METODE INTELIGENTE DE REZOLVARE A PROBLEMELOR REALE

Laura Dioşan
Tema 5

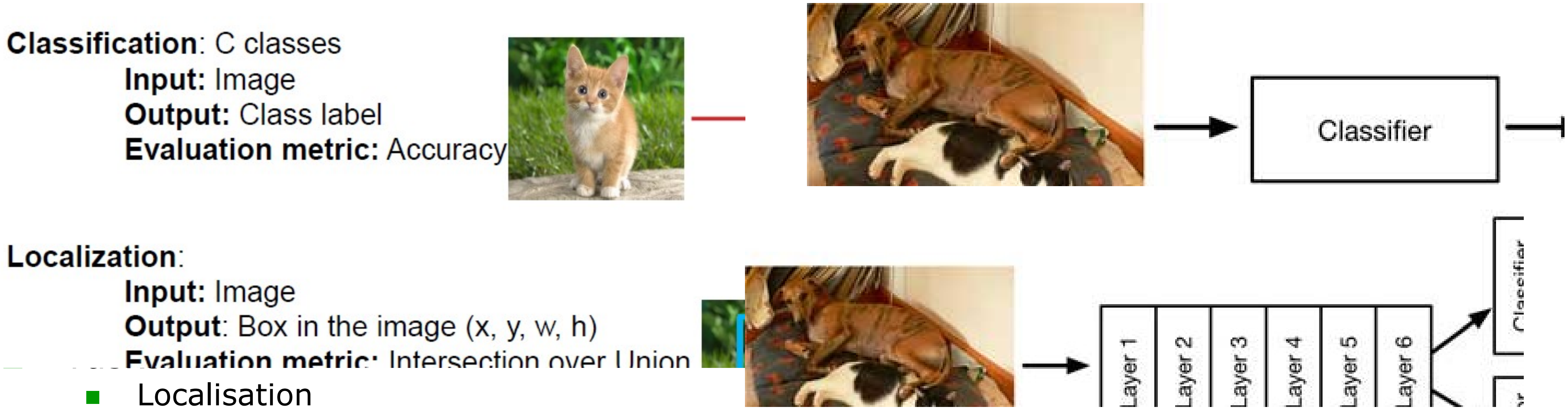
Automatic image processing

- Image classification
  - Does an image contain object X? [yes/no]
- Image detection and segmentation
  - Does an image contain object X? [yes/no]
  - Where is the object X? ➔ Location of the object
    - Pixel-based granularity ➔ semantic segmentation
    - Object-based granularity ➔ object detection
      - Single object
      - Multiple objects ➔ instance segmentation
  - Which object does this image contain? [where?]

  - Aprox. localisation (Bounding box)
  - Accurate localisation (contour) → Segmentation

Image detection

- Image detection
  - Identify which object appears and where the object appears in an image



Classification: C classes

Input: Image

Output: Class label

Evaluation metric: Accuracy

Localization:

Input: Image

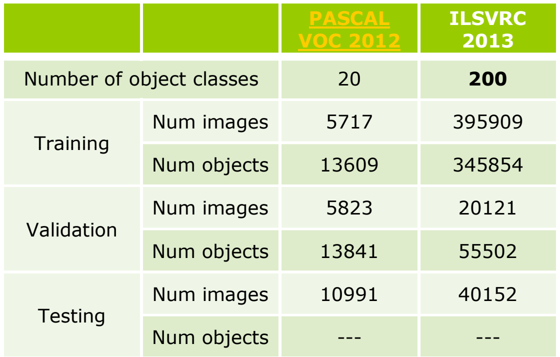
Output: Box in the image (x, y, w, h)

Evaluation metric: Intersection over Union

  - Localisation
    - Find a fixed number of objects (one or many)
    - L2 regression from CNN features to box coordinates
    - Much simpler than detection
    - Overfeat: Regression + efficient sliding window with FC -> conv conversion
    - Deeper networks do better
  - Detection
    - Find a variable number of objects by classifying image regions
    - Before CNNs: dense multiscale sliding window (HoG, DPM)
    - Avoid dense sliding window with region proposals
    - R-CNN: Selective Search + CNN classification / regression
    - Fast R-CNN: Swap order of convolutions and region extraction
    - Faster R-CNN: Compute region proposals within the network
    - Deeper networks do better

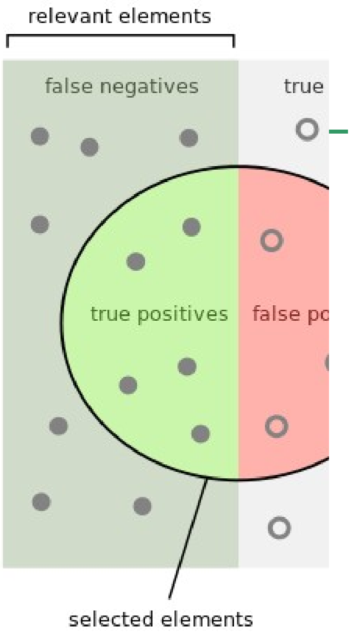
Image detection

- Databases
  - Pascal VOC http://host.robots.ox.ac.uk/pascal/VOC/
    - 2005 – image detection task (4 classes, 1578 images, 2209 objects)
    - 2006 – image detection task (10 classes, 2618 images, 4754 objects)
    - …
    - 2012 – image detection task (20 classes, 11 530 images, 6929 objects)
  - ImageNet http://www.image-net.org/
    - 2010 – image classification task only (1000 classes, 14,197,122 images,)
    - 2011 – classification and localisation task
    - 2012 = 2011 & **Fine-grained classification**
    - 2013 – detection task (200 classes)
    - 2014, 2015, 2016, 2017 – detection task (200 classes)
  - COCO http://cocodataset.org/#home
    - 2015 – detection task (more than 200,000 images and 80 object categories)
    - 2019 – detection task (more than 200,000 images and 80 object categories)



		PASCAL VOC 2012	ILSVRC 2013
Number of object classes		20	**200**
Training	Num images	5717	395909
	Num objects	13609	345854
Validation	Num images	5823	20121
	Num objects	13841	55502
Testing	Num images	10991	40152
	Num objects	---	---

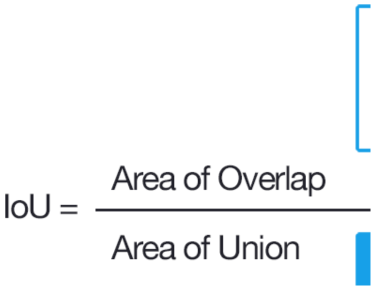
Image detection



Remember:

- Classification → Problem specification
  - Input: image
  - Output: class
    - Label
    - Class probability
  - Evaluation metric: accuracy, precision, recall

- Localisation → Problem specification
  - Input: image
  - Output: bounding box's
    - Class probability
    - coordinates
      - $(x_{corner}, y_{corner}, w, h)$
      - $(x_{corner1}, y_{corner1}, x_{corner2}, y_{corner2})$
      - $(x_{center}, y_{center}, w, h)$
    - Confidence score
  - Evaluation metric:
    - intersection over union
    - average precision



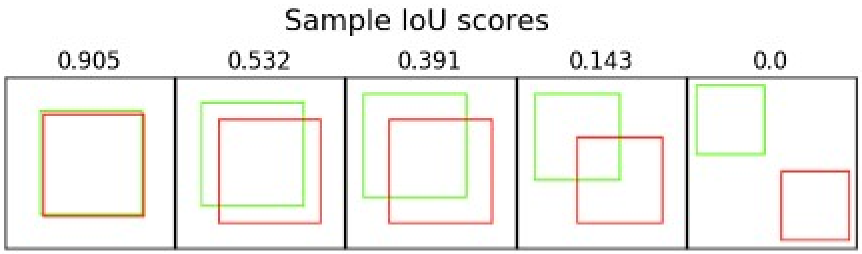
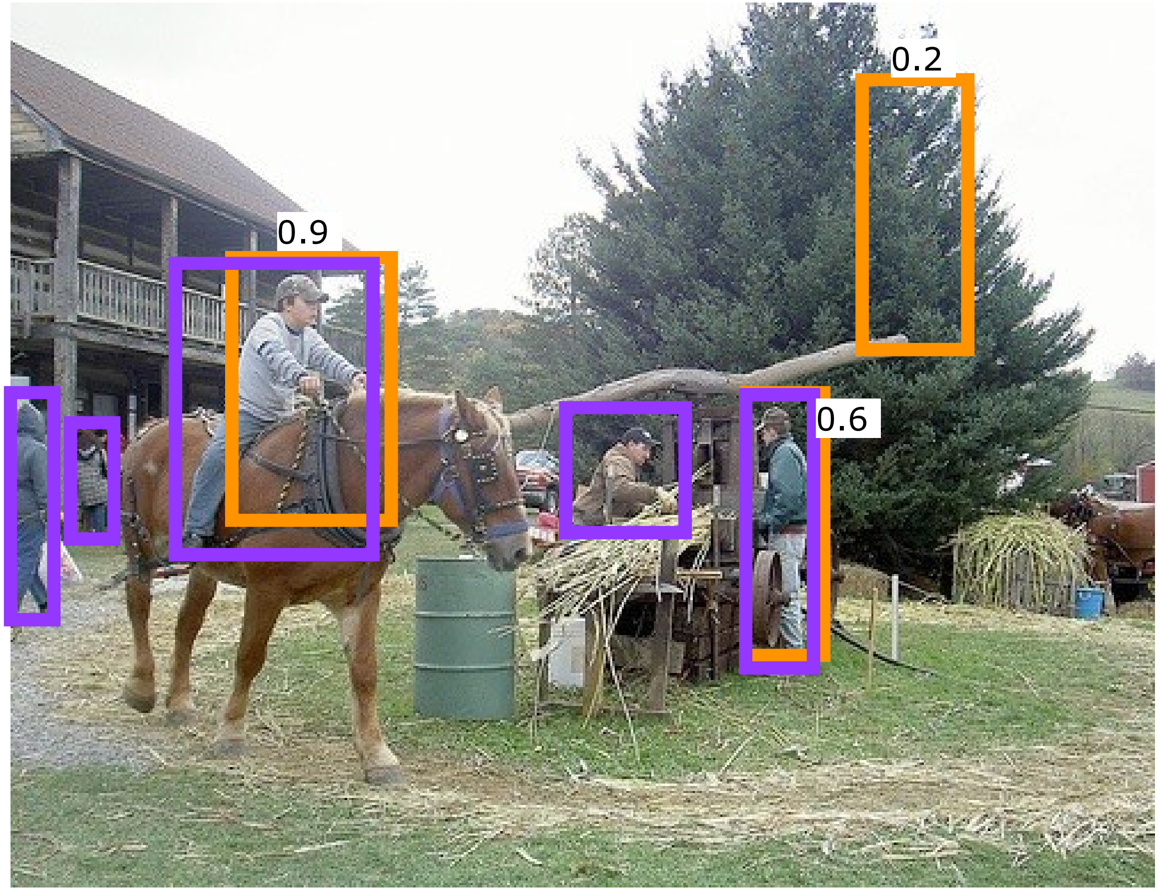


Image detection

- Localisation → Evaluation

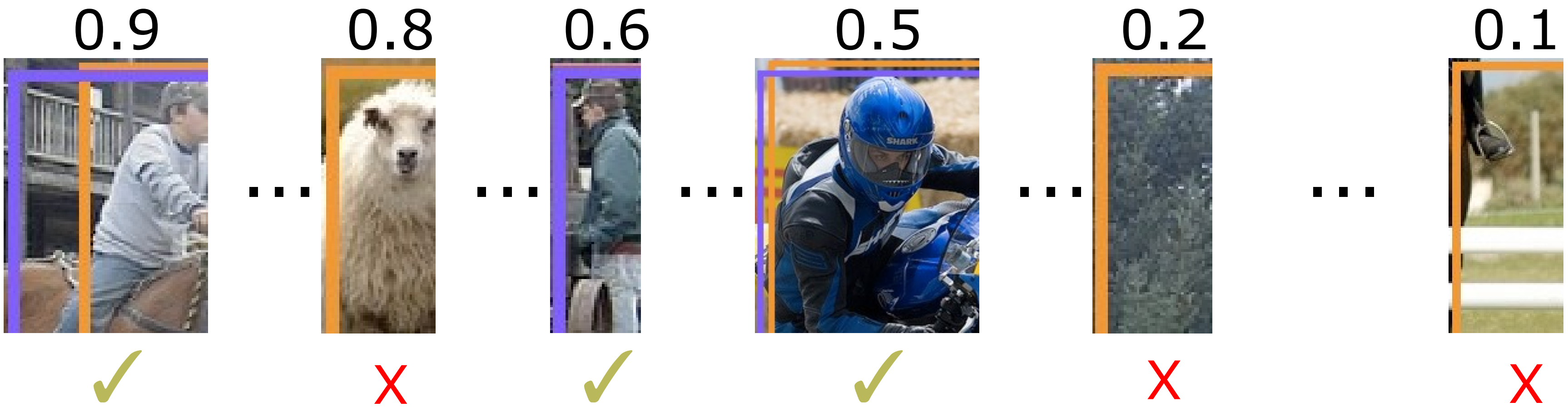


🟧 ‘person’ detector predictions

🟪 ground truth ‘person’ boxes

Image detection

- Localisation → Evaluation
  - Sort by confidence

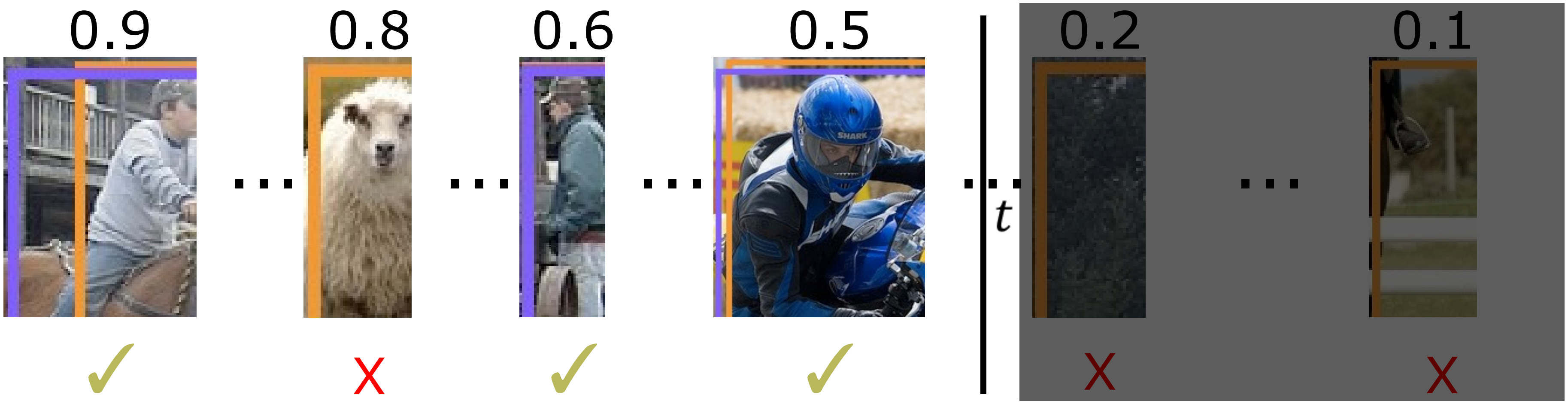


true positive
(high overlap)

false positive
(no overlap,
low overlap, or
duplicate)

Image detection

- Localisation → Evaluation
  - Sort by confidence

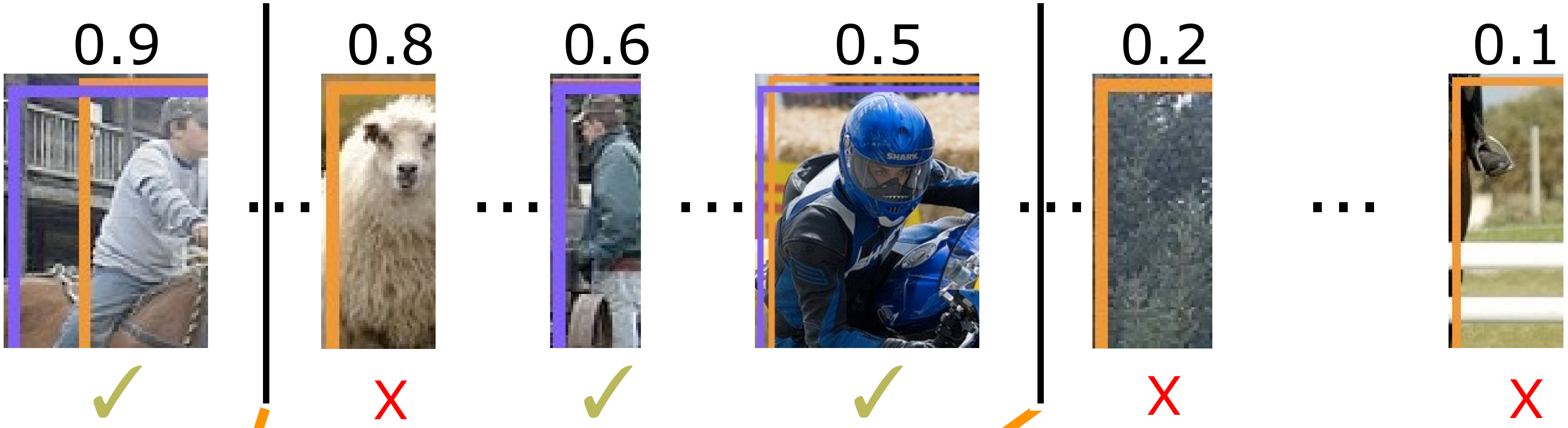


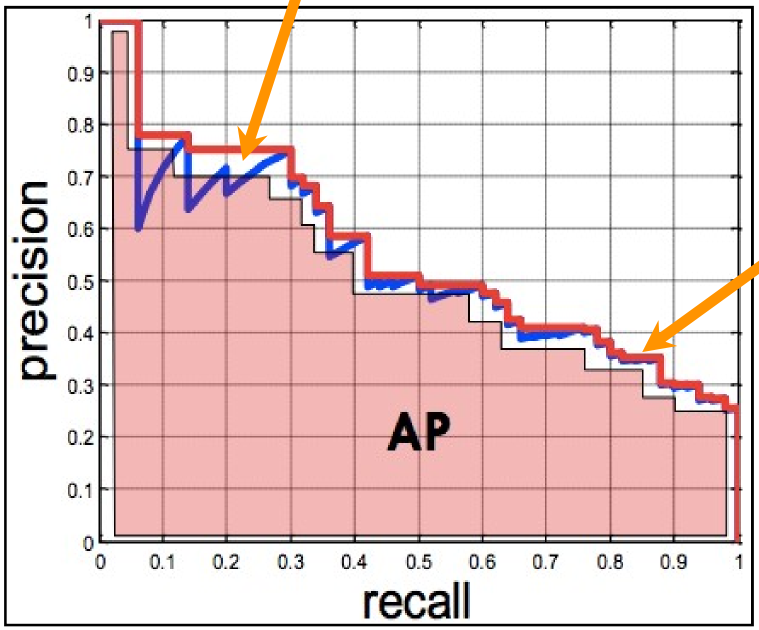
  - Precision = TP@t / (TP@t + FP@t) $\frac{\checkmark}{\checkmark + X}$

  - Recall = TP@t / #GroundTruthObjects

Image detection

- Localisation → Evaluation



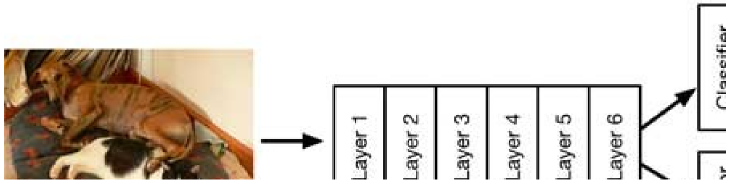


Average Precision (AP)
0% is worst
100% is best

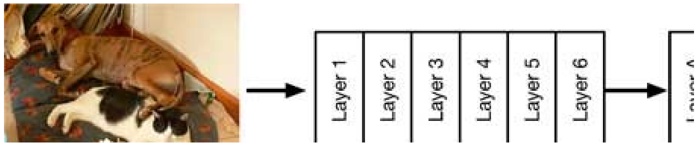
mean AP over classes (mAP)

Image detection

- Algorithms
  - Classification and localisation
    - Localisation as regression problem (predict bb’s coordinates)
    - Localisation as classification (sliding window and classify each window)
  - Object detection → as classification
    - Many positions and scales
      - HOG +SVM
      - DPM
    - 2-stages detectors
      - A model proposes a set of regions
        - Selective search, bing, superpixels, etc.
        - Regional proposal network
      - A classifier processes the proposed regions
        - R-CNN
        - SPP net
        - Fast R-CNN
        - Faster R-CNN
        - …

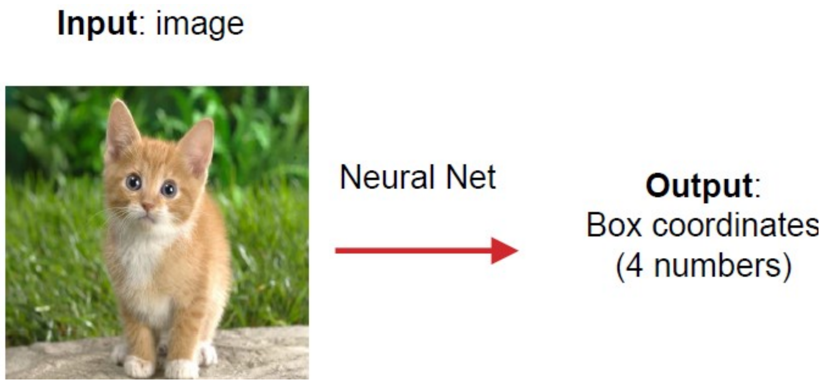


- 1-stage detectors ➔ Run classifier over a dense sampling of possible locations
  - assigning each bounding box detector to a specific position in the image
  - YOLO (v1, v2, v3), SSD, RetinaNet



Classification and localisation

- Localisation = regression problem
  - Predict the bounding-box coordinates
    - Input:
      - Images
    - Output:
      - $(x_{left}, x_{right}, y_{top}, y_{bottom})$ - coordinates of the bounding box edges.



Input: image

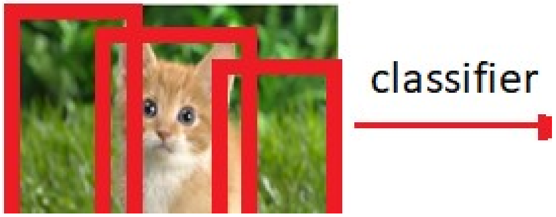
Neural Net

Output:
Box coordinates
(4 numbers)

Classification and localisation

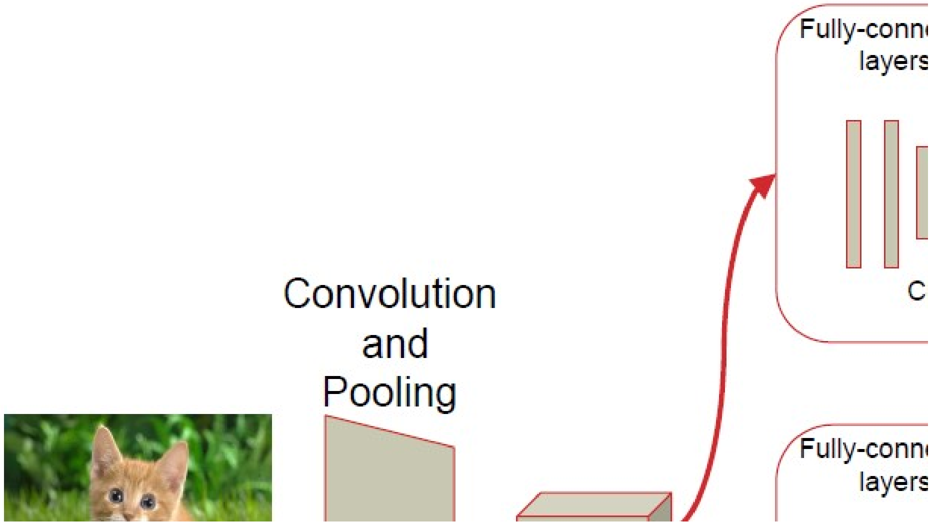
- Localisation = classification problem
  - Sliding window & classification of various patches
    - Input:
      - Images (transformed in a set of patches)
    - Output:
      - Label of each patch
      - Non-max supression processing
      - Resulted bounding box

Input: image



Classification and localisation

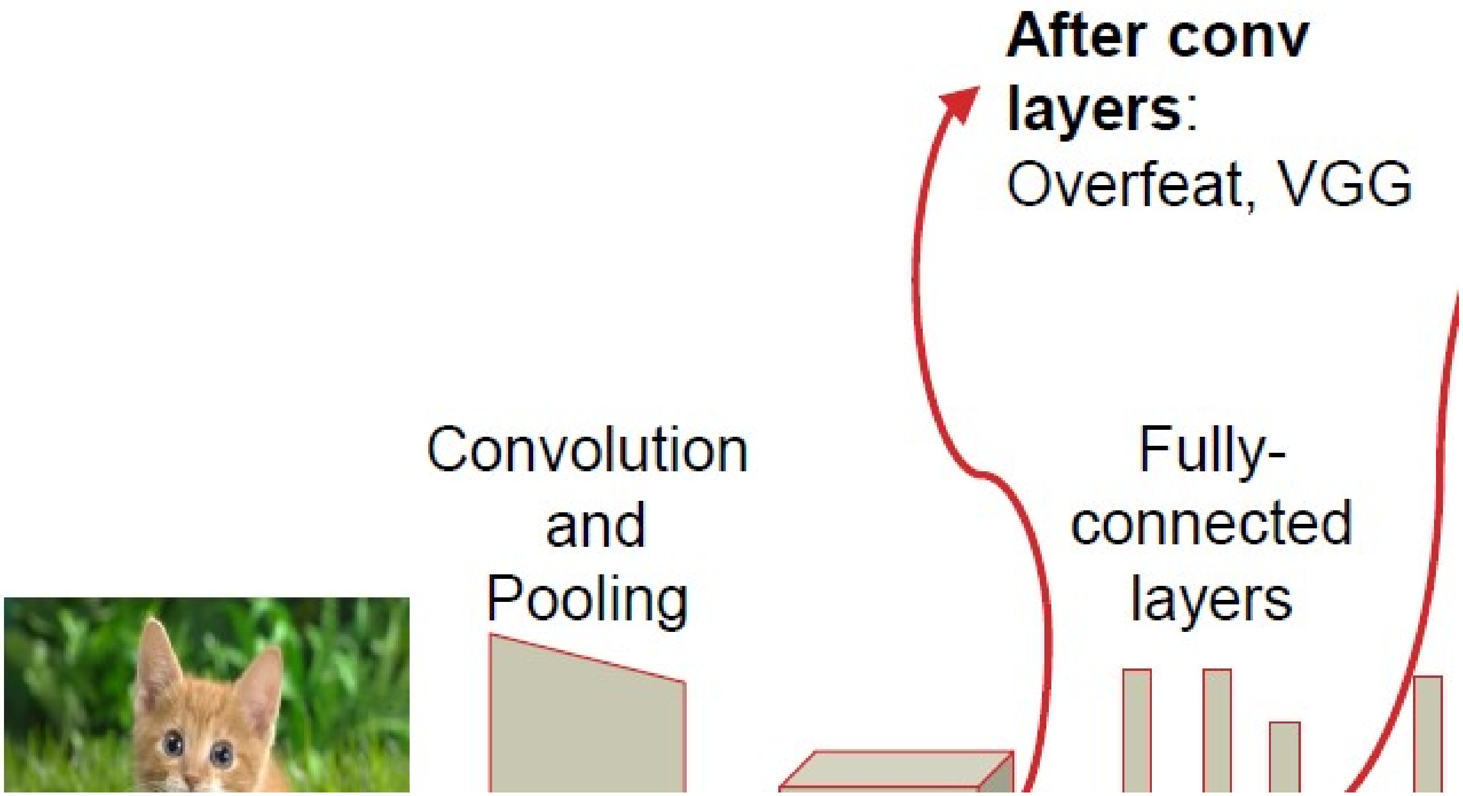
- As regression problem
  - Training
    - Train (or use a pretrained) a classification model
    - Attach new fully-connected "regression head" to the network
    - Train the regression head only with SGD and L2 loss



  - Testing
    - At test time use both heads

Classification and localisation

- As regression problem
  - Where to attach the regressor head?



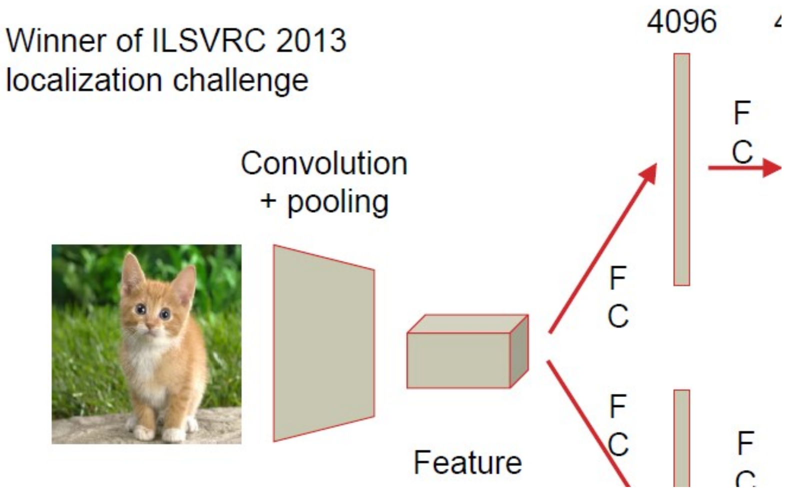
After conv layers: Overfeat, VGG

Convolution and Pooling

Fully-connected layers

Classification and localisation

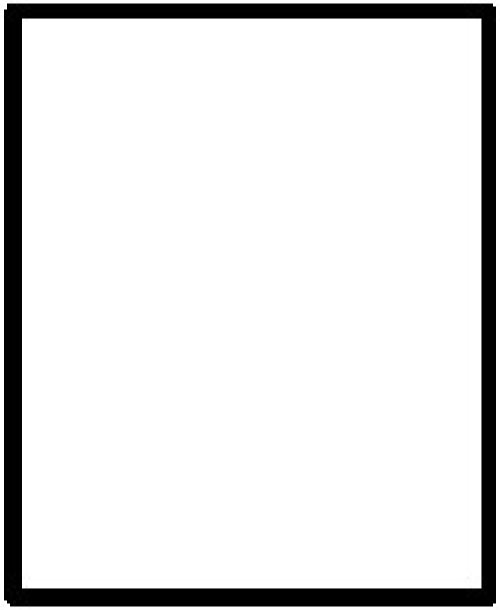
- As classification problem → sliding window
  - see Overfeat https://arxiv.org/pdf/1312.6229.pdf
  - At the training time
    - Train a CNN model (e.g. AlexNet, VGG, …) on the image classification task.
    - Replace the top classifier layers by a regression network and train it to predict object bounding boxes at **each spatial location and scale**.
      - The regressor is class-specific, each generated for one image class.
    - Loss:
      - The regressor is trained to minimize L2 norm between generated bounding box and the ground truth for each training example.
  - At the detection time,
    - Perform classification at each location using the pretrained CNN model.
    - Predict object bounding boxes on all classified regions generated by the classifier.
    - Merge bounding boxes with sufficient overlap from localization and sufficient confidence of being the same object from the classifier.



Winner of ILSVRC 2013 localization challenge

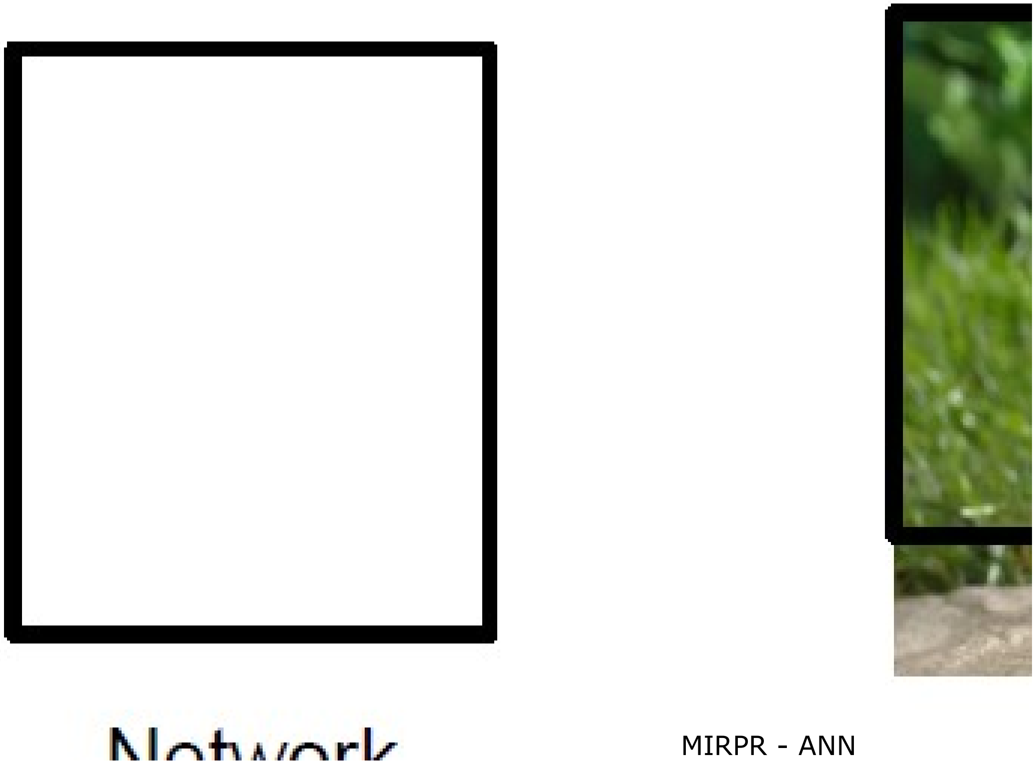
Classification and localisation

- As classification problem → sliding window
  - Network predicitions
    - Fixed input = patch of the image



Classification and localisation

- As classification problem → sliding window
  - Network predicitions
    - Fixed input = patch of the image

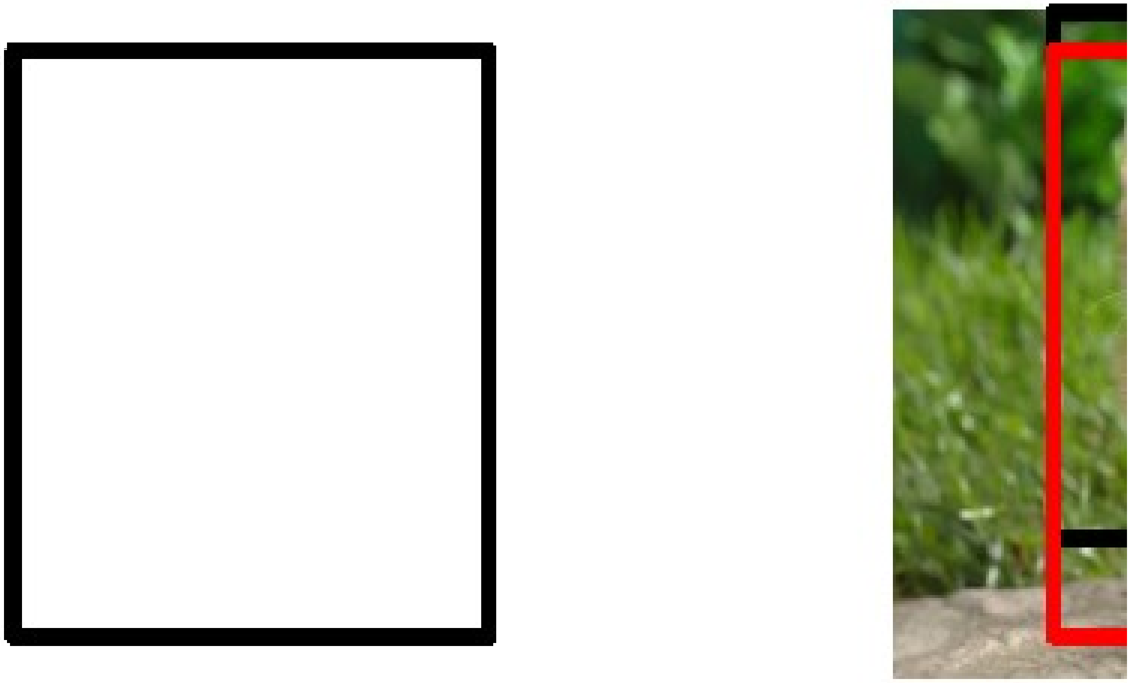


0.5	

Classification
score Prob(cat)

Classification and localisation

- As classification problem → sliding window
  - Network predicitions
    - Fixed input = patch of the image

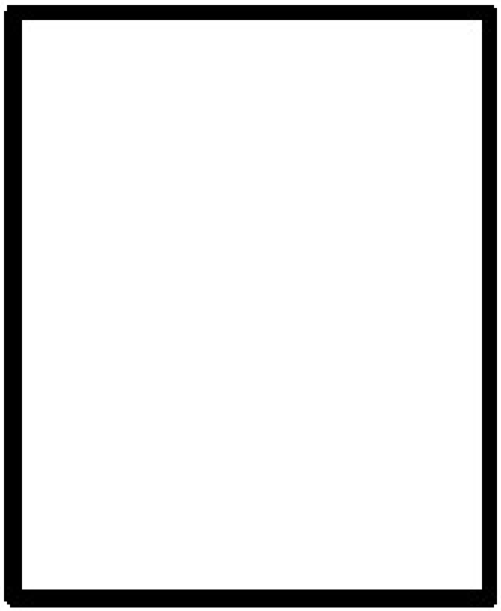


0.5	0.75

Classification score Prob(cat)

Classification and localisation

- As classification problem → sliding window
  - Network predicitions
    - Fixed input = patch of the image

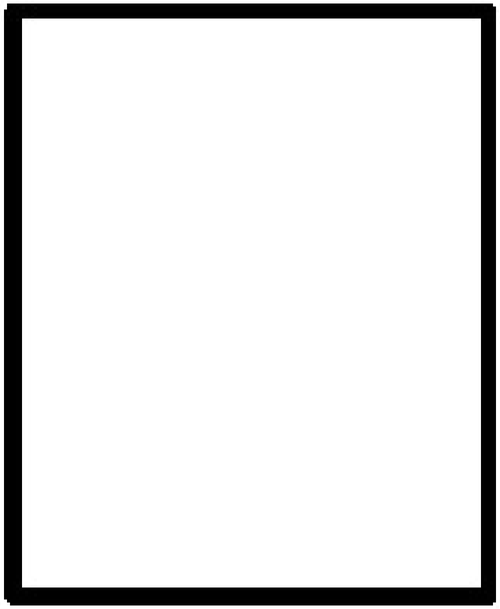


0.5	0.75
0.6	

Classification score Prob(cat)

Classification and localisation

- As classification problem → sliding window
  - Network predicitions
    - Fixed input = patch of the image

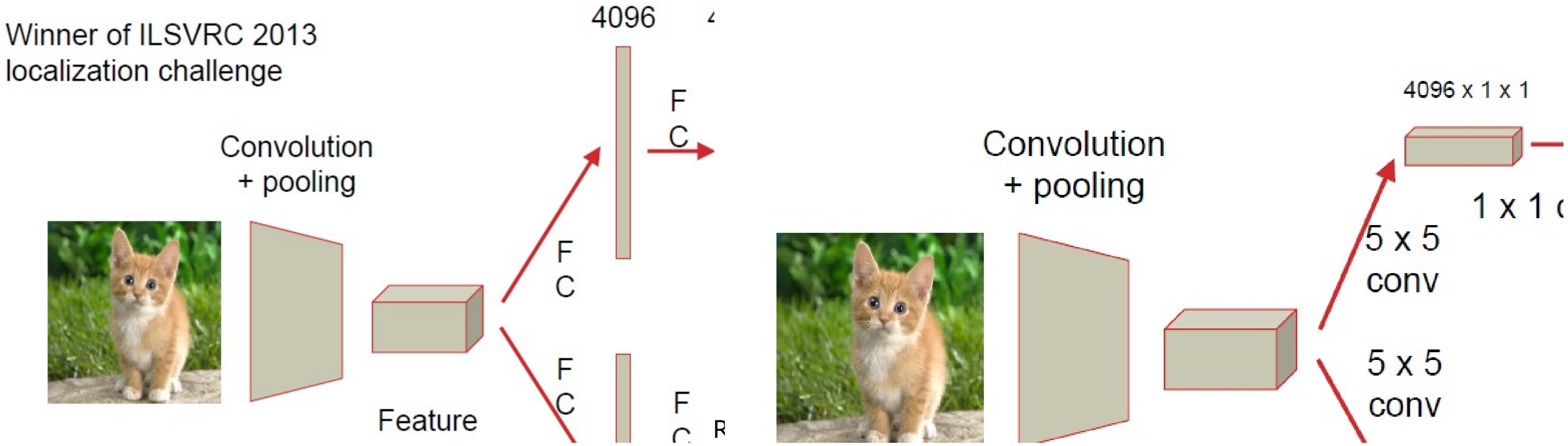


0.5	0.75
0.6	0.8

Classification score Prob(cat)

Classification and localisation

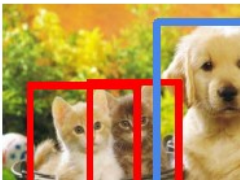
- As classification problem → sliding window
  - Run classification + regression network at multiple locations on a high-resolution image
    - One regressor for each class
  - Convert fully-connected layers into convolutional layers for efficient computation
  - Combine classifier and regressor predictions across all scales for final prediction



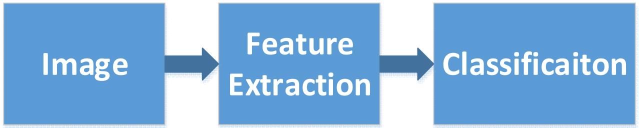
Winner of ILSVRC 2013 localization challenge

Object detection

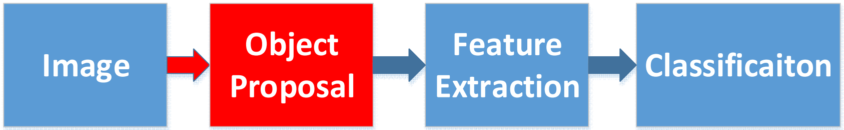
Object Dete



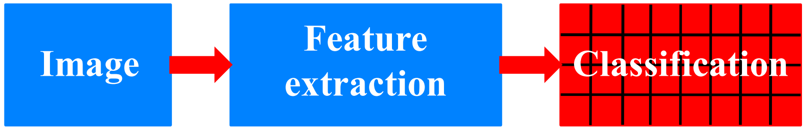
- Algorithms → Classification and localisation
- Algorithms → **Object detection → as classification**
  - Many/all positions and scales
    - Haar-like features
    - HOG +SVM
    - DPM



  - Using an object proposal
    - 2-stages detectors

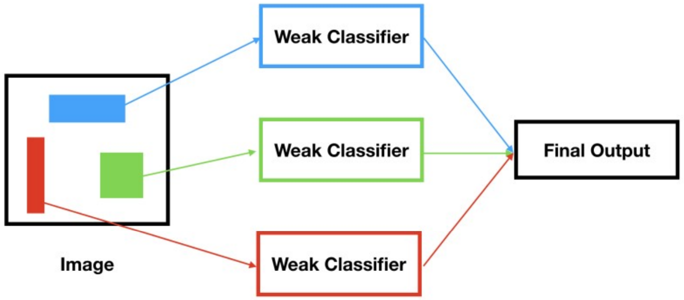


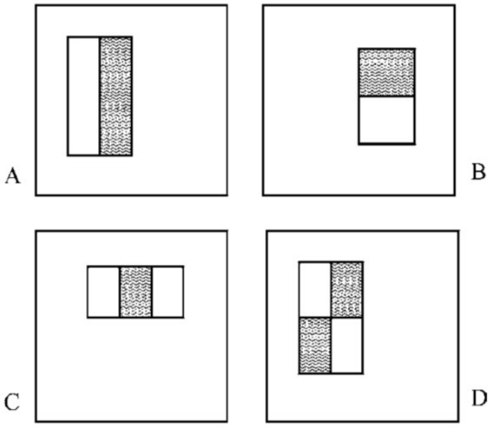
      - A model proposes a set of regions (instead of sliding window)
        - Selective search, bing, superpixels, etc.
        - Regional proposal network
      - A classifier processes the proposed regions
        - R-CNN
        - Fast R-CNN
        - Faster R-CNN
    - 1-stage detectors → Run classifier over a dense sampling of possible locations
      - assigning each bounding box detector to a specific position in the image
      - YOLO (v1, v2, v3), SSD, RetinaNet

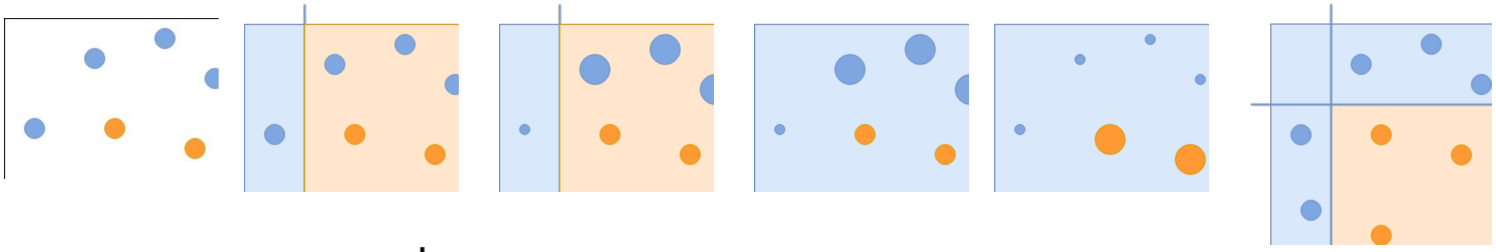


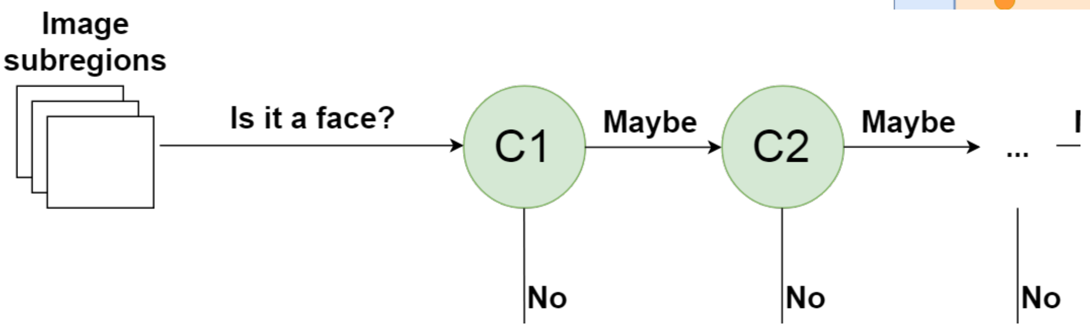
Object detection

- Many positions and scales → Haar features
  - See Viola and Jones, 2001



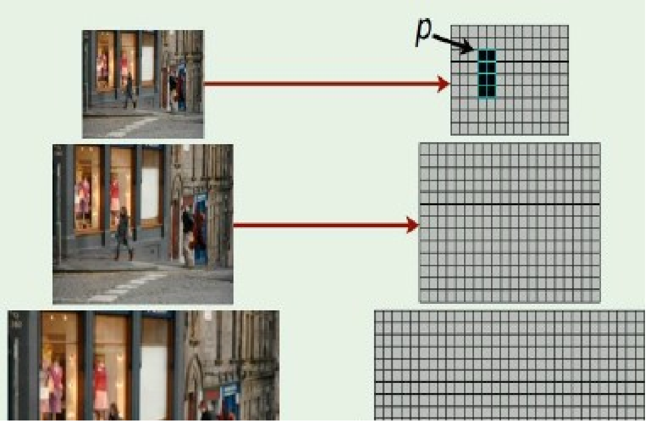






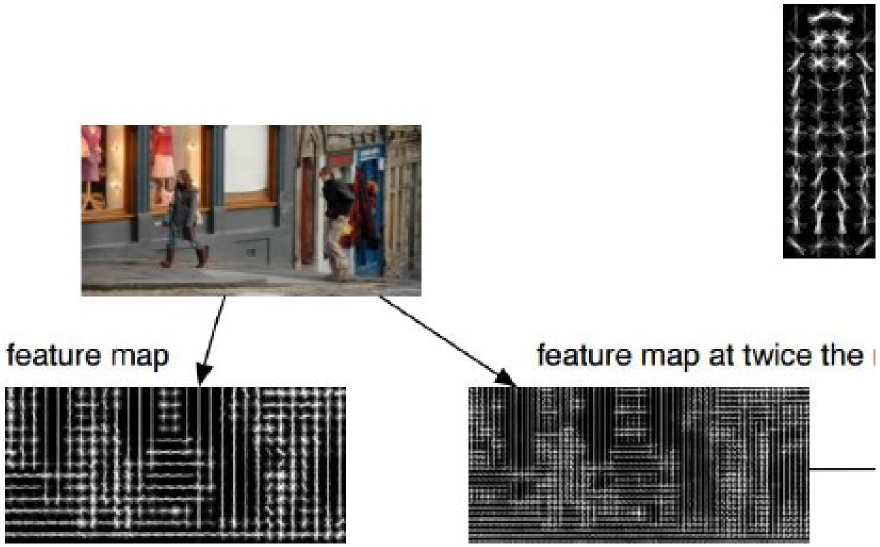
Object detection

- Many positions and scales → HOG
  - See https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
  - Compute HOG of the whole image at multiple resolutions
  - Classify/score each subwindow of the featured pyramid
  - Apply non-maximum supression (NMS)
    - Input:
      - More bounding boxes B = {b1, b2, …, bN}
      - More classification scores S = {s1, s2, …sN}
      - A classification threshold CT
      - An overlapping threshold OT
    - Output:
      - Best bounding box
    - Algorithm
      - Discard all boxes with si < CT
      - While B != empty
        - Pick bm, where m = argmax(S)
        - Discard from B that boxes of IoU with bm > OT
    - Note:
      - Class-wise operator
      - Soft NMS is more efficient https://arxiv.org/pdf/1704.04503.pdf
        - instead of completely removing the proposals with high IOU and high confidence, reduce the confidences of the proposals proportional to IOU value



Object detection

- Detection as classification → Deformable Parts Model
  - See http://cs.brown.edu/people/pfelzens/papers/lsvm-pami.pdf
  - Takes the HOG's idea a little further
  - Instead of one rigid HOG model, we have multiple HOG models in a spatial arrangement
  - One root part to find first and multiple other parts in a tree structure.



Object detection

- Detection → Evaluation
  - "mean average precision" (mAP)
    - Compute average precision (AP) separately for each class, then average over classes
    - A detection is a true positive if it has IoU with a ground-truth box greater than some threshold (usually 0.5) (mAP@0.5)
    - Combine all detections from all test images to draw a precision / recall curve for each class; AP is area under the curve

Object detection

- Algorithms → Object detection → as classification
  - Many/all positions and scales
    - HOG +SVM, DPM, …

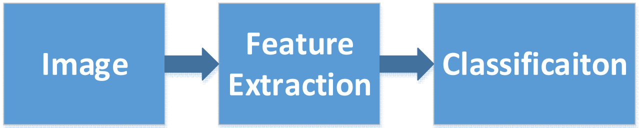


Image → Feature Extraction → Classificaiton

    - Problem
      - Need to test many positions and scales → use a computationally demanding classifier (CNN)

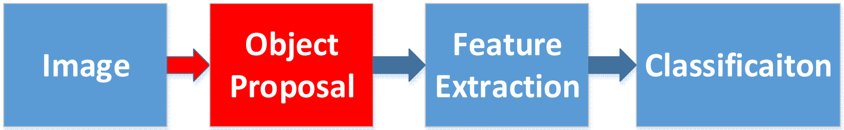
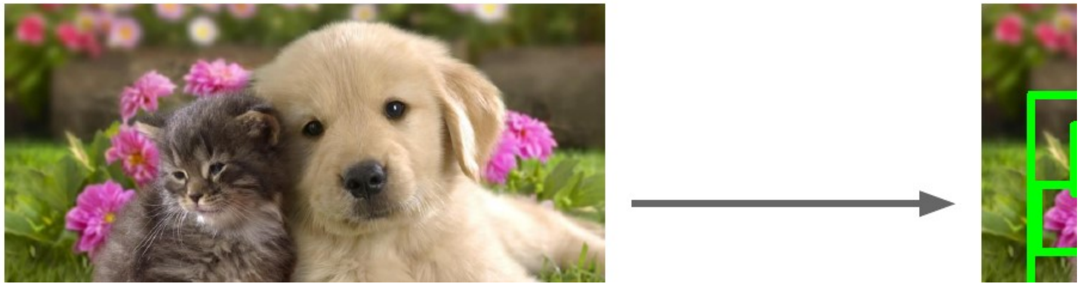


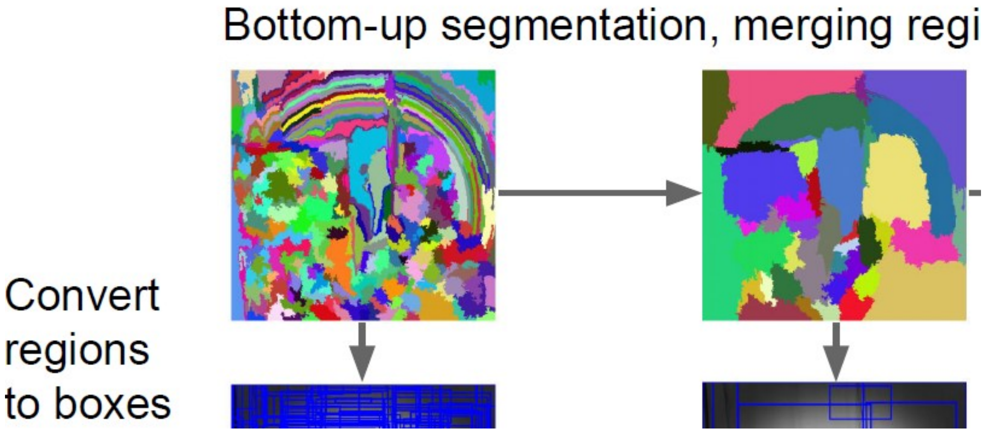
Image → Object Proposal → Feature Extraction → Classificaiton

    - Solution
      - Only look at a tiny subset of possible positions
      - Find "blobby" image regions that are likely to contain objects
      - "Class-agnostic" object detector (bb for each possible class)
      - Look for "blob-like" regions ➔ region-based algorithms



Object detection

- Region proposals
  - Selective search http://www.huppelen.nl/publications/selectiveSearchDraft.pdf



  - Edge boxes https://arxiv.org/pdf/1502.05082.pdf
  - Other methods

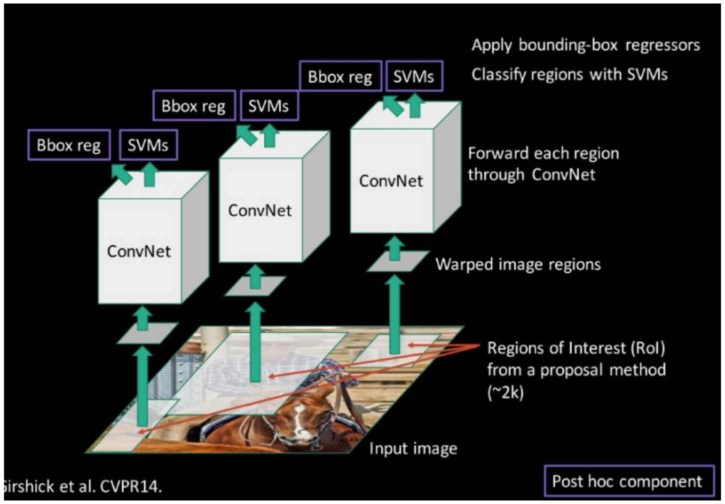
Object detection

- Region-based networks
  - R-CNN (two-stage)
  - SPP-net (two-stage)
  - Fast R-CNN (two-stage)
  - Faster R-CNN (one-stage)
  - R-FCN (two-stage)
  - FPN (two-stage)

Object detection

He already proposed DPM

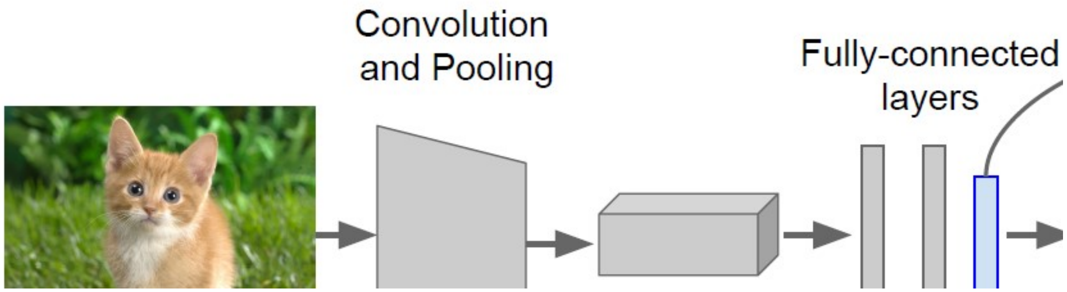
- R-CNN
  - See R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014
  - See https://github.com/rbgirshick/rcnn


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
Post hoc component

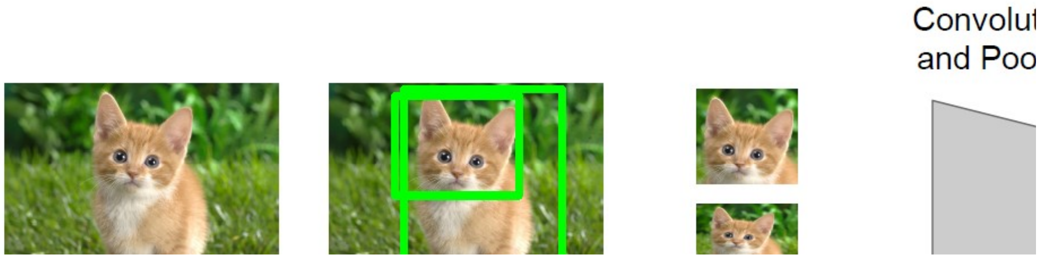
irshick et al. CVPR14.

  - Steps:
    - Extract region proposal from an image by
      - Selective search (super pixel)
      - BING, MCG, ….
    - Compute CNN features
      - Crop
      - Scale
      - Forward propagate through a CNN
        - Use a pre-trained CNN (for classification) [and fine-tune CNN (for detection)]
    - Classify regions (compute labels)
      - Linear classifier (SVM or softmax)
    - Object proposal refinement
      - Linear regression on CNN features → predict corrections to the bounding box

Object detection



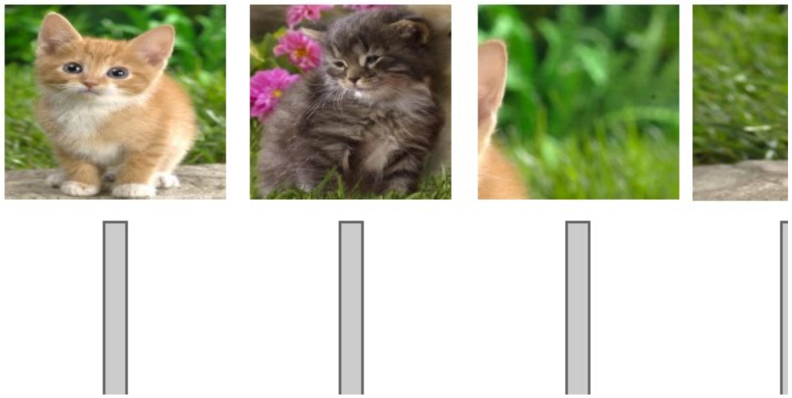
- R-CNN → Training
  - Extract region proposal from an image
    - By various methods (Selective search, BING, MCG, …)
    - Regions are class-independent
    - Regions are of different sizes
  - Train or use a pretrained classification model (e.g. AlexNet on ImageNet)



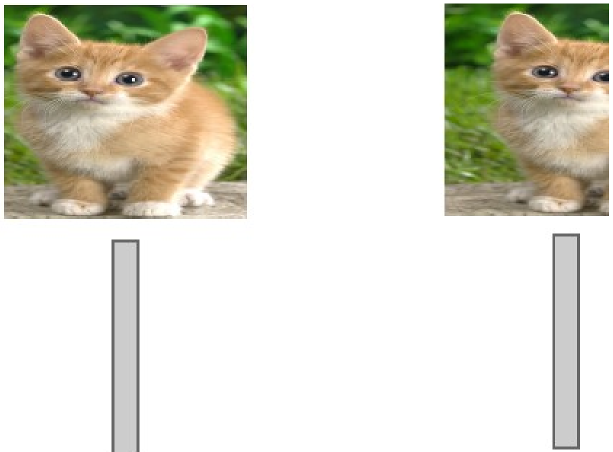
-
  - Fine-tune the classification model for detection (VOC)
    - Instead of 1000 classes use 20 object classes + background
    - Eliminate the final fully-connected layer, reinitialize from scratch
    - Keep training model using positive / negative regions from detection images
    - Compute features
      - Cropping: Extract region proposals for all images
      - For each region
        - warp/scale to CNN input size,
        - run forward through CNN
        - save pool5 features to disk (features are ~200GB for PASCAL dataset!)
    - use a much smaller learning rate
    - use the mini-batch oversamples the positive cases because most proposed regions are just background

Object detection

- R-CNN → Training



- Train one binary SVM per class to classify region features
  - ➔ scores for each proposal
  - Rank the proposal and apply NMS to get the bounding boxes



- BB regression
  - Refine the bounding boxes → reduce the localisation error
  - For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for "slightly wrong" proposals

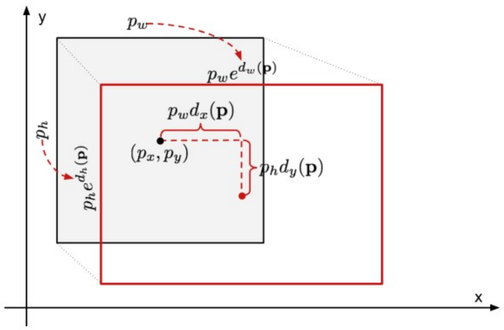
Object detection

- R-CNN → BB regression
  - Inputs:
    - Predicted BB ***p*** = (p_x, p_y, p_w, p_h)
    - Ground Truth BB ***g*** = (g_x, g_y, g_w, g_h)
  - Aim:
    - To learn
      - scale-invariant transformation between two centers
      - log-scale transformation between widths and heights

$$\hat{g}_x = p_w d_x(p) + p_x$$
$$\hat{g}_y = p_h d_y(p) + p_y$$
$$\hat{g}_w = p_w \exp(d_w(p))$$
$$\hat{g}_h = p_h \exp(d_h(p))$$

      - the bounding box correction functions, d_i(***p***) where i∈{x,y,w,h},can take any value between [-∞, +∞]
      - Targets for learn

$$t_x = (g_x - p_x) / p_w$$
$$t_y = (g_y - p_y) / p_h$$
$$t_w = \log(g_w / p_w)$$
$$t_h = \log(g_h / p_h)$$



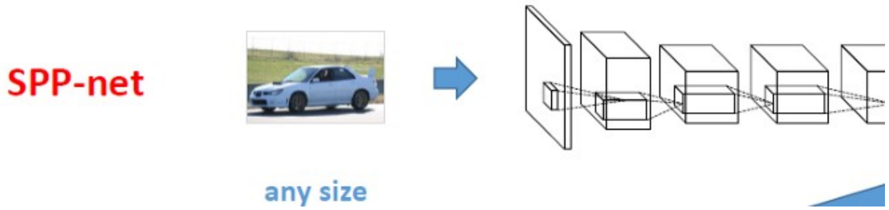
      - minimise SSE loss (L2 loss)

$$Loss_{reg} = \sum_{i \in \{x,y,w,h\}} (t_i - d_i(p))^2 + \lambda \|weight\|^2$$

        - Regularisation term → cross validation (R-CNN paper)
      - not all the predicted bounding boxes have corresponding ground truth boxes
        - only a predicted box with a nearby ground truth box with at least 0.6 IoU is kept for training the bbox regression model

Object detection

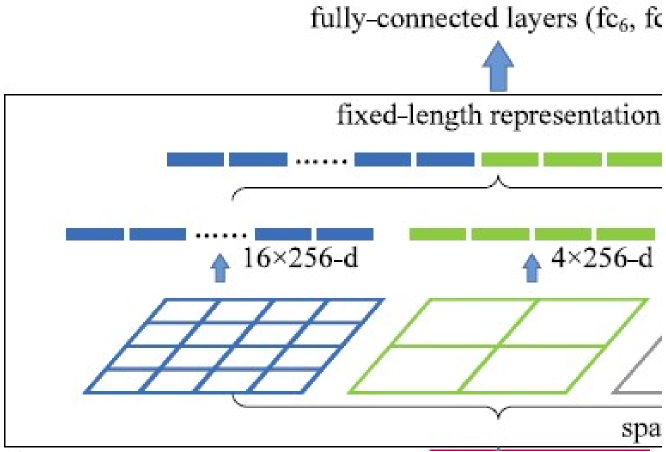
- R-CNN
  - Problems
    - Very slow (at training and testing time)
      - Each region proposal need to be warped to a fixed size
        - Cropping may loss some information about the object
        - Warpping may change the object appearance
      - each region proposal is forwarded into CNN
    - SVM and regressors are post-hoc → CNN features are not updated based on the errors of SVMs and regressors
    - Complex multi-stage trainir



  - Solution
    - Slow testing & fix size constraints ➔ multi-size training (SPP-net)
    - Process image (compute features) before RoI extraction = swap convolution and cropping (Fast R-CNN)

Object detection

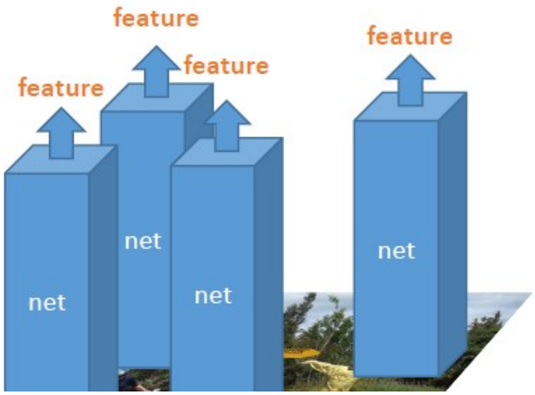
- SPP-net
  - See He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015): 1904-1916 https://arxiv.org/pdf/1406.4729.pdf

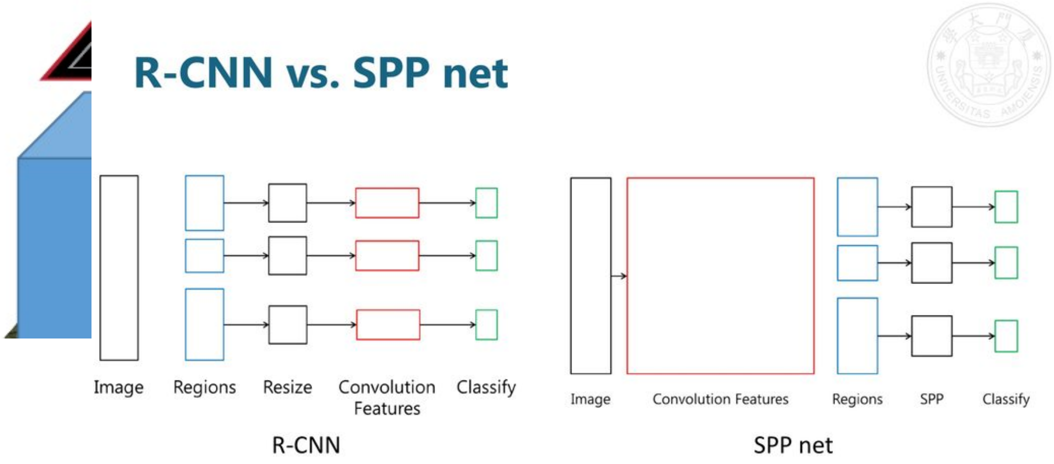

fully-connected layers (fc_6, fc
fixed-length representation
16×256-d
4×256-d
spa

  - Traditional
    - One pooling layer between Conv and FC
  - Spatial Pyramid Pooling
    - multiple pooling layers with different scales
      1. each feature map is pooled to become one value (grey), thus 256-d vector is formed.
      2. each feature map is pooled to have 4 values (green), and form a 4×256-d vector.
      3. each feature map is pooled to have 16 values (blue), and form a 16×256-d vector.
    - The above 3 vectors are concatenated to form a 1-d vector.
    - Finally, this 1-d vector is going into FC layers as usual.

Object detection

- SPP net
  - processes the image at conv layers for only one time
    - R-CNN processes the image at conv layers for 2k times since there are 2k region proposal




R-CNN vs. SPP net

Object detection

- SPP-net
  - Advantages
    - Run CNN once
    - Testing is faster
  - Weaknesses
    - Multi-stage training
    - No CNN fine tuning (frozen)
    - Ad-hoc training objectives
    - Solution → Fast-RCNN (better performance, single stage training, all nets are training)

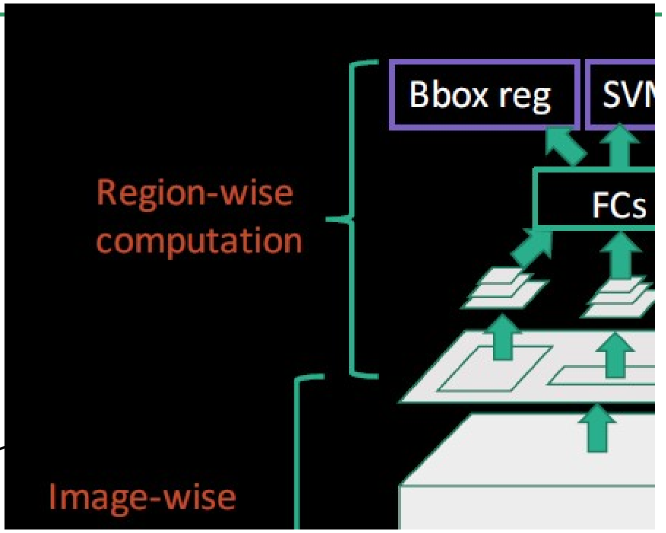
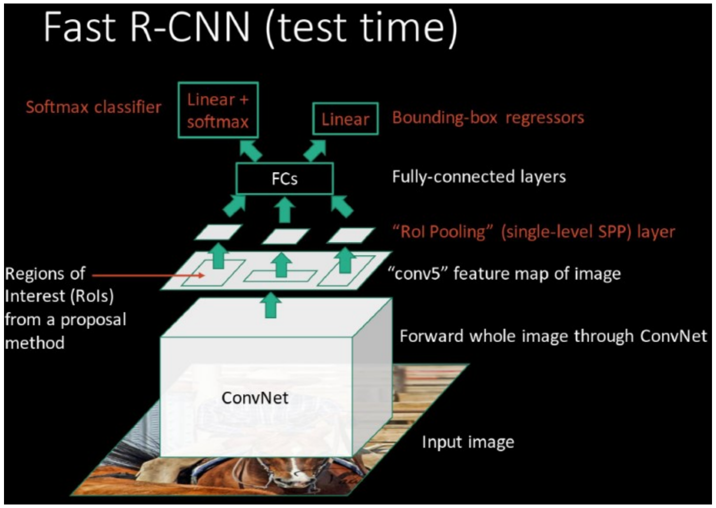


Image detection

- Fast R-CNN
  - See **Ross Girshick, Fast R-CNN,** IEEE International Conference on Computer Vision (ICCV), 2015
  - See https://github.com/rbgirshick/fast-rcnn

  - Join the feature extraction, classifier and regressor in a unified framework
  - RoI pooling layer → ~one scale SPP layer
    - Faster x10 than SPP-net
    - Scale invariance
  - Training
    - Classifier
      - Softmax (instead of SVM) → log loss
    - Regressor
      - Linear → Smooth L1 loss
    - Multi-task loss
      - Log loss
      - Smooth L1 loss
    - All layers

  - Steps:



Fast R-CNN (test time)

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

Object detection

- Fast R-CNN → training flow
  - Extract region proposal from an image
    - By various methods (Selective search, BING, MCG, …)
    - Regions are class-independent
    - Regions are of different sizes
  - Train or use a pretrained classification model (e.g. AlexNet on ImageNet)
  - Prepare the model for fine-tunning
    - Replace the last max pooling layer of the pre-trained CNN with a RoI pooling layer. The RoI pooling layer outputs fixed-length feature vectors of region proposals. Sharing the CNN computation makes a lot of sense, as many region proposals of the same images are highly overlapped.
    - Replace the last fully connected layer and the last softmax layer (K classes) with a fully connected layer and softmax over K + 1 classes.
      - Instead of 1000 classes use 20 object classes + background
      - reinitialize from scratch
  - Fine-tune the classification model for detection (VOC)
    - Processes the whole image to produce a convolutional feature map.
    - Then for each object proposal ROI pooling layer extracts fixed-length feature vector, which is finally passed to subsequent FC layers.
    - Finally the model branches into two output layers:
      - A softmax estimator of K + 1 classes (same as in R-CNN, +1 is the “background” class), outputting a discrete probability distribution per RoI.
      - A bounding-box regression model which predicts offsets relative to the original RoI for each of K classes.

Object detection

- Fast R-CNN → training flow
  - Reamrks:
    - 2000 proposals of particular image are not passed through the network as in R-CNN, Instead, The image is passed only once and the computed features are shared across ~2000 proposals like the same way it is done in SPP Net .
    - Also, the ROI pooling layer does max pooling in each sub-window of approximate size h/H X w/W. H and W are hyper-parameters. It is a special case of SPP layer with one pyramid level.
    - The two sibling output layers' outputs are used to calculate a multi-task loss on each labeled ROI to jointly train for classification and bounding-box regression.
    - They have used L1 loss for bounding box regression as opposed to L2 loss in R-CNN and SPP-Net which is more sensitive to outliers.
    - Batch size = 2 ➔ 2000 x 2 boxes → a lot of bg → sampling some (64 in original model) positive and negative boxes

Object detection

- Fast R-CNN → loss function
  - u - True class label, u∈0,1,…,K;
    - by convention, the catch-all background class has u=0
  - p - Discrete probability distribution (per RoI) over K + 1 classes: p=(p_0,…,p_K)
    - computed by a softmax over the K + 1 outputs of a fully connected layer.
  - v - True bounding box v=(v_x,v_y,v_w,v_h)
  - t^u - Predicted bounding box correction t^u=(t_x^u,t_y^y,t_w^u,t_h^u)

  - a loss combining two tasks
    - classification cost

$$Loss_{cls}(p,u) = -\log(p_u)$$

    - localization cost
      - A proposal is
        - fg if IoU with a BB > 0.5
        - bg if IoU in [0.1, 0.5]
      - Others are ignored

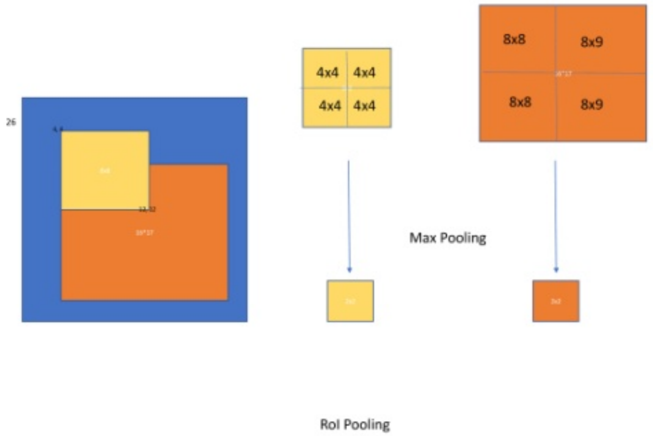
$$L_1(x) = |x|$$

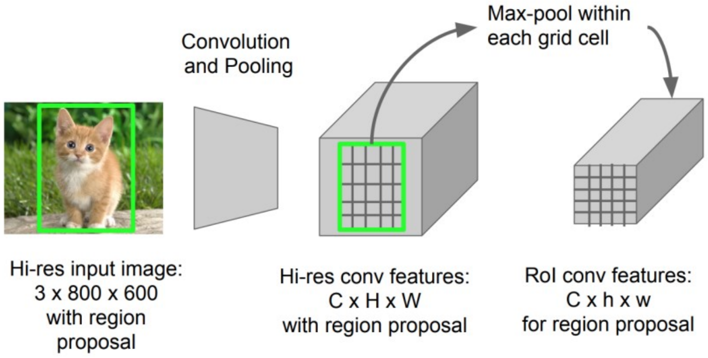
$$L_1^{smooth}(x) = \begin{cases} 0.5x^2, & if\,|x| < 1 \\ |x| - 0.5, & otherwise \end{cases}$$

$$Loss_{box}(t^u, v) = \sum_{i \in \{x,y,w,h\}} L_1^{smooth}(t_i^u - v_i)$$

Object detection

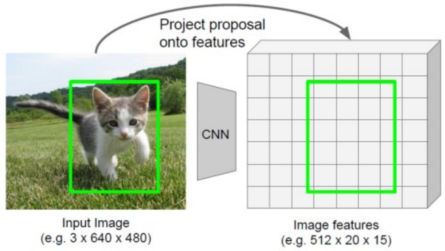
- Fast R-CNN
  - RoI pooling → Cropping features
    - Aim
      - Project proposal onto features = a type of max pooling to convert features in the projected region of the image of any size, h x w, into a small fixed window, H x W
    - How?
      - The input region is divided into H x W grids, approximately every subregion of size h/H x w/W.
        - "snap" to grid cells
        - Divide into 2x2 grid of (roughly) equal subregions
      - apply max-pooling within each subregion.
        - Dividing the region proposal into equal-sized sections (the number of which is the same as the dimension of the output)
        - Finding the largest value in each section
        - Copying these max values to the output buffer

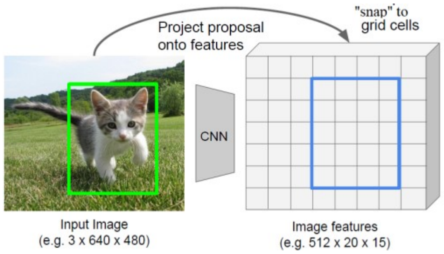


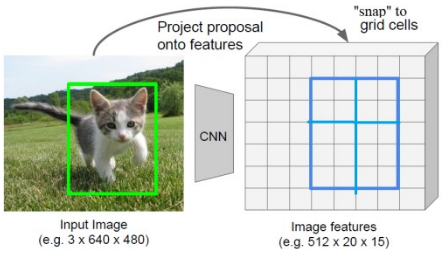

Convolution and Pooling
Max-pool within each grid cell
Hi-res input image: 3 x 800 x 600 with region proposal
Hi-res conv features: C x H x W with region proposal
RoI conv features: C x h x w for region proposal

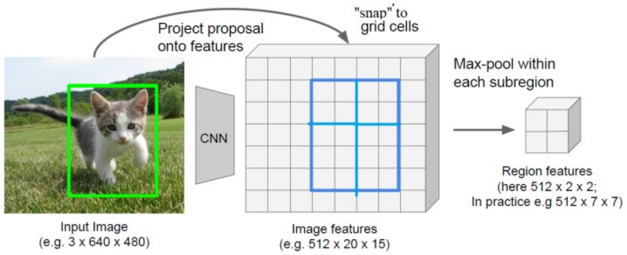
Object detection

- Fast R-CNN
  - RoI pooling → Cropping features
    - How?
      - The input region is divided into H x W grids, approximately every subregion of size h/H x w/W.
      - apply max-pooling within each subregion.
      - ➔ Region features always have the same size (even if input regions have different sizes)
    - RoI pooling versus classic pooling → control of output shape
      - Classic pooling: input + window, padding, stride ➔ output
      - RoI pooling: input + output's size ➔ output
    - Problem:
      - Region features slightly missaligned
    - Solution:
      - RoI alignment



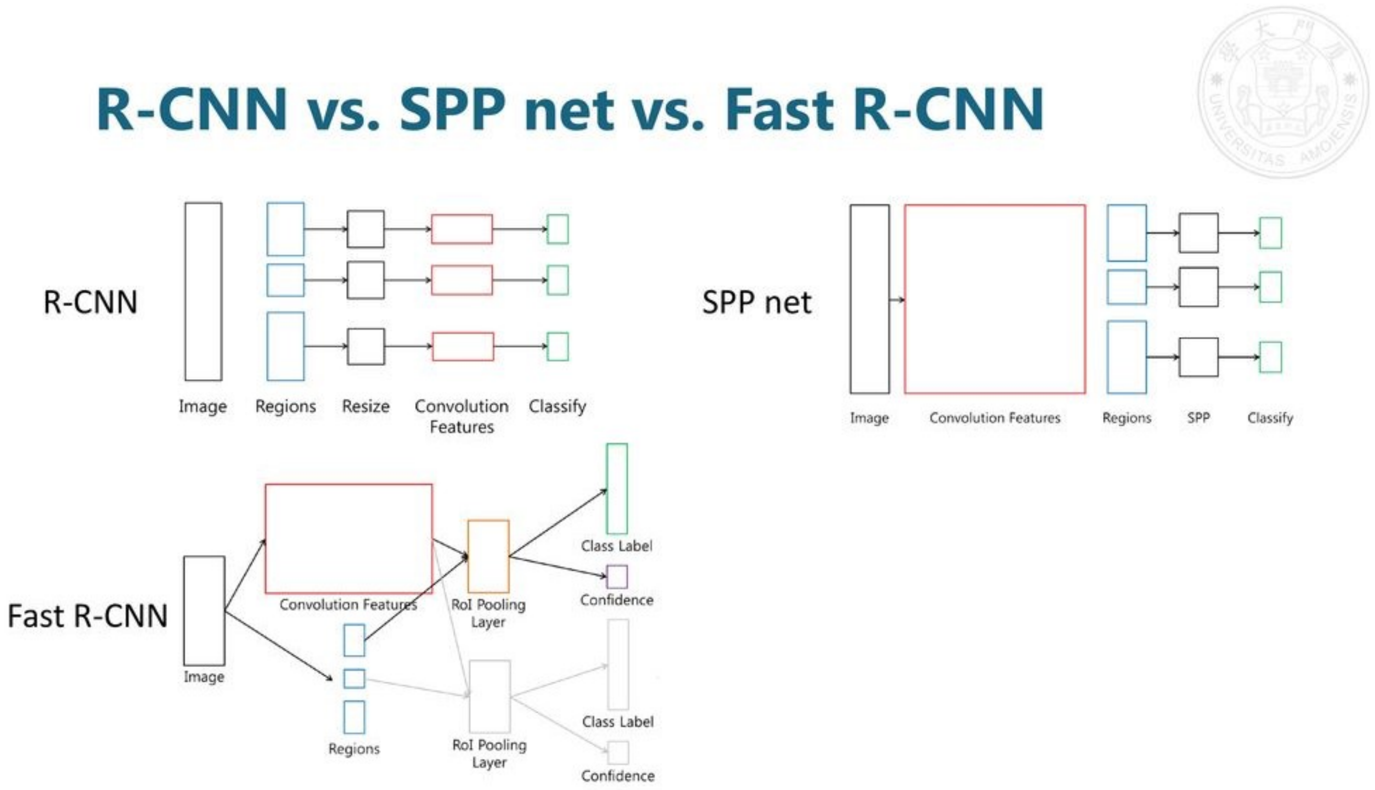






Object detection

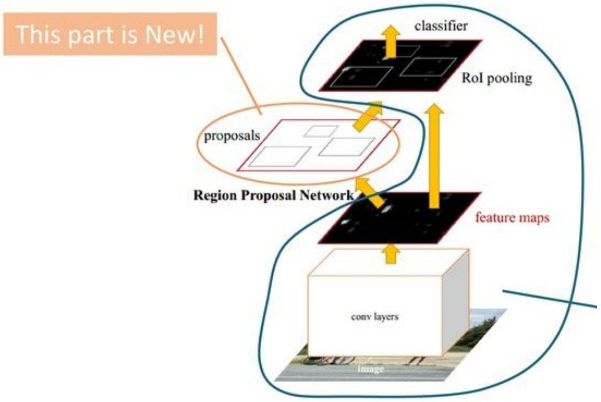
- R-CNN vs. SPP-net vs. Fast R-CNN



Object detection

- Fast-RCNN
  - Advantages
    - better performance
      - Fast test-time (~SPP net)
      - Higher mAP than slow R-CNN and SPP net)
    - single stage training,
    - all nets are training
  - Weaknesses
    - Object proposal sampling is still slow
    - Solution: sample BBs with CNN → Faster R-CNN

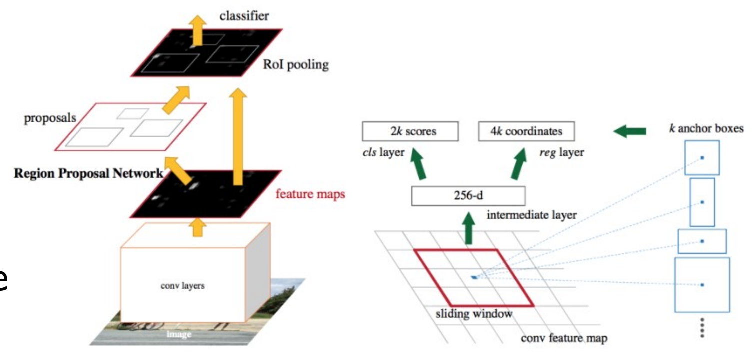
Object detection

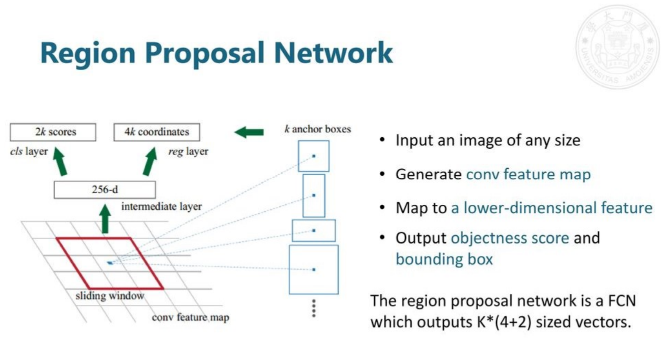


- Faster R-CNN
  - See Shaoqing Ren, Kaiming He, Ross Faster R-CNN: Towards Real-Time Ob Region Proposal Networks , Neural In Systems (NIPS), 2015 https://arxiv.org/pdf/1506.01497.pdf
  - See https://github.com/rbgirshick/py-faster-rcnn
  - Main idea
    - Fast R-CNN + changes the method for generating proposed regions from selective search to region proposal network
    - integrate the region proposal algorithm into the CNN model
    - a single, unified model composed of RPN (region proposal network) and fast R-CNN with shared convolutional feature layers
    - Introduce anchor boxes

Object detection

- Faster R-CNN → Region proposal network
  - a 3×3 convolutional layer with a padding of 1 to transform the CNN output and set the number of output channels to c ➔ each element in the feature map the CNN extracts from the image is a new feature with a length of cc.
  - each element in the feature map is used as a center to generate multiple anchor boxes of different sizes and aspect ratios and then label them.
  - the features of the elements of length c at the center on the anchor boxes are used to predict the binary category (object or background) and bounding box for their respective anchor boxes
  - non-maximum suppression removes similar bounding box results that correspond to category predictions of "object" ➔ the predicted bounding boxes as the proposed regions required by the RoI pooling layer.
  - RPN is trained together with the rest of model

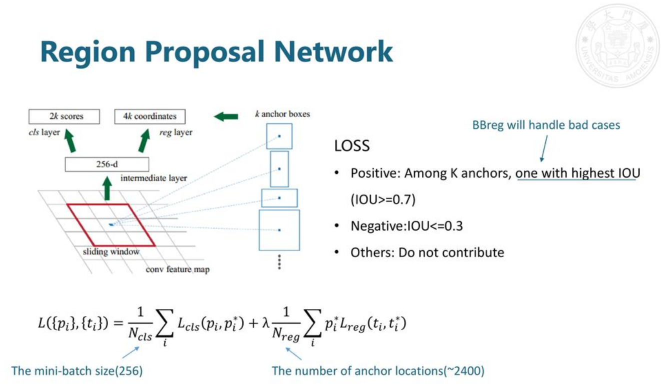




Region Proposal Network

- Input an image of any size
- Generate conv feature map
- Map to a lower-dimensional feature
- Output objectness score and bounding box

The region proposal network is a FCN which outputs K*(4+2) sized vectors.



Region Proposal Network

LOSS

BBreg will handle bad cases

- Positive: Among K anchors, one with highest IOU (IOU>=0.7)
- Negative:IOU<=0.3
- Others: Do not contribute

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*)$$

The mini-batch size(256)

The number of anchor locations(~2400)

Object detection

- Faster R-CNN → training flow

1. Pre-train a CNN network on image classification tasks.

2. Fine-tune the RPN (region proposal network) end-to-end for the region proposal task (which is initialized by the pre-train image classifier)
   - By using some samples
     - Positive samples have IoU (intersection-over-union) > 0.7,
     - negative samples have IoU < 0.3.
   - Slide a small n x n spatial window over the conv feature map of the entire image.
   - At the center of each sliding window, we predict multiple regions of various scales and ratios simultaneously.
     - An anchor is a combination of (sliding window center, scale, ratio).
     - E.g. 3 scales + 3 ratios => k=9 anchors at each sliding position.

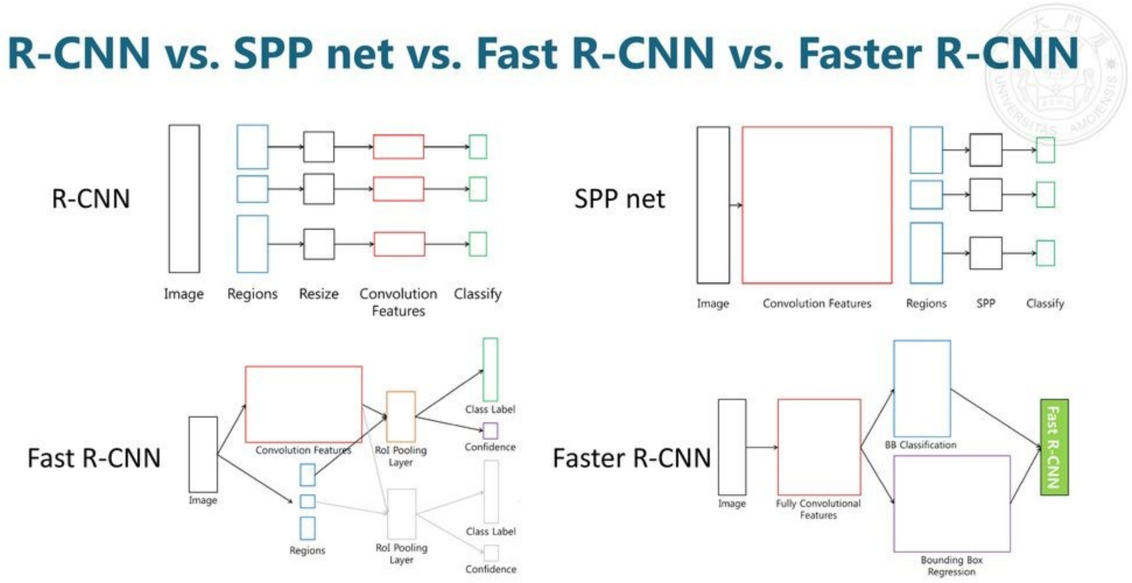
3. Train a Fast R-CNN object detection model using the proposals generated by the current RPN

4. Then use the Fast R-CNN network to initialize RPN training.
   - While keeping the shared convolutional layers, only fine-tune the RPN-specific layers.
   - RPN and the detection network have shared convolutional layers!

5. Finally fine-tune the unique layers of Fast R-CNN

6. Step 4-5 can be repeated to train RPN and Fast R-CNN alternatively if needed.

Object detection



Region proposal(SS)	
Feature extraction (Deep Net)	
Classification (SVM)	Bounding box (Regression)

RCNN

Region proposal(SS)
Feature extraction Classification+BB Regression (Deep Net)

Fast RCNN

Region proposal
Feature extraction
Classification+BB Regression
(Deep Net)

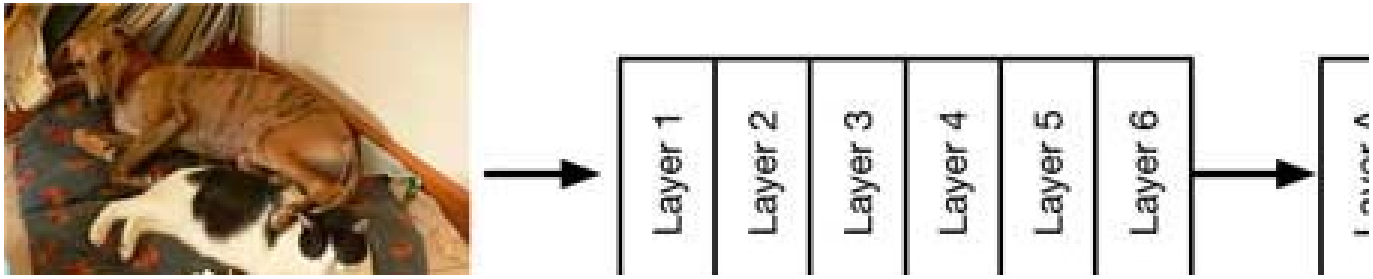
Faster RCNN

Object detection

- One stage detectors
  - YOLO (more versions)
  - SSD
  - Corner net
  - Retina net
  - Center net

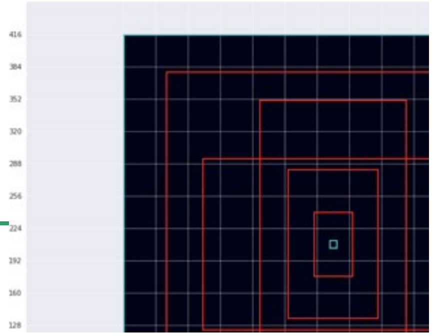
Object detection

- One-stage detectors
  - Images with
    - one object
    - two or more objects
  - require more detectors



  - without regional proposal ➔ grid of detectors
    - Grid dimension S x S cells
    - Each cell → one or more detectors
      - Output size: #detectors x (4 coordinates + 1 confidence score + K class probabilities)
    - Why more cells?
      - Detectors specialised in specific locations
      - Spatial constraints (~ proposed regions) → *where* in the image a detector can find objects
    - Why more detectors/cell?
      - Detectors specialised in specific shapes
      - Shape constraints → anchors
    - Why different grid-sizes?
      - Detectors specialised in specific scales

Object detection

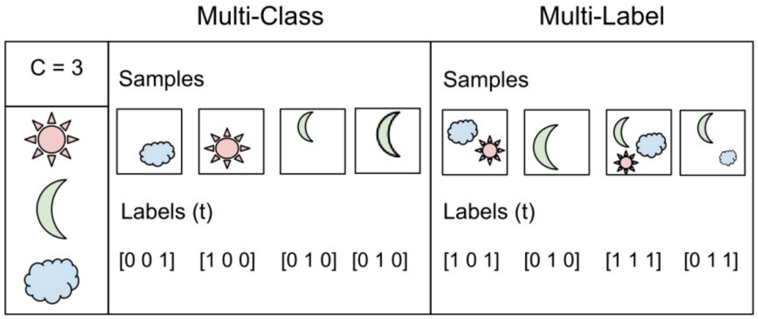


- One-stage detectors → Anchors
  - What?
    - describe the most common (average) object shapes in the dataset ➔ Constant elements (during training)
  - Other names
    - dimension priors (YOLO), biases (darknet), default boxes (SSD)
  - How to identify the anchors
    - Algorithms
      - K-mean algorithm over the BB of training data (YOLO)
      - Mathematical formula, independent of training data (SSD)
    - Constraints
      - the anchor's position is always in the center of the grid cell
  - Represented by:
    - width and height, relative to grid's coordinates (YOLO)
    - x, y, width, height (SSD)

Object detection

- One-stage detectors → detector output
  - Class probabilities
    - the kind of object
    - Softmax → max prob ➔ class of object
    - Sigmoid → multiclass (SSD, YOLO v3)
  - BB coordinates
    - The location of the object
    - Predict the offset of the BB to the real position
    - YOLO: Object's center must be located inside the grid cell
  - Confidence score
    - Does BB contain an object or not?
    - how likely the model thinks the predicted bounding box contains a real object
    - Objectness score → which BB can be ignored
    - YOLO → confidence score
    - SSD → a new class (background)
  - largely overlapping predictions
    - NMS remove duplicates

Object detection



- One-stage detectors → Loss
  - Real (target) values: **t** = [t_1, t_2, …, t_C]
  - Predicted scores: for each sample, there are C values (C = no of classes) ➔ **s** = [s_1, s_2, …, s_C]
  - Cross-entropy loss (Logistic Loss or Multinomial Logistic Loss)

$$CE = -\sum_{i=1}^{C} t_i \log(s_i)$$

  - Multi-class problems
    - **t** -> one-hot encodding (t_i = 1 for the positive clas, other values in t are 0)
    - Categorical cross-entropy (softmax loss)
      - **s** & softmax ➔ predicted values:

$$p = \left[\frac{e^{s_1}}{\sum_{j=1}^{C} e^{s_j}}, \frac{e^{s_2}}{\sum_{j=1}^{C} e^{s_j}}, \ldots, \frac{e^{s_C}}{\sum_{j=1}^{C} e^{s_j}} \right]$$

$$CE = -\sum_{i=1}^{C} t_i \log(p_i) = -\log(p_{correctClass})$$

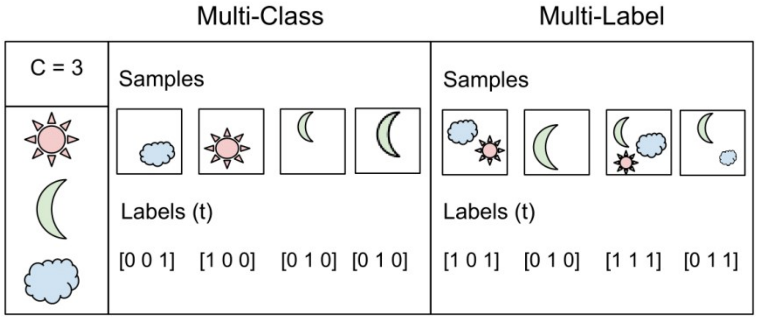
  - Multi-label problems
    - t → more vlaues of 1
    - C binary problems
      - Each score si & sigmoid ➔ predicted values
      - Binary cross entropy (sigmoid cross entropy) for each possible label

$$p = \left[\frac{1}{1+e^{-s_1}}, \frac{1}{1+e^{-s_2}}, \ldots, \frac{1}{1+e^{-s_C}} \right]$$

$$CE_i = -t_i \log(p_i) - (1-t_i)\log(1-p_i)$$

$$CE = \sum_{i=1}^{C} CE_i$$

Object detection



- One-stage detectors → Loss
  - Real (target) values: **t** = [t_1, t_2, …, t_C]
  - Predicted scores: for each sample, there are C values (C = no of classes) ➔ **s** = [s_1, s_2, …, s_C]
  - Cross-entropy loss (Logistic Loss or Multinomial Logistic Loss)

$$CE = -\sum_{i=1}^{C} t_i \log(s_i)$$

  - **Focal loss** → binary CE
    - weights the contribution of each sample to the loss based in the classification error

$$CE = -\sum_{i=1}^{C} (1 - p_i)^{\gamma} t_i \log(p_i)$$

    - γ>=0 - modulating factor to reduce the influence of correctly classified samples in the loss

Object detection

- One-stage detectors → Loss -> confidence to be a good detector → A detector
  - Has no GT bb
    - If confidence score > 0 → negative example
    - YOLO: noObjectLoss(i, j, d) = noObjectScale * (0 – sigm(confidenceScore(i, j, d)))^2
      - IoU(predictedBox, oneAnchor) > 0.6 → noObjectLoss(i, j, d) = 0
    - SSD: - (bg class)
  - Has a GT bb
    - objectLoss(i, j, d)=objectScale * (1 – sigm(confidenceScore(i, j, d)))^2
    - objectLoss(i, j, d)=objectScale * (IoU(anchor, predictedBB) – sigm(confidenceScore(i, j, d)))^2

Object detection

- One-stage detectors → Loss -> correctness of detector (correct class)
  - YOLO (v1, v2) → multi-class problem
    - SSE: classLoss(i, j, d] = classScale * (**t** - **p**))^2
      - **t – target classes** (one-hot encoded)
      - **p** is a softmax array (one-hot encoded)
  - YOLO (v3), SSD → multi-label problem
    - Binary cross-entropy:

$$classLoss(i,j,d) = classScale * \sum_{i=1}^{C} -t_i \log(p_i) - (1-t_i)\log(1-p_i)$$

Object detection

- One-stage detectors → Loss -> correctness of detector (correct location → bounding box’s coordinates)

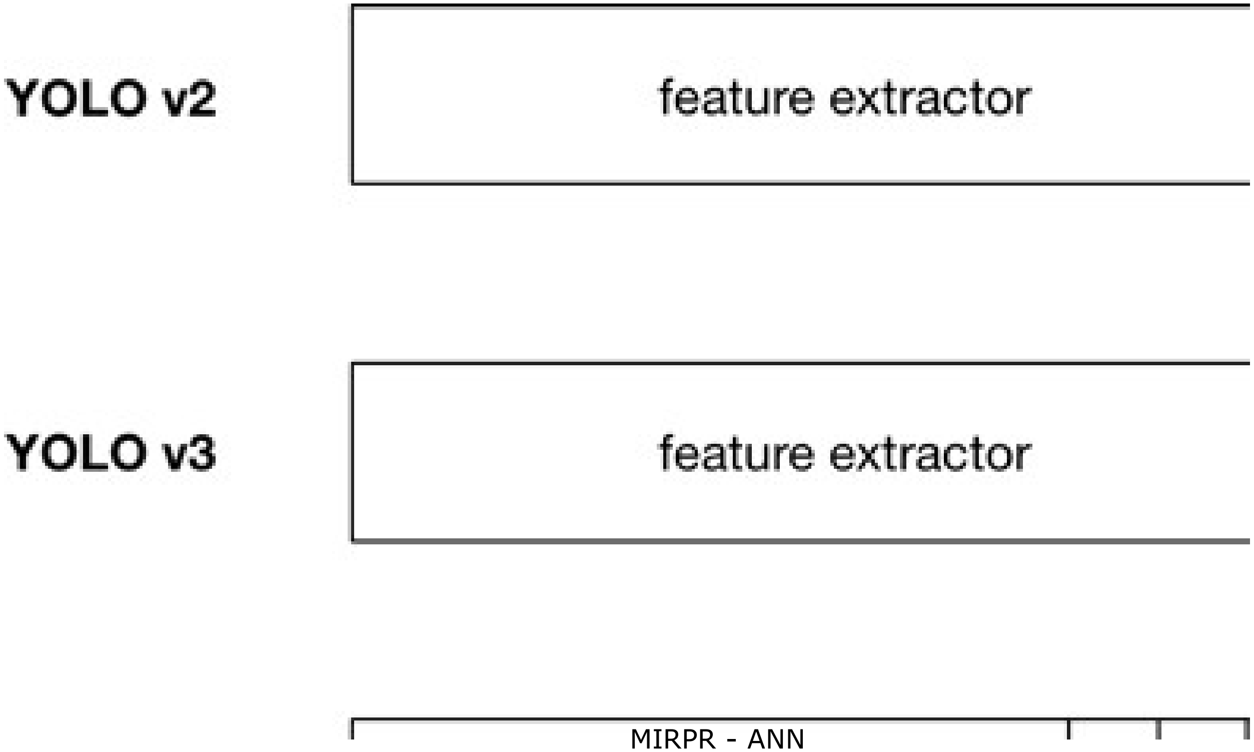
$$\mathrm{coordLoss(i,\ j,\ d)} = \mathrm{coordScale} * \sum_{r \in \{\mathrm{x,\ y,\ w,\ h}\}} \left(\mathrm{GT}_{\mathrm{bb}_\mathrm{r}}\mathrm{(i,\ j,\ d)} - \mathrm{pred}_{\mathrm{bb}_\mathrm{r}}\mathrm{(i,\ j,\ d)}\right)^2$$

- Loss → total loss

$$TotalLoss = \sum_{i=1}^{S}\sum_{j=1}^{S}\sum_{d=1}^{noDetectorsPerCell}\left(noObjectLoss(i,j,d) + objectLoss(i,j,d) + classLoss(i,j,d) + coordLoss(i,j,d)\right)$$

Object detection

- One-stage detectors → Architecture
  - YOLO v1
    - A single grid 7 x 7
    - FC layer for final prediction
    - No anchors
    - Predicted BB center coordinates are centered to the cell(sigm)
  - YOLO v2
    - A single grid 13 x 13
    - Conv layers for final prediction
    - Anchors (5 → k-means)
    - Scale constraints → anchors (determined by data)
  - YOLO v3
    - Several grids (a pyramid)
    - 3 detectors / grid cells at each scale
  - SSD
    - Several grids (a pyramid)
    - 3 detectors / grid cells at each scale
    - Predicted BB center coordinates *can* go outside their grid cells
    - No confidence score (there is a new class – background)
    - Scale constraints → different grids
    - anchors are used mostly to make the detectors specialize on different possible aspect ratios of the objects' shapes



Object detection

- One-stage detectors → Data augmentation
  - YOLO
    - Image → new w & h → crop → 416x416 (0 padding)
    - randomly flip the image horizontally (with 50% probability)
    - randomly distort the image’s hue, saturation, and exposure (brightness)
    - adjust the bounding box coordinates by shifting and scaling them to adjust for the cropping and resizing done earlier, and also for horizontal flipping
  - SSD
    - Randomly pick an image region so that the minimum IOU with the objects in the image is 0.1, 0.3, 0.5, 0.7, or 0.9. The smaller this IOU, the harder it will be for the model to detect the objects.
    - Using a “zoom out” augmentation that effectively makes the image smaller, which creates extra training examples with small objects. This is useful for training the model to do better on small objects

- Presented information was collected from various sources:
  - https://cs230.stanford.edu/
  - http://cs231n.stanford.edu/
  - https://d2l.ai/
  - https://berkeley-deep-learning.github.io/cs294-131-s17/
  - https://machinethink.net/
  - https://cedar.buffalo.edu/~srihari/CSE676/
  - http://karpathy.github.io/