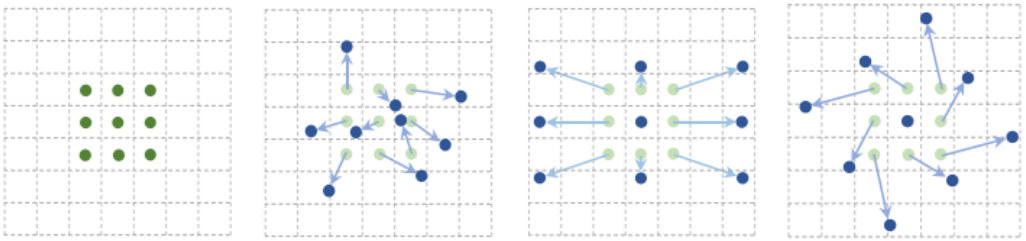
# 简介

CNN卷积本身其实并不能学习到旋转不变性，一般旋转不变性的学习靠的是数据增强这样的data driven方式去学习的。或者用SIFT等算法提取一些旋转不变的特征。
作者想提出一种新的卷积方式，可以自己无监督的学习到旋转不变特征。
思想很简单，卷积的模式保持不变，只不过卷积的采样点发生了变化。



(a) (b) (c) (d)

Figure 1: Illustration of the sampling locations in $3 \times 3$ standard and deformable convolutions. (a) regular sampling grid (green points) of standard convolution. (b) deformed sampling locations (dark blue points) with augmented offsets (light blue arrows) in deformable convolution. (c)(d) are special cases of (b), showing that the deformable convolution generalizes various transformations for scale, (anisotropic) aspect ratio and rotation.

作者提出了两个模块，deformable convolution和deformable RoI pooling。

# DCN

DC和dpooling的对象都是2d空间。

## DC

普通卷积的原理是先在输入feature map x上均匀采样(nxn矩形)，然后对应元素相乘权重W再相加。

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

如图对每个位置p0都有如上所示公式，而pn则代表输入feature map上当前nxn的采样位置R。

对应的可形变卷积版本，只是在采样位置上加入一个偏移量 $\Delta p_n$

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

由于偏移的位置范围是比较大的，所以这里用双线性插值计算出来：

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

其中 $p = p_0 + p_n + \Delta p_n$ 代表一个偏移位置，q是枚举了当前feature map上的所有位置。G代表双线性插值kernel。

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

其中 $g(a, b) = \max(0, 1 - |a - b|)$ 。公式(3)运算速度很快因为很多q的取值算出的G(q, p)结果都为0。
通过公式不难看出，q枚举了空间的所有位置，所以实际上每一个p都是p0位置和所有q位置线性插值的一个加权。距离p0越远，权值越小，对当前采样点的影响就越小。

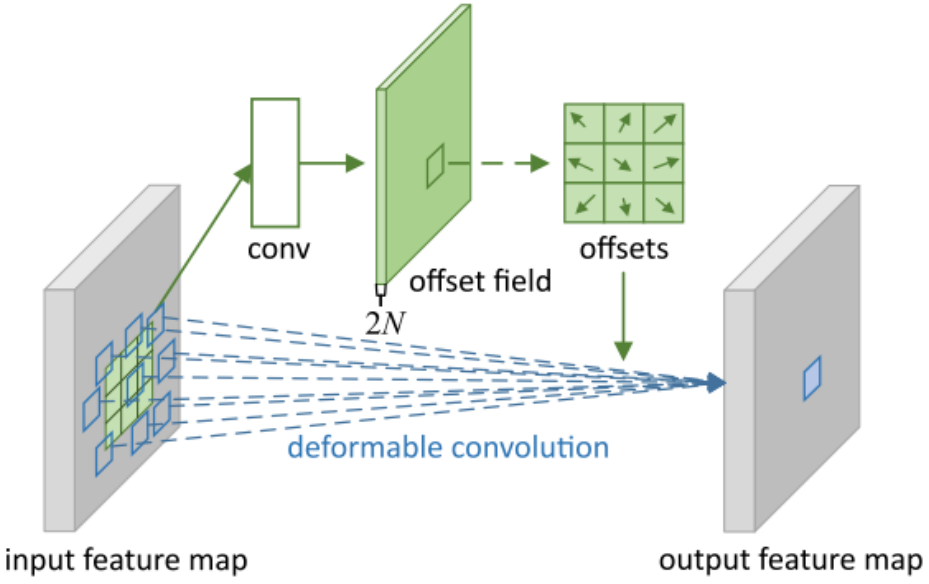



Figure 2: Illustration of $3 \times 3$ deformable convolution.

offset field的获取可以通过一个卷积核实现，这个卷积核的kernel和dilation和当前要使用的卷积一样，其输出的offset fields的spatial resolution和输入feature map一样。 channel数是2N，这个2N代表N个2D offsets。训练的时候这个offset的conv kernel和正常的kernel一起训练。其中N代表的是conv kernel的元素个数（3x3kernel N=9）

## Deformable RoI Pooling

给定输入的feature map x和尺寸为wxh的RoI，以及左上角的位置p0，RoI pooling将RoI分为kxk的bins然后输出kxk的feature map y。对于位置是(i,j)=th的bin(0<=i, j<=k)

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

其中 $n_{ij}$ 是每个bin当中pixels的数目。
deformable RoI pooling也是在bin postion加入了offsets $\Delta\{p_{i,j} \mid 0 \le i,j < k\}$

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

首先通过公式5的RoIpooling产生poolded feature maps。然后利用一个fc层来产生normalized offsets $\Delta\hat{p}_{i,j}$，最后 $\Delta p_{ij} = \gamma\Delta\hat{p}_{i,j}(w,h)$
其中gamma是一个调节系数，文章中取0.1。其中offset normailize非常必要，使得offset对于RoI尺寸是invariant。

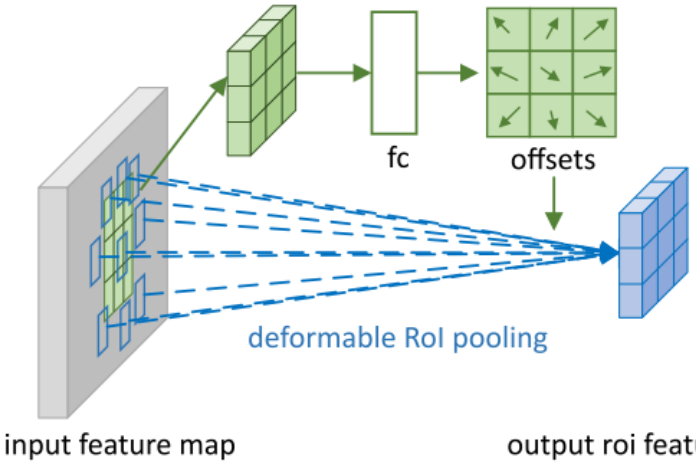



Figure 3: Illustration of $3 \times 3$ deformable RoI pooling.

## Position-Sensitive (PS)

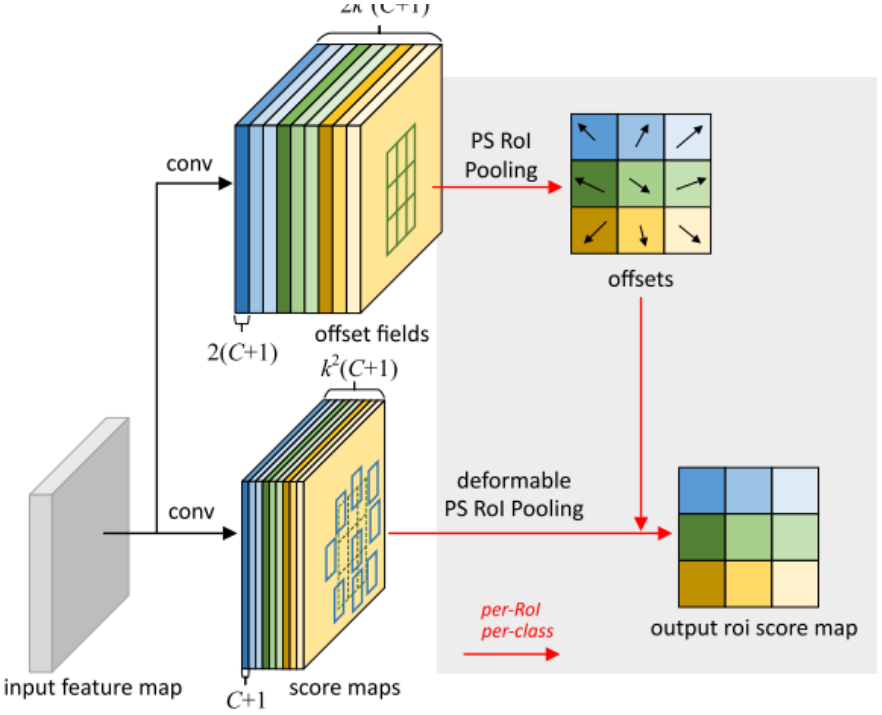



Figure 4: Illustration of $3 \times 3$ deformable PS RoI pooling.

# Deformable ConvNets

作者把DCN的结构加入到不同的任务当中，主要是用resnet-101和Aligned-Inception-Resnet backbone然后把最后的conv5层的stride=2变为stride=1然后dilation=2，这样最后的stride=16。

## 分割与检测

- Faster R-CNN，vanilla版本用resnet-101 backbone，其中RoI pooling层插入到conv4和conv5之间，然后后面接10层卷积。 但是作者最后用了FPN版本，每一个pooled RoI features上都接两个fc层1024 dim然后是检测和分类的回归。

# 实验

主要用了VOC dataset