SSD

Contents

Note: As in YOLO, Fast-RCNN, FCN, etc. two types of boxes are used

1. Ground-truth boxes called in the paper "ground-truth boxes"...
2. Default boxes. Those are the predictions the model is generating. They are called also prior boxes, anchor boxes.

Research Paper

The paper is available here.

Main properties

- Single deep neural network
- Output is a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes.
  Then a non-maximum suppression step is achieved to filter boxes with low scores.
- Base network is VGG16.
- The model is more accurate and faster than YOLO.
- The core is to predict category scores and box offsets for a fixed set of default bounding boxes using small convolutional filters applied to feature maps.
- Predictions are done on different scales from feature maps of different sizes making the model being able to detect objects with different sizes.

Multi-scale feature maps for detection

Convolutional layers are added on top of some of the base network layers. As the size of the feature maps decrease, those convolutional layers will produce feature maps of different size (through pooling).

Convolutional predictors for detection

On top of the previous feature layers that were added, are set convolutional filters for the purpose of detection predictions.
For a feature map of size $M * N$ and P channels, the element for prediction is sized $3 * 3 * P$ (kernel). This kernel is applied at each $M * N$ locations to produce either a confidence score or an offset on the default position of the anchor boxes.

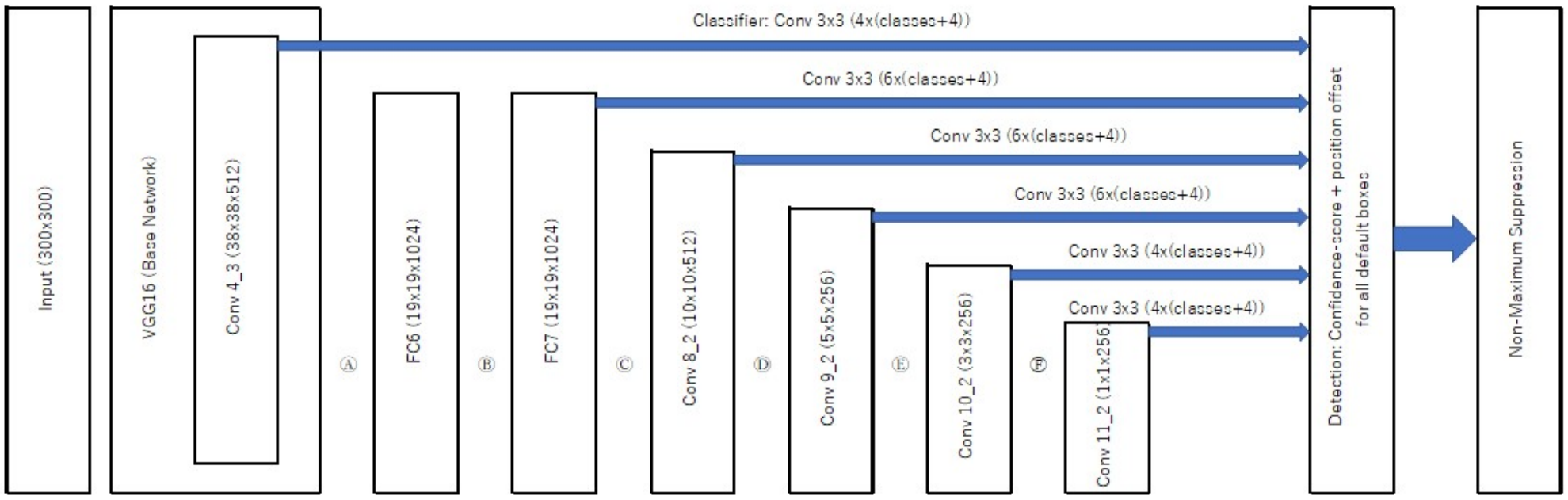
Default boxes and aspect ratios

For each feature map cell, a set of default (anchors, priors, etc.) bounding boxes is applied (meaning generated by the model) and for each of those cells, offsets and per-class confidence score for the boxes are predicted.
The boxes in a set are generated with different size / aspect ratios.
There are Kboxes set on each feature map cell, Cclasses and 4 offsets, so in total $(C+4)*K$filters.
So for a feature map, it climbs up to $(C+4)*K*M*N$.

Architecture

From the Paper

The basic architecture of the SSD is given below.



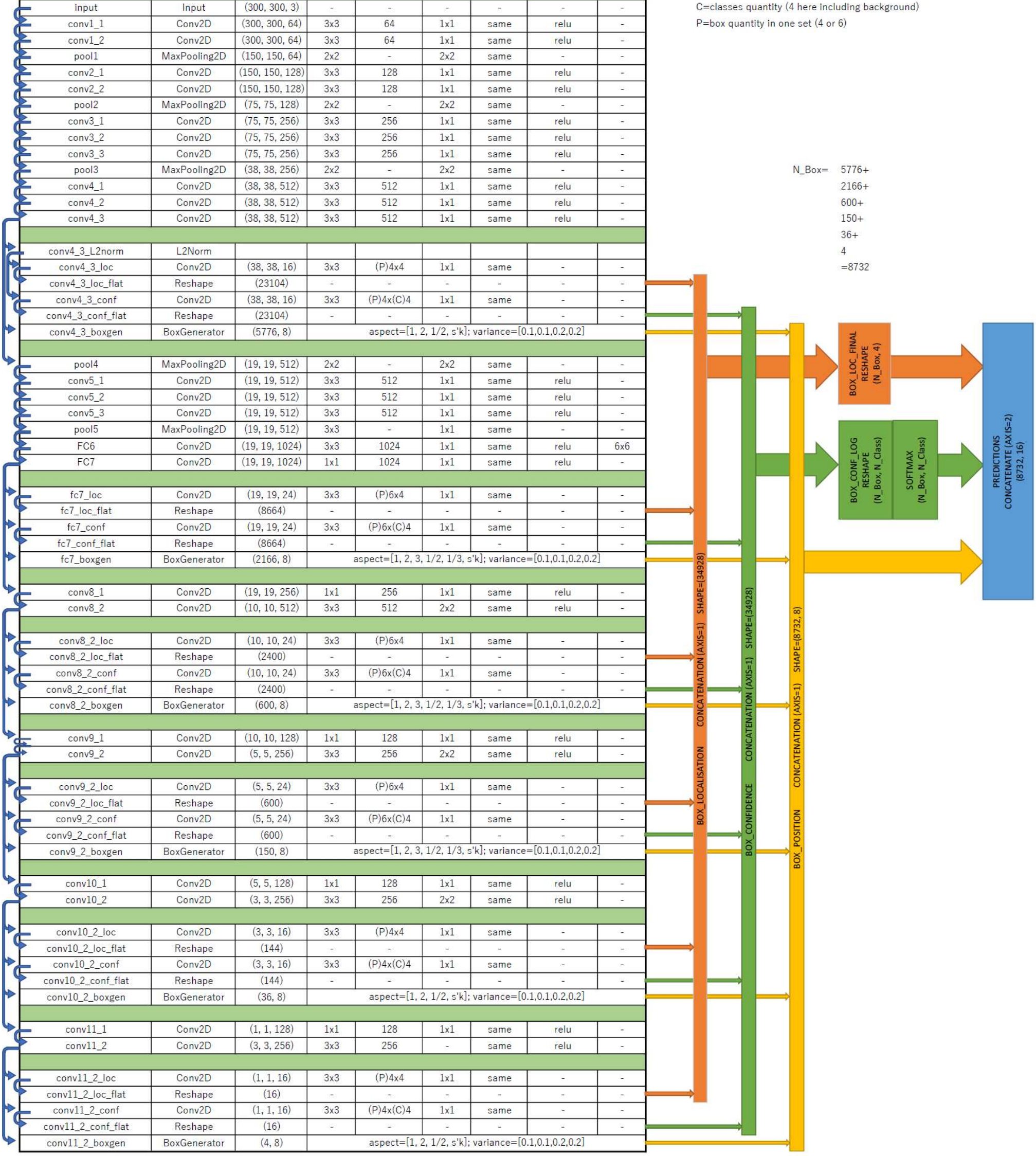
Ⓐ Convolution 3x3x1024
Ⓑ Convolution 1x1x1024
Ⓒ Convolution 1x1x256 + 3x3x256-s2
Ⓓ Convolution 1x1x128 + 3x3x256-s2
Ⓔ Convolution 1x1x128 + 3x3x256-s1
Ⓕ Convolution 1x1x128 + 3x3x256-s1

From an implementation

Name	Type	Output Shape	Kernel	Filters	Stride	Padding	Activation	Dilation
input	Input	(300, 300, 3)	-	-	-	-	-	-
conv1_1	Conv2D	(300, 300, 64)	3x3	64	1x1	same	relu	-
conv1_2	Conv2D	(300, 300, 64)	3x3	64	1x1	same	relu	-
pool1	MaxPooling2D	(150, 150, 64)	2x2	-	2x2	same	-	-
conv2_1	Conv2D	(150, 150, 128)	3x3	128	1x1	same	relu	-
conv2_2	Conv2D	(150, 150, 128)	3x3	128	1x1	same	relu	-
pool2	MaxPooling2D	(75, 75, 128)	2x2	-	2x2	same	-	-
conv3_1	Conv2D	(75, 75, 256)	3x3	256	1x1	same	relu	-
conv3_2	Conv2D	(75, 75, 256)	3x3	256	1x1	same	relu	-
conv3_3	Conv2D	(75, 75, 256)	3x3	256	1x1	same	relu	-
pool3	MaxPooling2D	(38, 38, 256)	2x2	-	2x2	same	-	-
conv4_1	Conv2D	(38, 38, 512)	3x3	512	1x1	same	relu	-
conv4_2	Conv2D	(38, 38, 512)	3x3	512	1x1	same	relu	-
conv4_3	Conv2D	(38, 38, 512)	3x3	512	1x1	same	relu	-
conv4_3_L2norm	L2Norm							
conv4_3_loc	Conv2D	(38, 38, 16)	3x3	(P)4x4	1x1	same	-	-
conv4_3_loc_flat	Reshape	(23104)	-	-	-	-	-	-
conv4_3_conf	Conv2D	(38, 38, 16)	3x3	(P)4x(C)4	1x1	same	-	-
conv4_3_conf_flat	Reshape	(23104)	-	-	-	-	-	-
conv4_3_boxgen	BoxGenerator	(5776, 8)	aspect=[1, 2, 1/2, s'k]; variance=[0.1,0.1,0.2,0.2]					
pool4	MaxPooling2D	(19, 19, 512)	2x2	-	2x2	same	-	-
conv5_1	Conv2D	(19, 19, 512)	3x3	512	1x1	same	relu	-
conv5_2	Conv2D	(19, 19, 512)	3x3	512	1x1	same	relu	-
conv5_3	Conv2D	(19, 19, 512)	3x3	512	1x1	same	relu	-
pool5	MaxPooling2D	(19, 19, 512)	3x3	-	1x1	same	-	-
FC6	Conv2D	(19, 19, 1024)	3x3	1024	1x1	same	relu	6x6
FC7	Conv2D	(19, 19, 1024)	1x1	1024	1x1	same	relu	-
fc7_loc	Conv2D	(19, 19, 24)	3x3	(P)6x4	1x1	same	-	-
fc7_loc_flat	Reshape	(8664)	-	-	-	-	-	-
fc7_conf	Conv2D	(19, 19, 24)	3x3	(P)6x(C)4	1x1	same	-	-
fc7_conf_flat	Reshape	(8664)	-	-	-	-	-	-
fc7_boxgen	BoxGenerator	(2166, 8)	aspect=[1, 2, 3, 1/2, 1/3, s'k]; variance=[0.1,0.1,0.2,0.2]					
conv8_1	Conv2D	(19, 19, 256)	1x1	256	1x1	same	relu	-
conv8_2	Conv2D	(10, 10, 512)	3x3	512	2x2	same	relu	-
conv8_2_loc	Conv2D	(10, 10, 24)	3x3	(P)6x4	1x1	same	-	-
conv8_2_loc_flat	Reshape	(2400)	-	-	-	-	-	-
conv8_2_conf	Conv2D	(10, 10, 24)	3x3	(P)6x(C)4	1x1	same	-	-
conv8_2_conf_flat	Reshape	(2400)	-	-	-	-	-	-
conv8_2_boxgen	BoxGenerator	(600, 8)	aspect=[1, 2, 3, 1/2, 1/3, s'k]; variance=[0.1,0.1,0.2,0.2]					
conv9_1	Conv2D	(10, 10, 128)	1x1	128	1x1	same	relu	-
conv9_2	Conv2D	(5, 5, 256)	3x3	256	2x2	same	relu	-
conv9_2_loc	Conv2D	(5, 5, 24)	3x3	(P)6x4	1x1	same	-	-
conv9_2_loc_flat	Reshape	(600)	-	-	-	-	-	-
conv9_2_conf	Conv2D	(5, 5, 24)	3x3	(P)6x(C)4	1x1	same	-	-
conv9_2_conf_flat	Reshape	(600)	-	-	-	-	-	-
conv9_2_boxgen	BoxGenerator	(150, 8)	aspect=[1, 2, 3, 1/2, 1/3, s'k]; variance=[0.1,0.1,0.2,0.2]					
conv10_1	Conv2D	(5, 5, 128)	1x1	128	1x1	same	relu	-
conv10_2	Conv2D	(3, 3, 256)	3x3	256	2x2	same	relu	-
conv10_2_loc	Conv2D	(3, 3, 16)	3x3	(P)4x4	1x1	same	-	-
conv10_2_loc_flat	Reshape	(144)	-	-	-	-	-	-
conv10_2_conf	Conv2D	(3, 3, 16)	3x3	(P)4x(C)4	1x1	same	-	-
conv10_2_conf_flat	Reshape	(144)	-	-	-	-	-	-
conv10_2_boxgen	BoxGenerator	(36, 8)	aspect=[1, 2, 1/2, s'k]; variance=[0.1,0.1,0.2,0.2]					
conv11_1	Conv2D	(1, 1, 128)	1x1	128	1x1	same	relu	-
conv11_2	Conv2D	(3, 3, 256)	3x3	256	-	same	relu	-
conv11_2_loc	Conv2D	(1, 1, 16)	3x3	(P)4x4	1x1	same	-	-
conv11_2_loc_flat	Reshape	(16)	-	-	-	-	-	-
conv11_2_conf	Conv2D	(1, 1, 16)	3x3	(P)4x(C)4	1x1	same	-	-
conv11_2_conf_flat	Reshape	(16)	-	-	-	-	-	-
conv11_2_boxgen	BoxGenerator	(4, 8)	aspect=[1, 2, 1/2, s'k]; variance=[0.1,0.1,0.2,0.2]					

C=classes quantity (4 here including background)
P=box quantity in one set (4 or 6)

N_Box= 5776+
2166+
600+
150+
36+
4
=8732



L2 Normalization Layer

The paper explains that it is needed to scale the features map on top of the conv4_3 layer using a L_2normalization.
This normalization is described in the paper "ParseNet: Looking wider to see better" and is detailed below:

$$\hat{x} = \frac{x}{||x||_2}$$with $||x||_2$the L_2norm.

$$||x||_2 = (\sum_{i=1}^{d} |x_i|^2)^{\frac{1}{2}}$$

Box/Prior Generation Layer

One important part of the architecture is its box generation layer. This layer basically gets as input the size of the feature map from a previous convolutional layer (for example conv8_2) and based on the scale/ratio policy described below, it generates the coordinates and sizes of all the default boxes for that feature map.
This layer is important because by using it, the final prediction contains the position of any of the default boxes.
In case of $N_{class} = 20$, one final prediction will have for size $20 + 1 + 4 + 8$.

- 20: N_{class}confidence scores.
- 1: Background class confidence score in case of a negative box.
- 4: Predicted position offsets for the box (X_{left}, X_{right}, Y_{top}, Y_{bottom}).
- 8: 4 positions plus 4 variances. Those are the positions generated by this layer.

Anchor / Default Boxes

A note on the relative box coordinates used internally by the model: It is important to understand that it is not possible for the model to predict absolute coordinates for the predicted bounding boxes. In order to be able to predict absolute box coordinates, the convolutional layers responsible for localization would need to produce different output values for the same object instance at different locations within the input image.
This is not possible, since for a given input to the filter of a convolutional layer, the filter will produce the same output regardless of the spatial position within the image because of the shared weights.
This is the reason why the model predicts offsets to anchor boxes instead of absolute coordinates, and why during training, absolute ground-truth coordinates need to be converted to anchor box offsets.

Choosing Scales and Ratios

We have Tfeature maps for prediction.

The scale of the default boxes for each feature map is defined as $S_k = S_{min} + \frac{S_{max} - S_{min}}{T - 1} * (K - 1)$, with $S_{min} = 0.2$, $S_{max} = 0.9$, $K \in [1, T]$
The lowest layer then has a 0.2 scale (S_{min}), while the highest layer has 0.9 (S_{max}).

For $K = 6$we have $S_K \in \{0.2, 0.34, 0.48, 0.62, 0.76, 0.9\}$

Aspect ratios are defined by $A_R \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$and then the corresponding width and height of the default boxes are $W_K^A = S_K * \sqrt{A_R}$, $H_K^A = \frac{S_K}{\sqrt{A_R}}$

For $A_R = 1$we add the scale $S'_K = \sqrt{S_K * S_{K+1}}$.

We can have then 6 default boxes per feature map location.

Following the architecture in the paper, we have 6 prediction layers ($M = 6$ and $k \in [1, 6]$).

Note: The layers "**Conv 4_3**", "**Conv 10_2**" and "**Conv 11_2**" are set with only three ratios $A_R^{1,5,6} \in \{1, 2, \frac{1}{2}\}$(but there is still the additional scale S'_K).

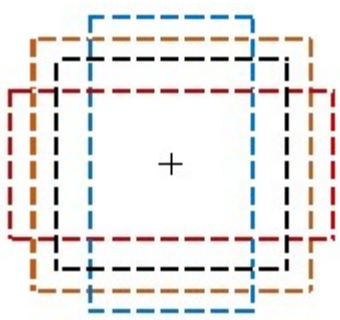
We summarize below the configuration of scales and ratio for the SSD architecture (document for the default boxes display is here).

- $K = 1$

Layer	S_K	A_R^K
Conv 4_3	$S_1 = 0.2$ $+S'_K = \sqrt{S_1 * S_2} = 0.26$	$A_R^1 \in \{1, 2, \frac{1}{2}\}$

The height and width of the default boxes for this level are:

W_K^R	Formula	Value	H_K^R	Formula	Value
W_1^1	$S_1 * \sqrt{1}$	0.2	H_1^1	$\frac{S_1}{\sqrt{1}}$	0.2
W_1^2	$S_1 * \sqrt{2}$	0.28	H_1^2	$\frac{S_1}{\sqrt{2}}$	0.14
W_1^3	$S_1 * \sqrt{\frac{1}{2}}$	0.14	H_1^3	$\frac{S_1}{\sqrt{\frac{1}{2}}}$	0.28

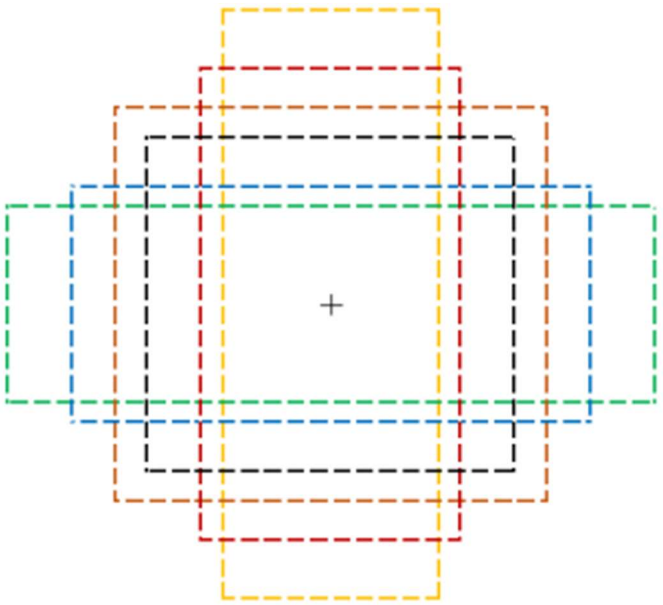


- $K = 2$

Layer	S_K	A_R^K
FC7	$S_2 = 0.34$ $+S'_K = \sqrt{S_2 * S_3} = 0.4$	$A_R^2 \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$

The height and width of the default boxes for this level are:

W_K^R	Formula	Value	H_K^R	Formula	Value
W_2^1	$S_2 * \sqrt{1}$	0.34	H_2^1	$\frac{S_2}{\sqrt{1}}$	0.34
W_2^2	$S_2 * \sqrt{2}$	0.48	H_2^2	$\frac{S_2}{\sqrt{2}}$	0.24
W_2^3	$S_2 * \sqrt{3}$	0.59	H_2^3	$\frac{S_2}{\sqrt{3}}$	0.2
W_2^4	$S_2 * \sqrt{\frac{1}{2}}$	0.24	H_2^4	$\frac{S_2}{\sqrt{\frac{1}{2}}}$	0.48
W_2^5	$S_2 * \sqrt{\frac{1}{3}}$	0.2	H_2^5	$\frac{S_2}{\sqrt{\frac{1}{3}}}$	0.59
W_2^6	$\sqrt{S_2 * S_3}$	0.4	H_2^6	$\sqrt{S_2 * S_3}$	0.4

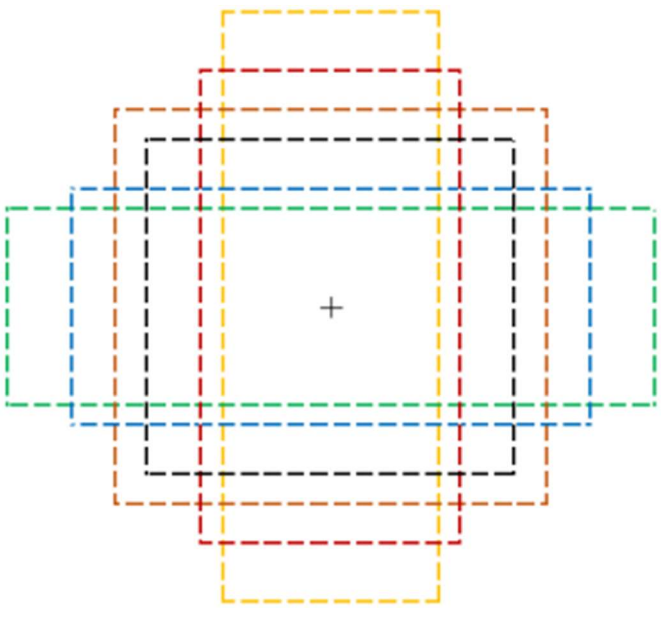


- $K = 3$

Layer	S_K	A_R^K
Conv 6_2	$S_3 = 0.48$ $+S'_K = \sqrt{S_3 * S_4} = 0.55$	$A_R^3 \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$

The height and width of the default boxes for this level are:

W_K^R	Formula	Value	H_K^R	Formula	Value
W_3^1	$S_3 * \sqrt{1}$	0.48	H_3^1	$\frac{S_3}{\sqrt{1}}$	0.48
W_3^2	$S_3 * \sqrt{2}$	0.68	H_3^2	$\frac{S_3}{\sqrt{2}}$	0.34
W_3^3	$S_3 * \sqrt{3}$	0.83	H_3^3	$\frac{S_3}{\sqrt{3}}$	0.28
W_3^4	$S_3 * \sqrt{\frac{1}{2}}$	0.34	H_3^4	$\frac{S_3}{\sqrt{\frac{1}{2}}}$	0.68
W_3^5	$S_3 * \sqrt{\frac{1}{3}}$	0.28	H_3^5	$\frac{S_3}{\sqrt{\frac{1}{3}}}$	0.83
W_2^6	$\sqrt{S_3 * S_4}$	0.55	H_2^6	$\sqrt{S_3 * S_4}$	0.55

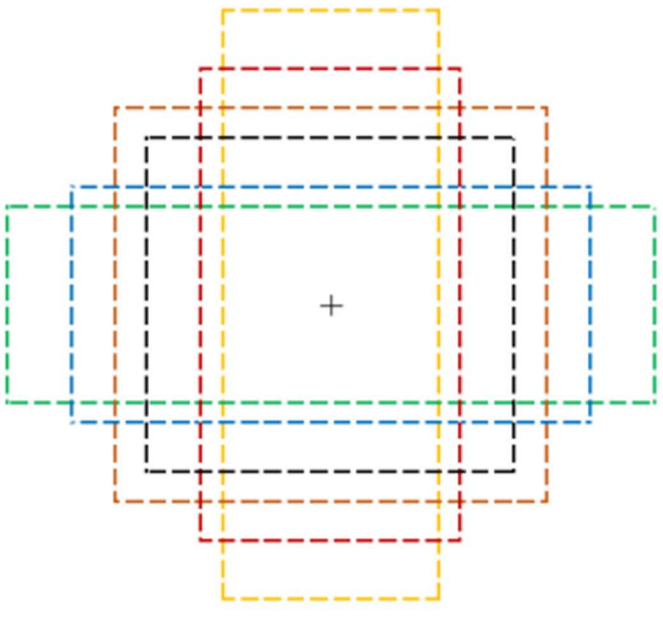


- $K = 4$

Layer	S_K	A_R^K
Conv 7_2	$S_4 = 0.62$ $+S'_K = \sqrt{S_4 * S_5} = 0.59$	$A_R^4 \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$

The height and width of the default boxes for this level are:

W_K^R	Formula	Value	H_K^R	Formula	Value
W_4^1	$S_4 * \sqrt{1}$	0.62	H_4^1	$\frac{S_4}{\sqrt{1}}$	0.62
W_4^2	$S_4 * \sqrt{2}$	0.88	H_4^2	$\frac{S_4}{\sqrt{2}}$	0.44
W_4^3	$S_4 * \sqrt{3}$	1.07	H_4^3	$\frac{S_4}{\sqrt{3}}$	0.36
W_4^4	$S_4 * \sqrt{\frac{1}{2}}$	0.44	H_4^4	$\frac{S_4}{\sqrt{\frac{1}{2}}}$	0.88
W_4^5	$S_4 * \sqrt{\frac{1}{3}}$	0.36	H_4^5	$\frac{S_4}{\sqrt{\frac{1}{3}}}$	1.07
W_4^6	$\sqrt{S_4 * S_5}$	0.69	H_4^6	$\sqrt{S_4 * S_5}$	0.69

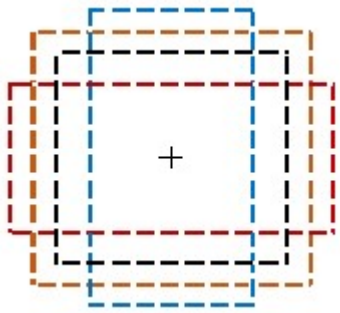


- $K = 5$

Layer	S_K	A_R^K
Conv 8_2	$S_5 = 0.76$ $+S'_K = \sqrt{S_5 * S_6} = 0.83$	$A_R^5 \in \{1, 2, \frac{1}{2}\}$

The height and width of the default boxes for this level are:

W_K^R	Formula	Value	H_K^R	Formula	Value
W_5^1	$S_5 * \sqrt{1}$	0.76	H_5^1	$\frac{S_5}{\sqrt{1}}$	0.76
W_5^2	$S_5 * \sqrt{2}$	1.07	H_5^2	$\frac{S_5}{\sqrt{2}}$	0.54
W_5^3	$S_5 * \sqrt{\frac{1}{2}}$	0.54	H_5^3	$\frac{S_5}{\sqrt{\frac{1}{2}}}$	1.07

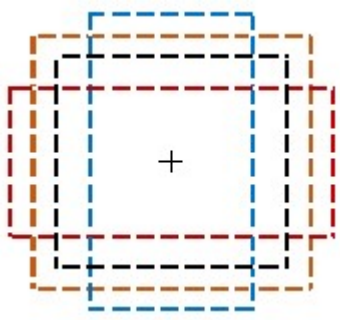


- $K = 6$

Layer	S_K	A_R^K
Conv 9_2	$S_6 = 0.9$ $+S'_K = \sqrt{S_6 * S_7} = ??$	$A_R^6 \in \{1, 2, \frac{1}{2}\}$

The height and width of the default boxes for this level are:

W_K^R	Formula	Value	H_K^R	Formula	Value
W_6^1	$S_6 * \sqrt{1}$	0.9	H_6^1	$\frac{S_6}{\sqrt{1}}$	0.9
W_6^2	$S_6 * \sqrt{2}$	1.27	H_6^2	$\frac{S_6}{\sqrt{2}}$	0.64
W_6^3	$S_6 * \sqrt{\frac{1}{2}}$	0.64	H_6^3	$\frac{S_6}{\sqrt{\frac{1}{2}}}$	1.27



Boxes Centers

Now, regarding the center of the boxes, they are defined as $\left(\frac{i+0.5}{|F_K|}, \frac{j+0.5}{|F_K|}\right)$, F_K: Size of the K-th square feature map, $i, j \in [0, |F_K|)$

The calculations are available is this document and some are given below.

Feature map 10x10:

i/j	0	1	2	3	4	5	6	7	8	9
0	(0.05, 0.05)	(0.05, 0.15)	(0.05, 0.25)	(0.05, 0.35)	(0.05, 0.45)	(0.05, 0.55)	(0.05, 0.65)	(0.05, 0.75)	(0.05, 0.85)	(0.05, 0.95)
1	(0.15, 0.05)	(0.15, 0.15)	(0.15, 0.25)	(0.15, 0.35)	(0.15, 0.45)	(0.15, 0.55)	(0.15, 0.65)	(0.15, 0.75)	(0.15, 0.85)	(0.15, 0.95)
2	(0.25, 0.05)	(0.25, 0.15)	(0.25, 0.25)	(0.25, 0.35)	(0.25, 0.45)	(0.25, 0.55)	(0.25, 0.65)	(0.25, 0.75)	(0.25, 0.85)	(0.25, 0.95)
3	(0.35, 0.05)	(0.35, 0.15)	(0.35, 0.25)	(0.35, 0.35)	(0.35, 0.45)	(0.35, 0.55)	(0.35, 0.65)	(0.35, 0.75)	(0.35, 0.85)	(0.35, 0.95)
4	(0.45, 0.05)	(0.45, 0.15)	(0.45, 0.25)	(0.45, 0.35)	(0.45, 0.45)	(0.45, 0.55)	(0.45, 0.65)	(0.45, 0.75)	(0.45, 0.85)	(0.45, 0.95)
5	(0.55, 0.05)	(0.55, 0.15)	(0.55, 0.25)	(0.55, 0.35)	(0.55, 0.45)	(0.55, 0.55)	(0.55, 0.65)	(0.55, 0.75)	(0.55, 0.85)	(0.55, 0.95)
6	(0.65, 0.05)	(0.65, 0.15)	(0.65, 0.25)	(0.65, 0.35)	(0.65, 0.45)	(0.65, 0.55)	(0.65, 0.65)	(0.65, 0.75)	(0.65, 0.85)	(0.65, 0.95)
7	(0.75, 0.05)	(0.75, 0.15)	(0.75, 0.25)	(0.75, 0.35)	(0.75, 0.45)	(0.75, 0.55)	(0.75, 0.65)	(0.75, 0.75)	(0.75, 0.85)	(0.75, 0.95)
8	(0.85, 0.05)	(0.85, 0.15)	(0.85, 0.25)	(0.85, 0.35)	(0.85, 0.45)	(0.85, 0.55)	(0.85, 0.65)	(0.85, 0.75)	(0.85, 0.85)	(0.85, 0.95)
9	(0.95, 0.05)	(0.95, 0.15)	(0.95, 0.25)	(0.95, 0.35)	(0.95, 0.45)	(0.95, 0.55)	(0.95, 0.65)	(0.95, 0.75)	(0.95, 0.85)	(0.95, 0.95)

Feature map 5x5:

i/j	0	1	2	3	4
0	(0.1, 0.1)	(0.1, 0.3)	(0.1, 0.5)	(0.1, 0.7)	(0.1, 0.9)
1	(0.3, 0.1)	(0.3, 0.3)	(0.3, 0.5)	(0.3, 0.7)	(0.3, 0.9)
2	(0.5, 0.1)	(0.5, 0.3)	(0.5, 0.5)	(0.5, 0.7)	(0.5, 0.9)
3	(0.7, 0.1)	(0.7, 0.3)	(0.7, 0.5)	(0.7, 0.7)	(0.7, 0.9)
4	(0.9, 0.1)	(0.9, 0.3)	(0.9, 0.5)	(0.9, 0.7)	(0.9, 0.9)

Feature map 3x3:

i/j	0	1	2
0	(0.167, 0.167)	(0.167, 0.5)	(0.167, 0.833)
1	(0.5, 0.167)	(0.5, 0.5)	(0.5, 0.833)
2	(0.833, 0.167)	(0.833, 0.5)	(0.833, 0.833)

Feature map 1x1:

i/j	0
0	(0.5, 0.5)

Training

For training SSD, ground-truth information needs to be assigned to specific outputs in the fixed set of detector outputs.
After the assignment is done, loss function and back-propagation are applied end-to-end.

- Define set of default boxes
- Define detection scales
- Hard negative mining
- Data augmentation

Matching Strategy

1. Match each ground-truth box to the default box with the highest IoU.
2. Match default boxes to any ground-truth where IoU > Threshold (0.5).

This produces a set of positive (object inside) and negative (background) default boxes.

Loss function

$x_{i,j}^{p}$: indicator for matching the i-th default box to the j-th ground-truth box of class P.

Based on the matching strategy, it is possible to have $\sum_{i} x_{i,j}^{P} \geq 1$.

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

N: Quantity of matched default box (if $N = 0$then $L = 0$).
l: Predicted box
g: Ground-truth box

Localization loss

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{C_x, C_y, W, H\}} x_{i,j}^{k} smoothL1(l_i^m - \hat{g}_j^m)$$

with:

$$\hat{g}_j^{C_x} = \frac{(g_j^{C_x} - d_i^{C_x})}{d_i^W}, \quad \hat{g}_j^{C_y} = \frac{(g_j^{C_y} - d_i^{C_y})}{d_i^H}, \quad \hat{g}_j^W = \log \frac{g_j^W}{d_i^W}, \quad \hat{g}_j^H = \log \frac{g_j^H}{d_i^H}$$

W: Width
H: Height
C_x, C_y: Center of the default bounding box (d)

Confidence loss

$$L_{conf} = -\sum_{i \in Pos}^{N} x_{i,j}^{P} \log \hat{C}_i^P - \sum_{i \in Neg} \log \hat{C}_i^0 \quad \text{where} \quad \hat{C}_i^P = \frac{e^{C_i^P}}{\sum_P e^{C_i^P}}$$

This is a softmax.

Hard negative mining

After the matching step, most of the default boxes are negatives (not including anything => background).
The quantity of them can be large so there is an unbalance between positive and negative results.
We have usually 2 or three objects in one picture resulting in 2 or 3 ground-truth boxes but thousands of boxes for background class. In order to decrease this unbalance, we keep only the negatives boxes with the highest confidence loss (with a ratio of 3:1 N/P).

Please refer to Hard Negative Mining for more information about this technique.

Data augmentation

- Use the entire original input image.
- Sample a patch so that the minimum IoU with the object is 0.1, 0.3, 0.5, 0.7 or 0.9.
- Randomly sample a patch.

The size of the sampled patch is [0.1, 1] of the original image. Aspect ratio is [0.5, 2].
The overlapped part of the ground-truth box is kept if its center is in the patch.
After sampling, resize to fix size (bi-cubic interpolation) and then random horizontal flip (0.5), photo-metric distortions.
Those photo-metric distortions are described as :

- Contrast
- Brightness
- Color

All three with a modification value between 0.5 and 1.5.

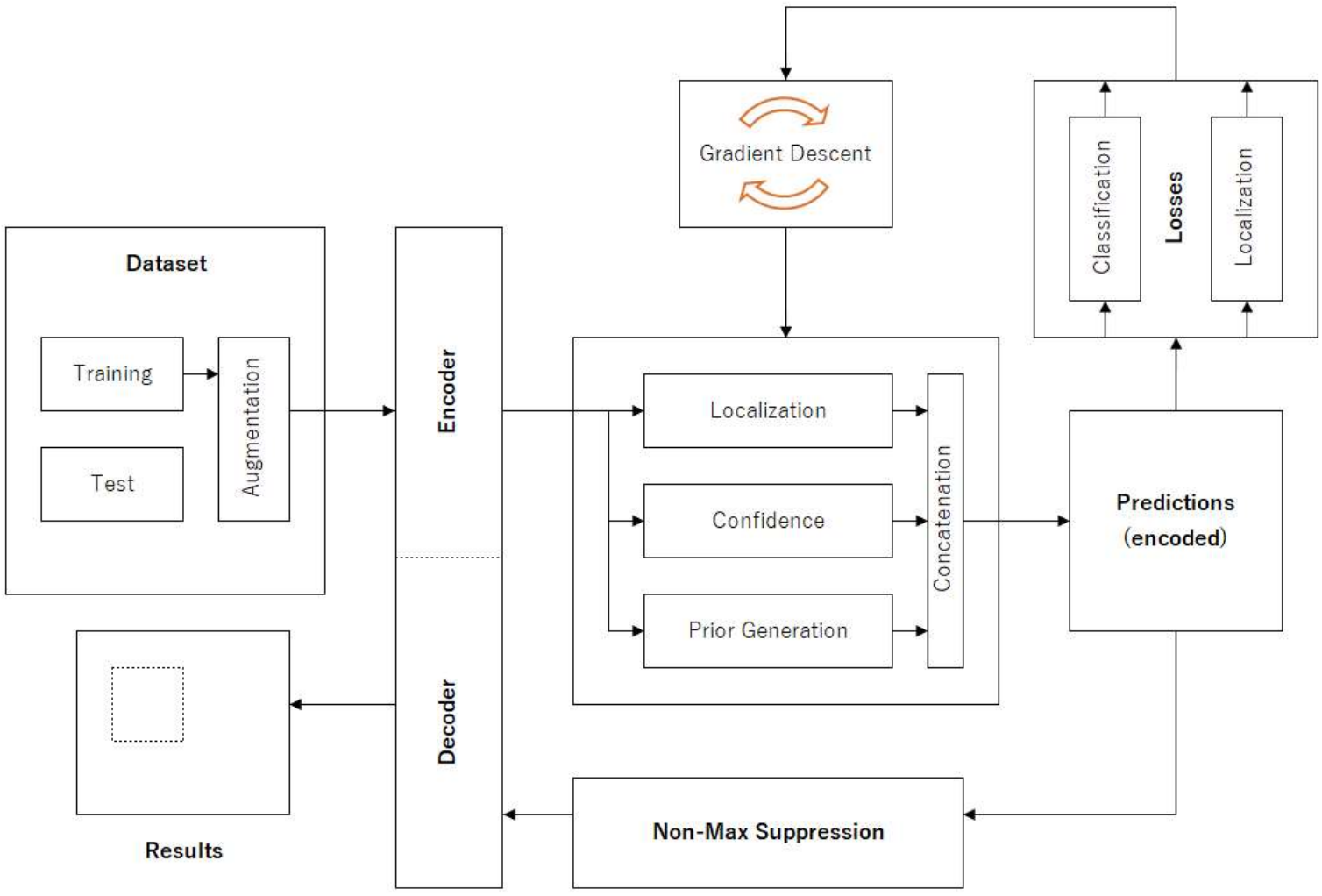
Note: It would be needed to have a custom generator that performs the augmentation AND then implements the matching strategy for the ground-truth boxes. This generator would have to generate also all the default boxes the same way it is done in the box generation layer.

Optimization

SGD, learning rate of 0.001, 0.9 momentum, 0.0005 weight decay and batch_size=32.

Implementation

The implementation of the SSD detector is more complex than a standard classification task implementation.
It involves different parts that are summarized below.



The important parts are:

- Augmentation: Data augmentation is needed for the training. It has been proven that doing augmentation can increase the mAP of the SSD model.
- Encoder & Decoder: The ground-truth boxes have to be encoded in the model's format before doing training. The predictions from the model are in the encoded format so when we want to get the boxes, we need to decode the predictions.
- Losses: The loss function for training the model needs to be calculated in a particular way. A special care for the function is necessary
- Model: The model is itself implemented directly using the API (Keras) however, it requires to generate the default boxes during the training. This is done by the "prior generation" layer.
- Non-Max Suppression: This will filter the default boxes in order to get only the best detection as results.

The basic steps are:

1. Use the custom generator to generate batch_size data for training and validation. The generator gets data from the dataset (a bunch of images and annotations for ground-truth boxes), augments the pictures (for x_train), generates the default boxes (for y_train) and outputs that to the model.
2. The model starts by fetching its training data to the convolutional blocks, they are processed and feature maps are output.
3. Some of those feature maps are taken and entered into the prediction branches, made-up of additional convolutional layers.
4. One prediction branch has two leaves, one for the box offsets prediction and the other one for the box per-class confidence scores.
5. The box generation layer works in parallel with the prediction branch and gives a set of default boxes (positions and variances) so that at each output of the prediction branch, we have in fact
    1. Position offsets
    2. Per-class confidence scores
    3. Positions and variance
6. The output of the box offsets prediction layers are concatenated.
7. The output of the per-class confidence score prediction layers are concatenated and entered in a softmax activation layer.
8. The output of the box generation layers are concatenated.
9. Everything is concatenated, we have then our y_pred.
10. y_pred and y_true are compared using the two loss functions.
11. Gradient-descent happens, the model's weights are updated.
12. Go back to step 1.

The Conv4_3 layer's problem

One issue with the SSD detector is its accuracy on detecting really small objects.
The paper Feature-Fused SSD by Cao et al. claims that this weakness is due to the fact that the conv4_3 layer where the smallest boxes are generated (because they are boxes sized on the 38x38 feature map) is not deep enough and its responses are not "strong" enough yet (they say "lack of semantic information").
They then use a clever concatenation/merge of the conv4_3 and conv5_3 (which is deeper) as output to the prediction branch allowing it to catch more "semantic information".

As the needed modification is quite simple, this can be easily added to the SSD implementation for a small boost of mAP without losing much in detection speed.