

Decoupled Networks Presentation

Kai Bian

bkk@glassix.com

2019/08/02

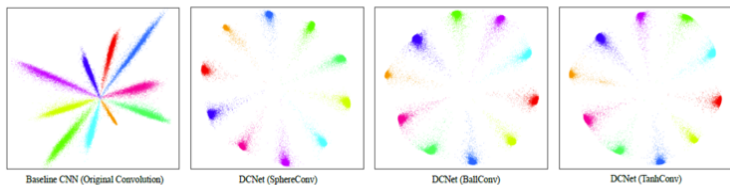


Figure 6: 2D feature visualization on MNIST dataset with natural training.

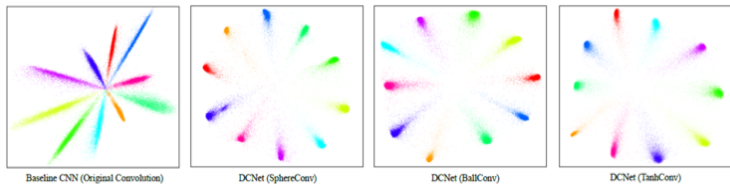


Figure 7: 2D feature visualization on MNIST dataset with adversarial training.

Original inner product-based convolution:

$$\langle w, x \rangle = \|w\| \cdot \|x\| \cdot \cos(\theta_{w,x})$$

naturally decoupled

Decoupled convolution:

$$f(w, x) = h(\|w\|, \|x\|) \cdot g(\theta_{w,x})$$

Magnitude: intra-class variation

Angle: semantic difference

special case

- Bounded by a finite constant regardless of its input and kernel

$$|f_d(\mathbf{w}, \mathbf{x})| \leq c$$

- Hyperspherical Convolution (SphereConv):

$$h(\|\mathbf{w}\|, \|\mathbf{x}\|) = \alpha, \alpha > 0 \quad \longrightarrow \quad f_d(\mathbf{w}, \mathbf{x}) = \alpha \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}),$$

- Hyperball Convolution (BallConv):

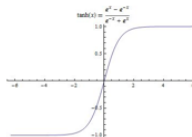
$$f_d(\mathbf{w}, \mathbf{x}) = \alpha \cdot \underbrace{\frac{\min(\|\mathbf{x}\|, \rho)}{\rho}}_{h(\|\mathbf{w}\|, \|\mathbf{x}\|)} \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}),$$

ρ controls saturation threshold.

α scales the output range.

- Hyperbolic Tangent Convolution (TanhConv):

$$f_d(\mathbf{w}, \mathbf{x}) = \alpha \tanh\left(\frac{\|\mathbf{x}\|}{\rho}\right) \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}), \quad \tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



- Linear Convolution (LinearConv):

$$f_d(\mathbf{w}, \mathbf{x}) = \alpha \|\mathbf{x}\| \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}),$$

- Segmented Convolution (SegConv):

$$f_d(\mathbf{w}, \mathbf{x}) = \begin{cases} \alpha \|\mathbf{x}\| \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}), & 0 \leq \|\mathbf{x}\| \leq \rho \\ (\beta \|\mathbf{x}\| + \alpha\rho - \beta\rho) \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}), & \rho < \|\mathbf{x}\| \end{cases}$$

- Logarithm Convolution (LogConv):

$$f_d(\mathbf{w}, \mathbf{x}) = \alpha \log(1 + \|\mathbf{x}\|) \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}),$$

- Mixed Convolution (MixConv):

$$f_d(\mathbf{w}, \mathbf{x}) = \underbrace{(\alpha \|\mathbf{x}\|)}_{\text{LinearConv}} + \underbrace{\beta \log(1 + \|\mathbf{x}\|)}_{\text{LogConv}} \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}),$$

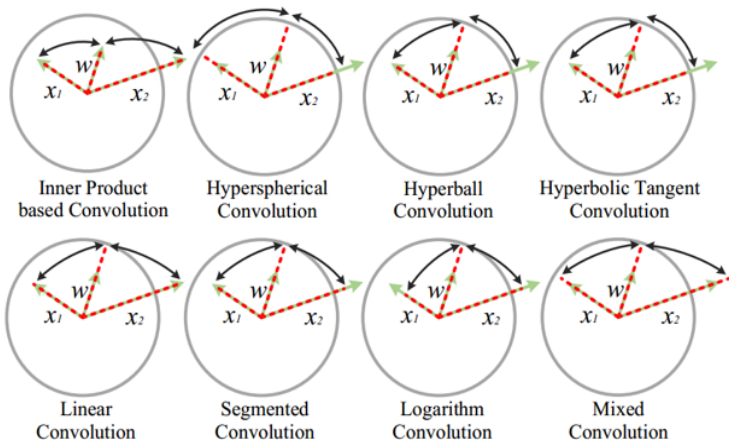


Figure 2: Geometric interpretations for decoupled convolution operators. Green denotes the original vectors, and red denotes the projected vectors.

➤ Operator Radius (ρ):

- gradient change point of the magnitude function

$$f_d(\mathbf{w}, \mathbf{x}) = \alpha \cdot \frac{\min(\|\mathbf{x}\|, \rho)}{\rho} \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}),$$

- SphereConv, LinearConv, LogConv have no operator radius

➤ Boundedness:

- improves the convergence speed and robustness
- makes variance of outputs small
- constrains the Lipschitz constant of neural network, making the entire network more smooth

In particular, a real-valued function $f: \mathbb{R} \rightarrow \mathbb{R}$ is called Lipschitz continuous if there exists a positive real constant K such that, for all real x_1 and x_2 ,

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|.$$

➤ Smoothness:

- better approximation rate, more stable, faster convergence
- more computationally expensive

- Linear Angular Activation:

$$g(\theta_{(w,x)}) = -\frac{2}{\pi}\theta_{(w,x)} + 1,$$

- Cosine Angular Activation:

$$g(\theta_{(w,x)}) = \cos(\theta_{(w,x)}),$$

- Sigmoid Angular Activation:

$$g(\theta_{(w,x)}) = \frac{1 + \exp(-\frac{\pi}{2k})}{1 - \exp(-\frac{\pi}{2k})} \cdot \frac{1 - \exp(\frac{\theta_{(w,x)}}{k} - \frac{\pi}{2k})}{1 + \exp(\frac{\theta_{(w,x)}}{k} - \frac{\pi}{2k})},$$

k controls curvature and can be learned using backpropagation.

- Square Cosine Angular Activation:

$$g(\theta_{(w,x)}) = \text{sign}(\cos(\theta)) \cdot \cos^2(\theta),$$

- Linearly Weighted Decoupled Operator:

$$f_d(\omega, \chi) = \alpha \bullet g(\theta_{(\omega, \chi)}) \quad \longrightarrow \quad f_d(\omega, \chi) = \alpha \bullet \|\omega\| \bullet g(\theta_{(\omega, \chi)})$$

- Non-linearly Weighted Decoupled Operator:

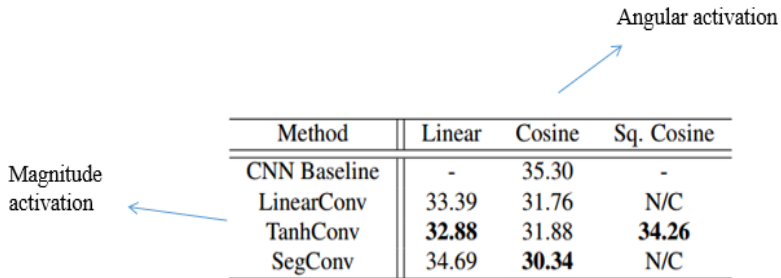
$$f_d(\mathbf{w}, \mathbf{x}) = \alpha \tanh\left(\frac{\|\mathbf{x}\|}{\rho}\right) \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}), \quad \longrightarrow \quad f_d(\mathbf{w}, \mathbf{x}) = \alpha \tanh\left(\frac{1}{\rho} \|\mathbf{x}\| \cdot \|\mathbf{w}\|\right) \cdot g(\theta_{(\mathbf{w}, \mathbf{x})}).$$

- In practice, linearly weighted operators are favored over nonlinearly weighted ones due to simplicity.

Layer	Plain CNN-9	CNN-9 for adversarial attacks	ResNet-32 for CIFAR	Standard ResNet-18	Modified ResNet-18
Conv0.x	N/A	N/A	$[3 \times 3, 96]$	$[7 \times 7, 64], S2$ $3 \times 3, \text{Max Pooling}, S2$	$[7 \times 7, 128], S2$ $3 \times 3, \text{Max Pooling}, S2$
Conv1.x	$[3 \times 3, 64] \times 3$ $2 \times 2 \text{ Max Pooling}, S2$	$[3 \times 3, 32] \times 3$ $2 \times 2 \text{ Max Pooling}, S2$	$\begin{bmatrix} 3 \times 3, 96 \\ 3 \times 3, 96 \end{bmatrix} \times 5$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 1, S2$
Conv2.x	$[3 \times 3, 128] \times 3$ $2 \times 2 \text{ Max Pooling}, S2$	$[3 \times 3, 64] \times 3$ $2 \times 2 \text{ Max Pooling}, S2$	$\begin{bmatrix} 3 \times 3, 192 \\ 3 \times 3, 192 \end{bmatrix} \times 5$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 1, S2$
Conv3.x	$[3 \times 3, 256] \times 3$ $2 \times 2 \text{ Max Pooling}, S2$	$[3 \times 3, 128] \times 3$ $2 \times 2 \text{ Max Pooling}, S2$	$\begin{bmatrix} 3 \times 3, 384 \\ 3 \times 3, 384 \end{bmatrix} \times 5$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 1, S2$
Conv4.x	N/A	N/A	N/A	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$[3 \times 3, 1024] \times 1, S2$
Final	512-dim fully connected	256-dim fully connected	Average Pooling		

- For Cifar, trained by ADAM with 128 batch size. Learning rate starts from 0.001.
- For ImageNet, trained by SGD with momentum 0.9 and batch size 40. Learning rate starts from 0.1.
- For adversarial attacks, trained by ADAM. Learning rates are divided by 10 when error plateaus.

➤ Learning without Batch Normalization:



Method	Linear	Cosine	Sq. Cosine
CNN Baseline	-	35.30	-
LinearConv	33.39	31.76	N/C
TanhConv	32.88	31.88	34.26
SegConv	34.69	30.34	N/C

Table 2: Testing error (%) of plain CNN-9 without BN on CIFAR-100. “N/C” indicates that the model can not converge. “-” denotes no result. The results of different columns belong to different angular activation.

➤ Learning without ReLU:

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

Table 3: Testing error rate (%) of plain CNN-9 on CIFAR-100. Note that, BN is used in all compared models. Baseline is the original plain CNN-9.

➤ Comparison on Cifar

Method	CIFAR-10	CIFAR-100
ResNet-110-original [5]	6.61	25.16
ResNet-1001 [6]	4.92	22.71
ResNet-1001 (64 mini-batch size) [6]	4.64	-
DCNet-32 (TanhConv + Cosine)	4.75	21.12
DCNet-32 (LinearConv + Sq. Cos.)	5.34	20.23

Table 5: Comparison to the state-of-the-art on CIFAR-10 and CIFAR-100.

➤ Comparison on ImageNet

Method	Standard ResNet-18 w/ BN	Modified ResNet-18 w/ BN	Modified ResNet-18 w/o BN
Baseline	12.63	12.10	N/C
SphereConv	12.68*	11.55	13.30
LinearConv	11.99*	11.50	N/C
TanhConv	12.47*	11.10	12.79

Table 6: Center-crop Top-5 error (%) of standard ResNet-18 and modified ResNet-18 on ImageNet-2012. * indicates we use the pretrained model of original CNN on ImageNet-2012 as initialization (see Section 4.3).

➤ FGSM – Fast Gradient Sign Method

- Explaining and Harnessing Adversarial Examples by [Goodfellow](#).
- Adds weak noise on the original images along the direction of gradients, making model misclassify image X .
- Finds adversarial perturbations which increase the value of the loss function.

$$X^{adv} = X + \epsilon \operatorname{sign}(\nabla_X J(X, y_{true}))$$

Offset
Gradient direction of loss function at point x

➤ BIM – Basic Iterative Method

- A straightforward extension of FGSM: apply it multiple times with small step size:

$$X_0^{adv} = X, \quad X_{N+1}^{adv} = \operatorname{Clip}_{X, \epsilon} \left\{ X_N^{adv} + \alpha \operatorname{sign}(\nabla_X J(X_N^{adv}, y_{true})) \right\}$$

Element-wise clipping

➤ **White-box attack:**

- Attackers know the ML algorithm and parameters. Attackers can interact with ML system during adversarial attacking.

➤ **Black-box attack:**

- Attackers do not know the ML algorithm or parameters, but attackers can interact with ML system, such as observing and judging the outputs of given inputs.

White-box attack:

Attack	Target models			
	Baseline	SphereConv	BallConv	TanhConv
None	85.35	88.58	91.13	91.45
FGSM	18.82	43.64	50.47	52.60
BIM	8.67	8.89	7.74	10.18
None	83.70	87.41	87.47	87.54
FGSM	78.96	85.98	82.20	81.46
BIM	7.96	35.07	17.38	19.86

Table 7: White-box attacks on CIFAR-10. Performance is measured in accuracy (%). The first three rows are results of naturally trained models, and the last three rows are results of adversarially trained models.

Black-box attack:

Attack	Target models			
	Baseline	SphereConv	BallConv	TanhConv
None	85.35	88.58	91.13	91.45
FGSM	50.90	56.71	49.50	50.61
BIM	36.22	43.10	27.48	29.06
None	83.70	87.41	87.47	87.54
FGSM	77.57	76.29	78.67	80.38
BIM	78.55	77.79	80.59	82.47

Table 8: Black-box attacks on CIFAR-10. Performance is measured in accuracy (%). The first three rows are results of naturally trained models, and the last three rows are results of adversarially trained models.