Multi-Proxy Wasserstein Classifier for Image Classification

Benlin Liu[1]*, Yongming Rao[2]*, Jiwen Lu[2], Jie Zhou[2], Cho-jui Hsie[1]
[1] UCLA
[2] Tsinghua University
{liubenlin, chohsieh}@cs.ucla.edu, raoyongming95@gmail.com,
{lujiwen, jzhou}@tsinghua.edu.cn

Abstract

Most widely-used convolutional neural networks (CNNs) end up with a global average pooling layer and a fully-connected layer. In this pipeline, a certain class is represented by one template vector preserved in the feature banks of fully-connected layer. Yet, a class may have multiple properties useful for recognition while the above formulation only captures one of them. Therefore, it is desired to represent a class by multiple proxies. However, directly adding multiple linear layers turns out to be a trivial solution as no improvement can be observed. To tackle this problem, we adopt optimal transport theory to calculate a non-uniform matching flow between the elements in the feature map of a sample and the proxies of a class in a closed way. By doing so, the models are enabled to achieve partial matching as both the feature maps and the proxy set can now focus on a subset of elements from the counterpart. Such formulation also enables us to embed the samples into the Wasserstein metric space, which has many advantages over the original Euclidean space. This formulation can be achieved by a lightweight iterative algorithm, which can be easily embedded into the automatic differentiation framework. Empirical studies are performed on two widely-used classification datasets, CIFAR, and ILSVRC2012, and the substantial improvements on these two benchmarks demonstrate the effectiveness of our method.

Introduction

Recent years have witnessed the great progress brought by deep neural networks, especially convolutional neural networks (CNNs), in various vision tasks, ranging from recognition (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016), object detection (He et al. 2017) and segmentation (Chen et al. 2017). Such strong power of neural network is mainly attributed to its great capability in extracting feature map directly from an input sample X, which contains the high-level information that is useful for downstream tasks. For recognition tasks, the extracted feature map is then usually aggregated across spatial domain through a pooling operation to get a feature vector of $\mathbb{R}^d$ to represent the input sample. Then we can calculate the inner product of this feature vector with the template vector of a certain class C preserved in the fully-connected layer as the similarity score for this $\langle sample, class \rangle$ pair.

In the above framework, each class is represented by a single template vector of $\mathbb{R}^d$ in the fully-connected layer, which is expected to capture the most discriminative pat-

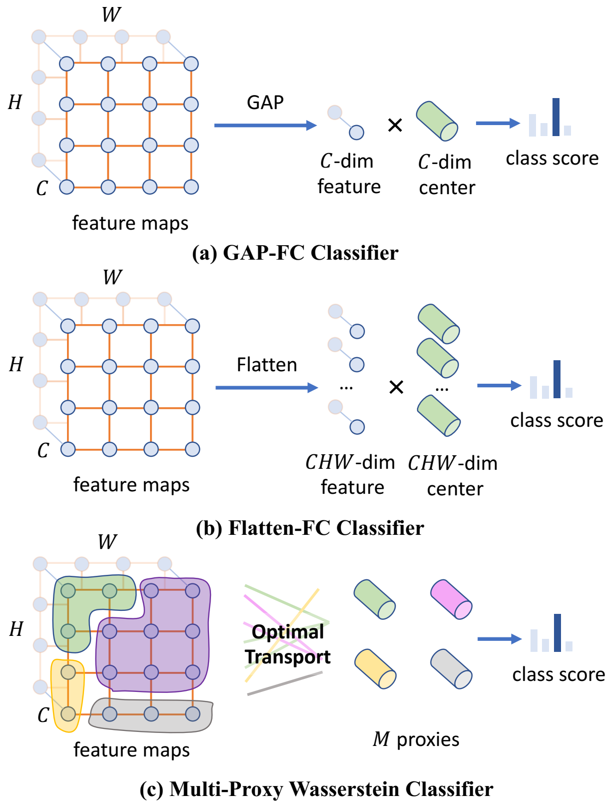

Figure 1: Comparison between the conventional (a) GAP-FC classifier, (b) Flatten-FC classifier, and (c) the proposed multi-proxy Wasserstein classifier. Our method utilizes matching flow between the feature maps of a sample and the template feature vectors of a class, and thus embeds the samples into the Wasserstein metric space and facilitates CNNs to learn multiple diverse clues for classification.

*Equal contribution.

tern of that class. However, a certain class often has multiple intrinsic properties that can be exploited to facilitate classification. In particular, we can determine the category of an object based on its shape, size, viewpoint, texture and even background. Such idea is similar to the disentangled representation learning used in unsupervised learning that breaks down, or disentangles, each feature into narrowly defined variables and encodes them as separate dimensions, while our objective here is to disentangle the template vectors in the fully-connected layers. And this modification would contribute to the performance even more when there exists more target class as the inter-class variation grows smaller in this case.

Given capturing multiple patterns can contribute to the classification, a trivial improvement is to add several more fully-connected layers. Thus, we now have multiple similarity scores for a $\langle sample, class \rangle$ pair. We can then average these scores to obtain the final similarity score for this $\langle sample, class \rangle$ pair. However, such formulation does not bring any improvement compared with the original formulation. We argue this is because a given sample may not have all patterns captured by different fully-connected layers. Instead, it may only have a subset of the captured patterns. Besides, as more patterns are captured, the pattern would become more fine-grained, so for a certain pattern, it is likely to exist in a subarea of the input sample/extracted feature map. However, the trivial solution treats each subarea in the extracted feature map and each captured pattern of a certain class in an equal way, and thus fails to tackle the above-mentioned two problems.

To overcome the above-mentioned difficulties, we propose a model called multi-proxy Wasserstein net in which we skip the traditional pooling operation and represent a sample X as $H \times W$ elements[1]. Besides, we represent each target class as M proxies to capture different patterns. In this work, an element and a proxy both refer to a vector of $\mathbb{R}^d$, while the former represents a subarea in an input sample and the latter represents a certain pattern of a class. Therefore, we now can calculate a similarity score between an element from a sample X and a proxy from a class C and we shall get $HW \times M$ similarity scores in total for $\langle X, C \rangle$ pair. In contrast to the trivial solution where we directly exploit a uniform matching flow to aggregate all these $HW \times M$ similarity scores as the final similarity score for the sample-class pair, we adopt optimal transport theory to compute a non-uniform semantic matching flow to aggregate all these similarity scores. The matching flow calculated by optimal transport assigns higher weights to the pair of elements that are close to each other. This property enables each subarea in the extracted feature map to pay more attention to a subset proxies that it has stronger correlation with, and each proxy can also focus on those closely-related subareas. As a result, our feature-proxy matching operation will give a high final similarity score only if there exists a subarea closely relates to a captured pattern, and thus allows partial matching to some extent, which helps to solve the problems described in the previous paragraph. The basic idea of our method is illustrated in Figure 1.

Moreover, by employing optimal transport to calculate the similarity score between a sample and a target class is equivalent to embed the samples and the class into the Wasserstein space whose representational capacity is larger than Euclidean space (Deza and Laurent 2009), and the relationship between data can therefore be better represented in this embedding space with low distortion as the geometry of the space would be taken into consideration (Kloeckner 2010). Therefore, we name our method as a multi-proxy Wasserstein classifier.

Compared to those elaborately designed attention mechanisms, no additional parameters are involved in the calculation of matching flow. However, exactly solving the optimal transport problem requires the complexity of $\mathcal{O}(n^3)$ where n is proportional to the number of elements. To reduce the computational cost, we resort to an entropically-regularized optimal transport problem, which can then be solved by a fast iterative algorithm (Cuturi 2013), and the time complexity will thus be lowered down to $\mathcal{O}(HW \times M)$. This iterative algorithm can be easily embedded into the back propagation pipeline and then optimized using off-the-shelf modern automatic differentiation library.

To show the effectiveness of our approach, we conduct experiments on two widely-used classification benchmarks, CIFAR dataset and ILSVRC2012 dataset. To verify the generalizability of our method, experiments are performed across various network architectures. The substantial improvements over baseline for different architectures and datasets strongly demonstrate the benefit of proposed method.

Related Work

Optimal Transport Optimal transport theory was firstly studied by Monge (Monge 1781) dating back to 1781. It aims at providing the correspondence between two distributions or two sets in a closed way, based on which we can then calculate the Wasserstein distance between this two distributions/sets. This theory also plays an important role in machine learning community. WGAN (Arjovsky, Chintala, and Bottou 2017) applies optimal transport to generative model to align the distribution of generated data with the true natural distribution. In addition, optimal transport has also been applied to domain adaptation (Courty et al. 2017), 3D shape understanding (Solomon et al. 2014), graph matching (Xu et al. 2019), style transfer (Kolkin, Salavon, and Shakhnarovich 2019) or measure synchronization (Birdal et al. 2020)

One big problem of optimal transport is that it is costly to compute, requiring the solution of a linear program. Cuturi (Cuturi 2013) proposes a light-speed iterative method to solve the entropically-regularized optimal transport problem, and it can be directly plugged into the back-propagation pipeline. In this work, we also employ this algorithm to calculate the matching flow between the elements in the extracted feature map of an input sample and the multiple proxies of a class.

[1]Here $H \times W$ represents the size of the feature maps from the last stage of convolution neural network instead of the original image size (e.g., 7×7 for a 224×224 input image).

Image Classification Deep learning has demonstrated great sauces in image classification tasks (He et al. 2016; Lin, RoyChowdhury, and Maji 2015; Hu et al. 2019; Rao et al. 2018; Liu et al. 2020). Many methods for image classification are in fact equivalent to using multiple proxies compared to the original setting. The dimension of the template vectors in fully-connected layer following bilinear pooling operation (Lin, RoyChowdhury, and Maji 2015) is square of the original average pooling setting, which means it uses as many proxies as the dimensions of output feature map and uses output feature map as matching flow for following aggregation. WS-DAN (Hu et al. 2019) improves upon bilinear pooling by reducing the number of proxies and uses an attention module to learn the matching flow. However, these methods mainly works for fine-grained classification but not for general classification tasks. We show the matching flow obtained by these method leads to inferior performance in the experiments while they often use much more proxies compared to our methods.

Attention-based Pooling Some previous works have explored the use of attention mechanism for image classification. Girdhar et al. (Girdhar and Ramanan 2017) proposes to aggregate spatial logits based on the importance weight given by a jointly-trained spatial attention module but only work for action recognition tasks. Attention Branch Network (ABN) (Fukui et al. 2019) designed a separate attention branch to learn a more elaborate attention module based on CAM for image classification. However, such attention module involves much more parameters and computational cost to learn the importance weight while our method requires no further parameters in the calculation of matching flow and need much less FLOPs. Further, all these methods only use one proxy to represent a class, while in this work we try to explore using multiple proxies to capture diverse patterns of a class from different angles.

Method

In this section, we detail the proposed Multi-Proxy Wasserstein classifier and its instantiations for image classification. We first introduce preliminary knowledge about optimal transport theory and Wasserstein distance. Then, we present the multi-proxy classifier for visual classification. At last, we describe how to model the feature-proxy match as an optimal transport problem and optimize the classifier in an end-to-end manner.

Preliminaries

We begin by briefly reviewing the preliminary knowledge about optimal transport theory and Wasserstein distance, where our method is built upon.

Optimal transport theory aims to seek the **minimal cost transport plan** between two distribution. Considering a source distribution μ_s and a target distribution μ_t that are defined on probability space $\mathcal{X}$ and $\mathcal{Y}$ respectively, the Wasserstein distance $\mathcal{W}_p$ between the two distributions is

$$\mathcal{W}_p(\mu_s, \mu_t) = \inf_{\pi \in \Pi(\mu_s, \mu_t)} \int_{\mathcal{X}\times\mathcal{Y}} c(x,y)^p d\pi(x,y), \qquad (1)$$

where the infimum is taken over all possible transport plans π (i.e., $\Pi(\mu_s, \mu_t)$ is the joint probability distribution with marginals μ_s and μ_t), $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ is the cost function of transportation. The Wasserstein distance is the cost of the optimal transport plan matching μ_s and μ_t (Villani 2008).

In this paper, we focus on discrete distributions supported on finite sets of points in $\mathbb{R}^d$, which can be written as:

$$\mu_s = \sum_i^{n_s} p_i^s \delta(x_i), \quad \mu_t = \sum_i^{n_t} p_i^t \delta(y_i), \qquad (2)$$

where p^s and p^t are the probability mass summing to 1, n_s and n_t are the number of samples, $\delta(\cdot)$ is the Dirac function, $\{x_i\}$ and $\{y_i\}$ are the support points. The transport plan π matching the two distributions also becomes discrete. In this case, we can further define cost matrix $M \in \mathbb{R}_+^{n_s \times n_t}$ with $M_{i,j} = c(x_i, y_j)^p$. Then, the above optimal transport problem is equivalent to:

$$\begin{aligned} \mathcal{W}_p(\mu_s, \mu_t) &= \min_{T \geq 0} \operatorname{tr}(MT^\top), \\ \text{subject to} \quad T\mathbf{1} &= \mu_s, \quad T^\top \mathbf{1} = \mu_t. \end{aligned} \qquad (3)$$

The corresponding optimal transport T^* is called the optimal transport plan, representing the optimal matching flow between these two distributions. $T^*_{i,j}$ is the amount of mass that need to move from x_i to y_j in order to reach an overall minimum cost.

Multi-Proxy Image Classification

Most recent convolution neural networks for image classification use a global average pooling (GAP) followed by a fully-connected (FC) layer to summarize spatial features and produce output logits following the practice of Network-in-Network (Lin, Chen, and Yan 2013) and ResNet (He et al. 2016). The GAP-FC scheme significantly reduces the redundant parameters in classifier compared to previous networks based on Flatten-FC strategy (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014). However, we argue that such scheme still has a drawback. In particular, a certain class often has multiple intrinsic properties that can be exploited to facilitate classification and the spatial distribution of visual patterns usually can also contribute to the classification. Therefore, we propose a new classification method based on multiple proxies instead of a single class center to model the multiple intrinsic properties and spatial clues for image classification.

Let $\mathbf{X} \in \mathbb{R}^{d \times HW}$ denote the output of CNN and y is the corresponding one-hot encoded classification label, where d, H and W are the number of channels, height, width of the last feature maps. The conventional GAP-FC scheme can be formulate as:

$$\hat{y}_{\text{GAP-FC}} = \text{softmax}\left(\mathbf{X}_{\text{GAP}}^T \mathbf{W}\right), \qquad (4)$$

where $\mathbf{X}_{\text{GAP}}^T \in \mathbb{R}^d$ is the aggregated feature after global average pooling, $\mathbf{W} \in \mathbb{R}^{d \times K}$ is the fully-connected classifier for K categories, and $\hat{y}_{\text{GAP-FC}}$ is the output probability distribution over these K categories.

Instead of modeling each class as a single d-dimension vector, our method constructs M proxies for each class and aggregates the spatial features according to the correlation between features and proxies. Specifically, given the set of proxies $\mathbf{P}_i \in \mathbb{R}^{d\times M}$ for a class i and the CNN feature maps $\mathbf{X}$, the classification score can be computed by:

$$s_i = \sum_{j=1}^{HW} \sum_{m=1}^{M} w_{i,j,m} \mathbf{X}_j^T \mathbf{P}_{i,m}, \tag{5}$$

and the normalized scores are the predicted probability distribution over k categories for the proposed multi-proxy (MP) classifier:

$$\hat{y}_{\text{MP}} = \text{softmax}\left([s_1, s_2, ..., s_k]\right), \tag{6}$$

where $\mathbf{X}_j$ is the feature at j-th location, $\mathbf{P}_{i,m}$ is the m-th proxy for class i, $w_{i,j,m}$ reflects the correlation between the feature and the proxy.

Besides the trivial solution we discussed at the beginning that adopts a uniformly-distributed $w_{i,j,m}$, a slightly more sophisticated approach is to obtain the correlation based on matching the features and proxies individually using cosine similarity, which can be computed as:

$$c_{i,j,m} = \frac{\mathbf{X}_j^T \mathbf{P}_{i,m}}{\|\mathbf{X}_j\| \|\mathbf{P}_{i,m}\|}. \tag{7}$$

However, this approach ignores the relationship among different features and proxies. Without considering any mutual relation, many local features may be assigned to the same proxy and proxies for the same category usually collapse to a similar value. This mode collapse phenomena make the multi-proxy classifier failing to capture the diverse intrinsic properties of the category. Our experimental study also shows this trivial approach will not significantly improve the classification performance (see our ablation study).

Feature-Proxy Matching via Optimal Transport

To overcome the above mentioned issues, we propose to model the feature-proxy matching as a optimal transport problem. Specifically, we use the matching flow obtained using optimal transport to reweigh the inner product of each feature-proxy pair. In our case, the source distribution is the discrete distribution of local features and the target distribution is the proxy distribution for each class. As suggested by (Frogner, Mirzazadeh, and Solomon 2019), we set both source and target distribution as uniform distribution for the sake of simplicity. We also observed that adaptively learned distribution will not boost performance but lead to unstable training. We set the cost matrix $M_i = 1 - C_i$ for each class in practice, where C_i is the cosine similarity matrix computed using Eq. 7. Therefore, we can obtain the optimal transport T_i^* by optimizing the global matching flow:

$$\begin{aligned} T_i^* &= \operatorname*{arg\,min}_{T_i \geq 0} \text{tr}({M_i}^p T_i^\top), \\ \text{subject to} \quad & T_i \mathbf{1} = \mu_s, \quad T_i^\top \mathbf{1} = \mu_t. \end{aligned} \tag{8}$$

Based on the optimal matching flow, we can obtain the classification score of multi-proxy classifier by:

$$s_i = \sum_{j=1}^{HW} \sum_{m=1}^{M} T^*_{i,j,m} \mathbf{X}_j^T \mathbf{P}_{i,m}. \tag{9}$$

Algorithm 1 Learning Multi-Proxy Wasserstein Classifier

```
 1: Input: An input sample $\mathcal{X}$; a proxy set $\mathbf{P}_i$ for a single
    class which contains $M$ proxies; a convolutional feature
    extractor $\mathcal{F}$;
 2: Output: A set of similarity scores between the input
    sample $\mathcal{X}$ and all $K$ target classes
 3: Use feature extractor $\mathcal{F}$ to extract a feature map $\mathbf{X}$ of
    size $\mathbb{R}^{d\times HW}$ from the sample $\mathcal{X}$
 4: for $i = 1, 2, \dots, K$ do
 5:     Use Eq (7) to calculate the similarity matrix $C_i$
 6:     $M_i \leftarrow 1 - C_i$
 7:     $K_i \leftarrow e^{\frac{-M_i^p}{\lambda}}$
 8:     $\beta \leftarrow \mathbf{1}, t \leftarrow 0$
 9:     while $t < t_{MAX}$ do
10:         $\alpha \leftarrow \mu_s / K\beta$
11:         $\beta \leftarrow \mu_t / K^\top \alpha$
12:         if $\overline{\Delta_\alpha} <$ thereshold then
13:             break
14:         end if
15:         $t \leftarrow t + 1$
16:     end while
17:     $T_i \leftarrow \text{diag}(\alpha) K \text{diag}(\beta)$
18:     Use Eq (9) to calculate $s_i$
19: end for
20: return $\{s1, s2, \dots, s_K\}$
```

Intuitively, the proposed optimal transport method adaptively assigns the local features to the target proxies. With the global cost planning, our method encourage the classifier to learn diverse proxy patterns. As an extension of the GAP-FC scheme, our method models the spatial distribution of visual features, which provides extra clues for more accurate classification. Different from the Flatten-FC strategy, our method is transition-invariant, which ensures more robust classification.

End-to-End Proxy Learning

Since exactly solving the optimal transport problem requires the complexity of $\mathcal{O}(n^3)$. To enable efficient training and inference, we adopt the Sinkhorn divergence method proposed in (Cuturi 2013), where an entropic regularizer is added to the original optimal transport problem, which lower down the complexity to $\mathcal{O}(HW \times M)$. Thanks to the efficient algorithm, our method only introduce a slight additional computational cost compared to the CNN backbone while clearly improving the classification performance. The modified objective becomes:

$$\begin{aligned} \mathcal{W}_p(\mu_s, \mu_t) &= \min_{T \geq 0} \text{tr}(M^p T^\top) + \lambda \text{tr}\left(T(\log(T) - \mathbf{1}\mathbf{1}^\top)^\top\right), \\ \text{subject to} \quad & T\mathbf{1} = \mu_s, \quad T^\top \mathbf{1} = \mu_t, \end{aligned} \tag{10}$$

where λ is a non-negative regularization parameter. Eq. 10 is a convex problem, which can be solved using Sinkhorn-Knopp algorithm (Sinkhorn 1967). With $K = \exp(\frac{-M^p}{\lambda})$, the problem can be solved by alternately projecting onto the marginal constraints:

$$\alpha \leftarrow \mu_s / K\beta, \quad \beta \leftarrow \mu_t / K^{\top}\alpha. \tag{11}$$

After converge, the optimal transport can be computed by:

$$T^{*} = \mathrm{diag}(\alpha) K \mathrm{diag}(\beta). \tag{12}$$

Note that the above iterative algorithm is fully differentiable, which makes our method easy to implement by using automatic differentiation library like Pytorch (Paszke et al. 2019) and train the proposed multi-proxy classifier in an end-to-end manner. The algorithm of learning multi-proxy Wasserstein classifier is summarized in Algorithm 1.

Experiments

To demonstrate the effectiveness of our method, we conduct experiments on two widely-used benchmarks for image classification task, namely CIFAR10/100 (Krizhevsky, Hinton et al. 2009), and ILSVRC2012 (Russakovsky et al. 2015).

Datasets

CIFAR CIFAR-10 (Krizhevsky, Hinton et al. 2009) dataset contains 60,000 low resolution RGB images of size 32×32 from 10 different classes. Each class consists of exactly 6,000 images. All these images are split into a training set of 50,000 images and 10,000 test images. CIFAR-100 uses the same training set and test set, but further divides the target categories into 100 different classes. We also use CIFAR-10.1 dataset (Recht et al. 2018) to identify the robustness of our model against unknown distribution shift during test phase. CIFAR-10.1 contains 2,000 test images in total, with exactly the same classes and creation process as original CIFAR-10 dataset. The objective is to investigate the effect of the subtle unknown distribution shifts between two test sets on recognition task. The traditional domain adaptation methods cannot handle this task well as the distribution shifts are just too subtle and the sample size is too small for most domain adaptation methods. We show our method can bring substantial improvement on this difficult dataset.

ILSVRC2012 ILSVRC2012 (Russakovsky et al. 2015) is a large-scale dataset used to evaluate the capability of a deep neural network. The training set contains around 1.2 million images and the test set contains 5K images, all of which are high-resolution images from 1,000 different categories. The distribution over target classes roughly uniform for training set and strictly uniform for test set.

Results on CIFAR

We first carry out the performance evaluation on CIFAR dataset. To show the generalizability of our method, we test on several different network architectures, including two types of ResNet (He et al. 2016) with different depth (ResNet-18 and ResNet-101), two types Wide-ResNet (Zagoruyko and Komodakis 2016) with different depth and width (Wide-ResNet-40-2 and Wide-ResNet-28-10), DenseNet-100 (Huang et al. 2017) (growth rate $k = 12$) and PyramidNet200 (Han, Kim, and Kim 2017) augmented with CutMix (Yun et al. 2019) (widening factor $\bar{\alpha} = 240$). Among above models, the spatial size of the feature maps from the last convolution layer are all 8×8 except ResNet-18 which is 4×4. We use ResNet-18 to show that our method can work with output feature maps of different sizes. All models are trained by 200 epochs in total with initial learning rate as 0.1. We adjust the learning rate in the training process by following the strategies described in the original works. We use stochastic gradient descent (SGD) as our optimizer, and we set momentum as 0.9 and weight decay as 0.0005. For the hyper-parameter in Algorithm. 1, we set $M = 4$ for all different network architectures. In our experiments, we find $\mathcal{W}_1$ works best for most architectures, so we set $p = 1$ for all cases. And we set the weight of entropically regularization term λ as 0.1. Besides, we set threshold as 0.1 and maximum iteration number as 100. All experiments are conducted with NVIDIA 1080 Ti GPUs using PyTorch 1.4.0 on Python 3.7.4 platform. We report the median accuracy after running each experiment 5 times. We re-run the baseline methods for both CIFAR-10 and CIFAR-100 and also report the results from original papers.

As shown in Table. 1 and Table. 2 , we find our method improves upon the corresponding baseline for all different architectures. Though the improvement is relative marginal on CIFAR-10, especially for Wide-ResNet-28-10 which only acquires 0.14% boost, we can see the improvement of accuracy on CIFAR-10.1 is much higher than CIFAR-10 except DenseNet-100. For Wide-ResNet-40-2, the accuracy on CIFAR-10.1 is even 1.6% higher than the corresponding baseline. This indicates our method can make the model more robust to such subtle unknown distribution shift. Considering both test sets are sampled from natural image distribution, our method shows much stronger ability to generalize to more unseen natural image data. Since the major objective of machine learning is to generalize to more unseen data, we believe such robustness to natural variations should be acknowledged as one great advantage of our approach.

On CIFAR-100, our method outperforms its counterparts consistently. Noticeably, the improvement on CIFAR-100 is much more obvious compared to CIFAR-10. For instance, we improve by 2.2% on CIFAR-100 for ResNet-110 while the improvement on CIFAR-10 is only 0.47%. This illustrates that our method can bring more boost when the number of target classes is more. This is due to the inter-class variation of CIFAR-100 is smaller than CIFAR-10, so learning more discriminative pattern can benefit classifier more.

Results on ILSVRC2012

Besides CIFAR dataset, we also perform experiments on ILSVRC2012 dataset to validate the effectiveness our method on large-scale dataset. We test our method with ResNet18, ResNet50 and ResNet101. The size of output feature map becomes 7×7 in this case. We run the experiments for 90 epochs in total, and multiply learning rate by 0.1 at epoch 30 and epoch 60. We set the batch size as 256 and the input size as 224×224. Other setting are same with CIFAR experiments. We show the results in Table. 3. The result clearly shows that our method can brings improvement to different network architectures consistently. For exam-

Model	# Params		FLOPs		CIFAR-10		CIFAR-10.1	
	Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours
ResNet-18	0.27M	0.27M	41.3M	41.4M	92.26	**92.85**$_{(+0.59)}$	83.55	**84.45**$_{(+0.90)}$
ResNet-110	1.73M	1.73M	255M	255M	94.03$_{93.57\star}$	**94.50**$_{(+0.47)}$	86.15	**87.05**$_{(+0.90)}$
WRN40-2	2.2M	2.2M	323M	323M	95.09$_{94.80\star}$	**95.13**$_{(+0.04)}$	87.60	**89.20**$_{(+1.60)}$
WRN28-10	36.4M	36.4M	5.25G	5.25G	95.95$_{95.83\star}$	**96.09**$_{(+0.26)}$	89.80	**91.15**$_{(+1.35)}$
DenseNet100	4.07M	4.07M	1.35G	1.35G	94.75$_{94.23\star}$	**95.08**$_{(+0.33)}$	**88.20**	87.90$_{(-0.30)}$
PyramidNet200	26.7M	26.7M	4.56G	4.56G	96.83$_{97.12\star}$	**97.32**$_{(+0.20)}$	88.15	**88.60**$_{(+0.45)}$

Table 1: The performance of different architectures on CIFAR-10 and CIFAR-10.1 benchmark. We re-implement all baseline methods for fair comparison. All the results available in the original paper denoted are listed in the table and denoted by *.

Model	# Params		FLOPs		Accuracy	
	Baseline	Ours	Baseline	Ours	Baseline	Ours
ResNet-18	0.28M	0.29M	41.3M	41.4M	69.95	**71.60**$_{(+1.65)}$
ResNet-110	1.73M	1.73M	255M	255M	73.25	**75.27**$_{(+2.02)}$
WRN40-2	2.2M	2.2M	323M	323M	76.22$_{74.73\star}$	**77.12**$_{(+2.39)}$
WRN28-10	36.5M	36.6M	5.25G	5.25G	80.14$_{80.13\star}$	**80.71**$_{(+0.57)}$
DenseNet100	4.12M	4.27M	1.35G	1.35G	76.82$_{76.21\star}$	**77.57**$_{(+0.65)}$
PyramidNet200	26.8M	27.0M	4.56G	4.56G	84.96$_{85.53\star}$	**85.28**$_{(+0.32)}$

Table 2: The performance of different architectures on CIFAR-100 benchmark. We re-implement all baseline methods for fair comparison. All the results available in the original paper denoted are listed in the table and denoted by *.

Model	# Params		FLOPs		Top-1		Top-5	
	Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours
ResNet-18	11.7M	13.3M	1.82G	1.87G	69.76	**71.08**$_{(+1.32)}$	89.06	**89.95**$_{(+0.89)}$
ResNet-50	25.6M	31.7M	3.88G	4.08G	76.15	**76.96**$_{(+0.81)}$	92.87	**93.33**$_{(+0.46)}$
ResNet-101	44.5M	50.6M	7.60G	8.02G	77.37	**77.94**$_{(+0.57)}$	93.56	**93.87**$_{(+0.31)}$

Table 3: The performance of different architectures on ILSVRC2012. We report Top-1 / Top-5 accuracies $(\%)$ for different architectures.

ple, it improves Top-1 accuracy on ResNet18 by 1.32%. On ResNet50 and ResNet101, it still improves Top-1 accuracy by 0.81% and 0.57% respectively. We can see our method can help small network even more than larger ones.

Ablation Study

Necessity of Optimal Transport We conduct experiments to show that it is necessary to exploit the optimal transport theory to make the multi-proxy classifier work. In particular, we first test the trivial solution that simply adds multiple fully-connected layers and take the average of outputs from all branches (Baseline + Multi-FC$_{Average}$). This is equivalent to using a uniform matching flow to aggregate $HW \times M$ similarity scores for the final score of a $\langle sample, class\rangle$ pair. We conduct experiments using ResNet18 and WideResNet40-2 on CIFAR-100.We report the results in Table. 4. We can see the trivial solution brings absolutely no change to the original baseline. This empirically shows that the trivial solution would degrade to the original setting.

More deeply, what makes our formulation works is that we embed the sample and the class as discrete distributions in the Wasserstein space. A simple modification to the trivial solution to overcome the degrade problem could be replacing the average operation by flatten operation (Baseline

	ResNet18	WRN40-2
Baseline	69.95	76.22
Baseline + Multi-FC$_{\text{Average}}$	69.94	76.23
Baseline + Multi-FC$_{\text{Flatten}}$	69.32	75.07
Baseline + Multi-FC$_{\text{Cosine}}$	70.06	76.42
Baseline + Bilinear	69.02	74.00
Baseline + WS-DAN	69.40	74.52
Ours	**71.60**	**77.12**

Table 4: We compare our method with some other methods that can also calculate the matching flow to illustrate the necessity of using optimal tranport theory. All the experiments are conducted on CIFAR-100. The details please refer to the ablation study section.

	Training	Test
ResNet-110	38.71s	2.54s
ResNet-110 + Ours	45.58s	2.93s

Table 5: We measure both training time and test time for ResNet-110 on CIFAR-100. The training time refers the average time of running 1 epoch, and test time is the time needed to evaluate the whole test set.

+ Multi-FC$_{Flatten}$), like what has been done in the ending layers of AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and VGG (Simonyan and Zisserman 2014). Yet, such formulation has been proved to be inferior to the original GAP-FC formulation by our experiments in Table 4. This is because the flatten operation makes the order of elements or proxies matters, while the discrete distribution is in fact an unordered set. However, we do not care where a pattern would appear on the input image sample in the classification, so flatten can not improve the trivial solution. Furthermore, flatten would require $H \times W$ proxies while our formulation needs much less than that.

Another slightly more sophisticated solution is to introduce non-uniformity by using the cosine similarity between an element and a proxy as the weight in the matching flow. However, we show in Table. 4 that the improvement brought by this modification (Baseline + Multi-FC$_{Cosine}$) is smaller compared to our method. As discussed above, directly using cosine similarity defined in the Euclidean space can not capture the geometry of the space like the embeddings in the wasserstain space. This again demonstrate the necessity of exploiting optimal transport for calculating matching flow.

Comparison with Fine-Grained Methods We also compared with methods used for fine-grained classification on general classification tasks as they are equivalent to using multiple proxies for classification. Different from above non-parametric solution, the matching flow given by fine-grained classification methods often requires additional parameters. We compare with bilinear pooling and WS-DAN (only considering the attention module but not augmentation step) with our method on CIFAR-100. The results are shown Table 4. We can find these methods even lead to decrease of accuracy on the general-purpose classification task. On the contrary, our method achieves around 2% lower on fine-grained classification task benchmark like CUB-2011 (Wah et al. 2011). This reveals the fine-grained methods and our method are good at dealing with different recognition tasks. We conjecture this is because the size of the output feature map in fine-grained tasks (26 × 26 on CUB-2011) is much larger than that in general classification tasks (8 × 8 / 4 × 4 for CIFAR, 7 × 7 for ILSVRC2012). Larger size of output feature map enlarges the variation among elements in output feature map, and this may make it easier for the attention module to learn the relationship between elements and proxies.

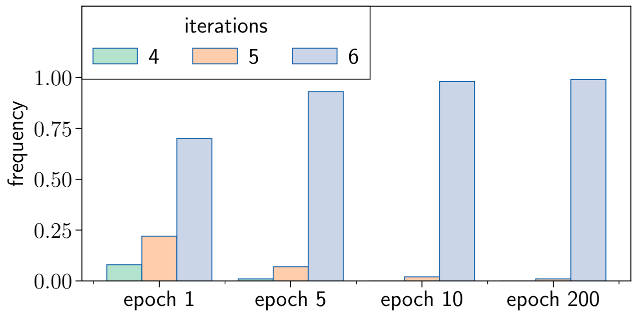

Figure 2: The number of iteration needed to calculate the correspondence flow gradually converges to 6 with the training goes on.

Time Analysis

One major concern of optimal transport is always its efficiency. We first conduct experiments to investigate the average iterations needed for the sinkhorn algorithm to be converged. We take the average across all batches, all epochs and all networks on CIFAR dataset. The average number of iterations is just 5.93, and we can find the variance is quite small from Figure 2. This indicates the additional computational cost is relatively small. More concretely, we report the training time (1 epoch) as well as the test time (on whole test set) in Table. 5. We can see that the training time increases by 17.7%, and 15.4%, which is acceptable given the improvement on the performance. The additional time cost is mainly attributed to the `for` iteration used in the iterative algorithm. In fact, considering the original ResNet110 requires 1.7GFLOPs in total, the increase of the number of FLOPs caused by our method is negligible, as we only use 4 proxies for each class in our experiments. Compared to ABN (Fukui et al. 2019) which require 5.7GFLOPs, our method can achieve comparable improvement with much less FLOPs.

Visualization

To have an intuitive understanding of the proposed multi-proxy Wasserstein classifier, we visualize the correspondence between local features and proxies. The results can be found in *Supplementary Material*.

Conclusion

In this work, we propose a new method called multi-proxy wasserstein network, where we represent each target class as multiple proxies instead of one to capture diverse useful patterns in the input sample from different angles. To make the multiple proxy can truly help the recognition, we propose to apply optimal transport theory to calculate a matching flow for each class to aggregate the similarity scores between each element in the feature map and each proxy. A fast iterative algorithm is exploited to achieve this. Through extensive experiments, We further validate the advantage of our method as well as the necessity of using optimal transport in this problem.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* .

Birdal, T.; Arbel, M.; Simsekli, U.; and Guibas, L. J. 2020. Synchronizing Probability Measures on Rotations via Optimal Transport. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1569–1579.

Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40(4): 834–848.

Courty, N.; Flamary, R.; Habrard, A.; and Rakotomamonjy, A. 2017. Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information Processing Systems*, 3730–3739.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, 2292–2300.

Deza, M. M.; and Laurent, M. 2009. *Geometry of cuts and metrics*, volume 15. Springer.

Frogner, C.; Mirzazadeh, F.; and Solomon, J. 2019. Learning Embeddings into Entropic Wasserstein Spaces. *arXiv preprint arXiv:1905.03329* .

Fukui, H.; Hirakawa, T.; Yamashita, T.; and Fujiyoshi, H. 2019. Attention branch network: Learning of attention mechanism for visual explanation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 10705–10714.

Girdhar, R.; and Ramanan, D. 2017. Attentional pooling for action recognition. In *Advances in Neural Information Processing Systems*, 34–45.

Han, D.; Kim, J.; and Kim, J. 2017. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5927–5935.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hu, T.; Qi, H.; Huang, Q.; and Lu, Y. 2019. See better before looking closer: Weakly supervised data augmentation network for fine-grained visual classification. *arXiv preprint arXiv:1901.09891* .

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Kloeckner, B. 2010. A geometric study of Wasserstein spaces: Euclidean spaces. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze* 9(2): 297–323.

Kolkin, N.; Salavon, J.; and Shakhnarovich, G. 2019. Style transfer by relaxed optimal transport and self-similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 10051–10060.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *Citeseer* .

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 1097–1105.

Lin, M.; Chen, Q.; and Yan, S. 2013. Network in network. *arXiv preprint arXiv:1312.4400* .

Lin, T.-Y.; RoyChowdhury, A.; and Maji, S. 2015. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, 1449–1457.

Liu, B.; Rao, Y.; Lu, J.; Zhou, J.; and Hsieh, C.-J. 2020. Metadistiller: Network self-boosting via meta-learned top-down distillation. In *European Conference on Computer Vision*, 694–709. Springer.

Monge, G. 1781. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris* .

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 8026–8037.

Rao, Y.; Lu, J.; Lin, J.; and Zhou, J. 2018. Runtime network routing for efficient image classification. *IEEE transactions on pattern analysis and machine intelligence* 41(10): 2291–2304.

Recht, B.; Roelofs, R.; Schmidt, L.; and Shankar, V. 2018. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451* .

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3): 211–252. doi:10.1007/s11263-015-0816-y.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Sinkhorn, R. 1967. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly* 74(4): 402–405.

Solomon, J.; Rustamov, R.; Guibas, L.; and Butscher, A. 2014. Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics (TOG)* 33(4): 1–12.

Villani, C. 2008. *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset. *California Institute of Technology* .

Xu, H.; Luo, D.; Zha, H.; and Carin, L. 2019. Gromov-wasserstein learning for graph matching and node embedding. *arXiv preprint arXiv:1901.06003* .

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, 6023–6032.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146* .