Deep learning

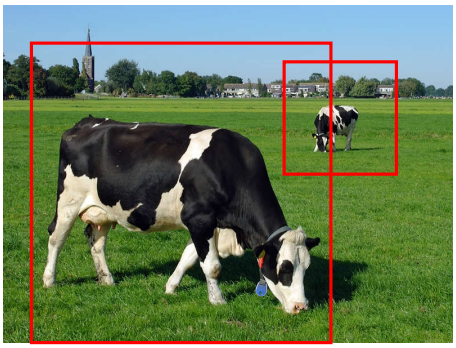
8.3. Networks for object detection

François Fleuret

https://fleuret.org/dlc/

The simplest strategy for object detection is to classify local regions, at multiple scales and locations.

Parsing at fixed scale



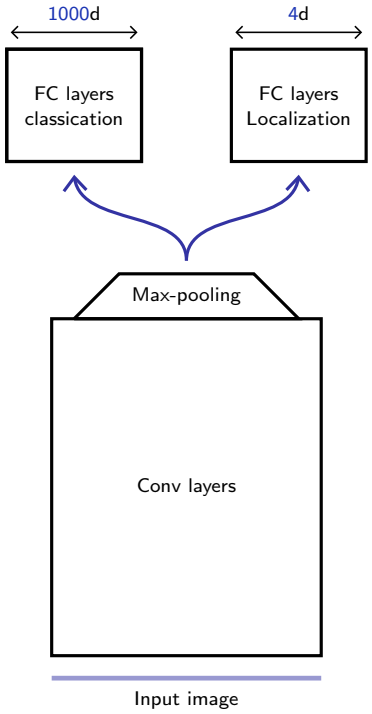
Final list of detections

This "sliding window" approach evaluates a classifier multiple times, and its computational cost increases with the prediction accuracy.

Notes

While image classification aims at predicting the class of the main object in the image, object detection aims at not only predicting the classes of all the objects which are visible, but also their locations.

This was mitigated in overfeat (Sermanet et al., 2013) by adding a regression part to predict the object's bounding box.



In the single-object case, the convolutional layers are frozen, and the localization layers are trained with a L_2 loss.

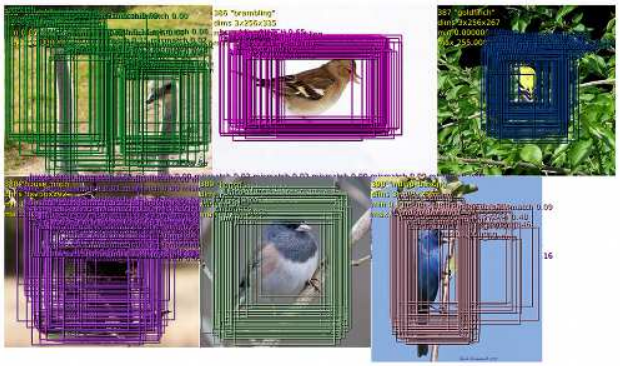


Figure 7: **Examples of bounding boxes produced by the regression network**, before being combined into final predictions. The examples shown here are at a single scale. Predictions may be more optimal at other scales depending on the objects. Here, most of the bounding boxes which are initially organized as a grid, converge to a single location and scale. This indicates that the network is very confident in the location of the object, as opposed to being spread out randomly. The top left image shows that it can also correctly identify multiple location if several objects are present. The various aspect ratios of the predicted bounding boxes shows that the network is able to cope with various object poses.

(Sermanet et al., 2013)

Combining the multiple boxes is done with an *ad hoc* greedy algorithm.

This architecture can be applied directly to detection by adding a class "Background" to the object classes.

Negative samples are taken in each scene either at random or by selecting the ones with the worst miss-classification.

Surprisingly, using class-specific localization layers did not provide better results than having a single one shared across classes (Sermanet et al., 2013).

Other approaches evolved from AlexNet, relying on **region proposals**:

- Generate thousands of proposal bounding boxes with a non-CNN "objectness" approach such as Selective search (Uijlings et al., 2013),
- feed to an AlexNet-like network sub-images cropped and warped from the input image ("R-CNN", Girshick et al., 2013), or from the convolutional feature maps to share computation ("Fast R-CNN", Girshick, 2015).

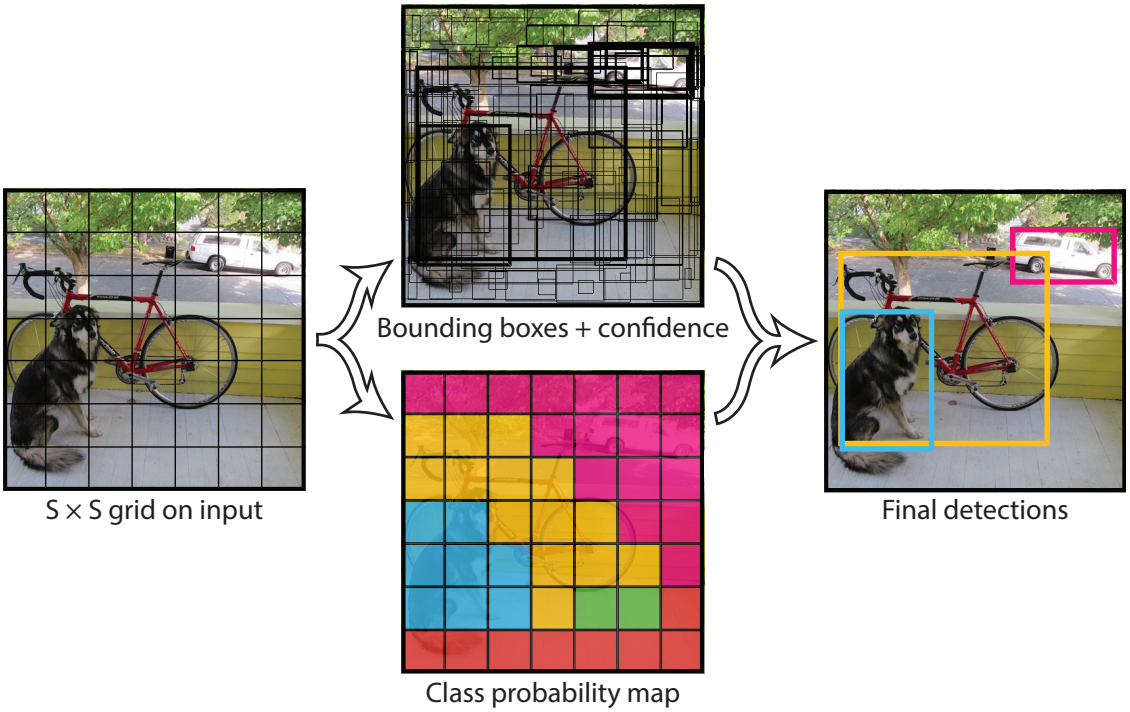
These methods suffer from the cost of the region proposal computation, which is non-convolutional and not implementable on GPU.

They were improved by Ren et al. (2015) in "Faster R-CNN" by replacing the region proposal algorithm with a convolutional processing similar to Overfeat.

The most famous algorithm from this lineage is "You Only Look Once" (YOLO, Redmon et al. 2015).

It comes back to a classical architecture with a series of convolutional layers followed by a few fully connected layers. It is sometime described as "one shot" since a single information pathway suffices.

YOLO's network is not a pre-existing one. It uses leaky ReLU, and its convolutional layers make use of the 1×1 bottleneck filters (Lin et al., 2013) to control the memory footprint and computational cost.



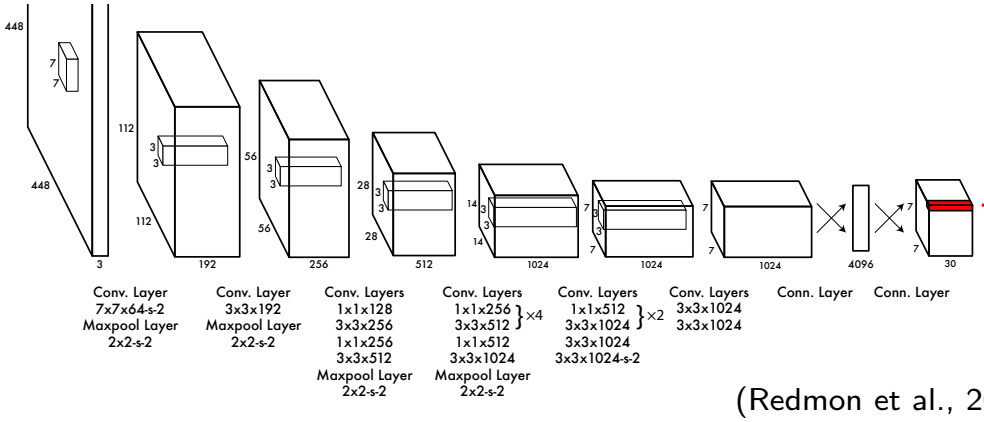
(Redmon et al., 2015)

Notes

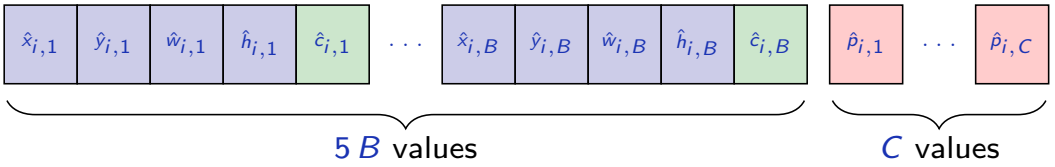
The processing of a scene is done by splitting it into a regular $S \times S$ grid, here $S = 7$. Each cell is responsible for detecting the object whose center lay inside, if any.

The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$, and for each cell, to predict a 30d vector:

- $B = 2$ bounding boxes coordinates and confidence,
- $C = 20$ class probabilities, corresponding to the classes of Pascal VOC.



(Redmon et al., 2015)



So the network predicts class scores and bounding-box regressions, and **although the output comes from fully connected layers, it has a 2D structure**.

It allows in particular YOLO to leverage the absolute location in the image to improve performance (e.g. vehicles tend to be at the bottom, umbrella at the top), which may or may not be desirable.

During training, YOLO makes the assumption that any of the S^2 cells contains at most [the center of] a single object. We define for every image, cell index $i = 1, \ldots, S^2$, predicted box index $j = 1, \ldots, B$ and class index $c = 1, \ldots, C$

- $\mathbf{1}_i^{obj}$ is 1 if there is an object in cell i and 0 otherwise,
- $\mathbf{1}_{i,j}^{obj}$ is 1 if there is an object in cell i and predicted box j is the most fitting one, 0 otherwise.
- $p_{i,c}$ is 1 if there is an object of class c in cell i, and 0 otherwise,
- x_i, y_i, w_i, h_i the annotated object bounding box (defined only if $\mathbf{1}_i^{obj} = 1$, and relative in location and scale to the cell),
- $c_{i,j}$ IOU between the predicted box and the ground truth target.

The training procedure first computes on each image the value of the $\mathbf{1}_{i,j}^{obj}$s and $c_{i,j}$, and then does one step to minimize

$$\lambda_{coord} \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbf{1}_{i,j}^{obj} \left((x_i - \hat{x}_{i,j})^2 + (y_i - \hat{y}_{i,j})^2 + \left(\sqrt{w_i} - \sqrt{\hat{w}_{i,j}} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_{i,j}} \right)^2 \right)$$

$$+ \lambda_{obj} \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbf{1}_{i,j}^{obj} (c_{i,j} - \hat{c}_{i,j})^2 + \lambda_{noobj} \sum_{i=1}^{S^2} \sum_{j=1}^{B} \left(1 - \mathbf{1}_{i,j}^{obj} \right) \hat{c}_{i,j}^2$$

$$+ \lambda_{classes} \sum_{i=1}^{S^2} \mathbf{1}_{i}^{obj} \sum_{c=1}^{C} \left(p_{i,c} - \hat{p}_{i,c} \right)^2 .$$

where $\hat{p}_{i,c}, \hat{x}_{i,j}, \hat{y}_{i,j}, \hat{w}_{i,j}, \hat{h}_{i,j}, \hat{c}_{i,j}$ are the network's outputs.

(slightly re-written from Redmon et al. 2015)

Notes

This loss is a good example of how one can mix multiple aspects of the prediction. The task has several purposes:

- bounding box coordinates, and confidence regression,
- classification.

The first part of the loss aims at minimizing the localization error of the detection. The square root is used to reduce the weight of the height and width (w_i, h_i) over the corner location (x_i, y_i).

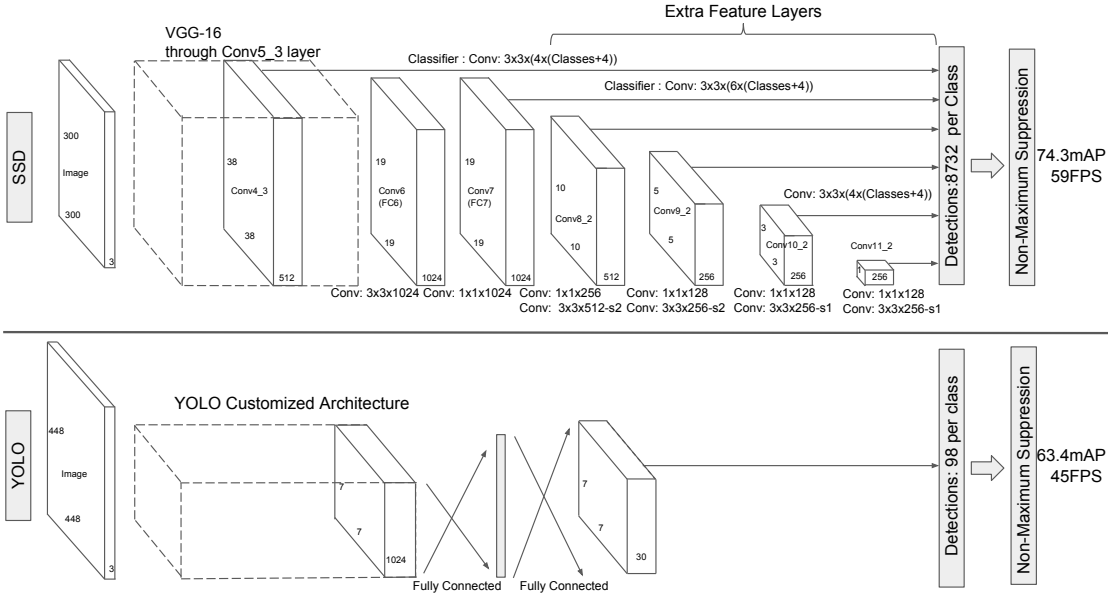
The second part of the loss estimates the confidence of a detection $\hat{c}_{i,j}$ to reflect the intersection over union $c_{i,j}$ of that bounding box with the ground truth. This part of the loss is weighted by $\lambda - obj$. And when there is no object, we want the confidence $\hat{c}_{i,j}$ of that bounding box to be low, this part being driven by λ_{noobj}.
Finally, the last part of the loss is for the class score. Note that while a natural choice is the cross-entropy, it is here a quadratic error.

Training YOLO relies on many engineering choices that illustrate well how involved is deep-learning "in practice":

- Pre-train the 20 first convolutional layers on ImageNet classification,
- use 448×448 input for detection, instead of 224×224,
- use Leaky ReLU for all layers,
- dropout after the first fully connected layer,
- normalize bounding boxes parameters in $[0, 1]$,
- use a quadratic loss not only for the bounding box coordinates, but also for the confidence and the class scores,
- reduce the weight of large bounding boxes by using the square roots of the size in the loss,
- reduce the importance of empty cells by weighting less the confidence-related loss on them,
- use momentum 0.9, decay $5e-4$,
- data augmentation with scaling, translation, and HSV transformation.

A critical technical point is the design of the loss function that articulates both a classification and a regression objectives.

The Single Shot Multi-box Detector (SSD, Liu et al., 2015) improves upon YOLO with a fully-convolutional architectures and multi-scale maps.

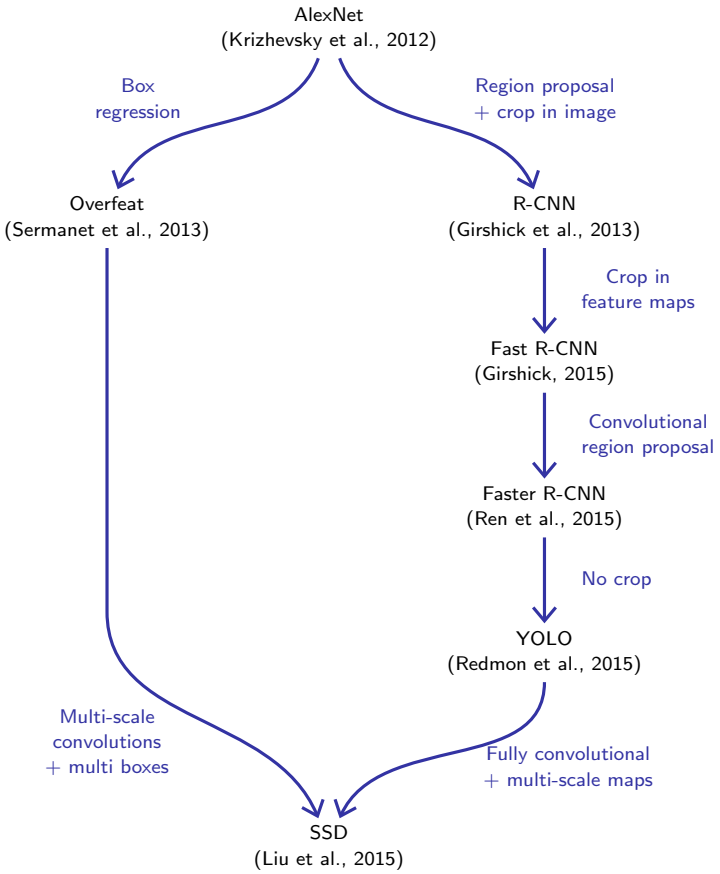


(Liu et al., 2015)

To summarize roughly how "one shot" deep detection can be achieved:

- networks trained on image classification capture localization information,
- regression layers can be attached to classification-trained networks,
- object localization does not have to be class-specific,
- multiple detection are estimated at each location to account for different aspect ratios and scales.

Object detection networks



References

R. Girshick. **Fast R-CNN**. CoRR, abs/1504.08083, 2015.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. **Rich feature hierarchies for accurate object detection and semantic segmentation**. CoRR, abs/1311.2524, 2013.

A. Krizhevsky, I. Sutskever, and G. Hinton. **Imagenet classification with deep convolutional neural networks**. In Neural Information Processing Systems (NIPS), 2012.

M. Lin, Q. Chen, and S. Yan. **Network in network**. CoRR, abs/1312.4400, 2013.

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. **SSD: single shot multibox detector**. CoRR, abs/1512.02325, 2015.

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

S. Ren, K. He, R. B. Girshick, and J. Sun. **Faster R-CNN: towards real-time object detection with region proposal networks**. CoRR, abs/1506.01497, 2015.

P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. **Overfeat: Integrated recognition, localization and detection using convolutional networks**. CoRR, abs/1312.6229, 2013.

J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. **Selective search for object recognition**. International Journal of Computer Vision, 104(2):154–171, 2013.