ScratchDet: Training Single-Shot Object Detectors from Scratch

Rui Zhu[1,4*], Shifeng Zhang[2*], Xiaobo Wang[1], Longyin Wen[3], Hailin Shi[1†], Liefeng Bo[3], Tao Mei[1]
[1]JD AI Research, China; [2]CASIA, UCAS, China; [3]JD Digits, USA; [4]Sun Yat-sen University, China
{zhurui10,wangxiaobo8,longyin.wen,shihailin,liefeng.bo,tmei}@jd.com, shifeng.zhang@nlpr.ia.ac.cn

Abstract

Current state-of-the-art object objectors are fine-tuned from the off-the-shelf networks pretrained on large-scale classification dataset ImageNet, which incurs some additional problems: 1) The classification and detection have different degrees of sensitivity to translation, resulting in the learning objective bias; 2) The architecture is limited by the classification network, leading to the inconvenience of modification. To cope with these problems, training detectors from scratch is a feasible solution. However, the detectors trained from scratch generally perform worse than the pretrained ones, even suffer from the convergence issue in training. In this paper, we explore to train object detectors from scratch robustly. By analysing the previous work on optimization landscape, we find that one of the overlooked points in current trained-from-scratch detector is the BatchNorm. Resorting to the stable and predictable gradient brought by BatchNorm, detectors can be trained from scratch stably while keeping the favourable performance independent to the network architecture. Taking this advantage, we are able to explore various types of networks for object detection, without suffering from the poor convergence. By extensive experiments and analyses on downsampling factor, we propose the Root-ResNet backbone network, which makes full use of the information from original images. Our ScratchDet achieves the state-of-the-art accuracy on PASCAL VOC 2007, 2012 and MS COCO among all the train-from-scratch detectors and even performs better than several one-stage pretrained methods. Codes will be made publicly available at `https://github.com/KimSoybean/ScratchDet`.

1. Introduction

Object detection has made great progress in the framework of convolutional neural networks (CNNs). The current state-of-the-art detectors are generally fine-tuned from high accuracy classification networks, *e.g.*, VGGNet [36], ResNet [12] and GoogLeNet [37] pretrained on ImageNet [29] dataset. The fine-tuning transfers the classification knowledge learned from the source domain to handle the object detection task. In general, fine-tuning from pretrained networks can achieve better performance than training from scratch.

However, there is no such thing as a free lunch. Fine-tuning pretrained networks to object detection has some critical limitations. On the one hand, the classification and detection tasks have different degrees of sensitivity to translation. The classification task prefers to translation invariance, and thus needs downsampling operations (*e.g.*, max-pooling and convolution with stride 2) for better performance. In contrast, the local texture information is more critical for object detection, making the usage of translation-invariant operations (*e.g.*, downsampling operations) with caution. On the other hand, it is inconvenient to change the architecture of networks (even small changes) in fine-tuning process. If we employ a new architecture, the pretraining should be re-conducted on the large-scale dataset (*e.g.*, ImageNet), requiring high computational cost.

Fortunately, training detectors from scratch is able to eliminate the aforementioned limitations. DSOD [32] is the first to train CNN detectors from scratch, in which the deep supervision plays a critical role. Deep supervision is introduced in DenseNet [13] as the dense layer-wise connection. However, DSOD is also limited by the predefined architecture of DenseNet. If DSOD employs other types of network (*e.g.*, VGGNet and ResNet), the performance decreases dramatically (sometimes even crashes in training). Besides, the currently best performance of trained-from-scratch detectors still remains in a lower place compared with the pretrained ones. Therefore, if we hope to take advantage of training detectors from scratch, it needs to achieve two improvement: (1) free the architecture limitations for any type of network while guarantee the training convergence, (2) give performance as good as pretrained networks (or even better).

To this end, we study the elements that make major impact to the optimization of detector given the randomly initialized network. As pointed out in [30], BatchNorm reparameterizes the optimization problem to make its landscape significantly smoother instead of reducing the internal covariate shift.

*Equally-contributed and this work was done at JD AI Research.
†Corresponding author.

Based on this theory, we assume that the lack of BatchNorm in training detector from scratch is the main reason for poor convergence. Thus, we integrate BatchNorm into both the backbone and detection head subnetworks (Figure 2), and find that BatchNorm helps the detector converge well in any form of network (including VGGNet and ResNet) without pretraining and surpass the accuracy of the pretrained baselines. Thereby, we are free to modify the architecture without restrictions from pretrained models. By taking this advantage, we analyze the performance of the ResNet and VGGNet based SSD[24] detectors with various configurations, and discover that the sampling stride in the first convolution layer has a great impact on detection performance. Based on this point, we redesign the architecture of the detector by introducing a new root block, which keeps the abundant information for detection feature maps and substantially improves the detection accuracy, especially for small objects. We report extensive experiments on PASCAL VOC 2007 [6], PASCAL VOC 2012 [7] and MS COCO [23] datasets, to demonstrate that our ScratchDet performs better than some pretrained based detectors and all the state-of-the-art train-from-scratch detectors, *e.g.*, improving the state-of-the-art mAP by 1.7% on VOC 2007, 1.5% on VOC 2012, and 2.7% of AP on COCO.

The main contributions of this paper are summarized as follows. (1) We present a single-shot object detector trained from scratch, named ScratchDet, which integrates BatchNorm to help the detector converge well from scratch, independent to the type of network. (2) We introduce a new Root-ResNet backbone network based on the new designed root block, which noticeably improves the detection accuracy, especially for small objects. (3) ScratchDet performs favourably against the state-of-the-art train-from-scratch detectors and some pretrained based detectors.

2. Related Work

Object detectors with pretrained network. Most of CNN-based object detectors are fine-tuned from pretrained networks on ImageNet. Generally, they can be divided into two categories: the two-stage and the one-stage approach. The two-stage approach first generates a set of candidate object proposals, and then predicts the accurate object regions and the corresponding class labels. With the gradual improvements from Faster R-CNN [28], R-FCN [4], FPN [21] to Mask R-CNN [11], the two-stage methods achieve top performance on several challenging datasets, *e.g.*, PASCAL VOC and MS COCO. Recent developments of two-stage approach focus on redesigning architecture diagram [20], convolution form [5], re-ranking detection scores [3], using contextual reasoning [1] and exploiting multiple layers for prediction [19].

Pursuing high efficiency, the one-stage approach attracts much attention in recent years, which simultaneously regresses the object locations and sizes, and the corresponding class labels. OverFeat [31] is one of the first one-stage detectors and since then, several other methods have been proposed, such as YOLO [26, 27] and SSD [24]. Recent researches on one-stage approach focus on enriching features for detection [8], designing different architecture [39] and addressing class imbalance issue [41, 22, 40].

Train-from-scratch object detectors. DSOD [32] first trains the one-stage object detector from scratch and presents a series of principles to produce good performance. GRP-DSOD [33] improves the DSOD algorithm by applying the Gated Recurrent Feature Pyramid. These two methods focus on deep supervision of DenseNet but lose sight of the effect of BatchNorm on optimization and the flexibility of network architecture for training detectors from scratch.

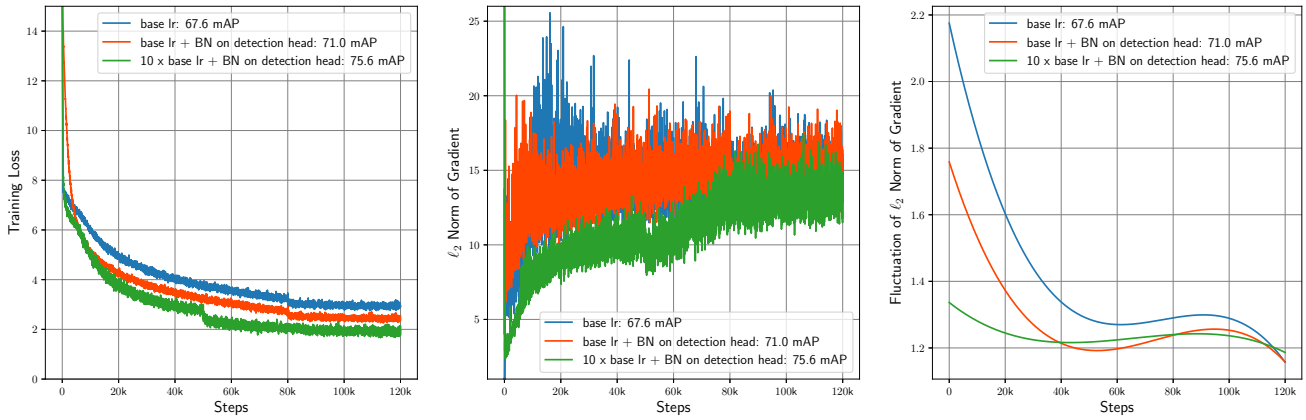
Batch normalization. BatchNorm[14] addresses the internal covariate shift problem by normalizing layer inputs, which makes using large learning rate to accelerate network training feasible. More recently, Santurkar *et al.* [30] provides both empirical demonstration and theoretical justification for the explanation that BatchNorm makes the optimization landscape significantly smoother instead of reducing internal covariate shift.

3. ScratchDet

In this section, we first study the effectiveness of BatchNorm for training SSD from scratch. Then, we redesign the backbone network by analyzing the detection performance of the ResNet and VGGNet based SSD.

3.1. BatchNorm for train-from-scratch

Without losing generality, we consider to apply BatchNorm in SSD which is the most common framework of one stage. SSD is formed by the backbone subnetwork (*e.g.*, truncated VGGNet-16 with several additional convolution blocks) and the detection head subnetwork (*i.e.*, the prediction blocks after each detection layer, which consists of one 3×3 bounding box regression convolution layer and one 3×3 class label prediction convolution layer). Notice that there is no BatchNorm in the original SSD framework. Motivated by recent work [30], we believe that using BatchNorm is helpful to train SSD from scratch. BatchNorm makes the optimization landscape significantly smoother, inducing a more predictable and stable behaviour of the gradients to allow for larger searching space and faster convergence. DSOD successfully trains detectors from scratch, however, it attributes the results to deep supervision of DenseNet without emphasizing the effect of BatchNorm. We believe that it is necessary to study the impact of BatchNorm on training detectors from scratch. To verify our argument, we train SSD from scratch using batch size 128 without BatchNorm as our baseline. As listed in the first column of Table 1, our baseline produces 67.6% mAP on VOC 2007 `test` set.



(a) Loss Value (b) L2 Norm of Gradient (c) Fluctuation of L2 Norm of Gradient

Figure 1. Optimization landscape analysis. (a) The training loss value. (b) L2 Norm of gradient. (c) Fluctuation of L2 Norm of gradient (smoothed). Blue curve is the original SSD, red and green curves represent the SSD trained with BatchNorm in head subnetwork with $1\times$ and $10\times$ base learning rate, respectively. The BatchNorm makes smoother optimization landscape and has more stable gradients (red *v.s* blue). With this advantage, we are able to set larger learning rate (green) to search larger space and converge faster, and thus better solution.

BatchNorm in the backbone subnetwork. We add BatchNorm in each convolution layer in the backbone subnetwork and then train it from scratch. As shown in Table 1, using BatchNorm in the backbone network improves 5.2% of mAP. More importantly, adding BatchNorm in the backbone network makes the optimization landscape significantly smoother. Thus, we can use larger learning rates (0.01 and 0.05) to further improve the performance (*i.e.*, mAP is improved from 72.8% to 77.8% and 78.0%). Both of them outperform SSD fine-tuned from the pretrained VGG-16 model (77.2% [24]). These results indicate that adding BatchNorm in the backbone subnetwork is one of the critical issues to train SSD from scratch.

BatchNorm in the detection head subnetwork. To analyze the effect of BatchNorm in the detection head subnetwork, we plot the training loss value, L2 Norm of gradient, and fluctuation of L2 Norm of gradient *v.s* training steps. As shown by the blue curve in Figure 1(b) and 1(c), training SSD from scratch with default learning rate 0.001 has a large fluctuation of L2 norm of gradient, especially in the initial phase of training, which makes the loss value suddenly change and converge to a bad local minima (*i.e.*, relatively high loss at the end of training process in Figure 1(a) and bad detection result 67.6% mAP). These results are useful to explain the phenomenon that using large learning rate to train SSD with the original architecture from scratch or pretrained networks usually leads to gradient explosion, poor stability and weak prediction of gradients (see Table 1) .

In contrast, integrating BatchNorm in the detection head subnetwork makes the loss landscape smoother (see red curves in Figure 1), which improves mAP from 67.6% to 71.0% (listed in Table 1). The smooth landscape allows us to set larger learning rate, which brings about larger searching space and faster convergence (see Figure 1(a) and 1(c)). As a result, the mAP improves from 71.0% to 75.6%. Besides, with BatchNorm, larger learning rate is also helpful to jump out of the bad local minima and produce stable gradients (green curve in Figure 1(b) and 1(c)).

BatchNorm in the whole network. We also study the performance of the detector using BatchNorm in both the backbone and detection head subnetworks. After using BatchNorm in the whole network of detector, we are able to use a larger base learning rate (0.05) to train the detector from scratch, which produces 1.5% higher mAP comparing to the detector initialized with the pretrained VGG-16 backbone (78.7% *v.s* 77.2%). Please see Table 1 for more details.

3.2. Backbone Network

As described above, we train SSD with BatchNorm from scratch and achieve better accuracy than the pretrained SSD. This encourages us to train detector from scratch while keeping the performance independent to the network architecture. By taking this advantage, we are able to explore various types of network for the object detection task.

Performance analysis of ResNet and VGGNet. The truncated VGG-16 and ResNet-101 are two popular backbone networks used in SSD (a brief structure overview in Figure 2). In general, ResNet-101 produces better classification results than VGG-16 (*e.g.*, 5.99% *v.s* 8.68%, 2.69% top-5 classification error lower on ImageNet). However, as indicated in DSSD [8], the VGG-16 based SSD performs favourably than the ResNet-101 based SSD with relatively small input size (*e.g.*, 300×300) on PASCAL VOC. We argue that this phenomenon is caused by the downsampling operation in the first convolution layer (*i.e.*, conv1_x with stride 2) of ResNet-101. This operation significantly affects the de-

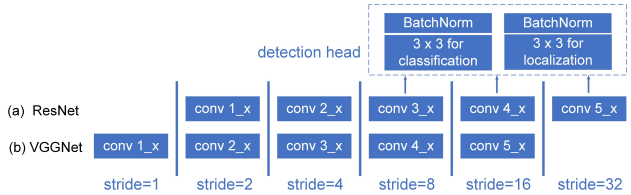


Figure 2. Brief overview of SSD based on VGG-16 and ResNet-101. The BatchNorm is covered for clearness. As shown in Figure 3 and Table 2, the first stride 2 of ResNet makes worse performance on PASCAL VOC with small input size.

tection accuracy, especially for small objects (see Table 2). After we remove the downsampling operation in conv1_x of ResNet-18 to form ResNet-18-B in Figure 3(c), the detection performance improves by a big margin from 73.1% to 77.6% mAP. We also remove the second downsampling operation to form ResNet-18-A in Figure 3(b), whose improvement is relatively small. In summary, the downsampling operation in the first convolution layer has a bad impact on the detection accuracy, especially for small objects.

Backbone network redesign for object detection. To overcome the disadvantages of ResNet based backbone network for object detection while retaining its powerful classification ability, we design a new architecture, named Root-ResNet, which is an improvement of the truncated ResNet in the original SSD detector, shown in Figure 3(d). We remove the downsampling operation in the first conv layer and replace the 7×7 convolution kernel by a stack of 3×3 convolution filters (similar as the stem block in DSOD[32], but denoted as the root block due to the large influence from the first stride). With abundant inputs, Root-ResNet is able to exploit more local information from the image, so as to extract powerful features for small object detection. Furthermore, we replace the four convolution blocks (added by SSD to extract the feature maps with different scales) with four residual blocks to the end of the Root-ResNet. Each residual block is formed by two branches. One branch is a 1×1 convolution layer with stride 2 and the other one consists of a 3×3 convolution layer with stride 2 and a 3×3 convolution layer with stride 1. The number of output channels in each convolution layer is set to 128. These residual blocks bring efficiency in parameters and computation without performance dropout.

4. Experiment

We conduct several experiments on the PASCAL VOC and MS COCO datasets, including 20 and 80 object classes. The proposed ScratchDet is implemented in Caffe library [15] and all the codes and the trained models will be made publicly available.

4.1. Training details

All models are trained from scratch using SGD with 0.0005 weight decay and 0.9 momentum on 4 NVIDIA

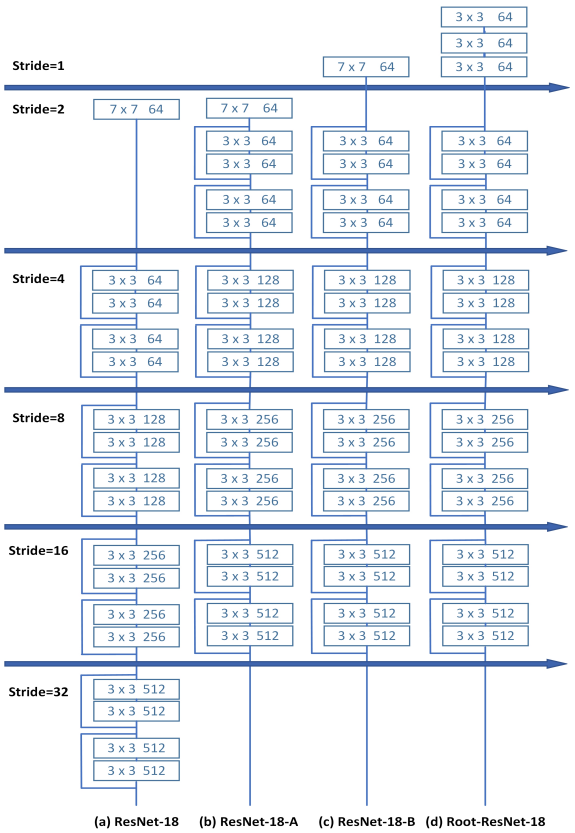


Figure 3. Illustration of networks in Section 4.2.2. (a) ResNet-18: original structure. (b) ResNet-18-A: removing the first max-pooling layer. (c) ResNet-18-B: changing the stride size in the first conv layer from 2 to 1. (d) Root-ResNet-18: replacing the 7×7 conv layer with three stacked 3×3 conv layers in ResNet-18-B. The corresponding mAPs on PASCAL 2007 test (training on "07+12" from scratch) are 73.1%, 75.3%, 77.6% and 78.5%, respectively. Notably, for a fairy comparison, no matter how we modify the structure, the spatial sizes of our selected detection layers are the same as SSD300 and DSOD300 (*i.e.*, 38×38, 19×19, 10×10, 5×5, 3×3, 1×1).

Tesla P40 GPUs. For a fair comparison, we use the same training settings as the original SSD, including data augmentation, anchor settings and loss function. We remove the L2 normalization [25]. Notably, all experiments select the detection layers with the fixed spatial size same as SSD300 and DSOD300, *i.e.*, do not use larger-size feature maps for detection. Following DSOD, we use a relatively large batch size 128 to train our ScratchDet from scratch, in order to ensure the stable statistical results of BatchNorm in training phase. Meanwhile, we use the default batch size 32 for the pretrained model based SSD (We also try 128 batch size for the pretrained model, but the performance has not improved).

Notably, we use the "Root-ResNet-18" redesigned from ResNet-18 as the backbone network in the model analysis by considering the computational cost in experiments. Whereas, in comparison with the state-of-the-art detectors, we use

Table 1. Analysis of BatchNorm and learning rate for SSD trained from scratch on VOC 2007 `test` set. All the networks are based on the truncated VGG-16 backbone network. The best performance (78.7% mAP) is achieved when three conditions are satisfied: (1) BatchNorm in backbone and head, (2) non pretraining, (3) larger learning rate. "NAN" indicates that the training is non-convergent.

Component	lr 0.001						lr 0.01						lr 0.05					
pretraining					✓	✓					✓	✓					✓	✓
BN in backbone			✓	✓		✓			✓	✓		✓			✓	✓		✓
BN in head		✓		✓	✓	✓		✓		✓	✓	✓		✓		✓	✓	✓
mAP (%)	67.6	71.0	72.8	71.8	77.1	77.6	NAN	75.6	77.8	77.3	76.9	78.2	NAN	NAN	78.0	78.7	NAN	75.5

a deeper backbone network "Root-ResNet-34" for better performance. All the parameters in our ScratchDet are initialized by the "xavier" method [10]. Besides, all the models are trained with the 300×300 input size and we believe that the accuracy of ScratchDet can be further improved using larger input size.

4.2. PASCAL VOC 2007

For PASCAL VOC 2007, all models are trained on the VOC 2007 and VOC 2012 `trainval` sets ($16,551$ images), and tested on the VOC 2007 `test` set ($4,952$ images). We use the same settings and configurations except for some specified changes of model components.

4.2.1 Analysis of BatchNorm

We construct several variants of the original SSD and evaluate them on VOC 2007 to demonstrate the effectiveness of BatchNorm in training SSD from scratch, shown in Table 1.

Without BatchNorm. We train the original SSD from scratch with the batch size 128. All the other settings are the same as that in [24]. As shown in the first column of Table 1, we get 67.6% mAP, which is 9.6% worse than the detector initialized by the pretrained classification network (*i.e.*, 77.2%). In addition, due to the unstable gradient and unsmooth optimization landscape, the training is able to successfully converge only with the learning rate 0.001 and goes to a bad local minima (see blue curves in Figure 1). As shown in Table 1, if we use larger learning rates (0.01 and 0.05), the training process will not converge.

BatchNorm in the backbone subnetwork. BatchNorm is a widely used to enable fast and stable training of deep neural networks. To validate the effectiveness of BatchNorm in the backbone subnetwork, we add the BatchNorm operation to each convolution layer in the truncated VGG-16 network, denoted as VGG-16-BN, and train the VGG-16-BN model based SSD from scratch. As shown in Table 1, using BatchNorm in the backbone network with relative large learning rate (0.05) improves mAP from 67.6% to 78.0%.

BatchNorm in the detection head subnetwork. We also study the effectiveness of BatchNorm in the detection head subnetwork. As described before, the detection head subnetwork in SSD is used to predict the locations, sizes and class labels of objects. The original SSD method [24] do not use BatchNorm in detection head subnetwork. As presented in Table 1, we find that using BatchNorm only on the detection head subnetwork improves 3.4% mAP from 67.6% to 71.0%. After using the 10 times larger base learning rate 0.01, the performance can be further improved from 71.0% to 75.6%. This noticeable improvement (8.0%) demonstrates the importance of using BatchNorm in the detection head subnetwork.

BatchNorm in the whole network. We use BatchNorm on every convolution layer in SSD and train it from scratch with three different base learning rates (0.001, 0.01 and 0.05). For the 0.001 and 0.01 base learning rates, we achieve 71.8% and 77.3% mAPs, respectively. When we use the largest learning rate 0.05, the performance will be further improved by 1.4% mAP to 78.7%, which outperforms the pretrained network based SSD detector (78.7% *v.s* 77.2%). These results indicate that using BatchNorm on each convolution layers in SSD is critical to train it from scratch.

BatchNorm for the pretrained network. To validate the effect of BatchNorm for SSD finetuning from pretrained networks, we construct a variant of the original SSD, *i.e.*, adding the BatchNorm operation to every convolution layer. The layers in backbone network are initialized by the pretrained VGG-16-BN model from ImageNet, which is converted from the PyTorch official model. As shown in Table 1, we observe that the best result achieves 78.2% with learning rate 0.01. Comparing to the original SSD fine-tuned from the pretrained network, BatchNorm improves only 1.0% mAP (77.2% *v.s* 78.2%) of the detector, which is rather small compared to the improvement of the trained-from-scratch detector (*i.e.*, 11.1% mAP improvement from 67.6% to 78.7%)[1]. We would also like to emphasize that ScratchDet produces better performance than the BatchNorm based SSD trained from the pretrained network (*i.e.*, 78.7% *v.s* 78.2%). The results demonstrate that BatchNorm is more critical for SSD trained from scratch than fine-tuned from pretrained models.

BatchNorm in DSOD. DSOD attributes its success to deep supervision of DenseNet and ignores the effect of BatchNorm. After removing all BatchNorm layers in DSOD, the mAP drops 6.2% from 77.7% to 71.5% on VOC 2007. Thus, we argue BatchNorm rather than deep supervision is the key to train detectors form scratch and experiments in Table 1 validate this point. Besides, training VGG16-based Faster R-CNN without BatchNorm from scratch cannot converge

[1] we also try the batch size 128 with default settings of SSD, producing 78.2% mAP for VGG-16-BN and 76.8% mAP for VGG-16 without improvement.

in the DSOD paper, but with BatchNorm it can converge successfully to 67.2% mAP, although it is still lower than the pretrained one (73.2% mAP).

4.2.2 Analysis of the backbone subnetwork.

We analyze the pros and cons of the ResNet and VGGNet based SSD detectors and redesign the backbone network, called Root-ResNet. Specifically, all the models are designed based on the ResNet-18 backbone network in experiments. We also use BatchNorm in the detection head subnetwork. In the training phase, the learning rate is set to 0.05 for the first $45k$ iterations, and is divided by 10 successively for another $30k$, $20k$ and $5k$ iterations, respectively. As shown in Table 2, training SSD from scratch based on ResNet-18 only produces 73.1% mAP. We analyze the reasons as follows.

Kernel size in the first layer. In contrast to VGG16, the first convolution layer in ResNet-18 uses relatively large kernel size 7×7 with stride 2. We aim to explore the effect of the kernel size of the first convolution layer on the detector trained from scratch. As shown in the first two rows of Table 2, the kernel size of convolution layer has no impact on the performance (*i.e.*, 73.1% for 7×7 *v.s* 73.2% for 3×3). Using smaller kernel size 3×3 produces a slightly better results with faster speed. The same conclusion can be obtained when we set the stride size of the first convolution layer to 1 without downsampling, see the fifth and the sixth row of Table 2 for more details.

Downsampling in the first layer. Compared to VGGNet, ResNet-18 uses downsampling on the first convolution layer, leading to considerable local information loss, which greatly impacts the detection performance, especially for small objects. As shown in Table 2, after removing the downsampling operation in the first layer (*i.e.*, ResNet-18-B in Figure 3), we can improve 4.5% and 4.6% mAPs for the 7×7 and 3×3 kernel sizes, respectively. When we only remove the second downsampling operation and keep the first stride = 2 (*i.e.*, ResNet-18-A in Figure 3), the performance achieves 75.3% mAP, 2.3% lower than modifying the first layer (77.6% mAP). These results demonstrate that the downsampling operation in the first convolution layer is the obstacle for good results. We need to remove this operation when training ResNet based SSD from scratch.

Number of layers in the root block. Inspired by DSOD and GoogLeNet-V3 [38], we use several convolution layers with kernel size 3×3 to replace the 7×7 convolution layers (*i.e.*, Root-ResNet-18 in Figure 3). Here, we study the impact of number of stacked convolution layers in the root block on the detection performance in Table 2. As the number of convolution layers increasing from 1 to 3, the mAP scores are improved from 77.8% to 78.5%. However, the accuracy decreases as the number of stacked layers becoming larger than 3. We believe that three 3×3 convolution layers in the root block are enough to learn the information from raw images, and adding more 3×3 layers cannot boost the accuracy any more. Empirically, we use three 3×3 convolution layers for detection task on PASCAL VOC 2007, 2012 and MS COCO datasets with 300×300 input size.

Table 2. Analysis of backbone network for SSD trained from scratch on VOC 2007 `test` set. All models are based on the ResNet-18 backbone. FPS is measured on one Tesla P40 GPU.

First conv layer	Root block	FPS	mAP
with downsmapling	**1**: 7×7	59.5	73.1
	1: 3×3	62.9	73.2
	2: 3×3	58.1	74.9
	3: 3×3	54.5	75.4
without downsmapling	**1**: 7×7	37.0	77.6
	1: 3×3	37.2	77.8
	2: 3×3	31.5	78.1
	3: 3×3	26.9	**78.5**
	4: 3×3	24.3	78.4
	5: 3×3	21.8	78.5

The aforementioned conclusions can be also extended to deeper ResNet backbone network, *e.g.*, ResNet-34. As shown in Table 3, using Root-ResNet-34, the mAP of our ScratchDet is improved from 78.5% to 80.4%, which is the best results with 300×300 input size. In comparison experiments on the benchmarks, we use Root-ResNet-34 as the backbone network.

4.2.3 Results

We compare ScratchDet to the state-of-the-art detectors in Table 3. With small input 300×300, ScratchDet produces 80.4% mAP without bells and whistles, better than several state-of-the-art one-stage pretrained object detectors (*e.g.*, 80.0% mAP of RefineDet320 and 79.7% mAP of DES300). Note that we keep most of original SSD configurations and the same epochs with DSOD. The result is much better than SSD300-VGG16 (80.4% *v.s* 77.2% and 3.2% mAP higher) and SSD321-ResNet101 (80.4% *v.s* 77.1%, 3.3% mAP higher). ScratchDet outperforms the state-of-the-art train-from-scratch detector with 1.7% improvements on mAP score (*i.e.*, 80.4% *v.s* 78.7% of GRP-DSOD). In the multi-scale testing, our ScratchDet achieves 84.1% (ScratchDet300+) mAP, which is the state-of-the-art.

4.3. PASCAL VOC 2012

Following the evaluation protocol of VOC 2012, we use VOC 2012 `trainval` set, and VOC 2007 `trainval` and `test` sets ($21,503$ images) to train our ScratchDet from scratch, and test on VOC 2012 `test` set ($10,991$ images). The detection results of ScratchDet are submitted to the public testing server for evaluation. The learning rate and batch size are set the same as that in VOC 2007.

Table 3 reports the accuracy of ScratchDet as well

Table 3. Detection results on the PASCAL VOC datasets. For VOC 2007, all methods are trained on the VOC 2007 and 2012 `trainval` sets and tested on the VOC 2007 `test` set. For VOC 2012, all methods are trained on the VOC 2007 and 2012 `trainval` sets plus the VOC 2007 `test` set, and tested on the VOC 2012 `test` set. The FPS of ScratchDet is measured on one TITAN X GPU for the fair comparison. †: `http://host.robots.ox.ac.uk:8080/anonymous/0HPCHC.html` ‡: `http://host.robots.ox.ac.uk:8080/anonymous/JSL6ZY.html`

Method	Backbone	Input size	FPS	mAP (%)	
				VOC 2007	VOC 2012
pretrained two-stage:					
HyperNet [19]	VGG-16	$\sim 1000 \times 600$	0.88	76.3	71.4
Faster R-CNN[28]	ResNet-101	$\sim 1000 \times 600$	2.4	76.4	73.8
ION[1]	VGG-16	$\sim 1000 \times 600$	1.25	76.5	76.4
MR-CNN[9]	VGG-16	$\sim 1000 \times 600$	0.03	78.2	73.9
R-FCN[4]	ResNet-101	$\sim 1000 \times 600$	9	80.5	77.6
CoupleNet[43]	ResNet-101	$\sim 1000 \times 600$	8.2	82.7	80.4
pretrained one-stage:					
RON384[18]	VGG-16	384×384	15	74.2	71.7
SSD321[8]	ResNet-101	321×321	11.2	77.1	75.4
SSD300*[24]	VGG16	300×300	46	77.2	75.8
YOLOv2[27]	Darknet-19	544×544	40	78.6	73.4
DSSD321[8]	ResNet-101	321×321	9.5	78.6	76.3
DES300[42]	VGG-16	300×300	29.9	79.7	77.1
RefineDet320[39]	VGG-16	320×320	40.3	80.0	78.1
trained from scratch:					
DSOD300[32]	DS/64-192-48-1	300×300	17.4	77.7	76.3
GRP-DSOD320[33]	DS/64-192-48-1	300×300	16.7	78.7	77.0
ScratchDet300	Root-ResNet-34	300×300	17.8	80.4	78.5†
ScratchDet300+	Root-ResNet-34	-	-	**84.1**	**83.6**‡

as the state-of-the-art methods. Using small input size 300×300, ScratchDet produces 78.5% mAP, surpassing some one-stage methods with similar input size, *e.g.*, SSD321-ResNet101 (75.4%, 3.1% higher mAP), DES300-VGG16 (77.1%, 1.4% higher mAP), and RefineDet320-VGG16 (78.1%, 0.4% higher mAP). Meanwhile, comparing to the two-stage methods based on pretrained networks with $\sim 1000 \times 600$ input size, ScratchDet also produces better results than R-FCN (77.6%, 0.9% higher mAP). In addition, our ScratchDet outperforms all the train-from-scratch detectors. It outperforms DSOD by 2.2% mAP with 60 less training epochs and surpasses GRP-DSOD by 1.5% mAP. Notably, in the multi-scale testing, ScratchDet obtains 83.6% mAP, much better than the state-of-the-arts of both one-stage and two-stage methods.

4.4. MS COCO

We also evaluate ScratchDet on MS COCO dataset. The model is trained from scratch on the MS COCO `trainval35k` set and tested on the `test-dev` set. We set the base learning rate to 0.05 for the first $150k$ iterations, and divide it by 10 successively for another $100k$, $60k$ and $10k$ iterations respectively.

Table 4 shows the results on the MS COCO `test-dev` set. ScratchDet produces 32.7% AP that is better than all the other methods with similar input size by a large margin, such as SSD300 (25.1%, 7.6% higher AP), SSD321 (28.0%, 4.7% higher AP), GRP-DSOD320 (30.0%, 2.7% higher AP), DSSD321 (28.0%, 4.7% higher AP), DES300 (28.3%, 4.4% higher AP), RefineDet320-VGG16 (29.4%, 3.3% higher AP), RetinaNet400 (31.9%, 0.8% higher AP) and RefineDet320-ResNet101 (32.0%, 0.7% higher AP). Notably, with the same input size, DSOD300 trains on the `trainval` set, which contains 5000 more images than `trainval35k` (*i.e.*, $123,287$ *v.s* $118,287$), and our ScratchDet produces a much better result (32.7% *v.s* 29.3%, 3.4% higher AP). Some methods use much bigger input sizes for both training and testing ($\sim 1000 \times 600$) than our ScratchDet300, *e.g.*, CoupleNet, Faster R-CNN and Deformable R-FCN. For a fair comparison, we also report the multi-scale testing AP results of ScratchDet300 in Table 4, *i.e.*, 39.1%, which is currently the best result, surpassing those prominent two-stage and one-stage approaches with large input image sizes.

Comparing to the state-of-the-art methods with similar input image size, ScratchDet300 produces the best AP_S (13.0%) for small objects, outperforming SSD321 by 6.8%. The significant improvement in small object demonstrates the superiority of our ScratchDet architecture for small object detection.

4.5. From MS COCO to PASCAL VOC

We also study how the MS COCO dataset help the detection on PASCAL VOC. Since the object classes in PASCAL VOC are from an subset of MS COCO, we directly finetune the detection models pretrained on MS COCO by subsampling parameters. As shown in Table 5, ScratchDet300 achieves 84.0% and 82.1% mAP on the VOC 2007 `test` set and VOC 2012 `test` set, outperforming other train-from-scratch methods. In the multi-scale testing, the detection

Table 4. Detection results on the MS COCO `test-dev` set.

Method	Data	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
pretrained two-stage:								
ION[1]	train	VGG-16	23.6	43.2	23.6	6.4	24.1	38.3
OHEM++ [34]	trainval	VGG-16	25.5	45.9	26.1	7.4	27.7	40.3
R-FCN[4]	trainval	ResNet-101	29.9	51.9	-	10.8	32.8	45.0
CoupleNet[43]	trainval	ResNet-101	34.4	54.8	37.2	13.4	38.1	50.8
Faster R-CNN+++ [12]	trainval	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [21]	trainval35k	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN w TDM[35]	trainval	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Deformable R-FCN[5]	trainval	Aligned-Inception-ResNet	37.5	58.0	40.8	19.4	40.1	52.5
Mask R-CNN[11]	trainval35k	ResNet-101-FPN	38.2	**60.3**	41.7	20.1	43.2	**51.2**
pretrained one-stage:								
YOLOv2[27]	trainval35k	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD300*[24]	trainval35k	VGG16	25.1	43.1	25.8	6.6	25.9	41.4
RON384++[18]	trainval	VGG-16	27.4	49.5	27.1	-	-	-
SSD321[8]	trainval35k	ResNet-101	28.0	45.4	29.3	6.2	28.3	49.3
DSSD321[8]	trainval35k	ResNet-101	28.0	46.1	29.2	7.4	28.1	47.6
DES300[42]	trainval35k	VGG-16	28.3	47.3	29.4	8.5	29.9	45.2
DFPR300 [17]	trainval	VGG-16	28.4	48.2	29.1	8.2	30.1	44.2
RefineDet320[39]	trainval35k	VGG-16	29.4	49.2	31.3	10.0	32.0	44.4
DFPR300 [17]	trainval	ResNet-101	31.3	50.5	32.0	10.5	33.8	49.9
PFPNet-R320 [16]	trainval35k	VGG-16	31.8	52.9	33.6	12.0	35.5	46.1
RetinaNet400[22]	trainval35k	ResNet-101	31.9	49.5	34.1	11.6	35.8	48.5
RefineDet320[39]	trainval35k	ResNet-101	32.0	51.4	34.2	10.5	34.7	50.4
trained from scratch:								
DSOD300[32]	trainval	DS/64-192-48-1	29.3	47.3	30.6	9.4	31.5	47.0
GRP-DSOD320[33]	trainval	DS/64-192-48-1	30.0	47.9	31.8	10.9	33.6	46.3
ScratchDet300	trainval35k	Root-ResNet-34	32.7	52.0	34.9	13.0	35.6	49.0
ScratchDet300+	trainval35k	Root-ResNet-34	**39.1**	59.2	**42.6**	**23.1**	**43.5**	51.0

Table 5. Detection results on PASCAL VOC dataset. All models are pretrained on MS COCO, and fine-tuned on PASCAL VOC.
†: `http://host.robots.ox.ac.uk:8080/anonymous/ZVCMYN.html`
‡: `http://host.robots.ox.ac.uk:8080/anonymous/OFHUPV.html`

Method	Backbone	mAP (%)	
		VOC 2007	VOC 2012
pretrained two-stage:			
Faster R-CNN[28]	VGG-16	78.8	75.9
OHEM++[34]	VGG-16	-	80.1
R-FCN[4]	ResNet-101	83.6	82.0
pretrained one-stage:			
SSD300[24]	VGG-16	81.2	79.3
RON384++[18]	VGG-16	81.3	80.7
RefineDet320[39]	VGG-16	84.0	82.7
trained without ImageNet:			
DSOD300[32]	DS/64-192-48-1	81.7	79.3
ScratchDet300	Root-ResNet-34	84.0	82.1†
ScratchDet300+	Root-ResNet-34	**86.3**	**86.3**‡

accuracies are promoted to 86.3% and 86.3%, respectively. By using the training data in MS COCO and PASCAL VOC, our ScratchDet obtains the top mAP scores on both VOC 2007 and 2012 datasets.

4.6. Comparison of the training time

ScratchDet uses obviously more time than fine-tuning a pretrained classifier on the 4 NVIDIA Tesla P40 GPUs workstation with the 300×300 input image size for the MS COCO dataset (*i.e.*, 84.6 vs. 29.7 hours). However, considering several weeks and millions of images involved in the pretraining phase, training detectors from scratch is more attractive than the pretrained detector. Notice that the comparison of training time is based on mmdetection framework [2].

5. Conclusion

In this work, we focus on training object detectors from scratch in order to tackle the problems caused by fine-tuning from pretrained networks. We study the effects of BatchNorm in the backbone and detection head subnetworks, and successfully train detectors from scratch. By taking the pretaining-free advantage, we are able to explore various architectures for detector designing. After analyzing the performance of the ResNet and VGGNet based SSD, we propose a new Root-ResNet backbone network to further improve the detection accuracy, especially for small objects. As a consequence, the proposed detector sets a new state-of-the-art performance on the PASCAL VOC 2007, 2012 and MS COCO datasets for the train-from-scratch detectors, even outperforming some one-stage pretrained methods.

Acknowledgements

We thank the engineers Jianhao Zhang and Peng Cheng in JD AI Research for their helpful suggestions for this work.

References

[1] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016.

[2] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. mmdetection. `https://github.com/open-mmlab/mmdetection`, 2018.

[3] Bowen Cheng, Yunchao Wei, Honghui Shi, Rogerio Feris, Jinjun Xiong, and Thomas Huang. Revisiting rcnn: On awakening the classification power of faster rcnn. In *ECCV*, 2018.

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *CVPR*, 2017.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. `http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html`.

[8] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *CoRR*, 2017.

[9] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *ICCV*, 2015.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

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

[13] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[15] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACMMM*, 2014.

[16] Seung-Wook Kim, Hyong-Keun Kook, Jee-Young Sun, Mun-Cheon Kang, and Sung-Jea Ko. Parallel feature pyramid network for object detection. In *ECCV*, 2018.

[17] Tao Kong, Fuchun Sun, Wenbing Huang, and Huaping Liu. Deep feature pyramid reconfiguration for object detection. In *ECCV*, 2018.

[18] Tao Kong, Fuchun Sun, Anbang Yao, Huaping Liu, Ming Lu, and Yurong Chen. Ron: Reverse connection with objectness prior networks for object detection. In *CVPR*, 2017.

[19] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, 2016.

[20] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet: A backbone network for object detection. In *ECCV*, 2018.

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *ICCV*, 2017.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[25] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. In *ICLR workshop*, 2016.

[26] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[27] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[30] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? (no, it is not about internal covariate shift). In *NIPS*, 2018.

[31] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2013.

[32] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch. In *ICCV*, 2017.

[33] Zhiqiang Shen, Honghui Shi, Rogerio Feris, Liangliang Cao, Shuicheng Yan, Ding Liu, Xinchao Wang, Xiangyang Xue, and Thomas S Huang. Learning object detectors from scratch with gated recurrent feature pyramids. *CoRR*, 2017.

[34] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[35] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond skip connections: Top-down modulation for object detection. *CoRR*, 2016.

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

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

[38] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[39] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *CVPR*, 2018.

[40] Shifeng Zhang, Longyin Wen, Hailin Shi, Zhen Lei, Siwei Lyu, and Stan Z Li. Single-shot scale-aware network for real-time face detection. *IJCV*, 2019.

[41] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. S^3FD: Single shot scale-invariant face detector. In *ICCV*, 2017.

[42] Zhishuai Zhang, Siyuan Qiao, Cihang Xie, Wei Shen, Bo Wang, and Alan L Yuille. Single-shot object detection with enriched semantics. In *CVPR*, 2018.

[43] Yousong Zhu, Chaoyang Zhao, Jinqiao Wang, Xu Zhao, Yi Wu, Hanqing Lu, et al. Couplenet: Coupling global structure with local parts for object detection. In *ICCV*, 2017.