Variational Metric Scaling for Metric-Based Meta-Learning

Jiaxin Chen, Li-Ming Zhan, Xiao-Ming Wu*, Fu-lai Chung*

Department of Computing
The Hong Kong Polytechnic University
{jiax.chen, lmzhan.zhan}@connect.polyu.hk, xiao-ming.wu@polyu.edu.hk, cskchung@comp.polyu.edu.hk

Abstract

Metric-based meta-learning has attracted a lot of attention due to its effectiveness and efficiency in few-shot learning. Recent studies show that metric scaling plays a crucial role in the performance of metric-based meta-learning algorithms. However, there still lacks a principled method for learning the metric scaling parameter automatically. In this paper, we recast metric-based meta-learning from a Bayesian perspective and develop a variational metric scaling framework for learning a proper metric scaling parameter. Firstly, we propose a stochastic variational method to learn a single global scaling parameter. To better fit the embedding space to a given data distribution, we extend our method to learn a dimensional scaling vector to transform the embedding space. Furthermore, to learn task-specific embeddings, we generate task-dependent dimensional scaling vectors with amortized variational inference. Our method is end-to-end without any pre-training and can be used as a simple plug-and-play module for existing metric-based meta-algorithms. Experiments on *mini*ImageNet show that our methods can be used to consistently improve the performance of existing metric-based meta-algorithms including prototypical networks and TADAM.

1. Introduction

Few-shot learning (Li, Fergus, and Perona 2006) aims to assign unseen samples (*query*) to the belonging categories with very few labeled samples (*support*) in each category. A promising paradigm for few-shot learning is meta-learning, which learns general patterns from a large number of tasks for fast adaptation to unseen tasks. Recently, metric-based meta-learning algorithms (Garcia and Bruna 2017; Koch, Zemel, and Salakhutdinov 2015; Snell, Swersky, and Zemel 2017; Vinyals et al. 2016) demonstrate great potential in few-shot classification. Typically, they learn a general mapping, which projects queries and supports into an embedding space. These models are trained in an episodic manner (Vinyals et al. 2016) by minimizing the distances between a query and same-labeled supports in the embedding space. Given a new task in testing phase, a nearest neighbour classifier is applied to assign a query to its nearest class in the embedding space.

*Corresponding authors.

Many metric-based meta-algorithms (short form for meta-learning algorithms) employ a softmax classifier with cross-entropy loss, which is computed with the logits being the distances between a query and supports in the embedding (metric) space. However, it has been shown that the scale of the logits – the metric scaling parameter, is critical to the performance of the learned model. Snell, Swersky, and Zemel (2017) found that Euclidean distance significantly outperforms cosine similarity in few-shot classification, while Oreshkin, López, and Lacoste (2018) and Wang et al. (2018) pointed out that there is no clear difference between them if the logits are scaled properly. They supposed that there exists an optimal metric scaling parameter which is data and architecture related, but they only used cross validation to manually set the parameter, which requires pre-training and cannot find an ideal solution.

In this paper, we aim to design an end-to-end method that can automatically learn an accurate metric scaling parameter. Given a set of training tasks, to learn a data-dependent metric scaling parameter that can generalize well to a new task, Bayesian posterior inference over learnable parameters is a theoretically attractive framework (Gordon et al. 2018; Ravi and Beatson 2019). We propose to recast metric-based meta-algorithms from a Bayesian perspective and take the metric scaling parameter as a global parameter. As exact posterior inference is intractable, we introduce a variational approach to efficiently approximate the posterior distribution with stochastic variational inference.

While a proper metric scaling parameter can improve classification accuracy via adjusting the cross-entropy loss, it simply rescales the embedding space but does not change the relative locations of the embedded samples. To transform the embedding space to better fit the data distribution, we propose a dimensional variational scaling method to learn a scaling parameter for each dimension, i.e., a metric scaling vector. Further, in order to learn task-dependent embeddings (Oreshkin, López, and Lacoste 2018), we propose an amortized variational approach to generate task-dependent metric scaling vectors, accompanied by an auxiliary training strategy to avoid time-consuming pre-training or co-training.

Our metric scaling methods can be used as pluggable modules for metric-based meta-algorithms. For example, it can

be incorporated into prototypical networks (PN) (Snell, Swersky, and Zemel 2017) and all PN-based algorithms to improve their performance. To verify this, we conduct extensive experiments on the *mini*ImageNet benchmark for few-shot classification progressively. First, we show that the proposed stochastic variational approach consistently improves on PN, and the improvement is large for PN with cosine similarity. Second, we show that the dimensional variational scaling method further improves upon the one with single scaling parameter, and the task-dependent metric scaling method with amortized variational inference achieves the best performance. We also incorporate the dimensional metric scaling method into TADAM (Oreshkin, López, and Lacoste 2018) in conjunction with other tricks proposed by the authors and observe notable improvement. Remarkably, after incorporating our method, TADAM achieves highly competitive performance compared with state-of-the-art methods.

To sum up, our contributions are as follows:

- We propose a generic variational approach to automatically learn a proper metric scaling parameter for metric-based meta-algorithms.

- We extend the proposed approach to learn dimensional and task-dependent metric scaling vectors to find a better embedding space by fitting the dataset at hand.

- As a pluggable module, our method can be efficiently used to improve existing metric-based meta-algorithms.

2. Related Work

Metric-based meta-learning. Koch, Zemel, and Salakhutdinov (2015) proposed the first metric-based meta-algorithm for few-shot learning, in which a siamese network (Chopra et al. 2005) is trained with the triplet loss to compare the similarity between a query and supports in the embedding space. Matching networks (Vinyals et al. 2016) proposed the episodic training strategy and used the cross-entropy loss where the logits are the distances between a query and supports. Prototypical networks (Snell, Swersky, and Zemel 2017) improved Matching networks by computing the distances between a query and the prototype (mean of supports) of each class. Many metric-based meta-algorithms (Oreshkin, López, and Lacoste 2018; Fort 2017; Sung et al. 2018; Li et al. 2019) extended prototypical networks in different ways.

Some recent methods proposed to improve prototypical networks by extracting task-conditioning features. Oreshkin, López, and Lacoste (2018) trained a network to generate task-conditioning parameters for batch normalization. Li et al. (2019) extracted task-relevant features with a category traversal module. Our methods can be incorporated into these methods to improve their performance.

In addition, there are some works related to our proposed dimensional scaling methods. Kang et al. (2019) trained a meta-model to re-weight features obtained from the base feature extractor and applied it for few-shot object detection. Lai et al. (2018) proposed a generator to generate task-adaptive weights to re-weight the embeddings, which can be seen as a special case of our amortized variational scaling method.

Metric scaling. Cross-entropy loss is widely used in many machine learning problems, including metric-based meta-learning and metric learning (Babenko and Lempitsky 2015; Ranjan, Castillo, and Chellappa 2017; Liu et al. 2017; Wang et al. 2017; Zhang et al. 2018; Babenko and Lempitsky 2015; Wan et al. 2018). In metric learning, the influence of metric scaling on the cross-entropy loss was first studied in Wang et al. (2017) and Ranjan, Castillo, and Chellappa (2017). They treated the metric scaling parameter as a trainable parameter updated with model parameters or a fixed hyperparameter. Zhang et al. (2018) proposed a "heating-up" scaling strategy, where the metric scaling parameter decays manually during the training process. The scaling of logits in cross-entropy loss for model compression was also studied in Hinton, Vinyals, and Dean (2015), where it is called temperature scaling. The temperature scaling parameter has also been used in confidence calibration (Guo et al. 2017).

The effect of metric scaling for few-shot learning was first discussed in Snell, Swersky, and Zemel (2017) and Oreshkin, López, and Lacoste (2018). The former found that Euclidean distance outperforms cosine similarity significantly in prototypical networks, and the latter argued that the superiority of Euclidean distance could be offset by imposing a proper metric scaling parameter on cosine similarity and using cross validation to select the parameter.

3. Preliminaries

3.1. Notations and Problem Statement

Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ be a domain where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. Assume we observe a meta-sample $\mathbf{S} = \{\mathcal{D}_i = \mathcal{D}_i^{tr} \cup \mathcal{D}_i^{ts}\}_{i=1}^n$ including n training tasks, where the i-th task consists of a support set of size m, $\mathcal{D}_i^{tr} = \{z_{i,j} = (x_{i,j}, y_{i,j})\}_{j=1}^m$, and a query set of size q, $\mathcal{D}_i^{ts} = \{z_{i,j} = (x_{i,j}, y_{i,j})\}_{j=m+1}^{m+q}$. Each training data point $z_{i,j}$ belongs to the domain $\mathcal{Z}$. Denote by θ the model parameters and α the metric scaling parameter. Given a new task and a support set $\mathcal{D}^{tr}$ sampled from the task, the goal is to predict the label y of a query x.

3.2. Prototypical Networks

Prototypical networks (PN) (Snell, Swersky, and Zemel 2017) is a popular and highly effective metric-based meta-algorithm. PN learns a mapping ϕ_θ which projects queries and the supports to an M-dimensional embedding space. For each class $k \in \{1, 2, \ldots, K\}$, the mean vector of the supports of class k in the embedding space is computed as the class *prototype* $\mathbf{c}_k$. The embedded query is compared with the prototypes and assigned to the class of the nearest prototype. Given a similarity metric $d : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}^+$, the probability of a query $z_{i,j}$ belonging to class k is,

$$p_\theta(y_{i,j} = k | x_{i,j}, \mathcal{D}_i^{tr}) = \frac{e^{-d(\phi_\theta(x_{i,j}), \mathbf{c}_k)}}{\sum_{k'=1}^K e^{-d(\phi_\theta(x_{i,j}), \mathbf{c}_{k'})}}. \quad (1)$$

Training proceeds by minimizing the cross-entropy loss, i.e., the negative log-probability $-\log p_\theta(y_{i,j} = k | x_{i,j}, \mathcal{D}_i^{tr})$ of its true class k. After introducing the metric scaling parameter

α, the classification loss of the ith task becomes

$$\mathcal{L}(\theta; \mathcal{D}_i) = -\sum_{j=m+1}^{m+q} \log \frac{e^{-\alpha * d(\phi_\theta(x_{i,j}), \mathbf{c}_{y_{i,j}})}}{\sum_{k'=1}^{K} e^{-\alpha * d(\phi_\theta(x_{i,j}), \mathbf{c}_{k'})}}. \quad (2)$$

The metric scaling parameter α has been found to affect the performance of PN significantly.

4. Variational Metric Scaling

4.1. Stochastic Variational Scaling

In the following, we recast metric-based meta-learning from a Bayesian perspective. The predictive distribution can be parameterized as

$$p_\theta(y|x, \mathcal{D}^{tr}, \mathbf{S}) = \int p_\theta(y|x, \mathcal{D}^{tr}, \alpha) p_\theta(\alpha|\mathbf{S}) d\alpha. \quad (3)$$

The conditional distribution $p_\theta(y|x, \mathcal{D}^{tr}, \alpha)$ is the discriminative classifier parameterized by θ. Since the posterior distribution $p_\theta(\alpha|\mathbf{S})$ is intractable, we propose a variational distribution $q_\psi(\alpha)$ parameterized by parameters ψ to approximate $p_\theta(\alpha|\mathbf{S})$. By minimizing the KL divergence between the approximator $q_\psi(\alpha)$ and the real posterior distribution $p_\theta(\alpha|\mathbf{S})$, we obtain the objective function

$$\begin{aligned}
\mathcal{L}(\psi, \theta; \mathbf{S}) &= \int q_\psi(\alpha) \log \frac{q_\psi(\alpha)}{p_\theta(\alpha|\mathbf{S})} d\alpha \\
&= -\int q_\psi(\alpha) \log \frac{p_\theta(\mathbf{S}|\alpha) p(\alpha)}{q_\psi(\alpha)} d\alpha + \log p(\mathbf{S}) \\
&= -\int q_\psi(\alpha) \log p_\theta(\mathbf{S}|\alpha) d\alpha + KL(q_\psi(\alpha)|p(\alpha)) + \text{const} \\
&= -\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \int q_\psi(\alpha) \log p_\theta(y_{i,j}|x_{i,j}, \mathcal{D}_i^{tr}, \alpha) d\alpha \\
&\quad + KL(q_\psi(\alpha)|p(\alpha)) + \text{const}.
\end{aligned} \quad (4)$$

We want to optimize $\mathcal{L}(\psi, \theta; \mathbf{S})$ w.r.t. both the model parameters θ and the variational parameters ψ. The gradient and the optimization procedure of the model parameters θ are similar to the original metric-based meta-algorithms (Vinyals et al. 2016; Snell, Swersky, and Zemel 2017) as shown in Algorithm 1.

To derive the gradients of the variational parameters, we leverage the re-parameterization trick proposed by Kingma and Welling (2013) to derive a practical estimator of the variational lower bound and its derivatives w.r.t. the variational parameters. In this paper, we use this trick to estimate the derivatives of $\mathcal{L}(\psi, \theta; \mathbf{S})$ w.r.t. ψ. For a distribution $q_\psi(\alpha)$, we can re-parameterize $\alpha \sim q_\psi(\alpha)$ using a differentiable transformation $\alpha = g_\psi(\epsilon)$, if exists, of an auxiliary random variable ϵ. For example, given a Gaussian distribution $q_{\mu,\sigma}(\alpha) = \mathcal{N}(\mu, \sigma^2)$, the re-parameterization is $g_{\mu,\sigma}(\epsilon) = \epsilon\sigma + \mu$, where $\epsilon \sim \mathcal{N}(0, 1)$. Hence, the first term in (4) is formulated as $-\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \mathbb{E}_{\epsilon \sim p(\epsilon)} \log p_\theta(y_{i,j}|x_{i,j}, \mathcal{D}_i^{tr}, g_\psi(\epsilon))$.

We apply a Monte Carlo integration with a single sample $\alpha_i = g_\psi(\epsilon_i)$ for each task to get an unbiased estimator. Note that α_i is sampled for the task $\mathcal{D}_i$ rather than for each instance, i.e., $\{z_{i,j}\}_{j=1}^{m+q}$ share the same α_i. The second term in (4) can

be computed with a given prior distribution $p(\alpha)$. Then, the final objective function is

$$\begin{aligned}
\mathcal{L}(\psi, \theta; \mathbf{S}) &= -\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \log p_\theta(y_{i,j}|x_{i,j}, \mathcal{D}_i^{tr}, g_\psi(\epsilon_i)) \\
&\quad + KL(q_\psi(\alpha)|p(\alpha))
\end{aligned} \quad (5)$$

Estimation of gradients. The objective function (5) is a general form. Here, we consider $q_\psi(\alpha)$ as a Gaussian distribution $q_{\mu,\sigma}(\alpha) = \mathcal{N}(\mu, \sigma^2)$. The prior distribution is also a Gaussian distribution $p(\alpha) = \mathcal{N}(\mu_0, \sigma_0^2)$. By the fact that the KL divergence of two Gaussian distributions has a closed-form solution, we obtain the following objective function

$$\begin{aligned}
\mathcal{L}(\mu, \sigma, \theta; \mathbf{S}) &= -\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \log p_\theta(y_{i,j}|x_{i,j}, \mathcal{D}_i^{tr}, g_{\mu,\sigma}(\epsilon_i)) \\
&\quad + \log \frac{\sigma_0}{\sigma} + \frac{\sigma^2 + (\mu - \mu_0)^2}{2\sigma_0^2},
\end{aligned} \quad (6)$$

where $g_{\mu,\sigma}(\epsilon_i) = \sigma\epsilon_i + \mu$. The derivatives of $\mathcal{L}(\mu, \sigma, \theta; \mathbf{S})$ w.r.t. μ and σ respectively are

$$\begin{aligned}
&\frac{\partial \mathcal{L}(\mu, \sigma, \theta; \mathbf{S})}{\partial \mu} = \\
&-\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \frac{\partial \log p_\theta(y_{i,j}|x_{i,j}, \mathcal{D}_i^{tr}, g_{\mu,\sigma}(\epsilon_i))}{\partial g_{\mu,\sigma}(\epsilon_i)} + \frac{\mu - \mu_0}{\sigma_0^2},
\end{aligned} \quad (7)$$

$$\begin{aligned}
&\frac{\partial \mathcal{L}(\mu, \sigma, \theta; \mathbf{S})}{\partial \sigma} = \\
&-\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \frac{\partial \log p_\theta(y_{i,j}|x_{i,j}, \mathcal{D}_i^{tr}, g_{\mu,\sigma}(\epsilon_i))}{\partial g_{\mu,\sigma}(\epsilon_i)} * \epsilon_i \\
&-\frac{1}{\sigma} + \frac{\sigma}{\sigma_0}.
\end{aligned} \quad (8)$$

In particular, we apply the proposed variational metric scaling method to Prototypical Networks with feature extractor ϕ_θ. The details of the gradients and the iterative update procedure are shown in Algorithm 1. It can be seen that the gradients of the variational parameters are computed using the intermediate quantities in the computational graph of the model parameters θ during back-propagation, hence the computational cost is very low.

For meta-testing, we use μ (mean) as the metric scaling parameter for inference.

The proposed variational scaling framework is general. Note that training the scaling parameter α together with the model parameters (Ranjan, Castillo, and Chellappa 2017) is a special case of our framework, when $q_\psi(\alpha)$ is defined as $\mathcal{N}(\mu, 0)$, the variance of the prior distribution is $\sigma_0 \to \infty$, and the learning rate is fixed as $l_\theta = l_\psi$.

4.2. Dimensional Stochastic Variational Scaling

Metric scaling can be seen as a transformation of the metric (embedding) space. Multiplying the distances with the scaling parameter accounts to re-scaling the embedding space.
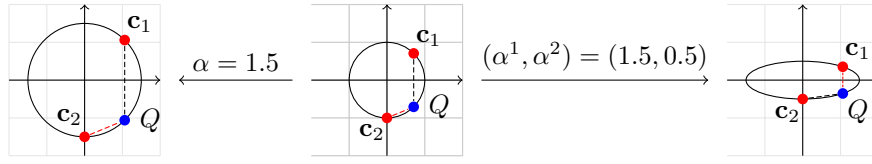
Figure 1: The middle figure shows a metric space in which the query (blue) and the support samples (red) are normalized to a unit ball. The left and right figures show the spaces scaled by a single parameter $\alpha = 1.5$ and a two-dimensional vector $(\alpha^1, \alpha^2) = (1.5, 0.5)$, respectively. The query Q is still assigned to class 1 in the left figure but to class 2 in the right one.

Algorithm 1: Stochastic Variational Scaling for Prototypical Networks

Input: Meta-sample $\{\mathcal{D}_i\}_{i=1}^n$, learning rates l_θ, l_ψ and μ_0, σ_0.
Initialize: μ, σ and θ randomly.

1 **for** i *in* $\{1, 2, \ldots, n\}$ **do**
2    $\epsilon_i \sim \mathcal{N}(0,1), \alpha_i = \sigma\epsilon_i + \mu$
             // Sample α_i for i^{th} task.
3   **for** k *in* $\{1, 2, \ldots, K\}$ **do**
4      $\mathbf{c}_k = \frac{1}{N} \sum_{z_{i,j} \in \mathcal{D}_i^{tr}, y_{i,j}=k} \phi_\theta(x_{i,j})$
             // Compute prototypes.
5   **for** j *in* $\{m+1, 2, \ldots, m+q\}$ **do**
6      $d(x_{i,j}, \mathbf{c}_k) = \|\phi_\theta(x_{i,j}) - \mathbf{c}_k\|_2^2$
7      $p(y_{i,j} = k) = \frac{e^{-\alpha_i * d(x_{i,j}, \mathbf{c}_k)}}{\sum_{k'=1}^K e^{-\alpha_i * d(x_{i,j}, \mathbf{c}_{k'})}}$
8    $\theta = \theta - l_\theta * \nabla_\theta \mathcal{L}(\mu, \sigma, \theta; \mathcal{D}_i)$
             // Update the model parameters θ.
9    $\mu = \mu - l_\psi * (\sum_{j=m+1}^{m+q}(-d(x_{i,j}, \mathbf{c}_{y_{i,j}}) + \sum_{k'=1}^K p(y_{i,j} = k') * d(x_{i,j}, \mathbf{c}_{k'})) + \frac{\mu - \mu_0}{\sigma_0^2})$
10    $\sigma = \sigma - l_\psi * (\sum_{j=m+1}^{m+q} \epsilon_i * (-d(x_{i,j}, \mathbf{c}_{y_{i,j}}) + \sum_{k'=1}^K p(y_{i,j} = k') * d(x_{i,j}, \mathbf{c}_{k'})) - \frac{1}{\sigma} + \frac{\sigma}{\sigma_0})$
             // Update the variational parameters $\psi = \{\mu, \sigma\}$.

By this point of view, we generalize the single scaling parameter to a dimensional scaling vector which transforms the embedding space to fit the data.

If the dimension of the embedding space is too low, the data points cannot be projected to a linearly-separable space. Conversely, if the dimension is too high, there may be many redundant dimensions. The optimal number of dimensions is data-dependent and difficult to be selected as a hyperparameter before training. Here, we address this problem by learning a data-dependent dimensional scaling vector to modify the embedding space, i.e., learning different weights for each dimension to highlight the important dimensions and reduce the influence of the redundant ones. Figure 1 shows a two-dimensional example. It can be seen that the single scaling parameter α simply changes the scale of the embedding space, but the dimensional scaling $\alpha = (\alpha^1, \alpha^2)$ changes the relative locations of the query and the supports.

The proposed dimensional stochastic variational scaling

method is similar to Algorithm 1, with the variational parameters $\mu = (\mu^1, \mu^2, \ldots, \mu^M)$ and $\sigma = (\sigma^1, \sigma^2, \ldots, \sigma^M)$. Accordingly, the metric scaling operation is changed to

$$d(x_{i,j}, \mathbf{c}_k)$$
$$= (\phi_\theta(x_{i,j}) - \mathbf{c}_k)^T \begin{pmatrix} \alpha_i^1 & & & \\ & \alpha_i^2 & & \\ & & \ddots & \\ & & & \alpha_i^M \end{pmatrix} (\phi_\theta(x_{i,j}) - \mathbf{c}_k). \quad (9)$$

The gradients of the variational parameters are still easy to compute and the computational cost can be ignored.

4.3. Amortized Variational Scaling

The proposed stochastic variational scaling methods above consider the metric scale as a global scalar or vector parameter, i.e., the entire meta-sample $\mathbf{S} = \{\mathcal{D}_i\}_{i=1}^n$ shares the same embedding space. However, the tasks randomly sampled from the task distribution may have specific task-relevant feature representations (Kang et al. 2019; Lai et al. 2018; Li et al. 2019). To adapt the learned embeddings to the task-specific representations, we propose to apply amortized variational inference to learn the task-dependent dimensional scaling parameters.

For amortized variational inference, α is a local latent variable dependent on $\mathcal{D}$ instead of a global parameter. Similar to stochastic variational scaling, we apply the variational distribution $q_{\psi(\beta)}(\alpha|\mathcal{D})$ to approximate the posterior distribution $p_\theta(\alpha|\mathcal{D})$. In order to learn the dependence between α and $\mathcal{D}$, amortized variational scaling learns a mapping approximated by a neural network G_β, from the task $\mathcal{D}_i$ to the distribution parameters $\{\mu_i, \sigma_i\}$ of α_i.

By leveraging the re-parameterization trick, we obtain the objective function of amortized variational scaling:

$$\mathcal{L}(\beta, \theta; \mathbf{S}) = -\sum_{i=1}^n \sum_{j=m+1}^{m+q} \log p_\theta(y_{i,j}|x_{i,j}, \mathcal{D}_i^{tr}, g_{\mu_i, \sigma_i}(\epsilon_i))$$
$$+ \log \frac{\sigma_0}{\sigma_i^2} + \frac{\sigma_i^2 + (\mu_i - \mu_0)^2}{2\sigma_0^2}, \quad (10)$$

where $g_{\mu_i, \sigma_i}(\epsilon_i) = \sigma_i \epsilon_i + \mu_i$. Note that the local parameters $\{\mu_i, \sigma_i\}$ are functions of β, i.e., $\{\mu_i, \sigma_i\} = G_\beta(\mathcal{D}_i)$. We iteratively update β and θ by minimizing the loss function (10) during meta-training

During meta-testing, for each task, the generator produces a variational distribution's parameters and we still use the mean vector as the metric scaling vector for inference.

Algorithm 2: Dimensional Amortized Variational Scaling for Prototypical Networks

Input: Meta-sample $\{\mathcal{D}_i\}_{i=1}^n$, learning rates l_θ, l_β, prior μ_0, σ_0 and step size l_λ.

Initialize: β and θ randomly, $\lambda = 1$.

1 **for** i *in* $\{1, 2, \ldots, n\}$ **do**
2     $\mathbf{C}_i = \frac{1}{m+q} \sum_{j=1}^{m+q} \phi_\theta(x_{i,j})$
        `// Compute the task prototype.`
3     $\mu_i, \sigma_i = G_\beta(\mathbf{C}_i)$
4     $\epsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \alpha_i = \sigma_i \epsilon_i + \mu_i$
        `// Generate` μ_i `and` σ_i `for` i^{th} `task.`
5   **for** k *in* $\{1, 2, \ldots, K\}$ **do**
6         $\mathbf{c}_k = \frac{1}{N} \sum_{z_{i,j} \in \mathcal{D}_i^{tr}, y_{i,j}=k} \phi_\theta(x_{i,j})$
7   **for** j *in* $\{m+1, m+2, \ldots, m+q\}$ **do**
8         $d(x_{i,j}, \mathbf{c}_k) =$
        $(\phi_\theta(x_{i,j}) - \mathbf{c}_k)^T \begin{pmatrix} \alpha_i^1 & & \\ & \alpha_i^2 & \\ & & \ddots & \\ & & & \alpha_i^M \end{pmatrix}(\phi_\theta(x_{i,j}) - \mathbf{c}_k)$.
9     $\theta = \theta - l_\theta * \nabla_\theta \mathcal{L}_\lambda(\beta, \theta; \mathcal{D}_i)$
        `// Update the model parameters` θ.
10    $\beta = \beta - l_\beta * \nabla_\beta \mathcal{L}_\lambda(\beta, \theta; \mathcal{D}_i)$
        `// Update the parameters` β `of the generator.`
11   **if** $\lambda \neq 0$ **then**
12        $\lambda = \lambda - l_\lambda$

Auxiliary loss. To learn the mapping G_β from a set $\mathcal{D}_i$ to the variational parameters of the local random variable α_i, we compute the mean vector of the embedded queries and the embedded supports as the task prototype to generate the variational parameters. A problem is that the embeddings are not ready to generate good scaling parameters during early epochs. Existing approaches including co-training (Oreshkin, López, and Lacoste 2018) and pre-training (Li et al. 2019) can alleviate this problem at the expense of computational efficiency. They pre-train or co-train an auxiliary supervised learning classifier in a traditional supervised manner over the meta-sample $\mathbf{S}$, and then apply the pre-trained embeddings to generate the task-specific parameters and fine-tune the embeddings during meta-training. Here, we propose an end-to-end algorithm which can improve training efficiency in comparison with pre-training or co-training. We optimize the following loss function (11) where an auxiliary weight λ is used instead of minimizing (10) in Algorithm 2 , i.e.,

$$\mathcal{L}_\lambda(\beta, \theta; \mathbf{S}) = (1 - \lambda)\mathcal{L}(\beta, \theta; \mathbf{S}) + \lambda \mathcal{L}(\theta; \mathbf{S}), \quad (11)$$

where $\mathcal{L}(\theta; \mathbf{S}) = -\sum_{i=1}^n \sum_{j=m+1}^{m+q} \log p_\theta(y_{i,j}|x_{i,j}, \mathbf{1})$, i.e., no scaling is used. Given a decay step size γ, λ starts from 1 and linearly decays to 0 as the number of epochs increases, i.e., $\lambda = \lambda - 1/\gamma$. During the first epochs, the weight of the gradients $\frac{\partial \mathcal{L}(\theta; \mathbf{S})}{\partial \theta}$ is high and the algorithm learns the embeddings of PN. As the training proceeds, β is updated to tune the learned embedding space. See the details in Algorithm 2.

5. Experiments

To evaluate our methods, we plug them into two popular algorithms, prototypical networks (PN) (Snell, Swersky, and Zemel 2017) and TADAM (Oreshkin, López, and Lacoste 2018), implemented by both Conv-4 and ResNet-12 backbone networks. To be elaborated later, Table 1 shows our main results in comparison to state-of-the-art meta-algorithms, where it can been that our dimensional stochastic variational scaling algorithm outperforms other methods substantially. For TADAM, we incorporate our methods into TADAM in conjunction with all the techniques proposed in their paper and still observe notable improvement.

5.1. Dataset and Experimental Setup

***mini*ImageNet.** The *mini*ImageNet (Vinyals et al. 2016) consists of 100 classes with 600 images per class. We follow the data split suggested by Ravi and Larochelle (2017), where the dataset is separated into a training set with 64 classes, a testing set with 20 classes and a validation set with 16 classes.

Model architecture. To evaluate our methods with different backbone networks, we re-implement PN with the Conv-4 architecture proposed by Snell, Swersky, and Zemel (2017) and the ResNet-12 architecture adopted by Oreshkin, López, and Lacoste (2018), respectively.

The Conv-4 backbone contains four convolutional blocks, where each block is sequentially composed of a 3×3 kernel convolution with 64 filters, a batch normalization layer, a ReLU nonlinear layer and a 2×2 max-pooling layer.

The ResNet-12 architecture contains 4 Res blocks, where each block consists of 3 convolutional blocks followed by a 2×2 max-pooling layer.

Training details. We follow the episodic training strategy proposed by (Vinyals et al. 2016). In each episode, K classes and N shots per class are selected from the training set, the validation set or the testing set. For fair comparisons, the number of queries, the sampling strategy of queries, and the testing strategy are designed in line with PN or TADAM.

For Conv-4, we use Adam optimizer with a learning rate of $1e-3$ without weight decay. The total number of training episodes is $20,000$ for Conv-4. And for ResNet-12, we use SGD optimizer with momentum 0.9, weight decay $4e-4$ and $45,000$ episodes in total. The learning rate is initialized as 0.1 and decayed 90% at episode steps 15000, 30000 and 35000. Besides, we use gradient clipping when training ResNet-12. The reported results are the mean accuracies with 95% confidence intervals estimated by 5 runs.

We normalize the embeddings before computing the distances between them. As shown in Eq. (7) and (8), the gradient magnitude of variational metric scaling parameters is proportional to the norm of embeddings. Therefore, to foster the learning process of these parameters, we adopt a separate learning rate l_ψ for all variational metric scaling parameters.

5.2. Evaluation

The effectiveness of our proposed methods is illustrated in Table 2 progressively, including stochastic variational scaling (SVS), dimensional stochastic variational scaling (D-SVS)

*mini*ImageNet test accuracy				
Backbones	**Model**	5-way 1-shot	5-way 5-shot
	Matching networks (Vinyals et al. 2016)	43.56 ± 0.84	55.31 ± 0.73
	Relation Net (Sung et al. 2018)	50.44 ± 0.82	65.32 ± 0.70
	Meta-learner LSTM (Ravi and Larochelle 2017)	43.44 ± 0.77	60.60 ± 0.71
Conv-4	MAML (Finn, Abbeel, and Levine 2017)	48.70 ± 1.84	63.11 ± 0.92
	LLAMA (Grant et al. 2018)	49.40 ± 1.83	$-$
	REPTILE (Nichol and Schulman 2018)	49.97 ± 0.32	65.99 ± 0.58
	PLATIPUS (Finn, Xu, and Levine 2018)	50.13 ± 1.86	$-$
	adaResNet (Munkhdalai et al. 2017)	56.88 ± 0.62	71.94 ± 0.57
	SNAIL (Mishra et al. 2018)	55.71 ± 0.99	68.88 ± 0.92
	TADAM (Oreshkin, López, and Lacoste 2018)	58.50 ± 0.30	76.70 ± 0.30
ResNet-12	TADAM Euclidean + D-SVS (**ours**)	$\mathbf{60.16 \pm 0.47}$	$\mathbf{77.25 \pm 0.15}$
	PN Euclidean (Snell, Swersky, and Zemel 2017) *	53.89 ± 0.38	73.59 ± 0.48
	PN Cosine (Snell, Swersky, and Zemel 2017) *	52.31 ± 0.83	70.74 ± 0.24
	PN Euclidean + D-SVS (**ours**) *	$\mathbf{55.30 \pm 0.08}$	$\mathbf{74.93 \pm 0.31}$
	PN cosine + D-SVS (**ours**) *	$\mathbf{56.09 \pm 0.19}$	$\mathbf{74.46 \pm 0.17}$

Table 1: Test accuracies of 5-way classification tasks on *mini*ImageNet using Conv-4 and ResNet-12 respectively. * indicates results by our re-implementation.

	5-way 1-shot		5-way 5-shot	
	Euclidean	Cosine	Euclidean	Cosine
PN	44.15 ± 0.39	42.20 ± 0.66	65.49 ± 0.53	60.91 ± 0.50
PN + SVS	47.84 ± 0.16	48.43 ± 0.20	66.86 ± 0.06	67.02 ± 0.14
PN + D-SVS	49.01 ± 0.39	49.20 ± 0.05	67.40 ± 0.32	67.33 ± 0.23
PN + D-AVS	$\mathbf{49.10 \pm 0.14}$	$\mathbf{49.34 \pm 0.29}$	$\mathbf{68.04 \pm 0.16}$	$\mathbf{67.83 \pm 0.16}$

Table 2: Results of prototypical networks (the first row) and prototypical networks with SVS, D-SVS and D-AVS respectively by our re-implementation using Conv-4.

and dimensional amortized variational scaling (D-AVS). On both 5-way 5-shot and 5-way 1-shot classification, noticeable improvement can be seen after incorporating SVS into PN. Compared to SVS, D-SVS is more effective, especially for 5-way 1-shot classification. D-AVS performs even better than D-SVS by considering task-relevant information.

Performance of SVS. We study the performance of SVS by incorporating it into PN. We consider both 5-way and 20-way training scenarios. The prior distribution of the metric scaling parameter is set as $p(\alpha) = \mathcal{N}(1, 1)$ and the variational parameters are initialized as $\mu_{init} = 100$, $\sigma_{init} = 0.2$. The learning rate is set to be $l_\psi = 1e - 4$.

Results in Table 3 show the effect of the metric scaling parameter (SVS). Particularly, significant improvement is observed for the case of PN with cosine similarity and for the case of 5-way 1-shot classification. Moreover, it can be seen that with metric scaling there is no clear difference between the performance of Euclidean distance and cosine similarity.

We also compare the performance of a fixed $\sigma = 0.2$ with a trainable σ. We add a shifted ReLU activation function ($x = \max\{1e-2, x\}$) on the learned σ to ensure it is positive. Nevertheless, in our experiments, we observe that the training is very stable and the variance is always positive even without the ReLU activation function. We also find that there is no significant difference between the two settings. Hence, we treat σ as a fixed hyperparameter in other experiments.

Performance of D-SVS. We validate the effectiveness of D-SVS by incorporating it into PN and TADAM, with the results shown in Table 1 and Table 2. On 5-way-1-shot classification, for PN, we observe about 4.90% and 1.41% absolute increase in test accuracy with Conv-4 and ResNet-12 respectively; for TADAM, 1.66% absolute increase in test accuracy is observed. The learning rate for D-SVS is set to be $l_\psi = 16$. Here we use a large learning rate since the gradient magnitude of each dimension of the metric scaling vector is extremely small after normalizing the embeddings.

Performance of D-AVS. We evaluate the effectiveness of D-AVS by incorporating it into PN. We use a multi-layer perception (MLP) with one hidden layer as the generator G_β. The learning rate l_β is set to be $1e - 3$. In Table 2, on both 5-way 1-shot and 5-way 5-shot classification, we observe about 1.0% absolute increase in test accuracy for dimensional amortized variational scaling (D-AVS) over SVS with a single scaling parameter. In our experiments, the hyperparameter γ is selected from the range of $[100, 150]$ with 200 training epochs in total.

Ablation study of D-AVS. To assess the effects of the auxiliary training strategy and the prior information, we provide an ablation study as shown in Table 4. Without the auxiliary training and the prior information, D-AVS degenerates to a task-relevant weight generating approach (Lai et al. 2018).

	5-way 1-shot		5-way 5-shot	
	5-way training	20-way training	5-way training	20-way training
PN Euclidean	44.15 ± 0.39	48.05 ± 0.47	65.49 ± 0.53	67.32 ± 1.20
PN Cosine	42.20 ± 0.66	46.75 ± 0.18	60.91 ± 0.50	66.28 ± 0.14
PN Euclidean + SVS ($\sigma = 0.2$)	47.84 ± 0.16	51.15 ± 0.16	66.86 ± 0.06	68.00 ± 0.22
PN Cosine + SVS ($\sigma = 0.2$)	48.12 ± 0.13	51.74 ± 0.13	66.95 ± 0.78	67.88 ± 0.10
PN Euclidean + SVS (learned σ)	48.28 ± 0.14	51.36 ± 0.15	66.84 ± 0.30	67.80 ± 0.06
PN Cosine + SVS (learned σ)	48.43 ± 0.20	51.68 ± 0.18	67.02 ± 0.14	67.72 ± 0.16

Table 3: Results of prototypical networks and prototypical networks with SVS by our re-implementation using Conv-4.

		5-way 1-shot		5-way 5-shot	
Auxiliary training	Prior	Euclidean	Cosine	Euclidean	Cosine
		47.79 ± 0.10	47.45 ± 0.17	66.26 ± 0.48	66.03 ± 0.34
	✓	48.12 ± 0.55	47.49 ± 0.26	66.69 ± 0.25	66.43 ± 0.38
✓		48.56 ± 0.44	49.13 ± 0.32	67.11 ± 0.14	67.23 ± 0.19
✓	✓	$\mathbf{49.10 \pm 0.14}$	$\mathbf{49.34 \pm 0.29}$	$\mathbf{68.04 \pm 0.16}$	$\mathbf{67.83 \pm 0.16}$

Table 4: Ablation study of prototypical networks with D-AVS by our re-implementation using Conv-4.

Noticeable performance drops can be observed after removing the two components. Removing either one of them also leads to performance drop, but not as significant as removing both. The empirical results confirm the necessity of the auxiliary training and a proper prior distribution for amortized variational metric scaling.

5.3 Robustness Study

We also design experiments to show: 1) The convergence speed of existing methods does not slow down after incorporating our methods; 2) Given the same prior distribution, the variational parameters converge to the same values in spite of different learning rates and initializations.

For the iterative update of the model parameters θ and the variational parameters ψ, a natural question is whether it will slow down the convergence speed of the algorithm. Figure 2 shows the learning curves of PN and PN+D-SVS on both 5-way 1-shot and 5-way 5-shot classification. It can be seen that the incorporation of SVS does not reduce the convergence speed.

We plot the learning curves of the variational parameter μ w.r.t. different initializations and different learning rates l_ψ. Given the same prior distribution $\mu_0 = 1$, Fig. 3(a) shows that the variational parameter μ with different initializations will converge to the same value. Fig. 3(b) shows that μ is robust to different learning rates.

Conclusion

In this paper, we have proposed a generic variational metric scaling framework for metric-based meta-algorithms, under which three efficient end-to-end methods are developed. To learn a better embedding space to fit data distribution, we have considered the influence of metric scaling on the embedding space by taking into account data-dependent and task-dependent information progressively. Our methods are lightweight and can be easily plugged into existing metric-based meta-algorithms to improve their performance.
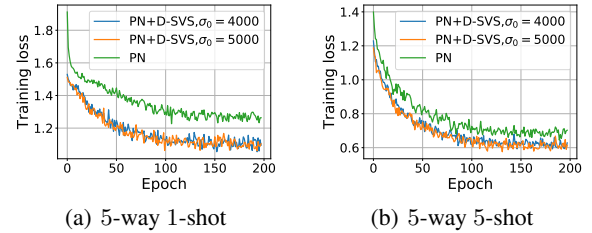

(a) 5-way 1-shot   (b) 5-way 5-shot

Figure 2: Learning curves of prototypical networks and prototypical networks with D-SVS.
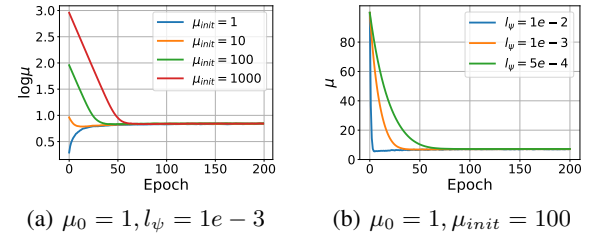

(a) $\mu_0 = 1, l_\psi = 1e-3$ (b) $\mu_0 = 1, \mu_{init} = 100$

Figure 3: Learning curves of μ (a) for different initializations and (b) for different learning rates.

Acknowledgements

References

[Babenko and Lempitsky 2015] Babenko, A., and Lempitsky, V. 2015. Aggregating deep convolutional features for image retrieval. *arXiv preprint arXiv:1510.07493*.

[Chopra et al. 2005] Chopra, S.; Hadsell, R.; LeCun, Y.; et al. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[Finn, Abbeel, and Levine 2017] Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1126–1135. JMLR. org.

[Finn, Xu, and Levine 2018] Finn, C.; Xu, K.; and Levine, S. 2018. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, 9516–9527.

[Fort 2017] Fort, S. 2017. Gaussian prototypical networks for few-shot learning on omniglot. *arXiv preprint arXiv:1708.02735*.

[Garcia and Bruna 2017] Garcia, V., and Bruna, J. 2017. Few-shot learning with graph neural networks. *International Conference on Learning Representations*.

[Gordon et al. 2018] Gordon, J.; Bronskill, J.; Bauer, M.; Nowozin, S.; and Turner, R. E. 2018. Meta-learning probabilistic inference for prediction. *International Conference on Learning Representations*.

[Grant et al. 2018] Grant, E.; Finn, C.; Levine, S.; Darrell, T.; and Griffiths, T. 2018. Recasting gradient-based meta-learning as hierarchical bayes. *International Conference on Learning Representations*.

[Guo et al. 2017] Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1321–1330. JMLR. org.

[Hinton, Vinyals, and Dean 2015] Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Kang et al. 2019] Kang, B.; Liu, Z.; Wang, X.; Yu, F.; Feng, J.; and Darrell, T. 2019. Few-shot object detection via feature reweighting. In *Proceedings of the IEEE International Conference on Computer Vision*, 8420–8429.

[Kingma and Welling 2013] Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *International Conference on Learning Representations*.

[Koch, Zemel, and Salakhutdinov 2015] Koch, G.; Zemel, R.; and Salakhutdinov, R. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

[Lai et al. 2018] Lai, N.; Kan, M.; Shan, S.; and Chen, X. 2018. Task-adaptive feature reweighting for few shot classification. In *Asian Conference on Computer Vision*, 649–662. Springer.

[Li et al. 2019] Li, H.; Eigen, D.; Dodge, S.; Zeiler, M.; and Wang, X. 2019. Finding task-relevant features for few-shot learning by category traversal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–10.

[Li, Fergus, and Perona 2006] Li, F.-F.; Fergus, R.; and Perona, P. 2006. One-shot learning of object categories. *IEEE*

transactions on pattern analysis and machine intelligence 28(4):594–611.

[Liu et al. 2017] Liu, W.; Wen, Y.; Yu, Z.; Li, M.; Raj, B.; and Song, L. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 212–220.

[Mishra et al. 2018] Mishra, N.; Rohaninejad, M.; Chen, X.; and Abbeel, P. 2018. A simple neural attentive meta-learner. *International Conference on Learning Representations*.

[Munkhdalai et al. 2017] Munkhdalai, T.; Yuan, X.; Mehri, S.; and Trischler, A. 2017. Rapid adaptation with conditionally shifted neurons. *arXiv preprint arXiv:1712.09926*.

[Nichol and Schulman 2018] Nichol, A., and Schulman, J. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999* 2.

[Oreshkin, López, and Lacoste 2018] Oreshkin, B.; López, P. R.; and Lacoste, A. 2018. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, 719–729.

[Ranjan, Castillo, and Chellappa 2017] Ranjan, R.; Castillo, C. D.; and Chellappa, R. 2017. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*.

[Ravi and Beatson 2019] Ravi, S., and Beatson, A. 2019. Amortized bayesian meta-learning. *International Conference on Learning Representations*.

[Ravi and Larochelle 2017] Ravi, S., and Larochelle, H. 2017. Optimization as a model for few-shot learning. *International Conference on Learning Representations*.

[Snell, Swersky, and Zemel 2017] Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 4077–4087.

[Sung et al. 2018] Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P. H.; and Hospedales, T. M. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1199–1208.

[Vinyals et al. 2016] Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, 3630–3638.

[Wan et al. 2018] Wan, W.; Zhong, Y.; Li, T.; and Chen, J. 2018. Rethinking feature distribution for loss functions in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9117–9126.

[Wang et al. 2017] Wang, F.; Xiang, X.; Cheng, J.; and Yuille, A. L. 2017. Normface: l 2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, 1041–1049. ACM.

[Wang et al. 2018] Wang, Y.; Wu, X.-M.; Li, Q.; Gu, J.; Xiang, W.; Zhang, L.; and Li, V. O. 2018. Large margin few-shot learning. *arXiv preprint arXiv:1807.02872*.

[Zhang et al. 2018] Zhang, X.; Yu, F. X.; Karaman, S.; Zhang, W.; and Chang, S.-F. 2018. Heated-up softmax embedding. *arXiv preprint arXiv:1809.04157*.

Appendix A. SVS

A.1. Comparison with a Special Case

Ranjan, Castillo, and Chellappa (2017) proposed to train the single scaling parameter together with model parameters. Their method can be seen as a special case of our stochastic variational scaling method SVS under the conditions of $q_\psi(\alpha) = \mathcal{N}(\mu, 0)$, $\sigma_0 \to \infty$ and $l_\psi = l_\theta$. We compare our method with theirs by varying the initialization of μ (μ_{init}).

Noticeably, our method achieves absolute improvements of 2.86%, 1.77%, 0.73% and 2.5% for four different initializations respectively. As shown in Table 5, our method is stable w.r.t. the initialization of μ, thanks to the prior information introduced in our Bayesian framework which may counteract the influence of initialization.

μ_{init} Method	1	10	100	1000
PN+SVS	66.45	66.95	67.02	66.72
PN (Training together)	63.59	65.18	66.29	64.22

Table 5: Comparison of PN (Training together) and PN+SVS implemented by Conv-4 backbone.

A.2. Sensitivity to the Prior and Initialization

In Bayesian framework, the prior distribution has a significant impact on learning posterior distribution. For stochastic variational inference, initialization is another key factor for learning the variational parameters. Here, we conduct experiments of PN+SVS with different prior distributions and initializations. The results of 5-way 5-shot classification are summarized in Table 6. It can be observed that our method is not sensitive to the prior and initialization as long as either one of them is not too small.

Appendix B. D-SVS

B.1. Distributions of the Mean Vector μ

Figure 4 illustrates the distributions of the mean vector $\mu = (\mu^1, \mu^2, \ldots, \mu^M)$ during the meta-training procedure of 5-way 1-shot and 5-way 5-shot classification respectively. Darker colour means more frequent occurrence.

At step 0, all dimensions of μ are initialized as 100. They diverge as the meta-training proceeds, which shows D-SVS successfully learns different scaling parameters for different dimensions. It is also worth noting that for both tasks, the distribution of μ converges eventually (after $15k$ steps).

Appendix C. D-AVS

C.1. Viewing the Learned Metric Scaling Parameters of PN+D-AVS.

D-AVS generates variational distributions for different tasks, from which the task-specific scaling parameters are sampled. Below we print out the scaling vectors learned by D-AVS on two different testing tasks for 5-way 5-shot classification, where only the first ten dimensions are displayed. It can be seen that D-AVS successfully learns tailored scaling vectors for different tasks.

Scaling parameters for Task 1: [64.9170, 22.4030, 13.4468, 2.3949, 28.2470, 29.7770, 54.4221, 60.9279, 2.3008, 147.5304].

Scaling parameters for Task 2: [60.2564, 21.1672, 12.8457, 2.3603, 26.6194, 27.9963, 50.6127, 56.5965, 2.2672, 135.2510].

Appendix D. Implementation Details

D.1. Sampling from the Variational Distribution

We adopt the following sampling strategy for the proposed three approaches. For meta-training, we sample once per task from the variational distribution for the metric scaling parameter; for meta-testing, we use the mean of the learned Gaussian distribution as the metric scaling parameter. The computational overhead is very small and can be ignored.

μ_0 \ μ_{init}	1	10	100	1000
1	60.25 ± 0.70	63.00 ± 0.34	66.86 ± 0.06	66.26 ± 0.24
10	63.95 ± 0.24	65.89 ± 0.32	66.79 ± 0.55	66.34 ± 0.22
100	65.94 ± 0.42	66.95 ± 0.27	67.02 ± 0.38	66.43 ± 0.10
1000	66.45 ± 0.17	66.66 ± 0.23	66.88 ± 0.16	66.72 ± 0.28

Table 6: Results of PN+SVS w.r.t. different initializations and priors implemented by Conv-4.
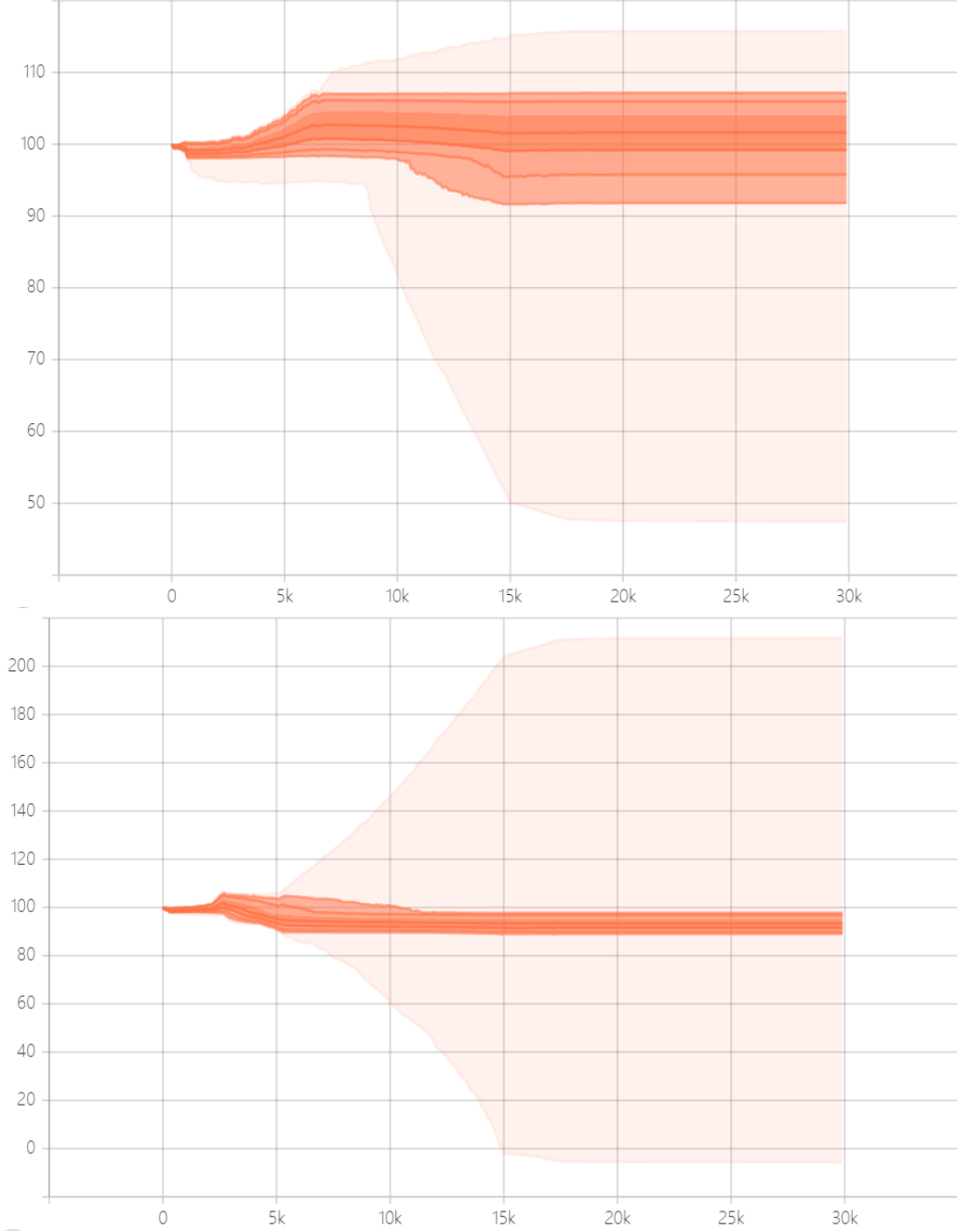

Figure 4: Distributions of the learned μ w.r.t. the number of training steps. The horizontal and vertical axes are the number of training steps and values of μ, respectively. The top is for 5-way 1-shot classification and the bottom is for 5-way 5-shot.