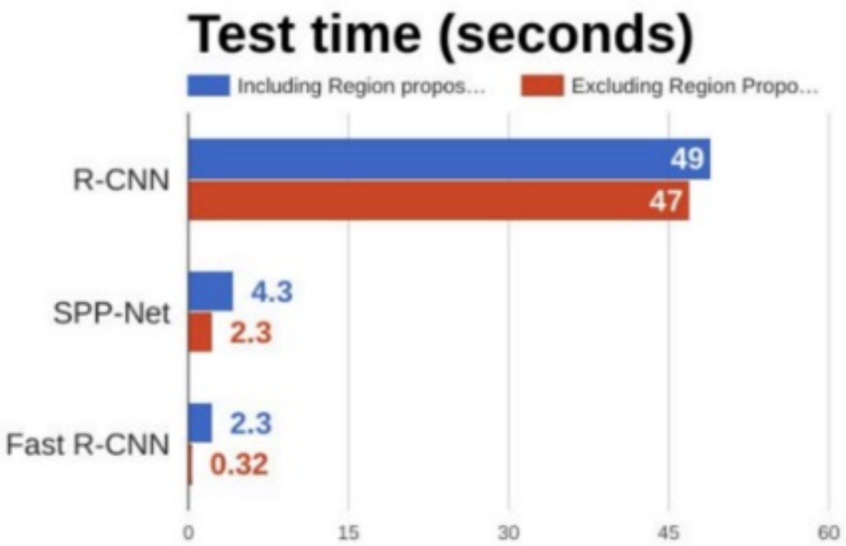
- The whole input image is fed into Deep ConvNet to produce Conv feature map
- RoI is projected into Deep ConvNet with the input image
- → region proposals are obtained from Conv feature map

Fast R-CNN



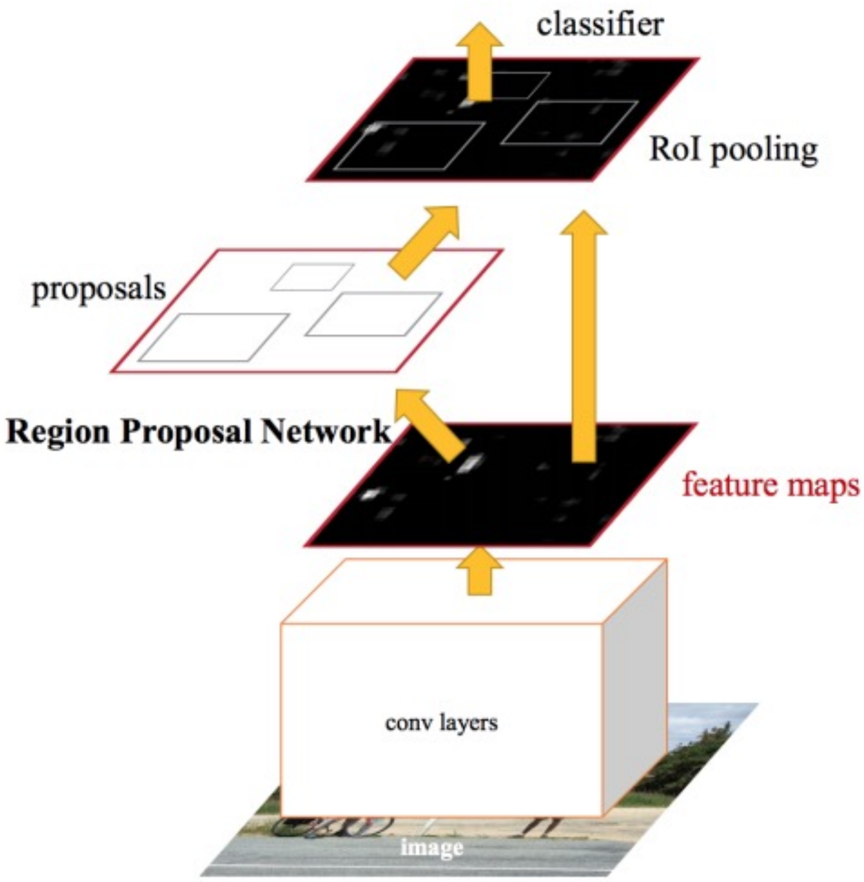
- RoI pooling layer is used to transform region proposals in the Conv feature maps to the same size
- Region proposals are flattened and fed into 2 FCs layers to predict class and regress the offset values of bounding box

R-CNN vs. Fast R-CNN



- Fast R-CNN is much faster than R-CNN
- But in Test time, calculating region proposals by selective search make Fast R-CNN slower
- → we can replace the selective search algorithm by deep learning?
- → Yes! Using Faster R-CNN

Faster R-CNN

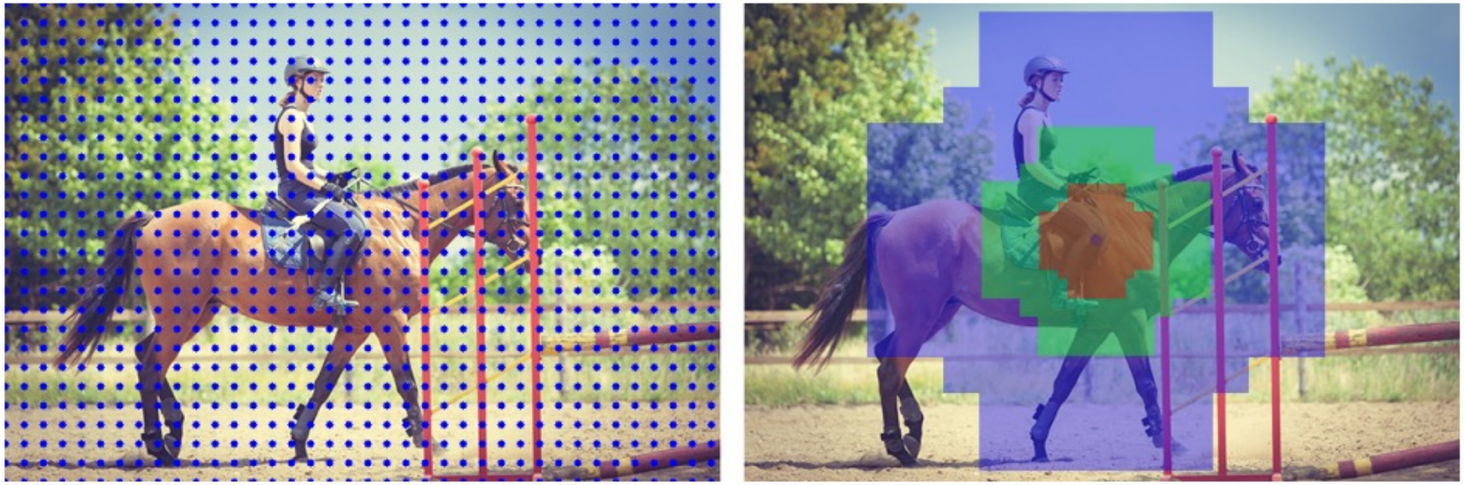


Architecture of Faster R-CNN

Region Proposal Network

- Replace selective search algorithm to obtain region proposals from feature maps
- Input: feature map
- Output: region proposals
- → anchor box is used to present region proposal

Concept of Anchor Box



- Each anchor box is defined by 4 paramaters (x_center, y_center, width, height)
- # of anchors are pre-defined → after passing through RPN → only anchor box containing objects are kept

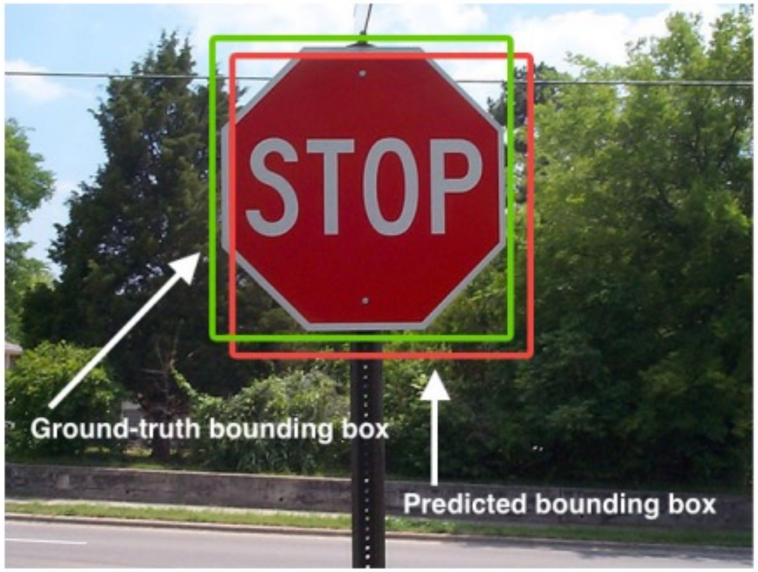
RPN algorithm

- Feature maps are fed into Conv layer 3*3, 512 kernels
- With each anchor, RPN calculates two steps:
  - Predict if anchor is foreground (contain object) or background (does not contain object)
  - Predict 4 offset values for x_center, y_center, width, height of anchor
- Non-maxima suppression is used to remove overlap anchor boxes
- Based on confidence score, RPN will get N (N can be 2000, 1000, etc.) anchor boxes to be the predicted region proposals

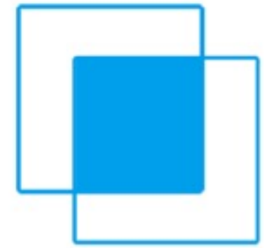
Non-maxima Suppression Algorithm

- Input: 9000 anchor boxes
- Input: 100 anchor boxes as region proposals
- Algorithm:
  - (1) Choose anchor box (A) with a maximum value of foreground probability
  - (2) Add anchor box A to the output set
  - (3) Remove A and a set of anchor boxes in input set which has IoU value with A > 0.5
  - (4) Check if input set is empty or output set is equal to 100, then stop, otherwise repeat step 1

Intersection over Union (IoU) Metric



$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

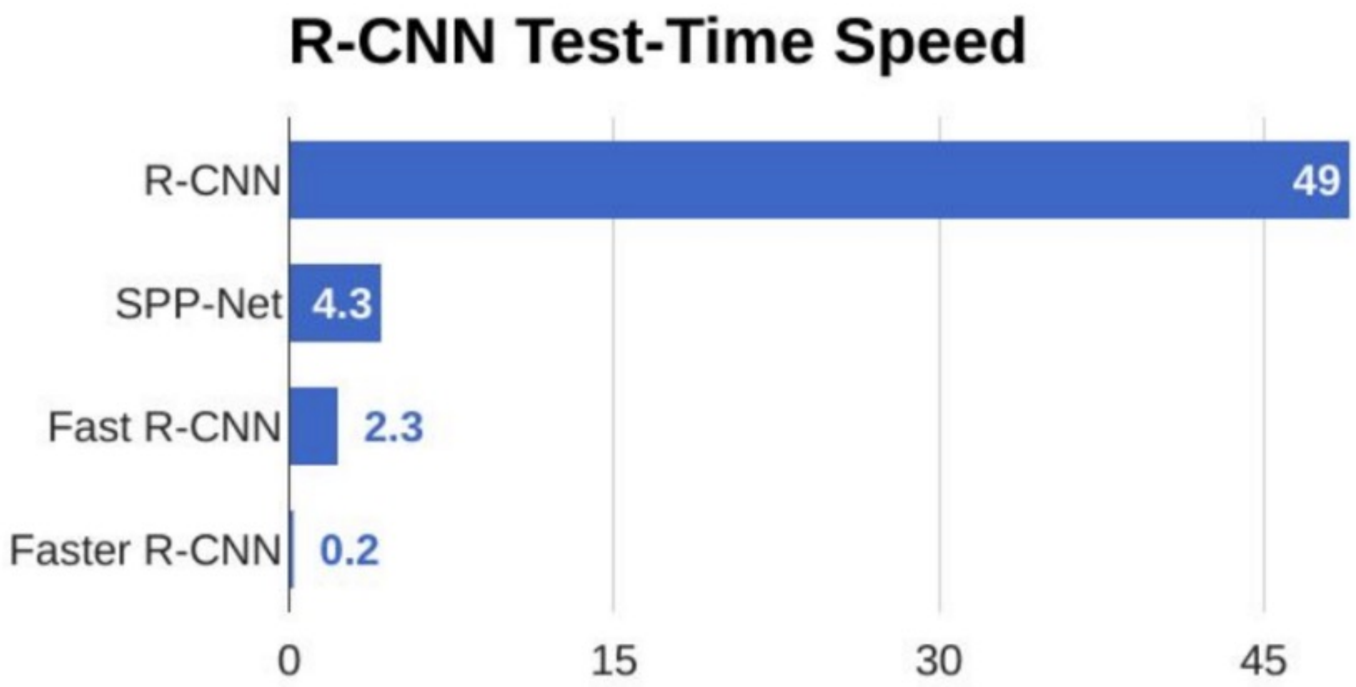


- IoU in the range [0,1]
- IoU → 1 then predicted bounding box → close to the ground truth

Intersection over Union (IoU) Metric

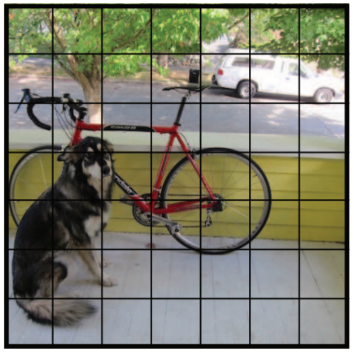
- Example of IoU metric

Faster R-CNN vs. Fast R-CNN vs. R-CNN



Comparison of test-time speed of object detection algorithms

YOLO



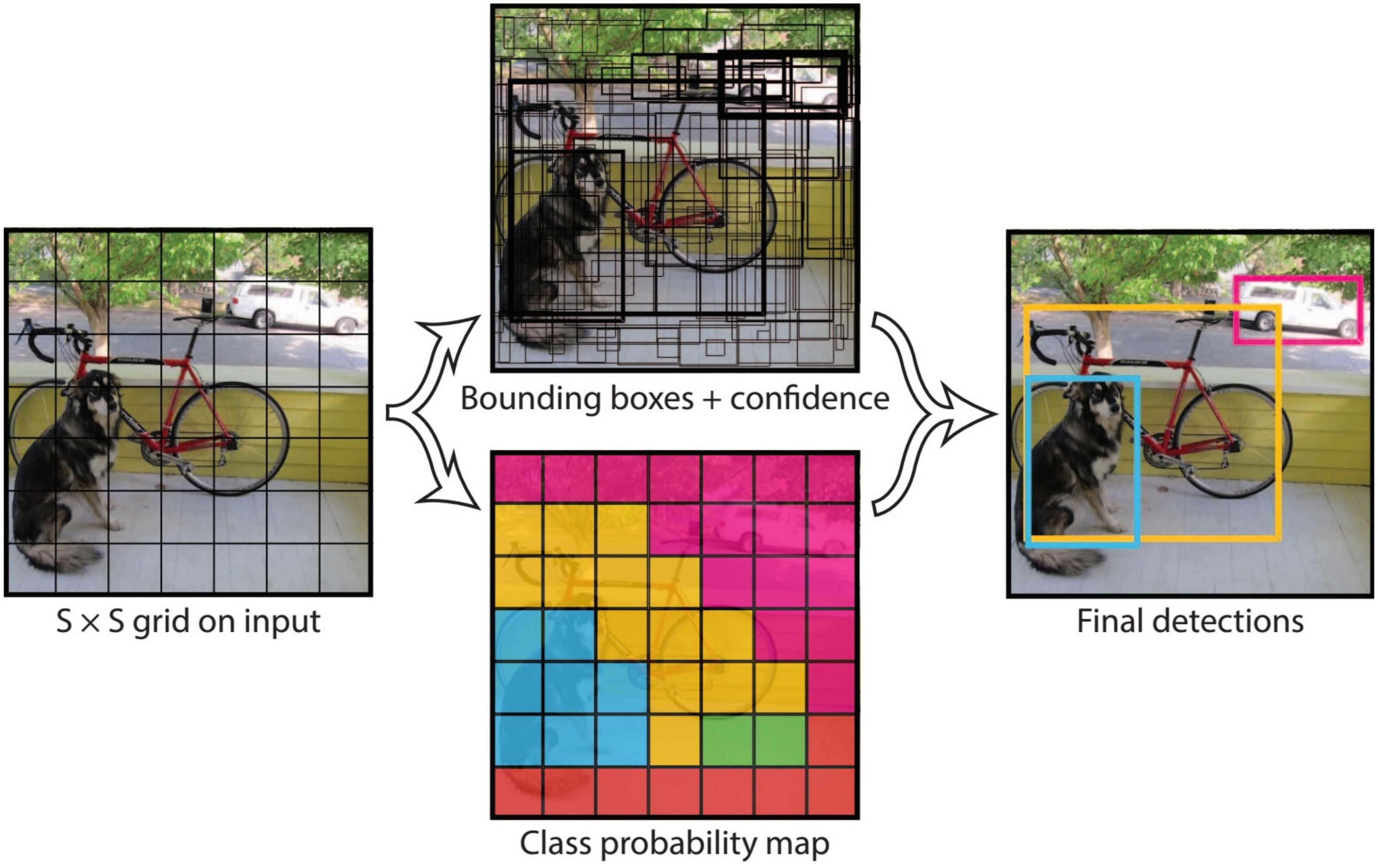
S × S grid on input

For each cell of the S x S predict:

- B boxes and confidence scores C (5 x B values) + classes c

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." CVPR (2016)

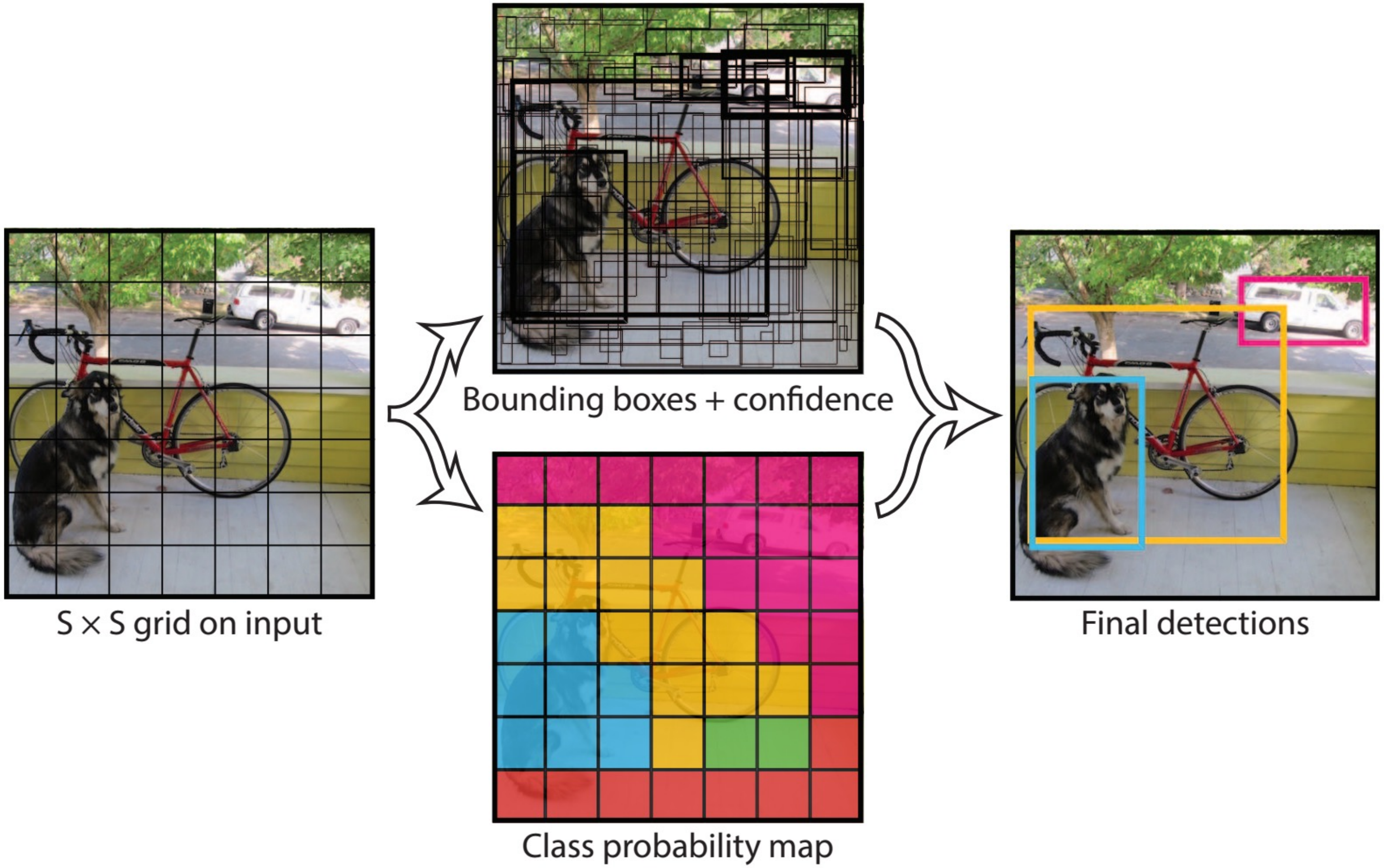
YOLO



For each cell of the S x S predict:

- B boxes and confidence scores C (5 x B values) + classes c

YOLO



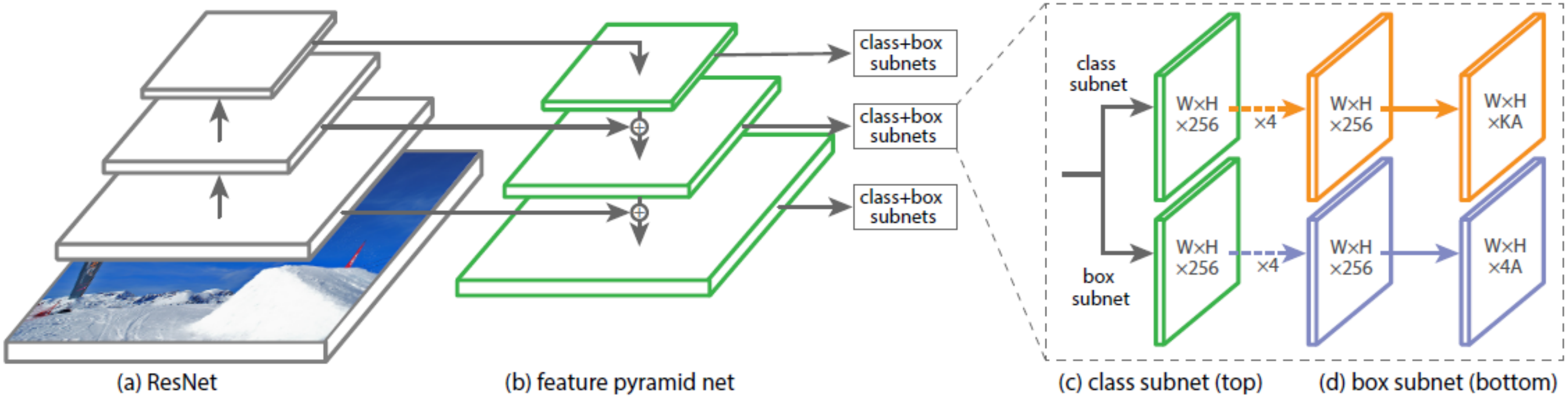
Final detections: C_j * prob(c) > threshold

YOLO

- After ImageNet pretraining, the whole network is trained end-to-end
- The loss is a weighted sum of different regressions:

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{aligned}
$$

RetinaNet

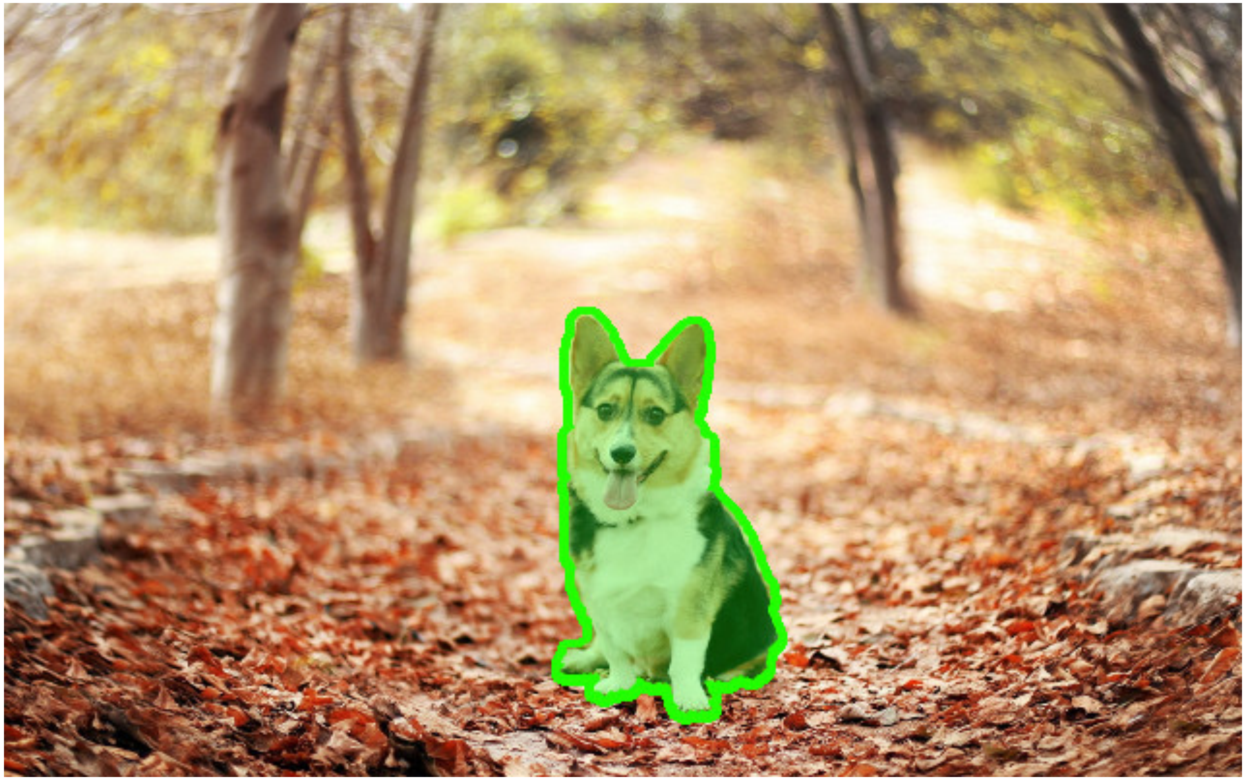


Single stage detector with:

- Multiple scales through a Feature Pyramid Network
- Focal loss to manage imbalance between background and real objects

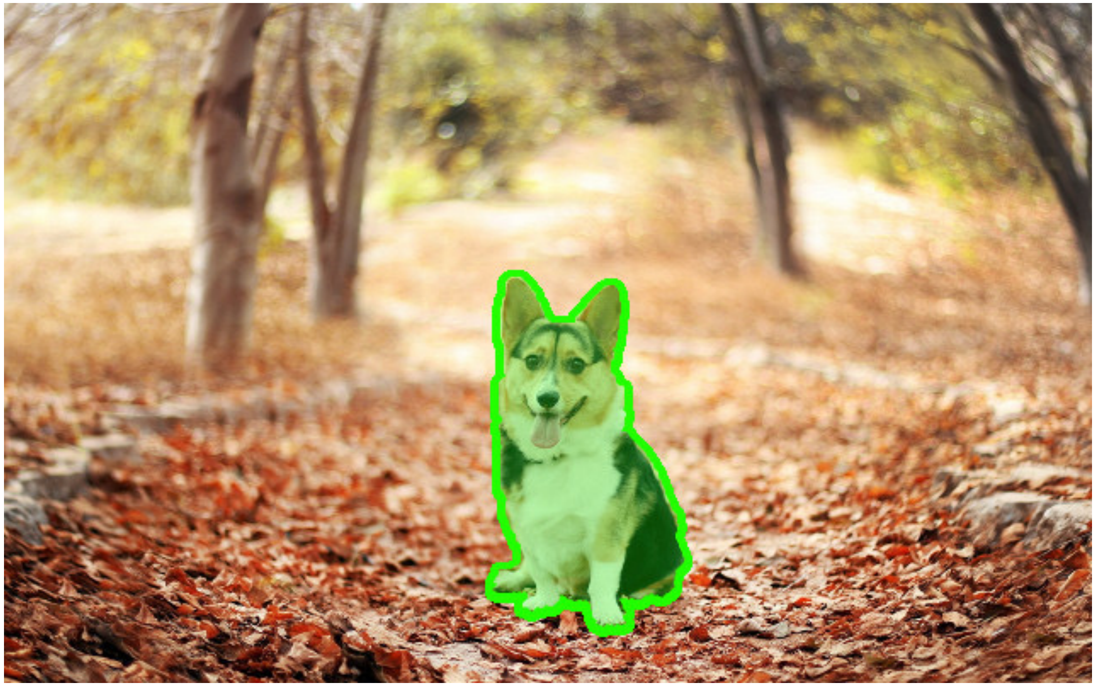
See this link for more information: https://towardsdatascience.com/review-retinanet-focal-loss-object-detection-38fba6afabe4

Image Segmentation



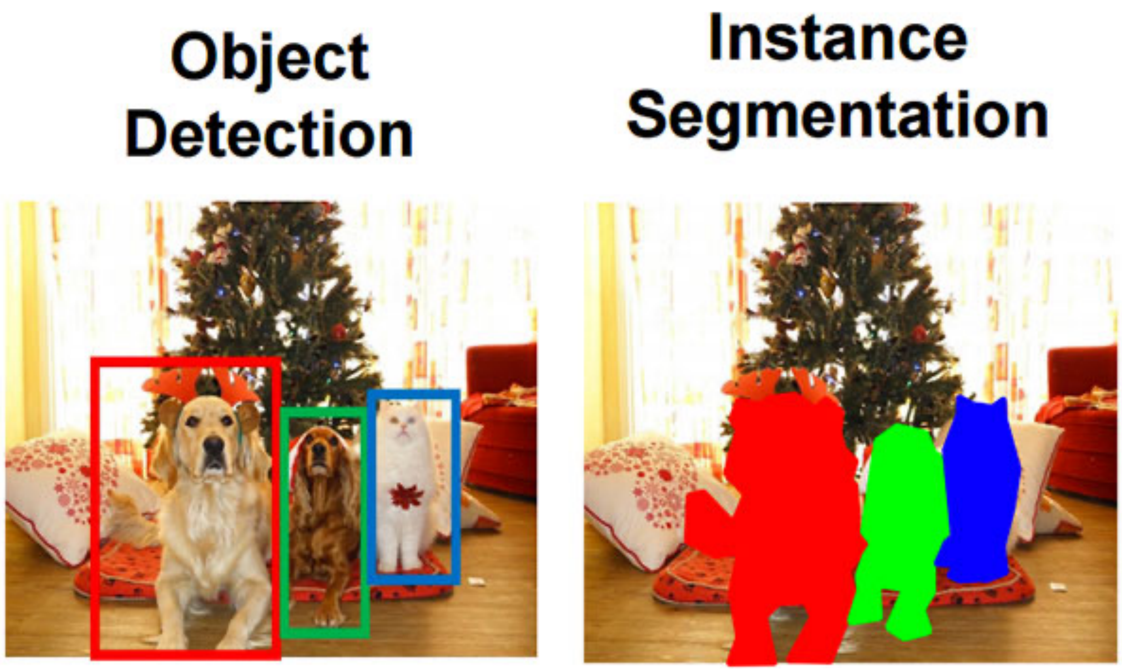
Output a class map for each pixel
(here: dog vs background)

Image Segmentation



- Instance segmentation: specify each object instance as well (two dogs have different instances)
- This can be done through object detection + segmentation

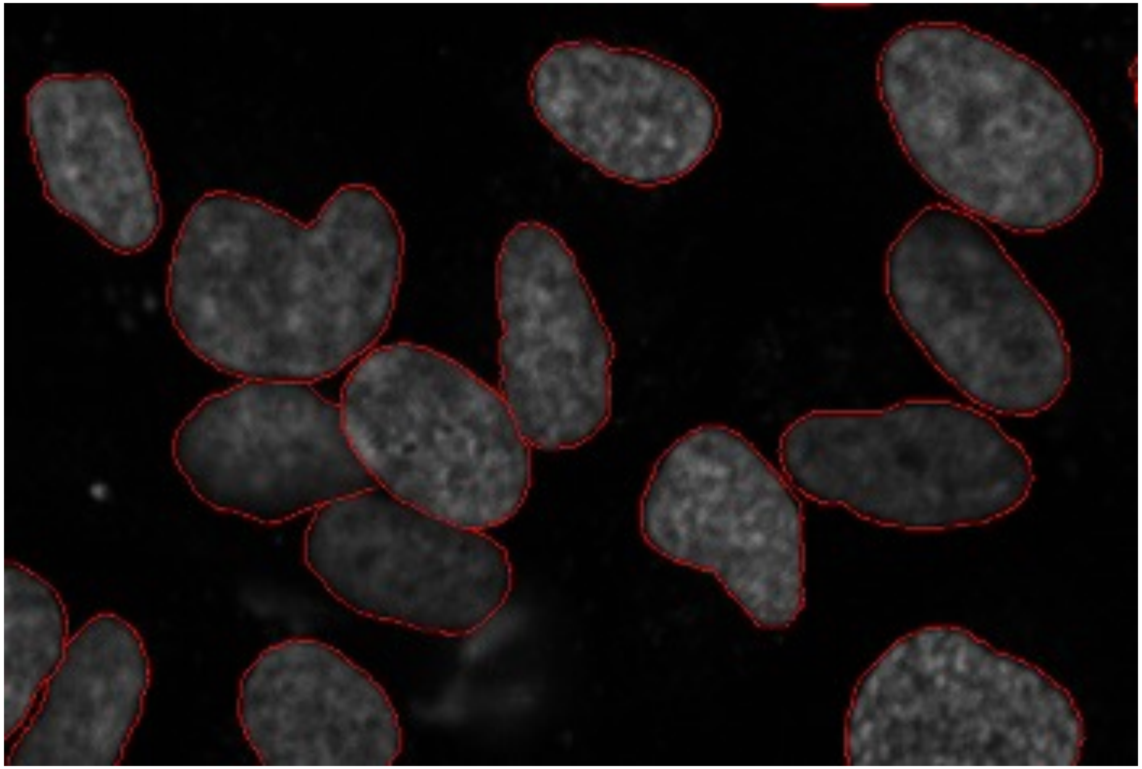
Image Segmentation



Object detection vs. Image segmentation

- Shape of the object is not important in object detection

Why care about image segmentation



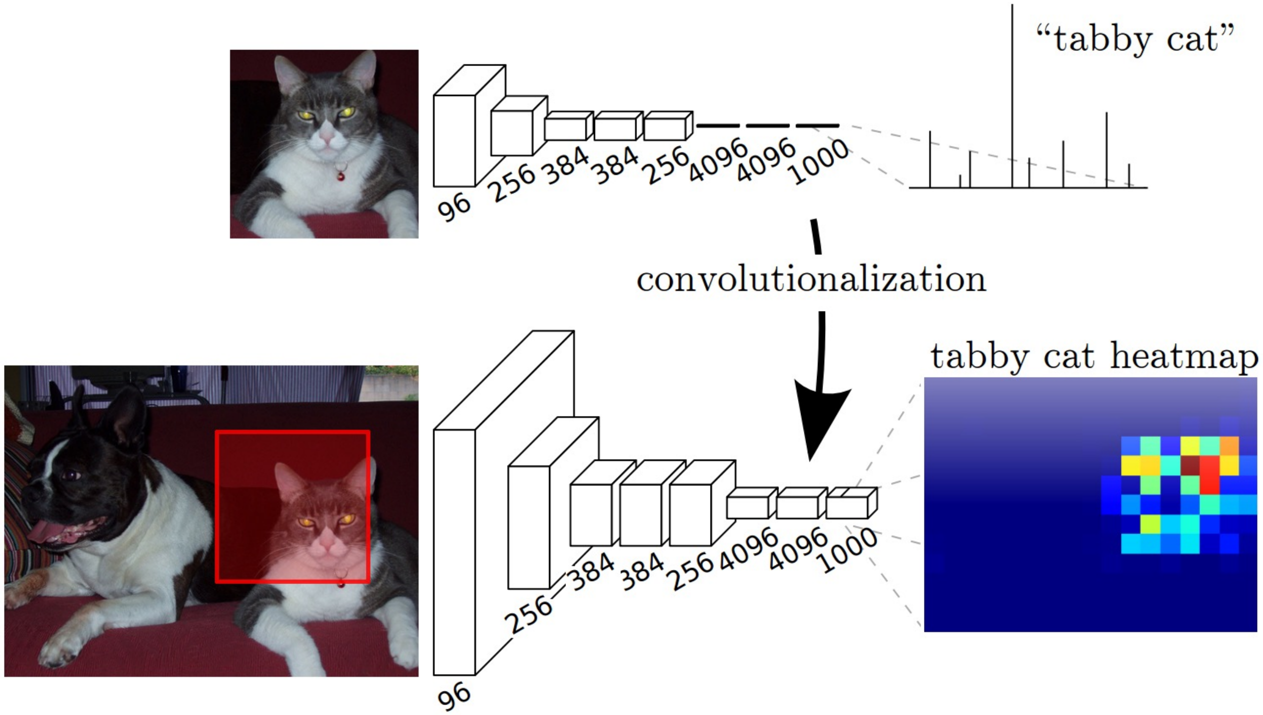
Shape of cancer cell is very helpful in supporting doctors in cancer diagnosis

Types of image segmentation

Object Detection | Semantic Segmentation | **Instance Segmentation**

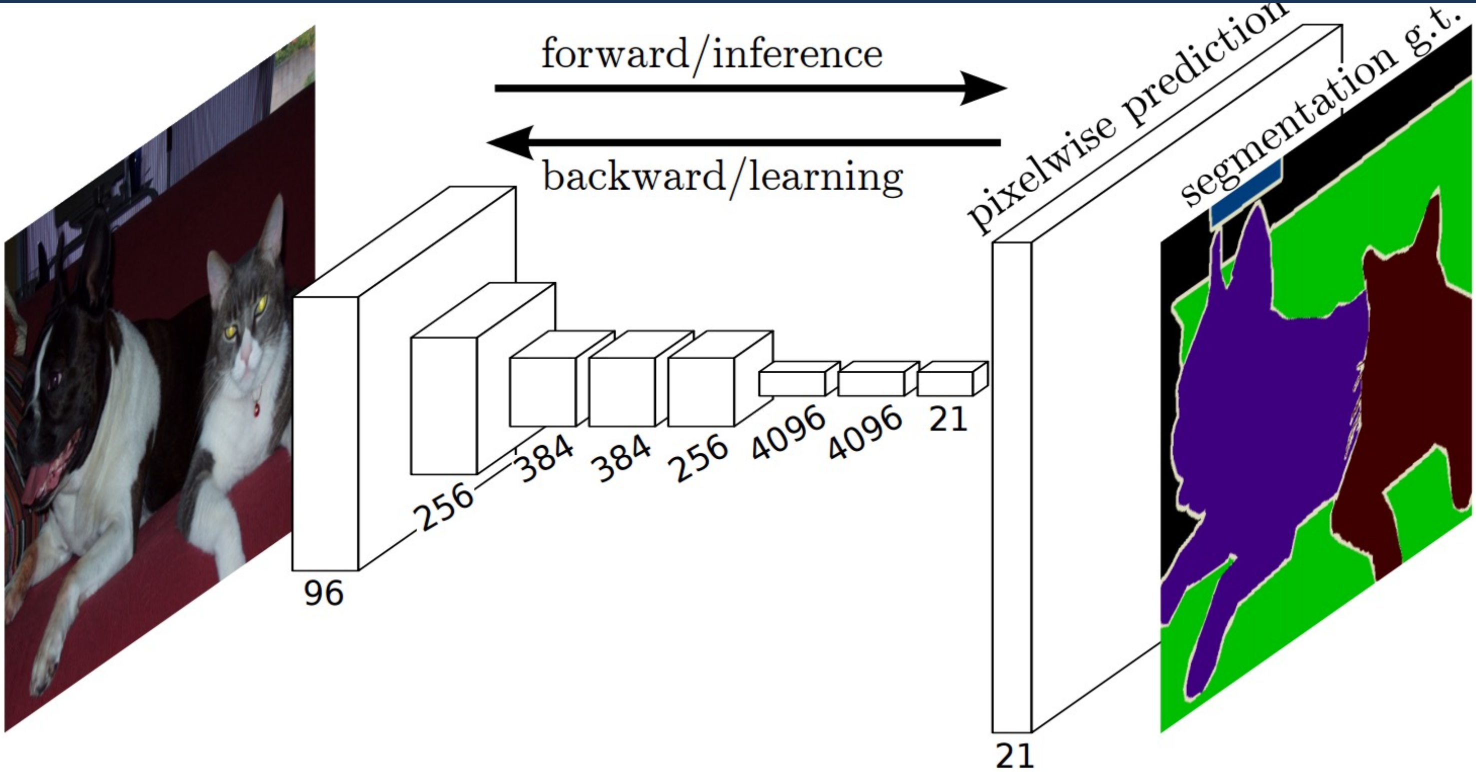
- Semantic segmentation: perform segmentation on each class
- Instance segmentation: perform segmentation on each object of the class
- → Depend on the problem to apply semantic or instance segmentation

Convolutionize



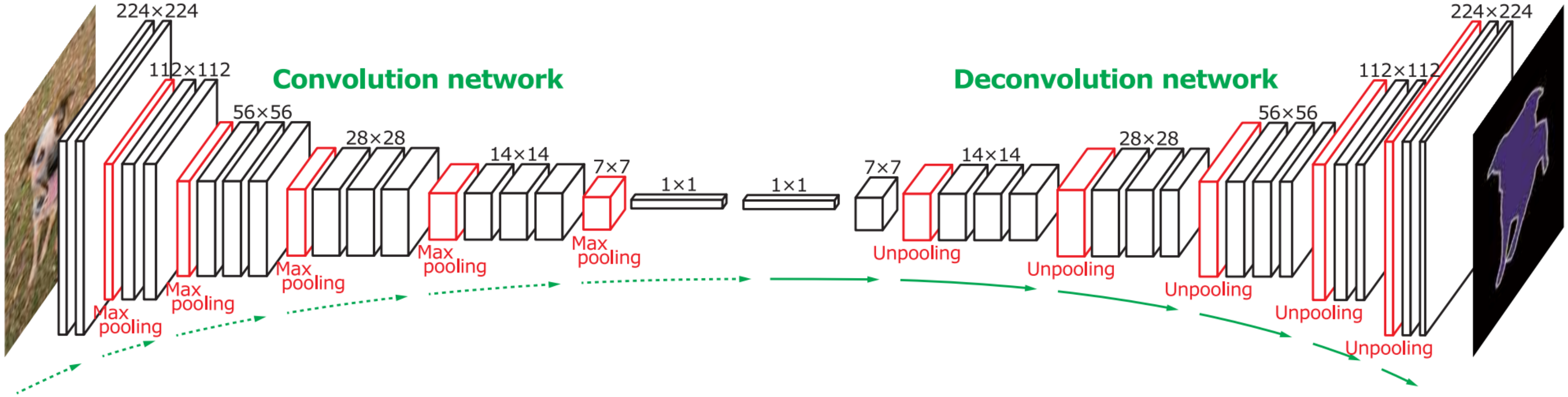
- Slide the network with an input of (224, 224) over a large image. Output of varying spatial size
- Convolutionize: change Dense (4096, 1000) to 1 x 1 convolution, with 4096 input and 1000 output channels
- Give a coarse segmentation (no extra supervision)

Fully Convolutional Network

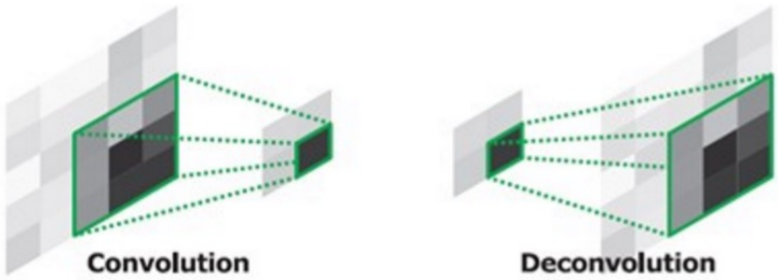


- Predict / backpropagate for every output pixel
- Aggregate maps from several convolutions at different scales for more robust results

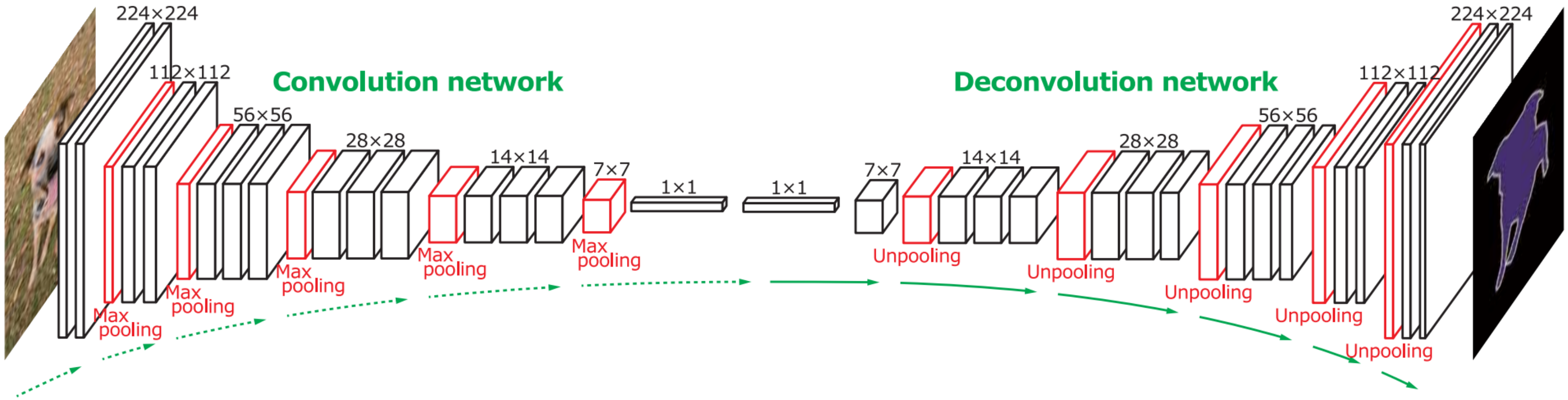
Deconvolution



- Deconvolution: transposed convolutions

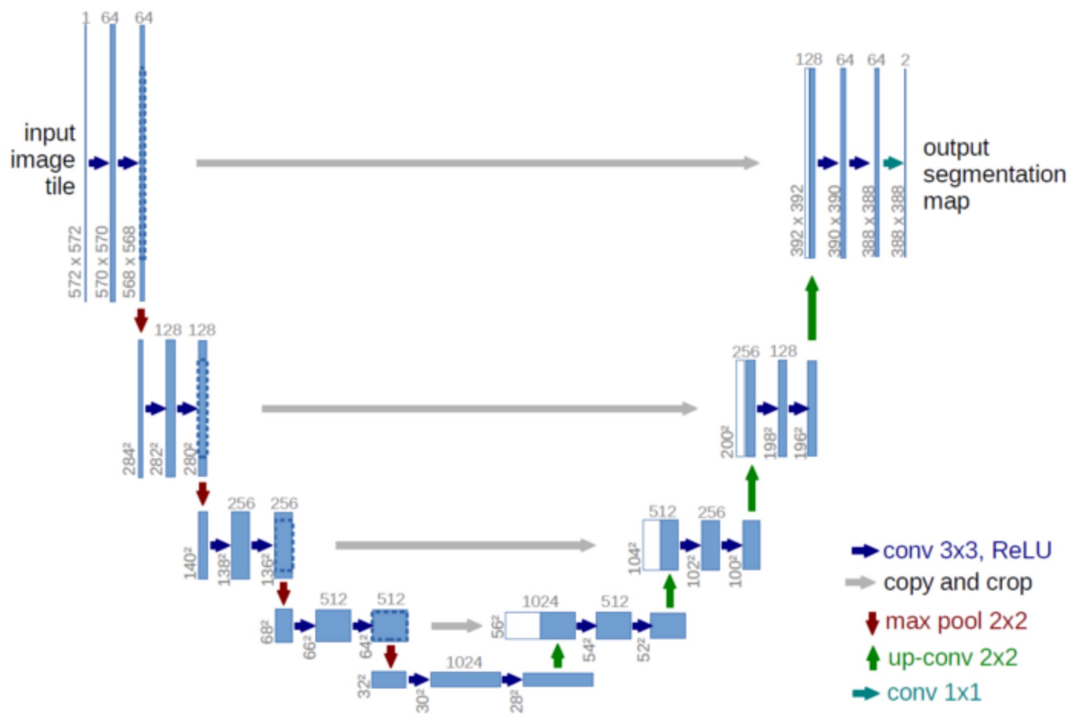


Deconvolution



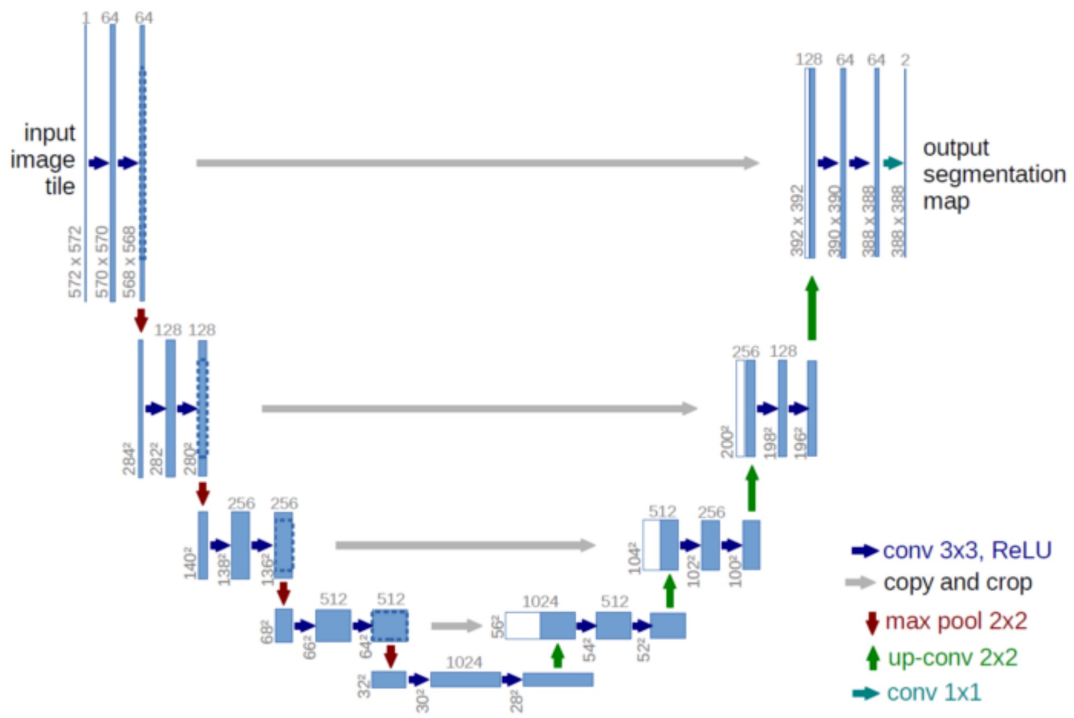
- Skip connections between corresponding convolution and deconvolution layers
- sharper masks by using precise spatial information (early layers)
- better object detection by using semantic information (late layers)

U-Net for semantic segmentation



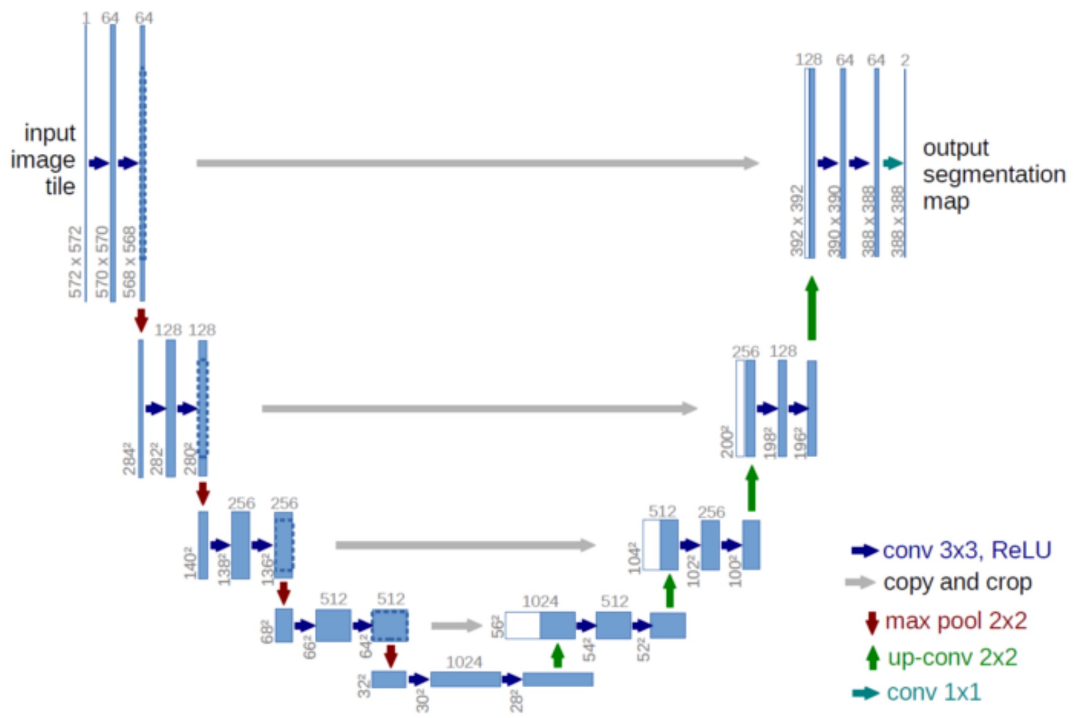
- The left part of U-Net is called encoder part
- The right part of U-Net is called decoder part

U-Net for semantic segmentation



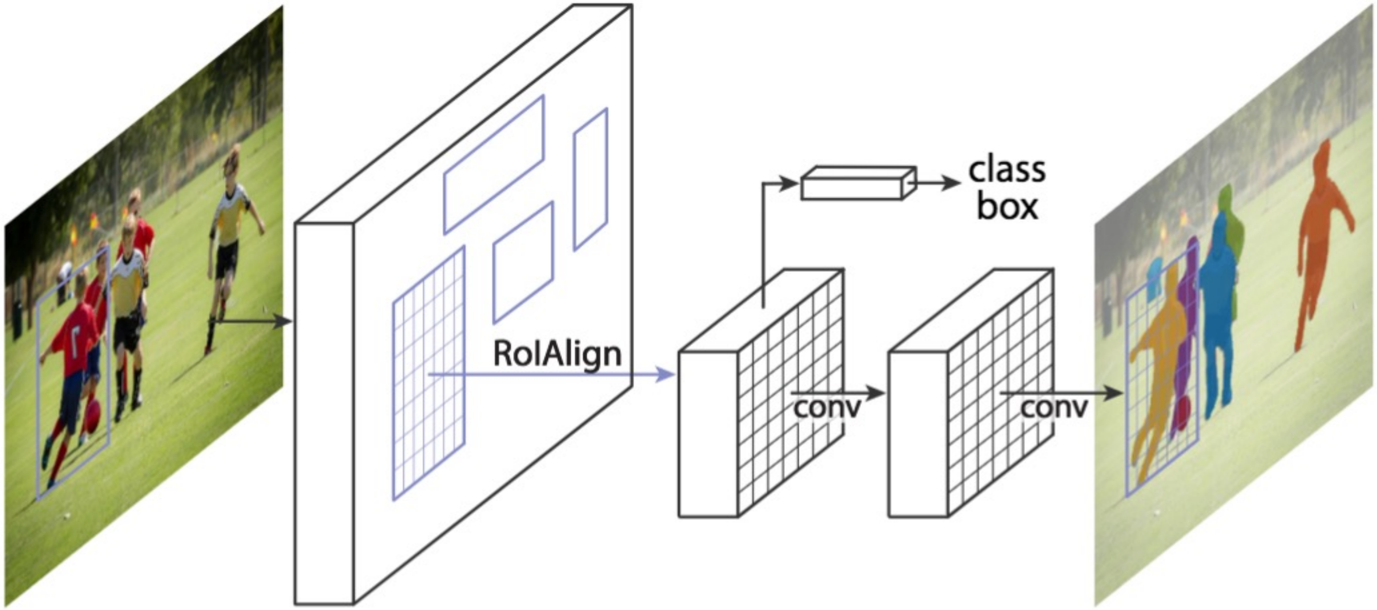
- Each blue box corresponds to a multi-channel feature map
- # of channel is denoted on top of the box
- width and height are denoted at the lower left edge of the box

U-Net for semantic segmentation



- White boxes represent copied feature maps
- The arrows denote different operations

Mask R-CNN



- Faster-RCNN architecture with a third, binary mask head

More Study

Go to this website to study more if you prefer:

- https://paperswithcode.com/area/computer-vision