Neural Similarity Learning

Weiyang Liu[1*], Zhen Liu[2*], James M. Rehg[1], Le Song[1]

*1. Georgia Tech 2. Mila, University of Montreal * Equal Contribution*

Code is available at
https://github.com/wy1iu/NSL

Background

- Convolution operator contains two components:
  - Learnable template (Kernel)
  - Similarity measure (inner product)
- Learning (modifying) the shape of kernel:
  - Dilated (atrous) convolution
  - Deformable convolution, Active convolution
- Learning (modifying) the similarity measure:
  - Hyperspherical convolution
  - Decoupled convolution
- Our work aims to generalize the current convolution operator by jointly learning both kernel shape and similarity measure.

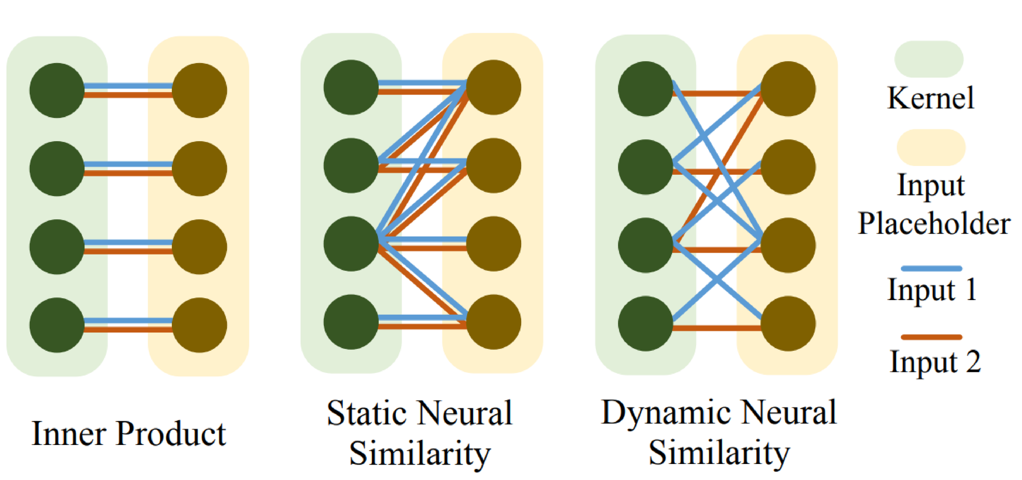
Motivation

- Hand-designed inner-product based convolution is unlikely to be optimal for every task.
- Optimizing an underdetermined quadratic objective over a matrix $\boldsymbol{W}$ with gradient descent on a factorization of this matrix leads to an implicit regularization for the solution

Main Contribution

- **Neural similarity** generalizes the inner product via bilinear similarity.
- **Neural similarity network** stacks convolution layers with neural similarity.
- **Static** and **dynamic** learning strategies for the neural similarity.
- Significant performance gain in visual recognition and few-shot learning.

High-level Comparison with Inner Product



- A line represents a multiplication operation and a circle denotes an element in a vector. Green color denotes kernel and yellow denotes input.

Neural Similarity Learning

- Notation:
  - $\tilde{\boldsymbol{W}}$: a convolution kernel with size $C\times H\times V$.
  - $\boldsymbol{W}=\{\tilde{\boldsymbol{W}}^F_{1,:,:},\tilde{\boldsymbol{W}}^F_{2,:,:},\cdots,\tilde{\boldsymbol{W}}^F_{C,:,:}\}\in\mathbb{R}^{CHV}$: a flatten kernel.
  - $\boldsymbol{X}$: a flatten input patch.
- Generalizing convolution with bilinear similarity:

$$f_{\boldsymbol{M}}(\boldsymbol{W},\boldsymbol{X})=\boldsymbol{W}^\top\boldsymbol{M}\boldsymbol{X}$$

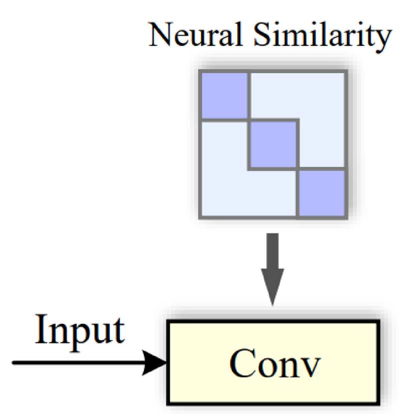
where $\boldsymbol{M}\in\mathbb{R}^{CHV\times CHV}$ denotes the bilinear similarity matrix.

- Constraining M to be block-diagonal:

$$f_{\boldsymbol{M}}(\boldsymbol{W},\boldsymbol{X})=\boldsymbol{W}^\top\begin{bmatrix}\boldsymbol{M}_s & & \\ & \ddots & \\ & & \boldsymbol{M}_s\end{bmatrix}\boldsymbol{X}$$

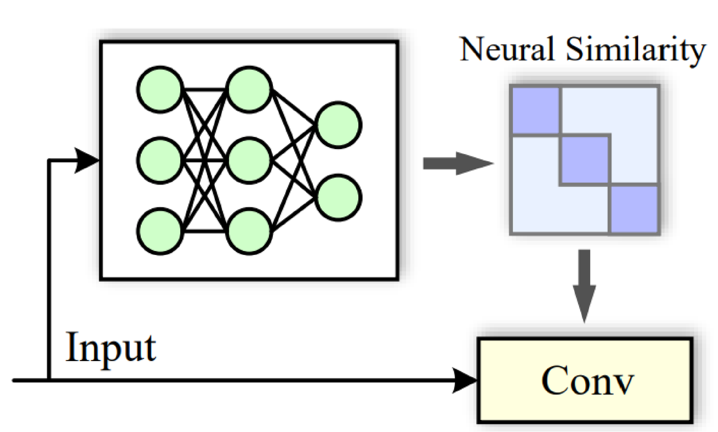
where $\boldsymbol{M}=\mathrm{diag}(\boldsymbol{M}_s,\cdots,\boldsymbol{M}_s)$ and $\boldsymbol{M}_s$ is of size $HV\times HV$. Note that, hyperspherical convolution becomes a special case of this bilinear formulation if $\boldsymbol{M}$ is a diagonal matrix with diagonal being $\frac{1}{\|\boldsymbol{W}\|\|\boldsymbol{X}\|}$.

Learning Static Neural Similarity



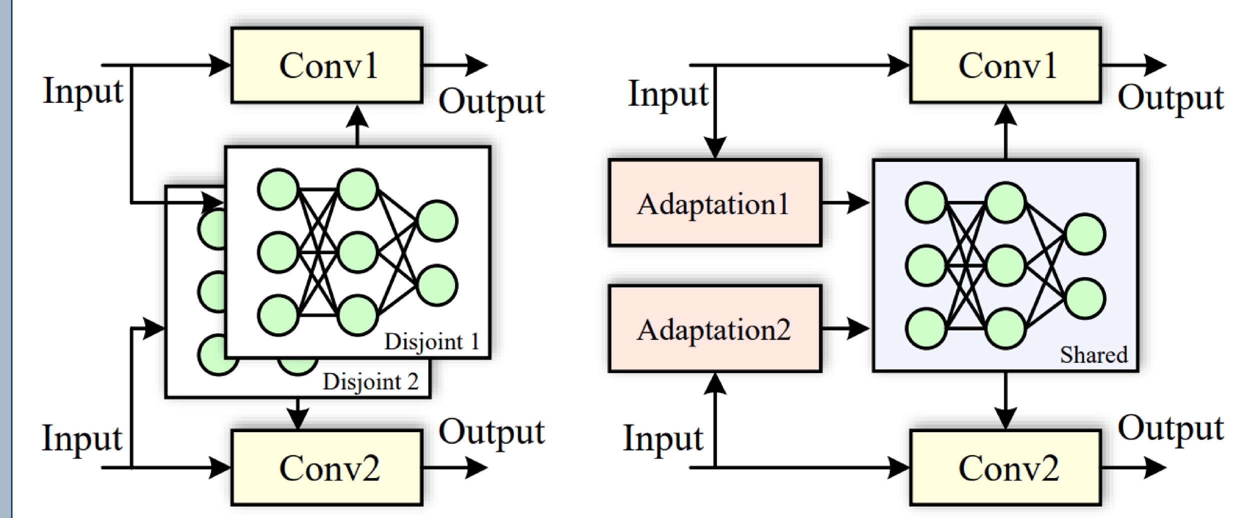
- We learn the matrix $\boldsymbol{M}$ jointly with the convolution kernel via back-propagation.
- Learning static neural similarity can be viewed as a factorized learning of neurons.
- Recent theories suggest that such factorization tends to give minimum nuclear norm solution.

Learning Dynamic Neural Similarity



- We use a neural network to predict the neural similarity.
- Such neural similarity is dynamic in the sense that it is dependent on the input and dynamically determines the neural similarity during inference.
- It is equivalent to a **dynamic neural network**.

Disjoint and Shared Parameterization



(a) Disjoint Parameterization (b) Shared Parameterization

Learning Both Kernel Shape and Similarity

$$f_{\boldsymbol{M}}(\boldsymbol{W},\boldsymbol{X})=\boldsymbol{W}^\top\begin{bmatrix}\boldsymbol{DR} & & \\ & \ddots & \\ & & \boldsymbol{DR}\end{bmatrix}\boldsymbol{X}$$

$$=\boldsymbol{W}^\top\cdot\underbrace{\begin{bmatrix}\boldsymbol{D} & & \\ & \ddots & \\ & & \boldsymbol{D}\end{bmatrix}}_{\text{Kernel Shape}}\cdot\underbrace{\begin{bmatrix}\boldsymbol{R} & & \\ & \ddots & \\ & & \boldsymbol{R}\end{bmatrix}}_{\text{Similarity Measure}}\cdot\boldsymbol{X}$$

where $\boldsymbol{D}=\mathrm{diag}(d_1,\cdots,d_{HV})$ and $d_i\in\{0,1\},\forall i$.

Theoretical Insights

- **Implicit regularization induced by NSL**: NSL can be viewed as a form of matrix multiplication where the weight matrix $\boldsymbol{W}$ is factorized as $\boldsymbol{M}^T\boldsymbol{W}'$.
- Such factorization form not only provides more modeling and regularization flexibility, but it also introduces an **implicit regularization** (in gradient descent).
- Comparison of gradient flow:

Standard derivative

$$\dot{\boldsymbol{W}}_t=\sum_i\boldsymbol{X}_i(\boldsymbol{y}_i-\boldsymbol{W}_t^\top\boldsymbol{X}_i)^\top=\sum_i\boldsymbol{X}_i(\boldsymbol{r}_t^i)^\top$$

NSL derivative

$$\dot{\boldsymbol{W}}_t=\boldsymbol{M}_t^\top\dot{\boldsymbol{W}}'_t+\dot{\boldsymbol{M}}_t^\top\boldsymbol{W}'_t$$
$$=\boldsymbol{M}_t^\top\boldsymbol{M}_t\sum_i\boldsymbol{X}_i(\boldsymbol{r}_t^i)^\top+\sum_i\boldsymbol{X}_i(\boldsymbol{r}_t^i)^\top\boldsymbol{W}'^\top_t\boldsymbol{W}'_t$$

- Connection to dynamic neural unit (DNU): an isolated DNU is given by a differential equation:

$$\dot{\boldsymbol{x}}(t)=-\alpha\boldsymbol{x}(t)+f(\boldsymbol{w},\boldsymbol{x}(t),\boldsymbol{u}),\ \boldsymbol{y}(t)=g(\boldsymbol{x}(t))$$

- Different from DNU, dynamic NSN does not have the state feedback and self-recurrence.

Generic Image Recognition

Method	Error (%)
Baseline CNN	7.78
Dynamic NSN (Shared)	7.20
Dynamic NSN (Disjoint)	**6.85**

Error of different parameterization on CIFAR-100

- Shared parameterization has better generalizability than disjoint parameterization.

Method	CIFAR-10	CIFAR-100
Baseline CNN	7.78	28.95
Baseline CNN++	7.29	28.70
Static NSN w/ DNS	7.15	28.35
Static NSN w/ UNS	7.38	28.11
Dynamic NSN w/ DNS	6.85	**27.81**
Dynamic NSN w/ UNS	**6.5**	28.02

Testing error on CIFAR-10 and CIFAR-100

Method	Top-1	Top-5	# params
Baseline CNN	42.72	19.11	8.90M
Baseline CNN++	42.11	18.98	9.71M
Dynamic NSN w/ DNS	**40.61**	**18.04**	9.61M

Testing error on ImageNet-2012

- NSL generally yields **better generalization power**.
- NSL has **better parameter efficiency**.
- NSL **does not affect the inference speed** and has the same inference speed as its CNN counterpart.

Few-shot Image Recognition

Method	Backbone	5-shot Accuracy
Finetuning Baseline	CNN-4	49.79 ± 0.79
Nearest Neightbor Baseline	CNN-4	51.04 ± 0.65
MatchingNet	CNN-4	55.31 ± 0.73
ProtoNet	CNN-4	68.20 ± 0.66
MAML	CNN-4	63.15 ± 0.91
RelationNet	CNN-4	65.32 ± 0.70
Static NSN (ours)	CNN-4	65.74 ± 0.68
Meta-learned static NSN (ours)	CNN-4	66.21 ± 0.69
Dynamic NSN (ours)	CNN-4	**71.26 ± 0.65**
Discriminative k-shot	ResNet-34	73.90 ± 0.30
Tadam	ResNet-12	76.7 ± 0.3
LEO	ResNet-28	**77.59 ± 0.12**
Dynamic NSN (ours)	CNN-9	77.44 ± 0.63

Few-shot classification on Mini-ImageNet test set

- Meta-learned static NSN is to meta-learn the neural similarity matrix $\boldsymbol{M}$ during training.
- NSL generally has **better generalization power** on few-shot learning.
- Dynamic NSL performs the best and also outperforms the variant where $\boldsymbol{M}$ is meta-learned instead of being learned by a neural network.