SSD: Single Shot MultiBox Detector

Wei Liu
University of North Carolina at Chapel Hill

Dragomir Anguelov
Google Inc.

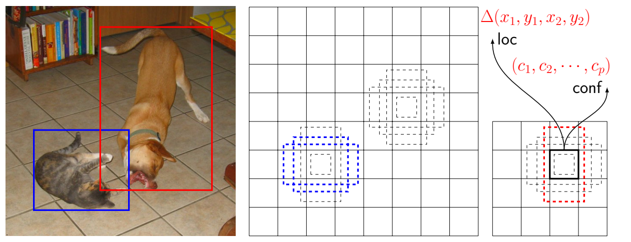
Dumitru Erhan
Google Inc.

SSD: Single Shot MultiBox Detector

Wei Liu[1], Dragomir Anguelov[2], Dumitru Erhan[2], Christian Szegedy[2], Scott Reed[3]
[1]UNC Chapel Hill [2]Google Inc. [3]University of Michigan, Ann-Arbor
[1]wliu@cs.unc.edu, [2]{dragomir,dumitru,szegedy}@google.com, [3]reedscot@umich.edu

Abstract

We present a method for detecting objects in images using a single deep neural network. Our approach, named SSD, discretizes the output space of bounding boxes into a set of bounding box priors over different aspect ratios and scales per feature map location. At prediction time, the network generates confidences that each prior corresponds to objects of interest and produces adjustments to the prior to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. Our SSD model is simple relative to methods that requires object proposals, such as R-CNN and MultiBox, because it completely discards the proposal generation step and encapsulates all the computation in a single network. This makes SSD easy to train and straightforward to integrate into systems that require a detection component. Experimental results on ILSVRC DET and PASCAL VOC dataset confirm that SSD has comparable performance with methods that utilize an additional object proposal step and yet is 100-1000× faster. Compared to other single stage methods, SSD has similar or better performance, while providing a unified framework for both training and inference.



(a) Input image (b) 8×8 feature map (c) 4×4 feature map

Figure 1. **SSD framework.** (a) SSD only needs an input image (and ground truth boxes for each object during training). At each location for several feature maps (e.g. 8×8 and 4×4 in (b) and (c)), we impose a small set of (e.g. 4) designated (normalized) convolutional priors of different aspect ratios and scales. For each prior, we use the underlying (1×1) feature to predict both the offsets $(\Delta(x_1, y_1, x_2, y_2))$ and the confidences for all object categories $((c_1, c_2, \cdots, c_p))$. During training time, we first match these priors to the ground truth. For example, we have matched two priors with the cat and one with the dog, which are treated as positives and the rest negatives. Then we compute the loss with a weighted sum between localization loss (e.g. L_2 loss) and confidence loss (e.g. multi-class logistic), and backpropagate the error.

1. Introduction

Deep convolutional neural networks [12, 20, 19] have recently demonstrated impressive levels of performance on the image classification task. However, object detection is a more difficult problem, because in addition to classifying objects it also requires localizing all object instances in an image. Region-based Convolutional Neural Networks (R-CNN) [6] or its faster variants [8, 5, 16] approach detection as a classification problem over object proposals, followed by (optional) regression of the bounding box coordinates. Alternatively, YOLO [15] directly predicts bounding boxes and confidences for all categories over a fixed grid using the whole topmost feature map; OverFeat [18] regresses a bounding box per feature map location after knowing the category of the underlying object. However these single shot methods do not perform as well as R-CNN type methods. So is the R-CNN framework the only method for achieving high quality object detections?

In this paper, as shown in Figure 1, we present Single Shot MultiBox Detector (SSD). Given a set of fixed bounding box priors of different aspect ratios and scales, we train a network to select which priors contain objects of interest and to adjust their coordinates to better match the object's shape. The set of bounding box priors we define is in the same spirit as the anchor boxes used in Faster R-CNN [16]. Unlike Faster R-CNN, our network can detect multiple object categories without the need to share convolution layers with Fast R-CNN [5]. Furthermore, the SSD architecture combines predictions from multiple feature maps at different resolutions in the network, which naturally handles objects of different sizes and improves detection quality.

Overall, SSD shares insights with many concurrent works [18, 2, 8, 5, 15, 16]. However, it is the first one to combine the most effective single shot detection ideas in a unified framework and to achieve performance competitive with object proposal based methods yet 100-1000$\times$ faster on the ILSVRC DET [17] and PASCAL VOC [3] detection benchmarks. We summarize our contributions as follows:

- SSD removes the object proposal step, which is typically expensive and reliant on low-level region grouping cues, as opposed to [8, 2, 5, 16]. The method is based on a single neural network that is capable of directly generating object bounding boxes and their confidences for a large number of object categories.
- SSD is the first work to combine the idea of convolutional bounding box priors with the ability to predict multiple object categories. In our method, the offset adjustment and confidences for multiple categories of each prior are predicted from the underlying 1×1 feature at each location on a feature map, as opposed to the whole feature map as done in MultiBox [2] and YOLO [15]. This results in a more compact network, which is crucial for efficient detection of a large number of object categories. Additionally, this adds translational invariance in the model output layers, and reduces overfitting and improves detection performance.
- The SSD priors are of different aspect ratios and scales, densely spread out on all locations from a feature map, and thus can cover various object shapes as opposed to YOLO [15] or OverFeat [18], which use a more rigid grid-based tiling.
- SSD is the first method to associate bounding box priors with features maps of different spatial resolution in the network. This naturally handles objects of different scales and improves detection accuracy with negligible computation overhead, as opposed to OverFeat [18] and SPPnet [8], which requires resizing images to different resolutions and individual processing.
- The overall SSD design is very efficient (100-1000$\times$ faster than the best region proposal detector) and provides a unified framework for both training and inference, even for hundreds of object categories.

The rest of the paper is organized as follows. In sec. 2 we review the related work. Sec. 3 describes the details of the SSD model and Sec. 4 reports the object detection results on the ILSVRC DET and PASCAL VOC dataset. We conclude our work in Sec. 5.

2. Related Work

There are two established classes of methods for object detection in images, one based on sliding windows and the other based on region proposal classification. Before the advent of convolutional neural networks, the state of the art for those two approaches – Deformable Part Model (DPM) [4] and Selective Search [23] – had comparable performance. However, after the dramatic improvement brought on by R-CNN [6], which combines selective search region proposals and convolutional network based post-classification, region proposal object detection methods became prevalent.

The original R-CNN approach has been improved in a variety of ways. The first set of approaches improve the quality and speed of post-classification, since it requires the classification of thousands of image crops, which is expensive and time-consuming. SPPnet [8] speeds up the original R-CNN approach significantly. It introduces a spatial pyramid pooling layer that is more robust to region size and scale and allows the classification layers to reuse features computed over feature maps generated at several image resolutions. Fast R-CNN [5] extends SPPnet so that it can fine-tune all layers end-to-end by minimizing a loss for both confidences and bounding box regression, which was first introduced in MultiBox [2] for learning objectness.

The second set of approaches improve the quality of proposal generation using deep neural network. In the most recent works like MultiBox [2, 21], the selective search region proposals, which are based on low-level image features, are replaced by proposals generated directly from a separate deep neural network. This further improves the detection accuracy but results in a somewhat complex setup, requiring the training of two neural networks with a dependency between them. Faster R-CNN [16] replaces selective search proposals by ones learned from a region proposal network (RPN), and introduces a method to integrate the RPN with Fast R-CNN by alternating between fine-tuning shared convolutional layers and prediction layers for these two networks. Our SSD is very similar to Faster R-CNN in that we also use the convolutional priors (or anchor boxes). However, we directly learn to predict both the offsets over these priors and the confidences for multiple categories, instead of treating all categories as class-agnostic object (same as MultiBox). Thus, our approach avoids the complication of merging RPN with Fast R-CNN and is much easier to train and straightforward to integrate in other tasks.

Another set of methods, which are directly related to our approach, skip the proposal step altogether and predict bounding boxes and confidences for multiple categories directly. OverFeat [18], a deep version of the sliding window method, predicts bounding box directly from each location of the topmost feature map after knowing the confidences of the underlying object categories. YOLO [15] uses the whole topmost feature map to predict both confidences for multiple categories and bounding boxes (which are shared for these categories). Our SSD method falls in this category because we do not have the proposal step but use the designated priors instead. However, our approach is more flexible than the existing methods because we can impose priors of different aspect ratios and scales on each feature location from multiple feature maps. If we only use one prior per location from the topmost feature map, our SSD has similar architecture as OverFeat [18]; if we use the whole topmost feature map for predictions instead of using the convolutional priors, we can approximately reproduce YOLO [15].

3. SSD

SSD is inspired by MultiBox [2, 21] and shares a similar training objective, yet has many differences and improvements. While MultiBox is very successful at generating object proposals by training an objectness detector, it still requires post-classification over the generated proposals for full object detection. However, our SSD can detect **multiple categories** with a single shot evaluation of an input image. We will now describe the key differences of our method.

3.1. Training method

Suppose we have n priors in total, denoted as b_i where $i \in [0, n)$. Each prior is associated with a bounding box shape and a set of object category confidences, corresponding to the probability that a specific category is present at the location specified by the prior. Specifically, let c_i^p be the confidence for category p of the i-th prior, $l_i \in \mathbb{R}^4$ be the i-th predicted box coordinates, and $g_j^p \in \mathbb{R}^4$ be the j-th ground truth box coordinates of category p. Note that the predicted box coordinates are computed by adding the offsets output by the network to the corresponding priors, and that both the priors and ground truth boxes are in normalized coordinates, relative to the full image. This normalization ensures the whole image region is the unit box so that we can safely compare coordinates without worrying about different input image sizes.

3.1.1 Matching strategy

Just like the Multibox method, at training time we need to establish the correspondence between the ground truth boxes and the priors. We can either use the prior coordinates directly or use the adjusted prior coordinates after applying the offsets predicted by the network. For simplicity, we refer to the prior-derived coordinates as *source boxes*. The source boxes that are matched to ground truth become the positive examples and the rest are treated as negatives.

We consider two possible matching approaches. The first is *bipartite matching*, where each ground truth box is greedily matched to the source box with the best jaccard overlap. This is the matching approach used by the original Multibox and it ensures that each ground truth box has exactly one matched source box. We also experimented with *per-prediction matching*, which first performs bipartite matching so that each ground truth box has one corresponding source box. It then matches the remaining source boxes to the most overlapped ground truth boxes if the jaccard overlap is higher than a threshold (e.g. 0.5). Unlike bipartite matching, per-prediction matching can generate several positive prior matches for each ground truth. This allows the network to predict high confidences for multiple overlapping priors rather than requiring it to always pick the best possible prior – a slightly simplified task.

Note that SSD aims to detect multiple categories, as opposed to MultiBox which only detects class-agnostic ”objectness” boxes, which are then used in a post-classification step to determine the object category. To detect multiple (hundreds or more) categories, for each prior, SSD predicts a single bounding box adjustment that is shared across all object categories. In other words, the source boxes are treated as class-agnostic during the matching step. However, after the matching, we keep the matched ground truth label as it will be used when computing the confidence loss.

3.1.2 Training objective

The SSD training objective is derived from the Multibox objective [2, 21] but is extended to handle multiple object categories. Let's denote $x_{ij}^p = 1$ to indicate that the i-th source box is matched to the j-th ground truth box of category p, and $x_{ij}^p = 0$ otherwise. As described above, for bipartite matching, we have $\sum_i x_{ij}^p = 1$. If we use per-prediction matching, $\sum_i x_{ij}^p \geq 1$, meaning there can be more than one source box matched to the j-th ground truth box. The overall objective loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf):

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

where the localization loss is L_2 loss between predicted box (not prior) and ground truth box:

$$L_{loc}(x, l, g) = \frac{1}{2} \sum_{i,j} x_{ij}^p ||l_i - g_j^p||_2^2 \tag{2}$$

and the confidence loss can be either multi-class logistic or softmax loss. We use multi-class logistic loss[1]:

$$L_{conf}(x, c) = -\sum_{i,j,p} x_{ij}^p \log(c_i^p) - \sum_{i,p} (1 - \sum_{j,q=p} x_{ij}^q) \log(1 - c_i^p) \tag{3}$$

and the weight term α is set to 0.06 by cross validation.

3.2. Fully convolutional priors

A key ingredient of the MultiBox method is that it uses the k-means centroids of the training set bounding box coordinates as priors. In the original design, the whole topmost feature map is used to predict the offsets for all of the priors. A better strategy might be to use the fully convolutional priors, where we impose a small set of priors per location on

[1] One can also choose softmax loss, we have not compared the multi-class logistic loss with softmax loss at this time.

a feature map as shown in Figure 1. Not only is this more computationally efficient, but it also reduces the number of parameters and thus reduces the risk of over-fitting.

These fully convolutional priors are very similar to the anchor boxes used in RPN [16]. We further simplify the model, by replacing the 3×3 convolution kernels with 1×1 kernels to predict the offsets and confidences and we do not require an intermediate layer. More importantly, RPN is still used to learn objectness and generate proposals which are used to merge with Fast R-CNN. Instead, SSD can be directly trained to detect multiple categories with shared bounding boxes among them. Specifically, suppose we have a $m \times m$ feature map, k priors (encoded with top-left and bottom-right corners) per location on the feature map, and c categories in total. We will have $4k$ offsets outputs for the k priors and ck confidences outputs for the c categories, resulting in $(4 + c)k$ outputs per feature map location. By accumulating predictions across all locations on the feature map, it has $(4+c)km^2$ outputs in total, but only has $(4+c)k$ parameters to learn. If we do not share locations for different categories, the total output will be $5ckm^2$ and has $5ck$ parameters. Obviously, these numbers can grow much rapidly when we have many categories ($c \gg 1$) to detect.

3.3. Combining predictions from multiple feature maps

Most convolutional networks reduce the size of feature maps at the deeper layers. Not only does this reduce computation and memory cost but it also provides some degree of translation and scale invariance. To handle different object scales, some methods [18, 8] suggest converting the image to different sizes, then processing each size individually and combining the results afterwards. However, by utilizing feature maps from several different layers in a single network we can mimic the same effect. [14, 7] have shown that using feature maps from the lower layers can improve semantic segmentation quality because the lower layers capture more fine details of the input objects. Similarly, [13] showed that adding global context pooled from the topmost feature map can help smooth the segmentation results. Motivated by these methods, we use both the lower and upper feature maps for making detection predictions. Figure 1 shows two exemplar feature maps (8×8 and 4×4) which are used in the framework, of course in practice we can use many more with negligible computation overhead.

Feature maps from different levels within a network are known to have different (empirical) receptive field sizes [24]. Fortunately, within the SSD framework, the convolutional priors do not necessary need to correspond to the actual receptive fields of each layer. We can design the tiling so that specific feature map locations learn to be responsive to specific areas of the image and particular scales of the objects. Suppose we want to use m (square) feature maps to do the predictions. For simplicity, we annotate f_k where $k \in [1, m]$ as the size for the k-th feature map in decreasing order. The scale of the priors for each feature map is computed as:

$$s_k = s_{\text{min}} + \frac{s_{\text{max}} - s_{\text{min}}}{m - 1}(k - 1), \tag{4}$$

where s_{min} is 0.1, s_{max} is 0.7, and s_{m+1} is 1, meaning the lowest layer has a scale of 0.1 and the highest layer has a scale of 0.7, and all layers in between are regularly spaced. We impose different aspect ratios for the prior bounding boxes, and denote them as $a_r \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$. We can compute the width ($w_k^a = s_k\sqrt{a_r}$) and height ($h_k^a = s_k/\sqrt{a_r}$) for each prior. For the aspect ratio of 1, we also add an additional prior whose scale is $s'_k = \sqrt{s_k s_{k+1}}$, thus resulting in 6 priors per feature map location. We set the center of each prior to $(\frac{i+0.5}{f_k}, \frac{j+0.5}{f_k})$ where $i, j \in [0, f_k)$ and we truncate the coordinates of the priors such that they are always within [0, 1]. In practice, one can also design his/her own priors for the different detection tasks.

By combining predictions for all priors with different scales and aspect ratios from all locations of many feature maps, we have a diverse set of predictions, covering various input object sizes and shapes. For example, in Figure 1, the dog is matched to a prior in the 4×4 feature map, but not to any priors in the 8×8 feature map. This is because those priors are within different scales and do not match the dog box and thus are considered as negatives during training.

3.4. Hard negative mining

After the matching step, most of the source boxes are negatives, especially when the number of priors is large. This introduces a significant imbalance between the positive and negative training examples. Instead of using all the negative examples, we sort them using the highest confidence across all categories for a source box and pick the top ones so that the ratio between the negatives and positives is at most 3:1. We found that this leads to faster optimization and a more stable training process.

3.5. Image processing

In order to make the model more robust to various input object sizes and shapes, each training image is randomly sampled by one of the following options:

- Use the entire original input image.
- Sample a patch so that the *minimum* jaccard overlap with the objects is 0.1, 0.3, 0.5, or 0.7.
- Sample a patch so that the *maximum* jaccard overlap with the objects is 0.5.

After the aforementioned sampling step, each sampled patch is horizontally flipped with probability of 0.5, in addition to applying some photometric distortions similar to those described in [10].

4. Experimental Results

We report results on two datasets: ILSVRC DET [17] and PASCAL VOC [3], and compare against other related methods. Our experiments start from a specific earlier variant[2] of Inception [22], which has 76.4% top-1 accuracy on the ILSVRC CLS-LOC `val` dataset, and fine-tune it for SSD using batch-normalization [11] and Adagrad [1] with initial learning rate 0.4.

4.1. ILSVRC 2014 DET results

We compare SSD to a top performing two-stage detector [21]. Both detectors start from the exactly same pre-trained Inception network for fair comparison, and are fine-tuned with the ILSVRC2014 DET `train` dataset and evaluated on the `val2` dataset [6].

4.1.1 Baseline two-stage detector

The details of the baseline detector is described in [21]. Specifically, the MultiBox network is used to generate objectness proposals, which are then post-classified for the 200 DET categories using a separate network. The MultiBox model uses the same underlying network structure and convolutional priors as SSD except that it treats all objects as class-agnostic object. The input image size for MultiBox is 299×299, and each proposal is also resized to 299×299 for post-classification. Compared to the winning entry in ILSVRC2014 [20], which has 38.0 mAP with single model and 43.9 mAP with 6 ensemble models on the `test` set, the latest two-stage detector has 44.7 mAP with single model on the `val2` set. This large improvement is due to a better proposal network (MultiBox), an improved post-classification network (Inception), and bigger network input size (299 vs. 224).

4.1.2 SSD vs. Baseline detector

SSD shares a similar objective as MultiBox, however, it has the ability to detect multiple categories in a single shot evaluation instead of using the two-stage method. Table 1 shows how SSD performance changes as we increase the number of categories. To make it a fair comparison with the baseline method, we first use the same input image size (299×299). SSD Person is a monolithic person detector where we treat person as positives and all other 199 categories as negatives. Compared to the baseline result 51.5, SSD Person achieves 52.7 which is slightly better because SSD learns to regress the priors as well as predict the confidences at the same time instead of using two decoupled steps. It is appealing given the dramatic efficiency gains of SSD and the fact that the two-stage method has approximately $2\times$ the number of parameters, since it uses two Inception networks. When we

Models		person	car	dog	mAP	# priors
input size	method					
299×299	Baseline	51.5	47.1	90.4	44.7	1420
299×299	SSD Person	52.7	N/A	N/A	N/A	1420
	SSD 3	46.7	38.8	89.4	N/A	1420
	SSD Full	45.2	38.9	88.7	31.0	1420
	SSD Multi	46.6	43.2	90.2	34.7	9864
443×443	SSD full	50.9	45.0	92.0	39.6	2704

Table 1. **Average precision (AP) on person, car, and dog as the number of categories increases for SSD**. SSD Person is a monolithic person detector. SSD 3 is a detector trained on these three categories. SSD Full is trained on all 200 DET categories and SSD Multi uses two extra lower level feature maps for predictions. Baseline is a two-stage detector [21] with same pre-trained network as SSD. mAP is the mean AP across all 200 DET categories.

include three categories, person, car, and dog, we see that the performance (of SSD 3) on person is worse than SSD Person. We hypothesize that this difference is because the two-stage method has resized each proposal to 299×299 for post-classification, which preserves very fine details even for small objects in the dataset, while SSD takes a whole image with the same resolution and thus loses many details and introduces more confusions. If we go further by training SSD on all 200 DET categories, we see that SSD Full has similar performance on these three categories with SSD 3. Finally, SSD Full achieves 31.0 mAP on the `val2` dataset, which is worse than the two-stage method (44.7) but still very promising given that SSD is about $100\times$ faster, and that SSD is much easier to train and integrate in other systems where detection is needed.

4.1.3 More feature maps are better

We notice that for some objects, such as dog, which are relatively big in the input images, the performance is not much different compared to the two-stage method (88.7 vs. 90.4). However, for other objects, such as car, SSD performs much worse than the two-stage method. We hypothesize that this is because SSD cannot handle those small objects since the input image is too small and loses many details of the small objects. For example, at training time, we resize each input image to 299×299, resulting in 8×8 feature map on top. Following [21], we further reduce the topmost feature map to 6×6, 4×4, 3×3, 2×2, and 1×1 by using extra convolutional layers, impose 1420 manually optimized priors instead of the ones described in Sec. 3.3 on these layers, and use all of these feature maps to do predictions. These feature maps all have large receptive fields, and thus can handle very large objects. But they cannot detect very small objects because of the down sampling effect in both input image and feature maps. Inspired by many works in seman-

[2] Details of the network structure can be found in [21].

[3] `http://host.robots.ox.ac.uk:8080/anonymous/ZOFLVS.html`
[4] `http://host.robots.ox.ac.uk:8080/anonymous/KXJIN8.html`

System	data	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	*mAP*
Fast R-CNN [5]	07+12	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4	70.0
Faster R-CNN [16]	07+12	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6	73.2
SSD (299 × 299)	07+12	64.5	65.6	55.5	54.5	31.5	75.1	65.7	75.2	41.3	53.9	69.9	70.1	70.4	69.9	64.3	36.0	54.9	69.8	78.7	57.0	61.2
SSD (443 × 443)	07+12	71.4	76.2	63.5	60.2	40.7	80.8	76.4	78.6	54.9	72.0	72.3	73.8	76.0	75.9	72.0	49.8	69.1	73.1	77.9	69.7	69.2

Table 2. **PASCAL VOC2007 `test` detection results.** Training data key: "07+12" – VOC07 `trainval` and VOC12 `trainval`. Both Fast R-CNN and Faster R-CNN use input images whose minimum dimension is 600. The two SSD models have exactly the same settings except that they have different input size (299 × 299 vs. 443 × 443). It is obvious that larger input size leads to much better results.

System	data	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	*mAP*
Fast R-CNN [5]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
Fast R-CNN [5]	07++12	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2	68.4
Faster R-CNN [16]	07++12	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5	70.4
YOLO [15]	07++12	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8	57.9
SSD (299 × 299)[3]	07+12	71.2	61.7	49.4	38.6	25.9	69.5	52.0	75.7	34.1	52.3	53.8	69.2	59.2	68.0	63.8	25.8	44.4	53.3	70.7	50.1	54.4
SSD (443 × 443)[4]	07+12	78.6	68.5	60.2	46.4	35.6	74.7	65.5	81.8	46.6	61.0	57.6	75.8	69.4	75.7	72.1	38.2	62.7	60.2	75.9	58.9	63.3

Table 3. **PASCAL VOC2012 `test` detection results.** Training data key: "12" – VOC12 `trainval`, "07+12" – VOC07 `trainval` and VOC12 `trainval`, "07++12" – VOC07 `trainval` and `test` and VOC12 `trainval`. Fast R-CNN and Faster R-CNN use input images whose minimum dimension is 600, while the input size for YOLO is 448 × 448. SSD models are the same as the ones in Table 2.

tic segmentation [14, 7, 13], we also use lower level feature maps to do predictions and combine the predictions for the final detection results. For example, instead of only using the 8 × 8 (and additional upper) feature maps, we also use lower feature maps such as 17 × 17 and 35 × 35 so that these lower feature maps can capture more fine details of the input objects. As a result, we have 9864 priors in total (6 priors per location), far more than 1420. However, the computation overhead is negligible because we use 1 × 1 features for predictions and share the prior for all categories. From Table 1, we can see that SSD Multi, a model combining predictions from lower level feature maps, is obviously better than SSD Full (34.7 vs. 31.0). Thus, including lower level feature maps can help improve performance with negligible additional computation time.

4.1.4 Larger input image size is better

Even if we use the lower feature maps to help improve SSD on small objects, it is still not enough because the input image size is small compared to the average image resolution on ILSVRC DET (299 × 299 vs. 482 × 415). Thus we have experimented with larger input image size (443 × 443), with all other settings the same as before except the increase of the number of priors (2704 vs. 1420) as we have larger feature maps. From Table 1, we can see that using larger input size has a much bigger improvement than using lower feature maps with the same input size. For example, it results in a 5.7 AP increase for person and 6.1 for car, making both of them have comparable performance to the baseline. We hypothesize that the bigger model can detect small objects much better because it sees more details and thus has less confusion between categories. AP also increases by 3.3 for dog, resulting in even better performance than the two-stage method (92.0 vs. 90.4). As a result, the mAP increases from 31.0 to 39.6; and the computation complexity only doubles. As far as we know, SSD is the first and best "single stage" method on the ILSVRC 2014 DET `val2` dataset.

Although there is still a gap between SSD and the two-stage method, we could expect to bridge the gap further by using the lower feature maps from the 443 × 443 model or using an even larger input image size (e.g. 600 × 600 as used in [5, 16]). These are left for future work.

4.2. VOC 2007 results

On this dataset, we mainly compare with other top "single stage" methods. In particular, we use VOC2007 `trainval` and VOC2012 `trainval` for training, and test on VOC2007 `test`. Notice that both Fast R-CNN [5] and Faster R-CNN [16] are fine-tuned from VGG [19] and use input images whose minimum dimension is 600. Our SSD is fine-tuned from a particular version of Inception, and uses either a 299 × 299 or 443 × 443 input image size.

Table 2 shows that Fast R-CNN and Faster R-CNN have slightly better performance than SSD, possibly because they have bigger input image size. However, Fast R-CNN is much slower because it still requires the proposal step. Faster R-CNN is hard to train because it has to alternatively fine-tune the shared convolutional layers and the extra prediction layers for RPN and Fast R-CNN. Our SSD is easy and straightforward to train, and completely discards the separate object proposal step. Table 3 shows comparisons on VOC2012 `test` set. We use the same models as Table 2. Fast R-CNN and Faster R-CNN have even better performance because they use extra 4952 images from VOC2007 `test` for training. However, the gap is smaller if Fast R-CNN is only trained on VOC2012 `trainval`. Compared to YOLO, SSD 299 × 299 already has compa-

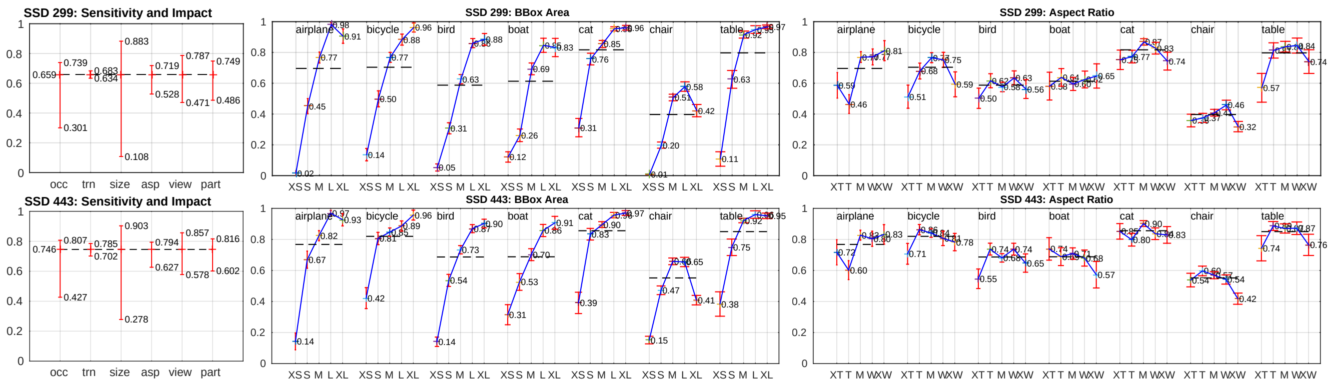


Figure 2. **Sensitivity and impact of different object characteristics on VOC2007 `test` set.** Each plot shows the normalized AP [9] with standard error bars (red). Black dashed lines indicate overall normalized AP. The most left plot summarizes the sensitivity for each characteristic over all categories. The rest two plots show detailed effects per categories. Key: BBox Area: XS=extra-small; S=small; M=medium; L=large; XL =extra-large. Aspect Ratio: XT=extra-tall/narrow; T=tall; M=medium; W=wide; XW =extra-wide.

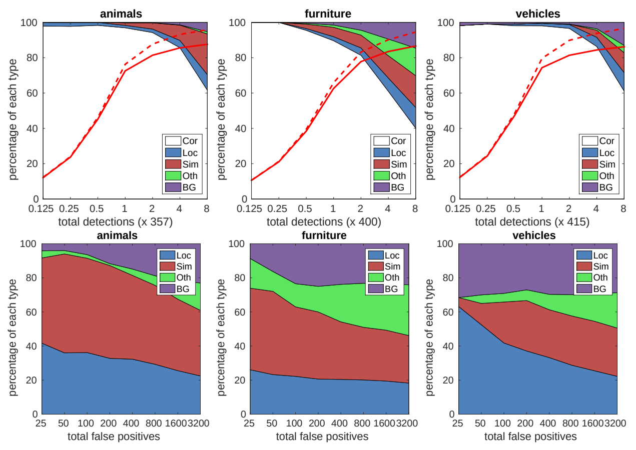


Figure 3. **Visualization of performance for SSD 443 on animal, furniture, and vehicle from VOC2007 `test`.** Top row shows the cumulative fraction of detections that are correct (Cor) or false positive due to poor localization (Loc), confusion with similar categories (Sim), with others (Oth), or with background (BG). Solid (red) line reflects the change of recall with "strong" criteria (0.5 jaccard overlap) as the number of detections increases. Dashed (red) line is using the "weak" criteria (0.1 jaccard overlap). Bottom row shows the distribution of top-ranked false positive types.

rable performance (54.4 vs. 57.9). When using the same input size, our SSD is much better than YOLO (63.3 vs. 57.9). It is because of the use of the convolutional priors from multiple feature maps and matching strategy during training, making SSD more flexible and better than YOLO.

In addition, we used the detection analysis tool from [9] to understand what characteristics affects SSD the most. Figure 3 shows that SSD can detect various object categories with high quality (large white area). It is very confident with most of the top detections. The recall is around 85-90%, and is much higher with "weak" (0.1 jaccard overlap) criteria. Compared to R-CNN [6], SSD has less localization error, indicating that SSD can localize objects better because it directly learns to regress priors to the objects instead of using two decoupled steps. However, SSD has more confusions with similar object categories, partly because we share locations for multiple categories and the input size is too small to distinguish the difference.

Figure 2 shows that SSD is very sensitive to the bounding box size. In other words, it has much worse performance on smaller objects than bigger objects. For example, it almost has 0 AP for extra-small (XS) objects if the input size is 299×299. This is not surprising because those XS objects may not even have any information at the topmost layer (8×8). Increasing the input size (e.g. from 299×299 to 443×443) can help improve detecting XS objects, but there is still a lot of room to improve. On the positive side, we can clearly see that SSD performs really well on large objects. And it is very robust to different aspect ratios and viewpoints of objects because we have imposed priors of various shapes per feature map location.

In the future, we would like to try bigger input size (e.g. 600×600) and/or using lower feature maps as was done in SSD Multi on ILSVRC. We expect this will further help remedy the confusion with similar categories problem and improve performance on small objects.

4.3. Training time

It takes a while for SSD to learn to regress priors to ground truth and predict confidences for all categories at the same time. For example, starting from a pre-trained network and using a batch size of 32 images, the network takes 3M steps to converge if trained on ILSVRC DET `train` dataset, and 500K steps on PASCAL VOC2012 + VOC2007 `trainval`. One reason is because there are many detections from all locations of multiple feature maps for multiple categories. We can reduce the training time by only updating all positives plus a small set of negatives (hard negatives) as described in Sec. 3.4. By doing this, perfor-

Source	Match method	Target	mAP	# matches
priors	Per-prediction	0/1	**39.0**	160
priors	Per-prediction	Jac	34.7	160
priors	Bipartite	0/1	31.7	28
priors	Bipartite	Jac	27.8	28
predictions	Per-prediction	0/1	20.8	25000
predictions	Per-prediction	Jac	19.5	26000
predictions	Bipartite	0/1	28.2	28
predictions	Bipartite	Jac	29.3	28

Table 4. **Design choices comparison on 20 PASCAL classes from ILSVRC2014 DET**. Using priors to do per-prediction matching and regressing to 0/1 target performed the best.

mance grows much faster in the beginning but still needs a similar number of steps to converge. Besides, compared to [6, 16], who regress selective search or RPN proposal boxes to ground truth, SSD requires more training steps because SSD priors have much less spatial information for the underlying objects and are thus more difficult to learn.

4.4. Design choices analysis

To understand SSD better, we have also carried out several controlled experiments to examine how each component affects the final performance. For all of the following experiments, we use the same underlying network and input size, and set batch size to 32. These experiments were performed on a subset of ILSVRC 2014 DET dataset, which has the same or equivalent 20 classes as PASCAL.

4.4.1 Matching source: Priors or Predictions?

During training time, a key step is to identify positives and negatives by matching source boxes with ground truth boxes. The source boxes can be either priors or actual predictions. If we use priors, which serve as ”anchors”, to do matching, we force the network to regress these priors to the ground truth and avoid letting the predictions free-float. However, if we use the prediction results to do matching, the network will be biased towards its own outputs. From Table 4, we can clearly see that using priors to do matching is much better than using predictions. Especially for per-prediction matching (as we will described in details later), using priors for matching doubles the average precision.

4.4.2 Matching method: Bipartite or Per-prediction?

As we have introduced in 3.1.1, there are two matching strategies: bipartite and per-prediction. Using different matching strategies will affect the number of positives and negatives during training time. In Table 4, we show the (average) number of matches when the models start saturating during training time. Because bipartite matching does a greedy one-to-one matching with ground truth boxes, it has an average of 28 matches for batch size of 32, which means that there are around 28 ground truth boxes on average in a batch. From Table 4, we can see that using the per-prediction matching strategy results in more positives. For example, if we use priors to do per-prediction matching, it has much more positives than bipartite (160 vs. 28), and also has better performance. But do we always get better performance if we have more positives? Not really. For example, if we use predictions to do per-prediction matching it has about 25000 matches, which means almost all predictions are matched to ground truth boxes – a clear overfit and bias toward its own outputs. Indeed, performance is much worse than using bipartite matching.

4.4.3 Target type: 0/1 or Jac

Since we used the multi-class logistic loss during the backpropagation step, we could either regress each category using 0/1 or Jac (jaccard overlap) in the objective function. From Table 4, it looks like regressing with 0/1 is always preferred, especially if we use priors for matching.

5. Conclusions

This paper introduces SSD, a unified and fast single shot object detector for multiple categories. We have shown that SSD has comparable results with many state-of-the-art methods on both ILSVRC DET and PASCAL VOC, and also have conducted many experiments to understand SSD in detail. We want to emphasize that SSD has demonstrated that sliding window methods can also achieve a similar level of performance, if designed carefully, compared to the methods with sparse object proposals [6, 2, 5, 16]. The use of many convolutional priors and tiling the priors to feature maps of multiple scales that we introduce both improve detection performance. It seems like using more ”windows” does not always lead to worse results as argued in [5], and actually can improve performance from our experiments. In the future, we would like to explore training with higher resolution input images and also use lower feature maps for predictions to further improve detection performance.

Additionally, we also want to explore new techniques for decreasing the training time. Currently SSD uses a single network to learn to predict both the offsets and confidences for multiple categories, which might be too hard to learn. In the future, we would like to use two separate towers for localization and classification, so that each tower can be more light weight for the specific subtask and thus decouple the complexity of the problem and boost the training speed.

6. Acknowledgment

We would like to thank Alex Toshev for helpful discussions. We are indebted to the Image Understanding and DistBelief teams at Google for their support. We also thank Philip Ammirato and Patrick Poirson for their helpful comments.

References

[1] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 2011. 5

[2] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *CVPR*, 2014. 1, 2, 3, 8

[3] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 2, 5

[4] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008. 2

[5] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. 1, 2, 6, 8

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 5, 7, 8

[7] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. *arXiv preprint arXiv:1411.5752*, 2014. 4, 6

[8] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*. 2014. 1, 2, 4

[9] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *ECCV 2012*. 2012. 7

[10] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013. 5

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 5

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[13] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. 4, 6

[14] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*, 2014. 4, 6

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640 v4*, 2015. 1, 2, 3, 6

[16] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 1, 2, 4, 6, 8

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F.-F. Li. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 2, 5

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 1, 2, 3, 4

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 6

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 1, 5

[21] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441 v3*, 2015. 2, 3, 5

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015. 5

[23] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013. 2

[24] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014. 4