Decoupled Networks

Weiyang Liu*, Zhen Liu*, Zhiding Yu, Bo Dai, Rongmei Lin, Yisen Wang, James M. Rehg, Le Song

(* indicates equal contributions)
Motivation I

• Convolution works well by incorporating our prior knowledge on images (compared to MLP).
• Specifically, it computes inner product-based similarity in a sliding window fashion.

Questions:
• Why should inner product be the optimal similarity measure (despite the simplicity)?
• Is there better similarity prior for specific tasks?
Motivation II

• The original inner product based convolution operator learns naturally *decoupled* features.

• **norm** of feature = **intra-class variation**
• **angle** of feature = **semantic difference**

• We can explicitly model both intra-class variation and semantic difference with *decoupled* convolution framework.
Decoupled Convolution

- Original Inner Product-based Convolution
  \[ <w, x> = ||w|| \cdot ||x|| \cdot \cos(\theta_{w,x}) \]
  naturally decoupled

- Decoupled Convolution
  \[ f(w, x) = h(||w||, ||x||) \cdot g(\theta_{w,x}) \]

  - Magnitude: intra-class variation
  - Angle: semantic difference
Case Study: Hyperspherical Convolution

- Hyperspherical Convolution (SphereConv) models the norm by $h(\|w\|, \|x\|) = \alpha$.
- SphereConv ignores intra-class variation by discarding all magnitude information.

Intuition: The identity information is preserved by phase in 2D Fourier transform.

2D feature visualization on MNIST shows that SphereConv can compress the intra-class variation.
Example Designs

Magnitude (norm) Functions

- **SphereConv** - Project onto unit sphere
  \[ h(\|w\|, \|x\|) = \alpha \]

- **BallConv** - Project onto unit ball
  \[ h(\|w\|, \|x\|) = \alpha \min(\|x\|, \rho)/\rho \]

- **TanhConv** - Soft BallConv
  \[ h(\|w\|, \|x\|) = \alpha \tanh \left( \frac{\|x\|}{\rho} \right) \]

- **LinearConv** - Only project weights
  \[ h(\|w\|, \|x\|) = \alpha \|x\| \]
Example Designs

Angular Functions

- Square Cosine
  \[ g(\theta_{(w,x)}) = \text{sign}(\cos(\theta)) \cdot \cos^2(\theta) \]
- Linear
  \[ g(\theta_{(w,x)}) = -\frac{2}{\pi} \theta_{(w,x)} + 1 \]
- Cosine
  \[ g(\theta_{(w,x)}) = \cos(\theta_{(w,x)}) \]
- Sigmoid (refer to the paper)
Properties of the magnitude function

• Smoothness: whether the magnitude function is differentiable everywhere. It affects the optimization of the neural network.

• Boundedness: whether the value of the magnitude function is bounded. It affects the adversarial robustness of the neural network (due to the Lipschitz constant).
Special Case: Bounded Magnitude Function

When the magnitude function is bounded, it has the following advantage:

- **Adversarial Robustness**: Bounded operators have smaller Lipschitz constant, which is more robust against adversarial attacks.

- **Optimization**: Boundedness can improve optimization, which is both theoretically and empirically validated by some recent results*

CNN without BN or without ReLU on CIFAR-100

- Operators can significantly outperform baselines w/o BN

<table>
<thead>
<tr>
<th>Method</th>
<th>Linear</th>
<th>Cosine</th>
<th>Sq. Cosine</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN Baseline</td>
<td>-</td>
<td>35.30</td>
<td>-</td>
</tr>
<tr>
<td>LinearConv</td>
<td>33.39</td>
<td>31.76</td>
<td>N/C</td>
</tr>
<tr>
<td>TanhConv</td>
<td><strong>32.88</strong></td>
<td>31.88</td>
<td><strong>34.26</strong></td>
</tr>
<tr>
<td>SegConv</td>
<td>34.69</td>
<td><strong>30.34</strong></td>
<td>N/C</td>
</tr>
</tbody>
</table>

- Operators can significantly outperform baselines w/o ReLU

<table>
<thead>
<tr>
<th>Method</th>
<th>Cosine w/o ReLU</th>
<th>Sq. Cosine w/o ReLU</th>
<th>Cosine w/ ReLU</th>
<th>Sq. Cosine w/ ReLU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>58.24</td>
<td>-</td>
<td>26.01</td>
<td>-</td>
</tr>
<tr>
<td>SphereConv</td>
<td>33.31</td>
<td>25.90</td>
<td>26.00</td>
<td>26.97</td>
</tr>
<tr>
<td>BallConv</td>
<td><strong>31.81</strong></td>
<td>25.43</td>
<td>25.18</td>
<td>26.48</td>
</tr>
<tr>
<td>TanhConv</td>
<td>32.27</td>
<td>25.27</td>
<td>25.15</td>
<td>26.94</td>
</tr>
<tr>
<td>LinearConv</td>
<td>36.49</td>
<td>24.36</td>
<td><strong>24.81</strong></td>
<td>25.14</td>
</tr>
<tr>
<td>SegConv</td>
<td>33.57</td>
<td><strong>24.29</strong></td>
<td>24.96</td>
<td><strong>25.04</strong></td>
</tr>
<tr>
<td>LogConv</td>
<td>33.62</td>
<td>24.91</td>
<td>25.17</td>
<td>25.85</td>
</tr>
<tr>
<td>MixConv</td>
<td>33.46</td>
<td>24.93</td>
<td>25.27</td>
<td>25.77</td>
</tr>
</tbody>
</table>
ImageNet Results

- Decoupled operators are exceptionally good on specific architectures.

<table>
<thead>
<tr>
<th>Method</th>
<th>Standard ResNet-18 w/ BN</th>
<th>Modified ResNet-18 w/ BN</th>
<th>Modified ResNet-18 w/o BN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>12.63</td>
<td>12.10</td>
<td>N/C</td>
</tr>
<tr>
<td>SphereConv</td>
<td>12.68*</td>
<td>11.55</td>
<td>13.30</td>
</tr>
<tr>
<td>LinearConv</td>
<td>11.99*</td>
<td>11.50</td>
<td>N/C</td>
</tr>
<tr>
<td>TanhConv</td>
<td>12.47*</td>
<td><strong>11.10</strong></td>
<td><strong>12.79</strong></td>
</tr>
</tbody>
</table>

DCNet (SphereConv)  
DCNet (TanhConv)
Robustness Against Adversarial Attacks

- Our operators are naturally robust against attacks.

- To attack decoupled networks, it takes much more efforts (L2 norm).

<table>
<thead>
<tr>
<th>Attack</th>
<th>Natural Training</th>
<th>Target models</th>
<th>Adversarial Training</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>SphereConv</td>
<td>BallConv</td>
</tr>
<tr>
<td>None</td>
<td>85.35</td>
<td>88.58</td>
<td>91.13</td>
</tr>
<tr>
<td>FGSM</td>
<td>18.82</td>
<td>43.64</td>
<td>50.47</td>
</tr>
<tr>
<td>BIM</td>
<td>8.67</td>
<td>8.89</td>
<td>7.74</td>
</tr>
</tbody>
</table>

![Graph showing L2 norm of perturbation](image1)

![Graph showing L2 norm of perturbation](image2)
From Function Approximation perspective

- Convolutional neural networks (CNNs) can be viewed as special cases of feed-forward neural networks (or multi-layer perceptrons).

- Although MLPs can achieve the universal approximation, but in practice, CNNs usually work much better than MLPs in vision tasks.
From Function Approximation perspective

• Implication I: which nonlinear function class we use really matters (even if they can be equivalent in theory).

• Implication II: the function class we use should be task-dependent. learning/regularizing/constraining the function class for different tasks is appealing.

• Implication III: our decoupled convolution can be viewed as regularizing the function class of CNNs by using a different similarity measure.
The End

Thank you!

Welcome to our poster for further questions!

F21, Tuesday 4:30-6:30 @ Halls C-E