Supplementary Material for "Data-Efficient and Robust Task Selection for Meta-Learning"

A. Introduction to Meta-Learning Algorithms

Gradient-based meta-learning The goal of gradient-based meta-learning is to learn initial parameters θ^* such that taking one (or a few) gradient steps on the support set $\mathcal{D}^s$ leads to a model that performs well on task $\mathcal{T}$. Consider model-agnostic meta-learning (MAML) [6] with base model f as an illustrative example. During the meta-training stage, the performance of the adapted model f_ϕ (i.e., $\phi = \theta - \eta \nabla_\theta \mathcal{L}\left(f_\theta; \mathcal{D}^s\right)$, η denotes the inner-loop learning rate) is evaluated on the corresponding query set $\mathcal{D}^q$ and is used to optimize the model parameter θ. Formally, the bi-level optimization process with expected risk is formulated as:

$$\theta^* \leftarrow \arg\min_\theta \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}\left[\mathcal{L}\left(f_\phi; \mathcal{D}^q\right)\right].$$

During the meta-testing stage, for task $\mathcal{T}_t$, the adapted parameter ϕ_t is found by fine-tuning meta-model θ_t on the support set $\mathcal{D}_t^s$. The almost no inner loop (ANIL) algorithm [33] simplifies the inner loop computation by only updating the classification head during meta-training task adaptation while keeping the remainder frozen. ANIL achieves a comparable performance with MAML with lower computational cost.

Metric-based meta-learning The aim of metric-based meta-learning is to conduct a non-parametric learner on top of meta-learned embedding space. Taking prototypical network (ProtoNet) with base model f_θ as an example [36], for each task $\mathcal{T}$, we first compute a class prototype representation $\{\mathbf{c}_r\}_{r=1}^R$ as the representation vector of the support samples belonging to class k as $\mathbf{c}_r = \frac{1}{N_r}\sum_{\left(\mathbf{x}_{k;r}^s, \mathbf{y}_{k;r}^s\right)\in\mathcal{D}_r^s} f_\theta^{PN}\left(\mathbf{x}_{k;r}^s\right)$, where $\mathcal{D}_r^s$ represents the subset of support samples labeled as class r, $\left(\mathbf{x}_{k;r}^s, \mathbf{y}_{k;r}^s\right)$ denotes the data with corresponding label in $\mathcal{D}_r^s$, and the size of this subset is N_r. Then, given a query data sample $\mathbf{x}_k^q$ in the query set, the probability of assigning it to the r-th class is measured by the distance d between its representation $f_\theta\left(\mathbf{x}_k^q\right)$ and prototype representation $\mathbf{c}_r$, and the cross-entropy loss of ProtoNet is formulated as:

$$\mathcal{L} = \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}\left[-\sum_{k,r}\log\frac{\exp\left(-d\left(f_\theta\left(\mathbf{x}_k^q\right), \mathbf{c}_r\right)\right)}{\sum_{r'}\exp\left(-d\left(f_\theta\left(\mathbf{x}_k^q\right), \mathbf{c}_{r'}\right)\right)}\right]$$

At the meta-testing stage, the predicted label of each query sample is assigned to the class with maximal probability, i.e., $\hat{\mathbf{y}}_k^q = \arg\max_r p\left(\mathbf{y}_k^q = r \mid \mathbf{x}_k^q\right)$.

B. Efficient Gradient Estimation

The updating process of the meta-model with explicit task gradient $\nabla_\theta \mathcal{L}\left(f_{\phi_j}; \mathcal{D}_j^q\right)$ is time-consuming and incurs a large computational cost. As shown in [12], the variation of the gradient norm is mainly captured by the gradient of the loss function with respect to the pre-activation outputs of the last layer. Therefore, for a few-shot classification task, the above estimation only requires a forward computation on the last layer. E.g., for a softmax layer as the last, the gradients of the loss with respect to the input of the softmax layer for $\left(\mathbf{x}_{i,j}^q, \mathbf{y}_{i,j}^q\right)$ is l_i-y_i, where l_i is the logits and y_i is the encoded label. In this section, we elaborate on the details of our efficient gradient estimation as mentioned in sec. 4.2. We extend the result of [32] to estimate the task-gradient $\nabla_\theta \mathcal{L}\left(f_{\phi_i}; \mathcal{D}_i^q\right)$ and denote it as $\tilde{g}_i$.

Generally, we follow the notation of [12, 32] to establish our analysis upon the estimated gradient. Suppose in a L-layer multilayer perceptron network, $\theta^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$ denotes the weight matrix for layer l with M_l hidden units and $\sigma^{(l)}(\cdot)$ be a Lipschitz continuous activation function. Then, for datapoint (x_i, y_i), let

$$\begin{aligned} x_i^{(0)} &= x_i \\ z_i^{(l)} &= \theta^{(l)} x_i^{(l-1)} \\ x_i^{(l)} &= \sigma^{(l)}\left(z_i^{(l)}\right) \\ f_\theta(x) &= x^{(L)} \end{aligned}$$

where $x_i^{(l)}$ denotes the output after the l-th layer of x_i ($l = 1, \ldots, L$). The gradient of the loss w.r.t. the output of the network is shown to be

$$\nabla_{\theta^{(L)}}^{(i)} \mathcal{L} = \nabla_{\theta^{(L)}} \mathcal{L} \left(f_\theta \left(x_i \right), y_i \right)$$

and the gradient of the loss w.r.t. the output of layer l is denoted as

$$\nabla_{\theta^{(l)}}^{(i)} \mathcal{L} = \Delta_i^{(l)} \Sigma_L' \left(z_i^{(L)} \right) \nabla_{\theta^{(L)}}^{(i)} \mathcal{L}$$

where

$$\begin{aligned} \Delta_i^{(l)} &= \Sigma_l' \left(z_i^{(l)} \right) \theta_{l+1}^T \ldots \Sigma_{L-1}' \left(z_i^{(L-1)} \right) \theta_L^T \\ \Sigma_l'(z) &= \operatorname{diag} \left(\sigma'^{(l)} \left(z_1 \right), \ldots, \sigma'^{(l)} \left(z_{M_l} \right) \right) \end{aligned}$$

Thus, datapoint i's gradient w.r.t. the parameters of the l-th layer $\theta^{(l)}$ can be written as

$$\nabla_{\theta^{(l)}} \mathcal{L} \left(f_\theta \left(x_i \right), y_i \right) = \left(\Delta_i^{(l)} \Sigma_L' \left(z_i^{(L)} \right) \nabla_{\theta^{(L)}}^{(i)} \mathcal{L} \right) \left(x_i^{(l-1)} \right)^T.$$

For a query set $\mathcal{D}_i^q$ of arbitrary task $\mathcal{T}_i$, the gradient of meta-model f_θ on $\mathcal{D}_i^q$ w.r.t. the l-th layer $\theta^{(l)}$ is

$$\nabla_{\theta^{(l)}} \mathcal{L}(f_\theta; \mathcal{D}_i^q) = \sum_k \nabla_{\theta^{(l)}} \mathcal{L} \left(f_\theta \left(x_{ik} \right), y_{ik} \right)$$

where x_{ik} and y_{ik} are the data and corresponding labels within query set $\mathcal{D}_i^q$.

Specifically, following previous work, we use the below gradient estimation $\tilde{g}_i$ to approximate the gradient:

$$\tilde{g}_i = \sum_k \Sigma_L' \left(z_{ik}^{(L)} \right) \nabla_{\theta^{(L)}}^{(ik)} \mathcal{L},$$

where L denotes the last layer.

Next, in **Proposition 1** we show how to efficiently bind the task gradient with the adapted model via the gradient of loss w.r.t. the input of the last layer.

Proposition 1 (Gradient Norm Upper Bound)**.** *Suppose the loss function is β-smooth, the norm of difference of task-specific meta-gradients $\nabla_\theta \mathcal{L} \left(f_{\phi_i}; \mathcal{D}_i^q \right)$ and $\nabla_\theta \mathcal{L} \left(f_{\phi_j}; \mathcal{D}_j^q \right)$ can be upper bounded by a constant C_1 times the norm of difference of $\tilde{g}_i$ and $\tilde{g}_j$ (gradients of the last layer of meta-model θ) with adding another constant C_2, i.e.,*

$$\left\| \nabla_\theta \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q) - \nabla_\theta \mathcal{L}(f_{\phi_j}; \mathcal{D}_j^q) \right\| \leq C_1 \left\| \tilde{g}_i - \tilde{g}_j \right\| + C_2.$$

Consider query sets $\mathcal{D}_i^q$ (with the same size) and $\mathcal{D}_j^q$ for two different tasks $\mathcal{T}_i$ and $\mathcal{T}_j$, we have

$$
\begin{aligned}
\|\nabla_{\theta^{(l)}}\ \mathcal{L}(f_\theta;\mathcal{D}_i^q) - \nabla_{\theta^{(l)}}\mathcal{L}(f_\theta;\mathcal{D}_j^q)\| &= \left\|\sum_k \nabla_{\theta^{(l)}}\mathcal{L}\left(f_\theta\left(x_{ik}\right),y_{ik}\right) - \sum_k \nabla_{\theta^{(l)}}\mathcal{L}\left(f_\theta\left(x_{jk}\right),y_{jk}\right)\right\| \\
&= \left\|\sum_k \left(\Delta_{ik}^{(l)}\Sigma_L'\left(z_{ik}^{(L)}\right)\nabla_{\theta^{(L)}}^{(ik)}\mathcal{L}\right)\left(x_{ik}^{(l-1)}\right)^T - \sum_k \left(\Delta_{jk}^{(l)}\Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right)\left(x_{jk}^{(l-1)}\right)^T\right\| \\
&\le \sum_k \left\{\left\|\Delta_{ik}^{(l)}\right\|\cdot\left\|x_{ik}^{(l-1)T}\right\|\cdot\left\|\Sigma_L'\left(z_{ik}^{(L)}\right)\nabla_{\theta^{(L)}}^{(ik)}\mathcal{L} - \Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right\|\right. \\
&\quad \left.+\left\|\Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right\|\cdot\left\|\Delta_{ik}^{(l)}\left(x_{ik}^{(l-1)}\right)^T - \Delta_{jk}^{(l)}\left(x_{jk}^{(l-1)}\right)^T\right\|\right\} \\
&\le \sum_k \left\{\left\|\Delta_{ik}^{(l)}\right\|\cdot\left\|x_{ik}^{(l-1)T}\right\|\cdot\left\|\Sigma_L'\left(z_{ik}^{(L)}\right)\nabla_{\theta^{(L)}}^{(ik)}\mathcal{L} - \Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right\|\right. \\
&\quad \left.+\left\|\Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right\|\cdot\left(\left\|\Delta_{ik}^{(l)}\right\|\cdot\left\|\left(x_{ik}^{(l-1)}\right)^T\right\| + \left\|\Delta_{jk}^{(l)}\right\|\cdot\left\|\left(x_{jk}^{(l-1)}\right)^T\right\|\right)\right\} \quad \text{By Triangle Inequality} \\
&\le \sum_k \left\{\max_{l,k}\left(\left\|\Delta_{ik}^{(l)}\right\|\cdot\left\|x_{ik}^{(l-1)T}\right\|\right)\cdot\left\|\Sigma_L'\left(z_{ik}^{(L)}\right)\nabla_{\theta^{(L)}}^{(ik)}\mathcal{L} - \Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right\|\right. \\
&\quad \left.+\left\|\Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right\|\cdot\max_{l,i,j}\left(\left\|\Delta_i^{(l)}\right\|\cdot\left\|x_i^{(l-1)}\right\| + \left\|\Delta_j^{(l)}\right\|\cdot\left\|x_j^{(l-1)}\right\|\right)\right\} \quad \text{Take Maximum over } l,k,i,j \\
&= \underbrace{\max_{l,k}\left(\left\|\Delta_{ik}^{(l)}\right\|\cdot\left\|x_{ik}^{(l-1)T}\right\|\right)}_{c_1}\cdot\sum_k\left\|\Sigma_L'\left(z_{ik}^{(L)}\right)\nabla_{\theta^{(L)}}^{(ik)}\mathcal{L} - \Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right\| \\
&\quad + \underbrace{\sum_k\left\|\Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right\|\cdot\max_{l,i,j}\left(\left\|\Delta_i^{(l)}\right\|\cdot\left\|x_i^{(l-1)}\right\| + \left\|\Delta_j^{(l)}\right\|\cdot\left\|x_j^{(l-1)}\right\|\right)}_{c_2}
\end{aligned} \tag{6}
$$

Thus, we derive the gradient of meta-model f_θ on $\mathcal{D}_i^q$ w.r.t. the l-th layer $\theta^{(l)}$ can be bounded by the gradient of the loss w.r.t. the pre-activation outputs. c_1 and c_2 will be used for further derivation.

According to (6), we can show that two arbitrary query sets' gradient of meta-model can be bounded by constant times the gradient of the loss w.r.t. the pre-activation outputs of the neural network as

$$
\left\|\nabla_\theta\mathcal{L}(f_\theta;\mathcal{D}_i^q) - \nabla_\theta\mathcal{L}(f_\theta;\mathcal{D}_j^q)\right\| \le L\cdot c_1\left\|\underbrace{\sum_k\left(\Sigma_L'\left(z_{ik}^{(L)}\right)\nabla_{\theta^{(L)}}^{(ik)}\mathcal{L}\right)}_{\tilde{g}_i} - \underbrace{\sum_k\left(\Sigma_L'\left(z_{jk}^{(L)}\right)\nabla_{\theta^{(L)}}^{(jk)}\mathcal{L}\right)}_{\tilde{g}_j}\right\| + L\cdot c_2 \tag{7}
$$

Due to the bi-level optimization structure of meta-learning, the intrinsic gradient for outer loop meta-model updating is $\nabla_\theta\mathcal{L}(f_{\phi_i};\mathcal{D}_i^q)$ for task $\mathcal{T}_i$. Suppose the loss function $\mathcal{L}$ is β-smooth, the norm of the outer loop gradient difference $\nabla_\theta\mathcal{L}(f_{\phi_i};\mathcal{D}_i^q) - \nabla_\theta\mathcal{L}(f_{\phi_j};\mathcal{D}_j^q)$ can be bounded based on the result of (7):

$$
\begin{aligned}
&\left\|\nabla_\theta\mathcal{L}(f_{\phi_i};\mathcal{D}_i^q) - \nabla_\theta\mathcal{L}(f_{\phi_j};\mathcal{D}_j^q)\right\| \\
&= \left\|\nabla_\theta\mathcal{L}(f_{\phi_i};\mathcal{D}_i^q) - \nabla_\theta\mathcal{L}(f_\theta;\mathcal{D}_i^q) + \nabla_\theta\mathcal{L}(f_\theta;\mathcal{D}_i^q) - \nabla_\theta\mathcal{L}(f_{\phi_j},\mathcal{D}_j^q) + \nabla_\theta\mathcal{L}(f_\theta;\mathcal{D}_j^q) - \nabla_\theta\mathcal{L}(f_\theta;\mathcal{D}_j^q)\right\| \\
&\le \|\nabla_\theta\mathcal{L}(f_{\phi_i};\mathcal{D}_i^q) - \nabla_\theta\mathcal{L}(f_\theta;\mathcal{D}_i^q)\| + \left\|\nabla_\theta\mathcal{L}(f_{\phi_j};\mathcal{D}_j^q) - \nabla_\theta\mathcal{L}(f_\theta;\mathcal{D}_j^q)\right\| + \left\|\nabla_\theta\mathcal{L}(f;\mathcal{D}_i^q) - \nabla_\theta\mathcal{L}(f_\theta;\mathcal{D}_j^q)\right\| \\
&= \underbrace{L\cdot c_1}_{C_1}\|\tilde{g}_i - \tilde{g}_j\| + \underbrace{L\cdot c_2 + \beta\cdot(\|\theta-\phi_i\| + \|\theta-\phi_j\|)}_{C_2}
\end{aligned} \tag{8}
$$

C. Proof of Theorem 1

We follow the high-level idea of [32] and adapt the analysis to the bi-level updating setting of meta-learning.

The updated formula:

$$\theta^{t+1} = \theta^t - \eta \nabla \sum_{i \in S} \mathcal{L}(f_{\phi_i}; \mathcal{D}^q)$$
$$\text{where} \quad \phi_i = \theta^t - \eta' \nabla \sum_{(x,y) \in \mathcal{D}_i^s} \mathcal{L}(f_\theta; \mathcal{D}_i^s)$$

First, consider the task gradient:

$$\begin{aligned}
\nabla_\theta \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q) &= \nabla_\theta \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q) \\
&= \nabla_\theta \phi_i \nabla_{\phi_i} \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q) \\
&= \nabla_\theta (\theta^t - \eta' \nabla_\theta \sum_{(x_i, y_i) \in \mathcal{D}_i^s} \mathcal{L}(f_\theta; (x_i^s, y_i^s)) \nabla_{\phi_i} \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q) \\
&= (-\eta' \sum_{(x_i, y_i) \in \mathcal{D}_i^s} \nabla_\theta^2 \mathcal{L}(f_\theta; (x_i^s, y_i^s)) \nabla_{\phi_i} \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q)
\end{aligned} \tag{9}$$

Similarly, by the β-smoothness (**Assumption 1**) of loss function $\mathcal{L}(f, \mathcal{D})$ and $\mu - PL^\star$ condition (**Assumption 2**), we can get:

$$\begin{aligned}
\mathcal{L}(\theta^{t+1}; \mathcal{D}) - \mathcal{L}(\theta^t; \mathcal{D}) &\leq -\frac{\eta}{2} \|\nabla_\theta \sum_{i \in \mathcal{S}} \gamma_j \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q)\|^2 \\
&\leq -\frac{\eta}{2} (\|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_j}; \mathcal{D}_j^q)\| - \epsilon)^2 \quad \text{Substitute weighted subset gradient as full gradient and } \epsilon \\
&= -\frac{\eta}{2} (\|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_j}; \mathcal{D}_j^q)\|^2 - 2\epsilon \|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_j}; \mathcal{D}_j^q)\| + \epsilon^2) \\
&= -\frac{\eta}{2} (\| - \eta' \sum_{(x_j, y_j) \in \mathcal{D}_j^s} \nabla_\theta^2 \mathcal{L}(f_{\phi_j}; (x_j^s, y_j^s)) \sum_{j \in \mathcal{M}} \nabla_{\phi_j} \mathcal{L}(f_{\phi_j}; \mathcal{D}_j^q)\|^2 - 2\epsilon \|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_j}; \mathcal{D}_j^q)\| + \epsilon^2) \quad \text{by (9)} \\
&\leq -\frac{\eta\eta' m}{2} \| \sum_{j \in \mathcal{M}} \nabla_{\phi_j} \mathcal{L}(f_{\phi_j}; \mathcal{D}_j^q)\|^2 + \eta\epsilon \|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_j}; \mathcal{D}_j^q)\| - \frac{\eta}{2}\epsilon^2 \quad \text{Bounded Hessian} \\
&\leq -\eta\eta' m\mu \mathcal{L}(f_\phi; \mathcal{D}^q) + \eta\epsilon \|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q)\| - \frac{\eta}{2}\epsilon^2 \quad \text{By } \mu - PL^\star \text{ Condition} \\
&= -\eta\eta' m\mu (\mathcal{L}(f_{\phi_i}; \mathcal{D}) - \mathcal{L}(f_{\theta^t}; \mathcal{D})) - \eta\eta' m\mu \mathcal{L}(f_{\theta^t}; \mathcal{D}) + \eta\epsilon \|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q)\| - \frac{\eta}{2}\epsilon^2 \\
&\leq -\eta\eta' m\mu \mathcal{L}(f_{\theta^t}; \mathcal{D}) + \eta\epsilon \|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q)\| - \frac{\eta}{2}\epsilon^2 + \eta\eta' m\mu |\mathcal{L}(f_{\phi*}; \mathcal{D}) - \mathcal{L}(f_{\theta^0}; \mathcal{D})|
\end{aligned} \tag{10}$$

Let $r = |\mathcal{L}(f_{\phi*}; \mathcal{D}) - \mathcal{L}(f_{\theta^0}, \mathcal{D})|$, we obtain:

$$\mathcal{L}(f_{\theta^{t+1}}; \mathcal{D}) \leq (1 - \eta\eta' m\mu) \mathcal{L}(f_{\theta^t}; \mathcal{D}) + \eta\epsilon \|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q)\| - \frac{\eta}{2}\epsilon^2 + \eta\eta' m\mu r.$$

This implies:

$$\begin{aligned}
\mathcal{L}(f_{\theta^t}; \mathcal{D}) &\leq (1 - \eta\eta' m\mu)^t \mathcal{L}(f_{\theta^0}; \mathcal{D}) + \sum_{k=0}^{t-1} (1 - \eta\eta' m\mu)^k (\eta\epsilon \|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q)\| - \frac{\eta}{2}\epsilon^2 + \eta\eta' m\mu r) \\
&\leq (1 - \eta\eta' m\mu)^t \mathcal{L}(f_{\theta^0}; \mathcal{D}) + \frac{1}{\eta\eta' m\mu} (\eta\epsilon \|\nabla_\theta \sum_{j \in \mathcal{M}} \mathcal{L}(f_{\phi_i}; \mathcal{D}_i^q)\| - \frac{\eta}{2}\epsilon^2 + \eta\eta' m\mu r),
\end{aligned}$$

giving:

$$\mathbb{E}[(\mathcal{L}(f_{\theta^t};\mathcal{D}))] \leq (1-\eta\eta' m\mu)^t \mathbb{E}[(\mathcal{L}(f_{\theta^0};\mathcal{D}))] + \frac{1}{\eta\eta' m\mu}(\eta\epsilon \|\mathbb{E}[\nabla_\theta \mathcal{L}(f_\phi;\mathcal{D})]\|_{L^\infty_{\mathcal{M}}} - \frac{\eta}{2}\epsilon^2) + r$$

where

$$\|\mathbb{E}[\nabla_\theta \mathcal{L}(f_\phi;\mathcal{D})]\|_{L^\infty_{\mathcal{M}}} = \sup_{\Gamma} \left\| \mathbb{E}\left[\sum_{j\in\mathcal{M}} \left(\nabla_\theta \mathcal{L}\left(f_{\phi_j};\mathcal{D}_j^q\right) - \nabla_\theta \mathcal{L}\left(f_{\phi_{\Gamma(j)}};\mathcal{D}_{\Gamma(j)}^q\right)\right)\right] \right\|.$$

D. Implementation Details of Noisy Task Setting

We provide the detailed noisy task-generating mechanism as follows.

Algorithm 2 Noisy Task Generating Mechanism (5-way 5-shot)

Require: Task $\mathcal{T}$; generating rate λ ; noise threshold t

Draw 5 samples $(\lambda_1, \cdots, \lambda_5) \sim Poisson(\lambda)$
for λ_i **do**
  if $\lambda_i \geq t$ **then**
    λ_i = t
  end if
end for
for class i **do**
  Randomly draw λ_i samples $(c_{i,1}, \cdots, c_{i,\lambda_i})$ from other 4 classes
  for $c_{i,j}$ **do**
    Randomly draw data $x_{c_{i,j},k}$ from class $c_{i,j}$ and data $x_{i,k}$ from class i
    Switch the label of two selected data
  end for
end for
Randomly split task $\mathcal{T}_i$ into support set and query set
Output the constructed noisy task

Algorithm 2 is based on symmetric label swap for few-shot learning [18]. By the above label noise-generating mechanism, the mislabeled data could exist in both the support set and the query set. The noise ratio is controlled by noise threshold t and $Poisson(\lambda)$ in **Algorithm 2**.

E. Additional Experiment

E.1. ResNet-12 as Large Backbone

To show DERTS works for a larger backbone, we explored the performance of ANIL and PN with ResNet-12 [30] in both limited data budget and noisy label task (noise ratio 40%) settings on Mini-Imagenet. We keep the ResNet-12 configuration details the same as CNN4.

From **Table 6** and **7**, we observe that DERTS generally holds the advantage of data efficiency and robustness for both settings when shifting the backbone to ResNet-12. In the limited data budget setting, DERTS shows its faster learning capability towards baselines. In the noisy label task setting, DERTS for ANIL outperforms baseline by at least 3% on accuracy, which significantly shows DERTS is effective for larger backbones. One thing worth mentioning here is that ANIL-US and PN-US perform comparably better on ResNet12 in the limited data budget setting than CNN4. We speculate the stronger representation capability of ResNet12 empowers ANIL-US and PN-US with a better ability to be aware of the difficulty of tasks, but it is still not robust in the noisy task setting compared to other methods.

E.2. Additional Experiment on Limited Data budget

In this subsection, we provide additional experiments and details for limited data budget setting. In the main context, we present the experiment results on only training 16 classes (25% classes and 25% data). Here, we provide experiment results for training 32 classes (50% classes and 50% data) in **Table 8**. According to the experiment results, DERTS outperforms all the

Dataset	Mini-ImageNet (5-way 5-shot)		
Iterations	10%	30%	All
ANIL	52.33 ± 0.7	62.75 ± 0.8	68.01 ± 0.7
ANIL-US	56.08 ± 0.7	58.92 ± 0.7	$\mathbf{68.34 \pm 0.6}$
ANIL-ATS	51.20 ± 0.6	60.62 ± 0.8	67.25 ± 0.8
ANIL-DERTS	$\mathbf{56.24 \pm 0.7}$	$\mathbf{67.19 \pm 0.7}$	68.23 ± 0.6
PN	59.07 ± 0.8	61.13 ± 0.8	67.56 ± 0.6
PN-US	58.35 ± 0.8	60.91 ± 0.9	67.72 ± 0.7
PN-DERTS	$\mathbf{60.17 \pm 0.9}$	$\mathbf{63.45 \pm 0.8}$	$\mathbf{67.81 \pm 0.7}$

Table 6. Average accuracy of 5-way 5-shot Mini-ImageNet Classification with Limited Data Budget Setting (ResNet-12 as the backbone). 10% (30%) in the table denotes the performance after learning on 10% (30%) tasks during the episodic training.

Dataset	Mini-ImageNet (5-way 5-shot)
Noise Ratio	40%
ANIL	62.60 ± 0.59
ANIL-US	55.16 ± 0.88
ANIL-ATS	60.17 ± 0.77
ANIL-DERTS	$\mathbf{64.87 \pm 0.72}$
PN	54.15 ± 0.68
PN-US	53.93 ± 0.70
PN-DERTS	$\mathbf{55.43 \pm 0.65}$

Table 7. Average accuracy of 5-way 5-shot Mini-ImageNe Classification with Noisy Task Setting (ResNet-12 as the backbone). 40% in the table denote the noise ratio (percentage of mislabeled data).

Dataset	Mini-ImageNet (32-Class)	
Method	5-way 5-shot	5-way 1-shot
ANIL	54.86 ± 0.72	42.85 ± 0.71
ANIL-US	55.07 ± 0.70	42.72 ± 0.60
ANIL-ATS	54.61 ± 0.68	42.55 ± 0.64
ANIL-DERTS	$\mathbf{56.10 \pm 0.64}$	$\mathbf{43.97 \pm 0.58}$
ProtoNet	59.29 ± 0.69	43.17 ± 0.61
ProtoNet-US	59.21 ± 0.61	42.75 ± 0.72
ProtoNet-DERTS	$\mathbf{60.17 \pm 0.68}$	$\mathbf{44.20 \pm 0.71}$

Table 8. 5-way 5-shot / 5-way 1-shot Mini-ImageNet Classification with 50% Class Training Set.

baselines with an average of 1.2% in accuracy, which further indicates that DERTS captures the task diversity in this setting with fewer training classes.

According to [46], the details of selected training classes are as follows. For 25% training classes (16 classes), we select: {n02823428, n13133613, n04067472, n03476684, n02795169, n04435653, n03998194, n02457408, n03220513, n03207743, n04596742, n03527444, n01532829, n02687172, n03017168, n04251144}.

In addition, the selected classes for 50% training classes are: {n03676483, n13054560, n04596742, n01843383, n02091831, n03924679, n01558993, n01910747, n01704323, n01532829, n03047690, n04604644, n02108089, n02747177, n02111277, n01749939, n03476684, n04389033, n07697537, n02105505, n02113712, n03527444, n03347037, n02165456, n02120079, n04067472, n02687172, n03998194, n03062245, n07747607, n09246464, n03838899 }.

E.3. Case Study for Task Selection

We provide a brief case study for further analysis of the tasks selected and not selected by DERTS. **Figure 3** displays typical examples of both selected and unselected tasks. From the two unselected tasks presented, we observe that these tasks are generally coarse-grained. The classes within the unselected tasks are easily distinguishable, indicating that they might be relatively simpler. In contrast, the classes and images in the selected tasks tend to be more visually confusing. Task 3 includes

three different classes of dogs, making this task more fine-grained than some of the unselected tasks. Task 4 consists of two visually similar pairs: the Seal and Diver pair, which might share the same background, and the Pot and Soup pair, which could have similar shapes and colors. The selected subset of tasks likely offers better diversity and is more challenging to learn, making them more informative for the meta-training process. This observation aligns with the claim made by related works on task sampling [2, 19, 46].

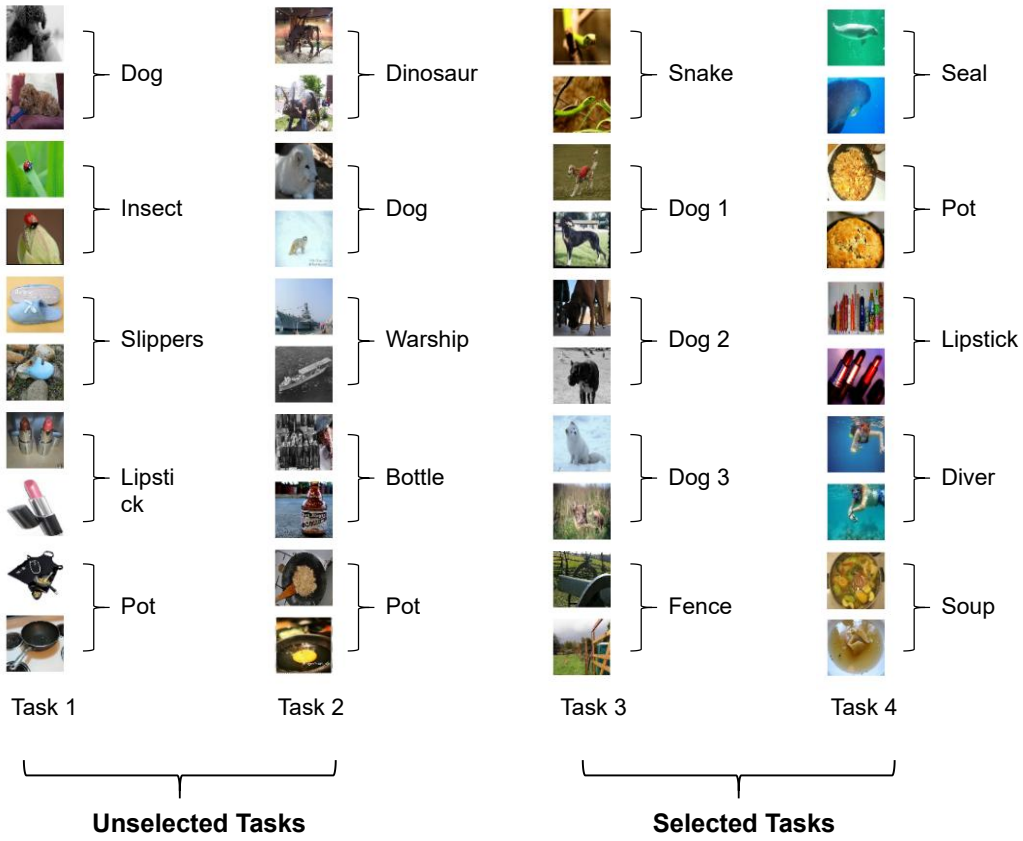

Figure 3. Typical examples of selected tasks by DERTS and unselected tasks.