One Shot Detectors

Philipp Duernay

February 2, 2018

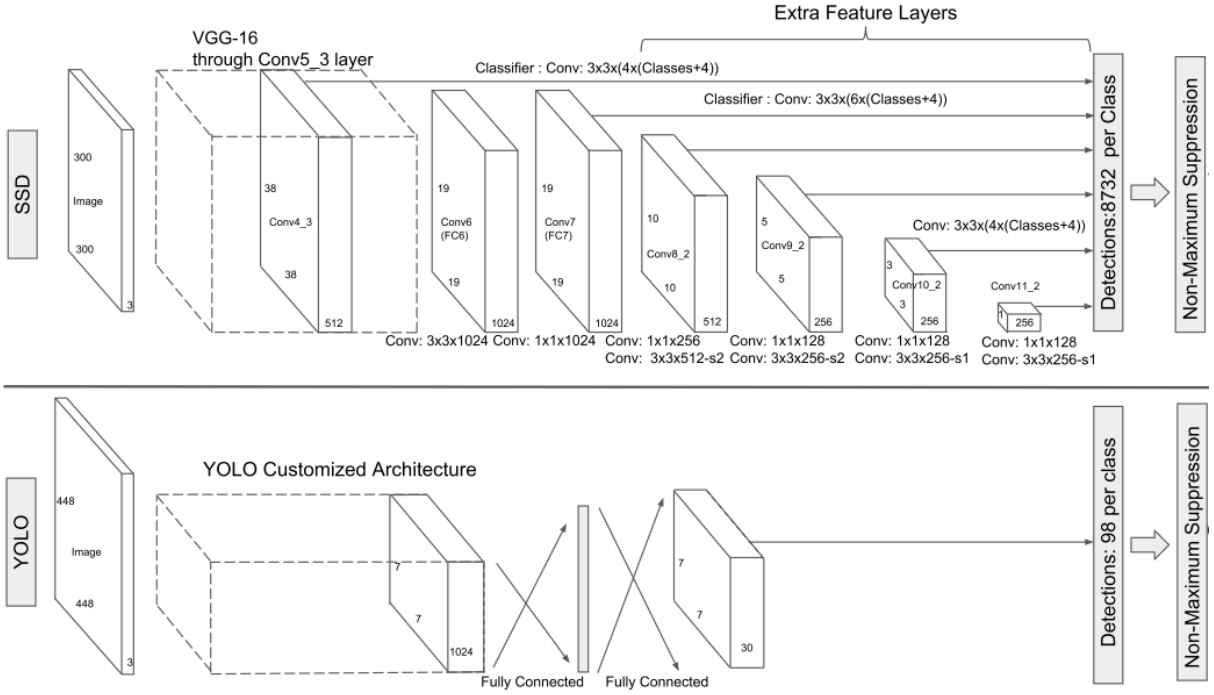

Figure 1: Two One Shot Detectors (old) Yolo and SSD [**?**]

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.

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.

3 Architecture

Usually a classification network like VGG-16 is used as *base network*. On the final feature map additional convolutional layers are set, that predict class confidences and box coordinates. As for each anchor box a separate output is predicted, each predictor can "specialize" on objects that mostly fall in its aspect ratio/scale/ location (?). The networks implement

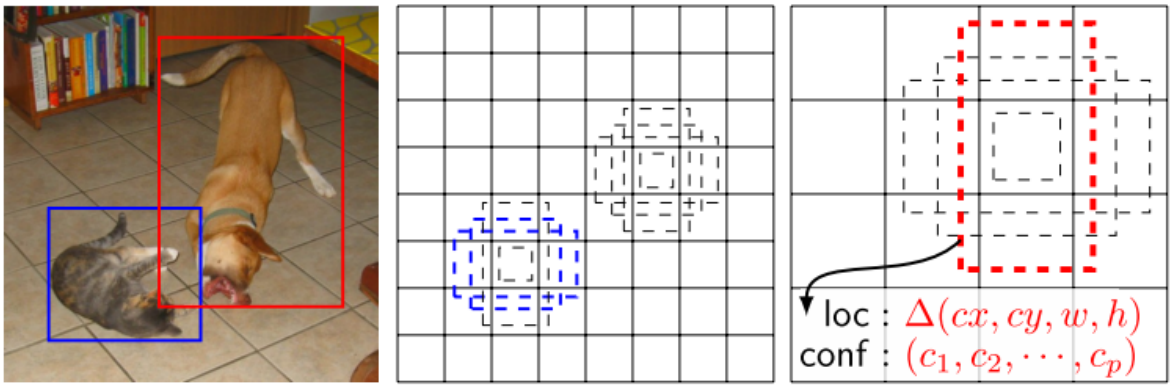

Figure 2: Illustration for anchor box concept [**?**]

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 **??** used a fully connected layer as regression head. However, in the new version they also switched to one convolutional layer.

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:

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

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

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

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

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

4.1 Negatives

5 Data Augmentation

6 Comparison

6.1 Yolo

a. Only one convolutional layer for prediction

b. Anchor boxes determined in pre-clustering step

6.2 SSD

a. 6 Convolutional layers

References