Automatic Scale Selection and NN based Pruning for Improved and/or Fast Detections with SSD

Neeraj Matiyali[1]
neermat@iitk.ac.in
Manikandan R.[2]
manikandan@hitachi.co.in
Gaurav Sharma[1]
grv@cse.iitk.ac.in

[1] Computer Sc. and Engg. Dept.
IIT Kanpur
[2] R & D Center
Hitachi India Pvt. Ltd.

Abstract

Object detection is a fundamental problem in computer vision with wide ranging applications, from image tagging and indexing to applications related to autonomous vehicles, and robotics. We address this problem by proposing practical improvements to an existing state-of-the-art object detection method i.e. Single Shot Detector (SSD). As a first contribution, we propose to automatically select the scales of the default boxes. The scales of the default boxes determine the absolute size of the objects being searched for at different layers of the network, respectively, and hence are better off being tuned for the specific statistics of the object and the current data under inspection. As a second contribution, we propose to prune the search space using semantically-nearest neighbor images from the training set. We show the advantages of the two contributions with quantitative and qualitative empirical results; while, the first one leads to improved performance, the second one provides a way to trade-off performance for higher detection speeds.

1 Introduction

Visual recognition technologies such as image classification, i.e. predicting the objects, scene, attributes etc. present in an image, and object detection, i.e. predicting the exact location of a given object in an image with a bouding box, have graduated from proof of concept level to being productizable. Starting from the seminal work of Krizhevsky et al. [11], proposing the first high performance deep convolutional neural network (CNN) for image classification, the algorithms and systems have very far, with better performing networks for image classification, such as the GoogLeNet [17], VGGNet [15] and ResNet [10], as well as networks for object detection, such as Fast-RCNN [8], Single Shot Detector (SSD) [12] and You Only Look Once (YOLO) [13]. The high performing object detection modules now find applications in upcoming autonomous cars for detecting other cars, bicycles and pedestrians in the driving environment.

In the present paper, we propose to improve the implementation of the popular and high performing Single Shot Detector (SSD) by Liu et al. [12] with two contributions. As a first

contribution, we propose to automatize the selection of some key parameters, i.e. the scales of the *default boxes* that control what sized objects will be searched for at the different resolution CNN layers. As the second contribution, we aim to improve the detection speed of the object detector by retrieving semantically similar images from the training set and using the spatial distribution of the object in the retrieved images as a prior for the search space in the current image. We note that, while the first contribution, of automatically selecting the scales of the default boxes, is specific to the SSD architecture, the second contribution, of fast pruning of image areas based on nearest neighbors, is generic and could be used with any object detector. We validate the proposed methods on two challenging and publicly available datasets, i.e. KITTI Car and Pedestrian detection [7], and Caltech pedestrian detection dataset [5]. We show empirically that (i) automatic tuning of default boxes leads to improvements in performances over the default parameter settings of SSD and (ii) the NN based pruning allows us to obtain a trade-off between accuracy and speed of prediction.

1.1 Related works

Object detection has a long history in computer vision with the initial works based on bag-of-words features with nonlinear SVMs [18] and Histogram of Gradient (HOG) features [4, 6]. The current state-of-the-art object detection systems are now exclusively based on deep Convolutional Neural Networks (CNN). Some representative works include Region based CNN (RCNN) [9], and its fast [8] and faster [14] variants, Single Shot Detector (SSD) [12] and You Only Look Once [13]. While the intial detectors focused on using existing deep networks [11] to classify a limited number of object proposals per image, the later methods tried to do both, the object proposal prediction and classification, together in one architecture [8, 14]. In the current generation methods, the architectures do not have object proposals, but do prediction for a large number of possible object positions [12, 13]. These methods obtain the best performances on public benchmark datasets while being very fast as well. We use the current generation of methods in the current work and build on them.

2 Approach

We now give details of the two novel methods we propose on top of the base SSD method to improve its performance and time efficiency. We start with a brief description of the SSD detector to set some context and then go forward and present our contributions. We refer the reader to the full SSD paper [12] for details.

2.1 Outline of Single Shot Detector (SSD)

SSD [12] (shown in Fig. 1) predicts object bounding boxes by predicting the offsets and class scores relative to a set of predefined default boxes of different scales[1] and aspect ratios (as shown in Fig. 2). The scale of the default box varies across multiple convolution layers of the SSD network. When the feature layer used is the conv4_3 (first one to have a classifier layer in Fig. 1), the sizes of the default boxes, and hence the predicted objects, will be smaller[2].

[1] By scale we mean, unless otherwise specifically mentioned, the square root of the area of a box, i.e. the geometric mean of height and width

[2] This also depends on the default box scales, i.e. if the default box scale for the high resolution layer is kept very high then it could predict bigger objects too, but in practice that is usually not the case.

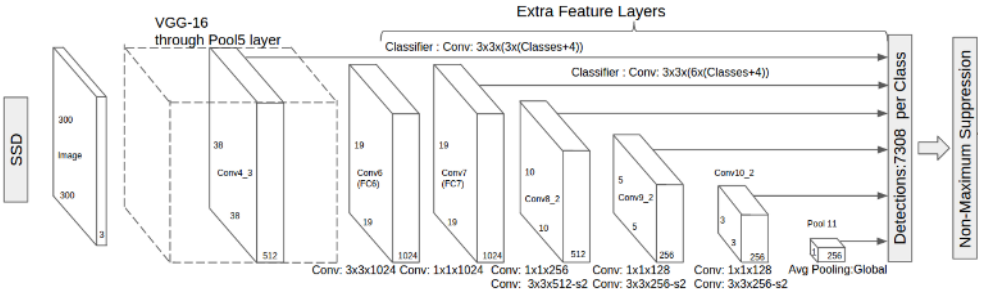
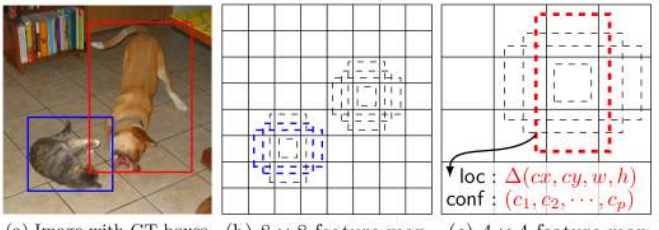


Figure 1: SSD network architecture (image from [12]). TODO Replace this with our own diagram



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Figure 2: Detection at multiple scales and aspect ratios using feature maps of different sizes and default boxes of different aspect ratios (image from [12]). TODO Replace this with our own diagram.

This is also demonstrated in Fig. 2, where it can be seen that the 8×8 feature map can (only) predict smaller objects cf. the 4×4 feature map. Thus, default boxes in the earlier layers have smaller scales than the ones further down in the network. Another related aspect is that the earlier layers will have larger number of default boxes; this can also be seen in 2, where the number of default boxes in 8×8 layer will be $8^2/4^2 = 4$ times more than that in the 4×4 layer, with other parameters same for both. Hence changing the distribution of the scales of default boxes changes the size of the objects, that we are effectively making SSD find, as well as the number of potential locations for the objects. While the default configuration of SSD default boxes performs reasonably well out of the box, it should be adjusted according to the specific statistics of the current task, i.e. as an extreme example, if the objects that we are interested in always appear in large sizes, then having small default boxes is not only inefficient but might introduce unwanted false positives as well.

Fig. 3 shows the histogram of scale of the bounding boxes in KITTI dataset [7]. We observe that even in a dataset with only two objects, i.e. cars and pedestrians, the distribution of the object scales vary – while there are many cars with scales more than 150 pixels, there are not many pedestrians at that scale. Thus in practice, for optimal performance, we should aim for a distribution of default boxes which is tuned wrt. the actual distribution in out target set of images. Such distribution could be potentially tuned to the object class, at the expense of linear scaling the complexity (as it will require as many different models as there are classes).

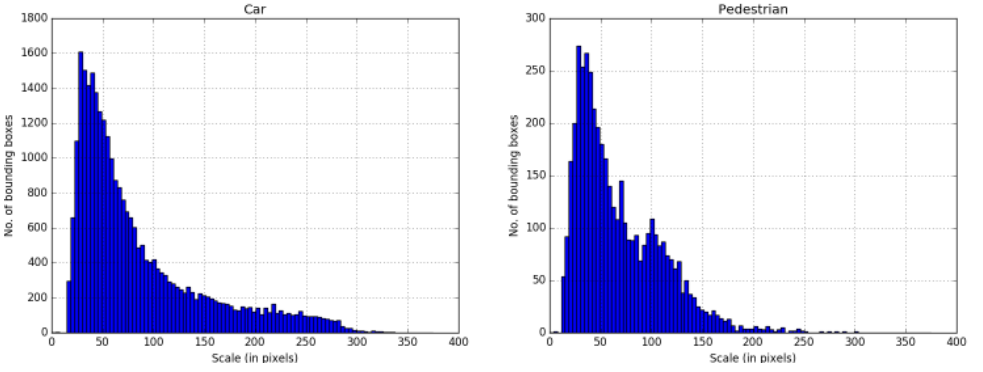


Figure 3: Histogram of scale of the bounding boxes in KITTI dataset, for car (left) and pedestrian (right) classes. Scale is the geometric mean of the width and the height.

Default box scale range (% of 275px) →			12-90	15-95	8-90	7,15-95
Layer	feature map resolution	#cells	Scale of default boxes (in px)			
`conv4_3`	114×35	3990	33.0	41.2	22.0	19.2
`fc7`	57×18	1026	68.7	77.9	59.6	41.2
`conv6_2`	29×9	261	104.5	114.6	97.2	85.2
`conv7_2`	15×5	75	140.2	151.2	134.7	129.2
`conv8_2`	8×3	24	176.0	187.9	172.3	173.2
`conv9_2`	4×2	8	211.6	224.6	209.9	217.2
`pool6`	1×1	1	247.5	261.2	247.5	261.2

Table 1: The distribution of (i) number of cells (third column) and (ii) the corresponding scale of default boxes (geometric mean of height and width, columns 4–7) for different parameter (scale range, top row) settings in SSD. The last two rows shows the performances on the moderate evaluation setting on KITTI `val`; it can be clearly seen that the change in default box scales has a large effect on performance.

Table 1 shows the empirical comparison of these different scale settings. First, the table shows the feature map resolutions of the different feature maps; the resolution, and hence number of cells, drops very quickly from the earlier layers to later, e.g. 114×35 with 3990 cells for conv4_3 to just 15×5 with 75 cells, three layers later at conv7_2. The default box scale range parametrization in SSD controls the sizes of the default boxes used at each such layer (which has a classifier connection).

2.2 Two Step Tuning of Default Box Scales

In an SSD network with L convolutional layers contributing to detection, let

$$\mathcal{S} := \{s_1, \ldots, s_L\} \tag{1}$$

denote the scale configuration of default boxes with $s_1 < s_2 < \ldots < s_L$. By assuming that the scale of default boxes increase linearly from one layer to next, scale configuration $\mathcal{S}$ can be

parameterized by $\{s_{\min}, s_{\max}\}$ as

$$\mathcal{S} = \left\{ s_i \middle| s_i = s_{\min} + \frac{i-1}{L-1}(s_{\max} - s_{\min}), i \in \{1, \ldots, L\} \right\} \tag{2}$$

thus reducing the number of parameters to be optimized to two. Given a training set of images $\mathcal{I}$ annotated with object bounding boxes, the optimal scale configuration potentially depends on the weighted combination of three factors, and could be found by solving the following optimization problem.

$$\mathcal{S}^{\star} = \underset{\mathcal{S}}{\operatorname{argmax}} \{\lambda_1 \text{Recall}_{\theta}(\mathcal{G}_{\mathcal{I}}, \mathcal{D}_{\mathcal{S}}) + \lambda_2 \Omega(\mathcal{R}, \mathcal{D}_{\mathcal{S}})\} + \lambda_3 \text{Perf}(\mathcal{S}) \tag{3}$$

where $\mathcal{G}_{\mathcal{I}}$ is the set of all ground-truth bounding boxes in $\mathcal{I}$, $\mathcal{D}_S$ is the set of default boxes from all L layers of the network with scale configuration $\mathcal{S}$ and $\mathcal{R}$ is the vector containing the receptive fields of the different SSD layers which have classifier connections. The λs are hyperparameters and fix the trade-off between the criteria. $\text{Recall}_{\theta}(\mathcal{G}_I, \mathcal{D}_{\mathcal{S}})$, is defined as the fraction of boxes in $\mathcal{G}_{\mathcal{I}}$ with intersection-over-union (IoU) overlap greater than threshold θ with at least one of the boxes in $\mathcal{D}_{\mathcal{S}}$. $\text{Perf}(\mathcal{S})$ is the average-precision performance of the network trained with the default box configuration $\mathcal{S}$ on a validation set. $\Omega(\cdot)$ measures the discrepancy between the receptive field of a layer and the size of the default box in that layer (for layers with classifier connections only); it is given by

$$\Omega(\mathcal{R}, \mathcal{D}_{\mathcal{S}}) = \sum_i \left(\frac{r_i}{d_i} - \alpha_i \right)^2, \tag{4}$$

where r_i is the receptive field, d_i is the default box size and α_i are fixed ratios for the layer i. This is to ensure that a layer with receptive field size (of 3×3 cells) of, say, 60 pixels should not be used to detect a much larger object i.e. should not have a default box size of, say, 150 pixels. The notion of larger is encoded in the choice of α_i, which are empirically arrived at.

With a representative validation set, we would expect only the $\text{Perf}(\cdot)$ to be important, akin to cross-validation, but doing that is costly as it requires separate training of the network for each element of $\mathcal{S}$. Here, we thus use a two step strategy – we use the $\text{Recall}(\cdot)$ and $\Omega(\cdot)$ (receptive field vs. default box size) to do an aggressive pruning of the scale ranges and then finally select only a small number of scale ranges to do the validation experiments requiring a full re-training of the network. The first stage thus requires calculating the different default boxes generated as a result of the different scale ranges, which is substantially cheaper to compute cf. full training for validation.

2.3 Pruning based on semantic retrieval

As a second contribution, we aimed at improving the test time of a trained model by actively pruning the search area. The pruning stategy was based on retrieving images with similar semantic appearances and using their bounding boxes to prune the search area in the current image. To do this effectively, we proposed to use a small and fast CNN network to encode a set of dataset images offline. At test time, two operations need to be done, (i) the test image has to be encoded with the same fast CNN and (ii) the nearest neighbors are then retrieved in that embedding space. The bounding boxes from the retrieved images are then overlayed on the current image and are used to prune the search area by running detection only on the

covering rectangle, and not the entire image. By doing so, we hoped to decrease the run time of the detection, while possibly trading off some performance.

However, we found that a generic similarity (based on an ImageNet pre-trained CNN) does not work effectively as it is overly biased by non-relevant artifacts such as shadows and backgrounds like buildings, roads and open markets. In particular with VGG Fast CNN[3] [3], pre-trained on the ImageNet dataset and compared using cosine similarity between ℓ_2 normalized last fc activations, we noticed that road and shadow regions in the images were affecting the retrievals significantly (Fig. 4), while what we would have liked to have were images which have similar object distributions.

To avoid such interference and obtain retrievals which reflect the semantics based on likely objects locations in the images, we then proposed to train Siamese network to learn the embedding. These embeddings would then be used instead of the embeddings obtained from ImageNet pretrained CNN. This can be seen as learning conditional priors, on object locations, from the training data. For each test image, similar images retrieved from the training data based on such embeddings would be expected to have similar object locations. The embeddings for the training images are computed offline and stored and only needs to be calculated for the test image at run time. The top retrieved images are then used to infer the possible object locations in the current image.

Oracle case. As a sanity check, before proceeding further, we did an oracle based experiment. We took the actual ground truth object boxes, instead of those from the nearest neighbors, and ran the pruning algorithm followed by detection. We found that the forward pass time, on an average, reduced from 40ms to 16ms while the performances on moderate set dropped from 82 to 72 for cars and 62 to 60 for pedestrians. Given this initial encouraging result, we proceeded to doing a fully automatic case where the bounding boxes of the nearest neighbors were used for pruning.

Fully automatic case. In this case, we assume the realistic setting and obtain pruning from the nearest neighbors (NN) from the train set. The main idea here is to train a Siamese network to have similarities in terms of similar object locations and not raw appearances of the frames. To train the Siamese network, we use a pairwise loss formulation similar to Bhattarai et al. [1]. The method requires two sets of pairs of images, one where the pair $(\mathbf{x}_i, \mathbf{x}_j)$ are similar, and have corresponding $y_{ij} = +1$, and the other where the pairs are dissimilar with $y_{ij} = -1$. The similarity and dissimilarity in our case is defined by the bounding boxes (of actual objects, of a certain class, present in the image) overlap – the images with high bounding box overlap are considered similar, while those for which bounding boxes do not overlap sufficiently are dissimilar. Once such set is constructed, $\mathcal{X} = \{(\mathbf{x}_i, \mathbf{x}_j, y_{ij})\}$, a projection $L \in \mathcal{R}^{D \times d}$ is learnt, where D is the dimension of the original image features and $d < D$ is the projection dimension. In our case the image features $\mathbf{x}_i$ are the fast CNN's last fc activations, and the projection is learnt on top of those. Thus, it is equivalent to learning a Siamese network, where we finetune only the top layer and do not finetune the CNN layers. The objective function that is optimized is given by,

$$\min_{L,b} \sum_{\mathcal{X}} \max(0, m - y_{ij}(b - d^2(\mathbf{x}_i, \mathbf{x}_j)) \tag{5}$$

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = \|L\mathbf{x}_i - L\mathbf{x}_j\|^2 \tag{6}$$

[3] https://gist.github.com/ksimonyan/97684708258292576a1

No. of NNs, k	2	4	8	16	No pruning
CNN forward pass time (ms)	0.93	0.93	0.93	0.93	
Average prior retrieval time (ms)	0.65	0.67	0.77	0.9	
Average SSD forward pass time (ms)	20.8	25.0	28.7	31.6	39.6
Total time (ms)	22.4	26.6	30.4	33.4	39.6
Speedup	1.8×	1.5×	1.3×	1.2×	
Average area pruned (%)	63.6	51.8	41.6	33.9	
Speedup expected based on AP	2.7×	2.1×	1.7×	1.5×	
AP for Car (Moderate)	55.8	66.3	72.7	75.9	82.2

Table 2: Speedup achieved by retrieval based pruning with 910×275 model.

where, $b \in \mathcal{R}^+$ is the bias, i.e. the threshold about which the similarity and dissimilarity is determined, and $m \in \mathcal{R}^+$ is a fixed margin, in our case $m = 1$ (cf. SVM).

The objective is minimized using stochastic gradient descent [1] for L and b. Once we have the L, we project all the training images using L and keep them in memory. When a new test image comes, we project it's feature using L and do NN with the training images. We retrieve k NNs, using Eclidean distance after projection, and use their ground truth bounding boxes to have a pruned search area as the minimum enclosing rectangle for all those bounding boxes, with an extra padding of 10 pixels for some context.

2.4 Comparison with state-of-the-art on test server

We also ran a limited set of experiments on the official test server of the KITTI dataset, to be able to compare with published results. Table 3 shows the results, along with those of a few recently published results. Since the number of runs on the test servers are controlled, we only made two of those so far, for the first one we tried our initial low resolution model trained on images of size 546×165 without the default box tuning, and for the second we tried 910×275 model with bounding box pruning. We see that the performances of the second model is much higher than that of the first model, e.g. 65.0 vs. 79.4 for car moderate set, which can be attributed to (i) higher scale of the model and (ii) tuning of default boxes. We also see that the performance of the higher resolution model is comparable to the published results for the car category, e.g. 79.4 (ours) vs. 76.7 of Stewart et al. (CVPR 2016) [16] and 83.5 of Yang et al. (CVPR 2016) [19], on moderate set. In the case of pedestrian detection, we are lacking behind slightly more, e.g. 54.0 (ours) vs. 58.7 of Cai et al. (ICCV 2015) [2] and 64.2 of Yang et al. (CVPR 2016) [19]. However, we have recently also started traning and tuning the higher resolutions model of size 1242×375 and we see that higher resolution is critical for high pedestrian detection, probably because many of the predestrians have very small widths and in lower resolutions there are not enough features to support small size (widths) bounding boxes. With the higher resolution model we have reached ~ 65 on moderate set of the `val`, but have not yet run it on the test server as we still need to tune the default boxes properly for the network.

References

[1] Binod Bhattarai, Gaurav Sharma, and Frederic Jurie. CP-mtML: Coupled projection multi-task metric learning for large scale face retrieval. In *CVPR*, 2016.

(a) (b)

Figure 4: Retrieval examples with fast CNN features only. In each column there is a `val` image at the top followed by 3 most similar `train_new` images. We observe that the retrievals are dominated by non-relevant aspects such as shadows, foliage etc.

(a) (b)

Figure 5: Retrieval examples with fast CNN features projected by matrix L. In each column there is a `val` image at the top followed by 3 examples among 5 nearest `train_new` images.

Method	Car			Pedestrian		
	E	M	H	E	M	H
Base SSD 546 × 165	78.2	65.0	56.3	50.0	37.9	36.1
Ours 910 × 275 w/ DBT	89.6	79.4	70.0	67.8	54.0	50.2
Stewart et al. (CVPR 2016) [16]	88.1	76.7	66.2	-	-	-
Cai et al. (ICCV 2015) [2]	-	-	-	70.7	58.7	52.7
Yang et al. (CVPR 2016) [19]	90.3	83.5	71.1	77.7	64.2	59.3
existing best*	96.9	92.6	86.5	86.6	77.0	72.4

Table 3: Comparison of our models (highlighted) on the official test server of the KITTI dataset with some published results. (*existing best method on the test server is probably a highly engineered combination of many methods and it given here for reference, DBT = default box tuning).

[2] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *CVPR*, 2015.

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference (BMVC)*, 2014.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[5] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 34, 2012.

[6] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] Ross Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*, 2015.

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[16] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *CVPR*, 2016.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[18] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *CVPR*, 2009.

[19] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016.