Object Recognition Chapter 10

Object Detection with Deep Neural Networks

Prof. Dr. Johannes Maucher

HdM CSM

Version 1.0
14.06.2018

Document History

Version Nr.	Date	Changes
1.0		Initial Version

Chapter 10: Object Detection with Deep Neural Networks

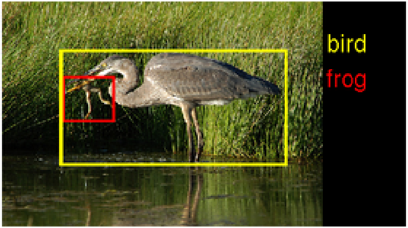
Goal of object Detection

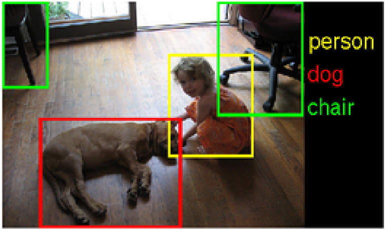
Object Detection Task

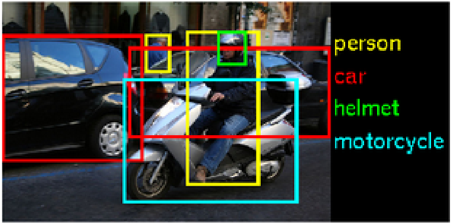
For a given image, determine:

- Which objects are in the image
- Where are these objects
- Confidence score of detection

Object Detection Examples





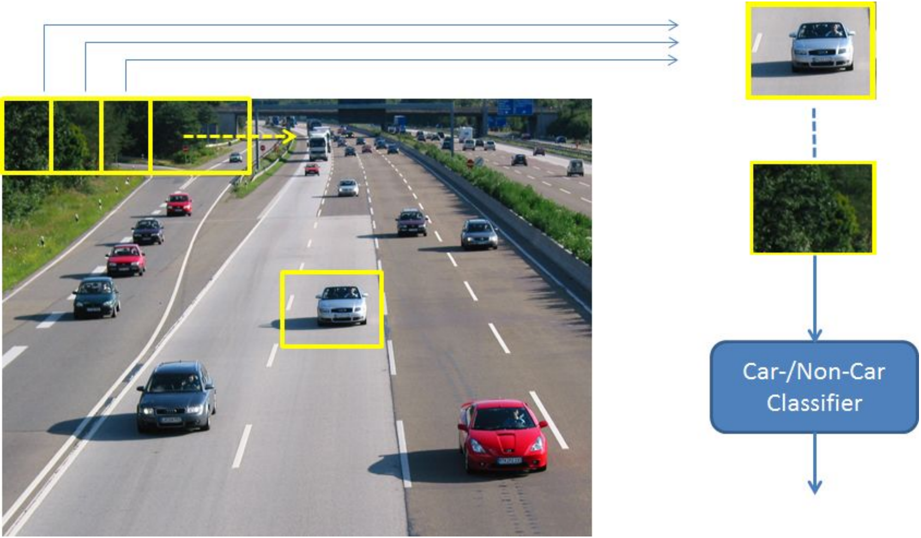


Source: http://image-net.org/challenges/LSVRC/2014/index

Object Detection: Approaches

- Sliding Window
- Overfeat: Train for each supported object category a CNN regressor-model, which predicts the coordinates of the bounding box
- In this lecture: Region Proposals and Deep Neural Networks:
    - R-CNN [Girshick et al., 2014]
    - Fast R-CNN
    - SPPnet
    - Yolo

Sliding Window Approach for Object Detection



Intersection over Union (IoU)

- If A is the set of detected pixels and B is the set of Groundtruth-pixel, their IoU is defined to be
$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|},$$
where $|X|$ is the number of pixels in set X.
- Usually two bounding boxes (pixel sets) are said to match, if their IoU is > 0.5.

Mean Average Precision (mAP)

- Calculate the average precision for each class.
- For determining TP,TN,FP and FN matches must be determined.
- Match: *detected object = groundtruth object* and *IoU of bounding boxes* > 0.5
- Average precision is related to the area under the precision-recall curve for a class:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r),$$

  where $p_{interp}(r)$ is the interpolated precision at recall r.
- The mean of these average individual-class-precisions is the Mean Average Precision
- More info: PASCAL VOC Challenge

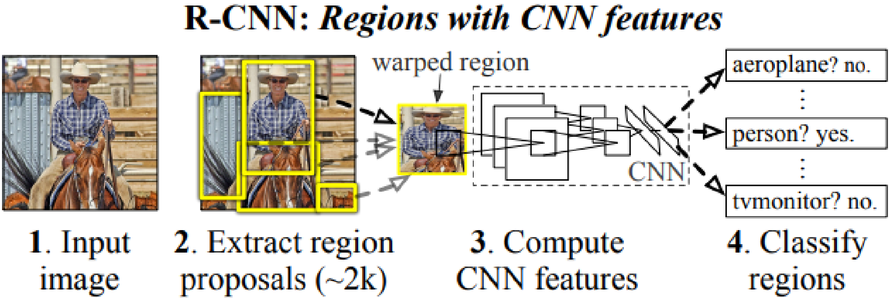
Region Proposals

- Goal: Given an input image find all possible places where objects can be located.
- Output: List of bounding boxes of likely positions of objects (= Region Proposals or Regions of Interest (ROI))
- Approach: Apply segmentation, e.g. hierarchical clustering, mean-shift clustering or graph-based segmentation
- Refine result from segmentation by e.g. selective search

R-CNN

- Published 2014 in [Girshick et al., 2014]
- Combines Region Proposals and CNN
- Region Proposals
  - are candidate boxes, which likely contain a object
  - can be computed by different algorithms, e.g. Selective Search
- mAP (mean Average Precision) of 53.7% on PASCAL VOC 2010- compared to 35.1% of an approach, which uses the same region proposals, but spatial pyramid matching + BoW
- mAP (mean Average Precision) of 33.4% on ILSVRC 2013 Detection benchmark- compared to 24.3% of the previous best result OverFeat [Sermanet et al.,]

R-CNN



Source: [Girshick et al., 2014]

R-CNN

Process:

1. Apply Selective Search for calculating about 2000 region proposals for the given image.
2. Warp each region proposal to a fixed size, since the CNN requires a fixed-size input.
3. Pass each warped region proposal through the CNN (modified AlexNet). Linear SVM-Classifier calculates a probability for each class.
4. Since the region proposals are not accurate, Bounding-Box Regression is applied to compute accurate bounding boxes. Input are coordinates of the region proposal. Output are the coordinates of the groundtruth bounding-box.

Training comprises:

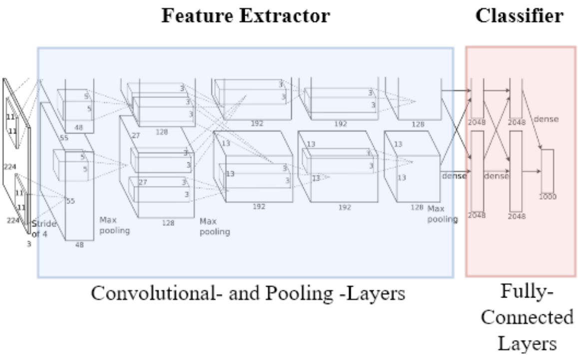
- Fine Tuning of CNN-Feature Extractor
- Train for each class a SVM Detector
- Train Bounding-Box regression (Ridge-Regression)

R-CNN

Drawback:

- 3 models must be learned
- Each of the $\sim$ 2000 region proposals must be passed through the entire CNN
- In particular for small regions the warping to a fixed size CNN-input is a waste of computational resources and yields slow detection
- $\Rightarrow \sim 13s/image$ on GPU

Why fixed size input to CNN?



- Convolutional- and pooling layer can easily cope with varying input-size.
- Fully connected layer require fixed-size input, because for this type varying size means varying number of weights.
- Idea: At the interface between Feature Extractor and Classifier: Find a method which can have variable size input but constant size output.

Spatial Pyramid Pooling in CNNs

- Published 2015 in [He et al., 2016]
- Integration of Spatial Pyramid Pooling ([Lazebnik et al., 2006]) into CNNs
- Applicable for classification and detection: ILSVRC 2014 rank in classification and rank 2 in detection task.
- As R-CNN based on region proposals
- No warping to fixed CNN-input size.
- Instead pass image only once through conv- and pool-layers of CNN.
- $24 - 102$ times faster than R-CNN in testing.

How SPPnet provides fixed-size representation to the classifier

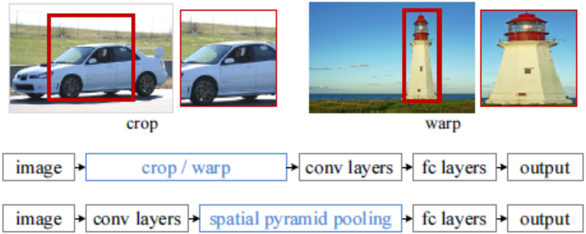
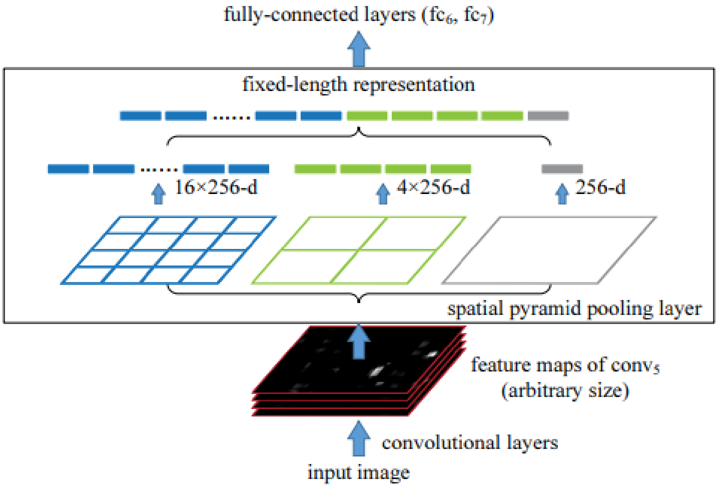


Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

Source: [He et al., 2016]

SPPnet



Source: [He et al., 2016]

SPPnet Process

Process:

1. Calculate Region Proposals
2. Pass image once and calculate feature maps for the entire image
3. Calculate the representation of each region proposal in the last conv-layer (before the fully-connected layers).
4. Crop the 256 feature maps in the last conv-layer to the size of the current region-proposal representation in this layer.
5. Apply spatial pyramid max-pooling (instead of simple max-pooling) on the cropped feature-maps of the last conv-layer and calculate fixed-size input to the fully-connected part of the CNN.
6. 4-level Pooling: $(1 \times 1, 2 \times 2, 3 \times 3, 6 \times 6)$
7. $\Rightarrow$: $1 + 4 + 9 + 36 = 50$ bins per feature map
8. $\Rightarrow$: $50 \cdot 256 = 12800$-length input to classifier.
9. Train binary linear SVM for each category.
10. Train Bounding-Box Regression

SPPnet Drawbacks

Drawback:

- As in R-CNN: 3 models must be trained
- Convolutional Layers that precede the spatial pyramid pooling can not be fine-tuned (because the gradients of the error function can not be passed efficiently through SPP). I.e. End-to-End training is not possible.

Fast R-CNN

- Published 2015 in [Girshick, 2015]
- Extension/Modification of R-CNN
- Also applies Region Proposals
- Enables End-to-End training by Hierarchical Sampling and Multi-Task-Loss
- Benefits:
  - Higher detection quality (mAP) than RCNN and SPPnet
  - No disk-storage for feature caching
  - Single-stage training: Fast R-CNN uses a streamlined training process with one fine-tuning stage that jointly optimizes a softmax classifier and bounding-box regressors. This is enabled by:
    - Hierarchical Minibatch Sampling
    - Multi-task loss function
    - Back-propagation learning through RoI-pooling layers

Fast RCNN

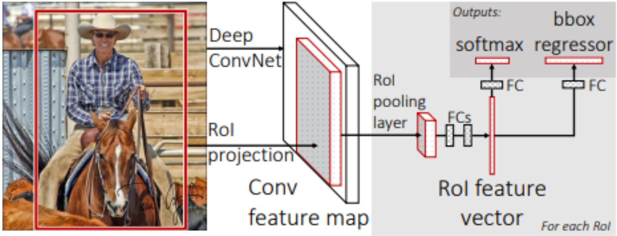


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Source: [Girshick, 2015]

Fast RCNN: Hierarchical Sampling

- In R-CNN and SPPnet: Minibatches of size $N = 128$ were constructed by sampling one RoI from 128 different images.
- Sampling multiple RoIs from a single image was supposed to be inadequate, because RoIs from the same image are correlated, causing slow training convergence.
- Since at the end of the Feature Extractor each RoI has a large receptive field (sometimes covering the entire image), fine-tuning of the Feature Extractor would be very expensive. Therefore, the Feature Extractor have not been fine-tuned in R-CNN and SPPnet.
- In Fast-RCNN minibatches are sampled hierarchically:
    - First sample N images ($N = 2$), than sampling R/N RoIs from each image ($R = 128$).
    - Samples from the same image share computation and memory in Forward- and Backwardpass
    - $64\times$ decrease in training time compared to sampling $R = 128$ different images.
    - The concern of slow convergence due to correlated samples within a minibatch has appeared to be not true in the research on Fast R-CNN.

Fast RCNN: Multi-Task Loss

- Two output-layers of Fast R-CNN:
  - Softmax-Output with $K+1$ for class-probabilities $p=(p_0,p_1,\ldots,p_K)$
  - Bounding-Box regressors $t^k=(t^k_x,t^k_y,t^k_w,t^k_h)$ for each of the K classes
- Each training RoI is labeled with a groundtruth class u and a groundtruth bounding-box regresssion target v.
- Multi-task loss L on each labeled RoI to jointly train for classification and bounding-box regression:

$$L(p,u,t^u,v)=L_{cls}+\lambda[u\geq 1]L_{loc}(t^u,v),$$

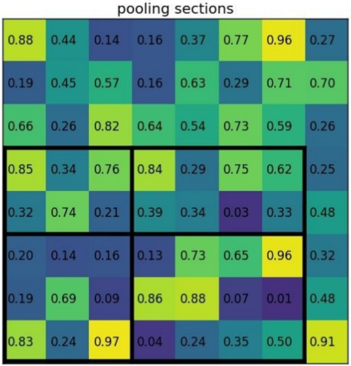
  where
  - the $[u\geq 1]$-operator evaluates to 1 for $u\geq 1$ and 0 otherwise (class-index $u=0$ indicates *no-known class*).
  - L_{cls} is the log-loss function for true class u and L_{loc} is a L_1 loss-function between the elements of the predicted- and the groundtruth bounding-box quadruple (see [Girshick, 2015]).

Fast RCNN: RoI Max-Pooling

- ROI of size $h \times w$ at the last convolutional layer is partitioned into a $H \times W$-grid, where each region is approximately of size $h/H \times w/W$. Typical values: $W = H = 7$.
- This is like level-1 partitioning in SPP.
- Features in each region are Max-pooled.
- Applied independently to each feature map

Fast RCNN: RoI Max-Pooling

RoI Partitioning



Source: https://deepsense.ai/region-of-interest-pooling-explained/

Result of Max Pooling

Yolo: You only look once

- Published 2015 in [Redmon et al., 2015]
- First real-time object detector with 45 fps on Titan X GPU. Faster version achieves 155 fps.
- https://pjreddie.com/darknet/yolo/
- Doesn't require other modules such as region proposals
- Instead: Single Regression Pipeline, which can be learned end-to-end
- YOLO sees the entire image during training and test time so it regards entire context (information, which is lost in sub-window approaches) $\Rightarrow$ YOLO makes less than half the number of background errors compared to Fast R-CNN.

Yolo Concept

- Partition entire image into a regular grid of $S \times S$ cells (typical: $S = 7$)
- The grid-cell which contains the center of an object is responsible to detect this object
- Each grid-cell predicts B bounding boxes (typical: $B = 2$) and confidence scores for these boxes.
- Confidence scores reflect
  - how confident the model is that the box contains an object: $P(object)$
  - how accurate the predicted box is: $IoU(pred, truth)$

$$Confidence = P(object) \cdot IoU(pred, truth)$$

- Each bounding box consists of 5 predictions: $x, y, w, h, Confidence$.
- Each grid-cell predicts C conditional class probabilities $P(C_j|object)$.
- At test time calculate class-specific confidence scores as follows

$$P(C_j|object) \cdot P(object) \cdot IoU(pred, truth) = P(C_j) \cdot IoU(pred, truth)$$

- This score encode probability of that class appearing in the box and how well the predicted box fits the object.

Yolo Concept

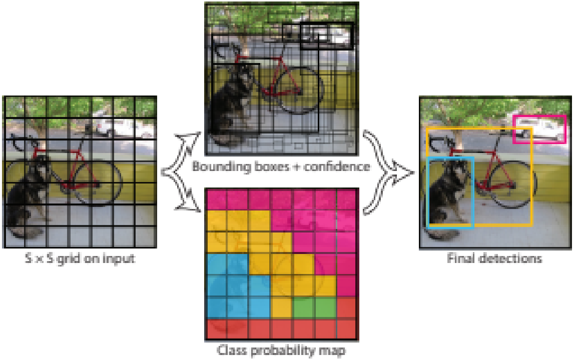


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Yolo Architecture

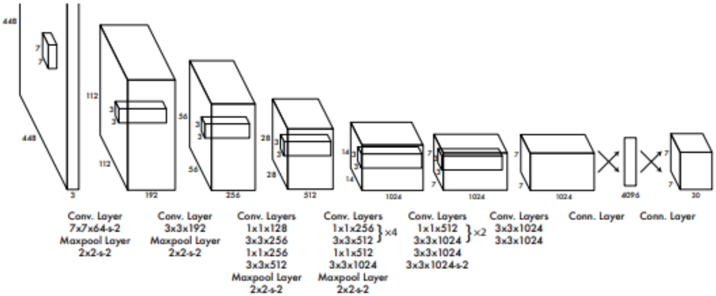


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Source: [Redmon et al., 2015]

Architecture

- Network output is a tensor of size $S \times S \times (5 \cdot B + C)$. With $C = 20$ classes and the parameters mentioned above, this tensor contains $49 \cdot 30$ elemenets.
- The faster Yolo version contains only 9 instead of 24 conv-layers and less feature maps.
- Conv-layers are pretrained with the ILSVRC classification benchmark.
- Loss-function: see [Redmon et al., 2015]

Performance compared to R-CNN

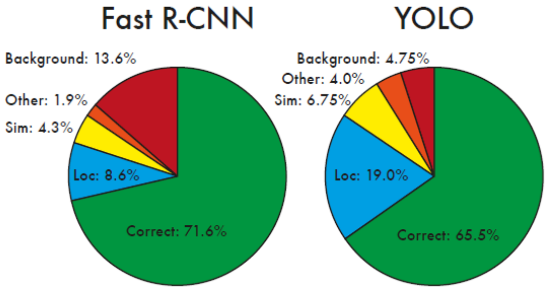


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Source: [Redmon et al., 2015]

Meanwhile Yolo has been improved significantly

References I

Girshick, R. (2015).
Fast r-cnn.
In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1440–1448, Washington, DC, USA. IEEE Computer Society.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014).
Rich feature hierarchies for accurate object detection and semantic segmentation.

He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Deep residual learning for image recognition.
In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Lazebnik, S., Schmid, C., and Ponce, J. (2006).
Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories.
In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2169–2178, Washington, DC, USA. IEEE Computer Society.

References II

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015).
You only look once: Unified, real-time object detection.
CoRR, abs/1506.02640.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and Lecun, Y.
Overfeat: Integrated recognition, localization and detection using convolutional networks.