DSN: A New Deformable Subnetwork for Object Detection

Shuai Wu, Yong Xu*

***Abstract*—Although deep convolutional neural networks have achieved great success in object detection, they depend heavily on considerable training data and do not have a specific mechanism to handle related challenging problems, such as object deformation. In this paper, we design a Deformable Subnetwork (DSN) to introduce the Deformable Part-based Model (DPM) in the deep object detection framework. It is effective for handling object deformation and is composed of two significant parts: the deformation coefficient part and the deformation pooling part. The deformation coefficient part is responsible for generating the deformation coefficients for each position of the input. The deformation pooling part calculates the final score for each position which takes into account its displacement penalty relative to the root position. DSN is convenient for being embedded into the most prevalent object detection frameworks such as Faster-RCNN or RFCN. More importantly, it does not impair the integrity of the original framework and only cause little time consumption. We show effectiveness of DSN via experiments on the PASCAL VOC and COCO datasets, achieving the state-of-the-art results, 82.7% for PASCAL VOC and 32.1% for COCO.**

***Index Terms*—object detection, deformable subnetwork, deformation coefficient, deformation pooling.**

I. Introduction

RECENTLY, Deep Convolutional Neural Networks (DCNNs) have almost dominated computer vision in many fields such as image classification [1-5], object detection [6-9] and semantic segmentation [10-13]. In particular for object detection, proposal-based deep models are currently the leading methods [14-16]. They always show powerful performance on both object classification and bounding box regression. This is mainly due to the Region Rroposal Network (RPN) and the Region of Interest (ROI) pooling layer [17]. RPN first generates hundreds of candidate proposals. Then, these proposals are fed into the ROI pooling layer to further extract features. Although proposal-based models yield excellent results, they are limited in handling object deformation which is considered a key challenging problem for object detection [18]. Objects show different appearances due to various poses and viewpoints, especially for nonrigid objects. This is because in the real world, the positions of object parts are not fixed but are changeable. How to model part alignment is a key point for handling object deformation. However, ROI pooling acts as an absolute feature extractor and ignores the part alignment process. It separates a candidate proposal into fixed spatial bins and calculates the average value for each bin. Each position is equally treated in one bin, and this is adverse for part alignment.

The work is supported by the National Natural Science Foundation of China (Grant No. 61876051), (Corresponding author: Yong Xu.)

Shuai Wu is with the Bio-Computing Research Center, Harbin Institute of Technology, ShenZhen (Email: shuaiwu9@gmail.com;)

Yong Xu is with the Bio-Computing Research Center, Harbin Institute of Technology, ShenZhen (Email: yongxu@ymail.com;)

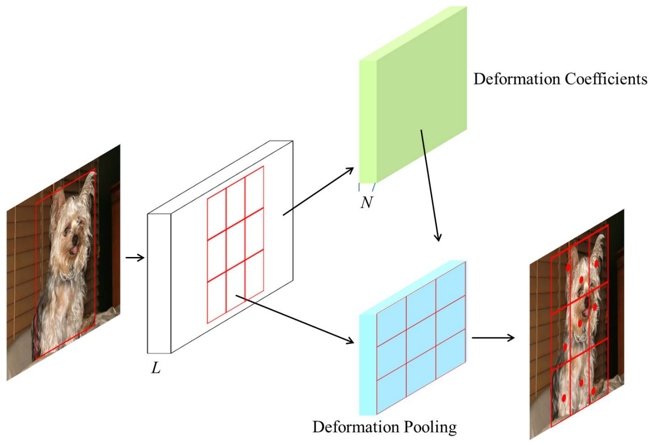


Fig. 1: The whole framework of the deformable subnetwork. It consists of two key parts: the deformation coefficient part and the deformation pooling part.

The Deformable Part-based Model (DPM) [19] is highly effective in addressing the object deformation problem. Before the deep convolutional networks are introduced into object detection, DPM is considered one of the most prevalent methods in object detection. It models object deformation by optimizing a star-structure representation. DPM consists of a root filter and a set of parts filters. Object matching is a part alignment process. DPM will find the best position for each part taking into account its displacement penalty relative to the anchor position. The anchor position for each part comes from a template that predefines the relative positions between the root and the parts. In real applications, DPM always applies dozens of templates for an object category because of its ever-changing appearance. Few studies have attempted to integrate the DPM with deep convolutional neural networks to handle object deformation and further improve object detection performance [20-24]. These methods can be called DPM-based deep models. Although these methods introduce DPM in deep networks to handle object deformation problem, they have two main drawbacks. First, most methods only consider the Convolution Neural Network (CNN) as a feature extraction tool, and object detection is divided into several independent processes. In other words, the DPM is not really embedded into the deep detection framework. Second, to introduce the DPM into the CNN, the detection models in these methods become very complex. This makes the models time consuming, and it is unclear how to train them end-to-end. Recently, the convolutional network has become increasingly deeper. Not only is detection accuracy important, but the detection efficiency is also significant. The above two drawbacks make DPM-based deep models hardly to be applied in real world.

Based on the above observations, we propose a Deformable Subnetwork (DSN), which is convenient for introducing DPM in deep detection models such as RFCN and Faster-RCNN. It can effectively handle object deformation and can improve the detection accuracy. More importantly, it does not impair the integrity of the original models and only consumes little time. Fig.1 illustrates the whole structure of the DSN, which consists of two key parts: the deformation coefficient part and the deformation pooling part. The deformation coefficient part is responsible for generating deformation coefficients corresponding to the displacement vector. Each position of the input is assigned N deformation coefficients to calculate its displacement penalty. These coefficients are generated online from the convolutional network. This is different from the original DPM in which deformation coefficients for each template are fixed after training. Generating deformation coefficients online is based on the following considerations. Although the DPM applies dozens of templates to model an object category, in real world, they are far from covering all situations. Moreover, more templates will greatly increase the computational cost. Generating coefficients online can be considered as generating fictitious templates for the objects of a given image. This is not only more robust but also more efficient than the original DPM. The deformation pooling part aims at seeking the most appropriate position for each part of an object. It first separates a candidate proposal into several bins relative to different parts. Then inside each bin, each neuron is penalized by its displacement vector relative to the anchor position. Finally, the position of the maximum value is considered the part-matching position as Fig. 1 illustrates. This is consistent with part alignment process in the DPM model and can help make the deep detection framework robust to moderate object deformation. Most DPM-based methods always apply varying size filters to represent different parts of an object. This is time-consuming and redundant for deep networks. According to [25], a neuron in the top layers of the deep convolution networks can completely represent a small fraction of the original image. Therefore, the deformation pooling acts on different neurons to perform part alignment which is more efficient. Normally, the center of each bin is set as the anchor position when performing part alignment. To further improve the detection performance, we also design deformation anchor part to generate anchor position for each bin from the deep network which is more reasonable and can get better results.

In summary, our main contributions are as follows:

1. We propose a deformable subnetwork which can introduce DPM model in deep neural network. It can generate 4 dimensional deformation coefficients and perform part alignment to handle object deformation.

2. To make the model more reasonable and robust, we also design deformation anchor part to generate anchor position for each bin from the deep network.

3. Our deformable subnetwork is convenient to be embeded into most prevalent proposal based frameworks. It does not break down the integrity of the original deep detection frameworks and only consumes little time.

4. We achieve the state-of-the-art results on both PASCAL and COCO benchmarks.

II. Related work

Recently, deep network based methods can achieve excellent performance in object detection. These methods can be divided into two categories: proposal-based methods and regression-based methods. The proposal-based methods need first generate hundreds of candidate proposals and then perform classification and bounding box regression on them. Faster-RCNN [17] is considered the most typical proposal-based model. It applies RPN, which is both efficient and accurate, to generate candidate proposals. Many improvements from different perspectives have been made on Faster-RCNN, such as FPN [26], ION [27] and Hypernet [28], but the basic framework is preserved. RFCN [29] designs the position-sensitive pooling layer to make the whole network fully convolutionally connected, which dramatically improves the detection efficiency. Mask-RCNN [30] designs to simultaneously train the network with both detection and segmentation tasks and achieves a large breakthrough. The regression-based methods do not generate candidate proposals and consider object detection as a one-step regression process. Szegedy et al. [31] train a deep network to generate a bitmap with the object mask being one. Yoo et al. [32] consider object detection as a corner point regression process. They train the network to make the initial top left point and the bottom right point move to the ground truth boxes. Yolo [33] and SSD [34] are proposed to achieve real-time detection efficiency. In their final feature maps, each neuron is responsible for directly predicting a fixed number of anchor boxes.

The Deformable Subnetwork (DSN) is designed for proposal-based models. The deformation pooling layer need act on different candidate proposals and perform part alignment to handle object deformation. Some other researches have also attempted to settle object deformation in deep network. In the following, we briefly introduce these methods.

DPM-Based Deep Models : Several DPM based deep models have attempted to integrate DPM with deep network. Savalle et al. [20] replace the HOG [35-37] feature pyramid with a CNN feature pyramid to improve the DPM performance. Girshick et.al [21] and Wan et al. [22] attempt to realize the DPM in the CNN through a distance transform layer and the geometry filters. Ouyang et al. [23] introduce the DPM mechanism in the CNN to settle part deformation for pedestrian detection and further extend it to generic object detection [24].

Spatial Transform Network : The Spatial Transform Network (STN) [38, 39] aims to perform spatial transform online through the deep convolutional network. An additional branch is added to the base network to generate the transform parameters for each position. Then, these parameters are utilized to perform spatial transform across the input. The whole framework can be trained end-to-end, which means that the transform parameters can be completely trained from the training data. STN can achieve good results on handwriting recognition. However, it is limited to generic object detection. In the real world, the ever-changing appearance of objects is too complex to model only by spatial transform.

Deformable Convolution Network : The Deformable Convolution Network (DCN) [40] aims to model geometric transformations by applying non-regularly shaped convolution and pooling. It designs two inspiring structures: deformable convolution and deformable ROI pooling. Deformable convolution applies an additional convolution branch to generate the deformation offset for each position of the input layer. Then, the convolution is no longer regularly shaped. Deformable ROI pooling applies an additional fully connected structure to generate an offset for each ROI bin. At first blush, DSN is similar to this DCN model. However, the deformation pooling part of DSN aims to calculate the displacement penalty for different positions in the ROI region and perform part alignment for each ROI bin. This is consistent with the idea of the DPM model and is completely different from DCN.

III. DEFORMABLE SUBNETWORK

This section presents the details of the Deformable Subnetwork (DSN). DSN introduces DPM in the deep detection model to address the object deformation problem. It consists of two main parts: the deformation coefficient part and the deformation pooling part. These two parts play different roles but are strongly related to each other.

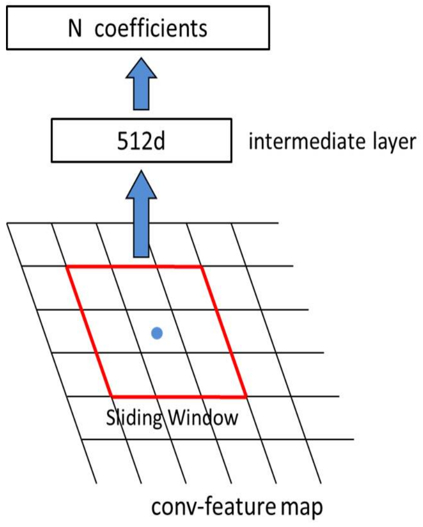


Fig. 2: The detail structure of the deformation coefficient part.

A. The Deformation Coefficient Part

Fig. 2 illustrates the details of the deformation coefficient part. A small network is slided on the input to generate the deformation coefficients for each position. First, an $n \times n$ spatial window centered at a position is mapped to a lower-dimensional feature vector (512-d for both VGG16 and ResNet-101). Then, this low-dimensional feature vector is fed into a fully connected layer to generate N coefficients. These coefficients are exploited to calculate the displacement penalty for the position. This small network operates in a sliding fashion so that the size of the output feature map is the same as that of the input feature map. Generally, such a design can be easily implemented by an $n \times n$ convolutional layer with ReLU layer followed and a 1×1 convolutional layer with N outputs. To take advantage of the context information for a single position, n is always larger than 1, and in this paper, we set n as 3. The value of N needs to correspond to the size of the displacement vector. To be consistent with the DPM model, DSN finally exploits the quadratic function, which has 4 values to define the displacement vector. Thus, the number of N is correspondingly set to 4. The output coefficients are denoted as $w_{ij}, i \in \{1, ..., N\}, j \in \{1, ..., W \times H\}$, where W and H represent the width and height of the output feature map respectively.

B. The Deformation Pooling Part

The deformation pooling part is based on candidate proposals. The whole deformation pooling process performs part alignment and generates a more robust representation for different candidate proposals. In this paper, we embed DSN into Faster-RCNN and RFCN in which the candidate proposals come from an effective generator called RPN [17]. DSN implements the deformation pooling part by a newly designed layer called the deformation pooling layer. This layer has the following three inputs: 1. the candidate proposals from RPN; 2. the intermediate feature map L; 3. the coefficient feature map from the deformation coefficient part.

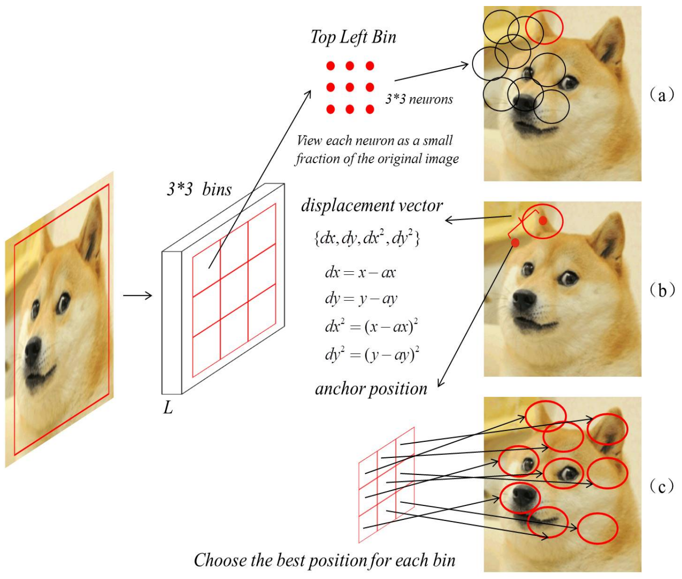


Fig. 3: The deformation pooling part: the process of deformation pooling is to perform part alignment for each bin.

Fig. 3 visualizes the whole process of the deformation pooling layer. It first maps a candidate proposal on the intermediate feature map L and then partitions the corresponding region into $k \times k$ bins. Each bin is responsible for predicting a latent part (e.g. the left top bin is responsible for predicting the dog ear in

Fig. 3). According to [25], in a deep convolutional network, although the receptive field of top layers is very large, the effective receptive field occupies only a small fraction. Thus, a neuron on L is completely capable for representing an object part. Owing to object deformation, different neurons in one bin are considered as all the possible positions for the latent part as Fig. 3 (a) illustrates. The deformation pooling aims to find the correct position for the latent part. This is consistent with the idea of DPM. For one bin, deformation pooling first calculates the displacement vector $\{dx, dy, dx^2, dy^2\}$ for each neuron as (1).

$$\begin{aligned} dx &= x - ax \\ dy &= y - ay \\ dx^2 &= (x - ax)^2 \\ dy^2 &= (y - ay)^2 \end{aligned} \tag{1}$$

where (ax, ay) represents the anchor position of the bin. Normally, we set the center position as the bin's root position. (x, y) is the neuron's position, and it is also considered as the center position of the neuron's effective receptive field, as illustrated in Fig. 3 (b). Once the displacement vector for a neuron is obtained, the corresponding deformation coefficients are integrated to calculate its displacement penalty by (2), where p is the position index for a neuron, and f_p represents the neuron value at position p. $\{w_{1p}, w_{2p}, w_{3p}, w_{4p}\}$ represents the corresponding deformation coefficients from the deformation coefficient part. λ is the predefined scalar to modulate the magnitude of the deformation costs. Then, each neuron is penalized by its displacement penalty and f_p^d denotes the result. Finally, the maximum value f^* of one bin is set as its output as in (3). This means that the position of the maximum value is considered the matching position of the latent part. The deformation pooling aims to find the best position of the latent part for each bin, as illustrated in Fig. 3 (c).

$$f_p^d = f_p + \lambda(w_{1p}dx + w_{2p}dy + w_{3p}dx^2 + w_{4p}dy^2) \tag{2}$$

$$f^* = \max_{p \in bin}(f_p^d) \tag{3}$$

Formulas (2) and (3) indicate that deformation pooling includes max ROI pooling. If the deformation coefficients are all set to zero, deformation pooling is equal to max ROI pooling. There are three inputs for our deformation pooling layer. Except for the candidate proposals, the other two inputs need to perform back propagation. The specific formulations are as follows.

$$\frac{\partial f_p^d}{\partial f_p} = \frac{\partial(f_p + \lambda(w_{1p}dx + w_{2p}dy + w_{3p}dx^2 + w_{4p}dy^2))}{\partial f_p} = 1 \tag{4}$$

$$\begin{aligned} \frac{\partial f_p^d}{\partial w_{1p}} &= \frac{\partial(f_p + \lambda(w_{1p}dx + w_{2p}dy + w_{3p}dx^2 + w_{4p}dy^2))}{\partial w_{1p}} = \lambda dx \\ \frac{\partial f_p^d}{\partial w_{2p}} &= \frac{\partial(f_p + \lambda(w_{1p}dx + w_{2p}dy + w_{3p}dx^2 + w_{4p}dy^2))}{\partial w_{2p}} = \lambda dy \\ \frac{\partial f_p^d}{\partial w_{3p}} &= \frac{\partial(f_p + \lambda(w_{1p}dx + w_{2p}dy + w_{3p}dx^2 + w_{4p}dy^2))}{\partial w_{3p}} = \lambda dx^2 \\ \frac{\partial f_p^d}{\partial w_{4p}} &= \frac{\partial(f_p + \lambda(w_{1p}dx + w_{2p}dy + w_{3p}dx^2 + w_{4p}dy^2))}{\partial w_{4p}} = \lambda dy^2 \end{aligned} \tag{5}$$

The backward process for the deformation pooling layer is simple and consumes little time at the training stage.

C. Understanding The Deformation Subnetwork

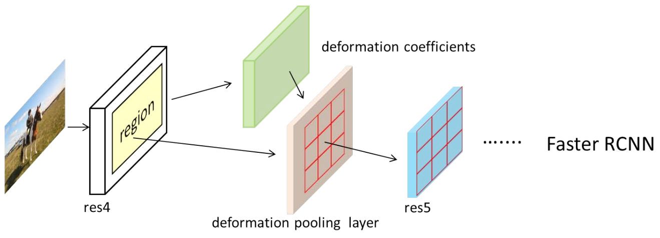
The deformation coefficient part and the deformation pooling part are based closely on each other. They jointly introduce DPM in the deep neural network. As illustrated in Fig. 1, they both take the intermetiate layer L as the input. In terms of one neuron on L, the deformation coefficient part is responsible for assigning it N (4 in this paper) coefficients. Such coefficients are of great significance in calculating the neuron's displacement penalty. In the deformation pooling part, once a candidate region is located on L, every neuron inside the region is viewed as a possible position of the latent parts for an object. The deformation pooling makes use of the clustering concept, which makes one bin responsible for predicting one latent part. Then, for one bin, all the neurons are considered as the possible positions of the latent part and will be penalized by their displacement penalty as in (2).

Formulas (2) and (3) indicate that the deformation coefficients are the crucial factor that finally determines which neuron is the best position of the latent part. In the original DPM, the deformation coefficients are tied together with the template and are fixed once training is complete. Although DPM applies multiple templates to represent an object category, they are still limited in their ability to cover all the ever-changing appearances. Moreover, more templates result in considerable computational costs. DSN generates deformation coefficients online from the convolutional network. Actually, the deformation coefficient part can be considered as generating fictitious templates online for the objects of a given image. The deformation pooling is used to perform part alignment according to the fictitious templates. In the deformation pooling step, one bin corresponds to one latent part of an object. However, the corresponding relationship is not fixed, and it is determined by the fictitious templates, which are independent of the candidate proposals. Thus, for one neuron, the deformation coefficients are only related to its contextual information. RPN generates thousands of candidate proposals with various sizes and positions. A neuron may have different displacement vectors relative to different proposals. However, its deformation coefficients do not change with the candidate proposals. They are totally trained from the data and converge to the ground truth box. Thus, DSN is more effective and robust than DPM. Moreover, it is very beneficial for obtaining the optimal bounding box for an object and consumes little time.

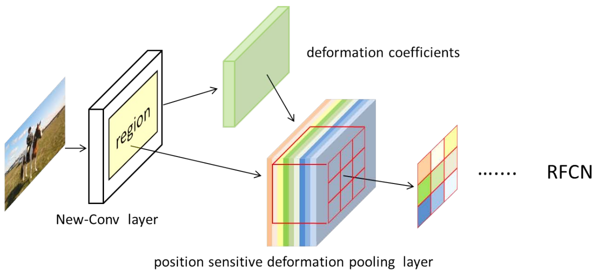
D. The Baseline Framework

It is convenient for the deformable subnetwork to be embedded in the prevalent proposal-based frameworks for object detection. In this paper, we choose Faster-RCNN and RFCN to verify the effectiveness of the deformable subnetwork. Fig. 4 illustrates the specific structure.

In terms of Faster-RCNN, we apply VGG16 [41] and ResNet101 [42] as the backbone network. We follow the design in [17] and set the last layer of res4 (conv5 for VGG16)



(a) Deformable subnetwork with Faster-RCNN



(b) Deformable subnetwork with RFCN

Fig. 4: The Deformable subnetwork with Faster-RCNN and RFCN

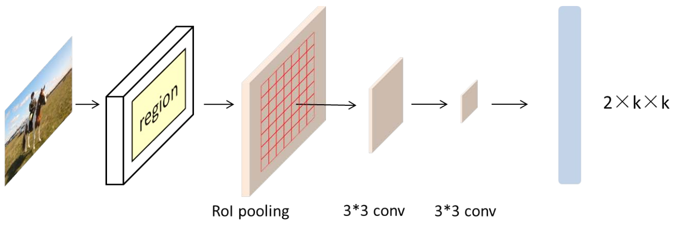
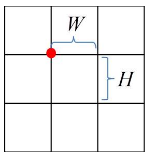


Fig. 5: Generating anchor position for each bin from the network

as the intermediate feature map L. In other words, this layer is set as the input layer for the deformation coefficient part and the deformation pooling part. Then, the deformation pooling output is fed into res5 (conv5 for VGG16), which finally generates the confidence score and bounding box regression results. For RFCN, we only apply ResNet101 [42] as its backbone network. RFCN designs a position-sensitive pooling layer that makes the whole framework fully convolutional. The core idea of the position-sensitive pooling layer is that the $k \times k$ bins do not originate from a single feature map but from $k \times k$ corresponding feature maps. The deformable subnetwork is very appropriate for such design because the deformation pooling for each bin is relatively independent. Thus, for RFCN, DSN performs the deformation pooling on $k \times k$ feature maps. We call this design the "position-sensitive deformation pooling layer". Moreover, RFCN applies the "hole algorithm" [11] to ensure that res4 and res5 have the same size. Then it adds a new convolutional layer after res5 which we set as the input of our deformation subnetwork. Finally, similar to RFCN, the output of the position sensitive deformation pooling layer votes for the confidence score. This is mainly implemented by averaging the output for each bin.

E. Generating Anchor Positions from The Network

The deformation pooling layer sets the center position for each bin as its anchor position, this is inspired by [24].



$$B_x^a = B_x^s + \sigma_x \times W$$
$$B_y^a = B_y^s + \sigma_y \times H$$

Fig. 6: Calculating the anchor position for each bin

To further strengthen the performance and rationality of the deformable subnetwork, we design an improved version in which the anchor position for each bin is generated from the network and is totally trained from the dataset. In this improved version, a new additional branch is added to the deformation subnetwork. It is mainly responsible for generating the anchor position for each bin and is called the deformation anchor part.

Fig. 5 shows the detailed structure of the deformation anchor part. In terms of a candidate proposal, ROI pooling is first applied to extract features. Then, the output will be fed into two 3×3 convolutional layers and one fully connected layer to generate a $2k^2$ sized vector for $k \times k$ bins. Each bin corresponds to 2 outputs σ_x and σ_y which will be finally transformed to the anchor position. We apply the sigmoid activation function after the fully connected layer, so the outputs are all between 0 and 1. Fig. 6 illustrates the detailed process to calculate the anchor position for each bin. (B_x^s, B_y^s) represents the coordinate of the left top neuron for each bin. H and W represent the height and width for each bin. (B_x^a, B_y^a) stands for the final anchor position for each bin. This improved version makes our deformation subnetwork more reasonable, and generating anchors can help the model better represent an object.

IV. Experiments

A. Experimental Setup

DSN is embedded in Faster-RCNN and RFCN and is evaluated on PASCAL VOC [44] and COCO [45] datasets.

In terms of PASCAL VOC, the mAP scores with 0.5 IoU threshold are reported. For Faster-RCNN, the joint training strategy is applied [17] to train the whole model. The images are resized to have a shorter side of 600 pixels. Considering RPN and the detection network, 256 and 128 proposals are sampled respectively. For deformation pooling, 14×14 bins are are adopted. A total of $110k$ iterations are performed on 2 GPUs, $80k$ for the 0.001 learning rate, and $30k$ for 0.0001 learning rate. The momentum and weight decay are set to 0.9 and 0.0001 respectively. For RFCN, we follow the training strategy in [26] and exploit pretrained and fixed proposals to train the whole model. Such proposals are generated from the first stage of the procedure in [17]. 7×7 bins are utilized in the deformation pooling. $120k$ iterations are performed on 2 GPUs, $80k$ for 0.001 learning rate, and $40k$ for 0.0001 learning rate. The momentum and weight decay are set the same values as those of Faster-RCNN.

For COCO, mAP@[0.5:0.95] are exploited as the final evaluation index. For both Faster-RCNN and RFCN, 7×7 bins are utilized in deformation pooling. The momentum and weight decay are set to 0.9 and 0.0001 respectively. Tow GPUs are used to perform $960k$ iterations, and the learning rates are

TABLE I: Detailed comparisons on PASCAL VOC 2007 test in average precision (%). The training data is the union set of *07 trainval* and *12 trainval*.

Method	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	Tv
C-DPM [20]	48.2	50.9	64.4	43.4	29.8	40.3	56.9	58.6	46.3	33.3	40.5	47.3	43.4	65.2	60.5	42.2	31.4	35.2	54.5	61.6	58.6
DeepID [24]	64.1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
SSD 513 [43]	80.6	84.3	87.6	82.6	71.6	59.0	88.2	88.1	89.3	64.4	85.6	76.2	88.5	88.9	87.5	83.0	53.6	83.9	82.2	87.2	81.3
DSSD 513 [43]	81.5	86.6	86.2	82.6	74.9	62.5	89.0	**88.7**	88.8	65.2	87.0	**78.7**	88.2	89.0	**87.5**	83.7	51.1	**86.3**	81.6	85.7	**83.7**
HyperNet [28]	76.5	77.4	83.3	75	69.1	62.4	83.1	87.4	87.4	57.1	79.8	71.4	85.1	85.1	80	79.1	51.2	79.1	75.7	80.9	76.5
ION [27]	77.6	82.5	86.2	79.9	71.3	67.2	88.6	87.5	88.7	60.8	84.7	72.3	87.6	87.7	83.6	82.1	53.8	81.9	74.9	85.8	81.2
Faster-RCNN [42]	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	72.0
DCN-Faster-RCNN [40]	78.7	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
DSN-Faster-RCNN	79.5	82.7	85.2	79.0	72.2	66.4	88.3	87.7	88.0	64.9	86.1	72.4	88.2	87.9	84.1	79.7	51.4	81.4	79.3	84.8	80.4
DSN-A-Faster-RCNN	79.9	83.7	85.7	79.5	73.2	67.4	88.2	88.2	88.0	65.5	86.6	72.8	88.3	87.4	85.2	80.1	52.4	83.4	78.3	83.3	80.1
RFCN [29]	80.5	50.9	64.4	43.4	29.8	40.3	56.9	58.6	46.3	33.3	40.5	47.3	43.4	65.2	60.5	42.2	31.4	35.2	54.5	61.6	58.6
DCN-RFCN [40]	82.6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
DSN-RFCN	82.0	84.9	88.7	82.2	**76.0**	71.3	87.5	88.3	89.7	68.1	87.7	73.9	89.2	87.5	86.5	85.4	57.5	84.3	81.7	87.6	82.1
DSN-A-RFCN	**82.7**	**86.7**	**88.8**	**84.2**	75.6	**71.7**	87.5	88.6	**89.8**	**69.2**	**88.7**	76.1	**89.3**	**89.0**	86.9	**85.4**	**58.5**	84.7	**83.3**	**88.2**	81.5

TABLE II: Comparisons on PASCAL VOC 2012 test in average precision (%). 07++12 represents the union set of *07 trainval+test* and *12 trainval*. $^{\wr}$:http://host.robots.ox.ac.uk:8080/anonymous/HK3FIL.html †:http://host.robots.ox.ac.uk:8080/anonymous/SRLUTV.html ‡:http://host.robots.ox.ac.uk:8080/anonymous/LRZXXT.html

Method	Backbone Net	Training data	mAP
SSD [43]	ResNet-101	07++12	75.4
DSSD [43]	ResNet-101	07++12	76.3
Faster-RCNN [42]	ResNet-101	07++12	73.8
DSN-Faster-RCNN	ResNet-101	07++12	75.9$^{\wr}$
RFCN [29]	ResNet-101	07++12	77.6
DSN-RFCN	ResNet-101	07++12	78.3†
DSN-A-RFCN	ResNet-101	07++12	78.8‡

set as 0.0005 and 0.00005 in the first 2/3 and the last 1/3 iterations, respectively.

B. Experiments on PASCAL VOC

The experiments are performed on VOC 2007 and VOC 2012. The 2007 dataset is composed of approximately 5000 trainval images and 5000 test images over 20 object categories. The 2012 dataset contains more than 10 thousand trainval images. DSN-Faster-RCNN and DSN-RFCN represent the plain counterparts with the deformable subnetwork for Faster-RCNN and RFCN respectively. The improved version DSN whose anchor position are generated from the convolution network are also embedded in Faster-RCNN and RFCN and the corresponding models are denoted as DSN-A-Faster-RCNN and DSN-A-RFCN.

Table I reports the comparison results between the deformable subnetwork and other state-of-the-art methods. For a fair comparison, all the models are trained on VOC 07+12, which contains VOC 2007 and VOC 2012 trainval datasets. C-DPM [20] and DeepID [24] are the typical DPM-based deep models. These models are very complex, and the detection results are relatively lower than other methods. DSN-Faster-RCNN outperforms Faster-RCNN by 3.1%. Furthermore, DSN-Faster-RCNN also outperforms HyperNet and ION by 3.0% and 1.9% respectively. SSD [34] is the typical regression based method and DSSD [43] is its improved version. Our DSN-RFCN outperforms SSD by 1.4% and also has better performance than DSSD. DSN-RFCN also outperforms RFCN by 1.5%. DCN [40] is a similar work that can handle object deformation, it can also be applied to Faster-RCNN and RFCN. Although DSN-RFCN obtains lower result than DCN-RFCN, DSN-A-RFCN can slightly outperform DCN-RFCN. Moreover, DSN-Faster-RCNN and DSN-A-Faster-RCNN can both outperform DCN-Faster-RCNN by 0.8% and 1.2% respectively. Generating anchor positions from network can slightly improve the performance of DSN, DSN-A-Faster-RCNN and DSN-A-RFCN could outperform DSN-Faster-RCNN and DSN-RFCN by 0.4% and 0.7% respectively. Furthermore, DSN-A-RFCN can achieve best results on 13 categories, especially for nonrigid objects like dog, cat, horse, bird and people. our deformable subnetwork aims at introducing DPM in deep network to handle with object deformation. So it has powerful ability to detect nonrigid object and this has been proved by the experiments.

Table II shows the results of different methods on VOC 2012 test. All the models are trained on VOC 07++12 which contains the VOC 2007 test, VOC 2007 trainval and VOC 2012 trainval datasets. DSN-Faster-RCNN can outperform Faster-RCNN by 2.1%. DSN-A-RFCN obtains the best result, and can outperform SSD, DSSD and RFCN by 3.4%, 2.5% and 1.2% respectively. Table I and Table II clearly demonstrate that the deformable subnetwork can effectively improve the object detection performance. Not only can DSN-Faster-RCNN and DSN-RFCN outperform Faster-RCNN and RFCN, but they can also outperform other state-of-the-art methods such as SSD and DSSD.

Table III shows the detection efficiency of DSN and the related DCN [40]. All the results are acquired on a workstation with GeForce GTX 1080 GPU and Intel E5-2650 v4

TABLE IV: Comprehensive comparisons on PASCAL VOC 2007 test. The training datasets are *07 trainval* and *12 trainval*. Net: Backbone network. O: Online hard example mining. M: Multi-scale training

Method	Net	O	M	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	Tv
Faster-RCNN [17]	VGG16	no	no	73.2	76.5	79.0	70.9	65.5	52.1	83.1	86.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
DSN-Faster-RCNN	VGG16	no	no	**76.7**	**78.1**	**83.1**	**77.9**	**66.6**	**62.9**	**85.5**	**86.8**	**87.7**	**59.6**	**83.1**	**71.2**	**86.2**	**87.1**	**77.6**	**78.4**	**50.7**	**77.5**	**74.5**	**82.6**	**75.8**
Faster-RCNN [42]	Res101	no	no	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	87.8	56.7	85.8	69.4	88.3	86.9	80.9	78.4	43.7	80.6	**79.8**	**85.3**	72
DSN-Faster-RCNN	Res101	no	no	79.5	82.7	85.2	79.0	72.2	66.4	88.3	87.7	88.0	64.9	86.1	72.4	88.2	**87.9**	84.1	79.7	51.4	81.4	79.3	84.8	80.4
DSN-A-Faster-RCNN	Res101	no	no	**79.9**	**83.7**	**85.7**	**79.5**	**73.2**	**67.4**	**88.2**	**88.2**	**88.0**	**65.5**	**86.6**	**72.8**	**88.3**	87.4	**85.2**	**80.1**	**52.4**	**83.4**	78.3	83.3	**80.1**
Faster-RCNN [46]	Res101	yes	no	78.9	80.4	85.7	79.8	69.9	60.8	88.3	87.9	89.3	59.7	85.1	**76.5**	87.1	87.3	82.4	78.8	**53.7**	80.5	78.7	84.5	80.4
DSN-Faster-RCNN	Res101	yes	no	**80.5**	**82.7**	**85.7**	**80.0**	**71.3**	**66.4**	**88.3**	**87.9**	**89.4**	**63.8**	**86.1**	72.4	**88.2**	**87.9**	**84.1**	**79.7**	51.4	**81.4**	**79.3**	**84.8**	**80.4**
RFCN [29]	Res101	yes	no	79.5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
DSN-RFCN	Res101	yes	no	81.3	82.0	87.6	81.5	75.2	70.3	86.5	87.7	88.9	67.1	88.0	74.4	89.5	87.8	85.3	84.6	56.4	83.6	81.6	86.8	80.4
RFCN [29]	Res101	yes	yes	80.5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
DSN-RFCN	Res101	yes	yes	82.0	84.9	88.7	82.2	**76.0**	71.3	87.5	88.3	**89.7**	68.1	87.7	73.9	89.2	87.5	86.5	85.4	57.5	84.3	81.7	87.6	**82.1**
DSN-A-RFCN	Res101	yes	yes	**82.7**	**86.7**	**88.8**	**84.1**	75.5	**71.7**	**87.5**	**88.6**	89.3	**69.2**	**88.7**	**76.1**	**89.3**	**88.4**	**86.9**	**85.4**	**58.5**	**84.7**	**83.3**	**88.2**	81.5

TABLE III: Test efficiency comparisons. The net.forward time and runtime are the average processing time of VOC 2007. the backbone network is ResNet-101.

Method	mAP(%)	Net.forward (sec)	test time (sec)
Faster-RCNN	76.4	0.119	0.135
DSN-Faster-RCNN	79.5	0.126	0.141
DSN-A-Faster-RCNN	79.9	0.141	0.154
DCN-Faster-RCNN	78.7	0.142	0.155
RFCN	80.5	0.078	0.101
DSN-RFCN	82.0	0.086	0.110
DSN-A-RFCN	82.7	0.112	0.121
DCN-RFCN	82.6	0.113	0.121

CPU. The overall runtime includes image resizing, network forward, and postprocessing(e.g., NMS for object detection). For both Faster-RCNN and RFCN, it is obvious that DSN only causes a slightly loss in detection time but achieves efficient gains in detection accuracy. Compared to DCN, DSN is superior in terms of detection efficiency. This is reasonable, as we mentioned above, DCN has two key parts: deformable convolution and deformable ROI pooling. The deformable convolution generates an offset which contributes to irregularly sized convolution for each position of the input. This is similar with the deformation coefficient part of DSN in terms of computational cost. However, DCN performs deformable convolution on three layers (res5a, res5b, res5c), while DSN only generates deformation coefficients through one layer. Moreover, DCN exploits bilinear interpolation to handle with the fractional problem, this is more complex than DSN and is relatively time consuming. Generating the anchor position can help increase the detection accuracy, but it is to some extent time consuming. This is mainly because the deformation anchor part is on the basis of ROI pooling. Thus, the post process (two convolutional layers and a fully connected layer) need act on all the candidate proposals. This is relatively time consuming and DCN suffers the similar problem, because the deformable ROI pooling part in DCN is also on the basis of ROI pooling. Thus, we can see that DSN-A-RFCN has similar detection efficiency as DCN-RFCN. Although DSN-RFCN obtains relative lower accuracy than DCN-RFCN and DSN-A-RFCN, it is much faster. Thus, DSN-RFCN is a trade-off between the detection accuracy and the detection efficiency.

C. Ablation Study on VOC 2007

To further validate the efficiency of DSN, comprehensive experiments are performed on VOC 2007.

Comprehensive comparisons: Table IV clearly demonstrates that the models with DSN always achieve better performance than the corresponding counterparts in all situations. In terms of Faster-RCNN baseline, regardless of whether the backbone network is VGG16 or ResNet-101, DSN-Faster-RCNN always outperform Faster-RCNN by more than 3%. The deformable subnetwork has no effect on the application of online hard example mining (OHEM) [46] during training. After introducing online hard example mining (OHEM), DSN-Faster-RCNN still outperforms Faster-RCNN by 1.6%. Moreover, it is obvious that the deformable subnetwork can obtain better results on most of the 20 categories, especially for nonrigid object. DSN aims to introduce DPM model to handle object detection. Thus, it has more powerful capability for detecting nonrigid objects. The results in Table IV completely demonstrate this, because the deformable subnetwork always obtains better results for all nonrigid objects like dog, person and so on. As for RFCN baseline, DSN-RFCN can achieve the mAP of 81.3% and outperform RFCN by 1.8%. Multiscale training can be very helpful to improve the detection accuracy, and it is introduced to train the DSN. The images are resized in each training iteration such that their scales are randomly sampled from 400,500,600,700,800 pixels. However, a single scale of 600 pixels is still applied at the test stage. Thus, multiscale training will not add test-time costs. After introducing multiscale training, DSN-RFCN and DSN-A-RFCN can achieve obvious gain and outperform RFCN by 1.5% and 2.2% respectively.

TABLE V: Influence of the displacement vector to the deformable subnetwork

Method	Backbone Net	Deformation Cost	mAP
Faster-RCNN	ResNet-101	0 distance costs	76.4
DSN-Faster-RCNN	ResNet-101	4 distance costs	79.5
DSN-Faster-RCNN	ResNet-101	8 distance costs	78.7
RFCN	ResNet-101	0 distance costs	79
DSN-RFCN	ResNet-101	4 distance costs	81.3
DSN-RFCN	ResNet-101	8 distance costs	80.6

Visualization analysis: To better understand the deformable subnetwork, Fig. 7 visualizes the deformation pooling results. The left column of Fig. 7 illustrates the original images, the middle column illustrates the position of the maximum neuron before deformation pooling for each bin, and the right column illustrates the position of the maximum neuron after deformation pooling. The red solid circles represent the maximum neuron for each bin and the ellipses stand for their effective receptive field. Luo et al. [25] proposed that the

TABLE VI: Influence of parameter λ to the deformable subnetwork

Method	Net	data	ohem	MutiScale	mAP($\lambda = 0.2$)	mAP($\lambda = 0.3$)	mAP($\lambda = 0.5$)	mAP($\lambda = 0.7$)	mAP($\lambda = 1$)
DSN-Faster-RCNN	VGG16	712	no	no	75.75	**76.65**	76.1	76.06	76.23
DSN-Faster-RCNN	ResNet-101	712	no	no	78.63	**79.50**	78.99	78.97	79.09
DSN-Faster-RCNN	ResNet-101	712	yes	yes	80.17	**80.52**	80.03	80.36	79.58
DSN-RFCN	ResNet-101	712	yes	no	80.90	**81.30**	80.83	80.67	81.20
DSN-RFCN	ResNet-101	712	yes	yes	81.91	**82.00**	81.63	81.87	81.76

TABLE VII: Comparisons on COCO dataset using ResNet-101. The COCO-style AP is evaluated @$\in$ [0.5,0.95]

Method	training data	test data	AP@0.5	AP@0.5:0.9	AP(small)	AP(medium)	AP(large)
Faster-RCNN [42]	train	val	48.4	27.2	6.6	28.6	45
DSN-Faster-RCNN	train	val	48.7	**27.9**	7.8	30.2	44.3
RFCN [29]	train	val	48.9	27.5	8.7	30.3	42
DSN-RFCN	train	val	48.6	**28.6**	10.3	31.4	40.7
SSD [43]	trainval	test-dev	45.4	28.0	6.2	28.3	49.6
DSSD [43]	trainval	test-dev	46.1	28.0	7.4	28.1	47.6
RFCN [29]	trainval	test-dev	51.9	29.9	10.8	32.8	45.0
DSN-RFCN	trainval	test-dev	53.8	**31.6**	12.5	34.9	46.5
DSN-A-RFCN	trainval	test-dev	54.3	**32.1**	12.9	35.2	47.1

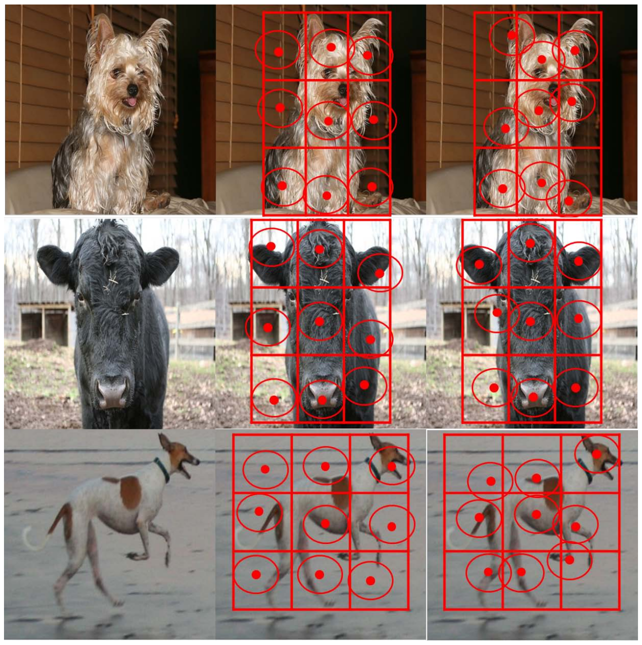


Fig. 7: Visualization Analysis. Left column: the original images. Middle column: the position of the maximum neuron before deformation pooling for each bin. Right column: the position of the maximum neuron after deformation pooling for each bin. The ellipses represent the effective receptive field centered at the maximum neurons

effective receptive field only occupy a small fraction of the receptive field. So we apply ellipse to approximately represent the effective field for a neuron. The effective field of the maximum value for each bin can be seen as different parts for an object. Fig. 7 clearly illustrates that the deformation pooling is beneficial for part alignment for each bin. Part alignment is the kernel idea of DPM and is considered an effective solution to handle object deformation. Each bin is responsible for predicting one latent part for an object, and the latent part is possible to be any position inside a bin. The deformation pooling is to find the right position for the latent part taking into account its displacement penalty relative to the anchor position. Therefore, DSN can effectively handle modest object deformation.

Influence of the displacement vector: DSN mainly applies a quadratic function $\{dx, dy, dx^2, dy^2\}$ to represent the displacement vector. However, this strategy is not fixed. In this paper, we attempted to extend the size of the displacement vector to 8 $\{dx, dy, dx^2, dy^2, \log dx, \log dy, \sqrt{dx^2 + dy^2}, \log\sqrt{dx^2 + dy^2}\}$, which means that the corresponding deformation coefficient channel number is extended to 8. Nevertheless, such an extension is not efficient in terms of improving the detection performance. In contrast, it causes accuracy losses on both Faster-RCNN and RFCN frameworks, as illustrated in Table V. More kinds of displacement may sometimes introduce redundant information, which is unfavorable for the model in the training process.

The influence of λ: The parameter λ in the deformation pooling layer is of great significance. It not only has a influence on the deformation pooling results, but also determines the learning rate of the deformation coefficient feature map. Table VI illustrates the detection results of the deformable subnetwork under different λ. The results slightly change with the value of λ under different situations, and the deformable subnetwork can will always get the best results on $\lambda = 0.3$.

D. Experiment on COCO

The deformable subnetwork is also evaluated on the COCO dataset. The experiments involve an $80k$ training set, a $40k$ val set, and a $20k$ test-dev set. Table VII shows the specific results. First, we set the $80k$ train set as the training data and set $40k$ val as the test data. mAP(@0.5:0.9) is applied as the final evaluation index. DSN-Faster-RCNN and DSN-RFCN outperform Faster-RCNN and RFCN by 0.7% and 1.1% respectively.

Second, we set both the $80k$ train set and the $40k$ val set as the training data and evaluate the deformable sub-network on the $20k$ test-dev set. In this situation, the deformable subnetwork is also compared with SSD and DSSD, and it can always get better performance in terms of AP(@0.5:0.9) as illustrated in Table VII. DSN-RFCN outperforms RFCN, SSD and DSSD by 1.7%, 3.6% and 3.6% respectively. DSN-A-RFCN can slightly improve the detection accuracy of DSN-RFCN by 0.5%. The experiments on COCO datasets can also demonstrate that the deformable subnetwork is effective. It can achieve great accuracy gain on Faster-RCNN and RFCN.

V. Conclusion

In this paper, we propose a deformable subnetwork which is consistent with the idea of the DPM model. It aims to introduce the DPM technique into the deep network to handle object deformation. Moreover, it is succinct and is very convenient for being embedded in the popular proposal based frameworks. The experimental results on both the PASCAL VOC and COCO datasets show that the deformable subnetwork can efficiently improve the performance of Faster-RFCN and RFCN and only consumes little time.

References

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

[2] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[3] J. Wen, Y. Xu, and H. Liu, "Incomplete multiview spectral clustering with adaptive graph learning," *IEEE transactions on cybernetics*, 2018.

[4] J. Wen, X. Fang, J. Cui, L. Fei, K. Yan, Y. Chen, and Y. Xu, "Robust sparse linear discriminant analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[5] J. Wen, Y. Xu, Z. Li, Z. Ma, and Y. Xu, "Inter-class sparsity based discriminative least square regression," *Neural Networks*, vol. 102, pp. 36–47, 2018.

[6] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[7] A. Jazayeri, H. Cai, J. Y. Zheng, and M. Tuceryan, "Vehicle detection and tracking in car video based on motion model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 583–595, 2011.

[8] X. Zhou, C. Yang, and W. Yu, "Moving object detection by detecting contiguous outliers in the low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 597–610, 2013.

[9] O. Oreifej, X. Li, and M. Shah, "Simultaneous video stabilization and moving object detection in turbulence," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 2, pp. 450–462, 2013.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[11] 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*, 2015, pp. 3431–3440.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[13] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[14] 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*, 2014, pp. 580–587.

[15] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[16] 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*. Springer, 2014, pp. 346–361.

[17] 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*, 2015, pp. 91–99.

[18] T. Mordan, N. Thome, M. Cord, and G. Henaff, "Deformable part-based fully convolutional network for object detection," *arXiv preprint arXiv:1707.06175*, 2017.

[19] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[20] P.-A. Savalle, S. Tsogkas, G. Papandreou, and I. Kokkinos, "Deformable part models with cnn features," in *European Conference on Computer Vision, Parts and Attributes Workshop*, 2014.

[21] R. Girshick, F. Iandola, T. Darrell, and J. Malik, "Deformable part models are convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2015, pp. 437–446.

[22] L. Wan, D. Eigen, and R. Fergus, "End-to-end integration of a convolution network, deformable parts model and non-maximum suppression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 851–859.

[23] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2056–2063.

[24] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy *et al.*, "Deepid-net: Deformable deep convolutional neural networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2403–2412.

[25] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 4898–4906.

[26] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection." in *CVPR*, vol. 1, no. 2, 2017, p. 4.

[27] 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*, 2016, pp. 2874–2883.

[28] 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*, 2016, pp. 845–853.

[29] 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*, 2016, pp. 379–387.

[30] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2980–2988.

[31] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in neural information processing systems*, 2013, pp. 2553–2561.

[32] D. Yoo, S. Park, J.-Y. Lee, A. S. Paek, and I. So Kweon, "Attentionnet: Aggregating weak directions for accurate object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2659–2667.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[34] 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*. Springer, 2016, pp. 21–37.

[35] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[36] X. Wang, T. X. Han, and S. Yan, "An hog-lbp human detector with partial occlusion handling," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 32–39.

[37] J. Wen, Z. Zhong, Z. Zhang, L. Fei, Z. Lai, and R. Chen, "Adaptive locality preserving regression," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[38] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.

[39] C.-H. Lin and S. Lucey, "Inverse compositional spatial transformer networks," *arXiv preprint arXiv:1612.03897*, 2016.

[40] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," *CoRR, abs/1703.06211*, vol. 1, no. 2, p. 3, 2017.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[42] 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*, 2016, pp. 770–778.

[43] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.

[44] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[45] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick, "Microsoft coco: Common objects in context," vol. 8693, pp. 740–755, 2014.

[46] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 761–769.