Deformable Convolutional Networks

Jifeng Dai^  
With Haozhi Qi*, Yuwen Xiong*, Yi Li*, Guodong Zhang*, Han Hu, Yichen Wei  
Visual Computing Group  
Microsoft Research Asia  
(* interns at MSRA, ^ equal contribution)
Highlights

• Enabling effective modeling of spatial transformation in ConvNets

• No additional supervision for learning spatial transformation

• Significant accuracy improvements on sophisticated vision tasks

Code is available at https://github.com/msracver/Deformable-ConvNets
Modeling Spatial Transformations

• A long standing problem in computer vision

  Deformation:  

  Scale:  

  Viewpoint variation:  

  Intra-class variation:  

(Some examples are taken from Li Fei-fei’s course CS223B, 2009-2010)
Traditional Approaches

• 1) To build training datasets with sufficient desired variations

• 2) To use transformation-invariant features and algorithms

• Drawbacks: geometric transformations are assumed fixed and known, hand-crafted design of invariant features and algorithms
Spatial transformations in CNNs

- Regular CNNs are inherently limited to model large unknown transformations
  - The limitation originates from the fixed geometric structures of CNN modules
Spatial Transformer Networks

- Learning a global, parametric transformation on feature maps
  - Prefixed transformation family, infeasible for complex vision tasks
Deformable Convolution

• Local, dense, non-parametric transformation
  • Learning to deform the sampling locations in the convolution/RoI Pooling modules

![Diagram showing regular, deformed, scale & aspect ratio, and rotation transformations]
Deformable Convolution

Regular convolution

\[ y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n) \]

Deformable convolution

\[ y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n + \Delta p_n) \]

where \( \Delta p_n \) is generated by a sibling branch of regular convolution
Deformable RoI Pooling

Regular RoI pooling
\[ y(i, j) = \sum_{p \in \text{bin}(i, j)} x(p_0 + p)/n_{ij} \]

Deformable RoI pooling
\[ y(i, j) = \sum_{p \in \text{bin}(i, j)} x(p_0 + p + \Delta p_{ij})/n_{ij} \]

where \( \Delta p_{ij} \) is generated by a sibling fc branch
Deformable ConvNets

• Same input & output as the plain versions
  • Regular convolution -> deformable convolution
  • Regular RoI pooling -> deformable RoI pooling

• End-to-end trainable without additional supervision
Sampling Locations of Deformable Convolution

(a) standard convolution  
(b) deformable convolution
Part Offsets in Deformable RoI Pooling
### Ablation Experiments on VOC & Cityscapes

- Number of deformable convolutional layers (using ResNet-101)

<table>
<thead>
<tr>
<th># deformable layers</th>
<th>DeepLab</th>
<th>Class-aware RPN</th>
<th>Faster R-CNN (2fc)</th>
<th>R-FCN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIoU@V (%)</td>
<td>mIoU @C (%)</td>
<td>mAP@0.5 (%)</td>
<td>mAP@0.7 (%)</td>
</tr>
<tr>
<td>None (0, baseline)</td>
<td>69.7</td>
<td>70.4</td>
<td>68.0</td>
<td>44.9</td>
</tr>
<tr>
<td>Res5c (1)</td>
<td>73.9</td>
<td>73.5</td>
<td>73.5</td>
<td>54.4</td>
</tr>
<tr>
<td>Res5b, c (2)</td>
<td>74.8</td>
<td>74.4</td>
<td>74.3</td>
<td>56.3</td>
</tr>
<tr>
<td>Res5a, b, c (default)</td>
<td>75.2</td>
<td>75.2</td>
<td>74.5</td>
<td>57.2</td>
</tr>
<tr>
<td>Res5 &amp; res4b22, b21, b20 (6)</td>
<td>74.8</td>
<td>75.1</td>
<td>74.6</td>
<td>57.7</td>
</tr>
</tbody>
</table>
# Deformable ConvNets v.s. dilated convolution

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

**Diagram:**
- Regular convolution
- Dilated convolution
- Deformable convolution
Model Complexity and Runtime on VOC & Cityscapes

- Deformable ConvNets v.s. regular ConvNets

<table>
<thead>
<tr>
<th>Method</th>
<th># params</th>
<th>Net forward (sec)</th>
<th>Runtime (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular DeepLab @Cityscapes</td>
<td>46.0M</td>
<td>0.610</td>
<td>0.650</td>
</tr>
<tr>
<td>Deformable DeepLab @Cityscapes</td>
<td>46.1 M</td>
<td>0.656</td>
<td>0.696</td>
</tr>
<tr>
<td>Regular DeepLab @VOC</td>
<td>46.0M</td>
<td>0.084</td>
<td>0.094</td>
</tr>
<tr>
<td>Deformable DeepLab @VOC</td>
<td>46.1 M</td>
<td>0.088</td>
<td>0.098</td>
</tr>
<tr>
<td>Regular Class-aware RPN</td>
<td>46.0 M</td>
<td>0.142</td>
<td>0.323</td>
</tr>
<tr>
<td>Deformable class-aware RPN</td>
<td>46.1 M</td>
<td>0.152</td>
<td>0.334</td>
</tr>
<tr>
<td>Regular Faster R-CNN (2fc)</td>
<td>58.3 M</td>
<td>0.147</td>
<td>0.190</td>
</tr>
<tr>
<td>Deformable Faster R-CNN (2fc)</td>
<td>59.9 M</td>
<td>0.192</td>
<td>0.234</td>
</tr>
<tr>
<td>Regular R-FCN</td>
<td>47.1 M</td>
<td>0.143</td>
<td>0.170</td>
</tr>
<tr>
<td>Deformable R-FCN</td>
<td>49.5 M</td>
<td>0.169</td>
<td>0.193</td>
</tr>
</tbody>
</table>
Object Detection on COCO

• Deformable ConvNets v.s. regular ConvNets
Conclusion

• Deformable ConvNets for dense spatial modeling
  • Simple, efficient, deep, and end-to-end
  • No additional supervision
  • Feasible and effective on sophisticated vision tasks for the first time