Deep Hyperspherical Learning

Weiyang Liu[1], Yan-Ming Zhang[2], Xingguo Li[1,3], Zhiding Yu[4], Bo Dai[1], Tuo Zhao[1], Le Song[1]

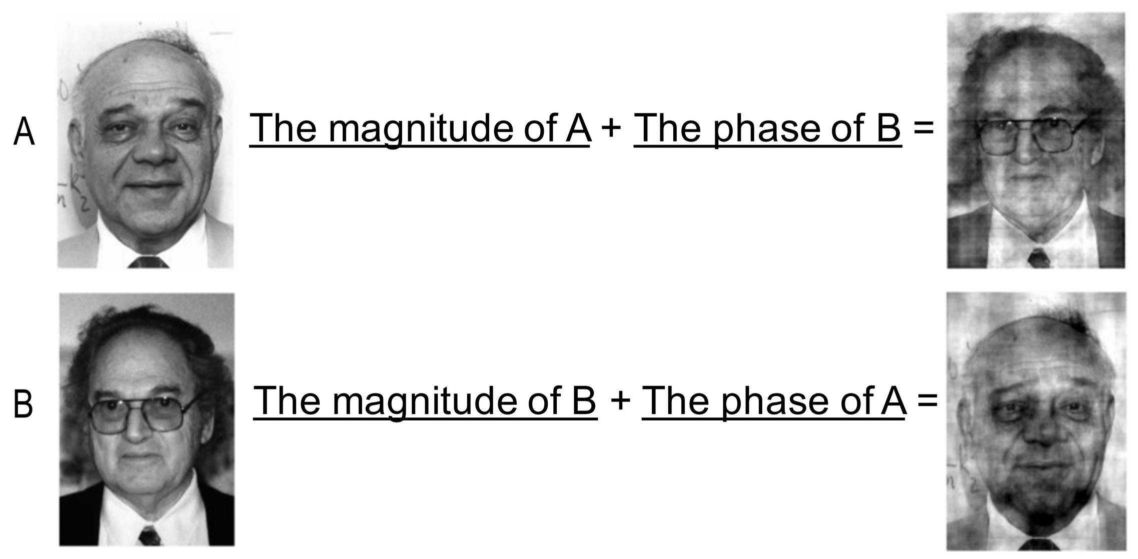
1. Georgia Tech 2. CASIA 3. Univ. of Minnesota 4. CMU

NIPS 2017
LONG BEACH CA | DEC 4 - 9 | NIPS.CC

Introduction

Motivation I

- 2D Fourier Transform for images



- Phase contains the crucial discriminative information!

Motivation II

- Angles (Phase) usually give bounded output, avoiding covariate shift problem and stablize the network training.
- For example, we usually use the cosine function of angles, which produces output from -1 to 1. The internal covariate shift can be largely prevented.

Motivation III

- By only using the angular information in the network learning, we could largely reduce the learning space of parameters, which could accelerate the training process and speed up the convergence.

SphereNet: A Neural Network Learned on Hyperspheres

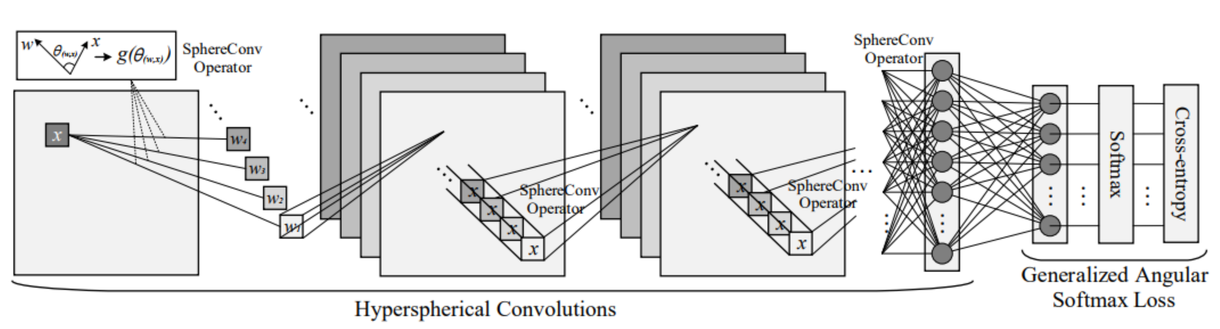
Hyperspherical Convolutional (SphereConv) Operator:

$$\mathcal{F}_s(\boldsymbol{w},\boldsymbol{x}) = g(\theta_{(\boldsymbol{w},\boldsymbol{x})}) + b_{\mathcal{F}_s}$$

where $\theta_{(\boldsymbol{w},\boldsymbol{x})}$ is the angle between the kernel parameter w and the local patch x. A simple example is cosine SphereConv:

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

We use this SphereConv operator to replace the original inner product based convolutional operator in the CNNs, and propose the *SphereNet*. (SphereNet comes from that angle can be viewed as the geodesic distance on a unit hypersphere)



Hyperspherical Convolutional Operator

- Three SphereConv operators:

linear SphereConv

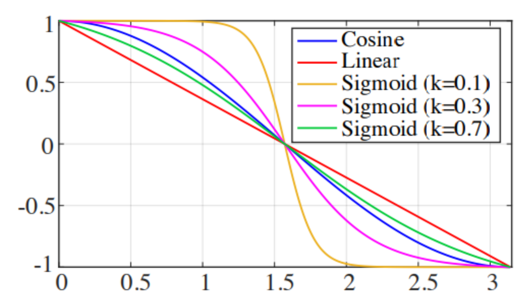
$$g(\theta_{(\boldsymbol{w},\boldsymbol{x})}) = a\theta_{(\boldsymbol{w},\boldsymbol{x})} + b$$

cosine SphereConv

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

sigmoid SphereConv

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



- Besides the predefined SphereConv operators, we further consider the *learnable SphereConv*.
- SphereConv can also be used to the fully connected layers, recurrent layers, etc.
- We also design angular loss functions for *SphereConv*, i.e., generalized angular softmax (GA-Softmax) loss

Theoretical Insights

- Suppose the observation is $\boldsymbol{F} = \boldsymbol{U}^*\boldsymbol{V}^{*\top}$ (ignore the bias), where $\boldsymbol{U}^* \in \mathbb{R}^{n\times k}$ is the weight, $\boldsymbol{V}^* \in \mathbb{R}^{m\times k}$ is the input that embeds weights from previous layers.

Scaling Issue of Neural Nets:

- Consider the objective:

$$\min_{\boldsymbol{U}\in\mathbb{R}^{n\times k},\boldsymbol{V}\in\mathbb{R}^{m\times k}} \mathcal{G}(\boldsymbol{U},\boldsymbol{V}) = \frac{1}{2}\|\boldsymbol{F} - \boldsymbol{U}\boldsymbol{V}^\top\|_{\mathrm{F}}^2$$

__Lemma1__: Consider a pair of global optimal points $\boldsymbol{U},\boldsymbol{V}$ satisfying $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{V}^\top$ and $\mathrm{Tr}(\boldsymbol{V}^\top\boldsymbol{V}\otimes\boldsymbol{I}_n) \leq \mathrm{Tr}(\boldsymbol{U}^\top\boldsymbol{U}\otimes\boldsymbol{I}_m)$. For any real $c>1$, let $\widetilde{\boldsymbol{U}} = c\boldsymbol{U}$ and $\widetilde{\boldsymbol{V}} = \boldsymbol{V}/c$, then we have $\kappa(\nabla^2\mathcal{G}(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}})) = \Omega(c^2\kappa(\nabla^2\mathcal{G}(\boldsymbol{U},\boldsymbol{V})))$, where $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$ is the restricted condition number with $\lambda_{\max}$ being the largest and $\lambda_{\min}$ being the smallest nonzero eigenvalues.

Insensitiveness to Scaling for SphereConv:

- Consider our proposed cosine SphereConv operator, an equivalent problem is:

$$\min_{\boldsymbol{U}\in\mathbb{R}^{n\times k},\boldsymbol{V}\in\mathbb{R}^{m\times k}} \mathcal{G}_S(\boldsymbol{U},\boldsymbol{V}) = \frac{1}{2}\|\boldsymbol{F} - \boldsymbol{D}_{\boldsymbol{U}}\boldsymbol{U}\boldsymbol{V}^\top\boldsymbol{D}_{\boldsymbol{V}}\|_{\mathrm{F}}^2$$

where $\boldsymbol{D}_{\boldsymbol{U}} = \mathrm{diag}\left(\frac{1}{\|\boldsymbol{U}_{1,:}\|_2},\dots,\frac{1}{\|\boldsymbol{U}_{n,:}\|_2}\right) \in \mathbb{R}^{n\times n}$ and $\boldsymbol{D}_{\boldsymbol{V}} = \mathrm{diag}\left(\frac{1}{\|\boldsymbol{V}_{1,:}\|_2},\dots,\frac{1}{\|\boldsymbol{V}_{m,:}\|_2}\right) \in \mathbb{R}^{m\times m}$ are diagonal matrices.

__Lemma2__: For any real $c>1$, let $\widetilde{\boldsymbol{U}} = c\boldsymbol{U}$ and $\widetilde{\boldsymbol{V}} = \boldsymbol{V}/c$, then we have $\lambda_i(\nabla^2\mathcal{G}_S(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}})) = \lambda_i(\nabla^2\mathcal{G}_S(\boldsymbol{U},\boldsymbol{V}))$ for all $i \in [(n+m)k] = \{1,2,\dots,(n+m)k\}$ and $\kappa(\nabla^2\mathcal{G}(\widetilde{\boldsymbol{U}},\widetilde{\boldsymbol{V}})) = \kappa(\nabla^2\mathcal{G}(\boldsymbol{U},\boldsymbol{V}))$, where κ is defined as in Lemma1.

- Regular Neural Nets: scales as $\Omega(c^2)$
- SphereConv: ***insensitive*** to scaling

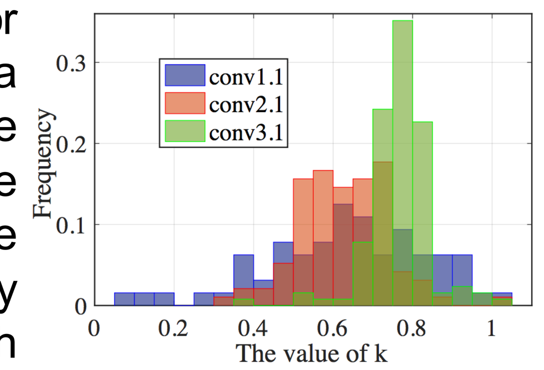
Learnable SphereConv Operator

- With sigmoid SphereConv, we naturally come up with a learnable SphereConv. Specifically, we propose to learn the parameter k in the sigmoid SphereConv.

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

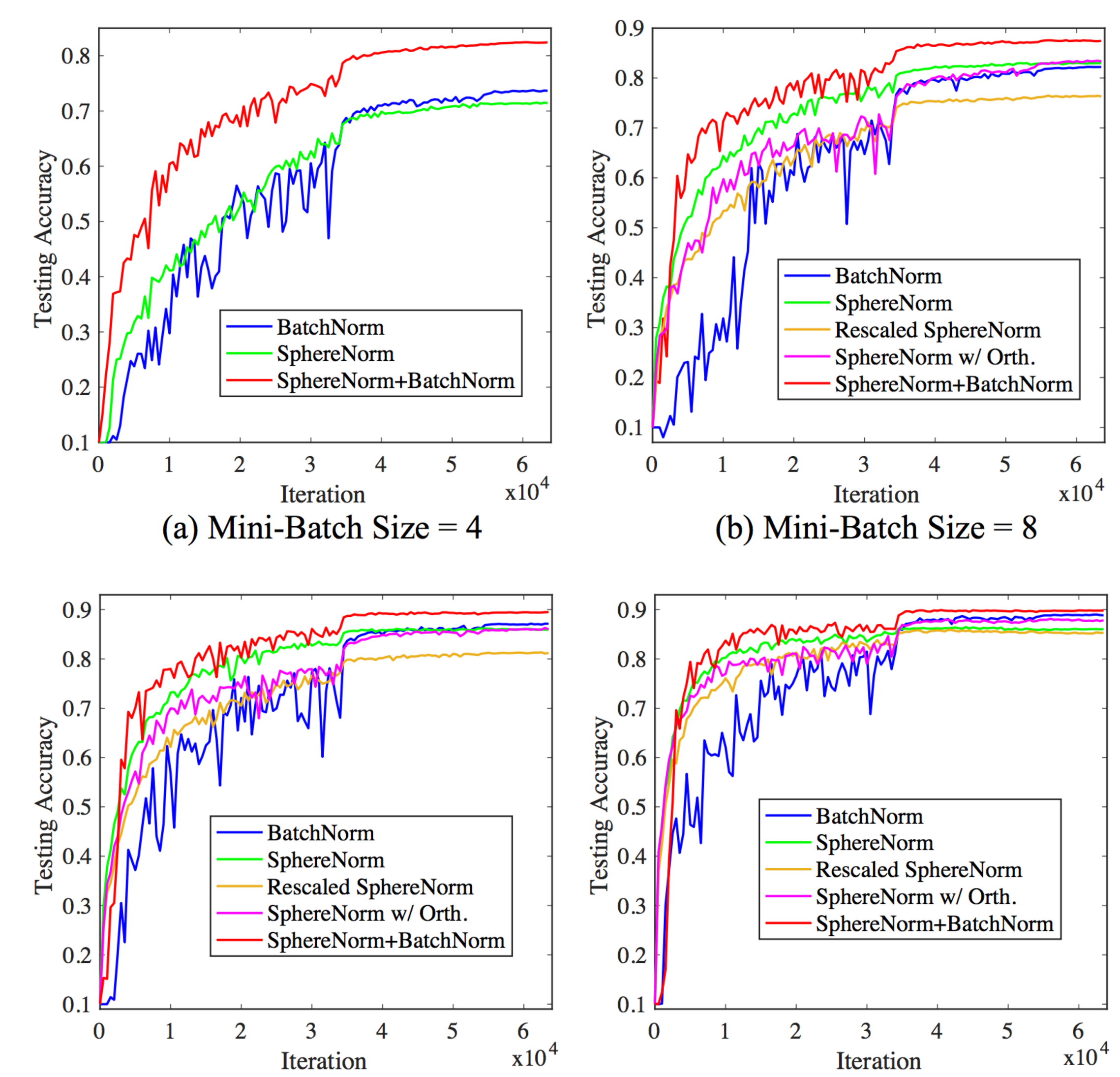
k is learned by back-prop!

- K is updated using $k^{t+1} = k^t + \eta\frac{\partial L}{\partial k}$ where t denotes the iteration and $\frac{\partial L}{\partial k}$ is computed by the chain rule.
- Preliminary results: we learn a parameter k independently for each kernel and draw a frequency histogram for the value of k. Note that, we initialize all k with the same constant 0.5. The final accuracy can be further boosted with learnable SphereConv.



SphereNorm: a New Normalization Method

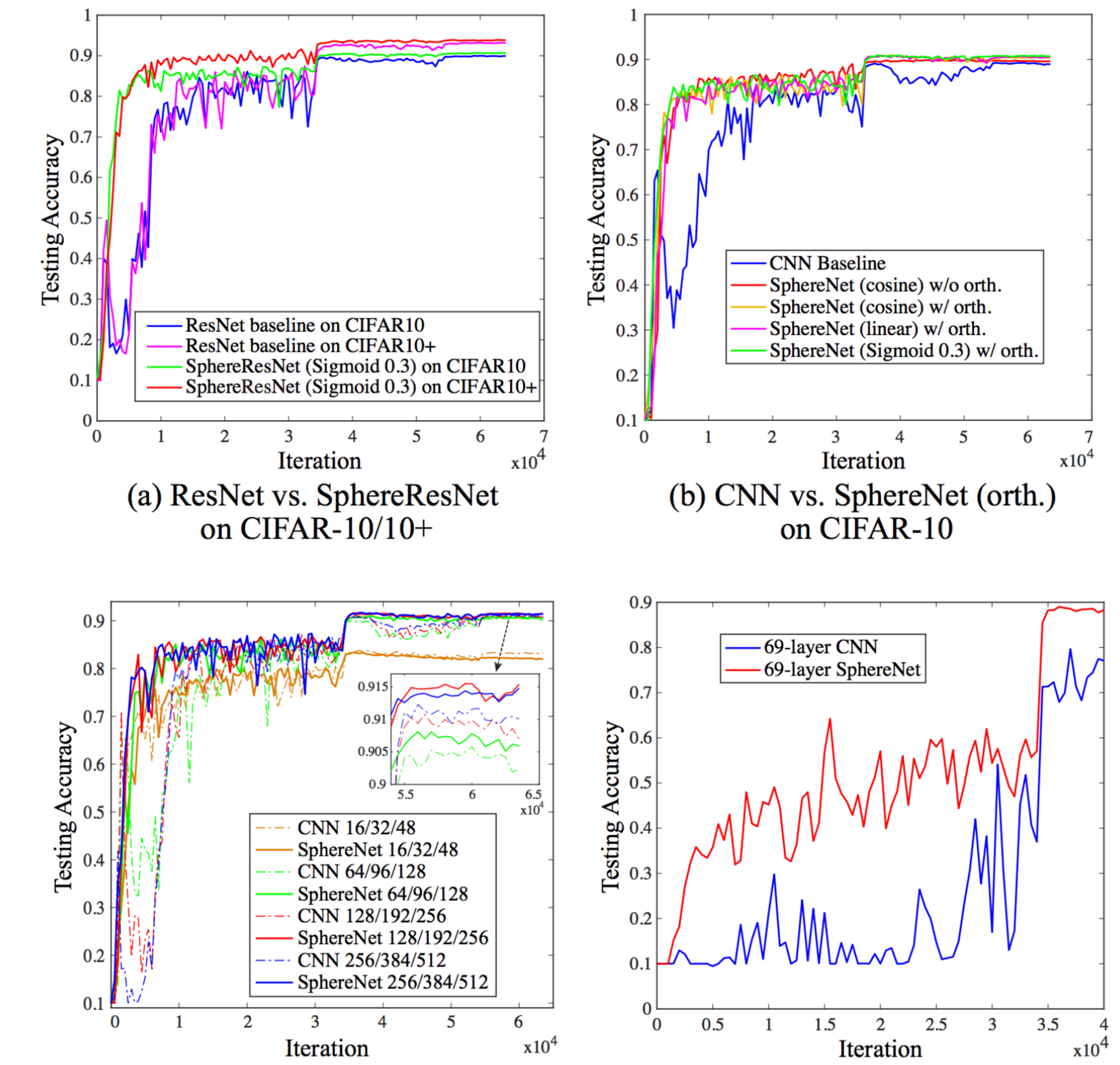
- Similar to batch normalization (BatchNorm), we note that the hyperspherical learning can also be viewed as a way of normalization, because SphereConv constrain the output value in [−1, 1] ([0, 1] after ReLU).
- Different from BatchNorm, SphereNorm normalizes the network based on spatial information and the weights, so it has nothing to do with the mini-batch statistic.
- SphereNorm and BatchNorm are complimentary to each other and could be used simultaneously.



(a) Mini-Batch Size = 4
(b) Mini-Batch Size = 8
(c) Mini-Batch Size = 16
(d) Mini-Batch Size = 32

Experiments on CIFAR-10 and CIFAR100

- On both CIFAR-10 and CIFAR-100, we observe the faster convergence on multiple network architectures like plain CNNs and ResNets.



(a) ResNet vs. SphereResNet on CIFAR-10/10+
(b) CNN vs. SphereNet (orth.) on CIFAR-10
(c) Different width of SphereNet on CIFAR-10
(d) Deep CNN vs. SphereNet on CIFAR-10

- Our SphereResNet uses only 34 layers to perform comparably to the 1001-layer ResNet.

Method	CIFAR-10+	CIFAR-100
ELU [2]	94.16	72.34
FitResNet (LSUV) [14]	93.45	65.72
ResNet-1001 [7]	**95.38**	**77.29**
Baseline ResNet-32 (softmax)	93.26	72.85
SphereResNet-32 (S-SW)	94.47	76.02
SphereResNet-32 (L-LW)	94.33	75.62
SphereResNet-32 (C-CW)	94.64	74.92
SphereResNet-32 (S-G)	**95.01**	**76.39**

Experiments on Imagenet-2012

- Our SphereResNet also shows much faster convergence on large-scale dataset like Imagenet-2012.
- The error is the single central crop error.