International Journal of Innovative Research in Science and Engineering
Volume 9, Issue 04, 2023
www.ijirse.com
ISSN: 2454-9665

Object Detection from Scratch Using Deep Supervision

Ch. Nagalakshmi , J. Roja Nalini, B. SivaKonda, A.Uday Pavan

(Under the guidance of Mr. S. Anil Kumar M.Tech(Ph. D),Associate Professor,

Department of Computer Science and Engineering,Tirumala Engineering College)

ABSTRACT

In this paper, we propose Deeply Supervised Object Detectors (DSOD), an object detection framework that can be trained from scratch. Recent advances in object detection heavily depend on the off-the-shelf models pre-trained on large-scale classification datasets like ImageNet and OpenImage. However, one problem is that adopting pre-trained models from classification to detection task may incur learning bias due to the different objective function and diverse distributions of object categories. Techniques like fine-tuning on detection task could alleviate this issue to some extent but are still not fundamental. Furthermore, transferring these pre-trained models across discrepant domains will be more difficult (e.g., from RGB to depth images).

Index Terms—Object detection, deeply supervised networks, learning from scratch, densely connected layers

1 INTRODUCTION

ENERIC object detection is the task that we aim to local- ize various objects in a natural image automatically. This task has been heavily studied due to its wide applica- tions in surveillance, autonomous driving, intelligent secu- rity, etc. In the recent years, with the progress of more and more innovative and powerful Convolutional Neural Net- works (CNNs) based object detection systems have been proposed, the object detection problem has been one of the fastest moving areas in computer vision.

To achieve desired performance, the common practice in advanced object detection systems is to fine-tune models pretrained on ImageNet [3]. This fine-tuning process can be viewed as transfer learning [4]. Specifically, as is shown

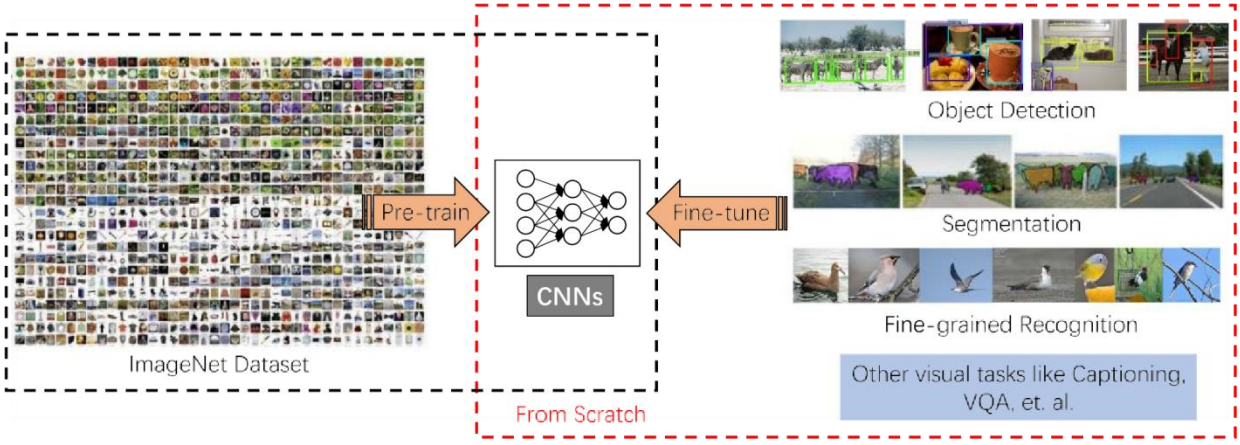

Fig. 1. Illustration of *training models from scratch*. The black dashed box (left) denotes we pre-train models on large-scale classification dataset like ImageNet [3].

International Journal of Innovative Research in Science and Engineering
Volume 9, Issue 04, 2023
www.ijirse.com
ISSN: 2454-9665

2 RELATED WORK

Proposal-based family includes R-CNN [5], Fast R- CNN [6], Faster R-CNN [7], R-FCN [8] and Mask RCNN [1]. R-CNN uses selective search [43] to first generate potential object regions in an image and then perform classification on the proposed regions. R-CNN requires high computa- tional costs since each region is processed by the CNN net- work separately. Fast R-CNN improves the efficiency by sharing computation of backbone networks and Faster R- CNN uses neural networks (i.e., RPN) to generate the region proposals. R-FCN further improves speed and accuracy by

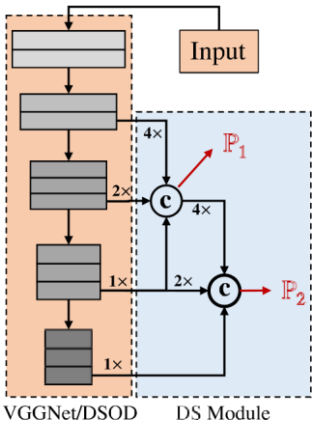

Fig. 3. Illustration of the deep-scale supervision (DSS) module. "4 × , 2 × and 1 × " denote that we reduce the resolution of feature maps to 1=4, 1=2 and the original size, respectively. "c" denotes concatenation operation. "P_1 and P_2" are the first (38 × 38) and second scales (19 × 19) of prediction modules in Fig. 2. "P_3-P_5" also use three-scale feature maps for prediction, which are not presented in this figure.

Network Architectures for Detection. Since there are sig- nificant efforts that have been devoted to design network architectures for image classification, many diverse and powerful networks are emerged, such as AlexNet [49], VGGNet [50], GoogLeNet [51], ResNet [52], DenseNet [39], etc. Meanwhile, several advanced regularization techni- ques [53], [54] also have been proposed to further enhance the model capabilities. In practice, most of the detection methods [5], [6], [7], [9] directly utilize these structures pre-trained on ImageNet as the backbone network for detection task.

Some other works try to design specific backbone net- work structures for object detection, but still require to pre- train on ImageNet classification dataset in advance. Specifi- cally, YOLO [10] defines a network with 24 convolutional layers followed by 2 fully-connected layers. YOLO9000 [55] improves YOLO by proposing a new network named Dar- knet-19, which is a simplified version of VGGNet [50]. YOLOv3 [56] further improve the performance through involving residual connection on Darknet-19 and other tech- niques. Kim et al. [57] proposes PVANet for fast object detection, which consists of the simplified "Inception" block from GoogleNet. Huang et al. [58] investigated various combination of network structures and detection frame- works, and found that Faster R-CNN [7] with Inception- ResNet-v2 [59] achieved very promising performance.

In this paper, we also consider designing a suitable backbone structure for generic object detection. However, the pre- training operation on ImageNet is no longer required by the proposed DSOD.

Learning Deep Models from Scratch. To the best of our knowledge, there are no previous works that train deep CNNbased object detectors from scratch. Thus, our pro- posed approach has very appealing advantages over exist- ing solutions. We will elaborate and validate the method in the following sections. In semantic segmentation, J´egou et al. [60] demonstrated that a well-designed network structure can outperform state-of-the-art solutions without using the pre-trained models. It extends DenseNets to fully-convolutional networks by adding an upsampling path to recover the full input resolution.

3 DSOD

In this section, we first introduce the whole framework of our DSOD architecture, following by several important design principles. Then we describe the objective function and training settings in detail.

3.1 Network Architecture

Similar to SSD [9], our proposed DSOD method is a multi- scale and proposal-free detection framework. The network structure of DSOD can be divided into two parts: the back- bone sub-network for feature extraction and the front-end sub-network for prediction over multi-resolution feature maps. The backbone sub-network is a variant of the deeply supervised DenseNets [39] structure, which is composed of a *stem block*, four *dense blocks*, two *transition layers* and two *tran- sition w/o pooling layers*. The front-end subnetwork (or named *DSOD prediction layers*) fuses multi-scale prediction responses with an elaborated *dense structure*. Fig. 2 illustrates the proposed DSOD prediction layers along with the plain structure used in SSD [9]. The full DSOD network architec- ture[1] is detailed in Table 1. Now we elaborate each compo- nent and the corresponding design principle in the following.

3.2 Design Principles

Principle 1: Proposal-Free. In order to reveal the potential influences in learning object detection from scratch, we

1. The visualization of the complete network structure is available at:
http://ethereon.github.io/netscope/#/gist/b17d01f3131e2a60f9057b5d 3eb9e04d.

investigated all the state-of-the-art CNN-based object detec- tors under the default settings. As aforementioned, R-CNN and Fast R-CNN require external object proposal generators like selective search. Faster R-CNN and R-FCN require inte- grated region-proposal-network (RPN) to generate rela- tively fewer region proposals. YOLO and SSD are single- shot and proposal-free methods (one-stage), which handle object location and bounding box coordinates as a regres- sion problem. We observe that only proposal-free methods (one-stage detectors) can converge successfully without the pre-trained models if we follow the original settings with- out involving some significantly modifications (e.g., replac- ing RoI pooling with RoI align [1], adopting Sync BN [61] or Group Norm [62] to mitigate small batch-size issue, etc.).

We conjecture this is due to the RoI pooling (Regions of Interest) in the other two categories of methods — RoI pool- ing uses quantization to generate features for each region proposals, which causes misalignments that hinders/ reduces the gradients being smoothly back-propagated from region-level to convolutional feature maps.

The pro- posal-based methods work well with pre-trained network models because the parameter initialization is good for those layers before RoI pooling, while this is not true for training from scratch.

Hence, we arrive at the first principle: training detection network from scratch requires a proposal-free framework, even if there is no BN layer [54] included in the network structures (In contrast, norm layer is critical for both Sync BN [61] and Group Norm [62] methods to train region- based/two-stage detectors from scratch). In practice, we derive a multi-scale proposal-free framework from the SSD framework [9], as it could reach state-of-the-art accuracy while offering fast processing speed.

Principle 2: Deep Supervision. Using deeply supervised structures to improve network performance has been dem- onstrated a effective practice in GoogLeNet [51], DSN [37], DeepID3 [63], etc. Among these network structures, the cen- tral idea is to provide integrated objective function as direct supervision to the earlier hidden layers, rather than only at the output one. These "companion" or "auxiliary" objective functions at multiple hidden layers can mitigate the "vanishing" gradients problem. The proposal-free detection framework contains both classification and localization loss. The explicit solution requires adding complex side-output layers to introduce "companion" objective at each hidden layer for the detection task, similar to [38]. In this work, we empower *deep supervision* with an elegant & implicit solution called layer-wise dense connections, as introduced in Dense- Nets [39]. A block is called *dense block* when all preceding layers in the block are connected to the current layer.

For low-level (coarse resolution) features, we use a 4 × 4 max pooling, stride ¼ 2 to reduce the resolution, following by a 1 × 1 conv-layer for reducing the number of feature maps. We use the 2 × 2 max pooling for middle level feature maps and do not include max pooling for high-level layers. Then, we concatenate these diverse feature maps together for final prediction. Each prediction layer can be formulated as

$$\mathbb{P}i \text{ ¼ } f_i\text{½P1}_{=}4\eth x_L\text{Þ; P1}_{=}2\eth x_M\text{Þ; } x_H\text{];} \qquad (1)$$

where $\mathbb{P}_i\eth i$ ¼ 1; 2; ... ; 5Þ denotes the *i*th prediction layer out- puts. $P_{1=k}$ denotes $k \times k$ max pooling. x_L, x_M and x_H denote feature maps from different layers. We will verify the bene- fit of deep supervision in Section 4.1.2.

Transition w/o Pooling Layer. In order to increase the num- ber of dense blocks without reducing the final feature map resolution, we introduce a new layer called *transition w/o pooling layer*. In the original design of DenseNet, each transi- tion layer contains a pooling operation to down-sample the feature resolution. The number of dense blocks is fixed (4 dense blocks in all DenseNet architectures) if one wants to maintain the same scale/size of outputs. The only way to increase network depth is adding layers inside each block for the original DenseNet. The *transition w/o pooling layer* eliminates this restriction of the number of dense blocks in DSOD architecture. You can include any number of blocks in a network as you want, which can also be adopted by the standard DenseNet.

Principle 3: Stem Block. Motivated by Inception-v3 [64] and v4 [59], we define stem block as a stack of three 3 × 3 convolution layers followed by a 2 × 2 max pooling layer. The first conv-layer works with stride = 2 and the other two are with stride = 1. We find that adding this simple stem structure can evidently improve the detection performance in our experiments. We conjecture that, compared with the original design in DenseNet (7 × 7 conv-layer, stride = 2 fol- lowed by a 3 × 3 max pooling, stride = 2), the stem block can reduce the

information loss from raw input images with small kernel size at the beginning of a network. We will show that the reward of this stem block is significant for object detection performance in Section 4.1.2.

Principle 4: Dense Prediction Structure. Fig. 2 illustrates the comparison of the plain structure (as in SSD) and our pro- posed dense structure in the front-end sub-network. SSD designs prediction-layers as an asymmetric hourglass struc- ture. For 300 × 300 input size, SSD applies six scales of fea- ture maps for predicting objects. The Scale-1 feature maps are from the middle layer of the backbone sub-network, which has the largest resolution (38 × 38) in order to handle the small objects in an image. The remaining five scales are on top of the backbone sub-network. Then, a plain transition layer with the *bottleneck* structure (a 1 × 1 conv-layer for reducing the number of feature maps plus a 3 × 3 conv- layer) [52], [64] is adopted between two contiguous scales of feature maps.

Learning Half and Reusing Half. In plain structure, each later scale of prediction layer is directly transited from the adjacent previous scale layer, as shown in Fig. 2, which is used in SSD framework. In this work, we propose to use dense structure for prediction. Each prediction layer combines multi-scale information from two stages of layers. For simplicity, we restrict that each scale outputs the same num- ber of channels for the prediction feature maps as is in the plain structure. In DSOD of each scale (except scale-1), half of the feature maps are learned from the previous scale layer with a series of conv-layers, while the remaining half feature maps are directly down-sampled from the contiguous high- resolution feature maps. The down-sampling block consists of a 2 × 2, stride ¼ 2 max pooling layer followed by a 1 × 1, stride = 1 conv-layer. The pooling layer aims to match resolu- tion to current size during concatenation. The 1 × 1 conv- layer is used to reduce the number of channels to 50 percent. The pooling layer is placed before the 1 × 1 conv-layer for the consideration of reducing computing cost. This down- sampling block actually brings each scale with the multi-res- olution feature maps from all of its preceding scales, which is essentially identical to the dense layer-wise connection intro- duced in DenseNets. For each scale, we only learn half of new feature maps and reuse the remaining half of the previ- ous ones. This dense prediction structure can yield more accurate results with fewer parameters than the plain struc- ture, as will be studied in Section 11.

References:

1. https://en.wikipedia.org/wiki/Fee_management_system
2. https://www.creatrixcampus.com/feemanagement-system
3. http://www.ifnoss.com/edu/collegeuniversity-software/student-feesmanagement.aspx