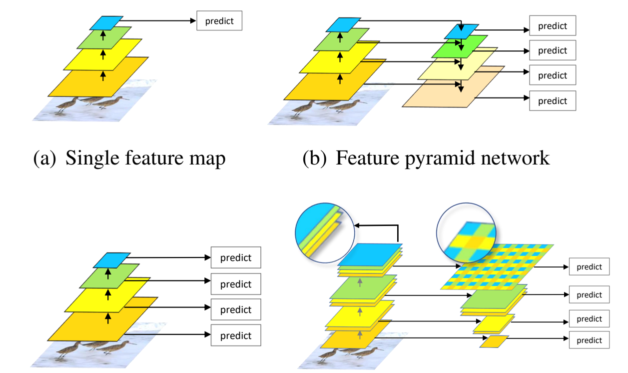
Scale-Transferrable Object Detection

Peng Zhou[1] Bingbing Ni[*,1] Cong Geng[1] Jianguo Hu[2] Yi Xu[1]
[1]Shanghai Key Laboratory of Digital Media Processing and Transmission,
Shanghai Institute for Advanced Communication and Data Science,
Shanghai Jiao Tong University, Shanghai 200240, China
[2]Minivision
[1]{zhoupengcv,nibingbing,gengcong,xuyi}@sjtu.edu.cn, [2]hujianguo@minivision.cn

Abstract

Scale problem lies in the heart of object detection. In this work, we develop a novel Scale-Transferrable Detection Network (STDN) for detecting multi-scale objects in images. In contrast to previous methods that simply combine object predictions from multiple feature maps from different network depths, the proposed network is equipped with embedded super-resolution layers (named as scale-transfer layer/module in this work) to explicitly explore the inter-scale consistency nature across multiple detection scales. Scale-transfer module naturally fits the base network with little computational cost. This module is further integrated with a dense convolutional network (DenseNet) to yield a one-stage object detector. We evaluate our proposed architecture on PASCAL VOC 2007 and MS COCO benchmark tasks and STDN obtains significant improvements over the comparable state-of-the-art detection models.



(a) Single feature map (b) Feature pyramid network

(c) Pyramidal feature hierarchy (d) Scale-transfer module

Figure 1. (a) Using only single scale features for prediction. (b) Integrating information from high-level and low-level feature maps. (c) Producing predictions from feature maps of different scales. (d) Our Scale-Transfer Module. It can be embedded directly into a DenseNet to obtain feature maps at different scales.

1. Introduction

Scale problem lies in the heart of object detection. In order to detect objects of different scales, a basic strategy is to use image pyramids [1] to obtain features at different scales. However, this will greatly increase memory and computational complexity, which will reduce the real-time performance of object detectors.

In recent years, convolutional neural networks(CNN) [18] have achieved great success in computer vision tasks, such as image classification [17], semantic segmentation [23], and object detection [10]. The hand-engineered features are replaced with features computed by convolutional neural networks, which greatly improves the performance of object detectors. Faster R-CNN [25] uses convolutional feature maps computed by one layer to predict candidate region proposals with different scales and aspect ratios (Figure 1(a)). Because the receptive field of each layer in CNN is fixed, there exists inconsistency between the fixed receptive field and the objects at different scales in natural images. This may compromise object detection performance. SSD [22] and MS-CNN [3] use feature maps from different layers within CNN to predict objects at different scales (See Figure 1(c)). Shallow feature maps have small receptive fields that are used to detect small objects, and deep feature maps have large receptive fields that are used to detect large objects. Nevertheless, shallow features have less semantic information, which may impair the performance of small object detection. FPN [20], ZIP [19] and DSSD [7] integrate semantic information on feature maps at all scales. As shown in Figure 1(b), a top-down architecture combines high-level semantic feature maps with low-level feature maps to yield more semantic feature maps at all scales. However, in order to improve detection performance, feature pyramids must be carefully constructed, and

*Corresponding author: Bingbing Ni.

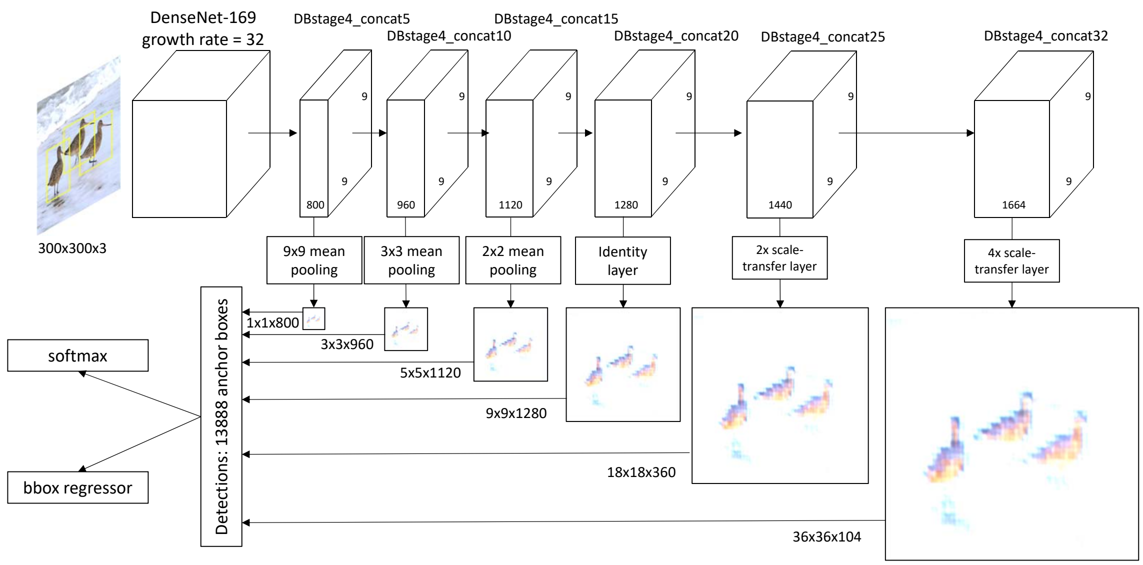


Figure 2. **Scale-Transferrable Detection Network (STDN) for detecting multi-scale objects in images.** We use DenseNet-169 [14] as the base network. The figure shows several layers in the last dense block of DenseNet-169. The output of each layer in dense block has the same height and width. We use scale-transfer layer to obtain high-resolution feature maps for detecting small objects, and use pooling layer to obtain feature maps with large receptive field for detecting large objects. These layers can be directly embedded into the base network with little computational overhead. The real-time performance of the detector is guaranteed by the scale-transfer module.

adding extra layers to build the feature pyramids brings additional computational cost.

In order to obtain high-level semantic multi-scale feature maps, and also without impairing the speed of the detector, we develop a Scale-Transfer Module (STM) and embed this module directly into a DenseNet [14]. The role of DenseNet is to integrate low-level and high-level features within a CNN to get more powerful features. Because of the densely connected network structure, the features of DenseNet are naturally more powerful than the ordinary convolutional features. STM consists of pooling and scale-transfer layers. Pooling layer is used to obtain small scale feature maps, and scale-transfer layer is used to obtain large scale feature maps. Scale-transfer layer is first proposed to do image super-resolution [28] because of its simplicity and efficiency, and some people also use it to do semantic segmentation [30]. We use this layer to efficiently expand the resolution of the feature map for object detection.

We construct a one-stage object detector named Scale-Transferrable Detection Network (STDN) using scale-transfer module. Figure 2 shows the architecture of our detector. Since the feature map of the last layer of the last dense block in DenseNet has most number of channels, scale-transfer layer expand the width and height of the feature map by compressing the number of channels (Figure 1(d)). This can effectively reduce the number of parameters of the next convolution layer.

STM naturally fits the base network and enable end-to-end training. We believe that the STM has two distinct advantages. First of all, combining DenseNet [14] the feature maps own both low-level object detail features and high-level semantic features naturally. We will prove that this will improve the accuracy of object detection. Second, STM is made up of pooling and super-resolution layers with no additional parameters and computation. The results of the experiment demonstrate that the proposed framework in this paper can detect objects accurately and meet the real-time requirements.

2. Related Work

State-of-the-art methods of object detection are based on convolutional neural networks. For example, SPPnet [11], Fast R-CNN [9], Faster R-CNN [25], R-FCN [5], and YOLO [24] use features from the top layer of the convolutional neural network to detect objects of different scales. However, since each layer of the convolutional neural network has a fixed receptive field, it is not optimal to predict objects of different scales with only features of one layer.

There are generally three main types of methods to further improve the accuracy of multi-scale object detection. One is to detect objects using the combinations of multi-layer features. The other is to use different layer features to predict objects at different scales. The last is a combination of the above two methods.

For the first type of method, ION [2] uses skip pooling to extract information at multiple layers, and then the object is detected by using the combined features. HyperNet [16] incorporates deep, intermediate and shallow features of the image for generating proposals and detecting objects. YOLOv2 [24] concatenates the higher resolution features with the low-resolution features by passthrough layer and runs detection on top of this expanded feature map. The basic idea of these methods is to enhance the power of features by combining low-level and high-level features.

For the second type of method, SSD [22], MS-CNN [3] and DSOD [27] combine predictions from multiple feature maps to handle objects of various sizes. For example, for small objects, shallow features are used, and for large objects, deep features are used. There exists a problem that shallow features may impair the performance of small object detection due to the lack of semantic information.

The last type of method is to use the above two methods simultaneously. There are multiple prediction layers to predict objects at different scales, and the feature of each prediction layer is obtained by combining features at different depths. FPN [20] and TDM [29] use a top-down architecture for building high-level semantic feature maps. DSSD [7] uses an hourglass structure to pass context information for prediction. These methods have to add additional layers to obtain multi-scale features, introducing cost that can not be neglected.

Our proposed method falls into the third class approach. We use DenseNet [14] to combine features of different layers and use scale-transfer module to obtain feature maps with different resolutions. Our module can be directly embedded into the DenseNet network with little additional cost.

3. Scale-Transferrable Detection Network

In this section, we first introduce the base network which is our feature extraction network component. We use DenseNet [14] as our base network. In each dense block of DenseNet, for each layer, its feature map is used as inputs for all subsequent layers. The output of the last layer of the dense block has highest number of channels and is suitable as input for our scale-transfer layer which expands the width and height of the feature map by compressing the number of channels. Then we describe the scale-transfer module that produces feature maps at different scales. Next, we describe the entire object detection/location prediction network architecture and the network training details.

3.1. Base Network : DenseNet

We use DenseNet-169 [14] as our base network for feature extraction and do pre-training on the ILSVRC CLS-LOC dataset [26]. DenseNet is a network with deep supervision. In each dense block of DenseNet, the output of each layer contains the output of all previous layers, and thus incorporates low-level and high-level features of the input image, which is suitable for object detection. Inspired by DSOD [27], we replace the input layers (7×7 convolution layer, stride = 2 followed by a 3×3 max pooling layer, stride = 2) into three 3×3 convolution layers and one 2×2 mean pooling layer. The stride of the first convolution layer is 2 and the others are 1. The output channels for all three convolution layers are 64. We call these layers "stem block". Table 1 depicts our network architecture in detail. Experiments show that this simple substitution can significantly improve the accuracy of object detection (see Table 3 in ablation study). One explanation could be that the input layers in the original DenseNet-169 have lost much information due to two consecutive down sampling. This will impair the performance of object detection, especially for small objects.

3.2. High Efficiency Scale-Transfer Module

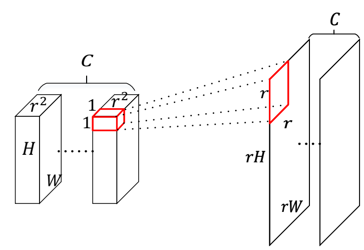


Figure 3. Scale-transfer layer

Scale problem lies in the heart of object detection. Combining predictions from multiple feature maps with different resolutions are beneficial for detecting multi-scale objects. However, as shown in Figure 2, in the last dense block of DenseNet, all outputs of layers have the same width and height, except for the number of channels. For example, when the input image is 300×300, the last dense block dimension of DenseNet-169 is 9×9. A simple approach is to predict directly using high-resolution feature maps of low layers, similar to SSD [22]. However, the low-level feature map lacks semantic information about objects, which may cause low performance on object detection. Ablation studies are shown in Table 3.

In order to obtain different resolution feature maps, inspired by the super-resolution approach [28], we develop a module dubbed as scale-transfer module. Scale-transfer module is very efficient and can be directly embedded into the dense block in DenseNet. To get strong semantic feature maps, we make use of the network structure of DenseNet to transfer the low-level features directly to the top of the network through the concat operation. The feature map at the top of the network has both low-level detail information and high-level semantic information so as to improve the performance of object localization and classification.

Layers	Output Size (Input $3 \times 300 \times 300$)	STDN
BatchNorm(BN), Convolution	$64 \times 150 \times 150$	3×3 conv, stride 2
BN, Relu, Convolution	$64 \times 150 \times 150$	3×3 conv, stride 1
BN, Relu, Convolution	$64 \times 150 \times 150$	3×3 conv, stride 1
Pooling	$64 \times 75 \times 75$	2×2 average pool, stride 2
Dense Block (1)	$256 \times 75 \times 75$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	$128 \times 75 \times 75$	1×1 conv
	$128 \times 37 \times 37$	2×2 average pool, stride 2
Dense Block (2)	$512 \times 37 \times 37$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	$256 \times 37 \times 37$	1×1 conv
	$256 \times 18 \times 18$	2×2 average pool, stride 2
Dense Block (3)	$1280 \times 18 \times 18$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Transition Layer (3)	$640 \times 18 \times 18$	1×1 conv
	$640 \times 9 \times 9$	2×2 average pool, stride 2
Dense Block (4)	$1664 \times 9 \times 9$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Scale-transfer module	$800 \times 1 \times 1$	9×9 average pool, stride 9 (Input DBstage4_concat5)
	$960 \times 3 \times 3$	3×3 average pool, stride 3 (Input DBstage4_concat10)
	$1120 \times 5 \times 5$	2×2 average pool, stride 2 (Input DBstage4_concat15)
	$1280 \times 9 \times 9$	Identity layer (Input DBstage4_concat20)
	$360 \times 18 \times 18$	$2\times$ scale-transfer layer (Input DBstage4_concat25)
	$104 \times 36 \times 36$	$4\times$ scale-transfer layer (Input DBstage4_concat32)

Table 1. STDN architecture (growth rate = 32 in each dense block)

We get feature maps of different scales from the last dense block of DenseNet. In scale-transfer module, we use the mean pooling layer to obtain low-resolution feature maps. For high-resolution feature maps, we use a technique called scale-transfer layer.

Suppose that the dimensions of the input tensor of the scale-transfer layer are $H \times W \times C \cdot r^2$, where r is the up sampling factor. The scale-transfer layer is an operation of periodic rearrangement of elements. As you can see from the Figure 3, scale-transfer layer expands the width and height by compressing the number of channels in the feature map. A mathematical formula can be expressed as the following form

$$I^{SR}_{x,y,c} = I^{LR}_{\lfloor x/r \rfloor, \lfloor y/r \rfloor, r \cdot mod(y,r) + mod(x,r) + c \cdot r^2} \qquad (1)$$

where I^{SR} are high resolution feature maps and I^{LR} are low resolution feature maps. Instead of using deconvolution layer, in which zeros have to be padded in the unpooling step before the convolution operation, the scale-transfer layer has no additional parameters and computational overhead. We call the mean pooling and scale-transfer layers as scale-transfer module. We embed the scale-transfer module directly into DenseNet to obtain six feature maps at different scales. And then we use these feature maps to construct a one-stage object detector named scale-transferrable detection network (STDN). Figure 2 shows the size of the six feature maps for the input image of 300×300.

Scale-transfer layer can effectively reduce the number of channels in the last layer of the dense block in DenseNet, and reduce the parameters and computation of the next convolutional prediction layer. This improves the speed of the detector.

Using the advantages of DenseNet, the feature maps have both low-level detail information and high-level semantic information. The experimental results show that our scale-transferrable detector performs well both in accuracy and speed of object detection (Table 5).

3.3. Object Localization Module

The Scale-Transferrable Detection Network (STDN) consists of a base network and two task specific prediction subnetworks. The role of the base network is to do feature extraction. The first subnet is used for object classification, and the second subnet is used for bounding box position regression. We have described the base network in detail above. Next we will detail two prediction subnetworks and training objective.

3.3.1 Anchor Boxes.

We associate a set of default anchor boxes with each feature map which is got by our scale-transfer module. The scale of anchor boxes is the same as that of SSD [22]. Following DSSD [7], we use $[1.6, 2.0, 3.0]$ aspect ratios at every prediction layer. Anchors are matched to any ground truth with intersection-over-union (IoU) higher than a threshold (0.5). The remaining anchors are treated as background. After the matching step, most of the default anchor boxes are negatives (no matched). We use hard negative mining so that the ratio between the negatives and positives is at most $3:1$.

3.3.2 Classification Subnet.

The role of the classification subnet is to predict the probability of each anchor belonging to a category. It contains a 1×1 convolution layer and two 3×3 convolution layers. Each convolution layer has a batchnorm layer [15] and a relu layer in front of it. The last convolution layer has KA filters, where K is the number of object classes and A is the number of anchors per spatial location. The classification loss is the softmax loss over multiple classes confidences.

3.3.3 Box Regression Subnet.

The purpose of this subnet is to regress the offset from each anchor box to the matched ground-truth object. The box regression subnet has the same structure as the classification subnet except that its last convolution layer has $4A$ filters. Smooth L1 loss [9] is used for the localization loss and the bounding box loss is only used for positive samples

3.3.4 Training Objective.

The training objective is to minimize a combined classification and localization loss:

$$\begin{aligned} L(a, I, \theta) =& L_{cls}(y_a, p_{cls}(I, a, \theta)) \\ &+ \lambda \cdot 1[y_a > 0] \cdot L_{loc}(\phi(b_a, a) - p_{loc}(I, a, \theta)) \end{aligned} \tag{2}$$

where a is the anchor, I is the image and θ is our optimized parameter. L_{cls} is the classification loss and L_{loc} is the localization loss. $y_a \in \{0, 1, \cdots, K\}$ is the class label, $y_a = 0$ when anchor a is not matched. $p_{loc}(I, a, \theta)$ and $p_{cls}(I, a, \theta)$ are predicted box encoding and corresponding class. $\phi(b_a, a)$ is an encoding of ground-truth box with respect to the matched anchor a. λ is a trade-off coefficient.

3.3.5 Training Settings.

Our detector is based on the MXNet [4] framework. All our models are trained with SGD solver on NVIDIA TITAN Xp GPU. We follow almost the same training strategy as SSD [22], including a random expansion data augmentation trick which is helpful for detecting small objects.

4. Experiments

4.1. Experiment Setup and Implementation

We evaluate our method on the PASCAL VOC [6] and MS COCO [21] datasets that have 20 and 80 object categories respectively. For PASCAL VOC, following the protocol in [9], we perform training on the union of VOC 2007 trainval and VOC 2012 trainval and test on the VOC 2007 test set. For COCO, following the standard protocol, training and evaluation are performed on the 120k images in the trainval and the 20k images in the test-dev, respectively.

For evaluation, we use the standard mean average precision (mAP) scores. For PASCAL VOC, we report mAP scores using IoU thresholds at 0.5. For COCO, we use the standard COCO metric.

For the model with 300×300 inputs on PASCAL VOC, we train the model with a mini-batch size 80 due to GPU memory constraints. We start the learning rate at 0.001 for the first 500 epochs. We decrease it to 10^{-4} at 600 epochs and 10^{-5} at 700 epochs. We take this well-trained model as pre-trained models for STDN321 and STDN513 on PASCAL VOC.

4.2. PASCAL VOC 2007

Table 2 shows our results on the PASCAL VOC2007 test detection. Among them, SSD uses only feature maps at different depths for prediction without fusing low-level and high-level features. We use it as our baseline. DSSD is an upgraded version of SSD, which replaces the base network of SSD from VGG to a deep residual network. Moreover, DSSD adds additional layers to fuse features at different depths. This will yield features with strong semantic information. DSSD321 and DSSD513 are about 1.1%-2% better than SSD300 and SSD512 respectively by adding these operations. Although DSSD improves accuracy compared to SSD, the speed of the object detector has been greatly damaged due to the extremely deep base network and inefficient feature fusion. The comparison of speed and accuracy will be discussed in Table 5. It is exciting that STDN embeds the efficient scale-transfer module into DenseNet, which improves the accuracy of object detection and hardly increases the running time. For input images of the same size, the accuracy of STDN300 is 0.6% higher than that of SSD300. STDN513 is 1.4% higher than SSD512. These results demonstrate the effectiveness of our approach. We also compare the results of STDN with DSSD. STDN321 is 0.7% higher than DSSD321, but STDN513 is 0.6% lower than DSSD513. We think the reason may be that Residual-101 has more network parameters than DenseNet169 ($42M$ vs $14M$), and thus has greater capacity. However, this slows

Method	network	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Faster [25]	VGG	73.2	76.5	79	70.9	65.5	52.1	83.1	84.7	86.4	52	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83	72.6
ION [2]	VGG	75.6	79.2	83.1	77.6	65.6	54.9	85.4	85.1	87	54.4	80.6	73.8	85.3	82.2	82.2	74.4	47.1	75.8	72.7	84.2	80.4
Faster [12]	Residual-101	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72
MR-CNN [8]	VGG	78.2	80.3	84.1	78.5	70.8	68.5	88	85.9	87.8	60.3	85.2	73.7	87.2	86.5	85	76.4	48.5	76.3	75.5	85	81
R-FCN [5]	Residual-101	80.5	79.9	87.2	81.5	72	69.8	86.8	88.5	89.8	67	88.1	74.5	89.8	90.6	79.9	81.2	53.7	81.8	81.5	85.9	79.9
SSD300 [22]	VGG	77.5	79.5	83.9	76	69.6	50.5	87	85.7	88.1	60.3	81.5	77	86.1	87.5	83.9	79.4	52.3	77.9	79.5	87.6	76.8
SSD512 [22]	VGG	79.5	84.8	85.1	81.5	73	57.8	87.8	88.3	87.4	63.5	85.4	73.2	86.2	86.7	83.9	82.5	55.6	81.7	79	86.6	80
DSSD321 [7]	Residual-101	78.6	81.9	84.9	80.5	68.4	53.9	85.6	86.2	88.9	61.1	83.5	78.7	86.7	88.7	86.7	79.7	51.7	78	80.9	87.2	79.4
DSSD513 [7]	Residual-101	**81.5**	86.6	86.2	82.6	74.9	62.5	89	88.7	88.8	65.2	87	78.7	88.2	89	87.5	83.7	51.1	86.3	81.6	85.7	83.7
STDN300	DenseNet-169	78.1	81.1	86.9	76.4	69.2	52.4	87.7	84.2	88.3	60.2	81.3	77.6	86.6	88.9	87.8	76.8	51.8	78.4	81.3	87.5	77.8
STDN321	DenseNet-169	79.3	81.2	88.3	78.1	72.2	54.3	87.6	86.5	88.8	63.5	83.2	79.4	86.1	89.3	88.0	77.3	52.5	80.3	80.8	86.3	82.1
STDN513	DenseNet-169	**80.9**	86.1	89.3	79.5	74.3	61.9	88.5	88.3	89.4	67.4	86.5	79.5	86.4	89.2	88.5	79.3	53.0	77.9	81.4	86.6	85.5

Table 2. **PASCAL VOC2007 test detection results.** Note that the minimum dimension of the input image for Faster and R-FCN is 600, and the speed is less than 10 frames per second. SSD300 indicates the input image dimension of SSD is 300×300. Large input sizes can give good results, but more running time is needed. All models is trained with the combined training set from VOC 2007 trainval and 2012 trainval, and is tested on the voc 2007 test set.

down the speed of the object detector at the same time.

4.3. Ablation Study on VOC2007

In order to verify the effectiveness of each component in STDN, we design ablation experiments on VOC2007. The results are summarized in Table 3.

4.3.1 Effect of Scale-transfer Module (STM).

In Table 3, DenseNet-169 + SSD indicates that we use DenseNet as the base network to extract features, and use feature maps at different depths to do object detection. This is similar to the SSD approach, but instead switches the base network from VGG to DenseNet-169. DenseNet-169 + STM indicates that we use scale-transfer module to obtain multi-scale feature maps for object detection. Under the same base network, the STM is 76.6% mAP, 2.8% higher than that of SSD. This verifies the effectiveness of the STM method.

4.3.2 Effect of Stem Block.

The third and fourth rows of Table 3 show that stem block can significantly improve object detection accuracy from 76.6% to 78.1%. We think the reason is that the original input layers in DenseNet-169 have two consecutive down sampling operations (conv and max pooling with stride 2). This results in too much loss of information, which impairs the performance of the detector.

4.3.3 Effect of Dense Convolutional Network.

DenseNet is a network with deep supervision, which connects each layer to every other layer in a feed-forward fashion. We compare the performance between deep supervised and non-deep supervised networks. We use Inception-v3 in comparison test, which has also been pre-trained on ILSVRC CLS-LOC dataset. We remove layers after the global pooling layer in Inception-v3, and then add the same structure as the last dense block in DenseNet-169. Multi-scale feature maps are obtained using STM. The results of the experiment show that the performance of DenseNet-169 exceeds that of Inception-v3 with a large margin. Using DenseNet-169 (with stem block) combined with STM, we have excellent performance on VOC2007 for images with only 300×300 input size.

Method	mAP	Anchor boxes	Input resolution
DenseNet-169 + SSD	73.8	14472	300×300
DenseNet-169 + STM	76.6	13888	300×300
DenseNet-169 + STM	76.6	13888	300×300
DenseNet-169 + stem block + STM	**78.1**	13888	300×300
Inception-v3 + STM	74.9	13888	332×332
DenseNet-169 + stem block + STM	**78.1**	13888	300×300

Table 3. **Ablation study on PASCAL VOC 2007 test set.** Note that DenseNet-169+SSD means replacing the base network of SSD from VGG to DenseNet-169. STM is the scale-transfer module. Stem block is explained in the above. For fair comparison, the input of Inception-v3 is resized to 332×332 so that the number of anchor box is consistent with the comparison test.

4.4. COCO

To further validate our approach, we train STDN300 and STDN513 on the COCO dataset [21]. We use trainval for training and test on the test-dev2015. Results are summarized in Table 4. STDN300 achieves 28.0%/45.6% on the test-dev set, which outperforms the baseline SSD300. The accuracy of SSD300 is the same as that of DSSD321, but has smaller input images and faster speed. We observe that the accuracy of STDN300 with 0.5 IoU is lower than DSSD321, but the [0.5:0.95] result is comparable. This indicates that STDN300 is more accurate under the large overlap. The accuracy of STDN513 is higher than SSD512, but lower than DSSD513. The reason may be that our base network has less parameters than Residual-101. Notably, the STDN513 is about 5 times as fast as DSSD513.

Method	Data	Network	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
Faster RCNN [25]	trainval	VGGNet	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [2]	train	VGGNet	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
R-FCN [5]	trainval	ResNet-101	29.2	51.5	-	10.3	32.4	43.3	-	-	-	-	-	-
R-FCNmulti-sc [5]	trainval	ResNet-101	29.9	51.9	-	10.8	32.8	45	-	-	-	-	-	-
DSOD300 [27]	trainval	DS/64-192-48-1	29.3	47.3	30.6	9.4	31.5	47	27.3	40.7	43	16.7	47.1	65
SSD300 [22]	trainval35k	VGG	25.1	43.1	25.8	6.6	25.9	41.4	23.7	35.1	37.2	11.2	40.4	58.4
SSD512 [22]	trainval35k	VGG	28.8	48.5	30.3	10.9	31.8	43.5	26.1	39.5	42	16.5	46.6	60.8
DSSD321 [7]	trainval35k	Residual-101	28.0	46.1	29.2	7.4	28.1	47.6	25.5	37.1	39.4	12.7	42	62.6
DSSD513 [7]	trainval35k	Residual-101	**33.2**	53.3	35.2	13	35.4	51.1	28.9	43.5	46.2	21.8	49.1	66.4
STDN300	trainval	DenseNet-169	28.0	45.6	29.4	7.9	29.7	45.1	24.4	36.1	38.7	12.5	42.7	60.1
STDN513	trainval	DenseNet-169	**31.8**	51.0	33.6	14.4	36.1	43.4	27.0	40.1	41.9	18.3	48.3	57.2

Table 4. COCO test-dev2015 detection results.

Method	Base network	mAP	Speed(*fps*)	Anchor boxes	GPU	Input resolution
Faster [25]	VGG16	73.2	7	6000	Titan X	$\approx 1000 \times 600$
Faster [12]	Residual-101	76.4	2.4	300	K40	$\approx 1000 \times 600$
R-FCN [5]	Residual-101	80.5	9	300	Titan X	$\approx 1000 \times 600$
DSOD300 [27]	DS/64-192-48-1	77.7	17.4	-	Titan X	300×300
YOLOv2 [24]	Darknet-19	78.6	40	-	-	544×544
SSD300 [22]	VGG16	77.5	46	8732	Titan X	300×300
SSD512 [22]	VGG16	79.5	19	24564	Titan X	512×512
DSSD321 [7]	Residual-101	78.6	9.5	17080	Titan X	321×321
DSSD513 [7]	Residual-101	81.5	5.5	43688	Titan X	513×513
STDN300	DenseNet-169	78.1	41.5	13888	Titan Xp	300×300
STDN321	DenseNet-169	79.2	40.1	17080	Titan Xp	321×321
STDN513	DenseNet-169	80.9	28.6	43680	Titan Xp	513×513

Table 5. **Comparison of Speed and Accuracy on PASCAL VOC2007 test set.** Training data is the union of VOC2007 and VOC2012 trainval. Faster R-CNN and R-FCN use input images whose minimum dimension is 600. The speed of STDN is measured with batch size 1.

4.5. Comparison of Speed and Accuracy

To test the speed of STDN, we test the total detection time for 2000 images sampled from PASCAL VOC2007 test set, and then calculate the frames per second (*fps*). We test speed with batch size 1 using Titan Xp with Intel Xeon CPU E5-2620v4@2.10GHz.

See Table 5 for a comparison of STDN with other frameworks on VOC 2007 test set. Compared to the SSD, DSSD combines low-level and high-level features. Although the accuracy has been improved, the speed has been greatly impaired (less than 10 frames per second). This is mainly due to the fact that the base network of DSSD is too deep and its feature fusion method is inefficient. However, our proposed method has a great advantage in speed and accuracy due to our efficient scale-transfer module.

In order to see the advantages and disadvantages of each method intuitively, we draw a scatter diagram of accuracy and speed. An excellent detector should be located in the upper right corner of the diagram. As shown in Figure 5, STDN performs well both in speed and accuracy. STDN300 and STDN513 are consistently superior in accuracy to SSD300 and SSD512. STDN321 are higher than YOLOv2 in accuracy and speed. At high resolution STDN513 achieves 80.9% mAP on VOC 2007 while still operating in real-time speeds.

5. Discussion

The original idea of scale-transfer layer is used to make image super resolution [28], and some people also use it to do semantic segmentation [30]. We apply this idea to object detection. Because of the multiple max (or mean) pooling layers in CNN, the size of the feature map on the top of the CNN is much smaller than the size of the input image. For example, for DenseNet-169 network, if the input image size is $300 \times 300 \times 3$, the size of the feature map on the top of the network will be $9 \times 9 \times 1664$, where 1664 is the number

Figure 4. Example of object detection results on the MS COCO test-dev set. The training data is COCO trainval. Each output box is associated with a category label and a softmax score in [0, 1]. A score threshold of 0.4 is used for displaying.

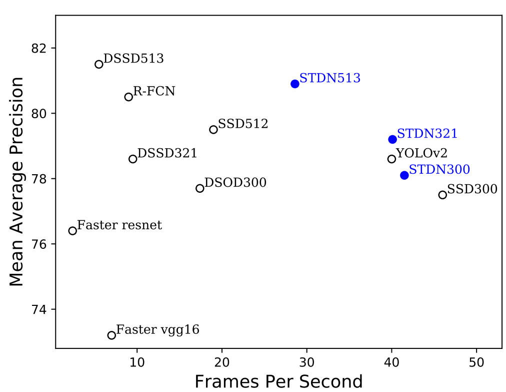


Figure 5. Accuracy and speed on PASCAL VOC2007

of channels. This means that one pixel in the feature map is related to about 33 pixels in the input image in the width and height space plane. It means that the CNN compresses the $33 \times 33 \times 3$ image information into the $1 \times 1 \times 1664$ feature map. If we use 4x scale-transfer layer, $1 \times 1 \times 1664$ feature map will be converted to $4 \times 4 \times 104$. This will increase the number of anchor boxes by 16 times, and fully explore the information contained in channels. Compared with other upscaling operation (deconvolution), there is no increase in parameters and computation. In addition, due to the reduction of the number of channels, this will effectively reduce the parameters of the next classification and box regression subnets, which is beneficial for efficiency of object detector. Because of these merits, the scale-transfer layer can be naturally embedded into DenseNet to obtain multi-scale feature maps, almost without increasing computational complexity and model size.

6. Conclusion

In this work, we propose a novel scale-transfer layer for efficient and compact object detection. We further develop scale-transferrable detection network based on this layer and extensive experiments show that STDN can produce markedly superior detection performance in terms of both accuracy and speed. At 41 FPS, STDN300 gets 78.1 mAP on VOC 2007. At 28 FPS, STDN513 gets 80.9 mAP, obtaining significant improvements over the comparable state-of-the-art detection methods.

Acknowledgements

This work was supported by National Science Foundation of China (U161146161502301,61671298,61521062). The work was partially supported by State Key Research and Development Program (2016YFB1001003), Chinas Thousand Youth Talents Plan, STCSM 17511105401 and 18DZ2270700.

References

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984. 1

[2] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2874–2883, 2016. 2, 6, 7

[3] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pages 354–370. Springer, 2016. 1, 3

[4] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015. 5

[5] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 2, 6, 7

[6] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 5

[7] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017. 1, 3, 5, 6, 7

[8] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1134–1142, 2015. 6

[9] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2, 5

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014. 2

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6, 7

[13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[14] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 3

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 5

[16] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 845–853, 2016. 3

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[18] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. 1

[19] H. Li, Y. Liu, W. Ouyang, and X. Wang. Zoom out-and-in network with recursive training for object proposal. *arXiv preprint arXiv:1702.05711*, 2017. 1

[20] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*, 2016. 1, 3

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5, 6

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 3, 5, 6, 7

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 1

[24] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. 2, 3, 7

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2, 6, 7

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 3

[27] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue. Dsod: Learning deeply supervised object detectors from scratch. In *ICCV*, 2017. 3, 7

[28] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016. 2, 3, 7

[29] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016. 3

[30] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. *arXiv preprint arXiv:1702.08502*, 2017. 2, 7

[31] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.