# Supplementary Material of Unleash the Power of Local Representations: Unbiased Features and Adaptive Metric for Few-Shot Learning

**Shi Tang**
School of Software
Tsinghua University

**Chaoqun Chu**
School of Software
Tsinghua University

**Guiming Luo**[*]
School of Software
Tsinghua University

**Xinchen Ye**
DUT-RU ISE
Dalian University of Technology

**Zhiyi Xia**
School of Software
Tsinghua University

**Haojie Li**
DUT-RU ISE
Dalian University of Technology

This supplementary material is organized as follows:

- Section A presents the proof of Eq. (4) of the main text;
- Section B describes our experimental setup in detail;
- Section C shows some additional experimental results, including more visualized transport matrices (Section C.1) as a supplement to Figure (5) of the main text and some analysis on computational time (Section C.2).

## A Proof of Eq. (4)

Let $\mathbf{z} = [z_1, z_2, ..., z_{n_c}] \in \mathbb{R}^{1\times n_c}$ denote the network output for a local patch where $n_c$ is the total number of base classes and $z_i$ represents the logit of the $i$-th base class. Denote the softmax function as $\sigma$ and the probabilities of the $i$-th binary classification as $\mathbf{b}_i = [p_i, p_{\neg i}]$ where

$$p_i = \frac{\exp(z_i)}{\sum_{j=i}^{n_c} \exp(z_j)}, \quad p_{\neg i} = \frac{\sum_{k=i+1}^{n_c} \exp(z_k)}{\sum_{j=i}^{n_c} \exp(z_j)}$$

Using the superscripts $\mathcal{T}$ and $\mathcal{S}$ to mark the variables calculated using the output of the teacher $\mathbf{z}^{\mathcal{T}}$ and that of the student $\mathbf{z}^{\mathcal{S}}$, respectively, we want to prove that

$$KL(\sigma(\mathbf{z}^{\mathcal{T}})||\sigma(\mathbf{z}^{\mathcal{S}})) = \sum_{i=1}^{n_c-1} \mathbf{w}_i \cdot KL(\mathbf{b}_i^{\mathcal{T}}||\mathbf{b}_i^{\mathcal{S}}), \quad \mathbf{w}_i = \frac{\sum_{k=i}^{n_c} \exp(z_k^{\mathcal{T}})}{\sum_{j=1}^{n_c} \exp(z_j^{\mathcal{T}})}$$

Given the definition of $p_i, p_{\neg i}$, we need to additionally introduce the ordinary probability on full set $q_i$ as

$$q_i = \frac{\exp(z_i)}{\sum_{j=1}^{n_c} \exp(z_j)}$$

We have

$$p_i = \frac{\exp(z_i)}{\sum_{j=i}^{n_c} \exp(z_j)} = \frac{\exp(z_i)}{\sum_{j=1}^{n_c} \exp(z_j)} \cdot \frac{\sum_{j=1}^{n_c} \exp(z_j)}{\sum_{j=i}^{n_c} \exp(z_j)} = \frac{q_i}{\sum_{k=i}^{n_c} q_k} \tag{A}$$

[*]Corresponding author, gluo@tsinghua.edu.cn.

Preprint. Under review.

$$
\begin{aligned}
\mathbf{w}_i &= \sum_{k=i}^{n_c} q_k^{\mathcal{T}} \\
q_i^{\mathcal{T}} &= \mathbf{w}_i \cdot p_i^{\mathcal{T}}
\end{aligned} \tag{B}
$$

Therefore we have

$$
\begin{aligned}
p_{\neg i} = 1 - p_i &= \frac{\sum_{k=i+1}^{n_c} q_k}{\sum_{k=i}^{n_c} q_k} \\
\sum_{k=i+1}^{n_c} q_k &= p_{\neg i} \cdot \sum_{k=i}^{n_c} q_k \\
\sum_{k=i+1}^{n_c} q_k^{\mathcal{T}} &= \mathbf{w}_i \cdot p_{\neg i}^{\mathcal{T}}
\end{aligned} \tag{C}
$$

From the above equation, it can be concluded that

$$
\sum_{k=i}^{n_c} q_k = p_{\neg(i-1)} \cdot \sum_{k=i-1}^{n_c} q_k = (\prod_{k=1}^{i-1} p_{\neg k}) \cdot (\sum_{k=1}^{n_c} q_k) = \prod_{k=1}^{i-1} p_{\neg k} \tag{D}
$$

The KL-Divergence can be reformulated as

$$
\begin{aligned}
KL(\sigma(\mathbf{z}^{\mathcal{T}})||\sigma(\mathbf{z}^{\mathcal{S}})) &= \sum_{i=1}^{n_c} q_i^{\mathcal{T}} \log \frac{q_i^{\mathcal{T}}}{q_i^{\mathcal{S}}} \\
&= \sum_{i=1}^{n_c} q_i^{\mathcal{T}} (\log \frac{p_i^{\mathcal{T}}}{p_i^{\mathcal{S}}} + \log \frac{\sum_{k=i}^{n_c} q_k^{\mathcal{T}}}{\sum_{k=i}^{n_c} q_k^{\mathcal{S}}}) && \text{(Eq. (A))} \\
&= \sum_{i=1}^{n_c} q_i^{\mathcal{T}} \log \frac{p_i^{\mathcal{T}}}{p_i^{\mathcal{S}}} + \sum_{i=1}^{n_c} q_i^{\mathcal{T}} \log (\prod_{k=1}^{i-1} \frac{p_{\neg k}^{\mathcal{T}}}{p_{\neg k}^{\mathcal{S}}}) && \text{(Eq. (D))} \\
&= \sum_{i=1}^{n_c} \mathbf{w}_i \cdot p_i^{\mathcal{T}} \log \frac{p_i^{\mathcal{T}}}{p_i^{\mathcal{S}}} + \sum_{i=1}^{n_c} \sum_{k=1}^{i-1} q_i^{\mathcal{T}} \log \frac{p_{\neg k}^{\mathcal{T}}}{p_{\neg k}^{\mathcal{S}}} && \text{(Eq. (B))} \\
&= \sum_{i=1}^{n_c} \mathbf{w}_i \cdot p_i^{\mathcal{T}} \log \frac{p_i^{\mathcal{T}}}{p_i^{\mathcal{S}}} + \sum_{k=1}^{n_c-1} \sum_{i=k+1}^{n_c} q_i^{\mathcal{T}} \log \frac{p_{\neg k}^{\mathcal{T}}}{p_{\neg k}^{\mathcal{S}}} \\
&= \sum_{i=1}^{n_c} \mathbf{w}_i \cdot p_i^{\mathcal{T}} \log \frac{p_i^{\mathcal{T}}}{p_i^{\mathcal{S}}} + \sum_{k=1}^{n_c-1} \mathbf{w}_k \cdot p_{\neg k}^{\mathcal{T}} \log \frac{p_{\neg k}^{\mathcal{T}}}{p_{\neg k}^{\mathcal{S}}} && \text{(Eq. (C))} \\
&= \sum_{i=1}^{n_c-1} \mathbf{w}_i \cdot (p_i^{\mathcal{T}} \log \frac{p_i^{\mathcal{T}}}{p_i^{\mathcal{S}}} + p_{\neg i}^{\mathcal{T}} \log \frac{p_{\neg i}^{\mathcal{T}}}{p_{\neg i}^{\mathcal{S}}}) \\
&= \sum_{i=1}^{n_c-1} \mathbf{w}_i \cdot KL(\mathbf{b}_i^{\mathcal{T}}||\mathbf{b}_i^{\mathcal{S}})
\end{aligned}
$$

## B Detailed Experimental Setup

Following the "pretraining + episodic training" paradigm, the training process of our method can be divided into two stages. For the pretraining stage, we pre-train the encoder in the form of self-distillation based on the proxy task of standard multi-classification on $D_{base}$ and select the model with the highest validation accuracy. For the episodic training stage, each epoch involves 50 iterations with a batch size of 4. We first pre-train RM with the parameters of the encoder fixed. Then, the parameters of both the encoder and RM are optimized jointly. Globally, we set $n_p = 4$, $m = 0.999$, and $n = 25$. The cropped patches are resized to $84 \times 84$ before being embedded. For evaluation, we randomly sample $5000/600$ episodes for testing and report the average accuracy with the $95\%$ confidence interval for 1-shot/5-shot experiment following [1]. In the following, we describe our detailed experimental setup according to different benchmarks:

(1) ***mini*ImageNet.** The encoder is first pre-trained for 360 epochs where the SGD optimizer with a momentum of $0.9$ and a weight decay of $5e$-4 is adopted. $\mathcal{L}_{SKD}$ will not be used for the first 120 epochs to ensure the teacher has well-converged before being used. For the latter 240 epochs, the learning rate is set to $0.01$, and $\lambda$ is set to $0.1$. Then, in an episodic manner, RM is pre-trained for 100 epochs with the parameters of the encoder fixed, in which the Adam optimizer with a weight decay of $5e$-4 is adopted. The learning rate starts from $1e$-3 and decays by $0.1$ at epoch 60 and 90. Finally, all the parameters will be optimized jointly for another 100 epochs where the SGD optimizer with a momentum of $0.9$ and a weight decay of $5e$-4 is adopted. The learning rate starts from $5e$-4 and decays by $0.5$ every 10 epochs.

(2) ***tiered*ImageNet.** The encoder is first pre-trained for 240 epochs where the SGD optimizer with a momentum of $0.9$ and a weight decay of $5e$-4 is adopted. $\mathcal{L}_{SKD}$ will not be used for the first 120 epochs to ensure the teacher has well-converged before being used. For the latter 120 epochs, the learning rate is set to $0.001$, and $\lambda$ is set to $0.05$. Then, in an episodic manner, RM is pre-trained for 100 epochs with the parameters of the encoder fixed, in which the Adam optimizer with a weight decay of $5e$-4 is adopted. The learning rate starts from $1e$-3 and decays by $0.1$ at epoch 60 and 90. Finally, all the parameters will be optimized jointly for another 100 epochs where the SGD optimizer with a momentum of $0.9$ and a weight decay of $5e$-4 is adopted. The learning rate starts from $1e$-4 and decays by $0.5$ every 10 epochs.

(3) **CUB-200-2011.** The encoder is first pre-trained for 360 epochs where the SGD optimizer with a momentum of $0.9$ and a weight decay of $5e$-4 is adopted. $\mathcal{L}_{SKD}$ will not be used for the first 120 epochs to ensure the teacher has well-converged before being used. For the latter 240 epochs, the learning rate is set to $0.03$, and $\lambda$ is set to $0.5$. Then, in an episodic manner, RM is pre-trained for 100 epochs with the parameters of the encoder fixed, in which the Adam optimizer with a weight decay of $5e$-4 is adopted. The learning rate starts from $1e$-3 and decays by $0.1$ at epoch 60 and 90. Finally, all the parameters will be optimized jointly for another 100 epochs where the SGD optimizer with a momentum of $0.9$ and a weight decay of $5e$-4 is adopted. The learning rate starts from $1e$-3 and decays by $0.5$ every 10 epochs.

## C Additional Experimental Results

### C.1 Visualization of Solved Transport Matrices

For Figure 5 of the main text, we provide more results in Figure A.

For sets consisting of similar local patches, the transport matrices solved by EMD (Figure A (a)) tend to be very sparse, which is not a desired property because it tries to match a patch with few "most" similar opposite patches, neglecting the fact that the opposite patches are homogeneous. In contrast, dSD (Figure A (b)) generates smoother transport matrices. By allowing "one-to-many" matching, it enables a comprehensive utilization of opposite patches and reduces the dependency on a few opposite patches.

By making $\varepsilon$ a learnable parameter, RM can control the smoothness of the transport matrix self-adaptively. For sets consisting of similar local patches, RM produces a relatively larger $\varepsilon$, resulting in a smoother transport matrix (Figure A (b)). While for sets consisting of dissimilar local patches, a relatively smaller $\varepsilon$ is predicted, making the transport matrix moderately sparse (Figure A (c)). RM makes it possible for our method to handle various local feature sets by realizing an adaptive metric.

### C.2 Analysis on Computational Time

Although RM will inevitably bring additional time costs during inference as a parameterized module, the total time cost for measuring two feature sets is acceptable since the solving of the OT problem is accelerated. Specifically, the introduced entropic regularization makes the OT problem a strictly convex problem [2]. Thus, it can be solved by the Sinkhorn-Knopp algorithm [3] which is known to have a linear convergence [4, 5]. We conduct an experiment on an RTX-3090 (Linux, PyTorch 3.6) using the same $10,000$ randomly sampled episodes to compare the time cost empirically. The average time spent on the metric process and the average total time to handle an episode are reported

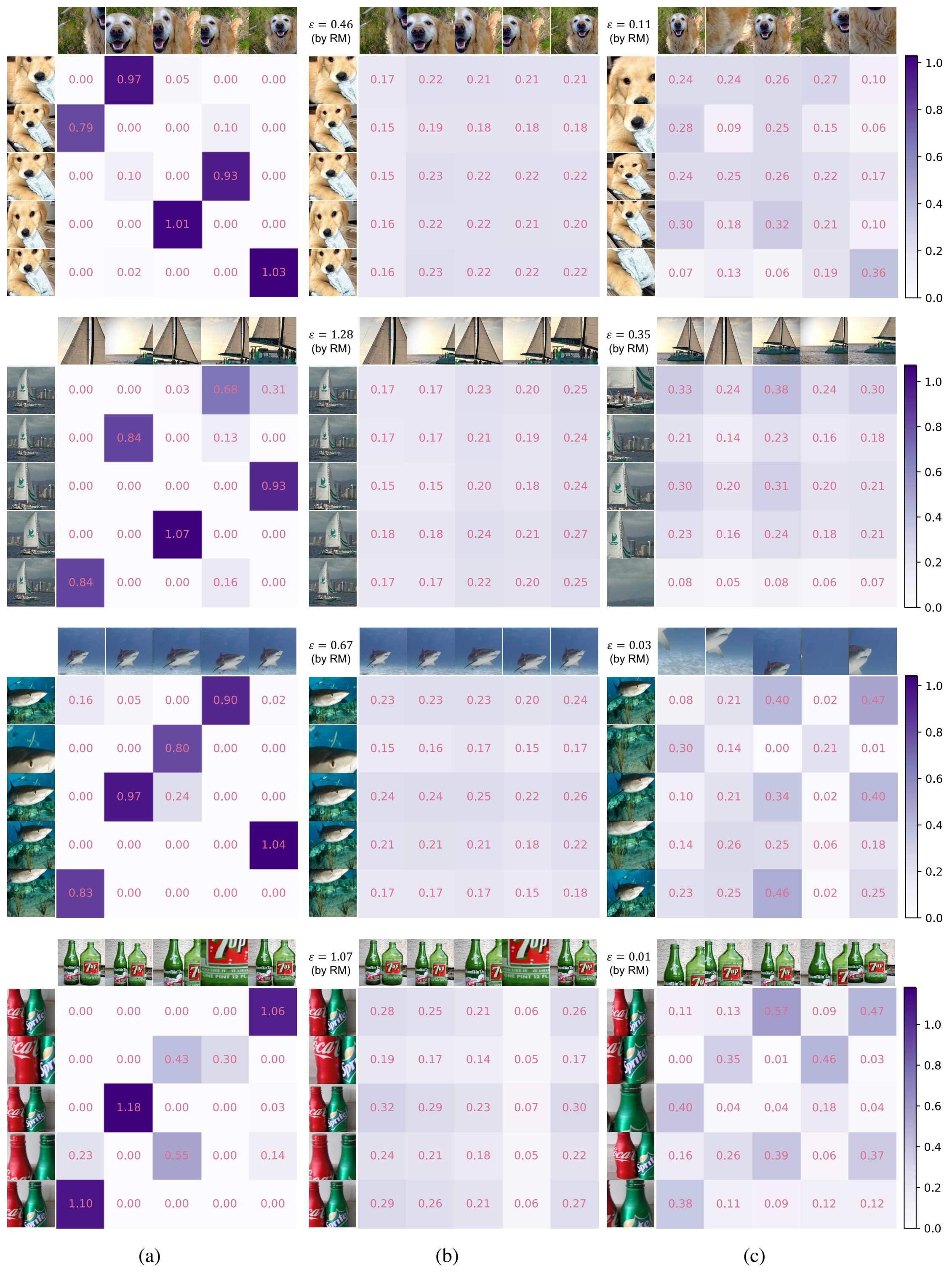

Figure A: Visualization of solved transport matrices. Results of (a) EMD and (b) AM for sets consisting of similar local patches, and the result of (c) AM for sets consisting of dissimilar local patches.

Table A: Comparison of the time cost between EMD and our adaptive metric.

| Metric | Average time for metric (ms) | Average total time (ms) |
|---|---|---|
| EMD | 893.58 | **1078.10** |
| Adaptive metric (ours) | **749.55** | 1081.82 |

in Table A. It can be seen that our adaptive metric (dSD + RM) spends less time on the metric process ($19.22\%$ faster). Although the total time spent processing an episode increases due to the additional feature processing, i.e., deep copy of local features and concatenation of features and set embeddings when preparing inputs for RM, the total time cost does not exceed the algorithm using EMD by much and is generally acceptable (only $0.34\%$ slower).

## References

[1] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Differentiable earth mover's distance for few-shot learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–17, 2022.

[2] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[3] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[4] Joel Franklin and Jens Lorenz. On the scaling of multidimensional matrices. *Linear Algebra and its Applications*, 114-115:717–735, 1989.

[5] Philip A. Knight. The sinkhorn–knopp algorithm: Convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.