Deformable Convolution and Active Convolution

王皓
wanghao@stanford.edu

August 8, 2017

1 相关工作

Deformable Convolution[1] 和 Active Convolution[2] 是两篇在今年三月份同时出的论文，均和更灵活的卷积有关。本文在 semantic segmentation 的问题下对这两类 convolution 进行讨论。

以下假设 input feature map 的 shape 为 (C_1, W_1, H_1)，output feature map 的 shape 为 (C_2, W_2, H_2)，kernel size 为 (K, K)。

1.1 Deformable Convolution

Deformable Convolution（其网络简称 DCN）的核心框架可见原论文中的该图（图1）。其关键内容在于从 input feature map 引出旁路，用一个新的 convolution 计算出 shape 为 $(W_1, H_1, K, K, 2)$ 的 offset，其中每个位置上 $(K, K, 2)$ 的值表示原 convolution 在该位置上所使用的 offset。offset 之后被输入到原 convolution 中。因此：(1) 该层原 convolution 的所有 kernel 使用的是一样的 offset，不同位置 offset 不同；(2)offset 的值与输入的图片有关；(3)offset shape 为 $(W_1, H_1, K, K, 2)$。

1.2 Active Convolution

Active Convolution（其网络简称 ACN）将 offset 作为 convolution 中的一个参数，地位与 weight、bias 相同。每个 convolution 的所有 kernel 共享一个 shape 为 $(K, K, 2)$ 的 offset。因此：(1) 该 convolution 的所有 kernel、所有位置使用的是一样的 offset；(2)offset 的值与输入的图片无关；(3)offset shape 为 $(K, K, 2)$。

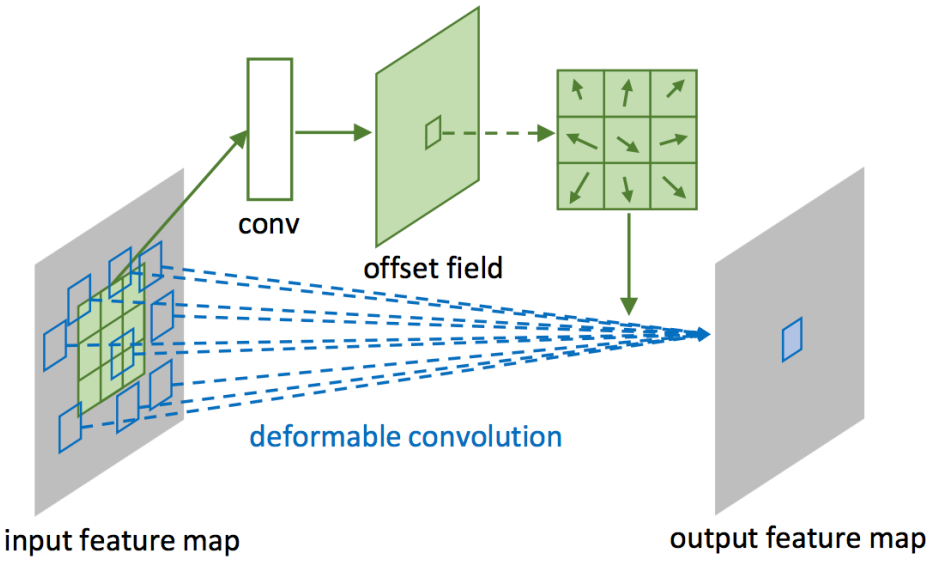

Figure 2: Illustration of 3×3 deformable convolution.

Figure 1: Deformable Convolution 的框架

由于 Active Convolution 原论文是在 Caffe 上实现的，而且没有 Cityscapes 数据集上做实验，实习过程中，我基于 Deformable Convolution 的代码在 MXNet 上实现了 Active Convolution，并在 Cityscapes 上做了一些实验与分析。

2 代码实现

2.1 核心文件

以下以 Deformable-ConvNets 项目目录为当前目录。

- ./deeplab/symbols/resnet_v1_101_deeplab_*.py：网络结构描述文件。添加新网络后注意要修改 class 名、文件名，并在同目录的 ___init___.py 文件中 import 相应 module。
- ./deeplab/symbols/feature_map.ipynb：用来可视化 DCN 的 offset。
- ./deeplab/symbols/offset.ipynb：用来输出 ACN 的 offset。
- ./deeplab/symbols/training_loss.ipynb：用来可视化 training log 文件中的 loss 变化。

- ./deeplab/symbols/weight.ipynb：用来输出 model 中的 weight。
- ./deeplab/symbols/{train.py, test.py}：用来 train、test deeplab 网络的代码。
- ./{train.sh, test.sh}：用来 train、test deeplab 网络的脚本。注意调用 test 时需要加入–ignore_cache 参数，否则第二遍测试结果不会发生变化。
- ./experiments/deeplab/cfgs/deeplab_resnet_v1_101_cityscapes_segmentation_*.yaml：用于描述 train、test 过程的 hyperparameters。主要要改 MXNET_VERSION、symbol、gpus、dataset_path、root_path、model_prefix 这几项。
- ./external/mxnet/mxnet-*:实验中尝试的各个版本的 MXNet,与 MXNET_VERSION 对应。
- ./external/mxnet/mxnet-aconv/src/operator/contrib/active_convolution.cc, active_convolution.cu, active_convolution-inl.h：用于实现 Active Convolution 的 CPU 代码、GPU 代码以及共用的部分。
- ./external/mxnet/mxnet-aconv/src/operator/contrib/nn/{active_im2col.cuh, active_im2col.h}：用来实现 im2col、col2im 的 GPU、CPU 代码。
- ./output/cityscape/deeplab_resnet_v1_101_cityscapes_segmentation_*:train 和 test (validation) 的结果。注意 leftImg8bit_val/results/中包含了 validation 数据集的每张图片的 segmentation 结果。

2.2 算法主要函数

以下提到的文件与函数在./external/mxnet/mxnet-*/src/operator/contrib 内，基于 Deformable Convolution 的代码修改。

- active_convolution-inl.h - ActiveConvolutionOp.Forward()：前向传播算法的核心函数。
- active_convolution-inl.h - ActiveConvolutionOp.Backward()：反向传播算法的核心函数，其中加入了用 CPU 计算 Normalized Gradient 的代码。关于一般的卷积的反向传播求导可见https://www.slideshare.net/kuwajima/cnnbp中第 10 页。在计算对 offset 的导数 in_grad[conv::kOffset] 时，Deformable Convolution 代码中使用了这样的算法：$\frac{dJ}{dOffset} = \frac{dx}{dOffset} * \frac{dJ}{dx}$，注意这未在 DCN 论文中提及；而在 Active Convolution 的论文中，则是提到了这样的算法：$\frac{dJ}{dOffset} = \frac{dy}{dOffset} * \frac{dJ}{dy}$；其中 x 与 y 是卷积前后

两层的 feature map。在 MXNet 实现 Active Convolution 时，我沿用了 Deformable Convolution 的算法。

- active_convolution-inl.h - ActiveConvolutionProp.InferShape()：由于引入了新的卷积参数 offset，需要在 InferShape() 中声明参数大小，并用 SHAPE_ASSIGN_CHECK() 分配空间。
- nn/active_im2col.h - active_im2col_gpu_kernel()：主要需要改变 data_offset_ptr、data_offset_h_ptr、data_offset_w_ptr 的维度。
- nn/active_im2col.h - active_col2im_gpu_kernel()：与 im2col 的改变相同。
- nn/active_im2col.h - active_col2im_coord_gpu_kernel()：除了与 im2col 相同的改变外，需要将最后 grad_offset[index] 改为 grad_offset[c]，因为 offset 的 shape 维度减少了。

3 实验与结果

3.1 Deformable Convolution

3.1.1 尝试给 Deformable Convolution 加 dilation

首先在 Cityscapes 数据集上复现了论文的结果。统计了三个 Deformable Convolution 层用到的 offset，发现相对于初始 offset，新的 offset 从总体上看基本是学到了一个按射线方向扩张的 dilated convolution。

根据检查，DCN 学到的 offset 值实际上超出了用于计算 offset 的卷积的感受野，这不是很合理。于是尝试给 offset convolution 添加了 (2, 2) 和 (3, 3) 的 dilation。结果上看，(1, 1)、(2, 2)、(3, 3) dilation 的 training loss 分别为 0.086104、0.086699、0.087274，validation mIOU 分别为 75.260、74.715、74.531（第一个即为复现 DCN 论文的结果）。加更大的 dilation 使 performance 稍有下降。

图2是 validation 数据集上一个 Deformable Convolution Layer（res5a_branch2b）的 3×3×x, yoffset output 值的直方图统计，红、绿、蓝颜色分别代表 (1, 1)、(2, 2)、(3, 3)offset convolution dilation 的结果。图表使用./deeplab/symbols/feature_map.ipynb 生成。可以看到，加上更大的 dilation 总体上使得 offset 更加扩张，但这并没有带来更好的 performance。

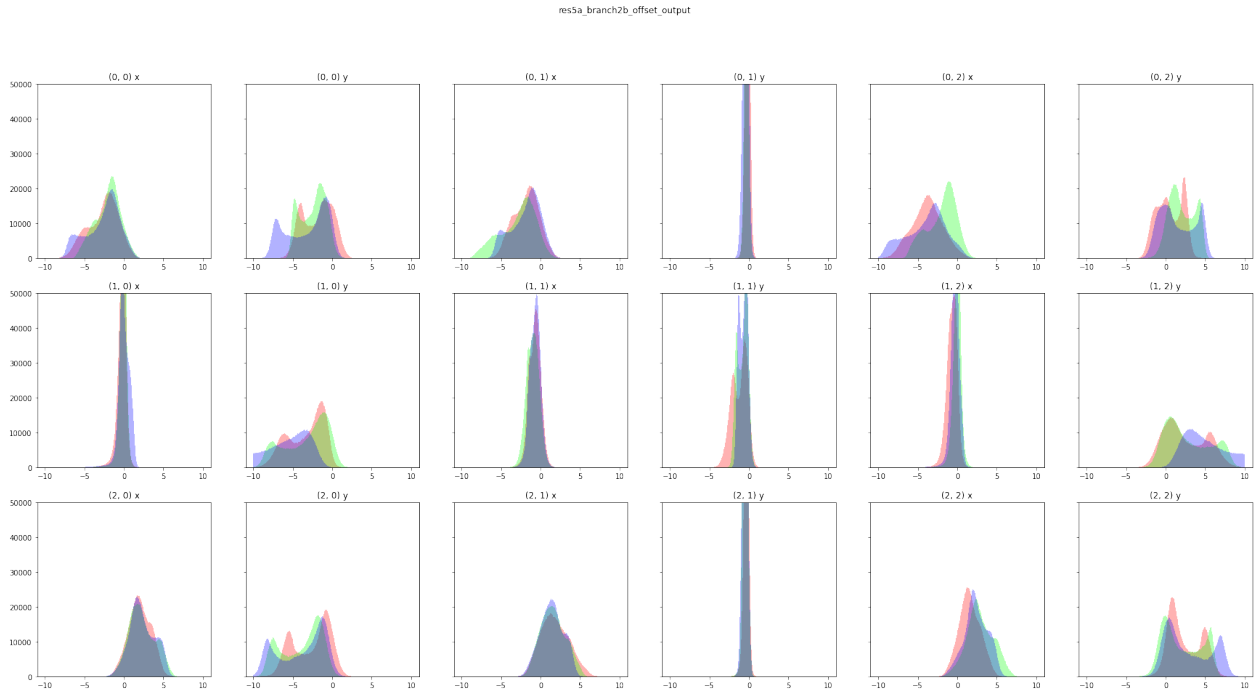

Figure 2: res5a_branch2b_offset_output

3.1.2 尝试给 baseline model 加 dilation

考虑到“新的 offset 从总体上看基本是学到了一个按射线方向扩张的 dilated convolution”，人为地给 baseline model（无 Deformable Convolution 的 DeepLab model）的 res5a, res5b, res5c 层加上 dilation。从结果上看，在 (8, 8) dilation 的时候能达到最大 3 个点的 mIOU 提升，但这其实还是低于 DCN 的结果（表3.1.2）。说明（1）DCN 的确是总体上学到了按射线方向扩展的 dilated convolution；（2）DCN 也的确做到了更灵活的卷积。

dilate	training loss	mIOU
(2, 2)		70.3
(3, 3)	0.085538	71.080
(4, 4)	0.085941	71.663
(5, 5)	0.087299	72.519
(6, 6)	0.087371	72.882
(7, 7)	0.088151	72.698
(8, 8)	0.088288	**73.018**
(9, 9)	0.089901	72.441
(10, 10)	0.090188	72.706

Table 1: Baseline Models with Dilation

3.1.3 增大 epoch

考虑到网络训练可能不完全，将 DCN 的 training epoch 从 53 提升到 100，lr_step 从 40 提升到 60，结果上 training loss 从 0.084309 下降到 0.075108，但 validation mIOU 提升不大，从 0.74704 变为 0.75364。

3.2 Active Convolution

3.2.1 朴素方法在 Cityscapes 上的结果

通过前述 “代码实现” 一节中的方法，将 Active Convolution 实现到了 MXNet 上。其 mIOU 为 0.73822（相比之下 deeplab 0.703、DCN 0.74704）。使用./deeplab/symbols/offset.ipynb 输出其 offset，得到三层 Active Convolution 的 offset 见图3。由于 feature map shape 为 $(512, 64, 128)$，offset 可能会使得卷积位置越界。注意到代码中的二线性插值时如果遇到越界的坐标，会直接取到最近的边界位置，因此仍可以计算。

```
res5a_branch2b_offset
( -6.52, -10.55) (-46.07, -58.80) ( -4.55,   0.19)
(  0.25, -16.44) ( -0.67,  -3.32) ( -0.29,   6.20)
(  0.77,   3.64) ( 10.67, -99.54) (  1.30,   2.45)
res5b_branch2b_offset
(-58.96,  47.11) (-10.51,   0.45) ( -5.48,  -2.57)
(  6.65,  95.81) ( -2.82,   0.09) ( -0.47,  22.07)
( 89.68,  75.42) ( -4.44,   4.38) (  5.05,  -2.32)
res5c_branch2b_offset
(-18.05,  61.15) ( -2.06,  -0.44) (  2.27, -25.54)
(-12.51,  96.35) ( -1.36,   1.53) ( -0.07,   2.34)
(  0.08,  61.96) ( -1.69,  -6.19) (  2.83,  -2.37)
```

Figure 3: ACN, mIOU=0.73822

3.2.2 Normalized Gradient

为了实现论文 [2] 中 2.5. 小节提到的 Normalized Gradient，我将 GPU 中的 offset 数据拷贝到 CPU 中，然后使用 MShadow 库进行计算，具体实现可见 active_convolution-inl.h - ActiveConvolutionOp.Backward()。在将每个位置的 (x, y) offset gradient normalize 之后，算法对其乘上了一个 offset_lr_rate。实验中尝试了 1、0.1、0.05、0.01 的 offset_lr_rate 值，实验的 mIOU 和 offset 结果如下。总体上看，在 offset_lr_rate=0.05 时，达到了最高的 mIOU=73.437。

```
res5a_branch2b_offset
(-482.31, -357.38) (424.58, 492.84) (513.45, 390.41)
(321.18, 542.54) ( 24.90, -344.23) (-302.43, 311.18)
(-519.30, 211.24) (261.75, -1019.36) (264.93, 233.63)
res5b_branch2b_offset
(-139.29, -562.21) (601.22, -34.21) (-106.18, 392.65)
(487.74, -315.17) (436.97, -118.10) (-418.79, -326.50)
(173.76, 221.49) (-101.55, 241.43) (-421.92, -164.05)
res5c_branch2b_offset
(470.77, -248.50) (1038.58, 307.98) (497.51, 228.77)
(-52.22, 309.20) (487.94, 555.05) (162.13, -640.07)
(-481.67, 369.18) (322.84, -244.70) (391.38, -252.51)
```

Figure 4: ACN with Normalized Gradient, offset_lr_rate=1, mIOU=0.68538

```
res5a_branch2b_offset
(-28.63, -33.44) (-16.82, -81.32) ( 52.40,  11.42)
( -9.75, 115.89) (  8.38,   0.69) ( 25.38,  33.64)
( 10.44, -24.26) ( 27.97, -117.39) (-33.45,  64.34)
res5b_branch2b_offset
(-21.44,  29.09) (  0.62,  -3.92) (  1.50,   2.64)
(-77.89,  44.65) (-12.34,  -3.81) (-60.21,  99.54)
( 44.80,  90.52) ( -7.32, -16.96) ( 53.55,  37.03)
res5c_branch2b_offset
( -2.70,  88.09) ( -3.93,   5.67) (-42.93,   2.84)
( -5.79, 119.19) (101.70,   3.05) ( 16.03,  79.86)
( -1.05,  26.93) (  1.82,  -0.22) ( -1.25,  61.86)
```

Figure 5: ACN with Normalized Gradient, offset_lr_rate=0.1, mIOU=0.72148

```
res5a_branch2b_offset
( -1.42, -14.15) (-11.37,  -1.21) (-12.00,  25.49)
(  1.22,  49.97) ( -0.48,  -4.34) ( 10.78,  40.24)
(  9.41,   2.59) ( -7.33,  -0.96) ( 21.86,   4.91)
res5b_branch2b_offset
(-13.64,  22.28) ( -6.51,   0.37) (  1.66,  19.58)
( -0.35, -45.20) ( -3.64,  -0.25) (-11.53,  18.79)
(  4.76,  25.43) (  2.64,  -0.17) ( -9.13,  33.12)
res5c_branch2b_offset
(-52.30,  25.80) (-37.38, -25.41) (-20.94,  -2.81)
(-10.77, -51.18) (-24.05,  19.52) (  1.74,  -6.39)
(  4.78,  54.22) (  6.19, -10.26) ( 11.15,   2.55)
```

Figure 6: ACN with Normalized Gradient, offset_lr_rate=0.05, mIOU=0.73437

```
res5a_branch2b_offset
( -2.38,   1.15) ( -0.77,   0.19) (  0.72,   3.53)
( -0.57,  -1.51) ( -0.53,  -1.26) ( -0.50,   1.43)
(  1.36,  -1.24) (  0.63,  -0.53) ( -0.30,   0.50)
res5b_branch2b_offset
( -1.48,  -0.62) ( -3.70,  -0.04) (  1.72,  -0.45)
( -0.47,  -0.52) ( -2.09,  -0.23) ( -2.50,  -0.69)
(  3.39,  -3.44) (  3.56,  -0.23) (  0.54,  -1.54)
res5c_branch2b_offset
( -3.51,  -8.54) ( -5.63,  -0.47) ( -1.54,   4.14)
( -0.25,  -9.45) (  0.64,  -0.28) (  0.28,   9.82)
( -1.40,  -4.54) (  5.38,   0.08) (  2.20,   6.45)
```

Figure 7: ACN with Normalized Gradient, offset_lr_rate=0.01, mIOU=0.72973

3.2.3 Warming Up

在 offset_lr_rate=0.05 的条件下，尝试在 53 个 epoch 中的前 20 个做 warming up，即这期间将 offset 均置为 0，得到 mIOU 为 0.74216。此为目前为止在 MXNet 上复现 ACN 得到的最高 mIOU，其 offset 值也相对合理（图8）。

```
res5a_branch2b_offset
(-12.84,  12.10) (-19.64,   3.21) ( -4.60,  -2.59)
( -2.80,   1.31) (  0.22,  -1.74) ( -1.22,   1.19)
(  1.30,   1.69) (  5.49,  -0.17) ( -0.21,   0.48)
res5b_branch2b_offset
( -4.37,   2.62) (-22.51,  -0.55) (-24.47,   4.94)
( -0.52,  -3.76) (  2.32,   1.41) (  6.44,   7.05)
( 23.08,   1.19) ( 30.52,  -0.41) ( 13.36,   9.80)
res5c_branch2b_offset
( -2.35, -12.85) (-28.92,  -0.41) (  4.35,  15.97)
( -2.54, -24.21) (  0.04,   5.67) ( -0.34,   6.87)
( -1.40, -16.70) ( 23.28,  -4.65) (  0.95,  -5.17)
```

Figure 8: ACN with Normalized Gradient and Warming Up, offset_lr_rate=0.05, mIOU=0.74216

4 后续工作

DCN 与 ACN 实现了两种灵活的卷积，其最主要的区别在于，DCN 中的 offset 是从输入图片中计算得到的；ACN 中的 offset 是卷积本身的参数，与输入无关，类似于 weight。他们的一个相同点在于，都对所有 channel 的 input feature map（channel 数量为 C_1），以及卷积的所有 kernel（kernel 数量为 C_2）使用了相同的 offset 值。这是一个可能可以进行拓展的地方：可能可以将 DCN 或 ACN 的不同 channel 或不同 kernel 使用不同的 offset 值。

一个有潜力的拓展方式是，将 ACN 的 offset shape 拓展到 $(C_2, K, K, 2)$，即卷积中每个 kernel 使用不同的 offset 值。由于现在实现 Forward 和 Backward 时都会使用 im2col 和 col2im 的变换来将卷积操作转换为矩阵乘法操作，实现拓展的 ACN 会遇到一些困难。例如，在 Forward 中需要将 input feature map 转换为 shape 为 $(C_1 * K, H_2 * W_2)$ 的 column，其每一列表示一次卷积操作，每一行表示一个位置上的所有卷积操作。由于现在每个卷积对应的 offset 不同，那么个 kernel 卷积操作需要的 column 也不同，总共需要计算 C_1 次 im2col，然后将结果连接起来；Backward 方面也需要有比较大的改动。这将是后续工作主要需要考虑的内容。

参考论文

[1] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable Convolutional Networks. *arXiv.org*, March 2017.

[2] Yunho Jeon and Junmo Kim. Active Convolution: Learning the Shape of Convolution for Image Classification. *arXiv.org*, March 2017.