# METODE INTELIGENTE DE REZOLVARE A PROBLEMELOR REALE

Laura Dioşan

Image detection

1

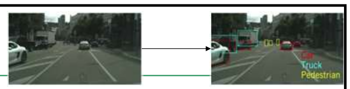
## Automatic image processing

- Image classification
  - Does an image contain object X? [yes/no]
- Object detection and image 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



2

## Object detection



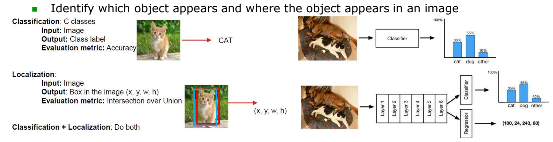

- Aim
  - Estimate the location and the class of all objects
  - A regression problem and a classification problem
- Challenges
  - Extent of objects is not fully observed
    - Occlusion
    - Truncation
  - Scale
  - Illumination changes
- Metrics
  - Intersection over union (IoU)
  - Average precision
- Methods
  - Traditional (sliding window)
  - Modern (neural networks)



3

## Object detection

- Aim
  - Identify which object appears and where the object appears in an image




- Tasks:
  - 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



4

## Object detection

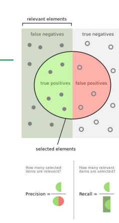
- Databases
  - Pascal VOC http://host.robots.ox.ac.uk/pascal/VOC/
    - 2005 – image detection task (4 classes, 1578 images, 2209 objects)
    - 2006 – image detection task (10 classes, 2618 images, 4754 objects)
    - …
    - 2012 – image detection task (20 classes, 11 530 images, 6929 objects)
  - 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 | --- | --- |

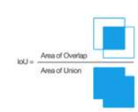


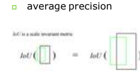
5

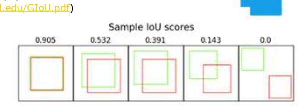
## Object detection




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[1]:
    - L1-distance, L2-distance – scale variant
    - intersection over union (IoU) – scale invariant
    - Generalised IoU (see https://giou.stanford.edu/GIoU.pdf)
    - average precision



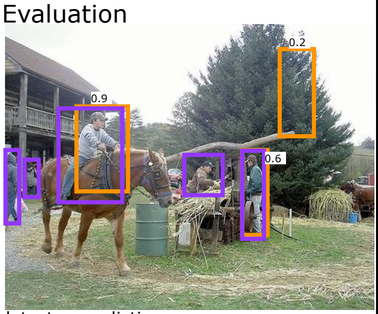





6

## Object detection

- Localisation → Evaluation




'person' detector predictions

ground truth 'person' boxes

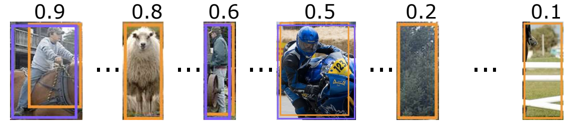


7

## Object detection

- Localisation → Evaluation
  - Sort by confidence

0.9 … 0.8 … 0.6 … 0.5 … 0.2 … 0.1



✓ X ✓ ✓ X X

true positive (high overlap)

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

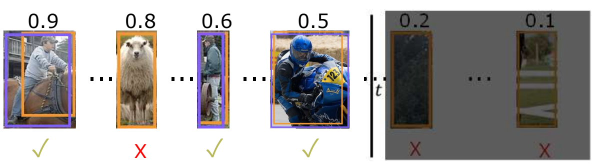


8

## Object detection

- Localisation → Evaluation
  - Sort by confidence

0.9 … 0.8 … 0.6 … 0.5 | t … 0.2 … 0.1



✓ X ✓ ✓ | X X

- Precision = TP@t / (TP@t + FP@t) $\frac{✓}{✓ + X}$
- Recall = TP@t / #GroundTruthObjects



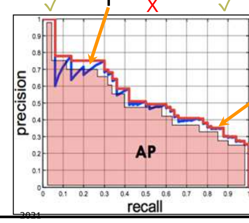
9

## Object detection

- Localisation → Evaluation

0.9 … 0.8 … 0.6 … 0.5 | … 0.2 … 0.1

✓ X ✓ ✓ X X




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

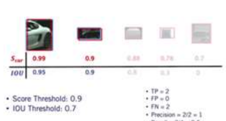
mean AP over classes (mAP)



10

## Object detection

- Detection → Evaluation
  - Intersection over union (IoU)
  - Average precision
    - TP: object class score > score threshold and IoU > IoU threshold
    - FP: object class score > score threshold and IoU < IoU threshold
    - FN: number of GT objects not detected by the algorithm
    - Precision = TP / (TP + FP)
    - Recall = TP / (TP + FN)
    - Precision-Recall curve (PR-Curve) – for different classification score thresholds
    - Average Precision (AP) – area under PR-Curve for a single class (approximation based on min 10 points)



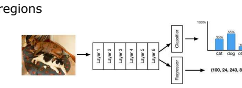




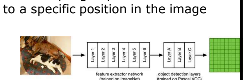
11

## Object 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

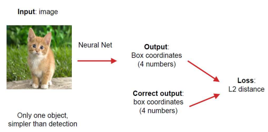




12

## 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.

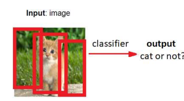





13

## 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

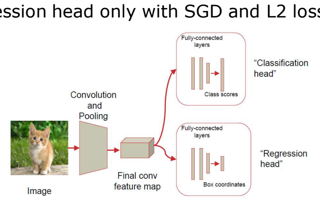





14

## 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



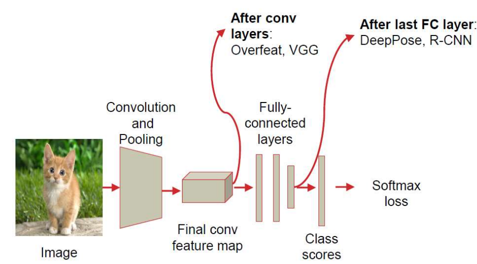

  - Testing
    - At test time use both heads



15

## Classification and localisation

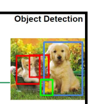
- As regression problem
  - Where to attach the regressor head?






16

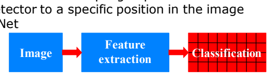
## Object detection




- 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

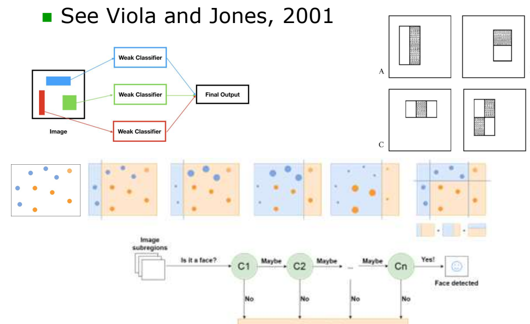





17

## Object detection

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

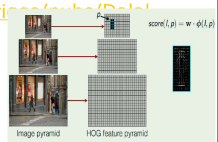





18

# Object detection

- Many positions and scales → HOG
  - Dalal 2005
  - See
    https://lear.inrialpes.fr/people/tri…/Dalal… cvpr05.pdf



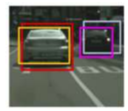

  - Compute HOG of the whole image at multiple resolutions
  - Classify/score each subwindow of the featured pyramid
  - Apply non-maximum suppression (NMS)



19

# Object detection

- Many positions and scales → HOG
  - Apply non-maximum suprpession (NMS)
    - Input:
      - More bounding boxes B = {b1, b2, …, bN}
      - More classification scores S = {cs1, cs2, …csN}
      - A classification threshold CT
      - An overlapping threshold OT
    - Output:
      - Best bounding boxes (Res is included in B)
    - Algorithm
      - Discard all boxes with si < CT
      - While B != empty
        - Pick bm, where m = argmax(S) and store it (Res <- Res + bm)
        - Discard from B that boxes of IoU with osi > OT



$\bar{B} = \{B_1, B_2, B_3, B_4\}$
$B = \{B_1, B_2, B_3, B_4\}$
$S = \{0.98, 0.94, 0.6, 0.45\}$
$D = \{\}$

OT = 0.7

bm = B1
Elimiante B1 from B and retain it in Res
IoU(B1,B2)=0 => nothing to do
IoU(B1, B3) = 0.8 > OT = 0.7 => eliminate B3
IoU(B1, B4)=0 => nothing to do
=> B = {B2, B4}
=
bm = B2
Eliminate B2 from B and retain it in Res
IoU(B2, B4) = 0.72 > OT = 0.7 => eliminate B4
=> B is empty

stop
=> final boxes {B1, B3}



20

# Object detection

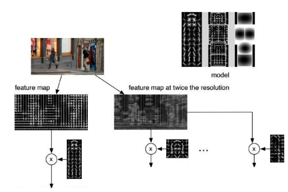
- Many positions and scales → HOG
  - Apply non-maximum suppression (NMS)
    - Note:
      - Class-wise operator
      - Soft NMS is more efficient
        https://arxiv.org/pdf/1704.04503.pdf
        - instead of completely removing the proposals with high IOU and high confidence, reduce the confidences of the proposals proportional to IOU value



21

# Object detection

- Detection as classification → Deformable Parts Model
  - Felzenszwalb 2009
    - 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.






22

# 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

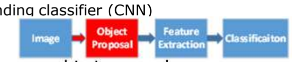


23

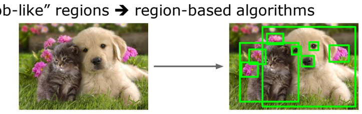
# Object detection

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


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




    - Solution = detectors based on an object proposal
      - 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





24

## Object detection


- Algorithms → Object detection → as classification -> detectors based on object proposal
  - Feature extraction
    - Most computationally expensive component
    - Output = a lower width and height element, but much greater depth (than the original input)
    - Most common extractors
      - VGG
      - ResNet
      - Inception

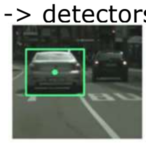

| | Image | Conv1 | Conv2 | Conv3 | Conv4 |
|---|---|---|---|---|---|
| Width | M | M/2 | M/4 | M/8 | M/16 |
| Height | N | N/2 | N/4 | N/8 | N/16 |
| Depth | 3 | 64 | 128 | 256 | 512 |

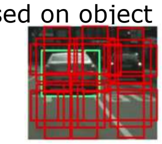


25

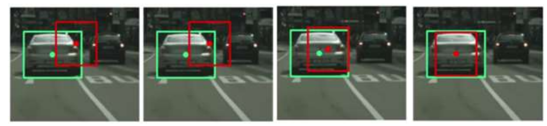
## Object detection

- Algorithms → Object detection → as classification -> detectors based on object proposal





  - Prior/Anchor bounding boxes
    - Residual learning

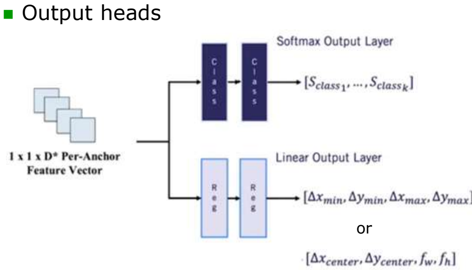




26

## Object detection

- Algorithms → Object detection → as classification -> detectors based on object proposal
  - Output heads

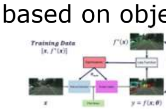





27

## Object detection

- Algorithms → Object detection → as classification -> detectors based on object proposal




  - Training
    - Mini batch selection
      - issue: A lot of anchors/priors are negatives
      - Solution: sample 3 neg and 1 pos anchors
      - Implications -> loss (classification and regression)

$$L_{cls} = \frac{1}{N_{total}} \sum_i CrossEntropy(s_i^*, s_i)$$

- $N_{total}$ is the size of our minibatch
- $s_i$ is the output of the neural network
- $s_i^*$ is the anchor classification target:
  - **Background** if anchor is negative
  - **Ground truth box class** if anchor is positive

$$L_{reg} = \frac{1}{N_p} \sum_i p_i L_2(b_i^*, b_i)$$

- $p_i$ is 0 if anchor is negative and 1 if anchor is positive
- $N_p$ is the number of positive anchors in the minibatch
- $b_i^*$ is the ground truth bounding box
- $b_i$ is the estimated bounding box, applying the regressed residuals to the anchor box parameters

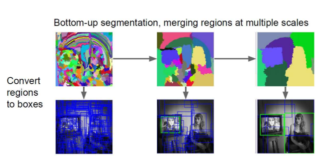
  - Testing
    - NMS (non-maximum suppression)



28

## Object detection

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




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



29

## 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)

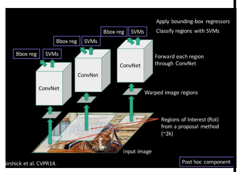


30

## 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

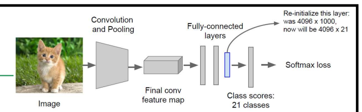



  - Steps:
    - Extract region proposal from an image by
      - Selective search (super pixel)
      - BING, MCG, ….
    - Compute CNN features
      - Crop
      - Scale
      - Forward propagate through a CNN
        - Use a pre-trained CNN (for classification) [and fine-tune CNN (for detection)]
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

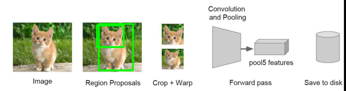


31

## 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)



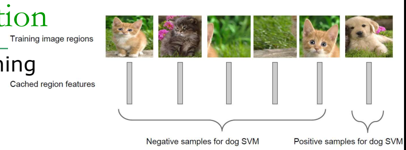

  - Fine-tune the classification model for detection (VOC)
    - Instead of 1000 classes use 20 object classes + background
    - Eliminate the final fully-connected layer, reinitialize from scratch
    - Keep training model using positive / negative regions from detection images
    - Compute features
      - 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



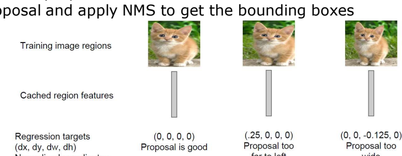
32

## 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



33

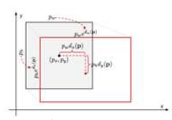
## Object detection

- 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, $d_i(\boldsymbol{p})$ 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)

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

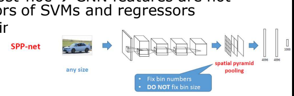
        - Regularisation term → cross validation (R-CNN paper)
      - not all the predicted bounding boxes have corresponding ground truth boxes
        - only a predicted box with a nearby ground truth box with at least 0.6 IoU is kept for training the bbox regression model



34

## 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



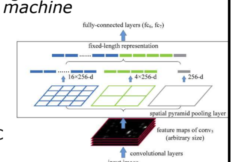

  - Solution
    - Slow testing & fix size constraints → multi-size training (SPP-net)
    - Process image (compute features) before RoI extraction = swap convolution and cropping (Fast R-CNN)



35

## 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



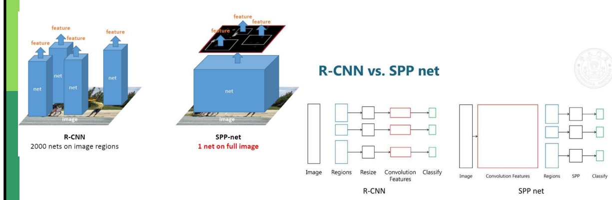

  - Traditional
    - One pooling layer between Conv and FC
  - Spatial Pyramid Pooling
    - multiple pooling layers with different scales
      1. each feature map is pooled to become one value (grey), thus 256-d vector is formed.
      2. each feature map is pooled to have 4 values (green), and form 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.



36

## 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

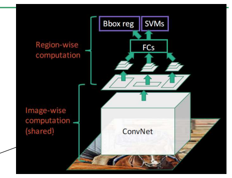





37

## 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)

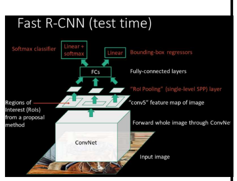





38

## 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:






39

## 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.



40

## Object detection

- Fast R-CNN → training flow
  - Remarks:
    - 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



41

## 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

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

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

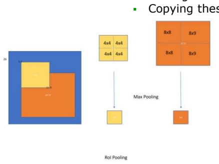
$$Loss_{box}(t^u,v)=\sum_{i\in\{x,y,w,h\}} L_1^{smooth}(t_i^u-v_i)$$

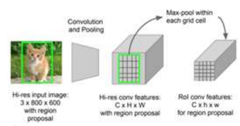


42

## 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



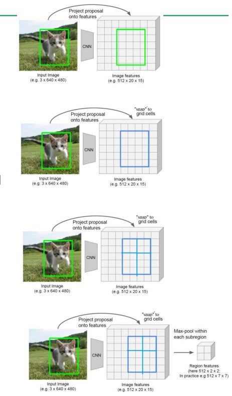




2021 MIRPR - Object detection 43

43

## 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

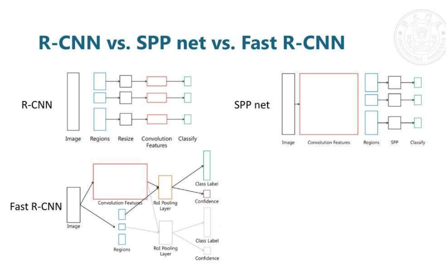


2021 MIRPR - Object detection 44

44

## Object detection

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




2021 MIRPR - Object detection 45

45

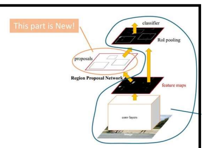
## Object detection

- 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

2021 MIRPR - Object detection 46

46

## Object detection



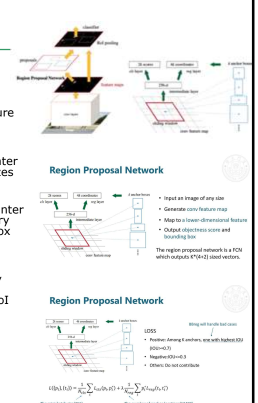

- Faster R-CNN
  - See Shaoqing Ren, Kaiming He, Ross Faster R-CNN: Towards Real-Time Ot Region Proposal Networks , Neural In Systems (NIPS), 2015 https://arxiv.org/pdf/1506.01497.pdf
  - See https://github.com/rbgirshick/py-faster-rcnn
  - Main idea
    - Fast R-CNN + changes the method for generating 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

2021 MIRPR - Object detection 47

47

## 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




2021 MIRPR - Object detection 48

48

## Object detection

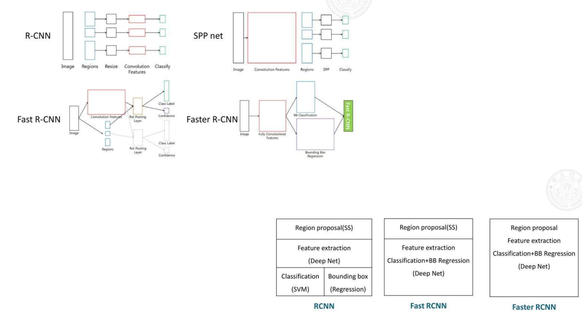
- Faster R-CNN → training flow
  1. Pre-train a CNN network on image classification tasks.
  2. Fine-tune the RPN (region proposal network) end-to-end for the region proposal task (which is initialized by the pre-train image classifier)
     - By using some samples
       - Positive samples have IoU (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.



49

## Object detection

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






50

## Object detection

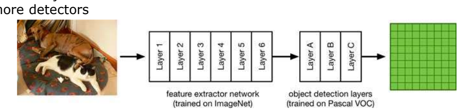
- One stage detectors
  - YOLO (more versions)
  - SSD
  - Corner net
  - Retina net
  - Center net
  - Detectron



51

## Object detection

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

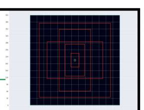



  - without regional proposal → grid of detectors
    - Grid dimension S x S cells
    - Each cell → one or more detectors
      - Output size: #detectors x (4 coordinates + 1 confidence score + K class probabilities)
    - Why more cells?
      - Detectors specialised in specific locations
      - Spatial 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



52

## Object detection



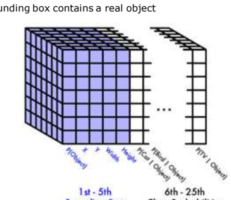
- One-stage detectors → Anchors
  - What?
    - describe the most common (average) object shapes in the dataset → Constant elements (during training)
  - Other names
    - dimension priors (YOLO), biases (darknet), default boxes (SSD)
  - How to identify the anchors
    - Algorithms
      - K-mean algorithm 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)



53

## Object detection

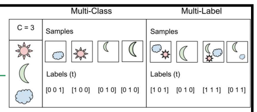
- One-stage detectors → detector output
  - Class probabilities
    - the kind of object
    - Softmax → max prob → class of object
    - Sigmoid → multiclass (SSD, YOLO v3)
  - BB coordinates
    - The location of the object
    - Predict the offset of the BB to the real position
    - YOLO: Object's 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






54

## 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

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

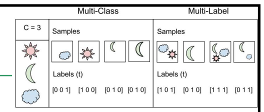
$$CE_i = -t_i \log(p_i) - (1-t_i)\log(1-p_i)$$

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



55

## 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

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

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



56

## 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)))$^2$



57

## 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:

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



58

## Object detection

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

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

- Loss → total loss

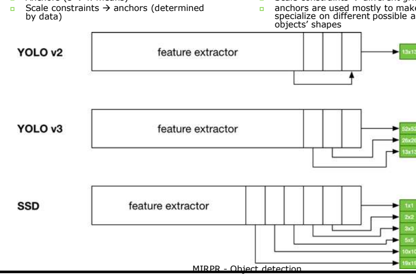
$$TotalLoss = \sum_{i=1}^{S} \sum_{j=1}^{S} \sum_{d=1}^{noDetectorsPerCell} \left(noObjectLoss(i,j,d) + objectLoss(i,j,d) + classLoss(i,j,d) + coordLoss(i,j,d)\right)$$



59

## 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






60

## 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



61

## Object detection

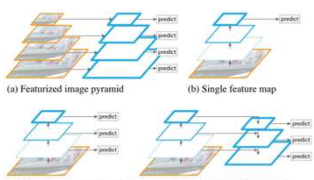
- Detectron2 notebook
  - https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5#scrollTo=7unkuuiqLdqd
- How to use a pretrained YOLO detector or how to train your YOLO detector
  - https://colab.research.google.com/drive/1_GdoqCJWXsChrOiY8sZMr_zbr_fH-0Fg?usp=sharing#scrollTo=voia4UtxlPMa
  - https://arxiv.org/pdf/2004.10934.pdf
  - https://medium.com/@alexeyab84/yolov4-the-most-accurate-real-time-neural-network-on-ms-coco-dataset-73adfd3602fe



62

## Object detection

- Feature pyramid network
  - provides a multiscale feature representation for object detection and instance segmentation
  - https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c




(a) Featurized image pyramid (b) Single feature map (c) Pyramidal feature hierarchy (d) Feature Pyramid Network



63