

Siyuan Qiao is a fourth-year Ph.D. student at Johns Hopkins University, where he is advised by Bloomberg Distinguished Professor Alan Yuille. Before that, he received B.E. in Computer Science at Shanghai Jiao Tong University. He has spent time at Adobe Inc., Baidu IDL, UCLA, and YITU Technology. His research interest lies in computer vision and deep learning.

<https://www.cs.jhu.edu/~syqiao/>

Few-Shot Image Recognition by Predicting Parameters from Activations

Siyuan Qiao, Chenxi Liu, Wei Shen, Alan L Yuille

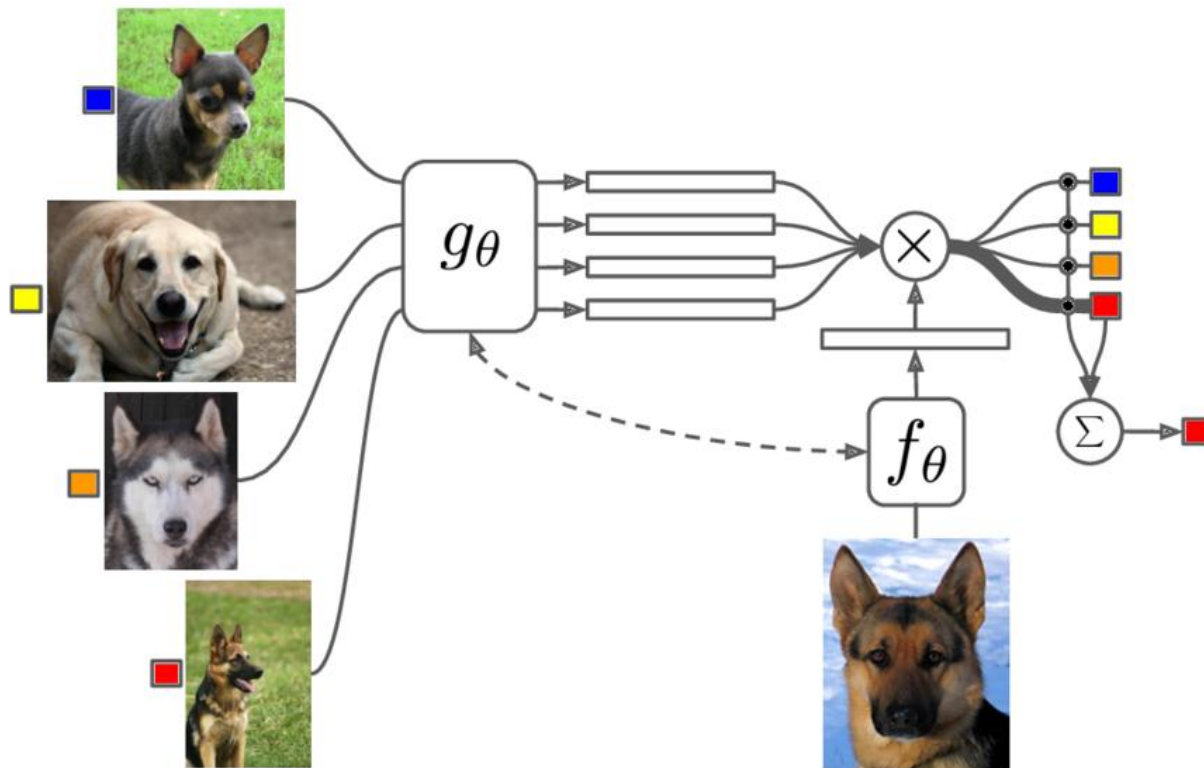
CVPR 2018, Salt Lake City

Feb 18, 2020

Introduction

Introduction

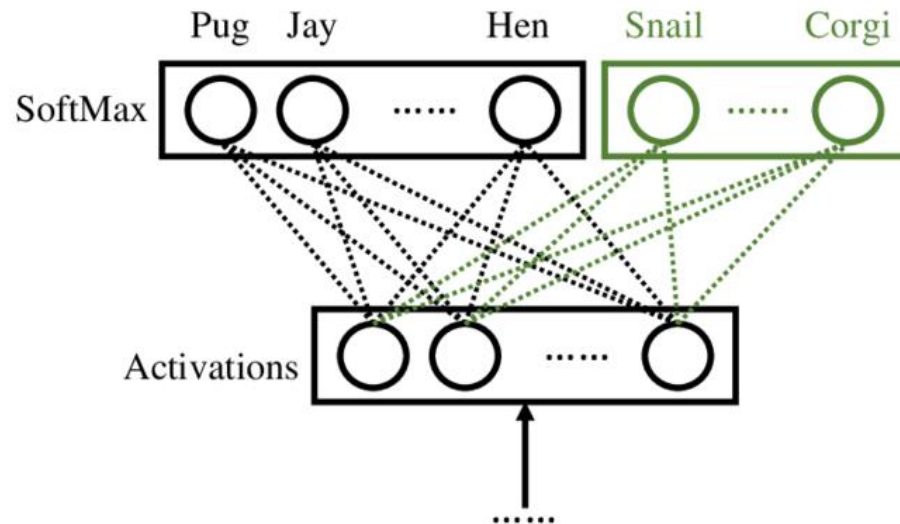
- m -shot n -way image recognition¹:
 - m -shot: each new class has m training images.
 - n -way: predict the class of test images from n classes.



¹Image source: Vinyals, Oriol, et al. "Matching networks for one shot learning." NeurIPS, 2016.

Introduction

- Few-shot + large-scale image recognition.



- Pre-training on large-scale datasets (black) and few-shot adaptation to new classes (green). The green circles are the novel classes, and the green lines represent the unknown parameters for categories.

Two dataset setting

Few-shot \mathcal{D}_{few} + large-scale $\mathcal{D}_{\text{large}}$.

Objective

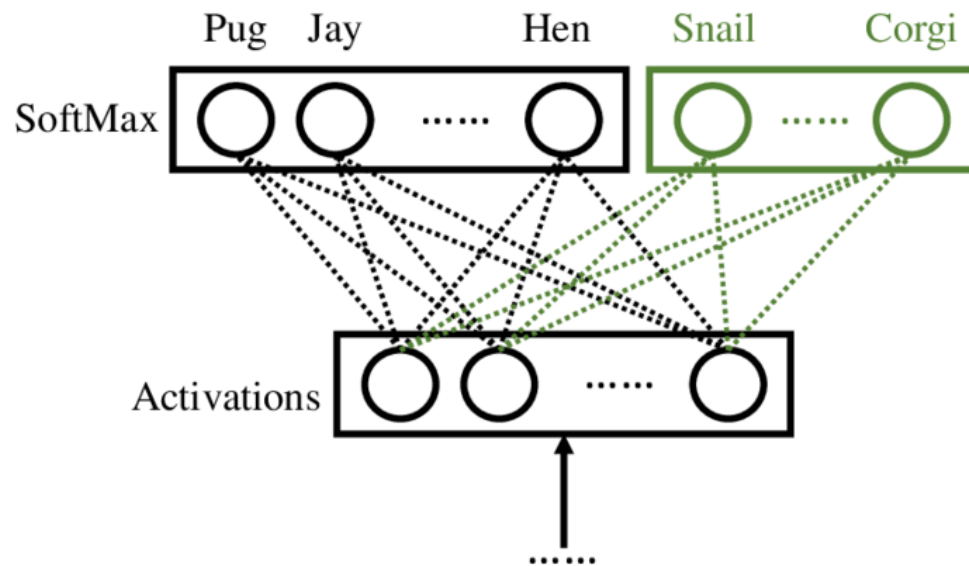
Good performances on both \mathcal{D}_{few} and $\mathcal{D}_{\text{large}}$.

Desired properties

- Good performances on \mathcal{D}_{few} .
- Little performance affects on $\mathcal{D}_{\text{large}}$.
- Fast inference and easy adaptation with little training.

An Example

- Training on large-scale datasets is standard: we can train a deep neural network for them – the black part of the figure.
- What is missing is the green part for the few-shot categories.
- It is hard to use gradient descent to get the green part due to the data insufficiency.



Observation

What we have

An activation function for each example trained on large-scale sets.

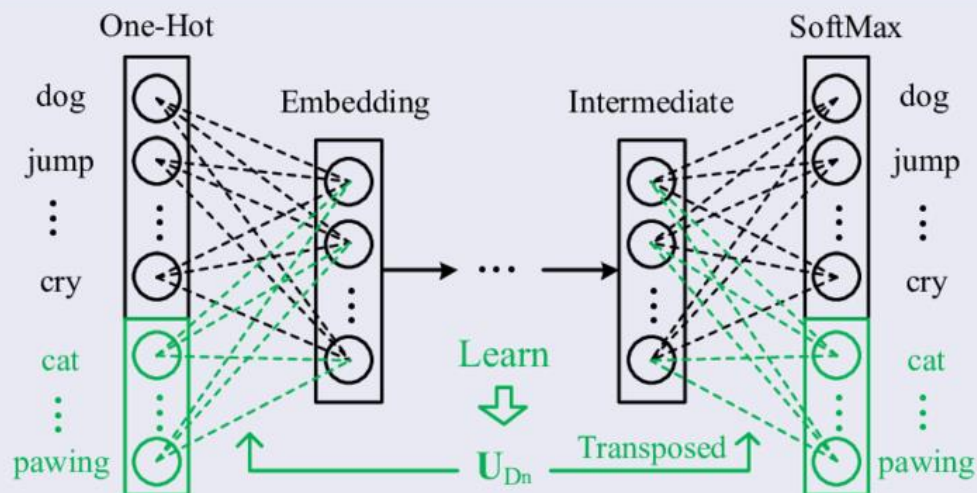
What we need

The unknown parameters for the few-shot categories.

A possible solution

It will be so good if we can bridge what we have and what we want.

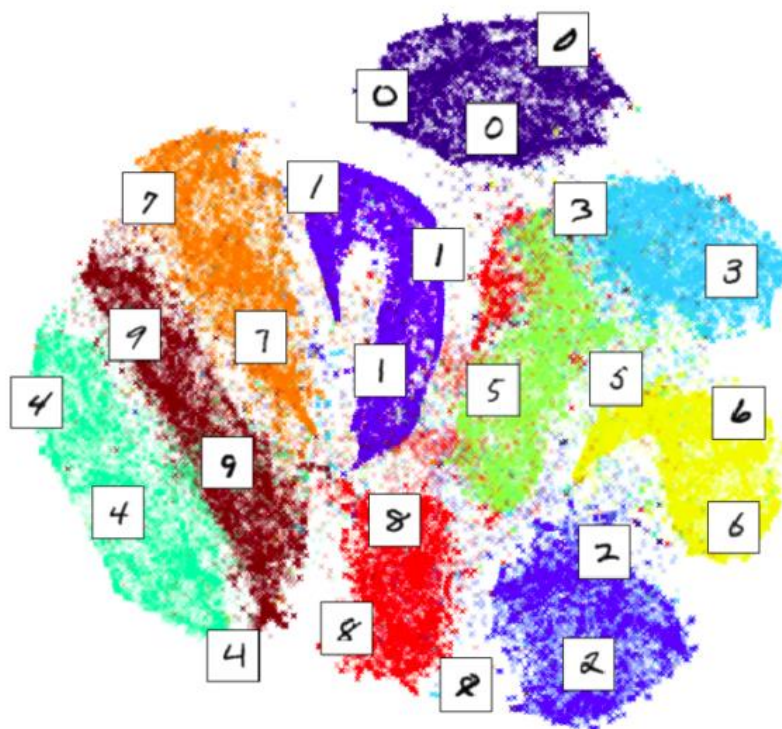
Why We Do This?



Based on a paper from our group:

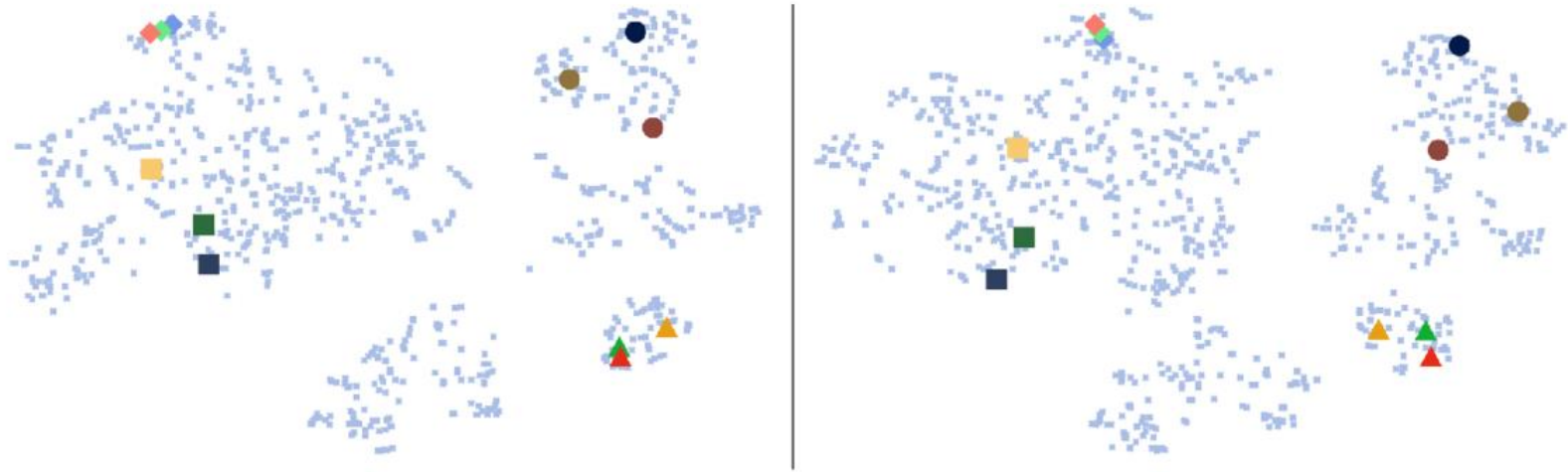
Mao, J., Wei, X., Yang, Y., Wang, J., Huang, Z. and Yuille, A.L., 2015. Learning like a child: Fast novel visual concept learning from sentence descriptions of images. In ICCV'15.

Method



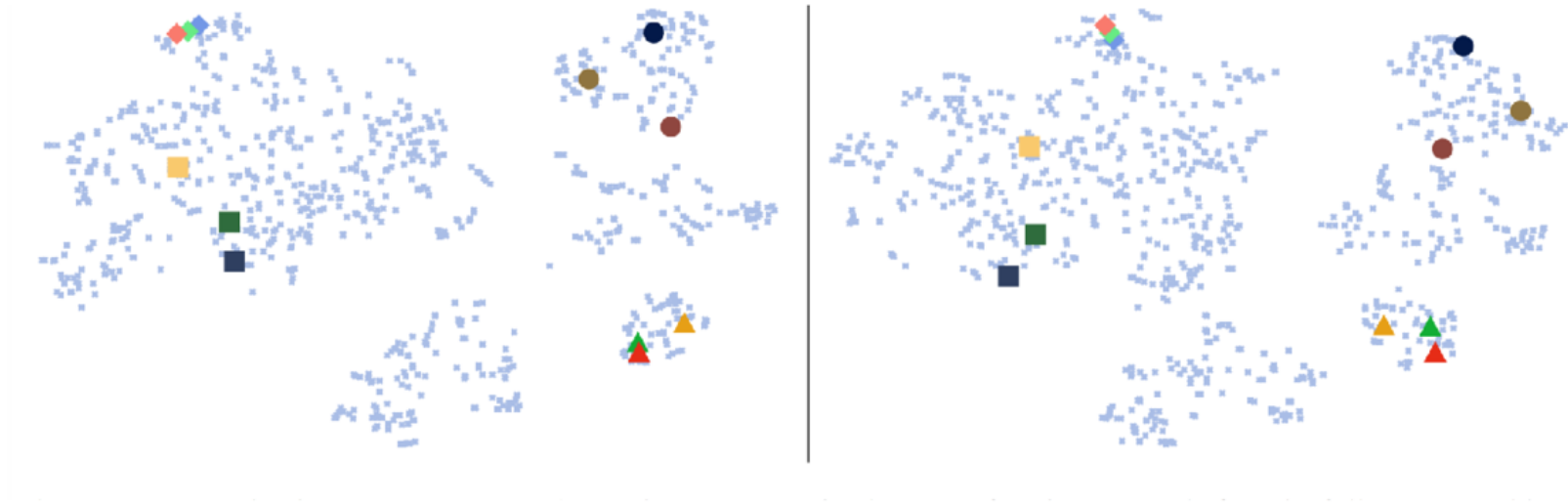
From Wikipedia: T-distributed Stochastic Neighbor Embedding (t-SNE) is a machine learning algorithm for visualization developed by Laurens van der Maaten and Geoffrey Hinton. It is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions.

Find the bridge



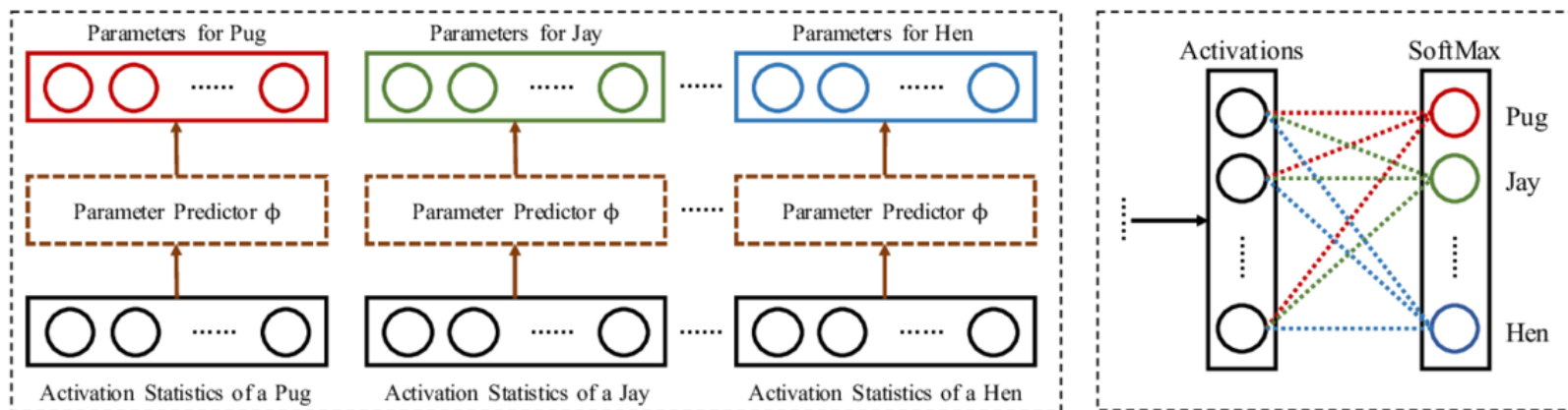
We train a ResNet-50 on ImageNet. We plot the t-SNE of the input mean activations (left) and the parameters (right) of the last fully-connected layer and they look very similar. Circles are mammals, triangles are birds, diamonds are buses, and squares are home appliances.

Find the bridge



This indicates that there is a mapping function from activation to parameter. We learn this mapping by modelling it as a neural network. In other words, we are learning a neural network that predicts the parameters of another neural network.

Learning the Mapping



Training Data

Do we have enough training data for learning the mapping from activation to parameter?

Yes, the large-scale examples and their parameters.

Training Loss

We use the mean activation to predict the parameters, which we then use to classify the activation. The standard classification loss can be used here.

$$\mathcal{L}(\phi) = \sum_{(y,x) \in \mathcal{D}_{\text{large}}} \left[-\phi(\bar{\mathbf{a}}_y) \mathbf{a}(x) + \log \sum_{y' \in \mathcal{C}_{\text{large}}} e^{\phi(\bar{\mathbf{a}}_{y'}) \mathbf{a}(x)} \right] + \lambda \|\phi\| \quad (1)$$

Problem

Eq. 1 models the parameter prediction for categories $y \in \mathcal{C}_{\text{large}}$. However, for the few-shot set \mathcal{C}_{few} , each category only has a few activations, whose mean value is the activation itself when each category has only one sample.

Learning the Mapping

- To model this few-shot setting in the large-scale training on $\mathcal{D}_{\text{large}}$, we allow both the individual activations and the mean activation to represent a category.
- Concretely, let $\mathbf{s}_y \in \mathcal{A}_y \cup \bar{\mathbf{a}}_y$ be a statistic for category y . Let S_{large} denote a statistic set $\{\mathbf{s}_1, \dots, \mathbf{s}_{|\mathcal{C}_{\text{large}}|}\}$ with one for each category in $\mathcal{C}_{\text{large}}$. We sample activations \mathbf{s}_y for each category y from $\mathcal{A}_y \cup \bar{\mathbf{a}}_y$ with a probability p_{mean} to use $\bar{\mathbf{a}}_y$ and $1 - p_{\text{mean}}$ to sample uniformly from \mathcal{A}_y . Now, we learn ϕ to minimize the loss defined by

$$\mathcal{L}(\phi) = \sum_{(y,x) \in \mathcal{D}_{\text{large}}} \mathbb{E}_{S_{\text{large}}} \left[-\phi(\mathbf{s}_y) \mathbf{a}(x) + \log \sum_{y' \in \mathcal{C}_{\text{large}}} e^{\phi(\mathbf{s}_{y'}) \mathbf{a}(x)} \right] + \lambda \|\phi\| \quad (2)$$

Training Strategy



We optimize Eq. 2 by using stochastic gradient decent. Red and solid arrows show the feedforward data flow, while blue and dashed arrow shows the backward gradient flow.

- During inference we include \mathcal{C}_{few} , which calls for a statistic set for all categories $S = \{\mathbf{s}_1, \dots, \mathbf{s}_{|\mathcal{C}|}\}$, where $\mathcal{C} = \mathcal{C}_{\text{large}} \cup \mathcal{C}_{\text{few}}$.
- Each statistic set S can generate a set of parameters $\{\phi(\mathbf{s}_1), \dots, \phi(\mathbf{s}_{|\mathcal{C}|})\}$ that can be used for building a classifier.
- Since we have more than one possible set S from the dataset $\mathcal{D} = \mathcal{D}_{\text{large}} \cup \mathcal{D}_{\text{few}}$, we can do classification based on all the possible S . Formally, we compute the probability of x being in category y by

$$P(y|x) = e^{\mathbb{E}_S[\phi(\mathbf{s}_y)\mathbf{a}(x)]} / \left(\sum_{y' \in \mathcal{C}} e^{\mathbb{E}_S[\phi(\mathbf{s}_{y'})\mathbf{a}(x)]} \right) \quad (3)$$

- However, classifying images with the above equation is time-consuming since it computes the expectations over the entire space of S which is exponentially large.
- In the linear case ϕ is a matrix Φ . The predicted parameter for category y is

$$\hat{\mathbf{w}}_y = \Phi \cdot \mathbf{s}_y \quad (4)$$

The inner product of x before the softmax function for category y is

$$h(\mathbf{s}_y, \mathbf{a}(x)) = \hat{\mathbf{w}}_y \cdot \mathbf{a}(x) = \Phi \cdot \mathbf{s}_y \cdot \mathbf{a}(x) \quad (5)$$

- If $\mathbf{a}(x)$ and \mathbf{s}_y are normalized, then by setting Φ as the identity matrix, $h(\mathbf{s}_y, \mathbf{a}(x))$ is equivalent to the cosine similarity between \mathbf{s}_y and $\mathbf{a}(x)$. Essentially, by learning Φ , we are learning a more general similarity metric on the activations $\mathbf{a}(x)$ by capturing correlations between different dimensions of the activations.

- If we assume ϕ to be a linear mapping, then this expectation can be computed efficiently:

$$\begin{aligned} P(y|x) &= e^{\mathbf{a}(x) \cdot \phi(\mathbb{E}_S[\mathbf{s}_y])} / \left(\sum_{y' \in \mathcal{C}} e^{\mathbf{a}(x) \cdot \phi(\mathbb{E}_S[\mathbf{s}_{y'}])} \right) \\ &= e^{\mathbf{a}(x) \cdot \Phi \cdot \mathbb{E}_S[\mathbf{s}_y]} / \left(\sum_{y' \in \mathcal{C}} e^{\mathbf{a}(x) \cdot \Phi \cdot \mathbb{E}_S[\mathbf{s}_{y'}]} \right) \end{aligned} \tag{6}$$

- Although it is ideal to keep the linearity to reduce the amount of computation, introducing non-linearity could potentially improve the performance. To keep the efficiency, we still push in the expectation and do the approximation.

Experiments

Method	1-Shot	5-Shot
Fine-Tuned Baseline	28.86 \pm 0.54%	49.79 \pm 0.79%
Nearest Neighbor	41.08 \pm 0.70%	51.04 \pm 0.65%
Matching Network	43.56 \pm 0.84%	55.31 \pm 0.73%
Meta-Learner LSTM	43.44 \pm 0.77%	60.60 \pm 0.71%
MAML	48.70 \pm 1.84%	63.11 \pm 0.92%
Ours-Simple	54.53 \pm 0.40%	67.87 \pm 0.20%
Ours-WRN	59.60 \pm 0.41%	73.74 \pm 0.19%

Table 1: 5-way accuracies on MinilmageNet with 95% confidence interval. Red: the best, and blue: the second best.

Methods	Backbone	miniImageNet		tieredImageNet	
		1-shot	5-shot	1-shot	5-shot
Matching Networks [33]	ConvNet-64	43.56±0.84	55.31±0.73	-	-
MAML [6]	ConvNet-32	48.70±1.84	63.11±0.92	51.67±1.81	70.30±1.75
Prototypical Networks‡ [31]	ConvNet-64	49.42±0.78	68.20±0.66	53.31±0.89	72.69±0.74
Relation Net [32]	ConvNet-256	50.44±0.82	65.32±0.70	54.48±0.93	71.32±0.78
SNAIL [19]	ResNet-12	55.71±0.99	68.88±0.92	-	-
LwoF [8]	ConvNet-128	56.20±0.86	73.00±0.64	60.35±0.88*	77.24±0.72*
AdaResNet [20]	ResNet-12	56.88±0.62	71.94±0.57	-	-
TADAM [23]	ResNet-12	58.50±0.30	76.70±0.30	-	-
Activation to Parameter‡ [24]	WRN-28-10	59.60±0.41	73.74±0.19	-	-
TPN [17]	ConvNet-64	55.51±0.86	69.86±0.65	59.91±0.94	73.30±0.75
LEO‡ [29]	WRN-28-10	61.76±0.08	77.59±0.12	66.33±0.05	81.44±0.09
MetaOptNet-SVM [14]	ResNet-12	62.64±0.61	78.63±0.46	65.99±0.72	81.56±0.53
LST [15]	ResNet-12	70.1±1.9	78.7±0.8	77.7±1.6	85.2±0.8
BFSL [7]	WRN-28-10	62.93±0.45	79.87±0.33	-	-
CSPN (ours)	WRN-28-10	61.84±0.79	78.64±0.60	69.20±0.87	84.31±0.65
BD-CSPN (ours)	WRN-28-10	70.31±0.93	81.89±0.60	78.74±0.95	86.92±0.63

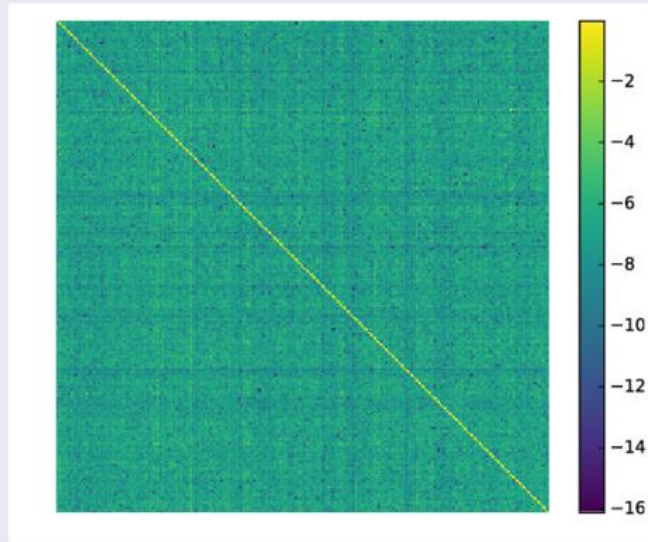
Results from:

Liu, Jinlu, Liang Song, and Yongqiang Qin. "Prototype Rectification for Few-Shot Learning." arXiv preprint arXiv:1911.10713 (2019).

Method	$\mathcal{D}_{\text{large}}$	\mathcal{D}_{few}	FT	Top-1 $\mathcal{C}_{\text{large}}$	Top-5 $\mathcal{C}_{\text{large}}$	Top-1 \mathcal{C}_{few}	Top-5 \mathcal{C}_{few}
NN + Cosine	100%	1	N	71.54%	91.20%	1.72%	5.86%
NN + Cosine	10%	1	N	67.68%	88.90%	4.42%	13.36%
NN + Cosine	1%	1	N	61.11%	85.11%	10.42%	25.88%
Triplet Network	100%	1	N	70.47%	90.61%	1.26%	4.94%
Triplet Network	10%	1	N	66.64%	88.42%	3.48%	11.40%
Triplet Network	1%	1	N	60.09%	84.83%	8.84%	22.24%
Fine-Tuned ResNet	100%	1	Y	76.28%	93.17%	2.82%	13.30%
Learning like a Child	100%	1	Y	76.71%	93.24%	2.90%	17.14%
Ours- ϕ^1	100%	1	N	72.56%	91.12%	19.88%	43.20%
Ours- ϕ^2	100%	1	N	74.17%	91.79%	21.58%	45.82%
Ours- ϕ^{2*}	100%	1	N	75.63%	92.92%	14.32%	33.84%

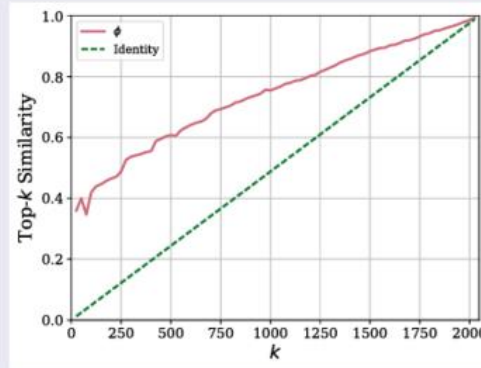
Table 2: Comparing 1000-way accuracies with feature extractor $\mathbf{a}(\cdot)$ pre-trained on $\mathcal{D}_{\text{large}}$. For different \mathcal{D}_{few} settings, red: the best few-shot accuracy, and blue: the second best.

Comparing Activation Impacts



- We visualize the matrix $\phi_{i,j}$ in log scale.
- The values on the diagonal dominates the matrix.
- Along the diagonal, the maximum is 0.976 and the minimum is 0.744. Channels are not used equally.
- Pre-trained activation channels have different distributions of magnitudes and different correlations with the classification task.

Top-K Similarity



- We define the *impact* of its j -th channel on mapping ϕ by $I_j(\phi) = \sum_i |\phi_{ij}|$.
- We compare the channel importance between ϕ and W of the last fully-connected layer. W is trained on large-scale datasets.
- We show this by comparing the orders of the channels sorted by their impacts. Let $\text{top-}k(S)$ find the indexes of the top- k elements of S .

$$\text{OS}(\phi, W, k) = \text{card}(\text{top-}k(I(\phi)) \cap \text{top-}k(I(W))) / k \quad (7)$$

where **card** is the cardinality of the set.

Multi-View

Improving results by using multiple views of the images.

Method	Multi-View	1-Shot	5-Shot
Ours-Simple		$54.53 \pm 0.40\%$	$67.87 \pm 0.20\%$
Ours-WRN		$59.60 \pm 0.41\%$	$73.74 \pm 0.19\%$
Ours-Simple	✓	$56.01 \pm 0.59\%$	$70.70 \pm 0.44\%$
Ours-WRN	✓	$63.62 \pm 0.58\%$	$78.83 \pm 0.36\%$

Indv \Rightarrow Mean \Rightarrow Param

Use another network to first map the individual activation to the mean activation, then train another network to convert the mean activation to the parameters.

Thanks