Meta-Learning

Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks ICML 2017

Meta-Learning with Latent Embedding Optimization ICLR 2019

LGM-Net:Learning to Generate Matching Networks for Few-Shot learning ICML 2019

Junjie Wang

Introduction

- Using episodic training scheme where each episode is designed to mimic a few-shot task.

- Meta-Learning and Metric-Learning

Introduction

- A generic meta-learning framework usually contains a meta-level learner and a base-level learner.
- The base-level learner is designed for specific tasks, such as classification, regression, and neural network policy.
- The meta-level learner aims to learn prior knowledge across different tasks. The prior knowledge can be transferred to the base-level learner to help quickly adapt to similar unseen tasks.

Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Contribution:

A simple model- and task-agnostic algorithm for meta-learning that trains a model's parameters such that a small number of gradient updates will lead to fast learning on a new task.

Aim to find model parameters that are sensitive to changes in the task, such that small changes in the task, such that small changes in the parameters will produce large imporvements on the loss function of any task drawn from $p(T)$

every task T_i drawn from $p(T)$

Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

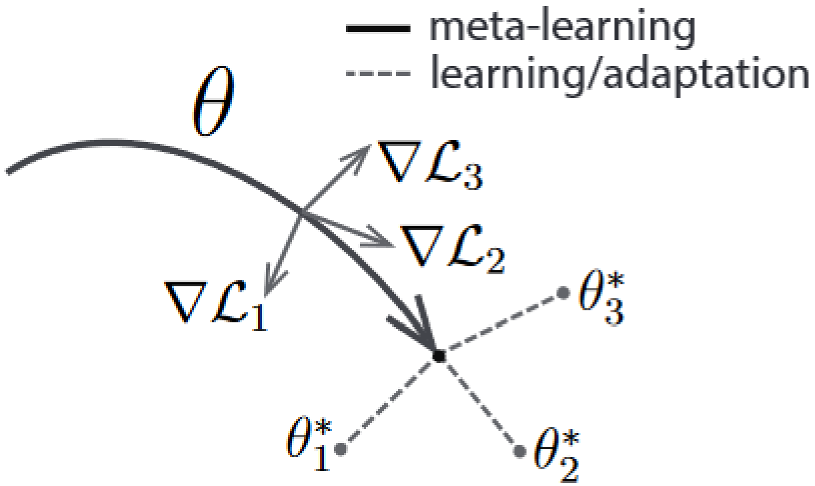


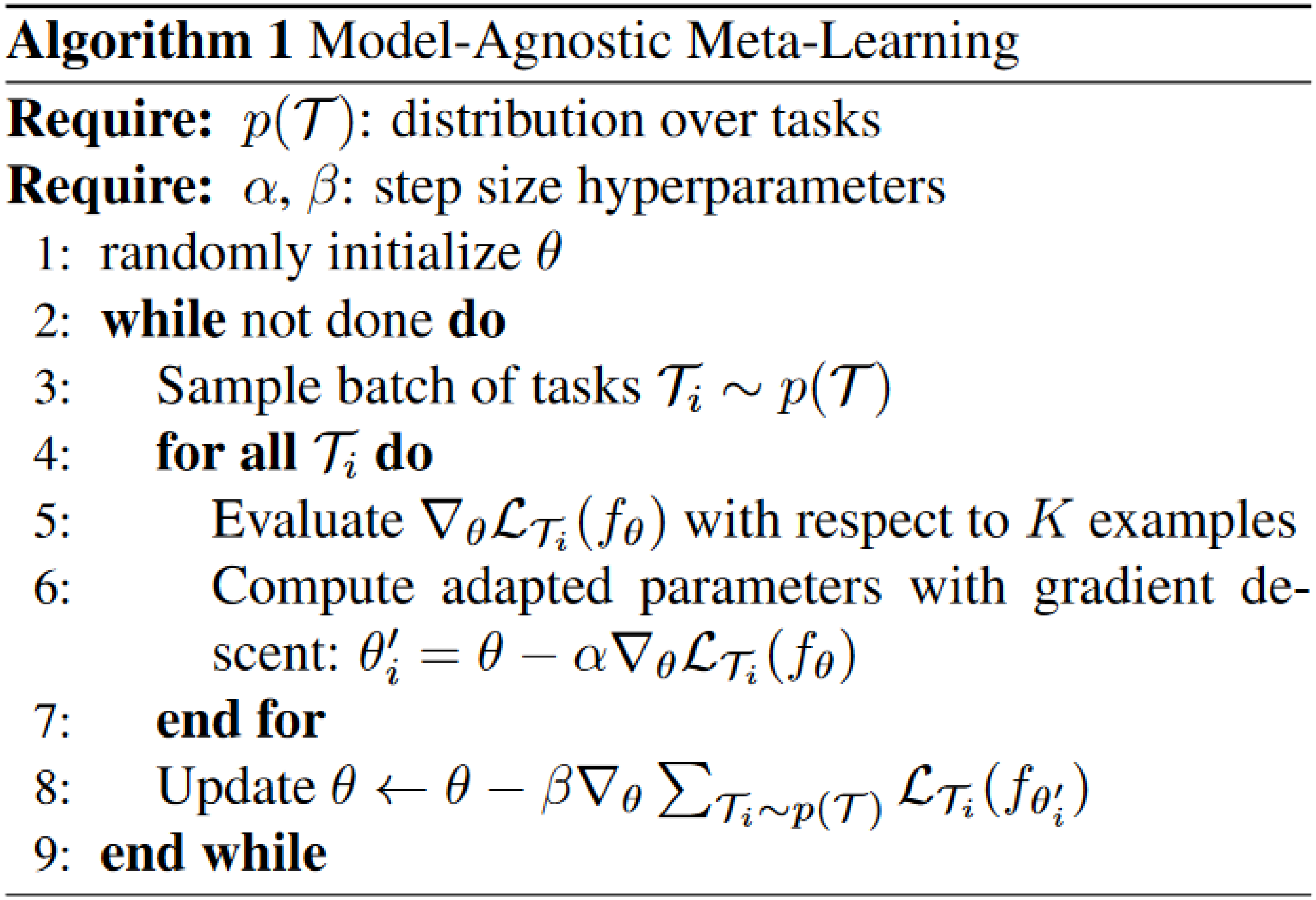
Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

$$\theta_i' = \theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_i}(f_\theta).$$

$$\min_\theta \sum_{\mathcal{T}_i\sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \sum_{\mathcal{T}_i\sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta-\alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_i}(f_\theta)})$$

$$\theta \leftarrow \theta - \beta\nabla_\theta \sum_{\mathcal{T}_i\sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks



Algorithm 1 Model-Agnostic Meta-Learning

```
Require: $p(\mathcal{T})$: distribution over tasks
Require: $\alpha$, $\beta$: step size hyperparameters
randomly initialize $\theta$
while not done do
    Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
    for all $\mathcal{T}_i$ do
        Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to $K$ examples
        Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    end for
    Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
end while
```

base-level learner

meta-level learner

Meta-Learning With Latent Embedding Optimization

- The former have parctical difficulties when operating on high-dimensional parameter spaces in extreme low-data regimes.

Meta-Learning With Latent Embedding Optimization

- Decoupling optimization-based meta-learning techniques from the high-dimensional space of model parameters by learning a stochastic latent space with an information bottleneck.

- finding a single optimal $\theta^{*} \in \Theta$,
- approximating a data-dependent conditional probability distribution over Θ .

Meta-Learning With Latent Embedding Optimization

- Encoding

$$\begin{aligned}\boldsymbol{\mu}_n^e, \boldsymbol{\sigma}_n^e &= \frac{1}{NK^2}\sum_{k_n=1}^{K}\sum_{m=1}^{N}\sum_{k_m=1}^{K} g_{\phi_r}\left(g_{\phi_e}\left(\mathbf{x}_n^{k_n}\right), g_{\phi_e}\left(\mathbf{x}_m^{k_m}\right)\right)\\ \mathbf{z}_n \sim q\left(\mathbf{z}_n|\mathcal{D}_n^{tr}\right) &= \mathcal{N}\left(\boldsymbol{\mu}_n^e, diag(\boldsymbol{\sigma}_n^{e\,2})\right)\end{aligned}$$

- Decoding

$$\begin{aligned}\boldsymbol{\mu}_n^d, \boldsymbol{\sigma}_n^d &= g_{\phi_d}\left(\mathbf{z}_n\right)\\ \mathbf{w}_n \sim p\left(\mathbf{w}|\mathbf{z}_n\right) &= \mathcal{N}\left(\boldsymbol{\mu}_n^d, diag(\boldsymbol{\sigma}_n^{d\,2})\right)\end{aligned}$$

Meta-Learning With Latent Embedding Optimization

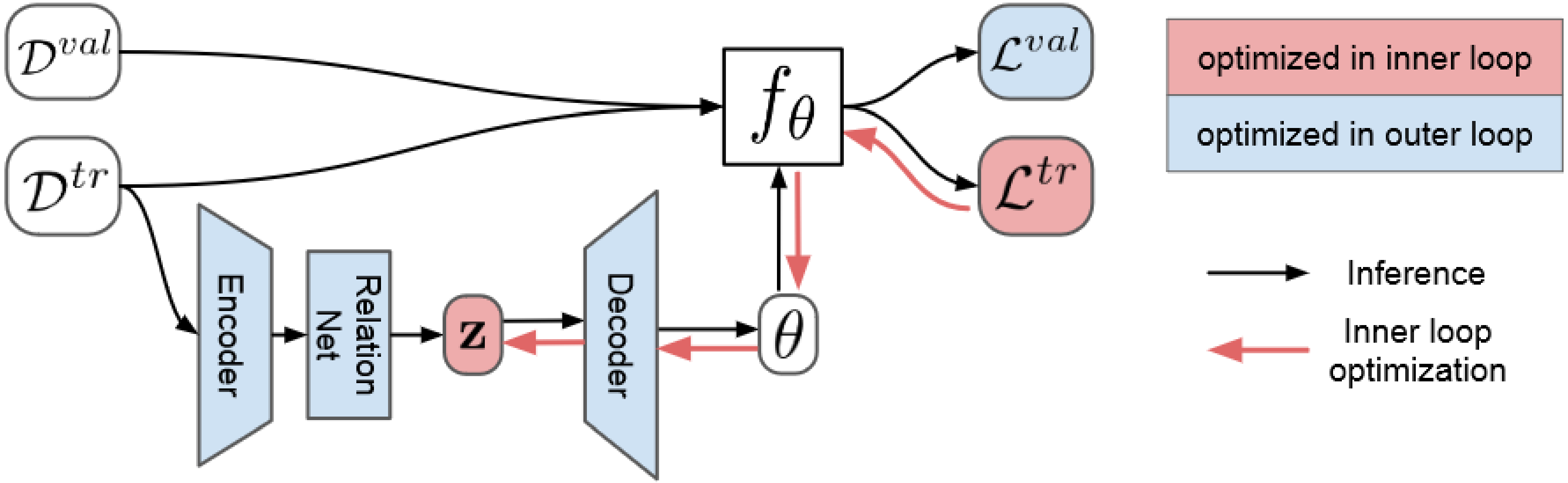
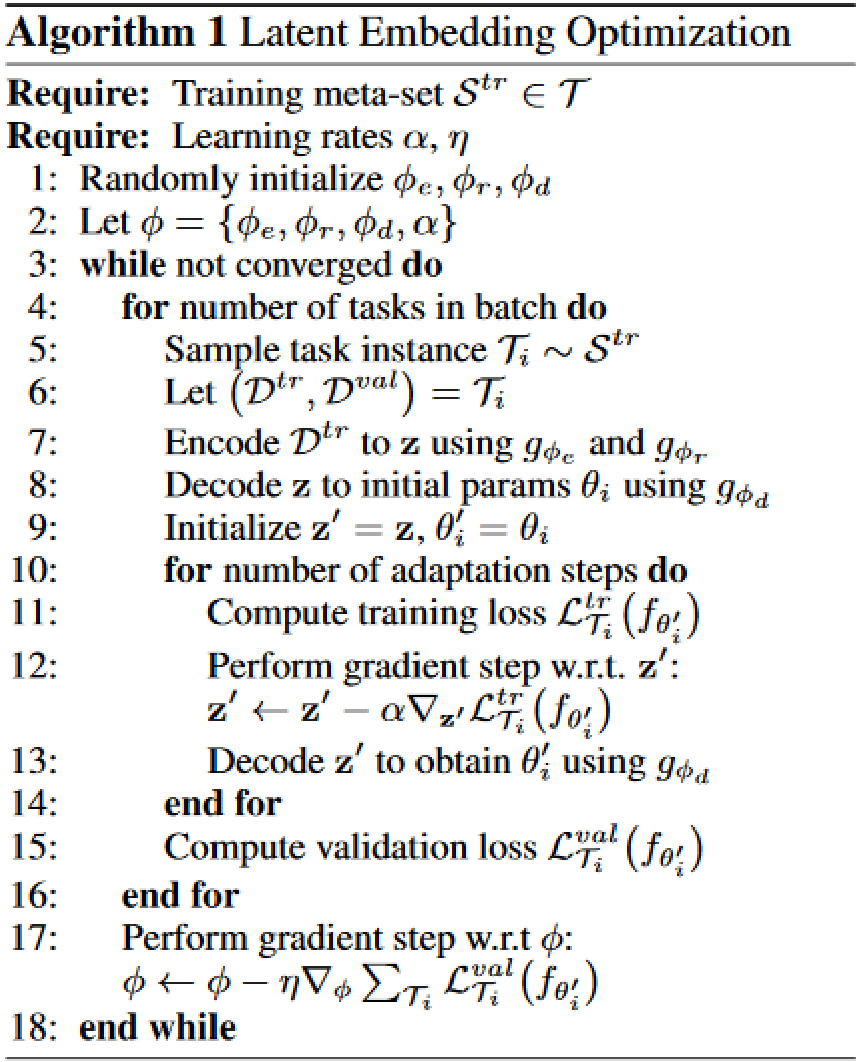


Figure 2: Overview of the architecture of LEO.

Meta-Learning With Latent Embedding Optimization



Algorithm 1 Latent Embedding Optimization

```
Require: Training meta-set $\mathcal{S}^{tr} \in \mathcal{T}$
Require: Learning rates $\alpha$, $\eta$
Randomly initialize $\phi_e, \phi_r, \phi_d$
Let $\phi = \{\phi_e, \phi_r, \phi_d, \alpha\}$
while not converged do
    for number of tasks in batch do
        Sample task instance $\mathcal{T}_i \sim \mathcal{S}^{tr}$
        Let $\left(\mathcal{D}^{tr}, \mathcal{D}^{val}\right) = \mathcal{T}_i$
        Encode $\mathcal{D}^{tr}$ to $\mathbf{z}$ using $g_{\phi_e}$ and $g_{\phi_r}$
        Decode $\mathbf{z}$ to initial params $\theta_i$ using $g_{\phi_d}$
        Initialize $\mathbf{z}' = \mathbf{z}$, $\theta'_i = \theta_i$
        for number of adaptation steps do
            Compute training loss $\mathcal{L}^{tr}_{\mathcal{T}_i}\left(f_{\theta'_i}\right)$
            Perform gradient step w.r.t. $\mathbf{z}'$:
            $\mathbf{z}' \leftarrow \mathbf{z}' - \alpha \nabla_{\mathbf{z}'} \mathcal{L}^{tr}_{\mathcal{T}_i}\left(f_{\theta'_i}\right)$
            Decode $\mathbf{z}'$ to obtain $\theta'_i$ using $g_{\phi_d}$
        end for
        Compute validation loss $\mathcal{L}^{val}_{\mathcal{T}_i}\left(f_{\theta'_i}\right)$
    end for
    Perform gradient step w.r.t $\phi$:
    $\phi \leftarrow \phi - \eta \nabla_\phi \sum_{\mathcal{T}_i} \mathcal{L}^{val}_{\mathcal{T}_i}\left(f_{\theta'_i}\right)$
end while
```

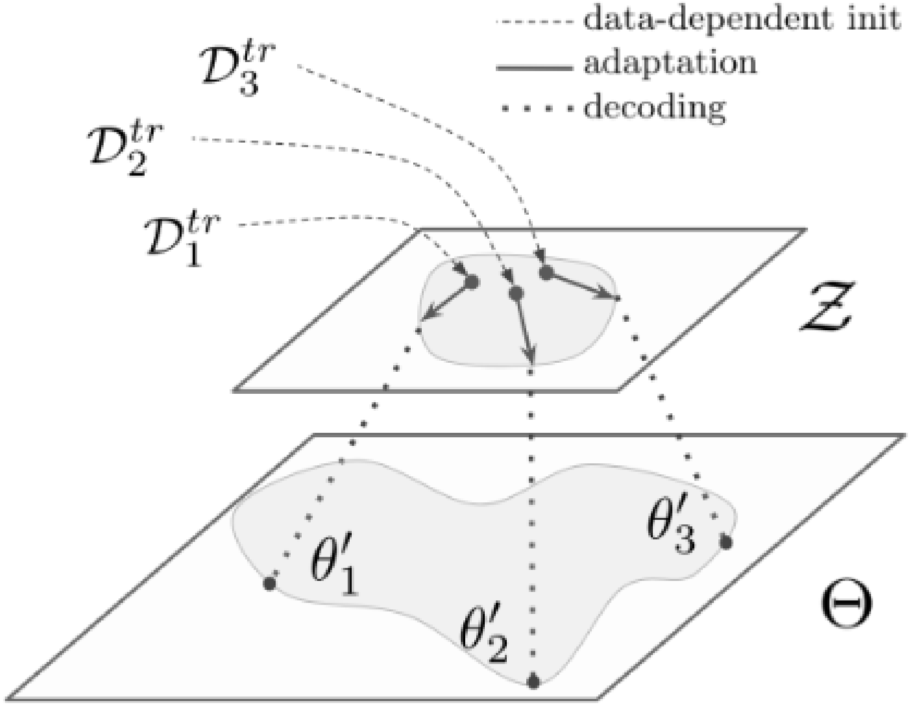


Figure 1: High-level intuition for LEO. While MAML operates directly in a high dimensional parameter space Θ, LEO performs meta-learning within a low-dimensional latent space $\mathcal{Z}$, from which the parameters are generated.

LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning

- This approach directly generates the functional weights of a network with limited training samples, which contains two key modules, namely, a TargetNet module(the base-level learner) and a MetaNet module(the meta-level learner).

LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning

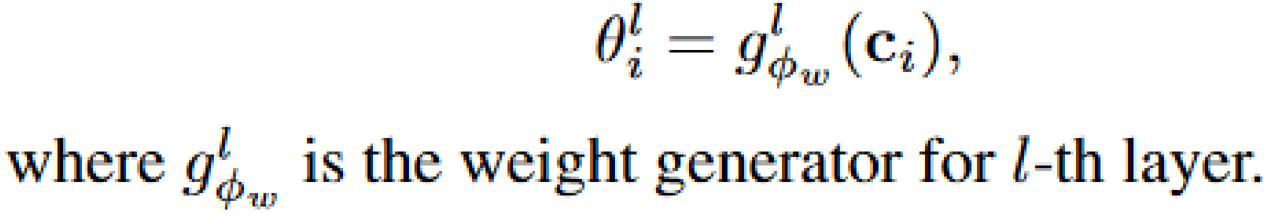
- The parameters in TargetNet are generated by the MetaNet module conditioned on training samples.
- MetaNet contains two parts, namely, a task context encoder and a weight generator.

$$\mu_i, \sigma_i = \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} g_{\phi_e}(x_i^{n,k}),$$

$$\mathbf{c}_i \sim q(\mathbf{c}_i | S_i^{train}) = \mathcal{N}\big(\mu_i, \mathrm{diag}({\sigma_i}^2)\big),$$

LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning

- weight generator



$$\theta_i^l = g_{\phi_w}^l(\mathbf{c}_i),$$

where $g_{\phi_w}^l$ is the weight generator for l-th layer.

LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning

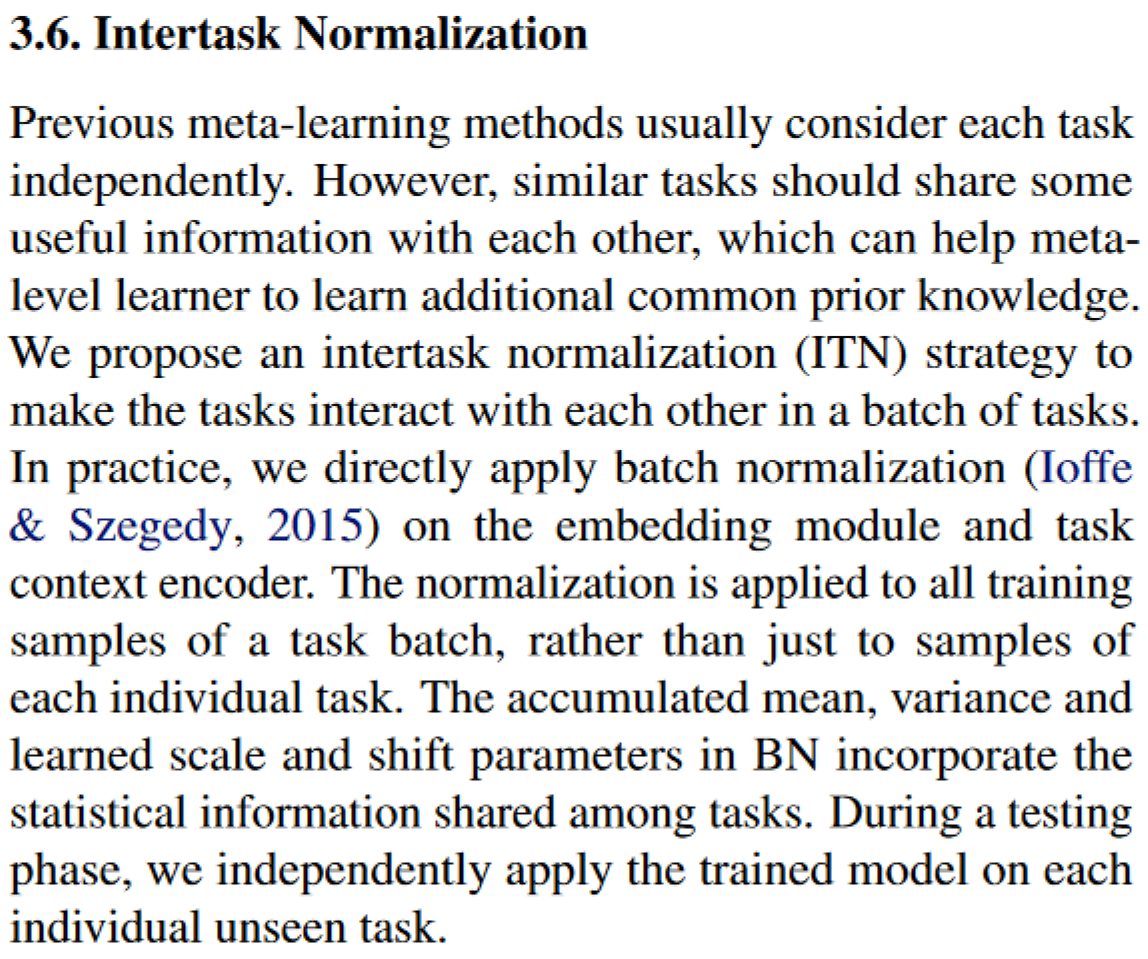
- TargetNet Module

$$a(\hat{x}, x_i) = \frac{e^{d(T_{\theta_i}(\hat{x}_i), T_{\theta_i}(x_i^{n,k}))}}{\sum_{n=1}^{N} \sum_{k=1}^{K} e^{d(T_{\theta_i}(\hat{x}_i), T_{\theta_i}(x_i^{n,k}))}},$$

$$\hat{\mathbf{p}}_i = \sum_{n=1}^{N} \sum_{k=1}^{K} a(\hat{x}_i, x_i^{n,k}) \mathbf{y}_i^{n,k}.$$

$$\mathcal{L}_{\mathcal{T}_i} = H(\hat{\mathbf{y}}_i, \hat{\mathbf{p}}_i).$$

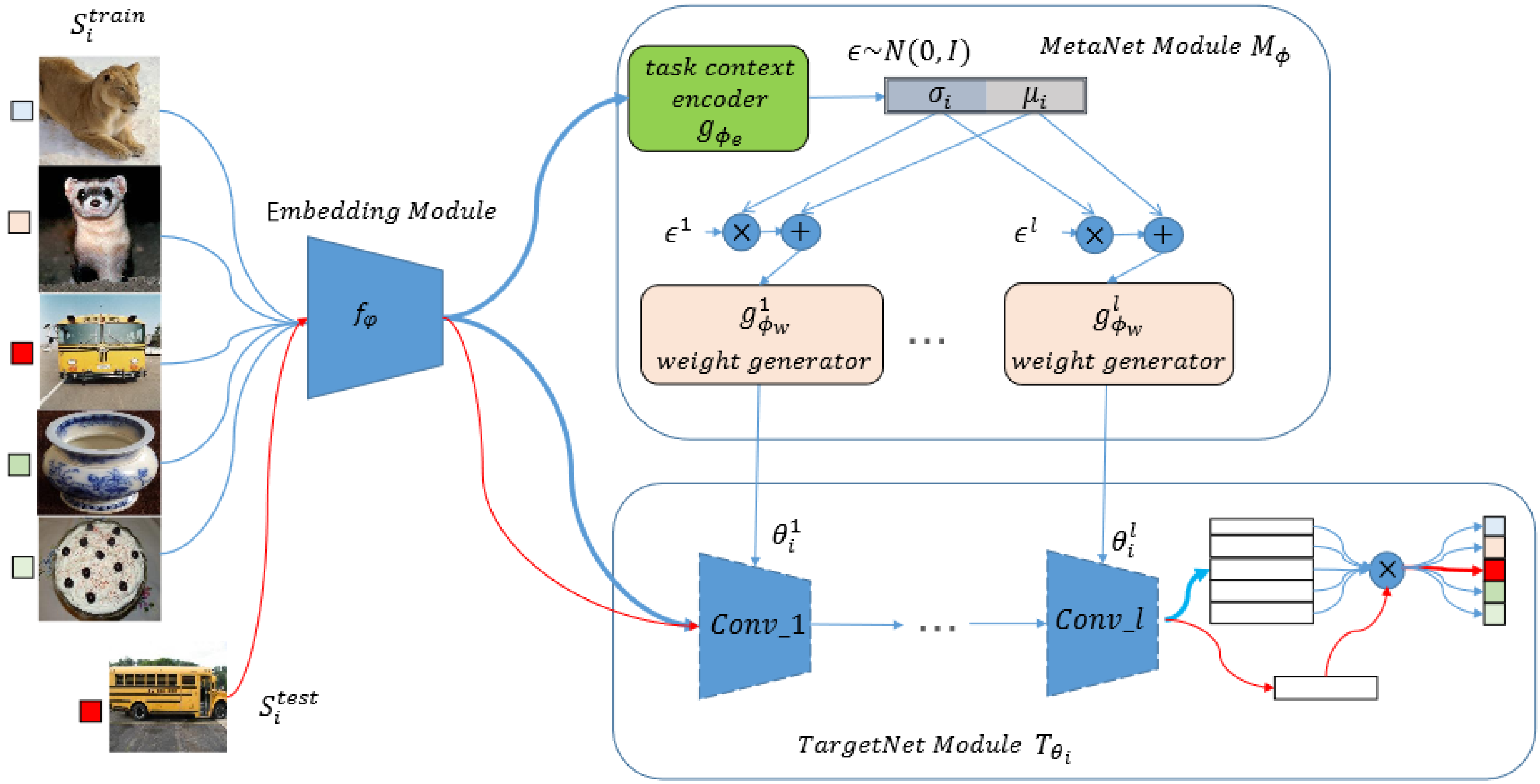
LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning



3.6. Intertask Normalization

Previous meta-learning methods usually consider each task independently. However, similar tasks should share some useful information with each other, which can help meta-level learner to learn additional common prior knowledge. We propose an intertask normalization (ITN) strategy to make the tasks interact with each other in a batch of tasks. In practice, we directly apply batch normalization (Ioffe & Szegedy, 2015) on the embedding module and task context encoder. The normalization is applied to all training samples of a task batch, rather than just to samples of each individual task. The accumulated mean, variance and learned scale and shift parameters in BN incorporate the statistical information shared among tasks. During a testing phase, we independently apply the trained model on each individual unseen task.

LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning



LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning

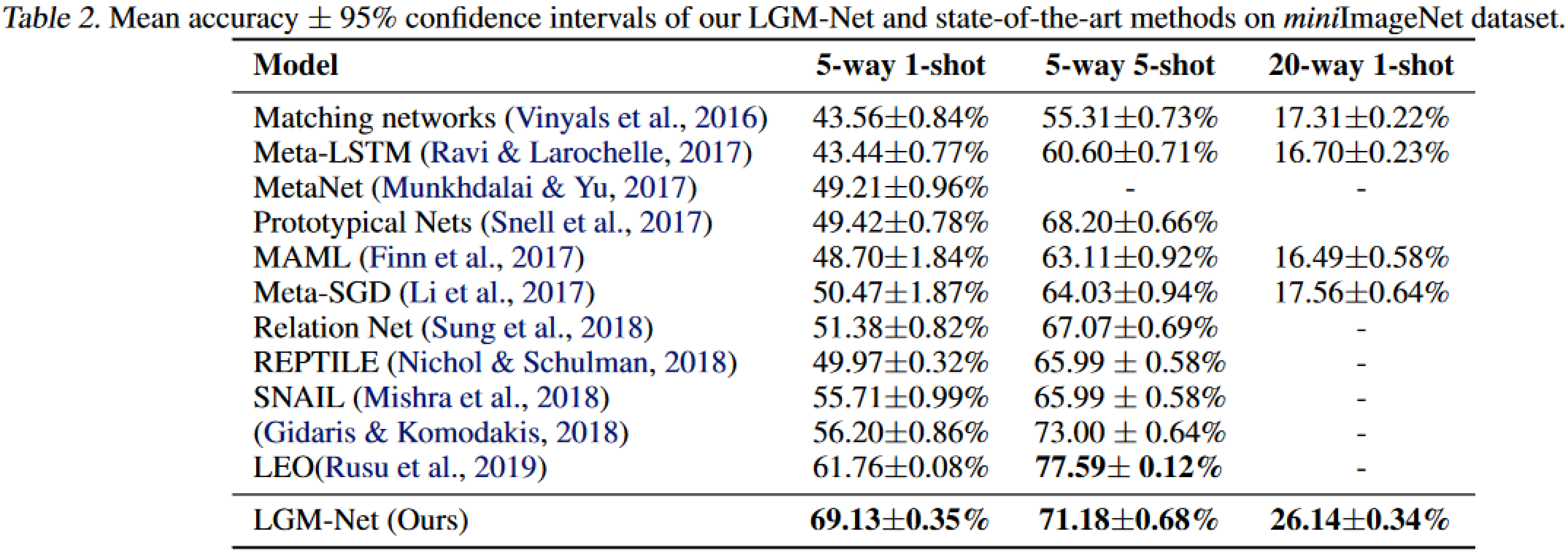


Table 2. Mean accuracy ± 95% confidence intervals of our LGM-Net and state-of-the-art methods on *mini*ImageNet dataset.

Model	**5-way 1-shot**	**5-way 5-shot**	**20-way 1-shot**
Matching networks (Vinyals et al., 2016)	43.56±0.84%	55.31±0.73%	17.31±0.22%
Meta-LSTM (Ravi & Larochelle, 2017)	43.44±0.77%	60.60±0.71%	16.70±0.23%
MetaNet (Munkhdalai & Yu, 2017)	49.21±0.96%	-	-
Prototypical Nets (Snell et al., 2017)	49.42±0.78%	68.20±0.66%	
MAML (Finn et al., 2017)	48.70±1.84%	63.11±0.92%	16.49±0.58%
Meta-SGD (Li et al., 2017)	50.47±1.87%	64.03±0.94%	17.56±0.64%
Relation Net (Sung et al., 2018)	51.38±0.82%	67.07±0.69%	-
REPTILE (Nichol & Schulman, 2018)	49.97±0.32%	65.99 ± 0.58%	-
SNAIL (Mishra et al., 2018)	55.71±0.99%	65.99 ± 0.58%	-
(Gidaris & Komodakis, 2018)	56.20±0.86%	73.00 ± 0.64%	-
LEO(Rusu et al., 2019)	61.76±0.08%	**77.59± 0.12%**	-
LGM-Net (Ours)	**69.13±0.35%**	**71.18±0.68%**	**26.14±0.34%**