LOCALIZING OBJECTS FAST AND ACCURATELY

Wei Liu

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2016

Approved by:

Alexander C. Berg

Tamara L. Berg

Dragomir Anguelov

Jan-Michael Frahm

Marc Niethammer

ABSTRACT

WEI LIU: LOCALIZING OBJECTS FAST AND ACCURATELY.
(Under the direction of Alexander C. Berg.)

A fundamental problem in computer vision is knowing what is in the image and where it is. We develop models to localize objects of multiple categories, such as person and car, fast and accurately. In particular, we focus on designing deep convolutional neural networks (CNNs) for object detection and semantic segmentation. A central theme of this dissertation is to explore the design choices of network structure to combine the full power of CNNs and the characteristics of each task to not only achieve high-quality results but also keep the model relatively simple and fast.

At the heart of object detection is the question of how to search efficiently through a continuous 2D bounding boxes space of various scales and aspect ratios at every possible location in an image. A brute force approach would be searching over all possibilities, but it is apparently not scalable and is quite difficult. An alternative is to propose some potential locations which might contain objects, and then classify each of the proposal. Because the search space is much smaller after the proposal step, we can use a more powerful feature to describe each proposal. A first contribution of this dissertation is to show that fine-tuning a much deeper network can boost the detection performance significantly, compared to a relatively shallower network.

A second contribution of this dissertation is that we show that the search can be approximated by discretizing the search space and then regressing the residual difference

between a discrete box and a target box. This is a departure from the proposal and then classify series of methods. We present a single stage framework, SSD, which can simultaneously detect and classify objects fast and accurately. SSD splits the space of small boxes more densely and the space of larger boxes more sparsely. As a result, it can discretize the space more efficiently and ease training notably. We have empirically shown that it is as accurate as or even better than the two-stage methods and yet is much faster.

Unlike object detection, semantic segmentation is usually treated as a per-pixel classification problem, especially in the era of deep networks. However, a major issue is how to incorporate global semantic context information when making local decision. Although there are concurrent works on using techniques from graphical models such as conditional random fields (CRFs) to introduce context and structure information, we present a simple yet effective method, ParseNet, by using the average feature for a layer to augment the features at each location. Experimental results show that it can be as effective as a method which uses CRFs as a post-processing step to include context information.

In order to make the above methods useful for many real-time systems, such as mobile devices or self-driving cars, we have collected large-scale video datasets for multiple categories, and hope that temporal consistency information in video can help further boost the performance and speed up the operations while lowering power consumption.

To my family.

ACKNOWLEDGMENTS

There are so many people I need to thank in my PhD journey, and I am very grateful.

I would like to thank my adviser, Alex Berg, from the deepest of my heart. His insight, wisdom and encouragement has always been an inspiration to me. I am very grateful for the freedom and guidance he gave and the trust for letting me organize the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) for two years.

Thanks to my committee members: Tamara Berg, Dragomir Anguelov, Jan-Michael Frahm and Marc Niethammer for giving many suggestions for the final thesis. Thanks to the Berg group: Kota Yamaguchi, Vicente Ordonez, Xufeng Han, Hadi Kiapour, Sirion Vittayakorn, Eunbyung Park, Cheng-Yang Fu, Licheng Yu, Yipin Zhou, Phil Ammirato, Patrick Poirson for so many wonderful memories, cheers, discussions, and working together towards many conference deadlines. Thanks to the UNC and Stony Brook Computer vision group. Thanks to ILSVRC team: Olga Russakovsky, Jia Deng, Fei-fei Li, and Alex Berg for making ILSVRC 2015-2016 a success.

Thanks to Total Recall team at Google. Thanks to Andrew Rabinovich for putting me in the GoogLeNet team, which turned out to be a great success and a turning point for my PhD journey. Thanks to Christian Szegedy for his patience for helping me understand Inception and use it well for detection. Thanks to Dumitru Erhan for introducing MultiBox and help make it single shot. Thanks to Dragomir Anguelov for being a men-

tor, a friend, and a great helper to make single shot detector great and having insightful brainstorming. Thanks to other folks at Google: Yangqing Jia, Pierre Sermanet, Scott Reed, Alex Tosev, George Toderici, Vincent Vanhoucke for helpful discussions. Thanks to Image Understanding and DistBelief team at Google as well.

Thanks to Fernando de la Torre and Alexander Hauptmann for introducing me to computer vision/multimedia research. Thanks to Yuan Shi for being a roommate and encouragement in the first year at CMU. Thanks to Svetlana Lazebnik for her supervision for my first year PhD and initiating the research topic on object detection from video. Thanks to Hongtao Huang for many cheers. Thanks to Lukas Marti for letting me play with computer vision on iPhone. Thanks to the Apple interns that made 2012 summer a great fun and a smooth transition from UNC to Stony Brook. Thanks to so many friends who came across during the journey. Thanks to NVIDIA for donating many GPUs and letting me use their GPU cluster. Thanks to Google Fellowship for supporting my last year of PhD. Thanks to NSF 1452851, 1446631, 1526367, 1533771 for support.

Finally thanks to my parents, Changbai Liu and Shangen Yang, for helping annotate many videos and for everything. Thanks to my wife Yi Zhao for everything always. Thanks to my daughter Angela Liu for giving me tremendous joy everyday. Because of your love and support, I can navigate through the journey and enjoy every moment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: Introduction

> What does it mean, to see? The plain man's answer (and Aristotle's, too). would be, to know what is where by looking.
>
> – David Marr, *Vision*

1.1 Motivation

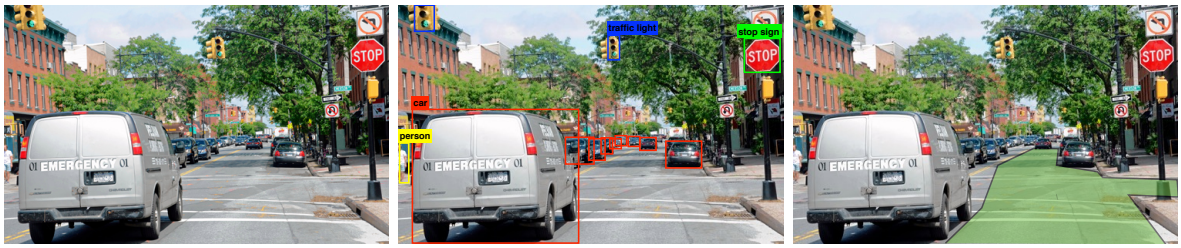


Figure 1.1: This dissertation studies the problem of *object detection* and *semantic segmentation*. Object detection is to know the location (e.g. bounding box) of each object; and semantic segmentation is to know what each pixel is (e.g. drivable path) in an image.

Suppose the left image in Figure 1.1 is what a self-driving car sees. In order to make a decision on what to do next, it needs to recognize there is a car in front on the left side so that it has to slow down to not hit the car; it also needs to recognize there is a stop sign on the right to notify itself to stop when reaching at the stop sign location. When the traffic light turns green, it not only needs to respond to it on time but also has to have a sense where is the drivable path so that it won't bump into curbs or other cars. These are only a few basic functionalities we want a self driving car to perceive the world, and are what this dissertation is trying to solve.

We develop models to localize objects of multiple categories, such as person and car, fast and accurately. In particular, we focus on designing deep convolutional neural networks (CNNs) for object detection and semantic segmentation. A central theme of this dissertation is to explore the design choices of network structure to combine the full power of CNNs and the characteristics of each task to not only achieve high-quality results but also keep the model relatively simple and fast.

At the heart of object detection is the question of how to search efficiently through a continuous 2D bounding boxes space of various scales and aspect ratios at every possible location in an image. A brute force approach would be searching over all possibilities, but it is apparently not scalable and is quite difficult. An alternative is to propose some potential locations which might contain objects, and then classify each of the proposal. Because the search space is much smaller after the proposal step, we can use a more powerful feature to describe each proposal. A first contribution of this dissertation is to show that fine-tuning a much deeper network can boost the detection performance significantly, compared to a relatively shallower network.

A second contribution of this dissertation is that we show that the search can be approximated by discretizing the search space and then regressing the residual difference between a discrete box and a target box. This is a departure from the proposal and then classify series of methods. We present a single stage framework, SSD, which can simultaneously detect and classify objects fast and accurately. SSD splits the space of small boxes more densely and the space of larger boxes more sparsely. As a result, it can discretize the space more efficiently and ease training notably. We have empirically shown that it is as accurate as or even better than the two-stage methods and yet is

much faster.

Unlike object detection, semantic segmentation is usually treated as a per-pixel classification problem, especially in the era of deep networks. However, a major issue is how to incorporate global semantic context information when making local decision. Although there are concurrent works on using techniques from graphical models such as conditional random fields (CRFs) to introduce context and structure information, we present a simple yet effective method, ParseNet, by using the average feature for a layer to augment the features at each location. Experimental results show that it can be as effective as a method which uses CRFs as a post-processing step to include context information.

In order to make the above methods useful for many real-time systems, such as mobile devices or self-driving cars, we have collected large-scale video datasets for multiple categories, and hope that temporal consistency information in video can help further boost the performance and speed up the operations while lowering power consumption.

1.2 Thesis statement

Carefully designing and training deep neural networks from large-scale datasets enables detecting and segmenting objects of multiple categories fast and accurately.

1.3 Outline

Deep convolutional neural networks (LeCun et al., 1998; Krizhevsky et al., 2012; Szegedy et al., 2014a; Simonyan and Zisserman, 2014; He et al., 2016) have demonstrated impressive levels of performance on the image classification task. In addition to knowing what objects are in the image, we are also interested in pinpointing where they are. This

is usually accomplished by modifying the structure of the image-level neural network and fine-tuning it for localization purposes.

Region-based Convolutional Neural Networks (R-CNN) (Girshick et al., 2014) approach object detection as a classification problem over object proposals, followed by regression of the bounding box coordinates. In Chapter 2 we show the process by which we fine-tuned GoogLeNet (Szegedy et al., 2014a) instead of AlexNet (Krizhevsky et al., 2012) using the R-CNN framework and achieved high quality detection results. Additionally, we augment the region proposals by combining the Selective Search boxes (Uijlings et al., 2013) with MultiBox predictions (Erhan et al., 2014) for higher recall. Finally, we use an ensemble of 6 GoogLeNet models when classifying each region which improves accuracy from 38.0% to 43.9%, which in turn enabled us to win the ILSVRC2014 DET challenge (Russakovsky et al., 2015) (a world renowned annual event for the visual recognition challenge). However, this approach requires classifying thousands of proposals per image, and is therefore very slow.

We are concerned with the efficiency of the model and the simplicity of the training methodology as much as we are about the quality. This not only enables faster turn around for development, but also is particularly useful when deploying the model on real time systems, such as mobile devices or self driving cars, which are sensitive to time delay and usually have limited computational power.

We have demonstrated that by careful design, it is possible to detect objects of multiple categories with a single evaluation of an input image and achieve state-of-the-art performance. In Chapter 3, 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 default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box 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. SSD is simple compared to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all 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 the PASCAL VOC, COCO, and ILSVRC datasets confirm that SSD has competitive accuracy to methods that utilize an additional object proposal step and is much faster, while providing a unified framework for both training and inference. Compared to other single stage methods, SSD has much better accuracy even with a smaller input image size.

For semantic segmentation, most successful techniques are based on Fully Convolutional Networks (FCN). It can produce semantic labeling per pixel in a single evaluation of an input image, but it disregards global semantic context. In order to integrate context, several approaches (Chen et al., 2014; Schwing and Urtasun, 2015; Zheng et al., 2015), propose using graphical models such as conditional random fields (CRFs) (Krähenbühl and Koltun, 2011), to introduce global context and structured information into a FCN. Although powerful, these architectures can be overly complex. At the least, this leads to time-consuming training and inference. In Chapter 4, we present a technique for adding global context to fully convolutional networks for semantic segmentation. The approach

is simple, using the average feature for a layer to augment the features at each location. In addition, we study several idiosyncrasies of training, significantly increasing the performance of baseline FCN network. When we add our proposed global feature, and a technique for learning normalization parameters, accuracy increases consistently even over our improved versions of the baselines. Our proposed approach, ParseNet, achieves state-of-the-art performance on SiftFlow and PASCAL-Context with small additional computational cost over baselines, and near state-of-the-art performance on PASCAL VOC 2012 semantic segmentation with a simple approach.

Our ultimate goal is to develop a system that can localize multiple objects in videos accurately in real time. To achieve this, we have collected a large-scale video dataset across many categories. A naive approach would be applying the techniques we developed from static images to each individual frame of a video, then tracking the most confident detections through time. However, this requires multiple decoupled components (e.g. detector and tracker) and is slow and not optimal. The key problem is how to detect objects accurately per frame and associate objects correctly through time. We believe that combining a recurrent neural network (RNN) and a convolutional neural network (i.e. SSD) can help solve this problem. We leave it for future work.

CHAPTER 2: Accurate Object Detection with GoogLeNet

We need to go deeper.

– Inception

2.1 Introduction

Object detection is a problem of knowing what is in an image and where it is. For example, given an image of a cat, it needs to know where the cat is in the image. It is usually treated as a classification problem over many locations of an image. Sliding window methods (Viola and Jones, 2001; Dalal and Triggs, 2005; Felzenszwalb et al., 2008) use an exhaustive search to find the best locations for objects in an image. The extremely large search space has become a bottleneck for such methods. In order to be computational efficient, these methods can only use "simple" features, such as HoG (Dalal and Triggs, 2005), to classify each location. On the other hand, region proposal methods (Uijlings et al., 2013; Girshick et al., 2014) have recently gained many attention. These methods first generate a pool of candidate regions, which may contain objects, and then classify each of the regions. The region proposals can be either generated using low-level information (Carreira and Sminchisescu, 2012; Endres and Hoiem, 2010; Arbelaez et al., 2011; Alexe et al., 2012; Uijlings et al., 2013; Zitnick and Dollár, 2014; Arbeláez et al., 2014) or learned from deep neural networks (Erhan et al., 2014). Because the search space is much smaller after the proposal step, one can use "rich" features to classify each proposal.

There are many research on how to construct better features (Csurka et al., 2004; Wang et al., 2010; Van De Sande et al., 2010; Jégou et al., 2010; Sánchez et al., 2013) from manually designed local descriptors (Lowe, 2004; Bay et al., 2006; Wang et al., 2009). Selective search (Uijlings et al., 2013) is the first work which shows that the region proposal method can outperform the state-of-the-art sliding window method (Felzenszwalb et al., 2008) on PASCAL VOC object detection dataset (Everingham et al., 2010). R-CNN (Girshick et al., 2014) then shows that using the features extracted from a fine-tuned convolutional neural network (i.e. AlexNet (Krizhevsky et al., 2012)) can outperform traditional features by a large margin. The key improvement is that the feature automatically learned from a large scale image dataset (Russakovsky et al., 2015) with CNNs is much better than the handcrafted features.

In this work, we show that we can get even better features from a much deeper network, which further boost the object detection performance. In specific, we build our accurate object detector using a much deeper and yet efficient neural network architecture – GoogLeNet (Szegedy et al., 2014a), and show that it significantly outperforms AlexNet (Krizhevsky et al., 2012) within the similar region proposal based framework. With significantly more layers, the network has much higher capacity to model the complexity of visual objects, which enables us to win the ILSVRC 2014 object classification and object detection task[1].

[1]http://image-net.org/challenges/LSVRC/2014/results

2.2 Related Work

Feature matters for almost all visual recognition tasks. There are two mainstream methods for solving object detection: sliding window and region proposal, both of which heavily rely on the power of the feature representation.

Sliding window methods usually requires searching over all possible locations and scales within an image to detect objects, which inevitably forces such methods to use simple features. (Viola and Jones, 2001) used a pool of simple feature detectors, such as rectangle detectors, to quickly reject negative regions in an image by using a boosting classifier. It also applied the cascade idea to further improve the speed of the detector. (Dalal and Triggs, 2005) introduced a simple-to-compute feature, Histogram of Gradient (HoG), and used a linear SVM classifier to detect person in an image. Deformable Part Model (DPM) (Felzenszwalb et al., 2008) extended it further by considering not only whole object but also object parts, and used a latent SVM method to automatically learn to detect both object and object parts and the spatial constraints between them.

On the other hand, there are other methods that do not share the exhaustive search idea but use the object proposal idea to reduce the number of regions to examine. Both (Carreira and Sminchisescu, 2012) and (Endres and Hoiem, 2010) proposed to generate a set of segmentation by using many different parameters from an accurate yet expensive contour detector (Arbelaez et al., 2011). Given such segmentation results, it then used the learned classifier to rank the segments. Selective search (Uijlings et al., 2013) suggested that rough object locations instead of precise object location is good enough for the task of object detection. It used a much more efficient segmentation

method (Felzenszwalb and Huttenlocher, 2004) to make selective search method computational feasible on large datasets. Besides, it also proposed the usage of hierarchical segmentation idea to speed up the computational time and maintain the performance. In the most recent works like MultiBox (Erhan et al., 2014; Szegedy et al., 2014b), the selective search region proposals, which are based on low-level image features, are replaced by proposals generated directly from a deep neural network.

For object proposal based methods, because there are much less regions to be considered, it is feasible, then, to try more powerful features to help recognize complicated and deformable objects. There are a large number of methods focusing on how to design better feature. Such methods first need to localize distinct regions (Lindeberg, 1998; Mikolajczyk and Schmid, 2004; Matas et al., 2004; Mikolajczyk et al., 2005; Rosten and Drummond, 2006) within an image and extract local image feature descriptors (Lowe, 1999; Berg and Malik, 2001; Belongie et al., 2002; Mikolajczyk and Schmid, 2005; Rublee et al., 2011; Leutenegger et al., 2011; Alahi et al., 2012)), construct visual codebooks by performing k-means clustering (Lloyd, 1982) on these local feature descriptors, and then encode the local features in many different ways (Csurka et al., 2004; Lazebnik et al., 2006; Zhang et al., 2007; Philbin et al., 2008; Van Gemert et al., 2008; Zhou et al., 2010; Wang et al., 2010; Perronnin et al., 2010; Jégou et al., 2010; Chatfield et al., 2011; Van de Sande et al., 2014).

Although powerful, these features are handcrafted and thus are sub-optimal and are not capable of describing the highly complicated visual world. From neuroscience we know that human visual cortex are hierarchical and have multi-stage processes for computing features with high level semantic meaning. Neocognitron (Fukushima, 1980) is

an early attempt to mimic such process with hand designed filters. Later (LeCun et al., 1998) used stochastic gradient descent (SGD) via backpropagation (Rumelhart et al., 1985) to train convolutional neural networks (CNNs) to learn the filters as well. The latest widely success of AlexNet on ILSVRC 2012 object classification task (Krizhevsky et al., 2012) has rekindled huge interests in CNNs. (Sharif Razavian et al., 2014) showed that a linear classifier with features extracted from OverFeat (Sermanet et al., 2013), a CNN pretrained on ILSVRC, outperforms all methods which uses traditional handcrafted features on various recognition tasks. Since feature representation of the input image is critical to many recognition tasks, deep network has a huge advantage over traditional computer vision methods because the feature representation learned from large-scale dataset is much better than a handcrafted feature.

The current state-of-the-art for object detection is R-CNN (Girshick et al., 2014), a method uses the object proposal framework. Such a two-stage approach leverages the accuracy of bounding box segmentation with low-level cues, as well as the highly powerful classification power of state-of-the-art CNNs. We adopted a similar pipeline, but have explored enhancements in both stages, such as MultiBox (Erhan et al., 2014) prediction for better region proposals, and ensemble approaches for better categorization of bounding box proposals.

2.3 Region proposal based object detection

In this work, we adopt the region proposal based object detection framework. Our object detection system needs to first generate category independent region proposals, as illustrated in Figure 2.1b. Then given a CNN pretrained on ILSVRC object classification

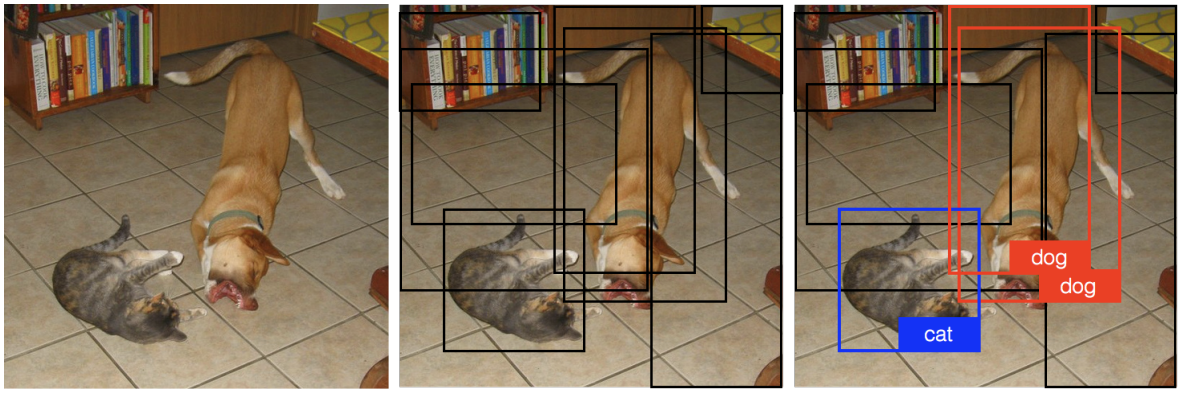
dataset, our system needs to fine-tune it on warped proposal windows to adapt it to the object detection dataset. Finally, our system needs to extract a fixed length feature vector from the fine-tuned CNN for each proposal and learn a set of class-specific linear SVMs to post classify each proposal, as shown in Figure 2.1c. In this section, we will describe each steps and the new network architecture in details.

2.3.1 Region proposals

Recently, there are many methods (Carreira and Sminchisescu, 2012; Endres and Hoiem, 2010; Arbelaez et al., 2011; Alexe et al., 2012; Uijlings et al., 2013; Zitnick and Dollár, 2014; Arbeláez et al., 2014) introduced for generating category independent (agnostic) region proposals. These methods use different low-level cues of image to generate a large pool of candidate regions of potential objects. In our system, we use the selective search (Uijlings et al., 2013) because it generates the best quality proposals. Besides, we also add about 200 proposals (Erhan et al., 2014) from MultiBox which are learned with a deep network. As a result, we can get very high recall with relatively small number (i.e. 1200) of proposals in each image.

2.3.2 Fine-tuning network

Since we need many data to train a CNN, we have to first use a large-scale dataset to pretrain a model so that it has a good representation of the visual world. Usually we use the ILSVRC classification dataset, which has 1000 classes with 1.2 million training images, to pretrain a model since image level labels are much easier to collect than more detailed annotations such as bounding boxes. Given a network that is first pretrained on



(a) Input image. (b) Region proposals. (c) Post classification.

Figure 2.1: **Region proposal based object detection.**

object classification datasets, we need to adopt it to the object detection datasets. As opposed to R-CNN (Girshick et al., 2014) which uses AlexNet (Krizhevsky et al., 2012), we use a much deeper and powerful network. We will describe the details of the network architecture later.

Given an object detection dataset where accurate bounding boxes for all objects in an image are provided, we first generate object proposals as described in Sec. 2.3.1. During training time, we treat all proposals with ≥ 0.5 IoU overlap with a ground truth box as a positive sample for that box's class and the rest negatives. At each SGD iteration, we sample 32 positive proposals and randomly select 96 negative proposals and crop/warp them to a fixed size (e.g. 227×227) to form a mini-batch to update the network's parameters to recognize proposals better.

Notice that without fine-tuning the network on the object proposals, the feature extracted directly from a network, which is pretrained on the ILSVRC classification dataset, does not give any improvement. However, after fine-tuning the network on the object proposals of the detection dataset, it shows dramatic improvement for the

detection performance.

2.3.3 Post classification

After the fine-tuning, we need to extract features for all proposals within an image. We follow the same procedure as proposed in R-CNN (Girshick et al., 2014). In specific, we crop and warp each proposal region to a fixed image size (e.g. 227×227). Then we forward the warped proposal to the fine-tuned network to extract the final feature layer (before the 1000-way classification layer) and save it as the feature representation for each proposal. Then we train a binary linear SVM classifier for each object category independently using the extracted features. We use the ground truth boxes as positive samples and use hard negative mining methods (Felzenszwalb et al., 2008) to select hard negative samples during training.

2.3.4 Network architecture

The major contribution of this work is that we use a much deeper and powerful network, which results in better feature and improve the object detection performance significantly. We now describe the motivation and details of the network.

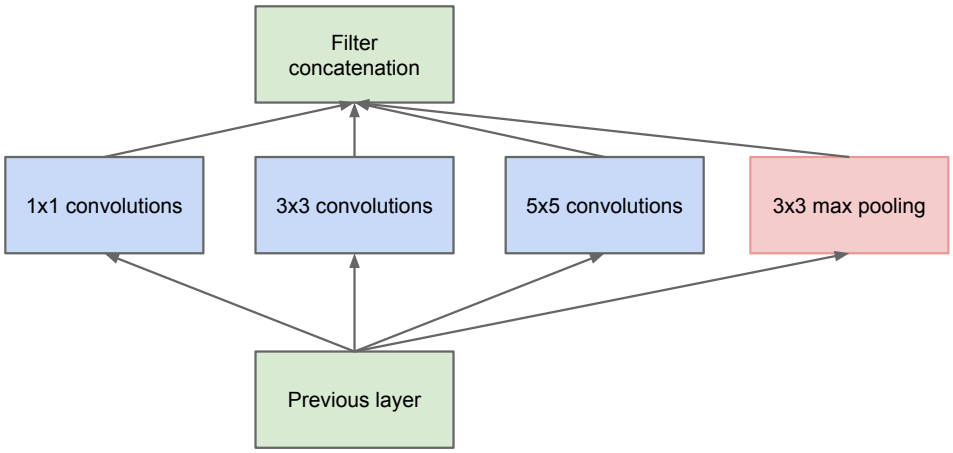
The most straightforward way of improving the performance of deep neural networks is by increasing their size. This includes both increasing the depth – the number of network levels – as well as its width: the number of units at each level. This is an easy and safe way of training higher quality models, especially given the availability of a large amount of labeled training data. However, this simple solution comes with two major drawbacks.

Bigger size typically means a larger number of parameters, which makes the enlarged network more prone to over-fitting. The other drawback of uniformly increased network size is the dramatically increased use of computational resources. For example, in a deep vision network, if two convolutional layers are chained, any uniform increase in the number of their filters results in a quadratic increase of computation.

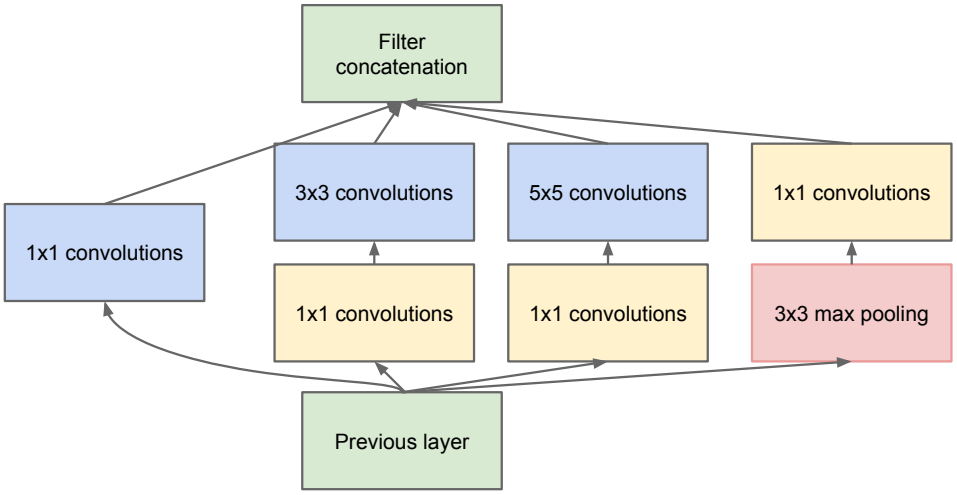
Inception module

A fundamental way of solving both of these issues would be to introduce sparsity and replace the fully connected layers by the sparse ones, even inside the convolutions. The well known Hebbian principle – neurons that fire together, wire together – suggests that we could use convolution kernels to learn to associate similar patterns. Besides, considering the natural of multiple scales of objects, we apply convolutional kernel of multiple sizes (e.g. 1, 3×3, and 5×5) on an input layer and concatenate the resulted feature maps, which becomes the input of the next stage. Additionally, since pooling operations have been essential for the success of current convolutional networks, it suggests that adding an alternative parallel pooling path in each such stage should have additional beneficial effect, too. Figure 2.2a shows an naive Inception module which embodies the idea.

One big problem with the above modules, at least in this naive form, is that even a modest number of 5×5 convolutions can be prohibitively expensive on top of a convolutional layer with a large number of filters. This problem becomes even more pronounced once pooling units are added to the mix: the number of output filters equals to the number of filters in the previous stage. The merging of output of the pooling layer with



(a) Naive Inception module.



(b) Inception module with dimensionality reduction.

Figure 2.2: **Inception module.**

outputs of the convolutional layers would lead to an inevitable increase in the number of outputs from stage to stage, leading to a computational blow up within a few stages.

This leads to the second idea of the Inception architecture: judiciously reducing dimension wherever the computational requirements would increase too much otherwise. In specific, 1×1 convolutions are used to compute reductions before the expensive 3×3 and 5×5 convolutions. The final result is depicted in Figure 2.2b. This allows for increasing both the width of each stage as well as the number of stages without getting into computational difficulties.

GoogLeNet

In general, an Inception network is a network consisting of modules of the above type stacked upon each other, with occasional max-pooling layers with stride 2 to halve the resolution of the grid. By the "GoogLeNet" name we refer to the particular incarnation of the Inception architecture used in our submission for the ILSVRC 2014 competition.

Table 2.1 illustrates the most common instance of Inception used in the competition. All the convolutions, including those inside the Inception modules, use rectified linear activation (Nair and Hinton, 2010). The size of the receptive field in our network is 224×224 in the RGB color space with zero mean. "$\#3 \times 3$ reduce" and "$\#5 \times 5$ reduce" stands for the number of 1×1 filters in the reduction layer used before the 3×3 and 5×5 convolutions. One can see the number of 1×1 filters in the projection layer after the built-in max-pooling in the "pool proj" column. All these reduction/projection layers use rectified linear activation as well.

The network was designed with computational efficiency and practicality in mind, so

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 2.1: **GoogLeNet incarnation of the Inception architecture.**

that inference can be run on individual devices including even those with limited computational resources, especially with low-memory footprint. Figure 2.3 shows the resulting network architecture. For more details, please refer to the original paper (Szegedy et al., 2014a).

2.4 Experiments

2.4.1 ILSVRC 2014 Classification Challenge Setup and Results

The ILSVRC 2014 classification challenge involves the task of classifying the image into one of 1000 leaf-node categories in the ImageNet hierarchy. There are about 1.2 million images for training, 50,000 for validation and 100,000 images for testing. Each image is associated with one ground truth category, and performance is measured based

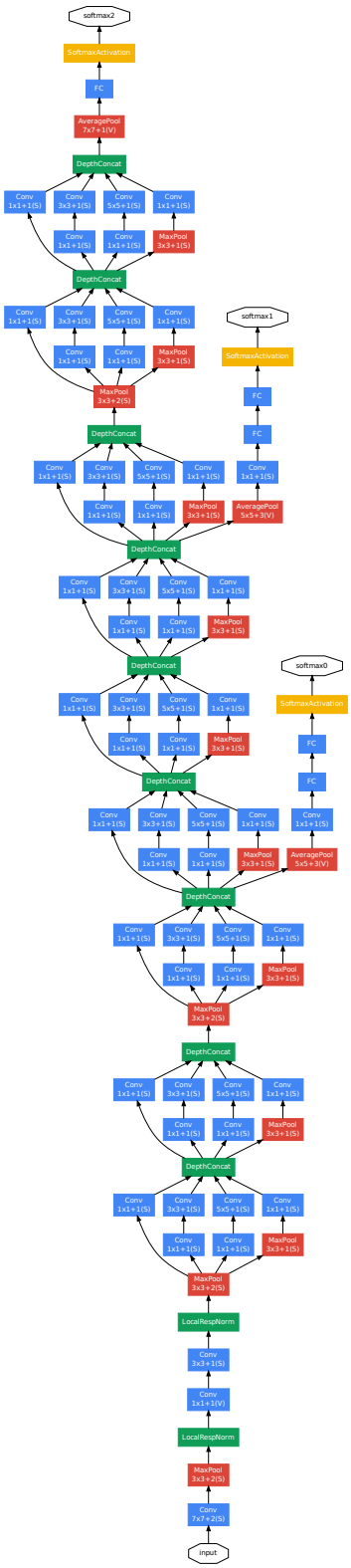


Figure 2.3: **GoogLeNet network structure.**

on the highest scoring classifier predictions. Two numbers are usually reported: the top-1 accuracy rate, which compares the ground truth against the first predicted class, and the top-5 error rate, which compares the ground truth against the first 5 predicted classes: an image is deemed correctly classified if the ground truth is among the top-5, regardless of its rank in them. The challenge uses the top-5 error rate for ranking purposes.

We participated in the challenge with no external data used for training. We adopted a set of techniques during testing to obtain a higher performance, which we describe next.

- We independently trained 7 versions of the same GoogLeNet model (including one wider version), and performed ensemble prediction with them. These models were trained with the same initialization (even with the same initial weights, due to an oversight) and learning rate policies. They differed only in sampling methodologies and the randomized input image order.

- During testing, we adopted a more aggressive cropping approach than that of AlexNet (Krizhevsky et al., 2012). Specifically, we resized the image to 4 scales where the shorter dimension (height or width) is 256, 288, 320 and 352 respectively, take the left, center and right square of these resized images (in the case of portrait images, we take the top, center and bottom squares). For each square, we then take the 4 corners and the center 224×224 crop as well as the square resized to 224×224, and their mirrored versions. This leads to $4 \times 3 \times 6 \times 2 = 144$ crops per image. A similar approach was used by Andrew Howard (Howard, 2013) in the previous year's entry, which we empirically verified to perform slightly worse

than the proposed scheme. We note that such aggressive cropping may not be necessary in real applications, as the benefit of more crops becomes marginal after a reasonable number of crops are present (as we will show later on).

- The softmax probabilities are averaged over multiple crops and over all the individual classifiers to obtain the final prediction. In our experiments we analyzed alternative approaches on the validation data, such as max pooling over crops and averaging over classifiers, but they lead to inferior performance than the simple averaging.

Our final submission to the challenge obtains a top-5 error of 6.67% on both the validation and testing data, ranking the first among other participants. This is a 56.5% relative reduction compared to the SuperVision approach in 2012, and about 40% relative reduction compared to the previous year's best approach (Clarifai) (Zeiler and Fergus, 2014), both of which used external data for training the classifiers. Table 2.2 shows the statistics of some of the top-performing approaches over the past 3 years.

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	**6.67%**	no

Table 2.2: **Results on ILSVRC2014 object classification *external* track.**

2.4.2 ILSVRC 2014 Detection Challenge Setup and Results

ILSVRC uses mean Average Precision (mAP) to measure the accuracy of object detection methods. A method produces arbitrary number of detection results for each object classes in each image. Each detection result has the format of (b_{ij}, s_{ij}) for image I_i and object class C_j, where b_{ij} is the bounding box and s_{ij} is the score. The detection results are first sorted in descending order based on detection scores, and are then greedily matched to the ground truth boxes. A detection result is considered as a true positive if the intersection over union (IoU) overlap with a ground truth box is more than 50%; otherwise it is considered as a false positive.

Note that it also penalizes duplicate detections. In other words, if there are multiple detections for an object (e.g. IoU threshold > 0.5), only the detection with highest score is a true positive and all others are false positives. Given this information, we can then compute precision as the fraction correct detections among all the detections reported, and recall as the fraction of detected ground truth objects, for each object class. We then compute average precision (AP) as the area under the precision/recall curve for each object class, and mAP as the average from all object classes.

The approach we take is similar to R-CNN by (Girshick et al., 2014), which first generates object proposals and then post-classify each proposal using deep convolutional network. Instead of using AlexNet (Krizhevsky et al., 2012) as the post-classifier, we replace it with the GoogLeNet (Szegedy et al., 2014a) which has many more layers and thus much larger capacity. Additionally, we augment the region proposals by combining the selective search boxes (Uijlings et al., 2013) with MultiBox predictions (Erhan et al.,

2014) for higher recall. In specific, we increase the proposal size by 2×, which halves the number of selective search object proposals. Besides, we add 200 region proposals generated by MultiBox (Erhan et al., 2014). As a result, we use about 60% of the proposals used in (Girshick et al., 2014), and increase the recall from 92% to 93%, which improve the mean average precision by 1% for a single model. Finally, we use an ensemble of 6 GoogLeNet models when classifying each region which improves accuracy from 38.0% to 43.9%. Note that we did not use bounding box regression as used in R-CNN. Figure 2.4 shows some high quality detection examples returned by the system.

Figure 2.4: High quality detection examples. We ranked **1st** in the ILSVRC2014 DET track (Russakovsky et al., 2015).

We first report the top detection results and show the progress since the first edition of the detection task. Compared to the 2013 result, the accuracy has almost doubled. The top performing teams all use convolutional networks. We report the official scores in Table 2.3 and common strategies for each team: the use of external data, ensemble models or contextual models. The external data is typically the ILSVRC12 classification data for pre-training a model that is later fine-tuned on the detection data. Some teams also mention the use of the localization data. Since a good portion of the localization task bounding boxes are not included in the detection dataset, one can pre-train a general bounding box regressor with this data the same way classification is used for pre-training. The GoogLeNet entry did not use the localization data for pretraining.

Team	Year	Place	mAP	external data	ensemble	approach
UvA-Euvision	2013	1st	22.6%	none	?	Fisher vectors
Deep Insight	2014	3rd	40.5%	CLS-LOC	3	CNN
CUHK DeepID-Net	2014	2nd	40.7%	CLS-LOC	?	CNN
GoogLeNet	2014	1st	**43.9%**	CLS-LOC	6	CNN

Table 2.3: **Results on ILSVRC2014 object detection *external* track.** Unreported values are noted with question marks.

In Table 2.4, we compare results using a single model only. All the methods followed the R-CNN framework (Girshick et al., 2014) with slightly different region proposals and different CNN models. Notably DeepID-Net proposed a new deformation layer which learns the deformable between object parts and the whole object as was done in DPM (Felzenszwalb et al., 2008), and was built on ZFNet (Zeiler and Fergus, 2014) – a network with same depth and accuracy as AlexNet. Our single model has similar accuracy even without this extra information because of the extraordinary power of GoogLeNet. The top performing model is by Deep Insight, which used context and bounding box regression as extra steps for their detector. Surprisingly it only improves by 0.3 points with an ensemble of 3 models while the GoogLeNet obtains significantly stronger results with the ensemble.

Team	mAP	Contextual model	Bounding box regression
Trimps-Soushen	31.6%	no	?
Berkeley Vision	34.5%	no	yes
UvA-Euvision	35.4%	?	?
CUHK DeepID-Net	37.7%	no	?
GoogLeNet	38.0%	no	no
Deep Insight	**40.2%**	yes	yes

Table 2.4: **Single model performance for ILSVRC2014 object detection *external* track.**

2.5 Conclusion

The R-CNN type of method formulate the object detection problem as a classification problem over region proposals. Although powerful, it is very slow and has several decouple steps, making it hard to train and evaluate. We are concerned with the efficiency of the model and the simplicity of the training methodology as much as we do about the quality. This not only enables faster turn around for development, but also is particularly useful when deploying on real time systems, such as mobile devices or self driving cars, which are sensitive to time delay and usually have limited computational power. In next chapter, we introduce an end-to-end simple system which is much faster and more accurate as well.

CHAPTER 3: Fast Object Detection from Static Images

Everything should be made as simple as possible, but not simpler.
– Albert Einstein

3.1 Introduction

Current state-of-the-art object detection systems are variants of the following approach: hypothesize bounding boxes, resample pixels or features for each box, and apply a high-quality classifier. This pipeline has prevailed on detection benchmarks since the Selective Search work (Uijlings et al., 2013) through the current leading results on PASCAL VOC, COCO, and ILSVRC detection all based on Faster R-CNN(Ren et al., 2015) albeit with deeper features such as (He et al., 2016). While accurate, these approaches have been too computationally intensive for embedded systems and, even with high-end hardware, too slow for real-time applications. Often detection speed for these approaches is measured in seconds per frame (SPF), and even the fastest high-accuracy detector, Faster R-CNN, operates at only 7 frames per second (FPS). There have been many attempts to build faster detectors by attacking each stage of the detection pipeline (see related work in Sec. 4.4), but so far, significantly increased speed comes only at the cost of significantly decreased detection accuracy.

We present the first deep network based object detector that does not resample pixels or features for bounding box hypotheses *and* and is as accurate as approaches that

do. This results in a significant improvement in speed for high-accuracy detection (59 FPS with mAP 74.3% on VOC2007 `test`, vs. Faster R-CNN 7 FPS with mAP 73.2% or YOLO 45 FPS with mAP 63.4%). The fundamental improvement in speed comes from eliminating bounding box proposals and the subsequent pixel or feature resampling stage. We are not the first to do this (cf (Sermanet et al., 2013; Redmon et al., 2015)), but by adding a series of improvements, we manage to increase the accuracy significantly over previous attempts. Our improvements include using a small convolutional filter to predict object categories and offsets in bounding box locations, using separate predictors (filters) for different aspect ratio detections, and applying these filters to multiple feature maps from the later stages of a network in order to perform detection at multiple scales. With these modifications—especially using multiple layers for prediction at different scales—we can achieve high-accuracy using relatively low resolution input, further increasing detection speed. While these contributions may seem small independently, we note that the resulting system improves accuracy on real-time detection for PASCAL VOC from 63.4% mAP for YOLO to 74.3% mAP for our SSD. This is a larger relative improvement in detection accuracy than that from the recent, very high-profile work on residual networks (He et al., 2016). Furthermore, significantly improving the speed of high-quality detection can broaden the range of settings where computer vision is useful.

We summarize our contributions as follows:

- We introduce SSD, a single-shot detector for multiple categories that is faster than the previous state-of-the-art for single shot detectors (YOLO), and significantly more accurate, in fact as accurate as slower techniques that perform explicit region

proposals and pooling (including Faster R-CNN).

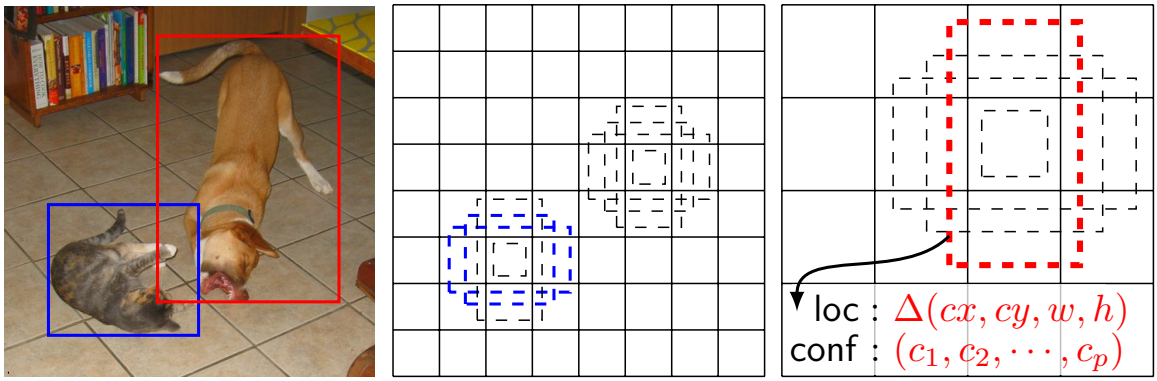
- The core of SSD is predicting category scores and box offsets for a fixed set of default bounding boxes using small convolutional filters applied to feature maps.
- To achieve high detection accuracy we produce predictions of different scales from feature maps of different scales, and explicitly separate predictions by aspect ratio.
- These design features lead to simple end-to-end training and high accuracy, even on low resolution input images, further improving the speed vs accuracy trade-off.
- Experiments include timing and accuracy analysis on models with varying input size evaluated on PASCAL VOC, COCO, and ILSVRC and are compared to a range of recent state-of-the-art approaches.

3.2 The Single Shot Detector (SSD)

This section describes our proposed SSD framework for detection (Sec. 3.2.1) and the associated training methodology (Sec. 3.2.2). Afterwards, Sec. 3.3 presents dataset-specific model details and experimental results.

3.2.1 Model

The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections. The early network layers are based on a standard architecture used for high



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

Figure 3.1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 (Girshick, 2015)) and confidence loss (e.g. Softmax).

quality image classification (truncated before any classification layers), which we will call the base network[1]. We then add auxiliary structure to the network to produce detections with the following key features:

Multi-scale feature maps for detection

We add convolutional feature layers to the end of the truncated base network. These layers decrease in size progressively and allow predictions of detections at multiple scales. The convolutional model for predicting detections is different for each feature layer (*cf* Overfeat(Sermanet et al., 2013) and YOLO(Redmon et al., 2015) that operate on a single scale feature map).

[1]We use the VGG-16 network as a base, but other networks should also produce good results.

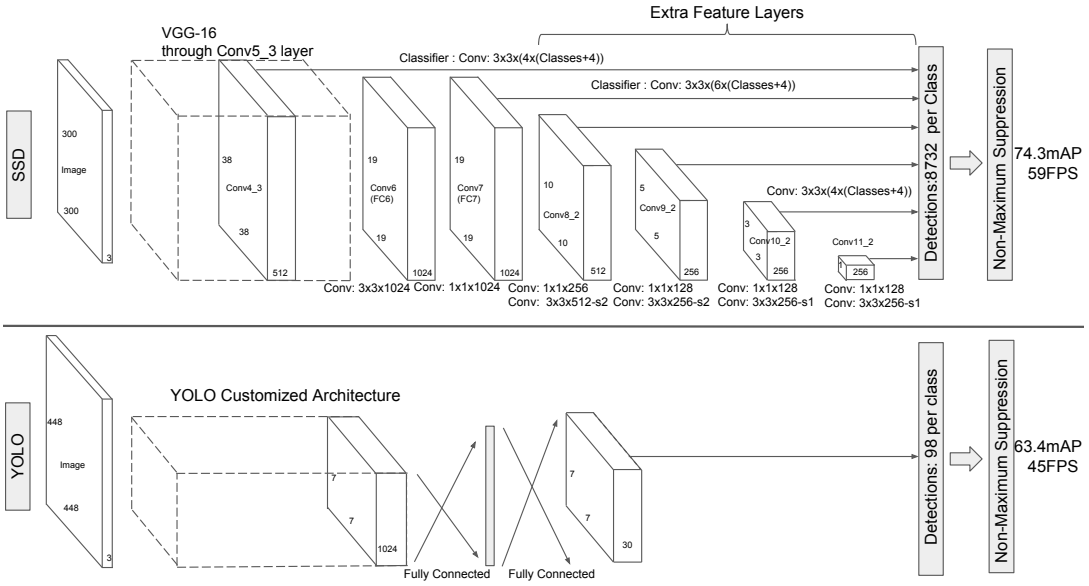


Figure 3.2: **A comparison between two single shot detection models: SSD and YOLO** (Redmon et al., 2015). Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

Convolutional predictors for detection

Each added feature layer (or optionally an existing feature layer from the base network) can produce a fixed set of detection predictions using a set of convolutional filters. These are indicated on top of the SSD network architecture in Fig. 3.2. For a feature layer of size $m \times n$ with p channels, the basic element for predicting parameters of a potential detection is a $3 \times 3 \times p$ *small kernel* that produces either a score for a category, or a shape offset relative to the default box coordinates. At each of the $m \times n$ locations where the kernel is applied, it produces an output value. The bounding box offset output values are measured relative to a default box position relative to each feature map location (*cf* the architecture of YOLO(Redmon et al., 2015) that uses an intermediate fully connected layer instead of a convolutional filter for this step).

Default boxes and aspect ratios

We associate a set of default bounding boxes with each feature map cell, for multiple feature maps at the top of the network. The default boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed. At each feature map cell, we predict the offsets relative to the default box shapes in the cell, as well as the per-class scores that indicate the presence of a class instance in each of those boxes. Specifically, for each box out of k at a given location, we compute c class scores and the 4 offsets relative to the original default box shape. This results in a total of $(c + 4)k$ filters that are applied around each location in the feature map, yielding $(c + 4)kmn$ outputs for a $m \times n$ feature map. For an illustration of default

boxes, please refer to Fig. 3.1. Our default boxes are similar to the *anchor boxes* used in Faster R-CNN (Ren et al., 2015), however we apply them to several feature maps of different resolutions. Allowing different default box shapes in several feature maps let us efficiently discretize the space of possible output box shapes.

3.2.2 Training

The key difference between training SSD and training a typical detector that uses region proposals, is that ground truth information needs to be assigned to specific outputs in the fixed set of detector outputs. Some version of this is also required for training in YOLO(Redmon et al., 2015) and for the region proposal stage of Faster R-CNN(Ren et al., 2015) and MultiBox(Erhan et al., 2014). Once this assignment is determined, the loss function and back propagation are applied end-to-end. Training also involves choosing the set of default boxes and scales for detection as well as the hard negative mining and data augmentation strategies.

Matching strategy

During training we need to determine which default boxes correspond to a ground truth detection and train the network accordingly. For each ground truth box we are selecting from default boxes that vary over location, aspect ratio, and scale. We begin by matching each ground truth box to the default box with the best jaccard overlap (as in MultiBox (Erhan et al., 2014)). Unlike MultiBox, we then match default boxes to any ground truth with jaccard overlap higher than a threshold (0.5). This simplifies the learning problem, allowing the network to predict high scores for multiple overlapping

default boxes rather than requiring it to pick only the one with maximum overlap.

Training objective

The SSD training objective is derived from the MultiBox objective (Erhan et al., 2014; Szegedy et al., 2014b) but is extended to handle multiple object categories. Let $x_{ij}^{p} = \{1, 0\}$ be an indicator for matching the i-th default box to the j-th ground truth box of category p. In the matching strategy above, we can have $\sum_i x_{ij}^{p} \geq 1$. The overall objective loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{3.1}$$

where N is the number of matched default boxes. If $N = 0$, wet set the loss to 0. The localization loss is a Smooth L1 loss (Girshick, 2015) between the predicted box (l) and the ground truth box (g) parameters. Similar to Faster R-CNN (Ren et al., 2015), we regress to offsets for the center (cx, cy) of the default bounding box (d) and for its width (w) and height (h).

$$\begin{aligned} L_{loc}(x, l, g) &= \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m) \\ \hat{g}_j^{cx} &= (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \\ \hat{g}_j^{w} &= \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right) \end{aligned} \tag{3.2}$$

The confidence loss is the softmax loss over multiple classes confidences (c).

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \tag{3.3}$$

and the weight term α is set to 1 by cross validation.

Choosing scales and aspect ratios for default boxes

To handle different object scales, some methods (Sermanet et al., 2013; He et al., 2014) suggest processing the image at different sizes and combining the results afterwards. However, by utilizing feature maps from several different layers in a single network for prediction we can mimic the same effect, while also sharing parameters across all object scales. Previous works (Long et al., 2014; Hariharan et al., 2015) 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, (Liu et al., 2015b) showed that adding global context pooled from a feature map can help smooth the segmentation results. Motivated by these methods, we use both the lower and upper feature maps for detection. Figure 3.1 shows two exemplar feature maps (8×8 and 4×4) which are used in the framework. In practice, we can use many more with small computational overhead.

Feature maps from different levels within a network are known to have different (empirical) receptive field sizes (Zhou et al., 2015). Fortunately, within the SSD framework, the default boxes do not necessary need to correspond to the actual receptive fields of each layer. We design the tiling of default boxes so that specific feature maps learn to be

responsive to particular scales of the objects. Suppose we want to use m feature maps for prediction. The scale of the default boxes for each feature map is computed as:

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m-1}(k-1), \quad k \in [1, m] \tag{3.4}$$

where $s_{\min}$ is 0.2 and $s_{\max}$ is 0.9, meaning the lowest layer has a scale of 0.2 and the highest layer has a scale of 0.9, and all layers in between are regularly spaced. We impose different aspect ratios for the default 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 default box. For the aspect ratio of 1, we also add a default box whose scale is $s'_k = \sqrt{s_k s_{k+1}}$, resulting in 6 default boxes per feature map location. We set the center of each default box to $(\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|})$, where $|f_k|$ is the size of the k-th square feature map, $i, j \in [0, |f_k|)$. In practice, one can also design a distribution of default boxes to best fit a specific dataset. How to design the optimal tiling is an open question as well.

By combining predictions for all default boxes 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 Fig. 3.1, the dog is matched to a default box in the 4×4 feature map, but not to any default boxes in the 8×8 feature map. This is because those boxes have different scales and do not match the dog box, and therefore are considered as negatives during training.

Hard negative mining

After the matching step, most of the default boxes are negatives, especially when the number of possible default boxes 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 loss for each default 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.

Data augmentation

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, 0.7, or 0.9.
- Randomly sample a patch.

The size of each sampled patch is [0.1, 1] of the original image size, and the aspect ratio is between $\frac{1}{2}$ and 2. We keep the overlapped part of the ground truth box if the center of it is in the sampled patch. After the aforementioned sampling step, each sampled patch is resized to fixed size and is horizontally flipped with probability of 0.5, in addition to applying some photo-metric distortions similar to those described in (Howard, 2013).

3.3 Experimental Results

Base network

Our experiments are all based on VGG16 (Simonyan and Zisserman, 2014), which is pre-trained on the ILSVRC CLS-LOC dataset (Russakovsky et al., 2015). Similar to DeepLab-LargeFOV (Chen et al., 2014), we convert fc6 and fc7 to convolutional layers, subsample parameters from fc6 and fc7, change pool5 from $2 \times 2 - s2$ to $3 \times 3 - s1$, and use the *à trous* algorithm (Holschneider et al., 1990) to fill the "holes". We remove all the dropout layers and the fc8 layer. We fine-tune the resulting model using SGD with initial learning rate 10^{-3}, 0.9 momentum, 0.0005 weight decay, and batch size 32. The learning rate decay policy is slightly different for each dataset, and we will describe details later. The full training and testing code is built on Caffe (Jia et al., 2014) and is open source at: `https://github.com/weiliu89/caffe/tree/ssd` .

3.3.1 PASCAL VOC2007

On this dataset, we compare against Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2015) on VOC2007 `test` (4952 images). All methods fine-tune on the same pre-trained VGG16 network.

Figure 3.2 shows the architecture details of the SSD300 model. We use conv4_3, conv7 (fc7), conv8_2, conv9_2, conv10_2, and conv11_2 to predict both location and confidences. We set default box with scale 0.1 on conv4_3[2]. We initialize the parameters for all the newly added convolutional layers with the "xavier" method (Glorot and Bengio, 2010).

[2]For SSD512 model, we add extra conv12_2 for prediction, set $s_{\min}$ to 0.15, and 0.07 on conv4_3.

For conv4_3, conv10_2 and conv11_2, we only associate 4 default boxes at each feature map location – omitting aspect ratios of $\frac{1}{3}$ and 3. For all other layers, we put 6 default boxes as described in Sec. 3.2.2. Since, as pointed out in (Liu et al., 2015b), conv4_3 has a different feature scale compared to the other layers, we use the L2 normalization technique introduced in (Liu et al., 2015b) to scale the feature norm at each location in the feature map to 20 and learn the scale during back propagation. We use the 10^{-3} learning rate for 40k iterations, then continue training for 10k iterations with 10^{-4} and 10^{-5}. When training on VOC2007 `trainval`, Table 3.1 shows that our low resolution SSD300 model is already more accurate than Fast R-CNN. When we train SSD on a larger 512×512 input image, it is even more accurate, surpassing Faster R-CNN by 1.7% mAP. If we train SSD with more (i.e. VOC2007 `trainval` + VOC2012 `trainval` – 07+12) data, we see that SSD300 is already better than Faster R-CNN by 1.1% and that SSD512 is 3.6% better. If we take models trained on COCO `trainval35k` as described in Sec. 3.3.4 and fine-tuning them on the 07+12 dataset with SSD512, we achieve the best results: 81.6% mAP.

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
Fast	07+12	70.0	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
Faster	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
Faster	07+12	73.2	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
Faster	07+12+COCO	78.8	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9
SSD300	07	68.0	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5
SSD300	07+12	74.3	75.5	80.2	72.3	66.3	47.6	83.0	84.2	86.1	54.7	78.3	73.9	84.5	85.3	82.6	76.2	48.6	73.9	76.0	83.4	74.0
SSD300	07+12+COCO	79.6	80.9	86.3	79.0	**76.2**	57.6	87.3	88.2	88.6	60.5	85.4	**76.7**	**87.5**	**89.2**	84.5	81.4	55.0	81.9	**81.5**	85.9	78.9
SSD512	07	71.6	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8
SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
SSD512	07+12+COCO	**81.6**	**86.6**	**88.3**	**82.4**	76.0	**66.3**	**88.6**	**88.9**	**89.1**	**65.1**	**88.4**	73.6	86.5	88.9	**85.3**	**84.6**	**59.1**	**85.0**	80.4	**87.4**	**81.2**

Table 3.1: **PASCAL VOC2007 `test` detection results.** Both Fast (Girshick, 2015) and Faster R-CNN (Ren et al., 2015) use input images whose minimum dimension is 600. The two SSD models have exactly the same settings except that they have different input sizes (300×300 vs. 512×512). It is obvious that larger input size leads to better results, and more data always helps. Data: "07": VOC2007 `trainval`, "07+12": union of VOC2007 and VOC2012 `trainval`. "07+12+COCO": first train on COCO `trainval35k` then fine-tune on 07+12.

To understand the performance of our two SSD models in more details, we used the detection analysis tool from (Hoiem et al., 2012). Figure 3.3 shows that SSD can detect various object categories with high quality (large white area). The majority of its confident detections are correct. The recall is around 85-90%, and is much higher with "weak" (0.1 jaccard overlap) criteria. Compared to R-CNN (Girshick et al., 2014), SSD has less localization error, indicating that SSD can localize objects better because it directly learns to regress the object shape and classify object categories instead of using two decoupled steps. However, SSD has more confusions with similar object categories (especially for animals), partly because we share locations for multiple categories. Figure 3.4 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. This is not surprising because those small objects may not even have any information at the very top layers. Increasing the input size (e.g. from 300×300 to 512×512) can help improve detecting small 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 object aspect ratios because we use default boxes of various aspect ratios per feature map location.

3.3.2 Model analysis

To understand SSD better, we carried out controlled experiments to examine how each component affects performance. For all the experiments, we use the same settings and input size (300×300), except for specified changes to the settings or component(s).

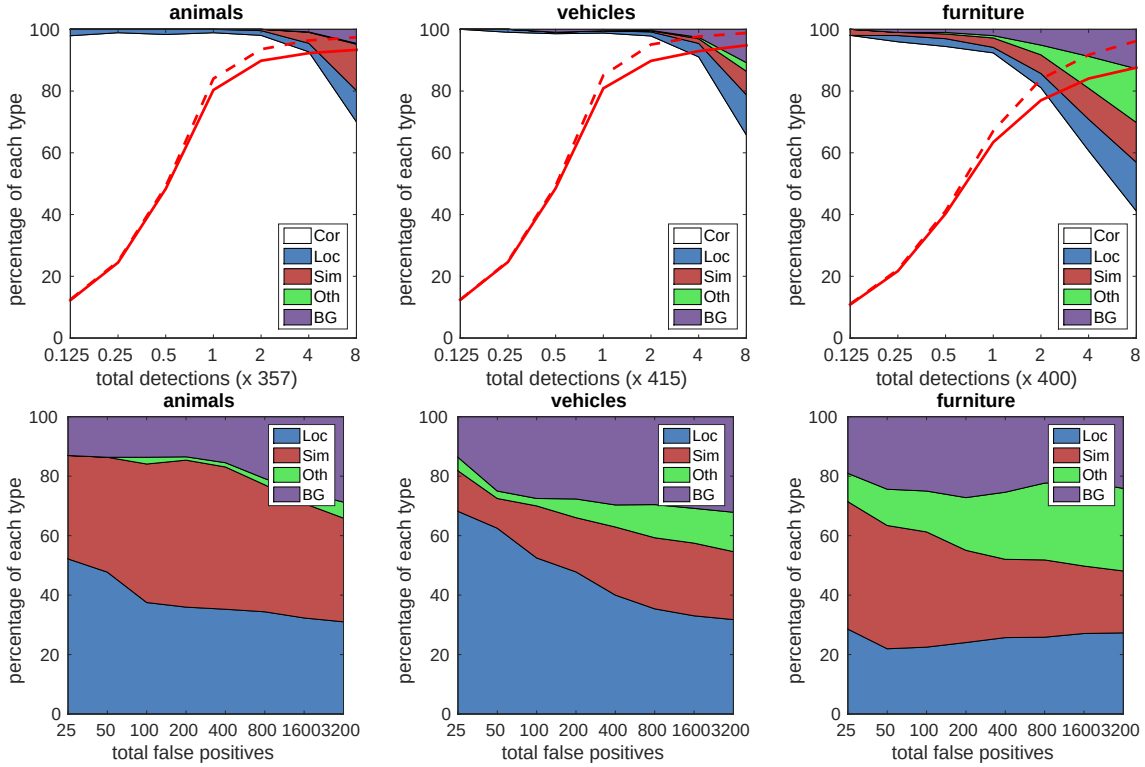


Figure 3.3: **Visualization of performance for SSD512 on animals, vehicles, and furniture from VOC2007 `test` using (Hoiem et al., 2012).** The top row shows the cumulative fraction of detections that are correct (Cor) or false positive due to poor localization (Loc), confusion with similar categories (Sim), with others (Oth), or with background (BG). The solid red line reflects the change of recall with "strong" criteria (0.5 jaccard overlap) as the number of detections increases. The dashed red line is using the "weak" criteria (0.1 jaccard overlap). The bottom row shows the distribution of top-ranked false positive types.

	SSD300				
more data augmentation?		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔			✔	✔
use atrous?	✔	✔	✔		✔
VOC2007 `test` mAP	65.5	71.6	73.7	74.2	**74.3**

Table 3.2: **Effects of various design choices and components on SSD performance.**

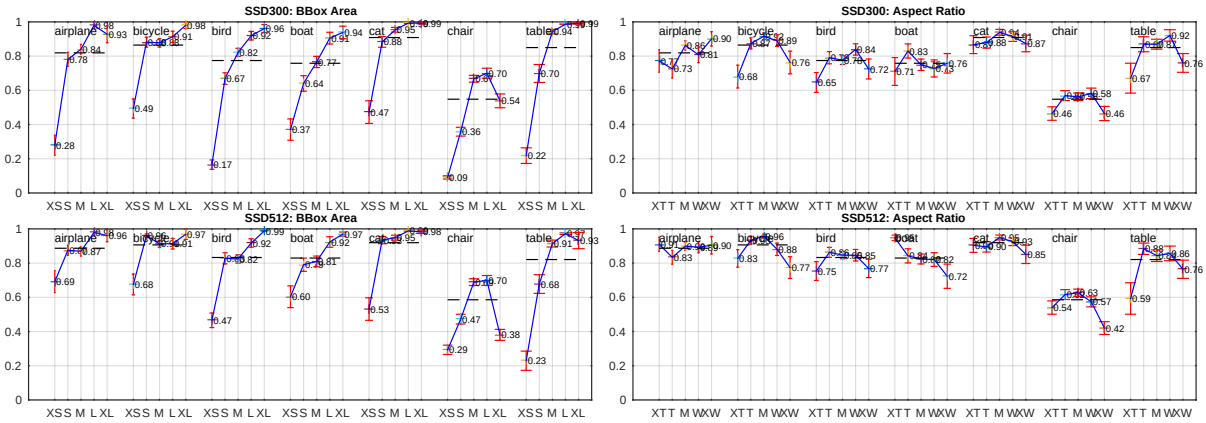


Figure 3.4: **Sensitivity and impact of different object characteristics on VOC2007 `test` set using (Hoiem et al., 2012).** Each plot shows the normalized AP (Hoiem et al., 2012) with standard error bars (red). Black dashed lines indicate overall normalized AP. The plot on the left shows the effects of BBox Area per category, and the right plot shows the effect of Aspect Ratio. 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.

Data augmentation is crucial

Fast and Faster R-CNN use the original image and the horizontal flip to train. We use a more extensive sampling strategy, similar to YOLO (Redmon et al., 2015). Table 3.2 shows that we can improve 8.8% mAP with this sampling strategy. We do not know how much our sampling strategy will benefit Fast and Faster R-CNN, but they are likely to benefit less because they use a feature pooling step during classification that is relatively robust to object translation by design.

More default box shapes is better

As described in Sec. 3.2.2, by default we use 6 default boxes per location. If we remove the boxes with $\frac{1}{3}$ and 3 aspect ratios, the performance drops by 0.6%. By further removing the boxes with $\frac{1}{2}$ and 2 aspect ratios, the performance drops another 2.1%. Using a variety of default box shapes seems to make the task of predicting boxes easier

Prediction source layers from:						mAP use boundary boxes?		# Boxes
conv4_3	conv7	conv8_2	conv9_2	conv10_2	conv11_2	Yes	No	
✔	✔	✔	✔	✔	✔	74.3	63.4	8732
✔	✔	✔	✔	✔		**74.6**	63.1	8764
✔	✔	✔	✔			73.8	68.4	8942
✔	✔	✔				70.7	69.2	9864
✔	✔					64.2	64.4	9025
	✔					62.4	64.0	8664

Table 3.3: **Effects of using multiple output layers.**

for the network.

Atrous is faster

As described in Sec. 3.3, we used the atrous version of a subsampled VGG16, following DeepLab-LargeFOV (Chen et al., 2014). If we use the full VGG16, keeping pool5 with $2 \times 2 - s2$ and not subsampling parameters from fc6 and fc7, and add conv5_3 for prediction, the result is about the same while the speed is about 20% slower.

Multiple output layers at different resolutions is better

A major contribution of SSD is using default boxes of different scales on different output layers. To measure the advantage gained, we progressively remove layers and compare results. For a fair comparison, every time we remove a layer, we adjust the default box tiling to keep the total number of boxes similar to the original (8732). This is done by stacking more scales of boxes on remaining layers and adjusting scales of boxes if needed. We do not exhaustively optimize the tiling for each setting. Table 3.3 shows a decrease in accuracy with fewer layers, dropping monotonically from 74.3 to 62.4. When we stack boxes of multiple scales on a layer, many are on the image boundary and need

to be handled carefully. We tried the strategy used in Faster R-CNN (Ren et al., 2015), ignoring boxes which are on the boundary. We observe some interesting trends. For example, it hurts the performance by a large margin if we use very coarse feature maps (e.g. conv11_2 (1×1) or conv10_2 (3×3)). The reason might be that we do not have enough large boxes to cover large objects after the pruning. When we use primarily finer resolution maps, the performance starts increasing again because even after pruning a sufficient number of large boxes remains. If we only use conv7 for prediction, the performance is the worst, reinforcing the message that it is critical to spread boxes of different scales over different layers. Besides, since our predictions do not rely on ROI pooling as in (Girshick, 2015), we do not have the *collapsing bins* problem (i.e. ROI pooling from low-resolution feature maps) (Zhang et al., 2016). The SSD architecture combines predictions from feature maps of various resolutions to achieve comparable accuracy to Faster R-CNN, while using lower resolution input images.

3.3.3 PASCAL VOC2012

We use the same settings as those used for our basic VOC2007 experiments above, except that we use VOC2012 `trainval` and VOC2007 `trainval` and `test` (21503 images) for training, and test on VOC2012 `test` (10991 images). We train the models with 10^{-3} learning rate for 60k iterations, then 10^{-4} for 20k iterations. Table 3.4 shows the results of our SSD300 and SSD512[3] model. We see the same performance trend as we observed on VOC2007 test. Our SSD300 improves accuracy over Fast/Faster R-CNN. By increasing the training and testing image size to 512×512, we are 4.5% more accurate than Faster

[3] `http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?cls=mean&challengeid=11&compid=4`

R-CNN. Compared to YOLO, SSD is significantly more accurate, likely due to the use of convolutional default boxes from multiple feature maps and our matching strategy during training. When fine-tuned from models trained on COCO, our SSD512 achieves 80.0% mAP, which is 4.1% higher than Faster R-CNN.

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast	07++12	68.4	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
Faster	07++12	70.4	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
Faster	07++12+COCO	75.9	87.4	83.6	76.8	62.9	59.6	81.9	82.0	91.3	54.9	82.6	59.0	89.0	85.5	84.7	84.1	52.2	78.9	65.5	85.4	70.2
YOLO	07++12	57.9	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
SSD300	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD300	07++12+COCO	77.5	90.2	83.3	76.3	63.0	53.6	83.8	82.8	92.0	59.7	82.7	63.5	89.3	87.6	85.9	84.3	52.6	82.5	**74.1**	**88.4**	74.2
SSD512	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
SSD512	07++12+COCO	**80.0**	**90.7**	**86.8**	**80.5**	**67.8**	**60.8**	**86.3**	**85.5**	**93.5**	**63.2**	**85.7**	**64.4**	**90.9**	**89.0**	**88.9**	**86.8**	**57.2**	**85.1**	72.8	**88.4**	**75.9**

Table 3.4: **PASCAL VOC2012 `test` detection results.** Fast (Girshick, 2015) and Faster R-CNN (Ren et al., 2015) use images with minimum dimension 600, while the image size for YOLO (Redmon et al., 2015) is 448×448. data: "07++12": union of VOC2007 `trainval` and `test` and VOC2012 `trainval`. "07++12+COCO": first train on COCO `trainval35k` then fine-tune on 07++12.

3.3.4 COCO

To further validate the SSD framework, we trained our SSD300 and SSD512 architectures on the COCO dataset. Since objects in COCO tend to be smaller than PASCAL VOC, we use smaller default boxes for all layers. We follow the strategy mentioned in Sec. 3.2.2, but now our smallest default box has a scale of 0.15 instead of 0.2, and the scale of the default box on conv4_3 is 0.07 (e.g. 21 pixels for a 300×300 image)[4].

We use the `trainval35k` (Bell et al., 2016) for training. We first train the model with 10^{-3} learning rate for 160k iterations, and then continue training for 40k iterations with 10^{-4} and 40k iterations with 10^{-5}. Table 3.5 shows the results on `test-dev2015`. Similar to what we observed on the PASCAL VOC dataset, SSD300 is better than Fast R-CNN

[4]For SSD512 model, we add extra conv12_2 for prediction, set $s_{\min}$ to 0.1, and 0.04 on conv4_3.
[5]Result is taken from (Bell et al., 2016).
[6]Result is taken from (COCO, 2016).

Method	data	Avg. Precision, IoU:			Avg. Precision, Area:			Avg. Recall, #Dets:			Avg. Recall, Area:		
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast[5]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster[6]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**

Table 3.5: **COCO `test-dev2015` detection results.** Fast (Girshick, 2015), Faster R-CNN (Ren et al., 2015), and ION (Bell et al., 2016) all use images with minimum dimension 600

in both mAP@0.5 and mAP@[0.5:0.95]. SSD300 has a similar mAP@0.75 as ION (Bell et al., 2016) and Faster R-CNN (COCO, 2016), but is worse in mAP@0.5. By increasing the image size to 512×512, our SSD512 is better than Faster R-CNN (COCO, 2016) in both criteria. Interestingly, we observe that SSD512 is 5.3% better in mAP@0.75, but is only 1.2% better in mAP@0.5. We also observe that it has much better AP (4.8%) and AR (4.6%) for large objects, but has relatively less improvement in AP (1.3%) and AR (2.0%) for small objects. Compared to ION, the improvement in AR for large and small objects is more similar (5.4% vs. 3.9%). We conjecture that Faster R-CNN is more competitive on smaller objects with SSD because it performs two box refinement steps, in both the RPN part and in the Fast R-CNN part. In Fig. 3.5, we show some detection examples on COCO `test-dev` with the SSD512 model.

3.3.5 Preliminary ILSVRC results

We applied the same network architecture we used for COCO to the ILSVRC DET dataset (Russakovsky et al., 2015). We train a SSD300 model using the ILSVRC2014

Figure 3.5: **Detection examples on COCO `test-dev` with SSD512 model.** We show detections with scores higher than 0.6. Each color corresponds to an object category.

DET `train` and `val1` as used in (Girshick et al., 2014). We first train the model with 10^{-3} learning rate for 320k iterations, and then continue training for 80k iterations with 10^{-4} and 40k iterations with 10^{-5}. We can achieve 43.4 mAP on the `val2` set (Girshick et al., 2014). Again, it validates that SSD is a general framework for high quality real-time detection.

3.3.6 Data Augmentation for Small Object Accuracy

Without a follow-up feature resampling step as in Faster R-CNN, the classification task for small objects is relatively hard for SSD, as demonstrated in our analysis (see Fig. 3.4). The data augmentation strategy described in Sec. 3.2.2 helps to improve the performance dramatically, especially on small datasets such as PASCAL VOC. The random crops generated by the strategy can be thought of as a "zoom in" operation and can generate many larger training examples. To implement a "zoom out" operation that creates more small training examples, we first randomly place an image on a canvas of 16× of the original image size filled with mean values before we do any random crop operation. Because we have more training images by introducing this new "expansion" data augmentation trick, we have to double the training iterations. We have seen a consistent increase of 2%-3% mAP across multiple datasets, as shown in Table 3.6. In specific, Figure 3.6 shows that the new augmentation trick significantly improves the performance on small objects. This result underscores the importance of the data augmentation strategy for the final model accuracy.

To understand the performance on COCO, we used the detection analysis tool from (Dollar, 2016). Because COCO has smaller objects than PASCAL VOC, Figure 3.7 shows

Method	VOC2007 `test`		VOC2012 `test`		COCO `test-dev2015` trainval35k		
	07+12	07+12+COCO	07++12	07++12+COCO			
	0.5	0.5	0.5	0.5	0.5:0.95	0.5	0.75
SSD300	74.3	79.6	72.4	77.5	23.2	41.2	23.4
SSD512	76.8	81.6	74.9	80.0	26.8	46.5	27.8
SSD300*	77.5	81.2	75.8	79.3	25.1	43.1	25.8
SSD512*	**79.5**	**83.2**	**78.5**	**82.2**	**28.8**	**48.5**	**30.3**

Table 3.6: **Results on multiple datasets when we add the image expansion data augmentation trick.** SSD300* and SSD512* are the models that are trained with the new data augmentation.

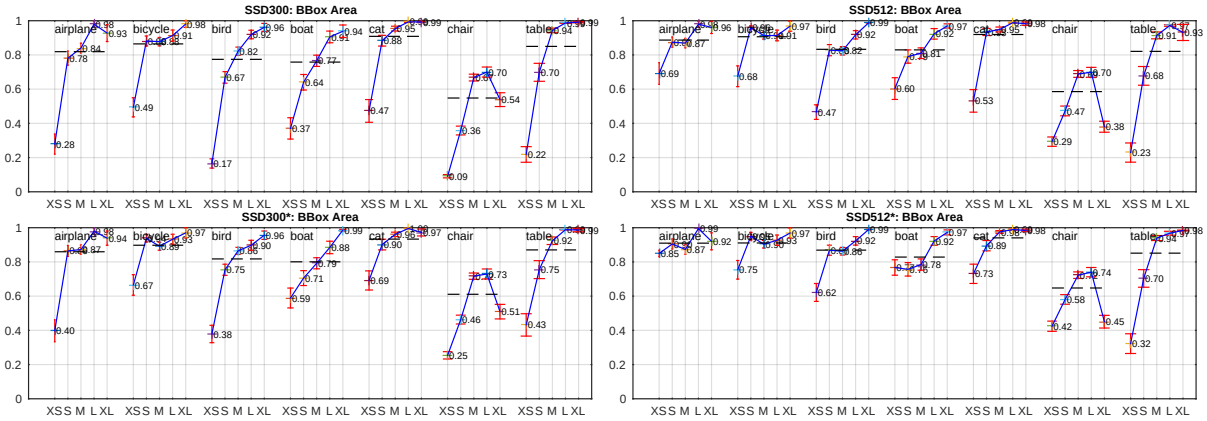


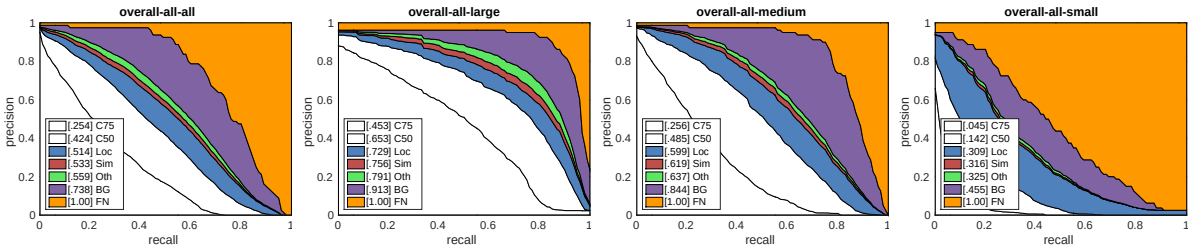
Figure 3.6: **Sensitivity and impact of object size with new data augmentation on VOC2007 `test` set using (Hoiem et al., 2012).** The top row shows the effects of BBox Area per category for the original SSD300 and SSD512 model, and the bottom row corresponds to the SSD300* and SSD512* model trained with the new data augmentation trick. It is obvious that the new data augmentation trick helps detecting small objects significantly.

that SSD struggles with localizing small objects well. Besides the false negatives (miss detections), we can also observe that there are two remaining major issues to solve: poor localization (LOC), confusion with background (BG). To improve the localization quality, we could design a better tiling of default boxes so that its position and scale are better aligned with the receptive field of each position on a feature map; we could also redesign the base network such that it is more suitable for object detection task instead of object classification task (e.g. having more equally distributed feature maps of various sizes). To help reduce confusion with background, we could mine more hard negative samples during training. We leave these for future work. We also show the analysis for person in Figure 3.8 and all 12 super categories in Figure 3.9, which have the same issues as described.

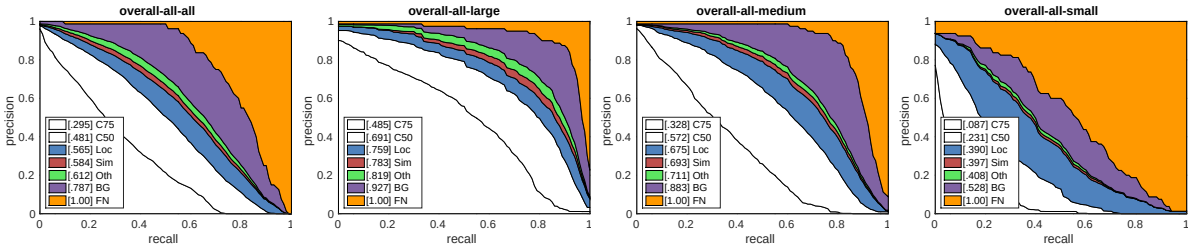
Inference time

Considering the large number of boxes generated from our method, it is essential to perform non-maximum suppression (nms) efficiently during inference. By using a confidence threshold of 0.01, we can filter out most boxes. We then apply nms with jaccard overlap of 0.45 per class and keep the top 200 detections per image. This step costs about 1.7 msec per image for SSD300 and 20 VOC classes, which is close to the total time (2.4 msec) spent on all newly added layers. We measure the speed with batch size 8 using Titan X and cuDNN v4 with Intel Xeon E5-2667v3@3.20GHz.

Table 3.7 shows the comparison between SSD, Faster R-CNN (Ren et al., 2015), and YOLO (Redmon et al., 2015). Both our SSD300 and SSD512 method outperforms Faster R-CNN in both speed and accuracy. Although Fast YOLO (Redmon et al., 2015) can

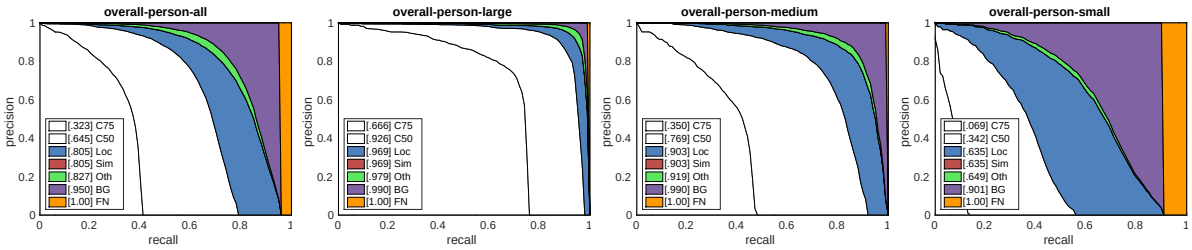


(a) SSD300*

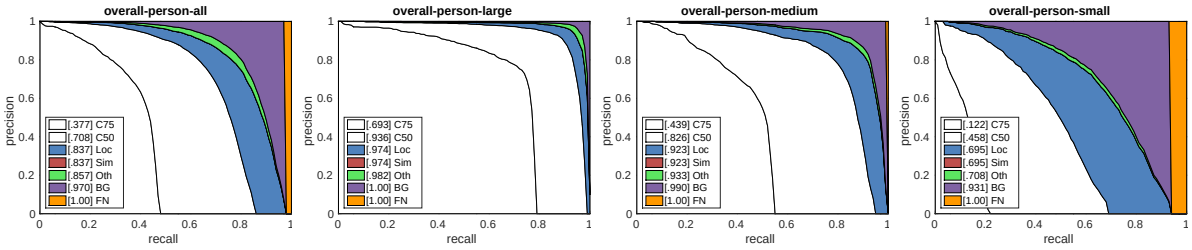


(b) SSD512*

Figure 3.7: **Comparison between SSD300* and SSD512* from COCO `minival` using COCO API (Dollar, 2016).** Each plot shows the cumulative fraction of detections that are correct with jaccard overlap 0.75 (C75) and 0.5 (C50) or false positive due to poor localization (Loc), confusion with similar categories (Sim), with others (Oth), or with background (BG), or false negative (FN). overall-all-all is a plot for all categories with all sizes, and the rest are for objects with large (area $> 96^2$ pixels), medium ($32^2 <$ area $< 96^2$ pixels), or small (area $< 32^2$ pixels) sizes.

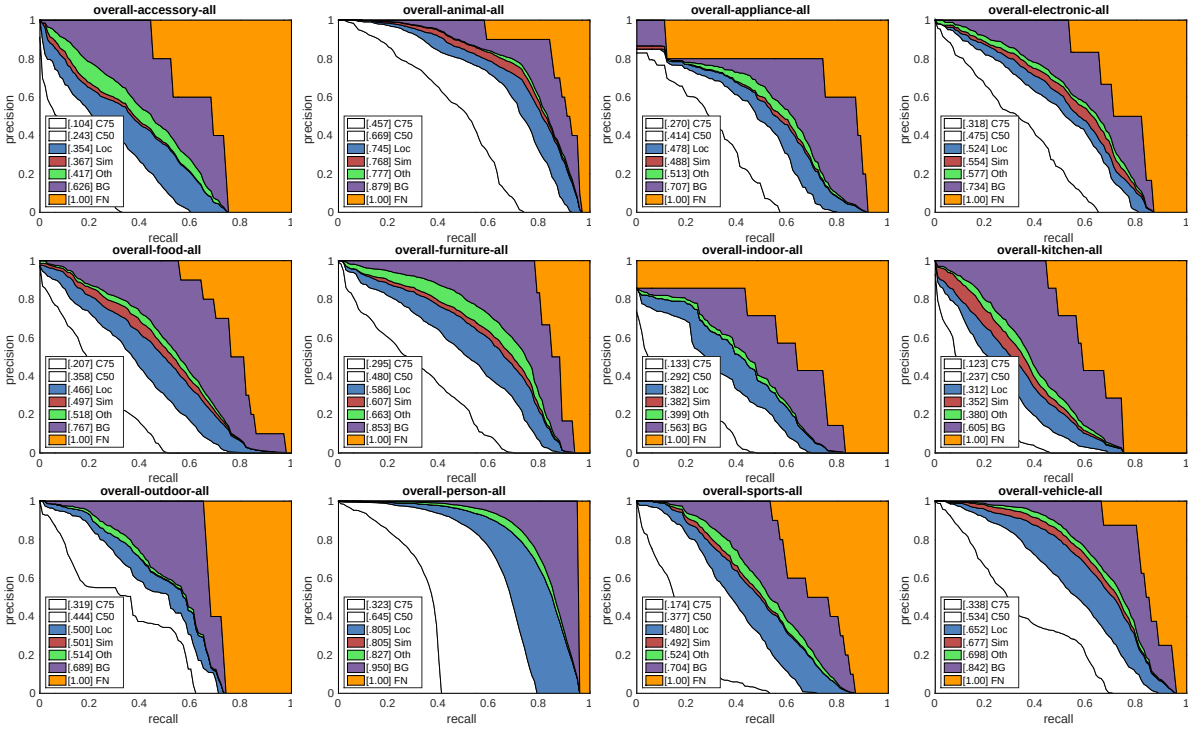


(a) SSD300*

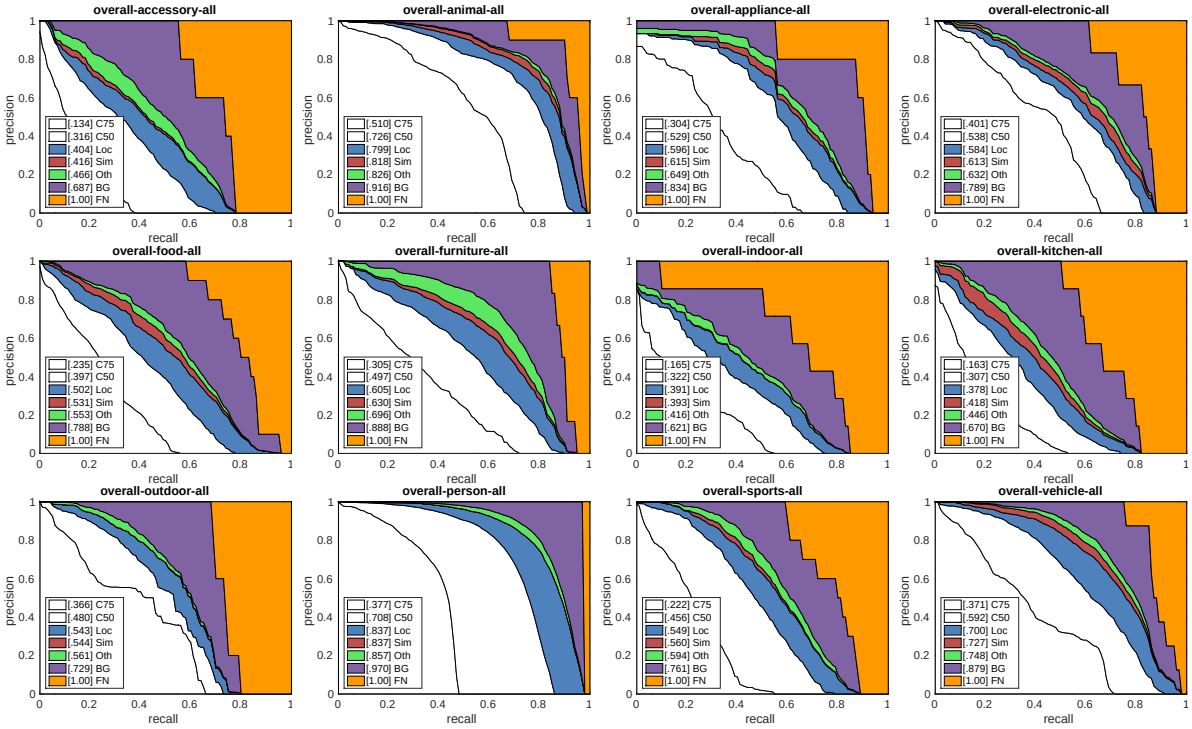


(b) SSD512*

Figure 3.8: **Comparison between SSD300* and SSD512* on coco person from COCO `minival` using COCO API (Dollar, 2016).** Each plot shows the cumulative fraction of detections that are correct with jaccard overlap 0.75 (C75) and 0.5 (C50) or false positive due to poor localization (Loc), confusion with similar categories (Sim), with others (Oth), or with background (BG), or false negative (FN). person-person-all is a plot for person with all sizes, and the rest are for person with large (area $> 96^2$ pixels), medium ($32^2 <$ area $< 96^2$ pixels), or small (area $< 32^2$ pixels) sizes.



(a) SSD300*



(b) SSD512*

Figure 3.9: **Comparison between SSD300* and SSD512* on all coco categories from COCO `minival` using COCO API (Dollar, 2016)**

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Table 3.7: **Results on Pascal VOC2007 `test`.** SSD300 is the only real-time detection method that can achieve above 70% mAP. By using a larger input image, SSD512 outperforms all methods on accuracy while maintaining a close to real-time speed.

run at 155 FPS, it has lower accuracy by almost 22% mAP. To the best of our knowledge, SSD300 is the first real-time method to achieve above 70% mAP. Note that about 80% of the forward time is spent on the base network (VGG16 in our case). Therefore, using a faster base network could even further improve the speed, which can possibly make the SSD512 model real-time as well.

3.4 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) (Felzenszwalb et al., 2008) and Selective Search (Uijlings et al., 2013) – had comparable performance. However, after the dramatic improvement brought on by R-CNN (Girshick et al., 2014), 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 (He et al., 2014) 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 (Girshick, 2015) 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 (Erhan et al., 2014) for learning objectness.

The second set of approaches improve the quality of proposal generation using deep neural networks. In the most recent works like MultiBox (Erhan et al., 2014; Szegedy et al., 2014b), 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 (Ren et al., 2015) 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. This way region proposals are used to pool mid-level features and the final classification step is less expensive. Our SSD is very similar to the region proposal network (RPN) in Faster R-CNN in that we also use a fixed set of (default) boxes for prediction, similar to the anchor boxes in the RPN. But instead of using these to pool features and evaluate another classifier, we simultaneously produce a

score for each object category in each box. Thus, our approach avoids the complication of merging RPN with Fast R-CNN and is easier to train, faster, 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 (Sermanet et al., 2013), a deep version of the sliding window method, predicts a bounding box directly from each location of the topmost feature map after knowing the confidences of the underlying object categories. YOLO (Redmon et al., 2015) 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 default boxes. However, our approach is more flexible than the existing methods because we can use default boxes of different aspect ratios on each feature location from multiple feature maps at different scales. If we only use one default box per location from the topmost feature map, our SSD would have similar architecture to OverFeat (Sermanet et al., 2013); if we use the whole topmost feature map and add a fully connected layer for predictions instead of our convolutional predictors, and do not explicitly consider multiple aspect ratios, we can approximately reproduce YOLO (Redmon et al., 2015).

3.5 Conclusions

We have introduced SSD, a fast single-shot object detector for multiple categories. A key feature of our model is the use of multi-scale convolutional bounding box outputs attached to multiple feature maps at the top of the network. This representation allows

us to efficiently model the space of possible box shapes. We experimentally validate that given appropriate training strategies, a larger number of carefully chosen default bounding boxes results in improved performance. We build SSD models with at least an order of magnitude more box predictions sampling location, scale, and aspect ratio, than existing methods (Redmon et al., 2015; Erhan et al., 2014). We demonstrate that given the same VGG-16 base architecture, SSD compares favorably to its state-of-the-art object detector counterparts in terms of both accuracy and speed. Our SSD512 model significantly outperforms the state-of-the-art Faster R-CNN (Ren et al., 2015) in terms of accuracy on PASCAL VOC and COCO, while being 3× faster. Our real time SSD300 model runs at 59 FPS, which is faster than the current real time YOLO (Redmon et al., 2015) alternative, while producing markedly superior detection accuracy.

Apart from its standalone utility, we believe that our monolithic and relatively simple SSD model provides a useful building block for larger systems that employ an object detection component. A promising future direction is to explore its use as part of a system using recurrent neural networks to detect and track objects in video simultaneously.

CHAPTER 4: Fast Semantic Segmentation with Context Cues

> I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees.
>
> – Max Wertheimer, *Laws of organization in perceptual forms*

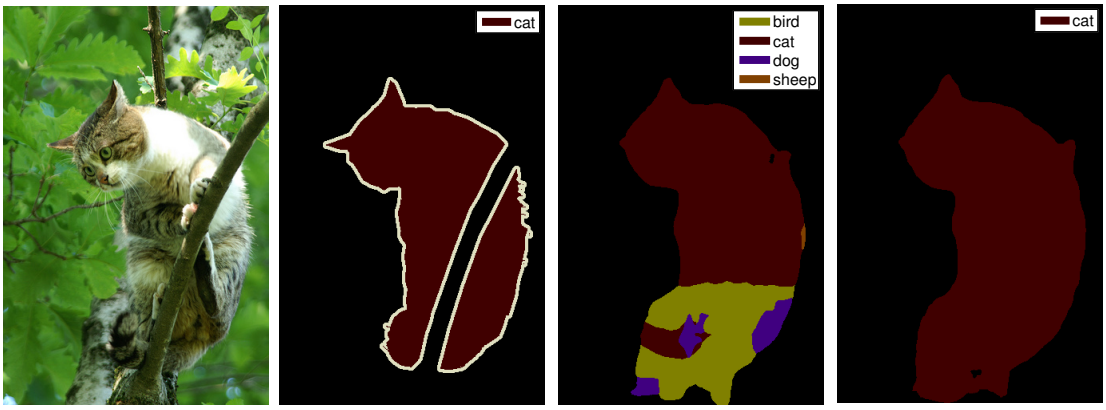
While localizing objects is a large step toward high level image understanding, we also want to do per-pixel labeling to get more fine-grained understanding of images. In this work, we propose *ParseNet*, an end-to-end simple and effective convolutional neural network for semantic segmentation. One of our main contributions, as shown in Fig. 4.1, is to use global context to help clarify local spurious predictions. This technique can be applied selectively to feature maps within a network, and can be used to combine information from multiple feature maps, as desired.

4.1 Introduction

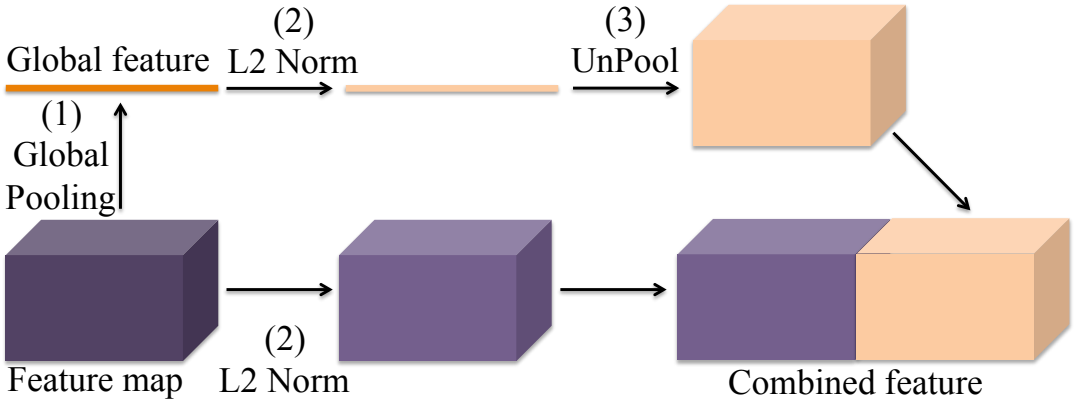
Semantic segmentation merges image segmentation with object recognition to produce per-pixel labeling of multiple classes in an image. The currently most successful techniques for semantic segmentation are based on the FCN framework (Long et al., 2014). These are adapted from networks designed to classify whole images (Krizhevsky et al., 2012; Szegedy et al., 2014a; Simonyan and Zisserman, 2014), and have demonstrated impressive level of performance. The FCN approach can be thought of as sliding a classification network around an input image, and processing each sliding window area

independently. Such unrolled convolutional operation can reuse the computation from the overlapped regions between the sliding windows, and is thus computational efficient. Also since FCN tries to optimize per-pixel accuracy, it disregards global information about an image, thus ignoring potentially useful scene-level semantic context. In order to integrate more context, several approaches (Chen et al., 2014; Schwing and Urtasun, 2015; Lin et al., 2015; Zheng et al., 2015), propose using techniques from graphical models such as conditional random fields (CRFs) models to introduce global context and structured information into a FCN. Although powerful, these architectures can be complex, combining both the challenges of tuning a deep neural network and a CRF, and require a fair amount of experience in managing the idiosyncrasies of training methodology and parameters. At the very least, this leads to time-consuming training and inference.

In this work, we propose *ParseNet*, an end-to-end simple and effective convolutional neural network for semantic segmentation. One of our main contributions, as shown in Fig. 4.1, is to use global context to help clarify local confusions. Looking back at previous work, adding global context for semantic segmentation is not a new idea, but has so far been pursued in patch-based frameworks (Lucchi et al., 2011). Such patch-based approaches have much in common with detection and segmentation works that have also shown benefits from integrating global context into classifying regions or objects in an image (Szegedy et al., 2014b; Mostajabi et al., 2014; Mottaghi et al., 2014). Our approach allows integrating global context in an end-to-end fully convolutional network (as opposed to a patch-based approach) for semantic segmentation with a small computational overhead. In our setting, the image is not divided into regions or objects, instead the network makes a joint prediction of all pixel values. Previous work on fully



(a) Image (b) Truth (c) FCN (d) ParseNet



(e) ParseNet contexture module overview.

Figure 4.1: **ParseNet uses extra global context to clarify local confusion and smooth segmentation.**

convolutional networks did not include global features, and there were limits in the pixel distance across which consistency in labeling was maintained.

The key idea that allows adding global context to the FCN framework is simple, but has several important consequences in addition to improving the accuracy of FCN. First, the entire end-to-end process is a single deep network, making training relatively straightforward compared to combining deep networks and CRFs. In addition, the way we add global context does not introduce much computational overhead versus training and evaluating a standard FCN, while improving performance significantly. In our ap-

proach, the feature map for a layer is pooled over the whole image to result in a context vector. This is appended to each of the features sent on to the subsequent layer of the network. In implementation, this is accomplished by unpooling the context vector and appending the resulting feature map with the standard feature map. The process is shown in Fig. 4.1. This technique can be applied selectively to feature maps within a network, and can be used to combine information from multiple feature maps, as desired. Notice that the scale of features from different layers may be quite different, making it difficult to directly combine them for prediction. We find that L_2 normalizing features for each layer and combining them using a scaling factor learned through backpropagation works well to address this potential difficulty.

In section 4.3, we demonstrate that these operations, appending global context pooled from a feature map along with an appropriate scaling, are sufficient to significantly improve performance over the basic FCN, resulting in accuracy on par with the method of (Chen et al., 2014) that uses detailed structure information for post processing. That said, we do not advocate ignoring the structure information. Instead, we posit that adding the global feature is a simple and robust method to improve FCN performance by considering contextual information. In fact, our network can be combined with explicit structure output prediction, e.g. a CRF, to potentially further increase performance.

4.2 ParseNet

4.2.1 Global Context

Context is known to be very useful for improving performance on detection and segmentation tasks using deep learning. (Mostajabi et al., 2014; Szegedy et al., 2014b) and references therein illustrate how context can be used to help in different tasks. For semantic segmentation, per pixel classification is often ambiguous in the presence of only local information. However, the task becomes much simpler if contextual information from the whole image is available.

It is known that features from the top layers of a network have very large receptive fields (e.g. fc7 in FCN with VGG has a 404×404 pixels receptive field). It is theoretically possible for an FCN to learn to capture context information if that is useful. However, in practice, we observe that the network learns to be "myopic" (e.g. focus on object parts) and to ignore some background information. The empirical size of the receptive fields is much smaller than the theoretical maximum, and is not enough to capture the global context. Following (Zhou et al., 2015), we slide a small patch of random noise across the input image, and measure the change in the activations of the desired layer. If the activation does not vary significantly, that suggests the given random patch is outside of the empirical receptive field, as shown in Figure 4.2. The effective receptive field at the last layer of this network barely covers $\frac{1}{4}$ of the entire image. Fortunately, it is rather straightforward to get the context within the FCN architecture. Specifically, we use global average pooling and pool the context features from the last layer or any layer if that is desired. The quality of semantic segmentation is greatly improved by adding

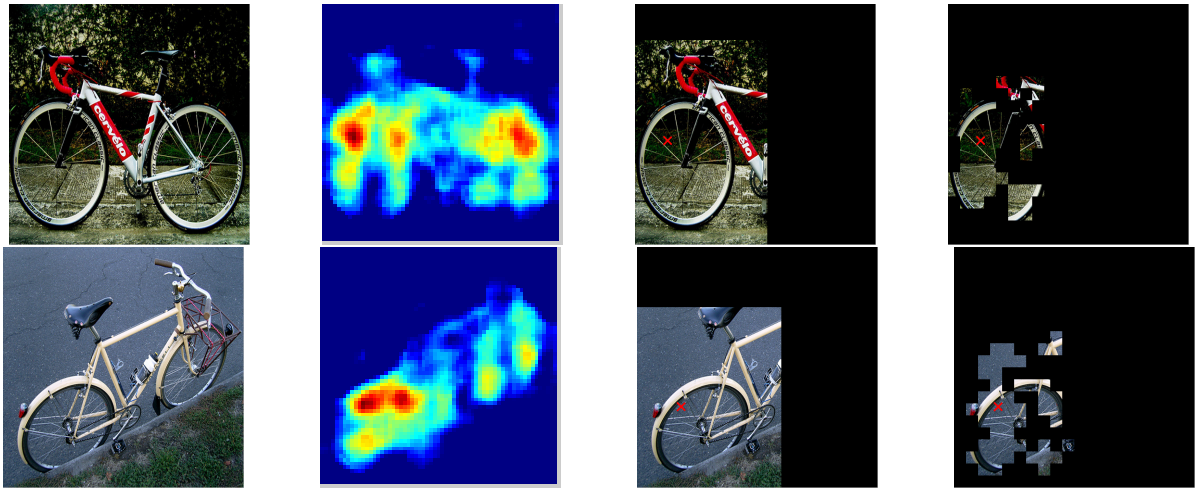
the additional global feature to local feature map, either with early fusion [1] or late fusion as discussed in Sec. 4.2.2. For example, Fig 4.1 has misclassified a large portion of the image as *bird* since it only used local information, however, adding contextual information in the loop, which might contain strong signal about the presence of *cat*, corrects the mistake. Experiment results on VOC2012 and PASCAL-Context dataset also verify our assumption. Compared with (Chen et al., 2014), the improvement is similar as of using CRF to post-process the output of FCN.

In addition, we also tried to follow the spatial pyramid idea (Lazebnik et al., 2006) to pool features from increasingly finer sub-regions and attach them to local features in the sub-regions, however, we did not observe significant improvements. We conjecture that it is because the (empirical) receptive field of high-level feature maps is larger than or is similar to those sub-regions. However features pooled from the whole image are still beneficial.

4.2.2 Early Fusion and Late Fusion

Once we get the global context feature, there are two general standard paradigms of using it with the local feature map. First, the *early fusion*, illustrated in in Fig. 4.1 where we unpool (replicate) global feature to the same size as of local feature map spatially and then concatenate them, and use the combined feature to learn the classifier. The alternative approach, is *late fusion*, where each feature is used to learn its own classifier, followed by merging the two predictions into a single classification score (Long et al.,

[1] Use the unpool operation by simply replicating the global feature horizontally and vertically to have the same size as the local feature map.



(a) Original Image (b) Activation map (c) Theoretical RF (d) Empirical RF

Figure 4.2: **Receptive field (RF) size for last layer.** (a) original image; (b) activation map on bicycle from a channel of the last layer of a network; (c) theoretical receptive field of the maximum activation (marked by red cross) is defined by the network structure; (d) empirical receptive field affecting the activation. Clearly empirical receptive field is not large enough to capture the global context.

2014; Chen et al., 2014). There are cons and pros for both fusion methods. If there is no additional processing on combined features, *early fusion* is quite similar to *late fusion* as pointed out in (Hariharan et al., 2015). With *late fusion*, there might be a case where individual features cannot recognize something but combining them may and there is no way to recover from independent predictions. Our experiments show that both methods work more or less the same if we normalize the feature properly for the early fusion case.

When merging the features, one must be careful to normalize each individual feature to make the combined feature work well; in classical computer vision this is referred to as the cue combination problem. As shown in Fig. 4.3, we extract a feature vector at a position combined from increasing higher level layers (from left to right), with the lower level features having a significantly larger scale than the higher level layers. As we show in Sec. 4.3.2, by naively combining features, the resultant feature will not be discriminative,

and heavy parameter tuning will be required to achieve sufficient accuracy. Instead, we can first L_2 normalize each feature and also possibly learn the scale parameter, which makes the learning more stable. We will describe more details in Sec. 4.2.3.

4.2.3 L_2 Normalization Layer

As discussed above and shown in Fig. 4.3, we need to combine two (or more) feature vectors, which generally have different scale and norm. Naively concatenating features leads to poor performance as the "larger" features dominate the "smaller" ones. Although during training, the weight might adjust accordingly, it requires very careful tuning of parameters and depends on the dataset, thus goes against the principle of simplicity and robustness. We find that by normalizing each individual feature first, and also learning to scale each differently, it makes the training more stable and improves performance.

The L_2 norm layer is not only useful for feature combination. As was pointed out above, in some cases *late fusion* also works equally well, but only with the help of L_2 normalization. For example, if we want to use lower level features to learn a classifier, as demonstrated in Fig. 4.3, some of the features will have very large norm. It is not trivial to learn with it without careful weight initialization and parameter tuning. A work around strategy is to apply an additional convolutional layer (Chen et al., 2014; Hariharan et al., 2015) and use several stages of fine-tuning (Long et al., 2014) with a much lower learning rate for the lower layer. This again goes against the principle of simplicity and robustness. In our work, we apply the L_2 norm and learn the scale parameter for each channel before using the feature for classification, which leads to more stable training.

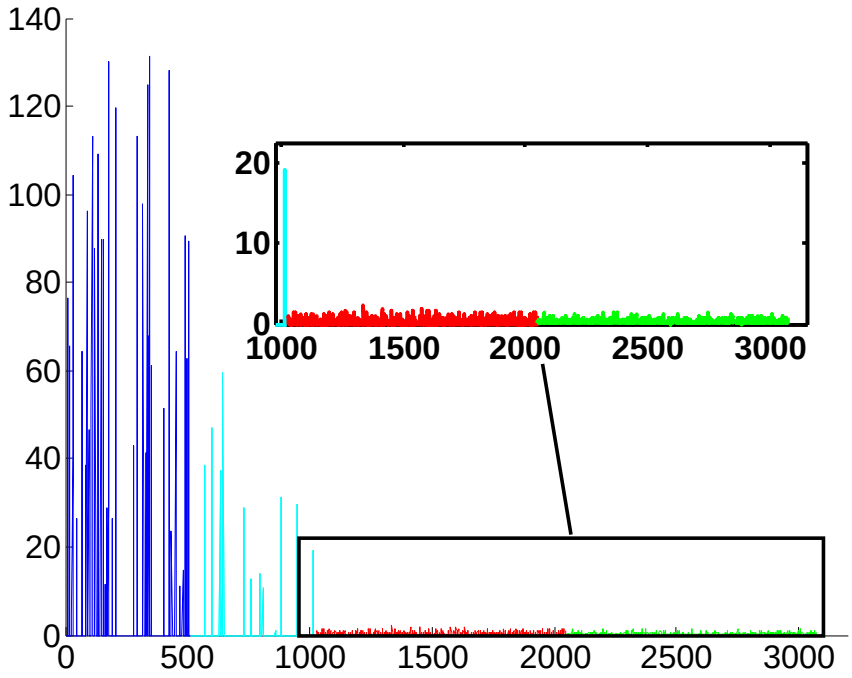


Figure 4.3: **Features from 4 different layers have activations that are of drastically different scales.** Each color corresponds to a different layers' feature. While *blue* and *cyan* are on a comparable scale, *red* and *green* features are of a scale 2 orders of magnitude less.

Formally, let ℓ be the loss we want to minimize. Here we use the summed softmax loss. For a layer with d-dimensional input $\mathbf{x} = (x_1 \cdots x_d)$, we will normalize it using L_2-norm[2] with $\hat{\mathbf{x}} = \frac{\mathbf{x}}{||\mathbf{x}||_2}$ where $||\mathbf{x}||_2 = \left(\sum_{i=1}^{d} |x_i|^2\right)^{1/2}$ is the L_2 norm of $\mathbf{x}$.

Note that simply normalizing each input of a layer changes the scale of the layer and will slow down the learning if we do not scale it accordingly. For example, we tried to normalize a feature s.t. L_2-norm is 1, yet we can hardly train the network because the features become very small. However, if we normalize it to e.g. 10 or 20, the network begins to learn well. Motivated by batch normalization (Ioffe and Szegedy, 2015) and PReLU (He et al., 2015), we introduce a scaling parameter γ_i, for each channel, which scales the normalized value by $y_i = \gamma_i \hat{x}_i$.

[2]We have only tried L_2 norm, but can also potentially try other l_p norms.

The number of extra parameters is equal to total number of channels, and are negligible and can be learned with backprogation. Indeed, by setting $\gamma_i = ||\mathbf{x}||_2$, we could recover the L_2 normalized feature, if that was optimal. Notice that this is simple to implement as the normalization and scale parameter learning only depend on each input feature vector and do not need to aggregate information from other samples as batch normalization does. During training, we use backpropagation and chain rule to compute derivatives with respect to scaling factor γ and input data $\mathbf{x}$

$$\frac{\partial \ell}{\partial \hat{\mathbf{x}}} = \frac{\partial \ell}{\partial \mathbf{y}} \cdot \gamma \qquad \frac{\partial \ell}{\partial \mathbf{x}} = \frac{\partial \ell}{\partial \hat{\mathbf{x}}} \Big(\frac{\mathbf{I}}{||\mathbf{x}||_2} - \frac{\mathbf{x}\mathbf{x}^T}{||\mathbf{x}||_2^3} \Big) \qquad \frac{\partial \ell}{\partial \gamma_i} = \sum_{y_i} \frac{\partial \ell}{\partial y_i} \hat{x}_i \tag{4.1}$$

For our case, we need to do L_2-norm per each pixel in a feature map instead of the whole. We can easily extend the equations by doing it elemental wise.

4.3 Experiments

In this section, we mainly report results on two benchmark datasets: VOC2012 (Everingham et al., 2014) and PASCAL-Context (Mottaghi et al., 2014). VOC2012 has 20 object classes and one background class. Following (Long et al., 2014; Chen et al., 2014), we augment it with extra annotations from (Hariharan et al., 2011) that leads to 10,582, 1,449, and 1,456 images for training, validation, and testing. PASCAL-Context (Mottaghi et al., 2014) fully labeled all scene classes appearing in VOC2010. We follow the same training + validation split as defined and used in (Mottaghi et al., 2014; Long et al., 2014), resulting in 59 object + stuff classes and one background classes with 4,998 and 5105 training and validation images. All the results we describe below use the training

images to train, and most of the results are on the validation set. We also report results on VOC2012 test set. All our models use the VGG-16 architecture pretrained on ILSVRC classification dataset (Russakovsky et al., 2015), and are fine-tuned for different training datasets using Caffe (Jia et al., 2014).

4.3.1 Best fine-tuning practices

As hyper-parameters are important for training/fine-tuning network, by exploring the hyper-parameter space a bit, we can reproduce better baseline performance for the state-of-the-art systems.

PASCAL-Context

We start from the public system FCN-32s[3] on PASCAL-Context. It uses the accumulated gradient and affine transformation tricks that were introduced in (Long et al., 2014). As such, it can deal with input images of any size without warping and cropping them to a fixed size, which can distort the image and affect the final segmentation result. Table 4.1 shows our different versions of reproduced baseline results. Baseline A uses the exactly same protocol, and our result is 1.5% lower. In Baseline B, we tried more iterations (160k vs. 80k) of fine-tuning and achieved similar performance to the reported numbers. Then, we modified the network a bit, i.e. we used "xavier" initialization (Glorot and Bengio, 2010), higher base learning rate (1e-9 vs. 1e-10), and lower momentum (0.9 vs. 0.99), and we achieved 1% higher accuracy as shown in Baseline C. Furthermore, we also removed the 100 padding in the first convolution layer and observed no significant

[3] `https://gist.github.com/shelhamer/80667189b218ad570e82\#file-readme-md`

difference but network trained slightly faster. Finally, we also used the "poly" learning rate policy (base_lr $\times (1 - \frac{\text{iter}}{\text{max\_iter}})^{\text{power}}$, where power is set to 0.9.) as it is proved to converge faster than the normal "step" policy, and thus can achieve 1.5% better performance with the same iterations (80k). All experimental results on PASCAL-Context are shown in table 4.1.

PASCAL-Context	Mean IoU
FCN-32s	35.1
Baseline A	33.57
Baseline B	35.04
Baseline C	36.16
Baseline D	**36.64**

Table 4.1: **Reproducing FCN-32s on PASCAL-Context.** There are various modifications of the architecture that are described in Section 4.1.

PASCAL VOC2012

We carry over the hyper-parameters we found on PASCAL-Context to VOC2012. We tried both DeepLab and DeepLab-LargeFOV models[4]. Table 4.2 shows the reproduced baseline results. DeepLab is very similar to FCN-32s, and our reproduced result, namely DeepLab Baseline, is 5.2% better (64.96 vs. 59.80) using the parameters we found in PASCAL-Context. DeepLab-LargeFOV uses the filter rarefication technique (*à trous* algorithm) that has much fewer parameters and is faster. We also use the same parameters on this architecture and can achieve 3.5% improvements. We name it as DeepLab-LargeFOV Baseline. The gap between these two models is not significant anymore as reported in (Chen et al., 2014).

Until now, we have seen that the parameters and details are important to get best

[4] `https://bitbucket.org/deeplab/deeplab-public/`

VOC2012	Mean IoU
DeepLab (Chen et al., 2014)	59.80
DeepLab-LargeFOV (Chen et al., 2014)	62.25
DeepLab Baseline	64.96
DeepLab-LargeFOV Baseline	**65.82**

Table 4.2: **Reproducing DeepLab and DeepLab-LargeFOV results on PASCAL VOC2012.**

performance using FCN models. Below, we report all our results with the reproduced baseline networks.

4.3.2 Combining Local and Global Features

In this section, we report results of combining global and local features on three datasets: SiftFlow (Liu et al., 2011), PASCAL-Context, and PASCAL VOC2012. For simplicity, we use pool6 as the global context feature, conv5 as conv5_3, conv4 as conv4_3, and conv3 as conv3_3 through the rest of paper.

SiftFlow

SiftFlow is a relatively small dataset that only has 2,688 images with 33 densely labeled semantic categories. The image size is 255×255. We do not use the geometric categories during training. We use the DeepLab Baseline network with the hyper-parameters found in PASCAL-Context. Instead of using two stages of learning as done in (Long et al., 2014), we used the concatenated features from different layers and global context features to directly learn to segment the images. As shown in Table 4.3, adding more layers can normally improve the performance as lower level layers have more detailed information. We also notice that adding global context features does not help much. We

hypothesize that global context is less helpful for small images.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-16s (Long et al., 2014)	85.2	51.7	39.5	76.1
fc7	85.1	44.1	35.4	75.6
pool6 + fc7	85.7	43.9	35.5	76.4
pool6 + fc7 + conv5	85.4	51.4	38.7	76.3
pool6 + fc7 + conv5 + conv4	**86.8**	**52.0**	**40.4**	**78.1**

Table 4.3: **Results on SiftFlow.** We report four metrics used in FCN (Long et al., 2014): pixel accuracy, mean accuracy, mean IU, and frequency weighted IU. Early fusion can work equally well as late fusion as used in FCN. With more layers of feature, the performance is consistently increasing. The global feature is not that helpful as the receptive field size of fc7 is large enough to cover most of the input image.

PASCAL-Context

We then apply the DeepLab Baseline model on PASCAL-Context by concatenating features from different layers of the network. As shown in Table 4.4, by adding global context pool6, it instantly helps improve by about 1.6%, which means that context is useful here as opposed to the observation in SiftFlow. Context becomes more important proportionally to the image size. Another interesting observation from the table is that, without normalization, performance keeps increasing until we add conv5. However, if we naively keep adding conv4, it starts decreasing the performance a bit; and if we add conv3, the network collapses. Interestingly, if we normalize all the features before we combine them, we don't see such a drop, instead, adding all the features together can achieve the state-of-the-art result on PASCAL-Context as far as we know.

	w/o Norm	w/ Norm
FCN-32s (Long et al., 2014)	35.1	N/A
FCN-8s (Long et al., 2014)	37.8	N/A
fc7	36.6	36.2
pool6 + fc7	38.2	37.6
pool6 + fc7 + conv5	39.5	39.9
pool6 + fc7 + conv5 + conv4	36.5	40.2
pool6 + fc7 + conv5 + conv4 + conv3	0.009	**40.4**

Table 4.4: **Results on PASCAL-Context with or without normalization.** Adding more layers helps if we L_2 normalize them.

PASCAL VOC2012

Since we have reproduced both DeepLab Baseline and DeepLab-LargeFOV Baseline on VOC2012, we want to see how global context, normalization, and early or late fusion affects the performance.

We start with using the DeepLab Baseline, and try to add pool6 to it. It improves from 64.92% to 67.49% by adding pool6 with normalization. Interestingly, without normalizing fc7 and pool6, we don't see any improvements as opposed to what we observed from SiftFlow and PASCAL-Context. We hypothesize this is due to images in VOC2012 mostly have one or two objects in the image versus the other two datasets, which have multiple labels per image, and we need to adjust the weight more carefully to make the context feature more useful. It also suggests that context is more useful for sparsely labeled images.

DeepLab-LargeFOV Baseline performance is higher than DeepLab Baseline and it is faster, thus we switch to use it for most of the experimental comparison for VOC2012. As shown in Table 4.5, we observe a similar pattern to DeepLab Baseline: adding pool6 can help improve the performance by 3.8%. However, we also notice that if we do not

normalize fc7 and pool6 and learn the scaling factors, its effect is diminished. Furthermore, we notice that early fusion and late fusion both work very similarly. Figure 4.4 illustrates some examples of how global context helps. We can clearly see that without using the context feature, the network confuses between similar categories and makes spurious predictions. Two similar looking patches are indistinguishable by the network if considered in isolation. However, adding context solves this issue as the global context helps to discriminate the local patches more accurately. On the other hand, sometimes context also confuses the predictions as shown in Figure 4.5. For example, in the first row, the global context feature definitely captured the spotty dog information that it used to help discriminate sheep from dog. However, it also added bias to classify the spotty horse as a dog. The other three examples have the same issue. Overall, by learning to weight pool6 and fc7 after L_2 normalization helps improve the performance greatly.

Layers	Norm (Y/N)	Early or Late (E/L)	Mean IoU
fc7	N	NA	65.82
fc7	Y	NA	65.66
pool6 + fc7	N	E	65.30
pool6 + fc7	Y	E	69.43
pool6 + fc7	Y	L	**69.55**
pool6 + fc7	N	L	69.29

Table 4.5: **Adding context for DeepLab-LargeFOV Baseline on VOC2012.**

We also tried to combine lower level features as was done with PASCAL-Context and SiftFlow, but no significant improvements using either *early fusion* or *late fusion* were observed. We believe it is because the fc7 of DeepLab-LargeFOV Baseline is the same size as of conv4, and including lower level features will not help much as they are not sufficiently discriminative. Besides, we also tried the idea similar to spatial

pyramid pooling where we pool 1×1 global features, 2×2 subregion features, and 4×4 subregion features, and tried both *early fusion* and *late fusion*. However, we observed no improvements. We conjecture that the receptive field of the high level feature map (e.g. fc7) is sufficiently large that sub-region global feature does not help much.

System	bkg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	*mean*
FCN-8s	-	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
Hypercolumn	-	68.7	33.5	69.8	51.3	70.2	81.1	71.9	74.9	23.9	60.6	46.9	72.1	68.3	74.5	72.9	52.6	64.4	45.4	64.9	57.4	62.6
TTI-Zoomout-16	89.8	81.9	35.1	78.2	57.4	56.5	80.5	74.0	79.8	22.4	69.6	53.7	74.0	76.0	76.6	68.8	44.3	70.2	40.2	68.9	55.3	64.4
DL-CRF-LFOV	**92.6**	83.5	36.6	**82.5**	62.3	**66.5**	85.4	78.5	83.7	**30.4**	72.9	**60.4**	**78.5**	75.5	**82.1**	**79.7**	**58.2**	**82.0**	48.8	73.7	63.3	**70.3**
DL-LFOV Base.[5]	92.3	82.6	36.1	76.1	59.3	62.3	81.6	79.5	81.4	28.1	70.0	53.0	73.2	70.6	78.8	78.6	51.9	77.4	45.5	71.7	62.6	67.3
ParseNet[6]	92.4	**84.1**	**37.0**	77.0	**62.8**	64.0	**85.8**	**79.7**	**83.7**	27.7	**74.8**	57.6	77.1	**78.3**	81.0	78.2	52.6	80.4	**49.9**	**75.7**	**65.0**	69.8

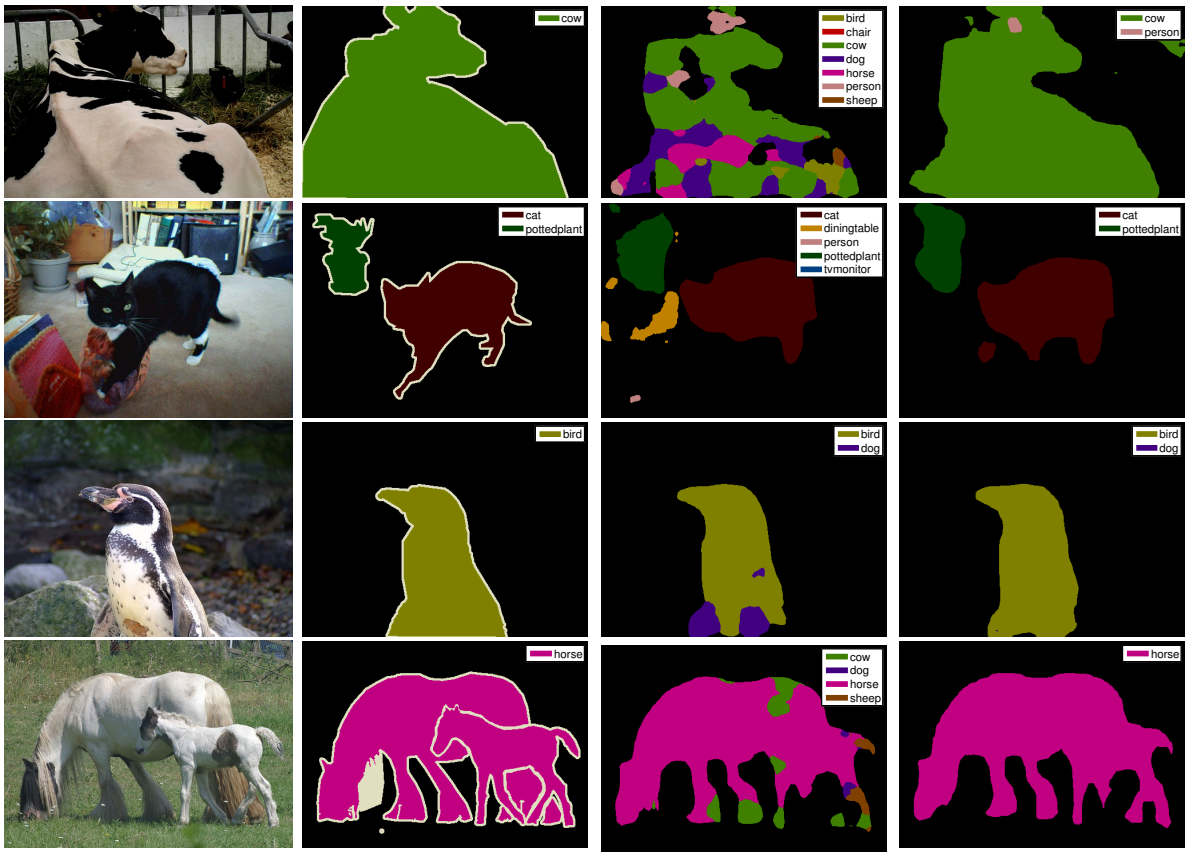
Table 4.6: **PASCAL VOC2012 test Segmentation results.**

Finally, we test two models, DeepLab-LargeFOV Baseline and ParseNet, on the VOC2012 test set. ParseNet is DeepLab-LargeFOV Baseline plus global context. As shown in Table 4.6, we can see that our baseline result is already higher than many existing methods due to proper fine-tuning. By adding the global context feature, we achieve performance that is within the standard deviation of the DeepLab-CRF-LargeFOV (Chen et al., 2014) using fully connect CRF to smooth the outputs and perform better on more than half of categories. Again, our approach is simple to implement and train. Using *late fusion* has almost no extra training/inference cost.

4.4 Related Work

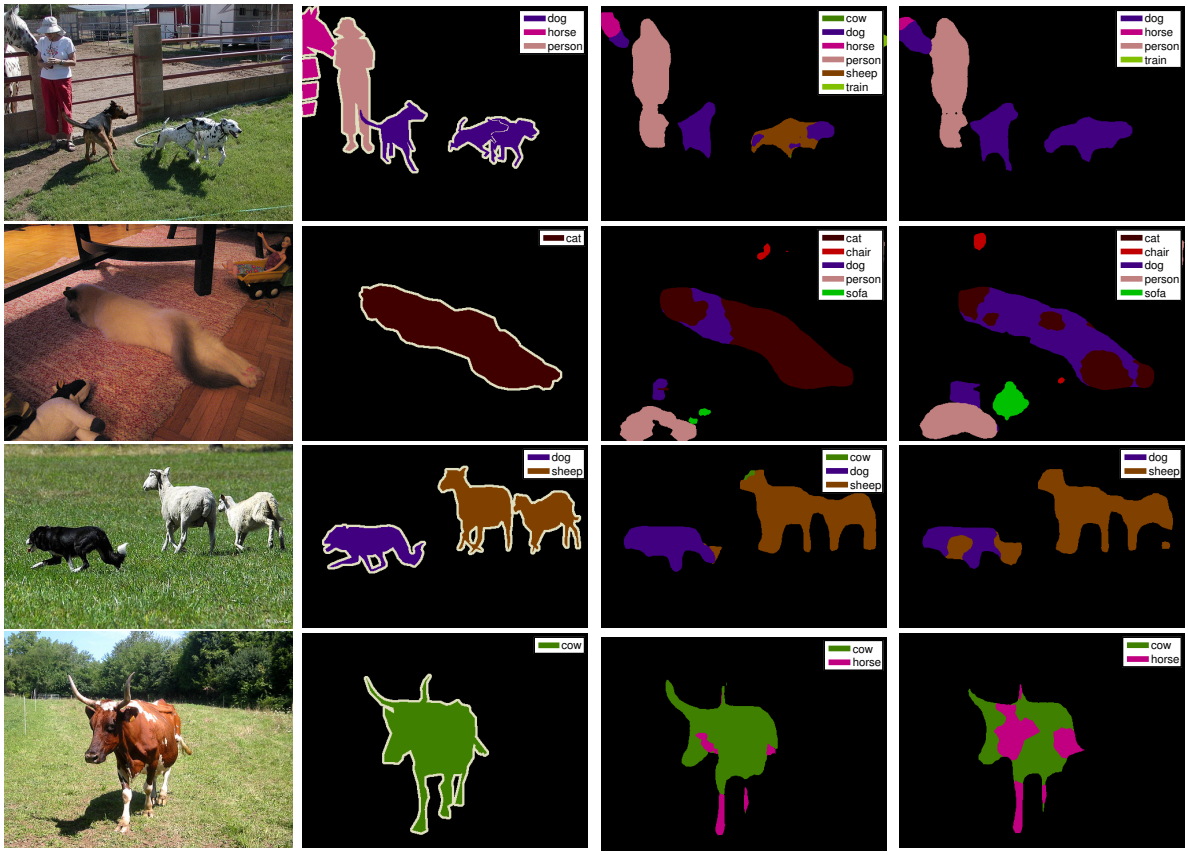
Deep convolutional neural networks (CNN) (Krizhevsky et al., 2012; Szegedy et al., 2014a; Simonyan and Zisserman, 2014) have become powerful tools not only for whole image classification, but also for object detection and semantic segmentation (Girshick

[5] `http://host.robots.ox.ac.uk:8080/anonymous/LGOLRG.html`
[6] `http://host.robots.ox.ac.uk:8080/anonymous/56QLXU.html`



(a) Original Image (b) Ground truth (c) DeepLab-LargeFOV (d) ParseNet

Figure 4.4: **Global context helps for classifying local patches.**



(a) Original Image (b) Ground truth (c) DeepLab-LargeFOV (d) ParseNet

Figure 4.5: **Global context confuse local patch predictions.**

et al., 2014; Szegedy et al., 2014b; Gupta et al., 2014). This success has been attributed to both the large capacity and effective training of the CNN. Following the *proposal + post-classification* scheme (Uijlings et al., 2013), CNNs achieve state-of-the-art results on object detection and segmentation tasks. As a caveat, even though a single pass through the networks used in these systems is approaching or already past video frame rate for individual patches, these approaches require classifying hundreds or thousands of patches per image, and thus are still slow. (He et al., 2014; Long et al., 2014) improve the computation by applying convolution to the whole image once, and then pool features from the final feature map of the network for each region proposal or pixel to achieve comparable or even better results. Yet, these methods still fall short of including whole image context and only classify patches or pixels locally. Our ParseNet is built upon the fully convolutional network architecture (Long et al., 2014) with a strong emphasis on including contextual information in a simple approach.

For semantic segmentation, using context information (Rabinovich et al., 2007; Shotton et al., 2009; Torralba, 2003; Gonfaus et al., 2010) from the whole image can significantly help classifying local patches. (Lucchi et al., 2011) shows that by concatenating features from the whole image to the local patch, the inclusion of post processing (i.e. CRF smoothing) becomes unnecessary because the image level features already encode the smoothness. (Mostajabi et al., 2014) demonstrate that by using the "zoom-out" features, which is a combination of features for each super pixel, region surrounding it, and the whole image, they can achieve impressive performance for the semantic segmentation task. These approaches pool features differently for local patches and the whole image, making it difficult to train the whole system end-to-end. Exploiting the FCN architec-

ture, ParsetNet can directly use global average pooling from the final (or any) feature map to generate the feature of the whole image and use it as context. Experimental results confirm that ParseNet can capture the context of the image and thus improve local patch prediction results.

There is another line of work that attempts to combine graphical models with CNNs to incorporate both context and smoothness priors. (Chen et al., 2014) first uses a FCN to estimate the unary potential, then applies a fully connected CRF to smooth the predictions spatially. As this approach consists of two decoupled stages, it is difficult to train the FCN properly to minimize the final objective of smooth and accurate semantic segments. A more unified and principled approach is to incorporate the structure information during training directly. (Schwing and Urtasun, 2015) propagates the marginals computed from the structured loss to update the network parameters, (Lin et al., 2015) uses piece-wise training to make learning more efficient by adding a few extra piece-wise networks, while (Zheng et al., 2015) convert CRF learning to a recurrent neural network (RNN) and use message passing to do the learning and inference. However, we show that our method can achieve comparable accuracy, with a simpler – hence more robust – structure, while requiring only a small amount of additional training/inference time.

4.5 Conclusion

In this work we presented ParseNet, a simple fully convolutional neural network architecture that allows for direct inclusion of global context for the task of semantic segmentation. We have explicitly demonstrated that relying on the largest receptive field of the FCN network does not provide sufficient global context, and the largest empiri-

cal receptive field is not sufficient to capture global context – modeling global context directly is required. On the PASCAL VOC2012 test set, the segmentation results of ParseNet are within the standard deviation of the DeepLab-LargeFOV-CRF, which suggests that adding a global feature has a similar effect of post processing FCN predictions to a graphical model. As part of developing and analyzing this approach we have provided analysis of many architectural choices for the network, discussed best practices for training, and demonstrated the importance of normalization and learning weights when combining features from multiple layers of the network. By themselves, our practices for training significantly improve the baselines we use before adding global context. The guiding principle in the design of ParseNet is simplicity and robustness of learning. Results are presented on three benchmark datasets, and are state of the art on SiftFlow and PASCAL-Context, and near the state of the art on PASCAL VOC2012. Given the simplicity and ease of training, we find these results very encouraging. In our ongoing work, we are exploring combining our technique with structure training/inference as done in (Schwing and Urtasun, 2015; Lin et al., 2015; Zheng et al., 2015).

CHAPTER 5: Object Detection from Video

> The abstract analysis of the world by mathematics and physics rests on the concepts of space and time.
>
> – James J. Gibson, *Reasons for Realism*

5.1 Introduction

Although we have witnessed revolutionary breakthroughs (Girshick et al., 2014; Girshick, 2015; Ren et al., 2015; Redmon et al., 2015; Liu et al., 2015a) for localizing objects from static images brought by deep convolutional neural networks together with millions of static training images, little is exploited in the temporal domain from videos. People do not acquire the full visual ability by staring at millions of static internet images, we rather move around all the time and recognize objects dynamically. In the same way, computer vision algorithms must evolve to deal with a dynamic environment and be able to handle both space and time.

We propose an important yet less explored problem – object detection from video (VID). VID could utilize the temporal consistency information to solve some of the difficulties, such as motion blur and occlusion, that are hard to solve from static images. Besides, there are a lot of unsolved problems in VID as well. For example, could we detect and track multiple objects consistently and robustly for very long time? Are there unified frameworks that can integrate detection and tracking together and train

both end-to-end? What dataset do we need to tackle this problem? What are the right metrics to measure the progress for such problems?

In this work, we have two major contributions. The first one is that we have constructed a large-scale video dataset. The dataset has 30 categories, which are a subset of the 200 ILSVRC object detection categories; and it has 3862 training, 555 validation, and 1861 testing snippets with more than 2 million bounding boxes fully labeled for all objects in the snippets. Compared to many existing activity recognition datasets (Laptev and Caputo, 2005; Soomro et al., 2012; Karpathy et al., 2014; Fabian Caba Heilbron and Niebles, 2015; Over et al., 2015), creating a video dataset with bounding boxes fully labeled is much more difficult. Compared to a similar dataset – Youtube-Objects dataset (Prest et al., 2012), which has 10 object classes, 155 videos, and 6975 bounding box annotated in 6087 frames, our dataset is much bigger and more challenge. To the best of our knowledge, it is the largest dataset of its kind and it serves as a new benchmark dataset in ILSVRC (Russakovsky et al., 2015) since 2015. We look forward to new methods which can detect and track objects efficiently for very long time within unified frameworks.

The second contribution is that we have defined two metrics to measure the performance of methods on this problem. The main metric we used is exactly the same as the metric used in evaluating object detection methods from static images (Russakovsky et al., 2015). In specific, we treated frames in a snippet as individual images, and compute mean average precision (mAP) based on per frame evaluation results. This metric can measure the accuracy of object detection methods in videos, but it lacks the temporal consistency – an important property of this new problem. Thus we also proposed a new

auxiliary metric which shifts from object centric to tracklet[1] centric. A detected tracklet is a true positive if the tracklet IoU overlap with a ground truth tracklet is more than a defined threshold (e.g. 0.5). To compute tracklet IoU overlap, we need to first greedily match a detected bounding box to a ground truth box and evaluate if it is a true positive or not[2]; and then compute the percentage of the number of correctly detected ground truth boxes among the union of both detected boxes and ground truth boxes. A closely related metric is the MOTA metric (Bernardin and Stiefelhagen, 2008) used in multiple object tracking (MOT) problem (Leal-Taixé et al., 2015). Unlike MOTA that counts the number of false negative objects, the number of false positive detections, and the number of detections with switching ID, our metric rather counts the number of true and false positive tracklets – a closer mimic of our main metric. A recent paper (Zhang and Wang, 2016) introduced a stability error metric for the VID problem, we think we can also incorporate such metric into our auxiliary metric by using multiple box and tracklet IoU overlap thresholds. We leave it for future work.

5.2 Data collection

As we have learned through many years of experience from ILSVRC (Russakovsky et al., 2015), high quality large scale data is a key to the recent success in recognizing or localizing objects from static images. Creating a benchmark dataset for a specific problem is a powerful thing and enables researchers to develop new methods to solve the problem and monitor the progress. For example, we saw that the whole community

[1] A tracklet is a collection of detected bounding boxes from many frames with same ID.

[2] This is achieved by computing the bounding box IoU overlap between a detected box and a ground truth box. A detected box is a true positive if the box IoU overlap is larger than a threshold (i.e. 0.5).

was making rapid progress within the last few years on ILSVRC benchmark datasets for three core computer vision problems: object classification, object localization, and object detection. What is more exciting, because the large scale characteristics of the ILSVRC datasets, methods developed through the years not only have improved the accuracy of these tasks, but also have evolved to be fast and have unified frameworks.

Since we aim to solve the problem of detecting and tracking multiple objects of multiple classes simultaneously within a video, a first important step is to build a large scale benchmark video dataset with high quality fully labeled bounding boxes. There already exist many datasets (Laptev and Caputo, 2005; Soomro et al., 2012; Karpathy et al., 2014; Fabian Caba Heilbron and Niebles, 2015; Over et al., 2015) on activity recognition. However, activity recognition is still a classification problem, albeit over all the frames of a video. To take it a step further, we have collected a large scale video dataset with bounding boxes fully labeled for all categories which appeared in each video, and hope researchers can come up with methods to detect and track objects simultaneously in videos. If we could do it accurately in real-time from videos, it will become useful to many more real applications, such as mobile devices or self driving cars. We also believe that it is a building block for high level understanding of videos.

5.2.1 Define the categories

To construct a dataset, the first step is to define interesting categories for the problem. We select the categories from the 200 object detection categories from ILSVRC. The categories are carefully chosen based on factors such as movement type, level of video clutterness, average number of object instances, etc. For example, we only select objects

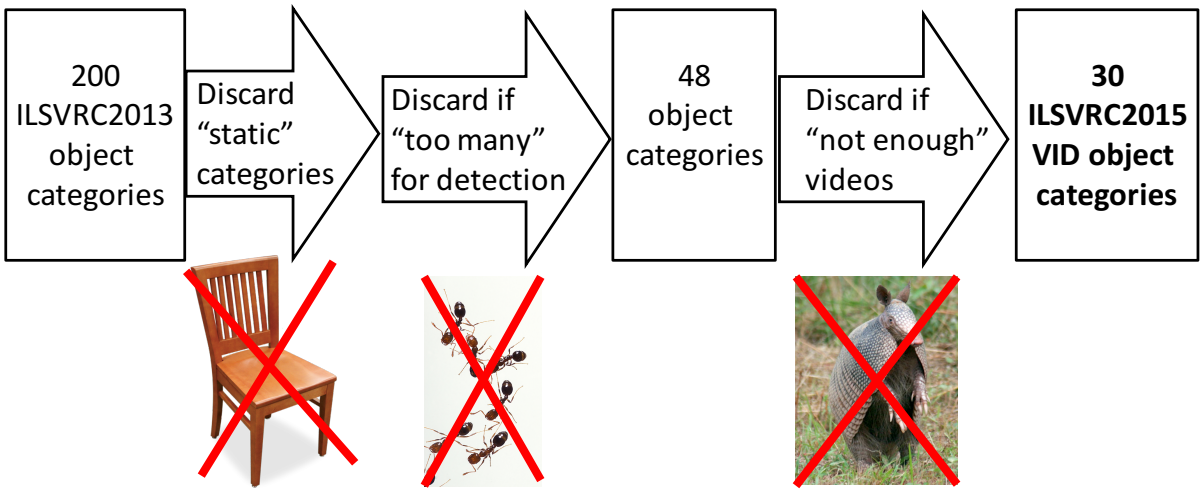


Figure 5.1: **Define video object categories.** We select the categories from the 200 object detection categories from ILSVRC. We discard static objects, crowded objects, and abnormal objects, which eventually gives us 30 categories.

which can move by themselves (e.g. car) instead of static objects (e.g. chair). We also discard objects which usually appear in a group (e.g. ant) because it is hard and expensive to annotate all the object instances, and remove objects which do not have enough videos from the web. Figure 5.1 shows the selection process, and it eventually gives us 30 video object categories. Compared to a similar dataset, Youtube-Objects (Prest et al., 2012), which only contains 10 categories, we have significantly more categories.

5.2.2 Curate the snippets

After defining the object categories, we need to collect video snippets which contain these object categories. A naive way would be searching on Youtube using the keywords related to the categories, which end up having too many random videos with non-relevant objects. Instead, for each object category, we first use Freebase[3] to manually search all the unique ids related to each of the category. Then, we curated most of the videos

[3]It is now deprecated. More information can be found at: `https://developers.google.com/freebase/` .

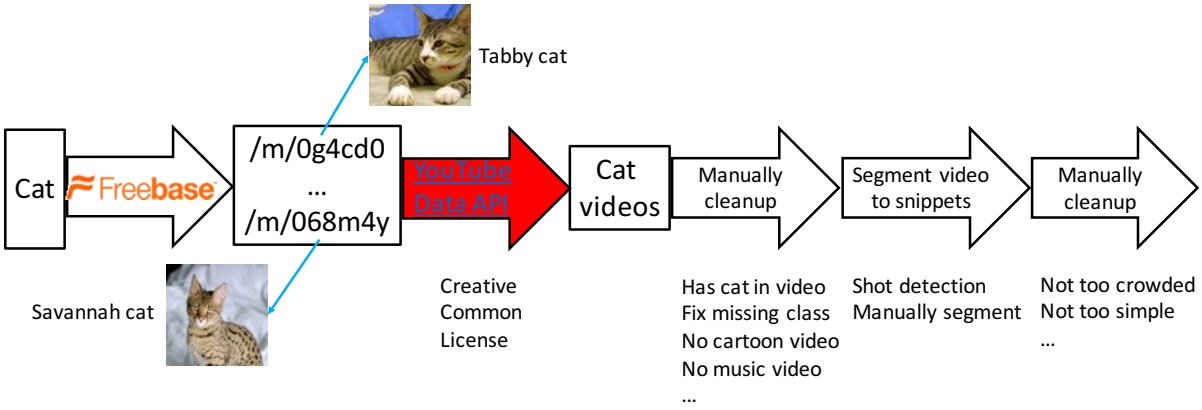


Figure 5.2: **Curate video snippets for each category.** The figure describes the key steps for collecting the final video snippets for cat.

using the YouTube Data API (Google, 2015) and manually selected or verified so that each snippet contains the correct categories. Since most of the videos on Youtube are usually composed of multiple scenes, we have to either use an automatic shot boundary detector (Mathe, 2015) to segment a video into many snippets, or manually segment it, with a follow up step to manually verify that each snippet contains the object categories. Besides, we also manually filter out snippets where objects are too crowded or are mostly stationary. Figure 5.2 describes the steps on how we collect the snippets for each category (e.g. cat).

5.2.3 Collect bounding boxes

With the above selection process, each snippet now has its associated object categories. Then we need to annotate every objects for all the categories in a snippet. We could annotate every single frame individually as how we annotate static images from the ILSVRC object detection task; but it is unavoidably time-consuming and expensive. It is easy to observe that there is a lot of redundancy within a snippet. We could utilize

the temporal consistency information within a snippet to help annotate objects more efficiently. In particular, we used a tool, VATIC (Vondrick et al., 2013)[4], to draw bounding boxes for objects in a video. We made a few minor modifications to the tool. With such tool, we can draw boxes for every few (e.g. 5) frames and get the rest boxes automatically either by simple linear interpolation or by tracking, which significantly boost the annotation speed. Figure 5.3 shows the GUI of our bounding box annotation tool. Code is available at: `https://github.com/weiliu89/vatic` .

5.2.4 Dataset statistics

In total, we have collected 6,258 snippets with more than 2 million of manually labeled bounding boxes. We split the dataset into 3862 snippets for training, 555 snippets for validation, and 1861 snippets reserved for testing. Ground truth tracklet annotations are released for the training and validation snippets. We kept the annotations for testing snippets private, which were used to evaluate methods during annually competition held in conjunction with ILSVRC. As far as we know, it is the first dataset of such scale with bounding boxes fully annotated for such a variety of object categories. It was added as a new challenge since ILSVRC2015. With such a dataset, we can explore many problems that are unique for localizing objects from video.

[4] `http://web.mit.edu/vondrick/vatic/`

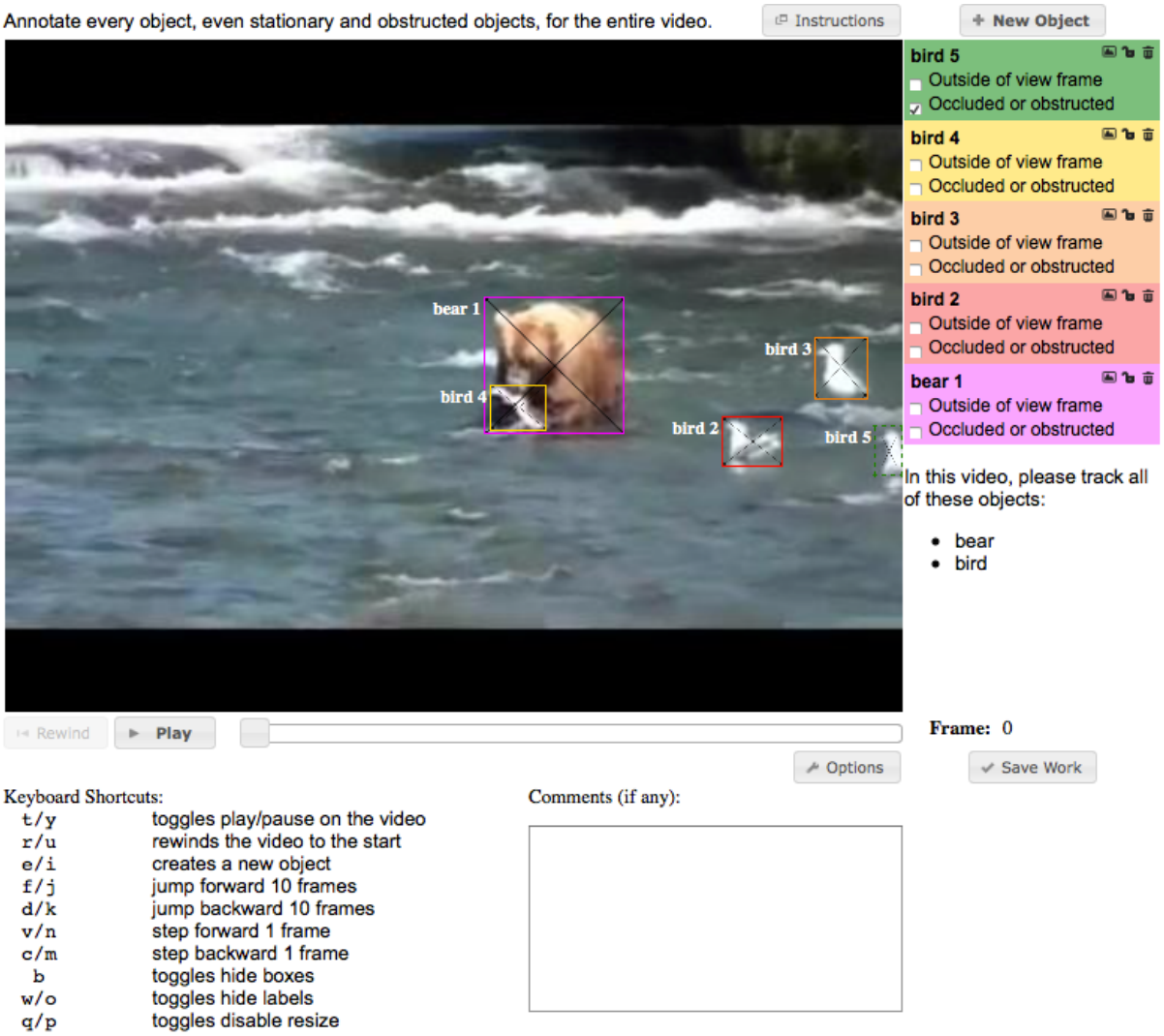


Figure 5.3: **Bounding box annotation GUI.** Our annotation system is a slightly modified VATIC. We asked annotators to annotate every objects appeared in a video.

5.3 Evaluation metric

5.3.1 Challenge

A key challenge of localization in the video, as shown in Fig. 5.4, is that objects may look completely different from two frames due to viewpoint change, illumination variation, complex dynamics, motion blur or occlusion. If we naively detect objects in each frame, we may pinpoint an object when it is clear and upfront without occlusion but

not able to capture the heavily occluded or blurred one. Unlike multi-object tracking, we do not know the number of objects in a snippet and do not know when they appear or disappear. To monitor the progress on this new benchmark dataset and problem, we have defined two metrics.

Figure 5.4: **Challenges of object detection from videos.** The elephant in the left image is relatively easy to detect. The middle image introduces illumination change and motion blur, making it harder to localize. The same elephant (marked in red) is heavily occluded in the right image, and is very hard to detect if the system do not reason with motion information from the video.

5.3.2 Main metric

A straightforward way to measure the performance is to evaluate it as an object detection problem in static images. In other words, we treat each frame in a snippet as an individual image. A method detect objects of multiple categories from each frame; and an average precision (AP) is computed for each object category across all frames from all snippets. The exact metric protocol is defined as follows:

- Algorithm outputs a list of bounding box detections with confidences.
- A detection is considered correct if intersection over union (IoU) overlap with ground truth $>$ threshold (0.5).

- Evaluated by average precision per object class.

- Winners of challenge is the team that wins the most object categories.

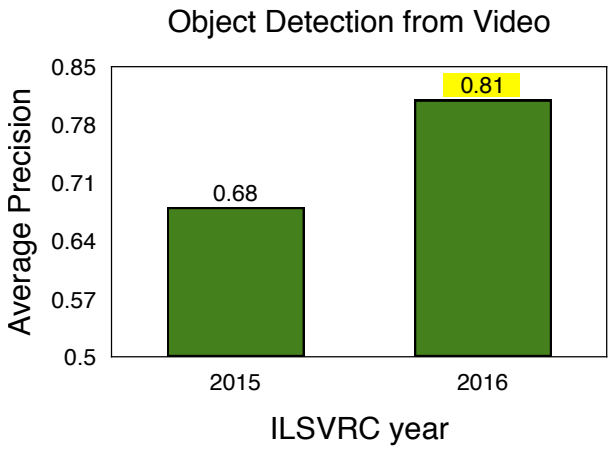


Figure 5.5: **Main metric result in VID over the years.**

Figure 5.5 shows the result on the video dataset using the main metric in last two years. We can see that researchers have made significant improvement on this dataset, thanks to the better object detectors developed through the years. It is noticeable that the performance on the video dataset is much higher than the one from static images. It is not surprising that detection in video is relatively simpler than detection in (random) web-crawled images because there are many redundancy in the videos. Actually these results are obtained by using a combination of object detection and object tracking methods, albeit in a ad-hoc way.

5.3.3 Auxiliary metric with tracking

As described above, the goal of introducing this new dataset and new problem is to encourage the community to develop unified frameworks which can detect and track multiple objects simultaneously. The main metric is a good way to measure the progress,

but misses temporal consistency information, which is critical to this problem. Thus, we have introduced a new auxiliary metric in addition to the main metric to take tracking information into account. In specific, a method not only has to return the detected bounding boxes with confidence for each object category, but also needs to assign a unique tracklet ID which is used to associate detections from different frames within a snippet. The exact evaluation protocol is defined as follows.

- Algorithm outputs a list of bounding box detections with confidences and tracklet ID.
- Tracklets are sorted by the mean confidence.
- A detection is considered correct if intersection over union (IoU) overlap with ground truth $>$ threshold (0.5).
- A tracklet is considered correct if intersection over union (IoU) overlap with ground truth tracklet $>$ threshold (0.25, 0.5, 0.75).
- Evaluated by average precision per class. Final score is an average over different thresholds.
- Winners of challenge is the team that has the highest mean average precision.

To understand the new metric better, let's examine a concrete example. Given a snippet with two giant panda ground truth tracklets, as shown in Figure 5.6a, a method has to return a list of detected tracklets. The first step is compute the score for a tracklet, which is the mean score across all the detected boxes in a tracklet. After this, we sort the

detected tracklets based on the mean score and then evaluate each tracklet in descending order.

(a) Exemplar snippet with ground truth tracklets.

(b) All detections are matched to panda1 based on bounding box IoU overlap, but one (i.e. the last) frame is missed. The tracklet IoU overlap between the detected tracklet and the ground truth tracklet is 0.75.

(c) The first detection is matched to panda1 while the rest two are matched to panda2. The tracklet IoU overlap is 0.67.

(d) Only one out of three frames are detected for panda2. The tracklet IoU overlap is 0.33.

Figure 5.6: **Exemplar snippet with ground truth and detected tracklets.**

For example, Figure 5.6b shows a detected tracklet with the highest mean score (i.e. 0.8). We first evaluate each detected bounding box from the tracklet by assigning it to the most overlapped ground truth if the IoU overlap is larger than a threshold (e.g. 0.5).

In this example, we see that all three detected boxes are assigned to panda1. However, since there are in total 4 ground truth boxes for panda1, and we miss detecting one frame. Thus, the IoU overlap between this tracklet and the ground truth tracklets is $\frac{3}{4} = 0.75$. Take another detected tracklet as shown in Figure 5.6c. The first detection is matched to panda1 while the other two are matched to panda2. We compute the tracklet IoU overlap between the detected tracklet with both ground truth tracklets and take the maximum, which gives us 0.67. Figure 5.6d shows another detected tracklet where only one out of three frames are detected for panda2, and thus the tracklet IoU overlap is 0.33. Based on the computed tracklet IoU overlap, we can check if a detected tracklet is true positive or false positive based on a given tracklet IoU overlap threshold[5]. In particular, we used three overlap thresholds: 0.25, 0.5, and 0.75. Obviously the task becomes much harder with higher overlap threshold because we need to get most of the bounding boxes detected correctly within a tracklet. The final score is the average over these different thresholds.

5.4 Conclusion and Future Work

We have collected the first large scale video dataset with bounding box fully labeled for 30 object categories and have defined two metrics for measuring the performance of methods on this dataset. We have already seen some progress on solving the object detection from video problem by combining state-of-the-art object detectors with tracking (Kang et al., 2016), albeit not in a unified framework. With this benchmark dataset, we firmly believe that the research community can come up with a unified framework

[5]Note that duplicated detected tracklets are considered as false positive.

to combine detection and tracking together and achieve better performance. Besides, we think such dataset can help the tracking community significantly as well. We have seen that it is been used to train a state-of-the-art real time tracker in (Bertinetto et al., 2016), and expect more progress in that direction in future.

In the long term, since it is expensive and not scalable to manually collect a large annotated dataset for thousands or millions of objects, we would also like to explore utilizing weakly supervised or unsupervised information in videos and synthetic dataset to further improve the performance.

BIBLOGRAPHY

Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). Freak: Fast retina keypoint. In *CVPR*.

Alexe, B., Deselaers, T., and Ferrari, V. (2012). Measuring the objectness of image windows. *PAMI*.

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *PAMI*.

Arbeláez, P., Pont-Tuset, J., Barron, J. T., Marques, F., and Malik, J. (2014). Multiscale combinatorial grouping. In *CVPR*.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *ECCV*.

Bell, S., Zitnick, C. L., Bala, K., and Girshick, R. (2016). Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *PAMI*.

Berg, A. C. and Malik, J. (2001). Geometric blur for template matching. In *CVPR*.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*.

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. (2016). Fully-convolutional siamese networks for object tracking. In *ECCV*.

Carreira, J. and Sminchisescu, C. (2012). Cpmc: Automatic object segmentation using constrained parametric min-cuts. *PAMI*.

Chatfield, K., Lempitsky, V. S., Vedaldi, A., and Zisserman, A. (2011). The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv:1412.7062*.

COCO (2016). Common Objects in Context. `http://mscoco.org/dataset/#detections-leaderboard`. [Online; accessed 25-July-2016].

Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*.

Dollar, P. (2016). Coco api. `https://github.com/pdollar/coco`.

Endres, I. and Hoiem, D. (2010). Category independent object proposals. In *ECCV*.

Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). Scalable object detection using deep neural networks. In *CVPR*.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2014). The pascal visual object classes challenge: A retrospective. *IJCV*.

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

Fabian Caba Heilbron, Victor Escorcia, B. G. and Niebles, J. C. (2015). ActivityNet: A large-scale video benchmark for human activity understanding. In *CVPR*.

Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *CVPR*.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *IJCV*.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*.

Girshick, R. (2015). Fast r-cnn. *arXiv preprint arXiv:1504.08083*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Gonfaus, J. M., Boix, X., Van de Weijer, J., Bagdanov, A. D., Serrat, J., and Gonzalez, J. (2010). Harmony potentials for joint classification and segmentation. In *CVPR*.

Google (2015). YouTube Data API. `https://developers.google.com/youtube/v3/?hl=en`.

Gupta, S., Girshick, R., Arbeláez, P., and Malik, J. (2014). Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*.

Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., and Malik, J. (2011). Semantic contours from inverse detectors. In *ICCV*.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In *CVPR*.

He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

Hoiem, D., Chodpathumwan, Y., and Dai, Q. (2012). Diagnosing error in object detectors. In *ECCV 2012*.

Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. (1990). A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer.

Howard, A. G. (2013). Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *CVPR*.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *MM*.

Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., et al. (2016). T-cnn: Tubelets with convolutional neural networks for object detection from videos. *arXiv preprint arXiv:1604.02532*.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *CVPR*.

Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Laptev, I. and Caputo, B. (2005). Recognition of human actions. `http://www.nada.kth.se/cvap/actions/`.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*.

Leal-Taixé, L., Milan, A., Reid, I., Roth, S., and Schindler, K. (2015). Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). Brisk: Binary robust invariant scalable keypoints. In *ICCV*.

Lin, G., Shen, C., Reid, I., et al. (2015). Efficient piecewise training of deep structured models for semantic segmentation. *arXiv:1504.01013*.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *IJCV*.

Liu, C., Yuen, J., and Torralba, A. (2011). Nonparametric scene parsing via label transfer. *PAMI*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2015a). Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*.

Liu, W., Rabinovich, A., and Berg, A. C. (2015b). Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*.

Long, J., Shelhamer, E., and Darrell, T. (2014). Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *ICCV*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*.

Lucchi, A., Li, Y., Boix, X., Smith, K., and Fua, P. (2011). Are spatial and global constraints really necessary for segmentation? In *ICCV*.

Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*.

Mathe, J. (2015). Automated shot detection software. `https://github.com/johmathe/Shotdetect`.

Mikolajczyk, K. and Schmid, C. (2004). Scale & affine invariant interest point detectors. *IJCV*.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *PAMI*.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. (2005). A comparison of affine region detectors. *IJCV*.

Mostajabi, M., Yadollahpour, P., and Shakhnarovich, G. (2014). Feedforward semantic segmentation with zoom-out features. *arXiv:1412.0774*.

Mottaghi, R., Chen, X., Liu, X., Cho, N.-G., Lee, S.-W., Fidler, S., Urtasun, R., and Yuille, A. (2014). The role of context for object detection and semantic segmentation in the wild. In *CVPR*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML*.

Over, P., Awad, G., Michel, M., Fiscus, J., Kraaij, W., Smeaton, A. F., Quéenot, G., and Ordelman, R. (2015). TRECVID 2015 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID*. NIST, USA.

Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *ECCV*.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*.

Prest, A., Leistner, C., Civera, J., Schmid, C., and Ferrari, V. (2012). Learning object class detectors from weakly annotated video. In *CVPR*.

Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. (2007). Objects in context. In *CVPR*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *ICCV*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

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

Sánchez, J., Perronnin, F., Mensink, T., and Verbeek, J. (2013). Image classification with the fisher vector: Theory and practice. *IJCV*.

Schwing, A. G. and Urtasun, R. (2015). Fully connected deep structured networks. *arXiv:1503.02351*.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops*.

Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2009). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Soomro, K., Zamir, A. R., and Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014a). Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*.

Szegedy, C., Reed, S., Erhan, D., and Anguelov, D. (2014b). Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*.

Torralba, A. (2003). Contextual priming for object detection. *IJCV*.

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

Van De Sande, K., Gevers, T., and Snoek, C. (2010). Evaluating color descriptors for object and scene recognition. *PAMI*.

Van de Sande, K. E., Snoek, C. G., and Smeulders, A. W. (2014). Fisher and vlad with flair. In *CVPR*.

Van Gemert, J. C., Geusebroek, J.-M., Veenman, C. J., and Smeulders, A. W. (2008). Kernel codebooks for scene categorization. In *ECCV*.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *CVPR*.

Vondrick, C., Patterson, D., and Ramanan, D. (2013). Efficiently scaling up crowdsourced video annotation. *IJCV*.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. In *CVPR*.

Wang, X., Han, T. X., and Yan, S. (2009). An hog-lbp human detector with partial occlusion handling. In *ICCV*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV*.

Zhang, H. and Wang, N. (2016). On the stability of video detection and tracking. *arXiv preprint arXiv:1611.06467*.

Zhang, J., Marszałek, M., Lazebnik, S., and Schmid, C. (2007). Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*.

Zhang, L., Lin, L., Liang, X., and He, K. (2016). Is faster r-cnn doing well for pedestrian detection. In *ECCV*.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. (2015). Conditional random fields as recurrent neural networks. *arXiv preprint arXiv:1502.03240*.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2015). Object detectors emerge in deep scene cnns. In *ICLR*.

Zhou, X., Yu, K., Zhang, T., and Huang, T. S. (2010). Image classification using supervector coding of local image descriptors. In *ECCV*.

Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *ECCV*.