Deformable Convolutional Networks

이다경

INDEX

1. Abstract

- CNN feature structure가 고정되어 있기 때문에 모델 변형이 제한
- 이에 본 논문은 CNN 변환 모델링 기능 두 가지를 소개
  - deformable convolution
  - deformable RoI pooling

- 추가적인 offset으로 정사각형이 아닌 직사각형 형태를 만들자

Convolution → Convolution + learnable offset
RoI pooling → RoI pooling + learnable offset

without additional supervision

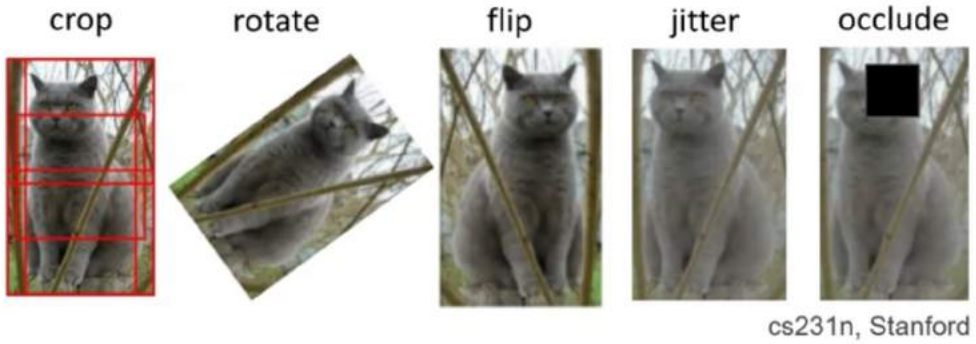
- 이 새로운 모듈들은 기존의 CNN모듈을 쉽게 대체할 수 있고,
standard backpropagation 로 end-to-end로 쉽게 학습할 수 있다.

- object detection and semantic segmentation 에 효과적이다.

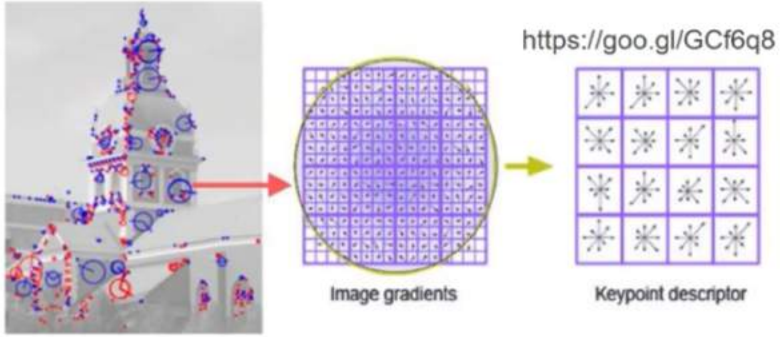
2. Introduction

- Visual recognition의 주요 이슈는 object scale, pose, viewpoint, and part deformation을 어떻게 수용하는가

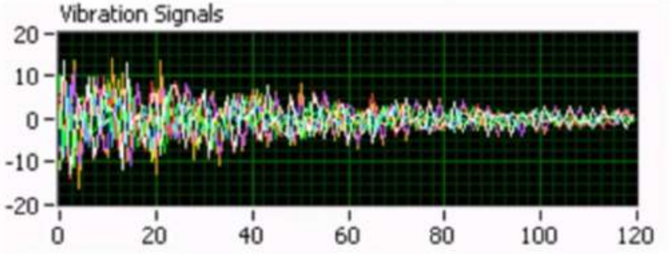
- data augmentation



- SIFT (scale invariant feature)



- Label-preserving augmentation?



https://goo.gl/fKvx8V

Ex) 암 detection은 scale x → 사전 지식이 필요함
이는 generalization에 한계

2. Introduction

- CNN모델은 이미 visual recognition tasks, such as image classification, semantic segmentation, and object detection에서 뛰어난 성능을 보이고 있지만, 다음과 같은 문제 존재

배경을 detection할 때도, 작은 물체를 큰 물체를 detection할 때도 같은 사이즈의 receptive filter

2. Introduction

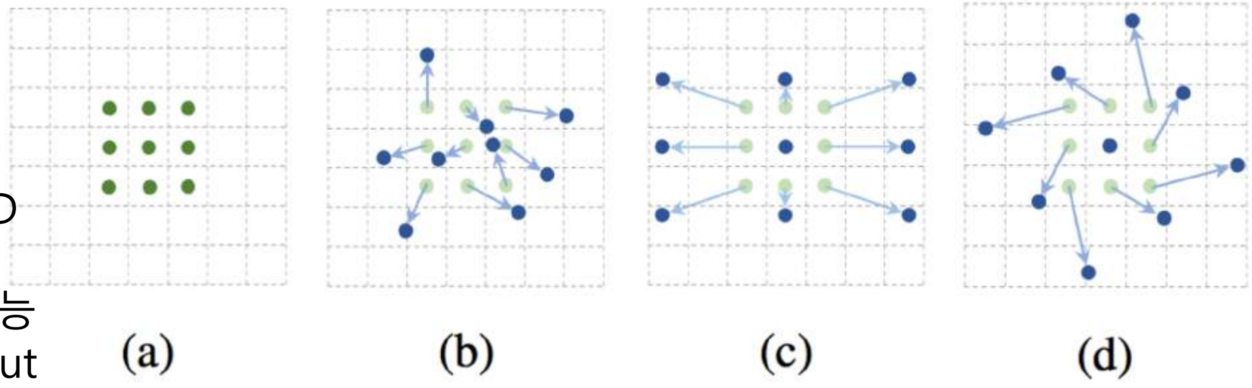
- 본 논문은 이를 해결하기 위한 두 가지 방법 제시

1. deformable convolution : standard convolution에 2D offsets 추가

- Dilated convolution, rotation convolution 등 커버 가능

→ 사람이 receptive size를 지정해 주는 것이 아니라, input에 따라 알아서 학습

2. deformable RoI pooling : 위와 같은 방식



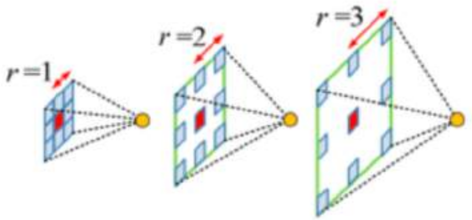
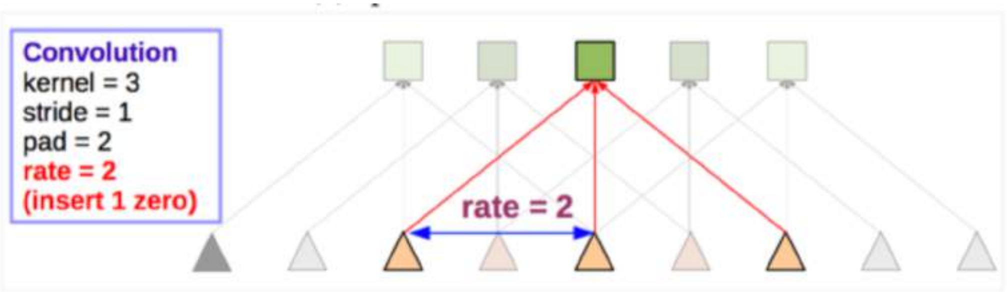


Figure 3. "Atrous" convolutions with $r = 1$, 2, and 3. The first convolution ($r = 1$) is actually the ordinary convolution.

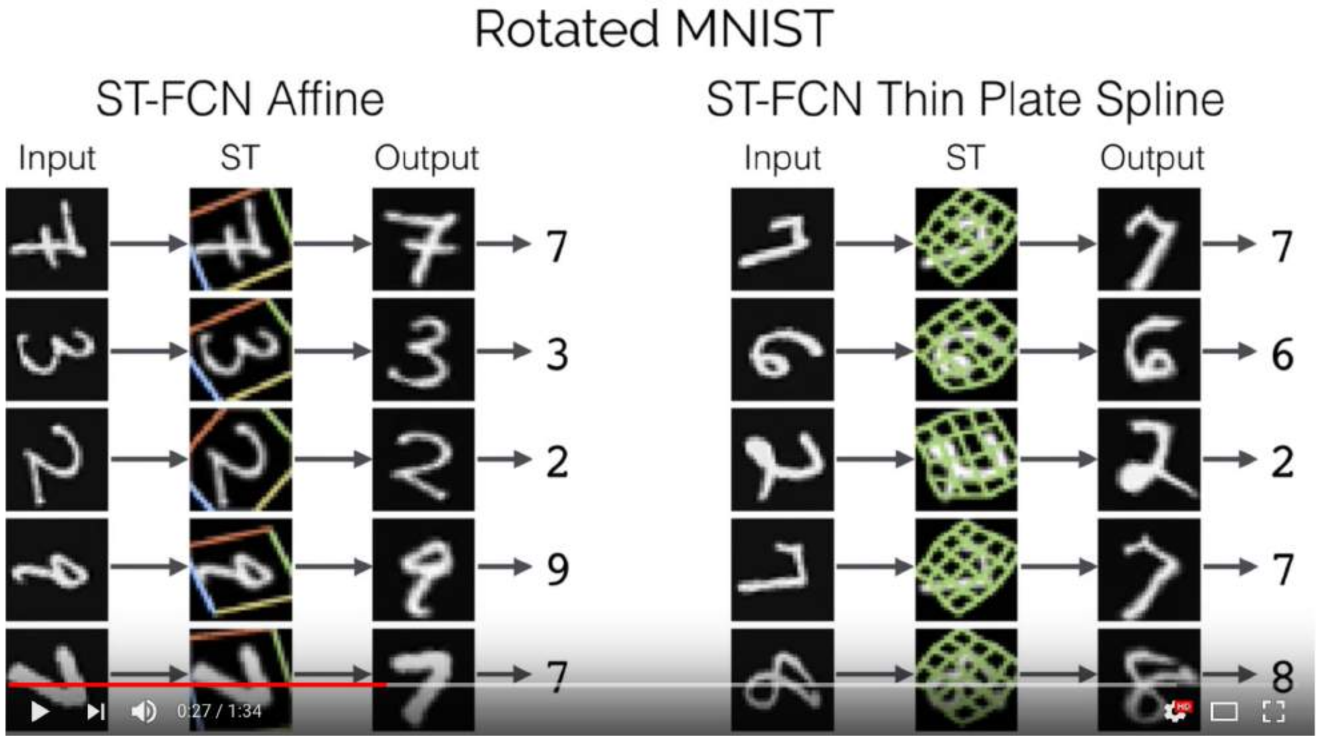


Dilated Convolution

2. Introduction

- 본 논문은 spatial transform networks 와 철학 공유

  https://www.youtube.com/watch?v=Ywv0Xi2-14Y



Deformable Convolution도 input에서 관심있는 영역을 detection.

But spatial transform networks처럼 가운데로 돌리고 다시 classification하는게 아니라 end-to-end로 바로 가능

Input이 구석에 있거나, 돌아가 있는 것을 가운데로 맞추어 인식

3. Deformable Convolutional Networks

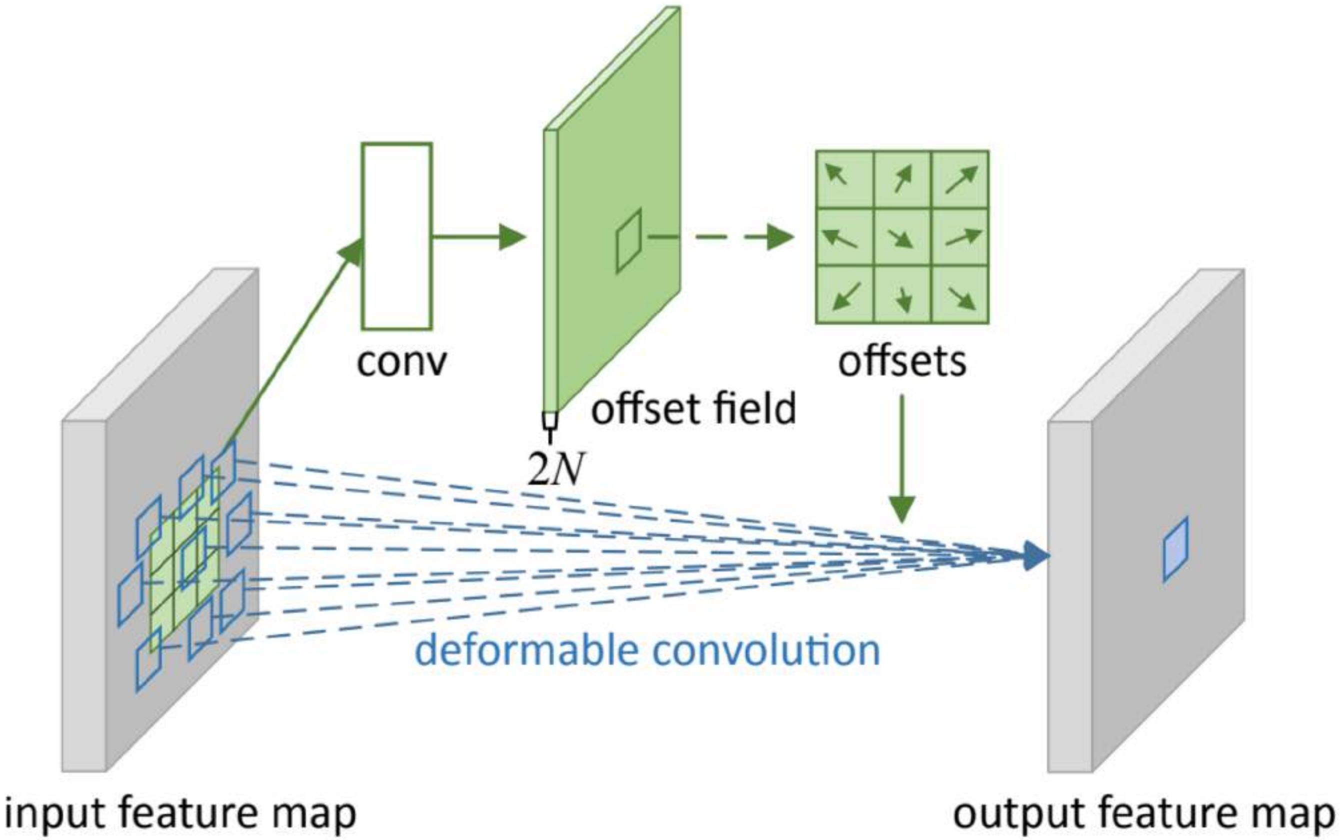


Figure 2: Illustration of 3×3 deformable convolution.

3. Deformable Convolutional Networks

3.1 Deformable Convolution

2.1. Deformable Convolution

The 2D convolution consists of two steps: 1) sampling using a regular grid $\mathcal{R}$ over the input feature map $\mathbf{x}$; 2) summation of sampled values weighted by $\mathbf{w}$. The grid $\mathcal{R}$ defines the receptive field size and dilation. For example,

$$\mathcal{R} = \{(-1,-1), (-1,0), \ldots, (0,1), (1,1)\}$$

defines a 3×3 kernel with dilation 1.

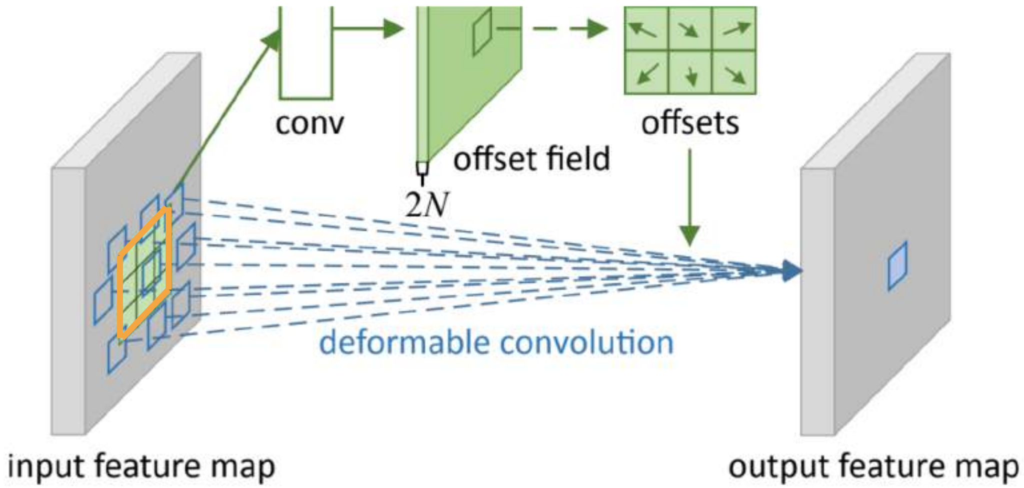
For each location $\mathbf{p}_0$ on the output feature map $\mathbf{y}$, we have

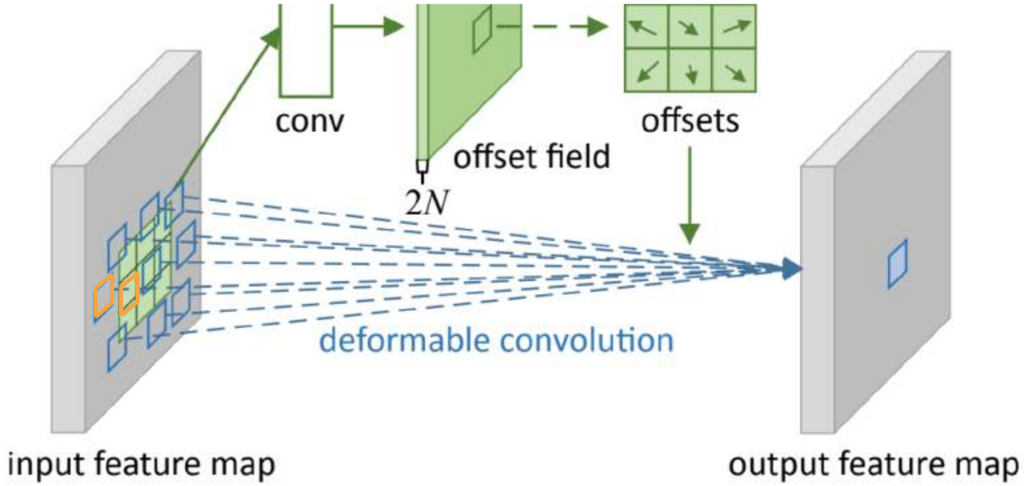
$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n), \tag{1}$$

where $\mathbf{p}_n$ enumerates the locations in $\mathcal{R}$.

In deformable convolution, the regular grid $\mathcal{R}$ is augmented with offsets $\{\Delta\mathbf{p}_n | n = 1, ..., N\}$, where $N = |\mathcal{R}|$. Eq. (1) becomes

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n). \tag{2}$$





3. Deformable Convolutional Networks

3.1 Deformable Convolution

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta \mathbf{p}_n). \qquad (2)$$

$$\mathbf{x}(\mathbf{p}) = \sum_{\mathbf{q}} G(\mathbf{q}, \mathbf{p}) \cdot \mathbf{x}(\mathbf{q}), \qquad (3)$$

where $\mathbf{p}$ denotes an arbitrary (fractional) location ($\mathbf{p} = \mathbf{p}_0 + \mathbf{p}_n + \Delta \mathbf{p}_n$ for Eq. (2)), $\mathbf{q}$ enumerates all integral spatial locations in the feature map $\mathbf{x}$, and $G(\cdot, \cdot)$ is the bilinear interpolation kernel. Note that G is two dimensional. It is separated into two one dimensional kernels as

$$G(\mathbf{q}, \mathbf{p}) = g(q_x, p_x) \cdot g(q_y, p_y), \qquad (4)$$

where $g(a, b) = max(0, 1 - |a - b|)$. Eq. (3) is fast to compute as $G(\mathbf{q}, \mathbf{p})$ is non-zero only for a few $\mathbf{q}$s.

P : 임의의 분수 값(ex - 0.5, 1.5 … 즉, 너무 터무니없는 값은 아님)
q : feature map의 모든 공간 열거

G : 쌍방향 보간 커널

3. Deformable Convolutional Networks

3.2 Deformable RoI Pooling

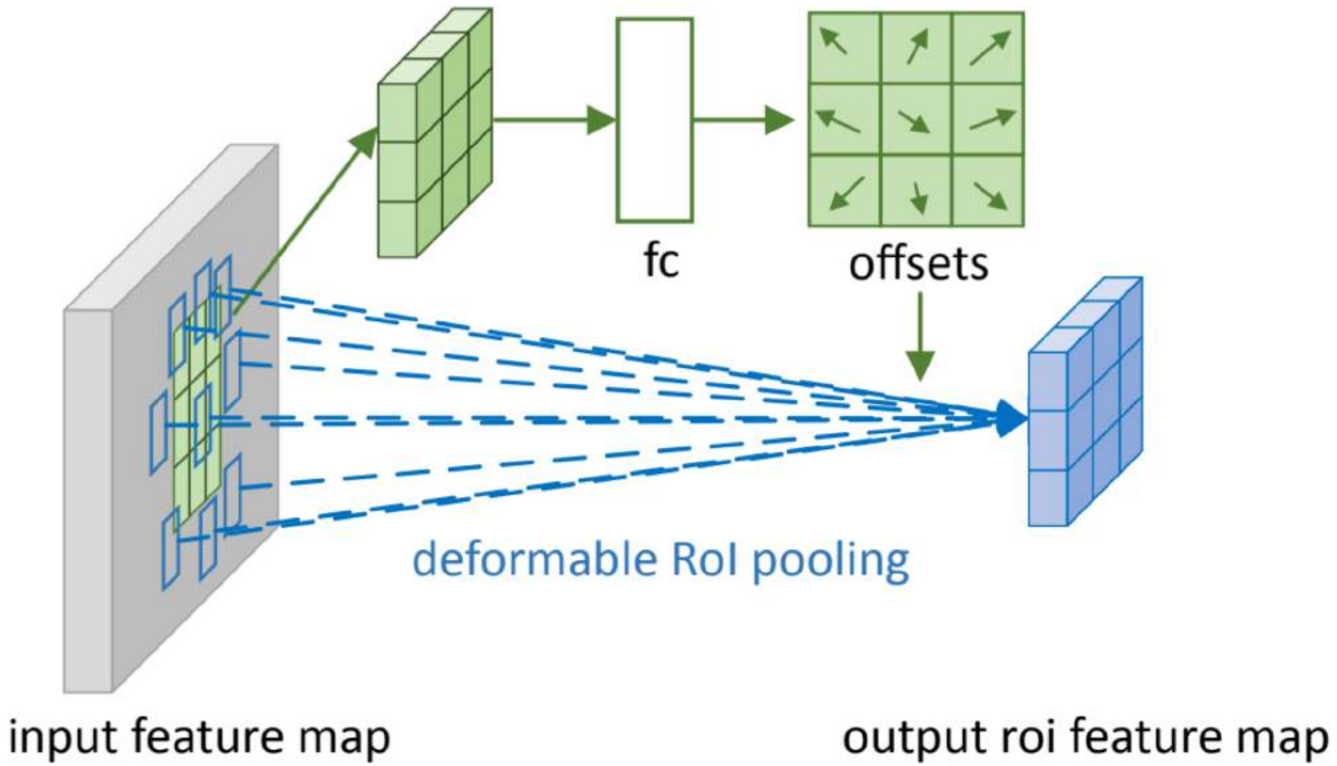


Figure 3: Illustration of 3 × 3 deformable RoI pooling.

3. Deformable Convolutional Networks

3.2 Deformable RoI Pooling

RoI Pooling [15] Given the input feature map $\mathbf{x}$ and a RoI of size $w \times h$ and top-left corner $\mathbf{p}_0$, RoI pooling divides the RoI into $k \times k$ (k is a free parameter) bins and outputs a $k \times k$ feature map $\mathbf{y}$. For (i, j)-th bin $(0 \le i, j < k)$, we have

$$\mathbf{y}(i,j) = \sum_{\mathbf{p} \in bin(i,j)} \mathbf{x}(\mathbf{p}_0 + \mathbf{p})/n_{ij}, \quad (5)$$

where n_{ij} is the number of pixels in the bin. The (i, j)-th bin spans $\lfloor i\frac{w}{k} \rfloor \le p_x < \lceil (i+1)\frac{w}{k} \rceil$ and $\lfloor j\frac{h}{k} \rfloor \le p_y < \lceil (j+1)\frac{h}{k} \rceil$.

Similarly as in Eq. (2), in deformable RoI pooling, offsets $\{\Delta \mathbf{p}_{ij} | 0 \le i, j < k\}$ are added to the spatial binning positions. Eq.(5) becomes

$$\mathbf{y}(i,j) = \sum_{\mathbf{p} \in bin(i,j)} \mathbf{x}(\mathbf{p}_0 + \mathbf{p} + \Delta \mathbf{p}_{ij})/n_{ij}. \quad (6)$$

Typically, $\Delta \mathbf{p}_{ij}$ is fractional. Eq. (6) is implemented by bilinear interpolation via Eq. (3) and (4).

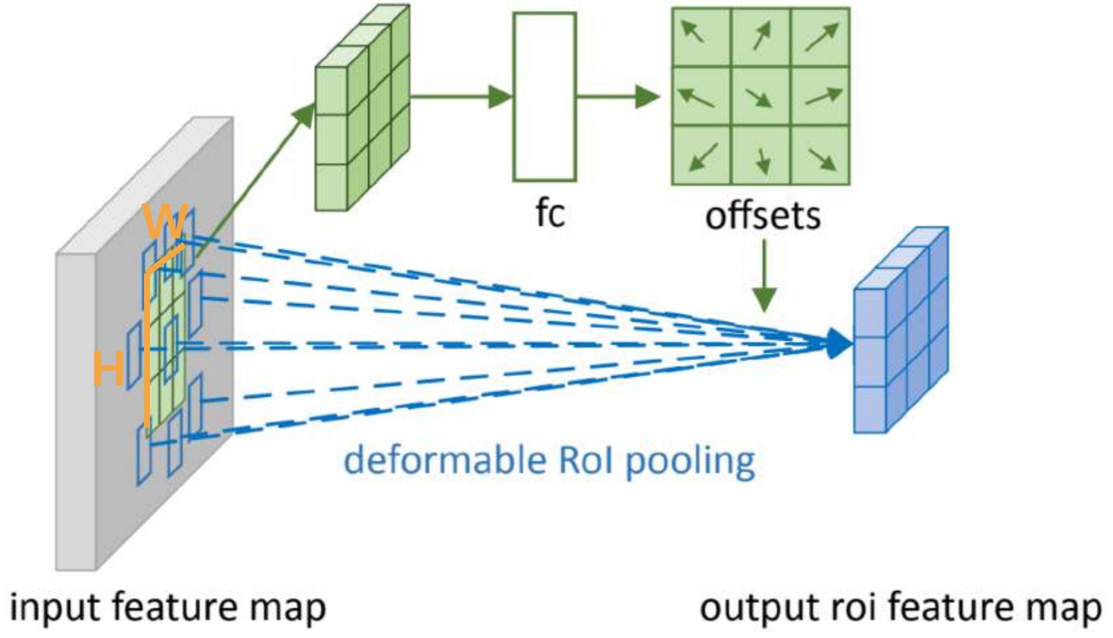


Figure 3: Illustration of 3×3 deformable RoI pooling.

k = 3,
ΔP_{ij} : 쌍방향 보간법

3. Deformable Convolutional Networks

3.2 Deformable RoI Pooling

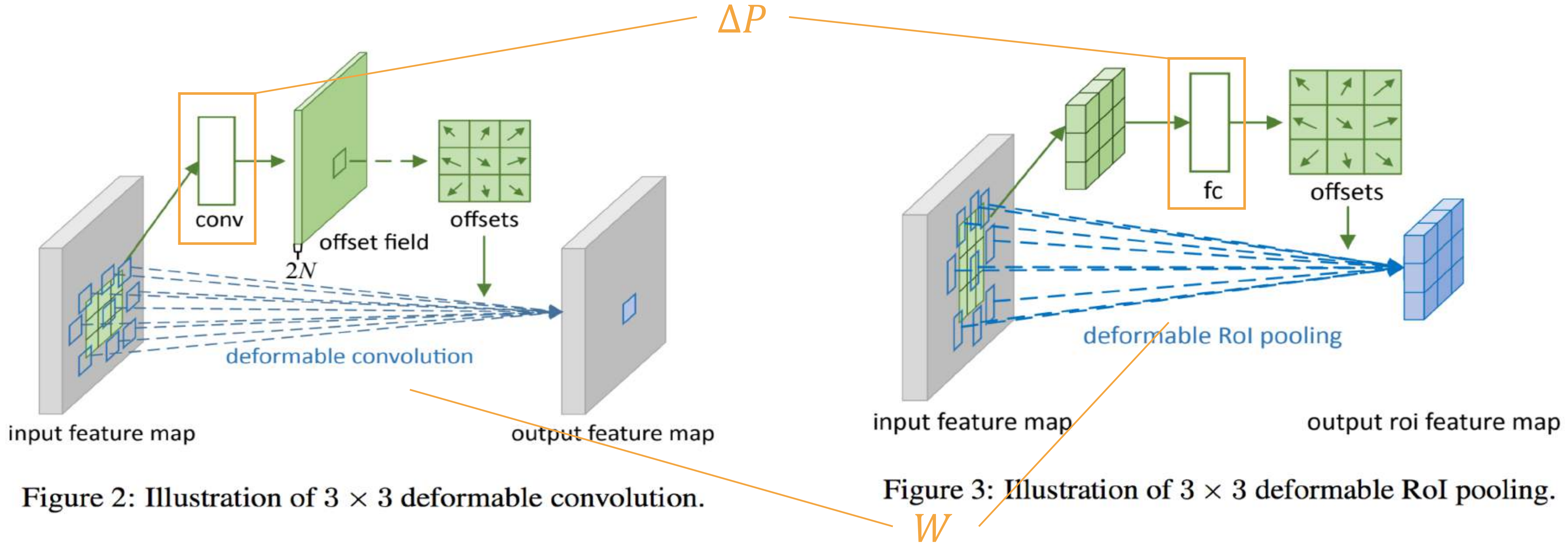


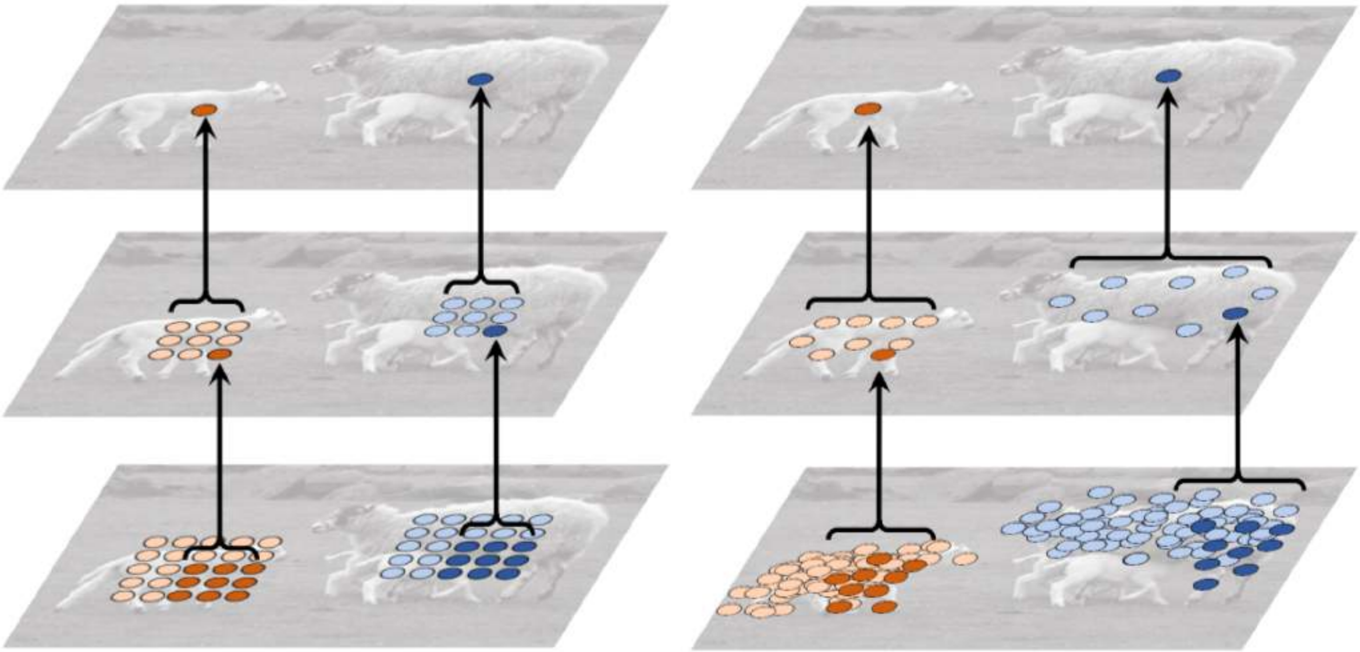
Figure 2: Illustration of 3 × 3 deformable convolution.

Figure 3: Illustration of 3 × 3 deformable RoI pooling.

둘다 backpropagation에 의해 한번에 학습

- 그냥 기존의 conv 레이어를 살짝만 바꾸어도 잘 작동
- Two stage, Three stage 필요 x → 간단

4. Understanding Deformable ConvNets



(a) standard convolution　(b) deformable convolution

Layer가 쌓이다 보면 엄청 큰 차이

4. Understanding Deformable ConvNets

background Small object large object

background Small object large object

4. Understanding Deformable ConvNets

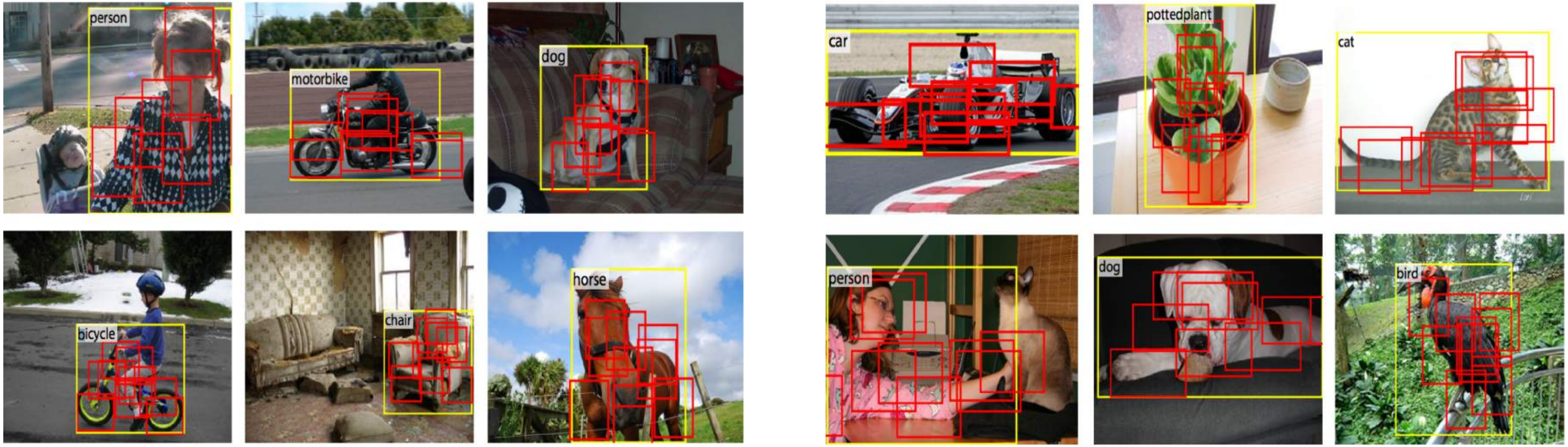


Figure 7: Illustration of offset parts in deformable (positive sensitive) RoI pooling in R-FCN [7] and 3 × 3 bins (red) for an input RoI (yellow). Note how the parts are offset to cover the non-rigid objects.

4. Understanding Deformable ConvNets

Test Accuracy	Regular CNN	Deformable CNN
Regular MNIST	98.74%	97.27%
Scaled MNIST	57.01%	92.55%

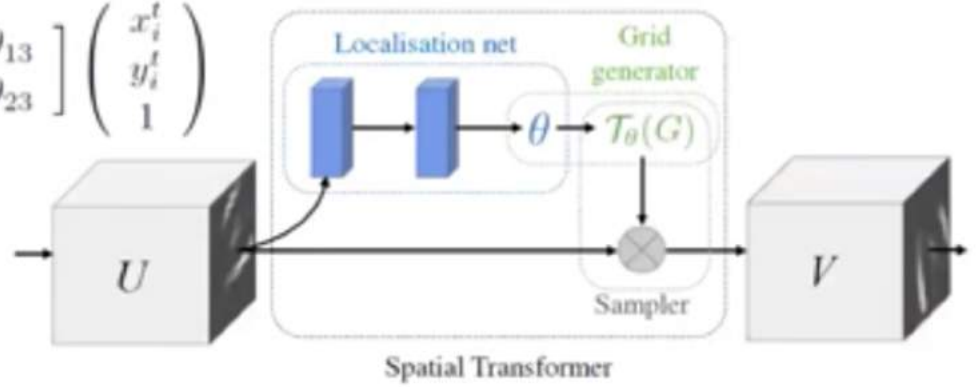
4. Understanding Deformable ConvNets

4.1 In Context of Related Works

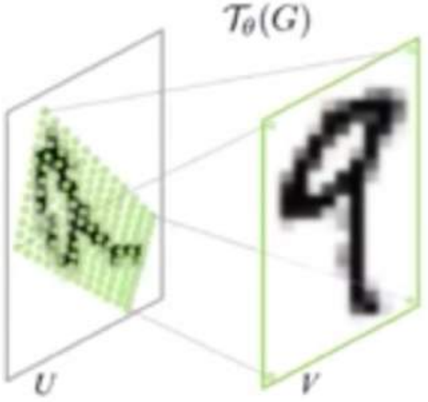
Spatial Transform Networks (STN)

- Linear transform으로 한정
- Small scale만 고성능

$$\begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$



- 찾은 bounding box를 다시 정면에 박고, 이후에 classification작업 필요



$$\frac{\partial V_i^c}{\partial U_{nm}^c} = \sum_n^H \sum_m^W \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

$$\frac{\partial V_i^c}{\partial x_i^s} = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |y_i^s - n|) \begin{cases} 0 & \text{if } |m - x_i^s| \geq 1 \\ 1 & \text{if } m \geq x_i^s \\ -1 & \text{if } m < x_i^s \end{cases}$$

W*X에서 그동안은 W에만 신경썼다면,
Deformable ConvNets는 X에도 신경썼다!

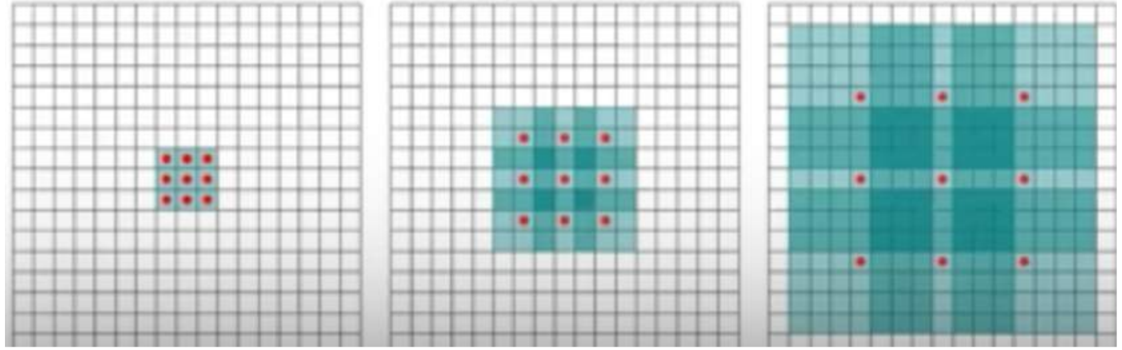
4. Understanding Deformable ConvNets

4.1 In Context of Related Works

Effective Receptive Field

이론상 3 by 3 → 9 by 9 → 27 by 27
But 사실 그렇게 Linear하게 커지지 X

Stride가 딱 3 by 3 이 아닐 수도 있어서 겹치는 부분이 존재하기 때문.



하지만 Deformable ConvNets는 receptive field가 효율적으로 커짐

5. Experiments

deformation modules	DeepLab mIoU@V / @C	class-aware RPN mAP@0.5 / @0.7	Faster R-CNN mAP@0.5 / @0.7	R-FCN mAP@0.5 / @0.7
atrous convolution (2,2,2) (default)	69.7 / 70.4	68.0 / 44.9	78.1 / 62.1	80.0 / 61.8
atrous convolution (4,4,4)	73.1 / 71.9	72.8 / 53.1	78.6 / 63.1	80.5 / 63.0
atrous convolution (6,6,6)	73.6 / 72.7	73.6 / 55.2	78.5 / 62.3	80.2 / 63.5
atrous convolution (8,8,8)	73.2 / 72.4	73.2 / 55.1	77.8 / 61.8	80.3 / 63.2
deformable convolution	**75.3 / 75.2**	**74.5 / 57.2**	78.6 / 63.3	81.4 / 64.7
deformable RoI pooling	N.A	N.A	78.3 / 66.6	81.2 / 65.0
deformable convolution & RoI pooling	N.A	N.A	**79.3 / 66.9**	**82.6 / 68.5**

Table 3: Evaluation of our deformable modules and atrous convolution, using ResNet-101.

5. Experiments

method	backbone architecture	M	B	mAP@[0.5:0.95]	mAP^r@0.5	mAP@[0.5:0.95] (small)	mAP@[0.5:0.95] (mid)	mAP@[0.5:0.95] (large)
class-aware RPN	ResNet-101			23.2	42.6	6.9	27.1	35.1
Ours				**25.8**	**45.9**	**7.2**	**28.3**	**40.7**
Faster RCNN	ResNet-101			29.4	48.0	9.0	30.5	47.1
Ours				**33.1**	**50.3**	**11.6**	**34.9**	**51.2**
R-FCN	ResNet-101			30.8	52.6	11.8	33.9	44.8
Ours				**34.5**	**55.0**	**14.0**	**37.7**	**50.3**
Faster RCNN	Aligned-Inception-ResNet			30.8	49.6	9.6	32.5	49.0
Ours				**34.1**	**51.1**	**12.2**	**36.5**	**52.4**
R-FCN	Aligned-Inception-ResNet			32.9	54.5	12.5	36.3	48.3
Ours				**36.1**	**56.7**	**14.8**	**39.8**	**52.2**
R-FCN	Aligned-Inception-ResNet	✓		34.5	55.0	16.8	37.3	48.3
Ours		✓		37.1	57.3	18.8	39.7	52.3
R-FCN		✓	✓	35.5	55.6	17.8	38.4	49.3
Ours		✓	✓	**37.5**	**58.0**	**19.4**	**40.1**	**52.5**

Table 5: Object detection results of deformable ConvNets v.s. plain ConvNets on COCO test-dev set. M denotes multi-scale testing, and B denotes iterative bounding box average in the table.