# Literature Review

Philipp Duernay

February 15, 2018

## 1 Single Shot Detectors

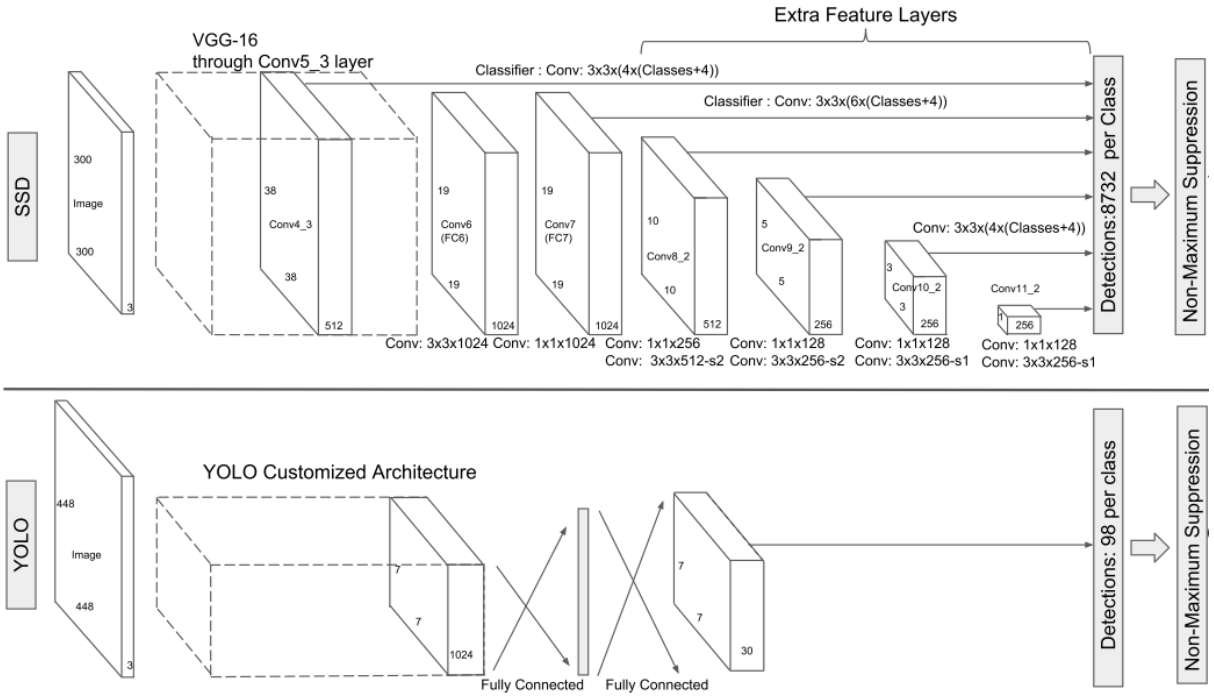


Figure 1: Two One Shot Detectors (old) Yolo and SSD [6]

### 1.1 Basic Concept

Single Shot Detectors are (fully) convolutional networks that produce several bounding box predictions in one network evaluation. Predictions are made for a fixed set of *anchor boxes* (also default/prior boxes). Each box resides at a pre-determined location and has a pre-determined size and aspect ratio. The detector predicts "adaptions" to these default coordinates for each of the boxes. Additionally it predicts class confidence scores for each of these boxes. Usually this produces numerous bounding boxes in one evaluation. The predictions are filtered in a final non-maximum-suppression step.

### 1.2 Anchor Boxes

Usually the image is split in several grid cells. To the center of each grid cell fall several anchor boxes with multiple aspect ratios. This can be seen in Figure 2.
For SSD the amount/location of anchor boxes is determined by the number of predictor layers. Each predictor layer produces boxes at a certain location/scale. The aspect ratios are selected by hand.
For Yolo the amount of anchors boxes doesn't depend on the architecture and is chosen via grid size/ number of boxes. Their aspect ratios are determined by a pre-clustering of the ground truth boxes of a dataset.

### 1.3 Architecture

Usually a classification network like VGG-16 is used as *base network*. On the feature maps of the base network additional convolutional layers are set, that predict class confidences and box coordinates. As for each anchor box a separate

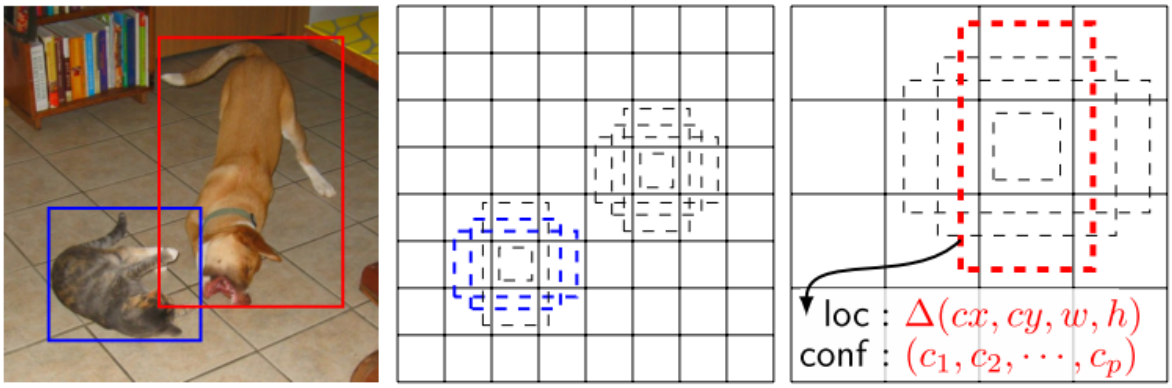


Figure 2: Illustration for anchor box concept [6]

output is predicted, each predictor can "specialize" on objects that mostly fall in its aspect ratio/scale/ location (?). The networks implement the final predictions as 1x1 convolutions, thus realizing a fully connected layer more efficiently(?) and allowing various image sizes.
SSD adds several convolutional predictor layers that predict bounding boxes for various scales. They also use one lower layer of the base network for prediction as they assume it preserves more fine grained features.
The first Yolo version as displayed in Figure 1 used a fully connected layer as regression head. However, in the new version they also switched to one convolutional layer.

### 1.4 Training Goal

SSDs treat object detection as a regression problem. The loss function usually incorporates the aforementioned bounding box coordinates and class scores. During training only one anchor box should be "responsible" for predicting that true box. This responsibility is determined by the location/size of the true box. The SSD paper refers to this as the *matching strategy*.
Yolo matches those boxes to the ones that have their center at a certain grid cell. Among the remaining boxes the one with the highest intersection-over-union (IoU) is chosen. SSD matches anchor and true boxes only based on the IoU.
Yolo Prediction:

$$t_x, t_y, t_w, t_h, t_o \quad \text{from cell:} c_x, c_y$$

$$x = \sigma(t_x) + c_x \quad y = \sigma(t_y) + c_y \quad w = p_w e^{t_w} \quad h = p_h e^{t_h} \quad conf = \sigma(t_o)$$

where $\sigma(x)$ is softmax function. Yolo Loss:

$$L = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i) +$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 +$$

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 +$$

$$\sum_{i=0}^{S^2} i_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

SSD Loss:

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g)$$

$$L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in cx, cy, w, h} x_{ij}^k smooth_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = log(\frac{g_j^w}{d_i^w}) \quad \hat{g}_j^h = log(\frac{g_j^h}{d_i^h})$$

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i{}^0) \quad \hat{c}_i{}^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)}$$

#### 1.4.1 Negatives

## 1.5 Data Augmentation

## 1.6 Comparison

#### 1.6.1 Yolo

a. Only one convolutional layer for prediction

b. Anchor boxes determined in pre-clustering step

c. At most one GT object per grid cell

d. Assigns GT based on grid cell, at most one GT box per grid cell

#### 1.6.2 SSD

a. 6 Convolutional layers

b. Assigns anchor based on iou, several anchors can be responsible for one GT, each GT has at least one anchor

## 1.7 Additional Work on One-Shot-Detectors

#### 1.7.1 LCDNet[8]

Aims to bring one shot detection to be runnable on embedded devices.

- Quantizies model for inference with 8bit. Records min and max value for each layer and quantizes everything between 0,255
- Replaces fully connected layers with convolutional layers
- Replaces LeakyRelu with Relu in all but the last (faster?)
- Softmax on classification, Sigmoid on confidence
- Sigmoid if only one class
- quantization mainly effects localization

## 1.8 Squeeze Det [10]

## 1.9 Summaries

#### 1.9.1 Object Detection

**Scalable Object Detection using Deep Neural Networks[4]**

- Generates number of bounding boxes as object candidates (class agnostic) and confidences for each box
- For each Bounding Box a classifier is run e.g. DNN
- Training: If the number of boxes k is larger than the number of objects b, only b boxes are matched while the confidence of the others is minimized
- Assignment problem

$$F_{match}(x,l) = \frac{1}{2}\sum_{i,j} x_{ij}||l_i - g_j||_2^2$$

  where $x_ij$ is one if the ith prediction is assigned to the jth ground truth object

- Confidence:

$$F_{conf}(x,c) = -\sum_{i,j} x_{ij} * \log(c_i) - \sum_i (1 - \sum_j x_{ij}) \log 1 - c_j$$

- Speed up training by clustering (kmeans) of ground truth and using it as prior (prior matching)
- Can be defined to output boxes only for a particular class by training the bounding boxes on that class
- Number of parameters grows linearly with number of classes
- Authors argue two step process (region proposal + classification) is better
- Architecture based on AlexNet
- Predicted boxes are merged using non-maxima surpression
- One shot(50%), +2scales (75%)
- OverFeat/ Selective Search are faster but much more expensive

#### 1.9.2 (Re-) Localization

**PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization [5]**

- Relocalizes, is trained on images from the scenes where it is applied
- Accuracy 2m and 3° in 50 $km^2$ outdoors, 0.5m and 5° indoors, 5ms per frame
- ConvNet 23 layers, Image resolution 224x224
- transfer learning from recognition/classification datasets (ConvNet is trained on classification tasks)
- based on GoogleNet, affine regressors instead of softmax
- automatic training data generation (structure from motion)
- learns p from arbitrary global reference frame
- $loss(I) = ||\hat{x} - x||_2 + \beta * ||\hat{q} - \frac{q}{||q||}||_2$
- separating position/orientation led to drop in performance
- PoseNet evaluation at single center crop + Dense PoseNet 128 uniformly spaced crops (time increase 95ms, only slight accuracy increase)
- Training data generated using structure from motion (Cambridge Scene) and 7 Scenes (Microsoft) for indoor

**A Deep Learning Based 6 Degree-of-Freedom Localization Method for Endoscopic Capsule Robots [9]**

- not published yet?
- Uses 6-DOF camera pose directly
- based on GoogleNet (9 Inception modules) trained on ImageNet
- $loss(I) = ||\hat{x} - x||_2 + ||\hat{q} - q||_2$
- Dataset of 10 000 frames taken from LM103 - EDG (EsophagoGastroDuodenoscopy) Simulator
- 0.18cm RMSE on a trajectory of 18cm
- Although 3 different cameras are used and the frames are separated for training and testing, its still the same "stomach". With 10 000 frames on a trajectory of 18 cm, won't the system just recognize the position?
- Ground truth determined by seperate cameras

#### 1.9.3 Object Pose Estimation

**3D generic object categorization localization and pose estimation [7]**

- Other approaches use different class for different poses
- Object model is separated in different parts of the object based on different view points (front view)
- Different parts are connected when another part is visible from the front view via affine transformation
- Generally such models can't handle inter class variations very good or increase in complexity as number of parts is increased. In this paper this is apparently not the case

**Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image Eric [1]**

- Intermediate representation are object coordinates, continious part labeling that are jointly regressed for every pixel in the image
- Based on auto context (Classifiers with several stages)
- (1) (Auto context) Random forest with L1 regularization predicts labels and object coordinates for every pixel (2) Ransac predicts poses from 2d-3d correspondences guided by uncertainty labels (3) Refinement
- Random forest predicts (probability to belong to object + 3d coordinate—given belonging to object)
- Stacked Forests (Auto context) refine output on previous smoothed output (Geodesic Forest). The smoothing is done to enforce coupling of neighbors
- RANSAC formulates hypothesis by drawing 4 correspondences and solving PnP
- Outperforms PoseNet in indoor localization
- 6D within 5cm and 5 degree only 40 % (With RGB-D 82.5%), on other set 50 % with unknown scene average median error 8.5cm 3.3
- Biggest translational error in z direction
- Multi object detection/pose estimation in 1-4 seconds, not optimized, most time spend in searching for object hypothesis

**A Comparative Analysis and Study of Multiview CNN Models for Joint Object Categorization and Pose Estimation[3]**

- While detection needs pose invariant features, pose estimation needs the pose
- Single instance 3d model
- Discrete pose approaches (pose as classification)
- Trains pose regressor and classifier on output of different levels to measure quality of features
- Later layers "forget" about pose, paper suggests early branching

#### 1.9.4 Other

**Deformable Convolutional Networks [2]**

- Addresses problem of modeling geometric transformations
- Introduces *Deformable Convolution* which adds 2D offsets to the regular sampling grid. The offsets are learned from the data.
- Introduces *Deformable RoI pooling* which adds offsets to bins of pooling layers. The offsets are also learned from the data.
- Further alternatives to have more variable feature maps: Spatial Transformer Networks, Active Convolution, Effective Receptive Field, Atrous Convolution, DeepID-Net, Spatial manipulation in RoI pooling (handcrafted), DPM (handcrafted)
- Light-weight version of STN, easier to train and to integrate
- Receptive fields seem to scale with the size of objects
- Model complexity is increased by only 1-2

Table 1: Object Detection

| | Traditional | | | Deep | | | |
|---|---|---|---|---|---|---|---|
| | Viola&Jones | HoG | DPM | R-CNN | YOLO | SSD | OverFeat |
| Feature Detector | Haar | HoG | Multiple Hogs and virtual springs | Learned by CNN | Learned by CNN | | |
| Detection | Sliding Window, high filter responses indicate there is an object | | | NN in sliding window detects regions for possible objects, For each proposed region a classification is run | Image is split in Grid each Grid spawns Bounding boxes and gives class probabilities | | |
| Accuracy (voc) | | | | 73.2 mAP | 63.4 mAP | 74.3 mAP | |
| Speed | | | | 7 FPS (Faster-RCNN) | 45 FPS | 59 FPS | |
| Strengths | | | | | | | |
| Weaknesses | | | | | | | |

# References


[1] Brachmann, E., Michel, F., Krull, A., Yang, M. Y., Gumhold, S., and Rother, C. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image.

[2] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. Deformable Convolutional Networks.

[3] Elhoseiny, M., El-Gaaly, T., Bakry, A., and Elgammal, A. A Comparative Analysis and Study of Multiview CNN Models for Joint Object Categorization and Pose Estimation.

[4] Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. Scalable Object Detection using Deep Neural Networks.

[5] Kendall, A., Grimes, M., and Cipolla, R. PoseNet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision* (may 2015), vol. 2015 Inter, pp. 2938–2946.

[6] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. SSD: Single Shot MultiBox Detector.

[7] Savarese, S., and Fei-Fei, L. 3D generic object categorization, localization and pose estimation.

[8] Tripathi San Diego, S. U., Dane, G., Kang, B., Bhaskaran, V., and Nguyen, T. LCDet: Low-Complexity Fully-Convolutional Neural Networks for Object Detection in Embedded Systems.

[9] Turan, M., Almalioglu, Y., Konukoglu, E., and Sitti, M. A Deep Learning Based 6 Degree-of-Freedom Localization Method for Endoscopic Capsule Robots.

[10] Wu, B., Iandola, F., Jin, P. H., and Keutzer, K. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving.