# AUTOMATED RELATIONAL META-LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In order to efficiently learn with small amount of data on new tasks, meta-learning transfers knowledge learned from previous tasks to the new ones. However, a critical challenge in meta-learning is the task heterogeneity which cannot be well handled by traditional globally shared meta-learning methods. In addition, current task-specific meta-learning methods may either suffer from hand-crafted structure design or lack the capability to capture complex relations between tasks. In this paper, motivated by the way of knowledge organization in knowledge bases, we propose an automated relational meta-learning (ARML) framework that automatically extracts the cross-task relations and constructs the meta-knowledge graph. When a new task arrives, it can quickly find the most relevant structure and tailor the learned structure knowledge to the meta-learner. As a result, the proposed framework not only addresses the challenge of task heterogeneity by a learned meta-knowledge graph, but also increases the model interpretability. We conduct extensive experiments on 2D toy regression and few-shot image classification and the results demonstrate the superiority of ARML over state-of-the-art baselines.

## 1 INTRODUCTION

Learning quickly with a few samples is the key characteristic of human intelligence, which remains a daunting problem in machine intelligence. The mechanism of learning to learn (a.k.a., meta-learning) is widely used to generalize and transfer prior knowledge learned from previous tasks to improve the effectiveness of learning on new tasks, which has benefited various applications, ranging from computer vision (Kang et al., 2019; Liu et al., 2019) to natural language processing (Gu et al., 2018; Lin et al., 2019). Most of existing meta-learning algorithms learn a globally shared meta-learner (e.g., parameter initialization (Finn et al., 2017), meta-optimizer (Ravi & Larochelle, 2016), metric space (Snell et al., 2017)). However, globally shared meta-learners fail to handle tasks lying in different distributions, which is known as *task heterogeneity*. Task heterogeneity has been regarded as one of the most challenging issues in few-shot learning, and thus it is desirable to design meta-learning models that effectively optimize each of the heterogeneous tasks.

The key challenge to deal with task heterogeneity is how to customize globally shared meta-learner by using task-aware information? Recently, a handful of works try to solve the problem by learning a task-specific representation for tailoring the transferred knowledge to each task (Oreshkin et al., 2018; Vuorio et al., 2019; Lee & Choi, 2018). However, the success of these methods relies on the impaired knowledge generalization among closely correlated tasks (e.g., the tasks sampled from the same distribution). Recently, learning the underlying structure among tasks provide a more effective way for balancing the customization and generalization. Representatively, Yao et al. propose a hierarchically structured meta-learning method to customize the globally shared knowledge to each cluster in a hierarchical way (Yao et al., 2019). Nonetheless, the hierarchical clustering structure completely relies on the handcrafted design which needs to be tuned carefully and may lack the capability to capture complex relationships.

Hence, we are motivated to propose a framework to automatically extract underlying relational structures from previously learned tasks and leverage those relational structures to facilitate knowledge customization on a new task. This inspiration comes from the way of structuring knowledge in knowledge bases (i.e., knowledge graphs). In knowledge bases, the underlying relational structures across text entities are automatically constructed and applied to a new query to improve the searching efficiency. In the meta-learning problem, similarly, we aim at automatically establishing the meta-knowledge graph between prior knowledge learned from previous tasks. When a new task arrives, it queries the meta-knowledge graph and quickly attends to the most relevant entities (nodes), and then takes advantage of the relational knowledge structures between them to boost the learning effectiveness with the limited training data.

The proposed meta-learning framework is named as **A**utomated **R**elational **M**eta-**L**earning (ARML). Specifically, the ARML framework automatically builds the meta-knowledge graph from meta-training tasks to memorize and organize learned knowledge from historical tasks, where each vertex represent one type of meta-knowledge (e.g., the common contour between birds and aircrafts). To learn the meta-knowledge graph at meta-training time, for each task, we construct a prototype-based relational graph for each class, where each vertex represents one prototype. The prototype-based relational graph not only captures the underlying relationship behind samples, but alleviates the potential effects of abnormal samples. The meta-knowledge graph is then learned by and summarizing the information from the corresponding prototype-based relational graphs of meta-training tasks. After constructing the meta-knowledge graph, when a new task comes in, the prototype-based relational graph of the new task taps into the meta-knowledge graph for acquiring the most relevant knowledge, which further enhances the task representation and facilitates its training process.

Our major contributions of the proposed ARML are three-fold: (1) it automatically constructs the meta-knowledge graph to facilitate learning a new task; (2) it empirically outperforms state-of-the-art meta-learning algorithms; (3) the meta-knowledge graph well captures the relationship among tasks and improves the interpretability of meta-learning algorithms.

## 2 Related Work

Meta-learning, allowing machines to learn new skills or adapt to new environments rapidly with a few training examples, has been demonstrated to be successful in both supervised learning tasks (e.g., few-shot image classification) and reinforcement learning settings. There are mainly three research lines of meta-learning: (1) black-box amortized methods design black-box meta-learners (e.g., neural networks) to infer the model parameters (Ravi & Larochelle, 2016; Andrychowicz et al., 2016; Mishra et al., 2018); (2) gradient-based methods aim to learn an optimized initialization of model parameters, which can be adapted to new tasks by a few steps of gradient descent (Finn et al., 2017; 2018; Lee & Choi, 2018); (3) non-parameteric methods combine parameteric meta-learners and non-parameteric learners to learn an appropriate distance metric for few-shot classification (Snell et al., 2017; Vinyals et al., 2016; Yang et al., 2018; Oreshkin et al., 2018; Yoon et al., 2019).

Our work is built upon the gradient-based meta-learning methods. In the line of gradient-based meta-learning, most algorithms learn a globally shared meta-learners from all previous tasks (Finn et al., 2017; Li et al., 2017; Flennerhag et al., 2019), to improve the effectiveness of learning process on new tasks. However, these algorithms typically lack the ability to handle heterogeneous tasks (i.e., tasks sample from sufficient different distributions). To tackle this challenge, recent works tailor the globally shared initialization to different tasks by leveraging task-specific information (Lee & Choi, 2018; Vuorio et al., 2019; Oreshkin et al., 2018) and using probabilistic models (Grant et al., 2018; Yoon et al., 2018; Gordon et al., 2019). Recently, HSML customizes the global shared initialization with a manually designed hierarchical clustering structure to balance the generalization and customization between previous tasks (Yao et al., 2019). However, the hierarchical structure may not accurately reflect the real structure since it highly relies on the hand-crafted design. In addition, the clustering structure further constricts the complexity of relational structures. However, to customize each task, our proposed ARML leverages the most relevant structure from meta-knowledge graph which are automatically constructed by previous knowledge. Thus, ARML not only discovers more accurate underlying structures to improve the effectiveness of meta-learning algorithms, but also the meta-knowledge graph can further enhance the model interpretability.

## 3 Preliminaries

**Few-shot Learning** Considering a task $\mathcal{T}_i$, the goal of few-shot learning is to learn a model with a dataset $\mathcal{D}_i = \{\mathcal{D}_i^{tr}, \mathcal{D}_i^{ts}\}$, where the labeled training set $\mathcal{D}_i^{tr} = \{\mathbf{x}_j^{tr}, \mathbf{y}_j^{tr} | \forall j \in [1, N^{tr}]\}$ only has a few samples and $\mathcal{D}_i^{ts}$ represents the corresponding test set. A learning model (a.k.a., base model) $f$ with parameters $\theta$ are used to evaluate the effectiveness on $\mathcal{D}_i^{ts}$ by minimizing the expected empirical loss on $\mathcal{D}_i^{tr}$, i.e., $\mathcal{L}(\mathcal{D}_{\mathcal{T}_i}^{tr}, \theta)$, and obtain the optimal parameters $\theta_i$. For the regression problem, the loss function is defined based on the mean square error (i.e., $\sum_{(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}_i^{tr}} \|f_\theta(\mathbf{x}_j) - \mathbf{y}_j\|_2^2$) and for the classification problem, the loss function uses the cross entropy loss (i.e., $-\sum_{(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}_i^{tr}} \log p(\mathbf{y}_j | \mathbf{x}_j, f_\theta)$). Usually, optimizing and learning parameter $\theta$ for the task $\mathcal{T}_i$ with a few labeled training samples is difficult. To address this limitation, meta-learning provides us a new perspective to improve the performance by leveraging knowledge from multiple tasks.

**Meta-learning and Model-agnostic Meta-learning** In meta-learning, a sequence of tasks $\{\mathcal{T}_1, ..., \mathcal{T}_I\}$ are sampled from a task-level probability distribution $p(\mathcal{T})$, where each one is a few-shot learning task. To facilitate the adaption for incoming tasks, the meta-learning algorithm aims to find a well-generalized meta-learner on $I$ training tasks at meta-learning phase. At meta-testing phase, the optimal meta-learner is applied to adapt the new tasks $\mathcal{T}_t$. In this way, meta-learning algorithms are capable of adapting to new tasks efficiently even with a shortage of training data for a new task.

Model-agnostic meta-learning (MAML) (Finn et al., 2017), one of the representative algorithms in gradient-based meta-learning, regards the meta-learner as the initialization of parameter $\theta$, i.e., $\theta_0$, and learns a well-generalized initialization $\theta_0^*$ during the meta-training process. The optimization problem is formulated as (one gradient step as exemplary):

$$\theta_0^* := \arg\min_{\theta_0} \sum_{i=1}^{I} \mathcal{L}(f_{\theta_i}, \mathcal{D}_i^{ts}) = \arg\min_{\theta_0} \sum_{i=1}^{I} \mathcal{L}(f_{\theta_0 - \alpha \nabla_\theta \mathcal{L}(f_\theta, \mathcal{D}_i^{tr})}, \mathcal{D}_i^{ts}). \tag{1}$$

At the meta-testing phase, to obtain the adaptive parameter $\theta_t$ for each new task $\mathcal{T}_t$, we finetune the initialization of parameter $\theta_0^*$ by performing gradient updates a few steps, i.e., $f_{\theta_t} = f_{\theta_0^* - \alpha \nabla_\theta \mathcal{L}(f_\theta, \mathcal{D}_t^{tr})}$.

## 4 Methodology

In this section, we introduce the details of the proposed ARML. To better explain how it works, we show its framework in Figure 1. The goal of ARML is to facilitate the learning process of new tasks by leveraging transferable knowledge learned from historical tasks. To achieve this goal, we introduce a meta-knowledge graph, which is automatically constructed at the meta-training time, to organize and memorize historical learned knowledge. Given a task, which is built as a prototype-based relational structure, it taps into the meta-knowledge graph to acquire relevant knowledge for enhancing its own representation. The enhanced prototype representation further aggregate and incorporate with meta-learner for fast and effective adaptions by utilizing a modulating function. In the following subsections, we elaborate three key components: prototype-based sample structuring, automated meta-knowledge graph construction and utilization, and task-specific knowledge fusion and adaptation, respectively.

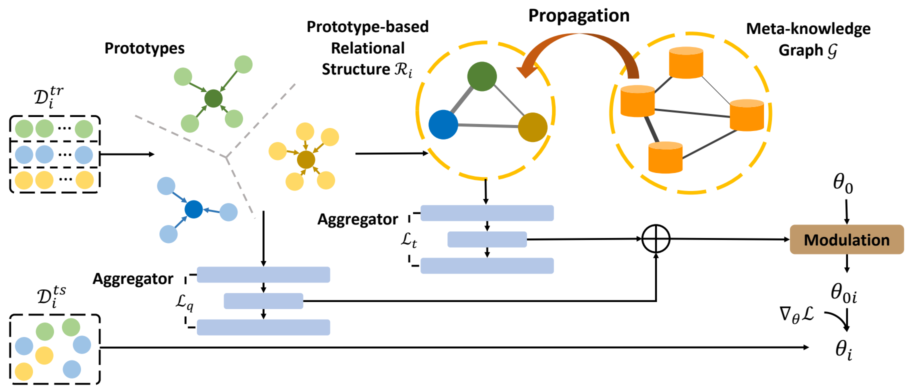


Figure 1: The framework of ARML. For each task $\mathcal{T}_i$, ARML first builds a prototype-based relational structure $\mathcal{R}_i$ by mapping the training samples $\mathcal{D}_i^{tr}$ into prototypes, with each prototype represents one class. Then, $\mathcal{R}_i$ interacts with the meta-knowledge graph $\mathcal{G}$ to acquire the most relevant historical knowledge by information propagation. Finally, the task-specific modulation tailors the globally shared initialization $\theta_0$ by aggregating of raw prototypes and enriched prototypes, which absorbs relevant historical information from the meta-knowledge graph.

### 4.1 Prototype-based Sample Structuring

Given a task which involves either classifications or regressions regarding a set of samples, we first investigate the relationships among these samples. Such relationship is represented by a graph, called prototype-based relational graph in this work, where the vertices in the graph denote the prototypes of different classes while the edges and the corresponding edge weights are created based on the similarities between prototypes. Constructing the relational graph based on prototypes instead of raw samples allows us to alleviate the issue raised by abnormal samples. As the abnormal samples, which locate far away from normal samples, could pose significant concerns especially when only a limited number of samples are available for training. Specifically, for classification problem, the prototype, denoted by $\mathbf{c}_i^k \in \mathbb{R}^d$, is defined as:

$$\mathbf{c}_i^k = \frac{1}{N_k^{tr}} \sum_{j=1}^{N_k^{tr}} \mathcal{E}(\mathbf{x}_j), \tag{2}$$

where $N_k^{tr}$ denotes the number of samples in class $k$. $\mathcal{E}$ is an embedding function, which projects $\mathbf{x}_j$ into a hidden space where samples from the same class are located closer to each other while samples from different classes stay apart. For regression problem, it is not straightforward to construct the prototypes explicitly based on class information. Therefore, we cluster samples by learning an assignment matrix $\mathbf{P}_i \in \mathbb{R}^{K \times N^{tr}}$. Specifically, we formulate the process as:

$$\mathbf{P}_i = \mathrm{Softmax}(\mathbf{W}_p \mathcal{E}^{\mathrm{T}}(\mathbf{X}) + \mathbf{b}_p),\ \mathbf{c}_i^k = \mathbf{P}_i[k]\mathcal{F}(\mathbf{X}), \tag{3}$$

where $\mathbf{P}_i[k]$ represents the $k$-th row of $\mathbf{P}_i$. Thus, training samples are clustered to $K$ clusters, which serve as the representation of prototypes.

After calculating all prototype representations $\{\mathbf{c}_i^k | \forall k \in [1, K]\}$, which serve as the vertices in the the prototype-based relational graph $\mathcal{R}_i$, we further define the edges and the corresponding edge weights. The edge weight $A_{\mathcal{R}_i}(\mathbf{c}_i^j, \mathbf{c}_i^m)$ between two prototypes $\mathbf{c}_i^j$ and $\mathbf{c}_i^m$ is gauged by the the similarity between them. Formally:

$$A_{\mathcal{R}_i}(\mathbf{c}_i^j, \mathbf{c}_i^m) = \sigma(\mathbf{W}_r(|\mathbf{c}_i^j - \mathbf{c}_i^m|/\gamma_r) + \mathbf{b}_r), \tag{4}$$

where $\mathbf{W}_r$ and $\mathbf{b}_r$ represents learnable parameters, $\gamma_r$ is a scalar and $\sigma$ is the Sigmoid function, which normalizes the weight between 0 and 1. For simplicity, we denote the prototype-based relational graph as $\mathcal{R}_i = (\mathbf{C}_{\mathcal{R}_i}, \mathbf{A}_{\mathcal{R}_i})$, where $\mathbf{C}_{\mathcal{R}_i} = \{\mathbf{c}_i^j | \forall j \in [1, K]\} \in \mathbb{R}^{K \times d}$ represent a set of vertices, with each one corresponds to the prototype from a class, while $\mathbf{A}_{\mathcal{R}_i} = \{|A_{\mathcal{R}_i}(\mathbf{c}_i^j, \mathbf{c}_i^m)| \forall j, m \in [1, K]\} \in \mathbb{R}^{K \times K}$ gives the adjacency matrix, which indicates the proximity between prototypes.

### 4.2 Automated Meta-knowledge Graph Construction and Utilization

In this section, we first discuss how to organize and distill knowledge from historical learning process and then expound how to leverage such knowledge to benefit the training of new tasks. To organize and distill knowledge from historical learning process, we construct and maintain a meta-knowledge graph. The vertices represent different types of meta-knowledge (e.g., the common contour between aircrafts and birds) and the edges are automatically constructed and reflect the relationship between meta-knowledge. When serving a new task, we refer to the meta-knowledge, which allows us to efficiently and automatically identify relational knowledge from previous tasks. In this way, the training of a new task can benefit from related training experience and get optimized much faster than otherwise possible. In this paper, the meta-knowledge graph is automatically constructed at the meta-training phase. The details of the construction are elaborated as follows:

Assuming the representation of an vertex $g$ is given by $\mathbf{h}^g \in \mathbb{R}^d$, we define the meta-knowledge graph as $\mathcal{G} = (\mathbf{H}_{\mathcal{G}}, \mathbf{A}_{\mathcal{G}})$, where $\mathbf{H}_{\mathcal{G}} = \{\mathbf{h}^j | \forall j \in [1, G]\} \in \mathbb{R}^{G \times d}$ and $\mathbf{A}_{\mathcal{G}} = \{A_{\mathcal{G}}(\mathbf{h}^j, \mathbf{h}^m) | \forall j, m \in [1, G]\} \in \mathbb{R}^{G \times G}$ denote the vertex feature matrix and vertex adjacency matrix, respectively. To better explain the construction of the meta-knowledge graph, we first discuss the vertex representation $\mathbf{H}_{\mathcal{G}}$. During meta-training, tasks arrive one after another in a sequence and their corresponding vertices representations are expected to be updated dynamically in a timely manner. Therefore, the vertex representation of meta-knowledge graph are defined to get parameterized and learned at the training time. Moreover, to encourage the diversity of meta-knowledge encoded in the meta-knowledge graph, the vertex representations are randomly initialized. Analogous to the definition of weight in the prototype-based relational graph $\mathcal{R}_i$ in equation 4, the weight between a pair of vertices $j$ and $m$ is constructed as:

$$A_{\mathcal{G}}(\mathbf{h}^j, \mathbf{h}^m) = \sigma(\mathbf{W}_o(|\mathbf{h}^j - \mathbf{h}^m|/\gamma_o) + \mathbf{b}_o), \tag{5}$$

where $\mathbf{W}_o$ and $\mathbf{b}_o$ represent learnable parameters and $\gamma_o$ is a scalar.

To enhance the learning of new tasks with involvement of historical knowledge, we query the prototype-based relational graph in the meta-knowledge graph to obtain the relevant knowledge in history. The ideal query mechanism is expected to optimize both graph representations simultaneously at the meta-training time, with the training of one graph facilitating the training of the other. In light of this, we construct a super-graph $\mathcal{S}_i$ by connecting the prototype-based relational graph $\mathcal{R}_i$ with the meta-knowledge graph $\mathcal{G}$ for each task $\mathcal{T}_i$. The union of the vertices in $\mathcal{R}_i$ and $\mathcal{G}$ contributes to the vertices in the super-graph. The edges in $\mathcal{R}_i$ and $\mathcal{G}$ are also reserved in the super-graph. We connect $\mathcal{R}_i$ with $\mathcal{G}$ by creating links between the prototype-based relational graph with the meta-knowledge graph. The link between prototype $\mathbf{c}_i^j$ in prototype-based relational graph and vertex $\mathbf{h}^m$ in meta-knowledge graph is weighted by the similarity between them. More precisely, for each prototype $\mathbf{c}_i^j$, the link weight $A_{\mathcal{S}}(\mathbf{c}_i^j, \mathbf{h}^m)$ is calculated by applying softmax over Euclidean distances between $\mathbf{c}_i^j$ and $\{\mathbf{h}^m | \forall m \in [1, G]\}$ as follows:

$$A_{\mathcal{S}}(\mathbf{c}_i^j, \mathbf{h}^k) = \frac{\exp(-\|(\mathbf{c}_i^j - \mathbf{h}^k)/\gamma_s\|_2^2/2)}{\sum_{k'=1}^{K} \exp(-\|(\mathbf{c}_i^j - \mathbf{h}^{k'})/\gamma_s\|_2^2/2)}, \tag{6}$$

where $\gamma_s$ is a scaling factor. We denote the intra-adjacent matrix as $\mathbf{A}_{\mathcal{S}} = \{A_{\mathcal{S}}(\mathbf{c}_i^j, \mathbf{h}^m) | \forall j \in [1, K], m \in [1, G]\} \in \mathbb{R}^{K \times G}$. Thus, for task $\mathcal{T}_i$, the adjacent matrix and feature matrix of super-graph $\mathcal{S}_i = (\mathbf{A}_i, \mathbf{H}_i)$ is defined as $\mathbf{A}_i = (\mathbf{A}_{\mathcal{R}_i}, \mathbf{A}_{\mathcal{S}}; \mathbf{A}_{\mathcal{S}}^{\mathrm{T}}, \mathbf{A}_{\mathcal{G}}) \in \mathbb{R}^{(K+G)\times(K+G)}$ and $\mathbf{H}_i = (\mathbf{C}_{\mathcal{R}_i}; \mathbf{H}_{\mathcal{G}}) \in \mathbb{R}^{(K+G)\times d}$, respectively.

After constructing the super-graph $\mathcal{S}_i$, we are able to propagate the most relevant knowledge from meta-knowledge graph $\mathcal{G}$ to the prototype-based relational graph $\mathcal{R}_i$ by introducing a Graph Neural Networks (GNN). In this work, following the "message-passing" framework (Gilmer et al., 2017), the GNN is formulated as:

$$\mathbf{H}_i^{(l+1)} = \mathrm{MP}(\mathbf{A}_i, \mathbf{H}_i^{(l)}; \mathbf{W}^{(l)}), \tag{7}$$

where $\mathrm{MP}(\cdot)$ is the message passing function and has several possible implementations (Hamilton et al., 2017; Kipf & Welling, 2017; Veličković et al., 2018), $\mathbf{H}_i^{(l)}$ is the vertex embedding after $l$ layers of GNN and $\mathbf{W}^{(l)}$ is a learnable weight matrix of layer $l$. The input $\mathbf{H}_i^{(0)} = \mathbf{H}_i$. After stacking $L$ GNN layers, we get the information-propagated feature representation for the prototype-based relational graph $\mathcal{R}_i$ as the top-$K$ rows of $\mathbf{H}_i^{(L)}$, which is denoted as $\hat{\mathbf{C}}_{\mathcal{R}_i} = \{\hat{\mathbf{c}}_i^j | j \in [1, K]\}$.

### 4.3 Task-specific Knowledge Fusion and Adaptation

After propagating information form meta-knowledge graph to prototype-based relational graph, in this section, we discuss how to learn a well-generalized meta-learner for fast and effective adaptions to new tasks with limited training data. To tackle the challenge of task heterogeneity, in this paper, we incorporate task-specific information to customize the globally shared meta-learner (e.g., initialization here) by leveraging a modulating function, which has been proven to be effective to provide customized initialization in previous studies (Wang et al., 2019; Vuorio et al., 2019).

The modulating function relies on well-discriminated task representations, while it is difficult to learn all representations by merely utilizing the loss signal derived from the test set $\mathcal{D}_i^{ts}$. To encourage such stability, we introduce two reconstructions by utilizing two auto-encoders. There are two collections of parameters, i.e, $\mathbf{C}_{\mathcal{R}_i}$ and $\hat{\mathbf{C}}_{\mathcal{R}_i}$, which contribute the most to the creation of the task-specific meta-learner. $\mathbf{C}_{\mathcal{R}_i}$ express the raw prototype information without tapping into the meta-knowledge graph, while $\hat{\mathbf{C}}_{\mathcal{R}_i}$ give the prototype representations after absorbing the relevant knowledge from the meta-knowledge graph. Therefore, the two reconstructions are built on $\mathbf{C}_{\mathcal{R}_i}$ and $\hat{\mathbf{C}}_{\mathcal{R}_i}$. To reconstruct $\mathbf{C}_{\mathcal{R}_i}$, an aggregator $\mathrm{AG}^q(\cdot)$ (e.g., recurrent network, fully connected layers) is involved to encode $\mathbf{C}_{\mathcal{R}_i}$ into a dense representation, which is further fed into a decoder $\mathrm{AG}^q_{dec}(\cdot)$ to achieve reconstructions. Then, the corresponded task representation $\mathbf{q}_i$ of $\mathbf{C}_{\mathcal{R}_i}$ is summarized by applying a mean pooling operator over prototypes on the encoded dense representation. Formally,

$$\mathbf{q}_i = \mathrm{MeanPool}(\mathrm{AG}^q(\mathbf{C}_{\mathcal{R}_i})) = \frac{1}{N^{tr}} \sum_{j=1}^{N^{tr}} (\mathrm{AG}^q(\mathbf{c}_i^j)), \;\; \mathcal{L}_q = \|\mathbf{C}_{\mathcal{R}_i} - \mathrm{AG}^q_{dec}(\mathrm{AG}^q(\mathbf{C}_{\mathcal{R}_i}))\|_F^2 \tag{8}$$

Similarly, we reconstruct $\hat{\mathbf{C}}_{\mathcal{R}_i}$ and get the corresponded task representation $\mathbf{t}_i$ as follows:

$$\mathbf{t}_i = \mathrm{MeanPool}(\mathrm{AG}^t(\hat{\mathbf{C}}_{\mathcal{R}_i})) = \frac{1}{N^{tr}} \sum_{j=1}^{N^{tr}} (\mathrm{AG}^t(\hat{\mathbf{c}}_i^j)), \;\; \mathcal{L}_t = \|\hat{\mathbf{C}}_{\mathcal{R}_i} - \mathrm{AG}^t_{dec}(\mathrm{AG}^t(\hat{\mathbf{C}}_{\mathcal{R}_i}))\|_F^2 \tag{9}$$

The reconstruction errors in Equations 8 and 9 pose an extra constraint to enhance the training stability, leading to improvement of task representation learning.

**Algorithm 1** Meta-Training Process of ARML

```
Require: $p(\mathcal{T})$: distribution over tasks; $K$: Number of vertices in meta-knowledge graph; $\alpha$: stepsize
    for gradient descent of each task (i.e., inner loop stepsize); $\beta$: stepsize for meta-optimization (i.e.,
    outer loop stepsize); $\mu_1, \mu_2$: balancing factors in loss function
 1: Randomly initialize all learnable parameters $\Phi$
 2: while not done do
 3:     Sample a batch of tasks $\{\mathcal{T}_i | i \in [1, I]\}$ from $p(\mathcal{T})$
 4:     for all $\mathcal{T}_i$ do
 5:         Sample training set $\mathcal{D}_i^{tr}$ and testing set $\mathcal{D}_i^{ts}$
 6:         Construct the prototype-based relational graph $\mathcal{R}_i$ by computing prototype in equation 2
            and weight in equation 4
 7:         Compute the similarity between each prototype and meta-knowledge vertex in equation 6
            and construct the super-graph $\mathcal{S}_i$
 8:         Apply GNN on super-graph $\mathcal{S}_i$ and get the information-propagated representation $\hat{\mathbf{C}}_{\mathcal{R}_i}$
 9:         Aggregate $\mathbf{C}_{\mathcal{R}_i}$ in equation 8 and $\hat{\mathbf{C}}_{\mathcal{R}_i}$ in equation 9 to get the representations $\mathbf{q}_i$, $\mathbf{t}_i$ and
            reconstruction loss $\mathcal{L}_q$, $\mathcal{L}_t$
10:         Compute the task-specific initialization $\theta_{0i}$ in equation 10 and update parameters $\theta_i =$
            $\theta_{0i} - \alpha \nabla_\theta \mathcal{L}(f_\theta, \mathcal{D}_i^{tr})$
11:     end for
12:     Update $\Phi \leftarrow \Phi - \beta \nabla_\Phi \sum_{i=1}^{I} \mathcal{L}(f_{\theta_i}, \mathcal{D}_i^{ts}) + \mu_i \mathcal{L}_t + \mu_2 \mathcal{L}_q$
13: end while
```

After getting the task representation $\mathbf{q}_i$ and $\mathbf{t}_i$, the modulating function is then used to tailor the task-specific information to the globally shared initialization $\theta_0$, which is formulated as:

$$\theta_{0i} = \sigma(\mathbf{W}_g(\mathbf{t}_i \oplus \mathbf{q}_i) + \mathbf{b}_g) \circ \theta_0, \tag{10}$$

where $\mathbf{W}_g$ and $\mathbf{b}_g$ is learnable parameters of a fully connected layer. Note that we adopt the Sigmoid gating as exemplary and more discussion about different modulating functions can be found in ablation studies of Section 5.

For each task $\mathcal{T}_i$, we perform the gradient descent process from $\theta_{0i}$ and reach its optimal parameter $\theta_i$. Combining the reconstruction loss $\mathcal{L}_t$ and $\mathcal{L}_q$ with the meta-learning loss defined in equation 1, the overall objective function of ARML is:

$$\min_{\Phi} \mathcal{L}_{all} = \min_{\Phi} \mathcal{L} + \mu_1 \mathcal{L}_t + \mu_2 \mathcal{L}_q = \min_{\Phi} \sum_{i=1}^{I} \mathcal{L}(f_{\theta_0 - \alpha \nabla_\theta \mathcal{L}(f_\theta, \mathcal{D}_i^{tr})}, \mathcal{D}_i^{ts}) + \mu_1 \mathcal{L}_t + \mu_2 \mathcal{L}_q, \tag{11}$$

where $\mu_1$ and $\mu_2$ are introduced to balance the importance of these three items. $\Phi$ represents all learnable parameters. The algorithm of meta-training process of ARML is shown in Alg. 2. The details of the meta-testing process of ARML are available in Appendix A.

## 5 EXPERIMENTS

In this section, we conduct extensive experiments to demonstrate the effectiveness of the ARML on 2D regression and few-shot classification with the goal of answering the following questions: (1) Can ARML outperform other meta-learning methods?; (2) Can our proposed components improve the learning performance?; (3) Can ARML framework improve the model interpretability by discovering reasonable meta-knowledge graph?

### 5.1 EXPERIMENTAL SETTINGS

**Methods for Comparison** We compare our proposed ARML with two types of baselines: gradient-based meta-learning algorithms and non-parameteric meta-learning algorithms.

*For gradient-based meta-learning methods*: both globally shared methods (MAML (Finn et al., 2017), Meta-SGD (Li et al., 2017)) and task-specific methods (MT-Net (Lee & Choi, 2018), MUMO-MAML (Vuorio et al., 2019), HSML (Yao et al., 2019)) are considered for comparison.

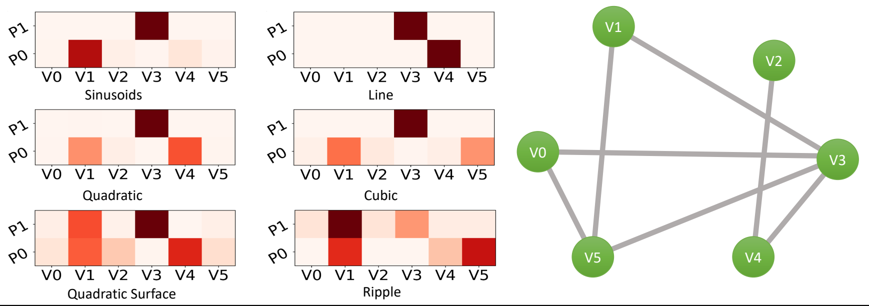
*For non-parametric meta-learning methods*: we select globally shared method Prototypical Network (ProtoNet) (Snell et al., 2017) and task-specific method TADAM (Oreshkin et al., 2018) as baselines. Note that, following the traditional settings, non-parametric baselines are only used in few-shot classification problem. The detailed implementations of baselines are discussed in Appendix B.3.

**Hyperparameter Settings** For the aggregated function in autoencoder structure ($\text{AG}^t$, $\text{AG}^t_{dec}$ $\text{AG}^q$, $\text{AG}^q_{dec}$), we use the GRU as the encoder and decoder in this autoencoder framework. We adopt one layer GCN (Kipf & Welling, 2017) with tanh activation as the implementation of GNN in equation 7. For the modulation network, we try both sigmoid, tanh Film modulation and find that sigmoid modulation performs better. Thus, in the future experiment, we use the sigmoid modulation as modulating function. More detailed discussion about experiment settings are presented in Appendix B.

### 5.2 2D Regression

**Dataset Description.** In 2D regression problem, we adopt the similar regression problem settings as (Finn et al., 2018; Vuorio et al., 2019; Yao et al., 2019; Rusu et al., 2019), which includes several families of functions. In this paper, to model more complex relational structures, we design a 2D regression problem rather than traditional 1D regression. Input $x \sim U[0.0, 5.0]$ and $y \sim U[0.0, 5.0]$ are sampled randomly and random Gaussian noisy with standard deviation 0.3 is added to the output. Furthermore, six underlying functions are selected, including (1) *Sinusoids:* $z(x, y) = a_s sin(w_s x + b_s)$, where $a_s \sim U[0.1, 5.0]$, $b_s \sim U[0, 2\pi]$ $w_s \sim U[0.8, 1.2]$; (2) *Line:* $z(x, y) = a_l x + b_l$, where $a_l \sim U[-3.0, 3.0]$, $b_l \sim U[-3.0, 3.0]$; (3) *Quadratic:* $z(x, y) = a_q x^2 + b_q x + c_q$, where $a_q \sim U[-0.2, 0.2]$, $b_q \sim U[-2.0, 2.0]$, $c_q \sim U[-3.0, 3.0]$; (4) *Cubic:* $z(x, y) = a_c x^3 + b_c x^2 + c_c x + d_c$, where $a_c \sim U[-0.1, 0.1]$, $b_c \sim U[-0.2, 0.2]$, $c_c \sim U[-2.0, 2.0]$, $d_c \sim U[-3.0, 3.0]$; (5) *Quadratic Surface:* $z(x, y) = a_{qs} x^2 + b_{qs} y^2$, where $a_{qs} \sim U[-1.0, 1.0]$, $b_{qs} \sim U[-1.0, 1.0]$; (6) *Ripple:* $z(x, y) = sin(-a_r(x^2 + y^2)) + b_r$, where $a_r \sim U[-0.2, 0.2]$, $b_r \sim U[-3.0, 3.0]$. Note that, function 1-4 are located in the subspace of $y = 1$. Follow (Finn et al., 2017), we use two fully connected layers with 40 neurons as the base model. The number of vertices of meta-knowledge graph is set as 6.

**Results and Analysis.** In Figure 2, we summarize the interpretation of meta-knowledge graph (see top figure) and the the qualitative results (see bottom table) of 10-shot 2D regression. In the bottom table, we can observe that ARML achieves the best performance as compared to competitive gradient-based meta-learning methods, i.e., globally shared models and task-specific models. This finding demonstrates that the meta-knowledge graph is necessary to model and capture task-specific information. The superior performance can also be interpreted in the top figure. In the left, we show the heatmap between prototypes and meta-knowledge vertices (deeper color means higher similarity). We can see that sinusoids and line activate V1 and V4, which may represent curve and line, respectively. V1 and V4 also contribute to quadratic and quadratic surface, which also show the similarity between these two families of functions. V3 is activated in P0 of all functions and the quadratic surface and ripple further activate V1 in P0, which may show the different between 2D functions and 3D functions (sinusoid, line, quadratic and cubic lie in the subspace). Specifically, in the right figure, we illustrate the meta-knowledge graph, where we set a threshold to filter the link with low similarity score and show the rest. We can see that V3 is the most popular vertice and connected with V1, V5 (represent curve) and V4 (represent line). V1 is further connected with V5, demonstrating the similarity of curve representation.



| Model | MAML | Meta-SGD | MT-Net | MUMOMAML | HSML | ARML |
|---|---|---|---|---|---|---|
| 10-shot | $2.292 \pm 0.163$ | $2.908 \pm 0.229$ | $1.757 \pm 0.120$ | $0.523 \pm 0.036$ | $0.494 \pm 0.038$ | $\mathbf{0.438 \pm 0.029}$ |

Figure 2: In the top figure, we show the interpretation of meta-knowledge graph. The left heatmap shows the similarity between prototypes (P0, P1) and meta-knowledge vertices (V0-V5). The right part show the meta-knowledge graph. In the bottom table, we show the overall performance (mean square error with 95% confidence) of 10-shot 2D regression.

## 5.3 Few-shot Classification

**Dataset Description and Settings** In the few-shot classification problem, we first use the benchmark proposed in (Yao et al., 2019), where four fine-grained image classification datasets are included (i.e., CUB-200-2011 (Bird), Describable Textures Dataset (Texture), FGVC of Aircraft (Aircraft), and FGVCx-Fungi (Fungi)). For each few-shot classification task, it samples classes from one of four datasets. In this paper, we call this dataset as *Plain-Multi* and each fine-grained dataset as subdataset.

Then, to demonstrate the effectiveness of our proposed model for handling more complex underlying structures, in this paper, we increase the difficulty of few-shot classification problem by introducing two image filters: blur filter and pencil filter. Similar as (Jerfel et al., 2019), for each image in Plain-Multi, one artistic filters are applied to simulate a changing distribution of few-shot classification tasks. After applying the filters, the total number of subdatasets is 12 and each tasks is sampled from one of them. This data is named as *Art-Multi*. More detailed descriptions of the effect of different filters is discussed in Appendix C.

Following the traditional meta-learning settings, all datasets are divided into meta-training, meta-validation and meta-testing classes. The traditional N-way K-shot settings are used to split training and test set for each task. We adopt the standard four-block convolutional layers as the base learner (Finn et al., 2017; Snell et al., 2017). The number of vertices of meta-knowledge graph for Plain-Multi and Art-Multi datasets are set as 4 and 8, respectively. Additionally, for the miniImagenet, similar as (Finn et al., 2018), which tasks are constructed from a single domain and do not have heterogeneity, we compare our proposed ARML with other baselines and present the results in Appendix D.

### 5.3.1 Performance Validation

**Overall Qualitative Analyses** Experimental results for Plain-Multi and Art-Multi are shown in Table 1 and Table 2, respectively. For each dataset, the performance accuracy with 95% confidence interval are reported. Note that, due to the space limitation, in Art-Multi dataset, we only show the average value of each filter and the full results table are shown in Table 9 of Appendix E. In these two tables, first, we can observe that task-specific models (MT-Net, MUMOMAML, HSML, TADAM) significantly outperforms globally shared models (MAML, Meta-SGD, ProtoNet) in both gradient-based and non-parametric meta-learning research lines. Second, compared ARML with other task-specific gradient-based meta-learning methods, the better performance confirms that ARML can model and extract task-specific information more accurately by leveraging the constructed meta-knowledge graph. Especially, the performance gap between the ARML and HSML verifies the benefits of relational structure compared with isolated clustering structure. Finally, as a gradient-based meta-learning algorithm, ARML can also outperform ProtoNet and TADAM, two representative non-parametric meta-learning algorithms.

Table 1: Overall few-shot classification results (accuracy $\pm$ 95% confidence) on Plain-Multi dataset.

| Settings | Algorithms | Data: Bird | Data: Texture | Data: Aircraft | Data: Fungi |
|---|---|---|---|---|---|
| 5-way 1-shot | MAML | $53.94 \pm 1.45\%$ | $31.66 \pm 1.31\%$ | $51.37 \pm 1.38\%$ | $42.12 \pm 1.36\%$ |
| | MetaSGD | $55.58 \pm 1.43\%$ | $32.38 \pm 1.32\%$ | $52.99 \pm 1.36\%$ | $41.74 \pm 1.34\%$ |
| | MT-Net | $58.72 \pm 1.43\%$ | $32.80 \pm 1.35\%$ | $47.72 \pm 1.46\%$ | $43.11 \pm 1.42\%$ |
| | MUMOMAML | $56.82 \pm 1.49\%$ | $33.81 \pm 1.36\%$ | $53.14 \pm 1.39\%$ | $42.22 \pm 1.40\%$ |
| | HSML | $60.98 \pm 1.50\%$ | $35.01 \pm 1.36\%$ | $57.38 \pm 1.40\%$ | $44.02 \pm 1.39\%$ |
| | ProtoNet | $54.11 \pm 1.38\%$ | $32.52 \pm 1.28\%$ | $50.63 \pm 1.35\%$ | $41.05 \pm 1.37\%$ |
| | TADAM | $56.58 \pm 1.34\%$ | $33.34 \pm 1.27\%$ | $53.24 \pm 1.33\%$ | $43.06 \pm 1.33\%$ |
| | **ARML** | $\mathbf{62.33 \pm 1.47\%}$ | $\mathbf{35.65 \pm 1.40\%}$ | $\mathbf{58.56 \pm 1.41\%}$ | $\mathbf{44.82 \pm 1.38\%}$ |
| 5-way 5-shot | MAML | $68.52 \pm 0.79\%$ | $44.56 \pm 0.68\%$ | $66.18 \pm 0.71\%$ | $51.85 \pm 0.85\%$ |
| | MetaSGD | $67.87 \pm 0.74\%$ | $45.49 \pm 0.68\%$ | $66.84 \pm 0.70\%$ | $52.51 \pm 0.81\%$ |
| | MT-Net | $69.22 \pm 0.75\%$ | $46.57 \pm 0.70\%$ | $63.03 \pm 0.69\%$ | $53.49 \pm 0.83\%$ |
| | MUMOMAML | $70.49 \pm 0.76\%$ | $45.89 \pm 0.69\%$ | $67.31 \pm 0.68\%$ | $53.96 \pm 0.82\%$ |
| | HSML | $71.68 \pm 0.73\%$ | $48.08 \pm 0.69\%$ | $73.49 \pm 0.68\%$ | $56.32 \pm 0.80\%$ |
| | ProtoNet | $68.67 \pm 0.72\%$ | $45.21 \pm 0.67\%$ | $65.29 \pm 0.68\%$ | $51.27 \pm 0.81\%$ |
| | TADAM | $69.13 \pm 0.75\%$ | $45.78 \pm 0.65\%$ | $69.87 \pm 0.66\%$ | $53.15 \pm 0.82\%$ |
| | **ARML** | $\mathbf{73.34 \pm 0.70\%}$ | $\mathbf{49.67 \pm 0.67\%}$ | $\mathbf{74.88 \pm 0.64\%}$ | $\mathbf{57.55 \pm 0.82\%}$ |

Table 2: Overall few-shot classification results (accuracy $\pm$ 95% confidence) on Art-Multi dataset.

| Settings | Algorithms | Avg. Origninal | Avg. Blur | Avg. Pencil |
|---|---|---|---|---|
| 5-way, 1-shot | MAML | $42.70 \pm 1.35\%$ | $40.53 \pm 1.38\%$ | $36.71 \pm 1.37\%$ |
| | MetaSGD | $44.21 \pm 1.38\%$ | $42.36 \pm 1.39\%$ | $37.21 \pm 1.39\%$ |
| | MT-Net | $43.94 \pm 1.40\%$ | $41.64 \pm 1.37\%$ | $37.79 \pm 1.38\%$ |
| | MUMOMAML | $45.63 \pm 1.39\%$ | $41.59 \pm 1.38\%$ | $39.24 \pm 1.36\%$ |
| | HSML | $45.68 \pm 1.37\%$ | $42.62 \pm 1.38\%$ | $39.78 \pm 1.36\%$ |
| | Protonet | $42.08 \pm 1.34\%$ | $40.51 \pm 1.37\%$ | $36.24 \pm 1.35\%$ |
| | TADAM | $44.73 \pm 1.33\%$ | $42.44 \pm 1.35\%$ | $39.02 \pm 1.34\%$ |
| | **ARML** | $\mathbf{47.92 \pm 1.34\%}$ | $\mathbf{44.43 \pm 1.34\%}$ | $\mathbf{41.44 \pm 1.34\%}$ |
| 5-way, 5-shot | MAML | $58.30 \pm 0.74\%$ | $55.71 \pm 0.74\%$ | $49.59 \pm 0.73\%$ |
| | MetaSGD | $57.82 \pm 0.72\%$ | $55.54 \pm 0.73\%$ | $50.24 \pm 0.72\%$ |
| | MT-Net | $57.95 \pm 0.74\%$ | $54.65 \pm 0.73\%$ | $49.18 \pm 0.73\%$ |
| | MUMOMAML | $58.60 \pm 0.75\%$ | $56.29 \pm 0.72\%$ | $51.15 \pm 0.73\%$ |
| | HSML | $60.63 \pm 0.73\%$ | $57.91 \pm 0.72\%$ | $53.93 \pm 0.72\%$ |
| | Protonet | $58.12 \pm 0.74\%$ | $55.07 \pm 0.73\%$ | $50.15 \pm 0.74\%$ |
| | TADAM | $60.35 \pm 0.72\%$ | $58.36 \pm 0.73\%$ | $53.15 \pm 0.74\%$ |
| | **ARML** | $\mathbf{61.78 \pm 0.74\%}$ | $\mathbf{58.73 \pm 0.75\%}$ | $\mathbf{55.27 \pm 0.73\%}$ |

**Model Ablation Study** In this section, we perform the ablation study of the proposed ARML to demonstrate the effectiveness of each component. The results of ablation study on 5-way, 5-shot scenario for Art-Multi dataset are presented in Table 3. In Appendix F, we also show the full results for Art-Multi in Table 6 and the ablation study of Plain-Multi in Table 7. Specifically, to show the effectiveness of prototype construction, in ablation I, we use the mean pooling aggregation of each sample rather than the prototype-based relational graph to interact with meta-knowledge graph. In ablation II, we use all samples to construct the sample-level relational graph without using the prototype. Compared with ablation I and II, the better performance of ARML shows that structuring samples can (1) better handling the underlying relations (2) alleviating the effect of potential anomalies by structuring samples as prototypes.

In ablation III, we remove the meta-knowledge graph and use the prototype-based relational graph structure with aggregator $\mathrm{AG}^q$ as the task representation. The better performance of ARML demonstrates the effectiveness of meta-knowledge graph for capturing the relational structure and facilitating the classification performance. We further remove the reconstruction loss and show the results in ablation IV and the results demonstrate that the autoencoder structure can benefit the process of learning the representation.

In ablation VI and VII, we change the modulate function to film (Perez et al., 2018) and tanh, respectively. We can see that ARML is not very sensitive to the modulating function, and sigmoid function is slightly better than other activation functions in most cases.

Table 3: Results (accuracy $\pm$ 95% confidence) of Ablation Models (5-way, 5-shot) on Art-Multi.

| Ablation Models | Ave. Original | Ave. Blur | Ave. Pencil |
|---|---|---|---|
| I. no prototype-based graph | $60.80 \pm 0.74\%$ | $58.36 \pm 0.73\%$ | $54.79 \pm 0.73\%$ |
| II. no prototype | $61.34 \pm 0.73\%$ | $58.34 \pm 0.74\%$ | $54.81 \pm 0.73\%$ |
| III. no meta-knowledge graph | $59.99 \pm 0.75\%$ | $57.79 \pm 0.73\%$ | $53.68 \pm 0.74\%$ |
| IV. no reconstruction loss | $59.07 \pm 0.73\%$ | $57.20 \pm 0.74\%$ | $52.45 \pm 0.73\%$ |
| V. tanh modulation | $\mathbf{62.34 \pm 0.74\%}$ | $58.58 \pm 0.75\%$ | $54.01 \pm 0.74\%$ |
| VI. film modulation | $60.06 \pm 0.75\%$ | $57.47 \pm 0.73\%$ | $52.06 \pm 0.74\%$ |
| ARML | $61.78 \pm 0.74\%$ | $\mathbf{58.73 \pm 0.75\%}$ | $\mathbf{55.27 \pm 0.73\%}$ |

### 5.3.2 ANALYSIS OF CONSTRUCTED META-KNOWLEDGE GRAPH

In this section, we conduct extensive analysis for the constructed meta-knowledge graph, which is regarded as the key component in ARML. Due to the space limit, we only present the results on Art-Multi datasets. For Plain-Multi, the analysis with similar observations are discussed in Appendix G.

**Performance v.s. Vertice Numbers** We first investigate the impact of vertice numbers in meta-knowledge graph. The results are shown in Table 4. From the results, we can notice that the performance saturates as the number of vertices researches around 8. One potential reason is that 8 vertices is enough to capture the potential relations. If we have a larger datasets with more complex relations, more vertices may be needed. In addition, if the meta-knowledge graph do not have enough vertices, the worse performance suggests that the graph may not be able to capture enough relations across tasks.

Table 4: Sensitivity analysis with different # of vertices in meta-knowledge graph (5-way, 5-shot).

| # of vertices | Ave. Original | Ave. Blur | Ave. Pencil |
|---|---|---|---|
| 4 | $61.18 \pm 0.72\%$ | $58.13 \pm 0.73\%$ | $54.88 \pm 0.75\%$ |
| 8 | $61.78 \pm 0.74\%$ | $58.73 \pm 0.75\%$ | $55.27 \pm 0.73\%$ |
| 12 | $61.66 \pm 0.73\%$ | $58.61 \pm 0.72\%$ | $55.07 \pm 0.74\%$ |
| 16 | $61.75 \pm 0.73\%$ | $58.67 \pm 0.74\%$ | $55.26 \pm 0.73\%$ |
| 20 | $61.91 \pm 0.74\%$ | $58.92 \pm 0.73\%$ | $55.24 \pm 0.72\%$ |

**Model Interpretation Analysis of Meta-Knowledge Graph** We then analyze the learned meta-knowledge graph. For each subdataset, we randomly select one task as exemplary. For each task, in the left part of Figure 3 we show the similarity heatmap between prototypes and vertices in meta-knowledge graph, where deeper color means higher similarity. V0-V8 and P1-P5 denotes the different vertices and prototypes, respectively. The meta-knowledge graph is also illustrated in the right part. Similar as the graph in 2D regression, we set a threshold to filter links with low similarity and illustrate the rest of them. First, We can see that the V1 is mainly activated by bird and aircraft (including all filters), which may reflect the shape similarity between bird and aircraft. Second, V2, V3, V4 are firstly activated by texture and they form a loop in the meta-knowledge graph. Especially, V2 also benefits images with blur and pencil filters. Thus, V2 may represent the main texture and facilitate the training process on other subdatasets. The meta-knowledge graph also shows the importance of V2 since it is connected with almost all other vertices. Third, when we use blur filter, in most cases (bird blur, texture blur, fungi blur), V7 is activated. Thus, V7 may show the similarity of images with blur filter. In addition, the connection between V7 and V2 and V3 show that classify blur images may depend on the texture information. Fourth, V6 (activated by aircraft mostly) connects with V2 and V3, justifying the importance of texture information to classify the aircrafts.

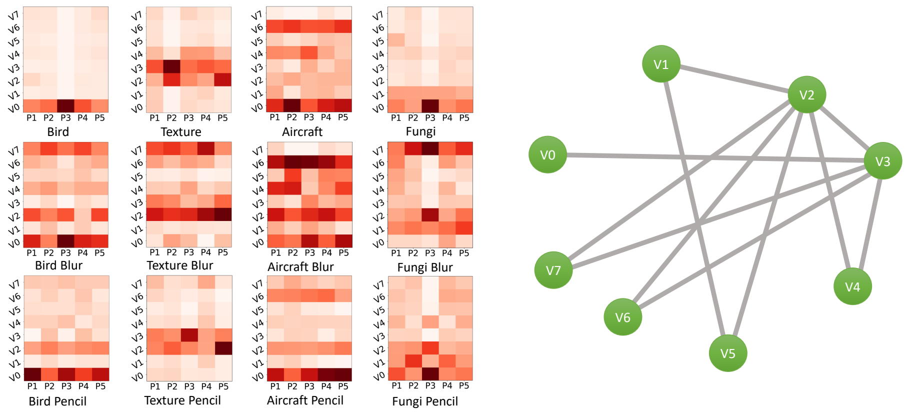


Figure 3: Interpretation of meta-knowledge graph on Art-Multi dataset. For each subdataset, we randomly select one task from them. In the left, we show the similarity heatmap between prototypes (P0-P5) and meta-knowledge vertices (V0-V7). In the right part, we show the meta-knowledge graph.

## 6 Conclusion

In this paper, to improve the effectiveness of meta-learning for handling heterogeneous task, we propose a new framework called ARML, which automatically extract relation across tasks and construct a meta-knowledge graph. When a new task comes in, it can quickly find the most relevant relations through the meta-knowledge graph and use this knowledge to facilitate its training process. The experiments demonstrate the effectiveness of our proposed algorithm.

## REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, pp. 3981–3989, 2016.

Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *ICLR*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pp. 1126–1135, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, 2018.

Sebastian Flennerhag, Pablo G Moreno, Neil D Lawrence, and Andreas Damianou. Transferring knowledge across learning processes. *ICLR*, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, pp. 1263–1272. JMLR. org, 2017.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. In *ICLR*, 2019.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *ICLR*, 2018.

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. Meta-learning for low-resource neural machine translation. In *EMNLP*, 2018.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pp. 1024–1034, 2017.

Ghassen Jerfel, Erin Grant, Thomas L Griffiths, and Katherine Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. *NeurIPS*, 2019.

Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *ICCV*, 2019.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, pp. 2933–2942, 2018.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Zhaojiang Lin, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. Personalizing dialogue agents via meta-learning. 2019.

Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. *arXiv preprint arXiv:1905.01723*, 2019.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *ICLR*, 2018.

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2018.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, pp. 721–731, 2018.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *ICLR*, 2016.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pp. 4077–4087, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, pp. 3630–3638, 2016.

Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Toward multimodal model-agnostic meta-learning. *NeurIPS*, 2019.

Xin Wang, Fisher Yu, Ruth Wang, Trevor Darrell, and Joseph E Gonzalez. Tafe-net: Task-aware feature embeddings for low shot learning. In *CVPR*, pp. 1831–1840, 2019.

Flood Sung Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *ICML*, pp. 7045–7054, 2019.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *NeurIPS*, pp. 7343–7353, 2018.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *ICML*, 2019.

## A Algorithm in Meta-testing Process

**Algorithm 2** Meta-Testing Process of ARML

```
Require: Training data $\mathcal{D}_t^{tr}$ of a new task $\mathcal{T}_t$
1: Construct the prototype-based relational graph $\mathcal{R}_t$ by computing prototype in equation 2 and
   weight in equation 4
2: Compute the similarity between each prototype and meta-knowledge vertice in equation 6 and
   construct the super-graph $\mathcal{S}_t$
3: Apply GNN on super-graph $\mathcal{S}_t$ and get the updated prototype representation $\hat{\mathbf{C}}_{\mathcal{R}_t}$
4: Aggregate $\mathbf{C}_{\mathcal{R}_t}$ in equation 8, $\hat{\mathbf{C}}_{\mathcal{R}_t}$ in equation 9 and get the representations $\mathbf{q}_t$, $\mathbf{t}_t$
5: Compute the task-specific initialization $\theta_{0t}$ in equation 10
6: Update parameters $\theta_t = \theta_{0t} - \alpha\nabla_\theta\mathcal{L}(f_\theta, \mathcal{D}_t^{tr})$
```

## B Hyperparameters Settings

### B.1 2D Regression

In 2D regression problem, we set the inner-loop stepsize (i.e., $\alpha$) and outer-loop stepsize (i.e., $\beta$) as $0.001$ and $0.001$, respectively. The embedding function $\mathcal{E}$ is set as one layer with 40 neurons. The autoencoder aggregator is constructed by the gated recurrent structures. We set the meta-batch size as 25 and the inner loop gradient steps as 5.

### B.2 Few-shot Image Classification

In few-shot image classification, for both Plain-Multi and Art-Multi datasets, we set the corresponding inner stepsize (i.e., $\alpha$) as 0.001 and the outer stepsize (i.e., $\beta$) as 0.01. For the embedding function $\mathcal{E}$, we employ two convolutional layers with $3 \times 3$ filters. The channel size of these two convolutional layers are 32. After convolutional layers, we use two fully connected layers with 384 and 64 neurons for each layer. Similar as the hyperparameter settings in 2D regression, the autoencoder aggregator is constructed by the gated recurrent structures, i.e., $\mathrm{AG}^t$, $\mathrm{AG}^t_{dec}$ $\mathrm{AG}^q$, $\mathrm{AG}^q_{dec}$ are all GRUs. The meta-batch size is set as 4. For the inner loop, we use 5 gradient steps.

### B.3 Detailed Baseline Settings

For the gradient-based baselines (i.e., MAML, MetaSGD, MT-Net, BMAML. MUMOMAML, HSML), we use the same inner loop stepsize and outer loop stepsize rate as our ARML. As for non-parametric based meta-learning algorithms, both TADAM and Prototypical network, we use the same meta-training and meta-testing process as gradient-based models. Additionally, TADAM uses the same embedding function $\mathcal{E}$ as ARML for fair comparison (i.e., similar expressive ability).

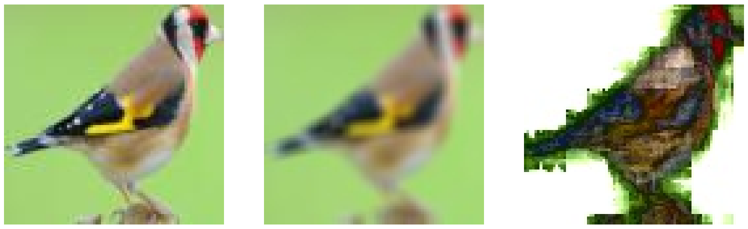
## C Additional Discussion of Datasets

In this dataset, we use pencil and blur filers to change the task distribution. To investigate the effect of pencil and blur filters, we provide one example in Figure 4. We can observe that different filters result in different data distributions. All used filter are provided by OpenCV[1].

## D Results on MiniImagenet

For miniimagenet, since it do not have the characteristic of task heterogeneity, we show the results in Table 5. In this table, we compare the MiniImagenet dataset with other gradient-based meta-learning models (the first four baselines are globally shared models and the next four are task-specific models). Similar as (Finn et al., 2018), we also apply the standard 4-block convolutional layers for each

[1]https://opencv.org/



(a) : Plain Image (b) : with blur filter (c) : with pencil filter

Figure 4: Effect of different filters.

baseline. For MT-Net, we use the reported results in (Yao et al., 2019), which control the model with the same expressive power. The results indicate that our proposed ARML can outperform the original MAML and achieves comparable performance with task-specific models (e.g., MT-Net, PLATIPUS, HSML). Most task-specific models achieve the similar performance on the standard benchmark due to the homogeneity between tasks.

Table 5: Performance comparison on the 5-way, 1-shot MiniImagenet dataset.

| Algorithms | 5-way 1-shot Accuracy |
|---|---|
| MAML (Finn et al., 2017) | $48.70 \pm 1.84\%$ |
| LLAMA (Finn & Levine, 2018) | $49.40 \pm 1.83\%$ |
| Reptile (Nichol & Schulman, 2018) | $49.97 \pm 0.32\%$ |
| MetaSGD (Li et al., 2017) | $50.47 \pm 1.87\%$ |
| MT-Net (Lee & Choi, 2018) | $49.75 \pm 1.83\%$ |
| MUMOMAML (Vuorio et al., 2019) | $49.86 \pm 1.85\%$ |
| HSML (Yao et al., 2019) | $50.38 \pm 1.85\%$ |
| PLATIPUS (Finn et al., 2018) | $50.13 \pm 1.86\%$ |
| ARML | $50.42 \pm 1.73\%$ |

## E Additional Results of Few-shot Image Classification

### E.1 Full Overall Results Table of Art-Multi Dataset

We provide the full results table of Art-Multi Dataset in Table 9. In this table, we can see our proposed ARML outperforms almost all baselines in every sub-datasets.

## F Further Investigation of Ablation Study

In this section, we first show the full evaluation results of model ablation study on Art-Multi dataset in 6. Note that, for the tanh activation (ablation model V), the performance is similar as applying the sigmoid activation. On some subdatasets, the results are even better. We choose the sigmoid activation for ARML because it achieves overall better performance than the tanh activation on more subdatasets. Then, for Plain-Multi dataset, we show the results in 7. The conclusion of ablation study in Plain-Multi dataset is similar as the conclusion drawn from the results on Art-Multi dataset. The improvement on these two datasets verifies the necessity of the joint framework in ARML.

## G Additional Analysis of Meta-knowledge Graph

In this section, we add more interpretation analysis of meta-knowledge graph. First, we show the full evaluation results of sensitivity analysis on Art-Multi dataset in Table 8.

Table 6: Full evaluation results of model ablation study on Art-Multi dataset. B, T, A, F represent bird, texture, aircraft, fungi, respectively. Plain means original image.

| Model | B Plain | B Blur | B Pencil | T Plain | T Blur | T Pencil |
|---|---|---|---|---|---|---|
| I. no prototype-based graph | 72.08% | 71.06% | 66.83% | 45.23% | 39.97% | 41.67% |
| II. no prototype | 72.99% | 70.92% | 67.19% | 45.17% | 40.05% | 41.04% |
| III. no meta-knowledge graph | 70.79% | 69.53% | 64.87% | 43.37% | 39.86% | 41.23% |
| IV. no reconstruction loss | 70.82% | 69.87% | 65.32% | 44.02% | 40.18% | 40.52% |
| V. tanh | 72.70% | 69.53% | 66.85% | **45.81%** | **40.79%** | 38.64% |
| VI. film | 71.52% | 68.70% | 64.23% | 43.83% | 40.52% | 39.49% |
| ARML | **73.05%** | **71.31%** | **67.14%** | 45.32% | 40.15% | **41.98%** |

| Model | A Plain | A Blur | A Pencil | F Plain | F Blur | F Pencil |
|---|---|---|---|---|---|---|
| I. no prototype-based graph | 70.06% | 68.02% | 60.66% | 55.81% | 54.39% | 50.01% |
| II. no prototype | 71.10% | 67.59% | 61.07% | 56.11% | 54.82% | 49.95% |
| III. no meta-knowledge graph | 69.97% | 68.03% | 59.72% | 55.84% | 53.72% | 48.91% |
| IV. no reconstruction loss | 66.83% | 65.73% | 55.98% | 54.62% | 53.02% | 48.01% |
| V. tanh | **73.96%** | **69.70%** | 60.75% | **56.87%** | 54.30% | 49.82% |
| VI. film | 69.13% | 66.93% | 55.59% | 55.77% | 53.72% | 48.92% |
| ARML | 71.89% | 68.59% | **61.41%** | 56.83% | **54.87%** | **50.53%** |

Table 7: Results of Model Ablation (5-way, 5-shot results) on Plain-Multi dataset.

| Ablation Models | Bird | Texture | Aircraft | Fungi |
|---|---|---|---|---|
| I. no sample-level graph | $71.96 \pm 0.72\%$ | $48.79 \pm 0.67\%$ | $74.02 \pm 0.65\%$ | $56.83 \pm 0.80\%$ |
| II. no prototype | $72.86 \pm 0.74\%$ | $49.03 \pm 0.69\%$ | $74.36 \pm 0.65\%$ | $57.02 \pm 0.81\%$ |
| III. no meta-knowledge graph | $71.23 \pm 0.75\%$ | $47.96 \pm 0.68\%$ | $73.71 \pm 0.69\%$ | $55.97 \pm 0.82\%$ |
| IV. no reconstruction loss | $70.99 \pm 0.74\%$ | $48.03 \pm 0.69\%$ | $69.86 \pm 0.66\%$ | $55.78 \pm 0.83\%$ |
| V. tanh | $\mathbf{73.45 \pm 0.71\%}$ | $49.23 \pm 0.66\%$ | $74.39 \pm 0.65\%$ | $57.38 \pm 0.80\%$ |
| VI. film | $72.95 \pm 0.73\%$ | $49.18 \pm 0.69\%$ | $73.82 \pm 0.68\%$ | $56.89 \pm 0.80\%$ |
| ARML | $73.34 \pm 0.70\%$ | $\mathbf{49.67 \pm 0.67\%}$ | $\mathbf{74.88 \pm 0.64\%}$ | $\mathbf{57.55 \pm 0.82\%}$ |

Then, we analyze the meta-knowledge graph on Plain-Multi dataset by visualizing the learned meta-knowledge graph on Plain-Multi dataset (as shown in Figure 5). In this figure, we can see that different subdatasets activate different vertices. Specifically, V2, which is mainly activated by texture, plays a significantly important role in aircraft and fungi. Thus, V2 connects with V3 and V1 in the meta-knowledge graph, which are mainly activated by fungi and aircraft, respectively. In addition, V0 is also activated by aircraft because of the similar contour between aircraft and bird. Furthermore, in meta-knowledge graph, V0 connects with V3, which shows the similarity of environment between bird images and fungi images.

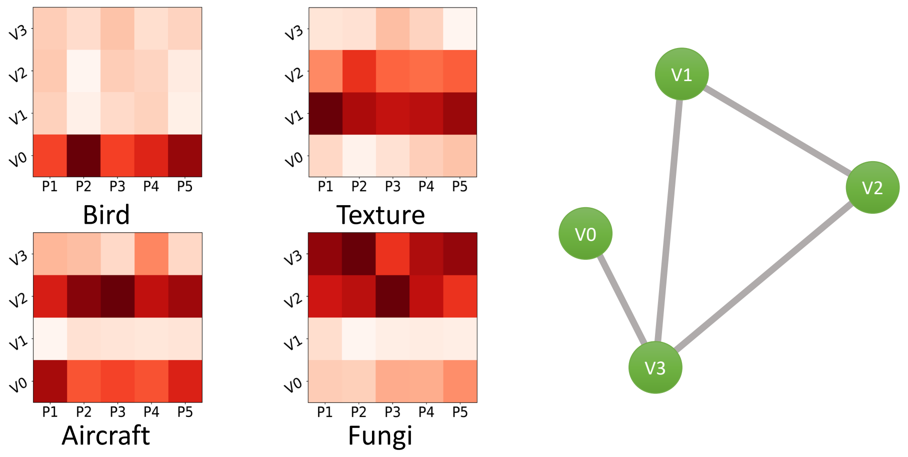


Figure 5: Interpretation of meta-knowledge graph on Plain-Multi dataset. For each subdataset, one task is randomly selected from them. In the left figure, we show the similarity heatmap between prototypes (P1-P5) and meta-knowledge vertices (denoted as E1-E4), where deeper color means higher similarity. In the right part, we show the meta-knowledge graph, where a threshold is also set to filter low similarity links.

Table 8: Full evaluation results of performance v.s. # vertices of meta-knowledge graph on Art-Multi. B, T, A, F represent bird, texture, aircraft, fungi, respectively. Plain means original image.

| # of Vertices | B Plain | B Blur | B Pencil | T Plain | T Blur | T Pencil |
|---|---|---|---|---|---|---|
| 4 | 72.29% | 70.36% | 67.88% | 45.37% | 41.05% | 41.43% |
| 8 | 73.05% | 71.31% | 67.14% | 45.32% | 40.15% | 41.98% |
| 12 | 73.45% | 70.64% | 67.41% | 44.53% | 41.41% | 41.05% |
| 16 | 72.68% | 70.18% | 68.34% | 45.63% | 41.43% | 42.18% |
| 20 | 73.41% | 71.07% | 68.64% | 46.26% | 41.80% | 41.61% |

| # of Vertices | A Plain | A Blur | A Pencil | F Plain | F Blur | F Pencil |
|---|---|---|---|---|---|---|
| 4 | 70.98% | 67.36% | 60.46% | 56.07% | 53.77% | 50.08% |
| 8 | 71.89% | 68.59% | 61.41% | 56.83% | 54.87% | 50.53% |
| 12 | 71.78% | 67.26% | 60.97% | 56.87% | 55.14% | 50.86% |
| 16 | 71.96% | 68.55% | 61.14% | 56.76% | 54.54% | 49.41% |
| 20 | 72.02% | 68.29% | 60.59% | 55.95% | 54.53% | 50.13% |

Table 9: Full results on Art-Multi dataset. In this table, B, T, A, F represent bird, texture, aircraft, fungi, respectively. Plain means original image.

| Settings | Algorithms | B Plain | B Blur | B Pencil | T Plain | T Blur | T Pencil | A Plain | A Blur | A Pencil | F Plain | F Blur | F Pencil |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5-way 1-shot | MAML | 55.27% | 52.62% | 48.58% | 30.57% | 28.65% | 28.39% | 45.59% | 42.24% | 34.52% | 39.37% | 38.58% | 35.38% |
| | MetaSGD | 55.23% | 53.08% | 48.18% | 29.28% | 28.70% | 28.38% | 51.24% | 47.29% | 35.98% | 41.08% | 40.38% | 36.30% |
| | MT-Net | 56.99% | 54.21% | 50.25% | 32.13% | 29.63% | 29.23% | 43.64% | 40.08% | 33.73% | 43.02% | 42.64% | 37.96% |
| | MUMOMAML | 57.73% | 53.18% | 50.96% | 31.88% | 29.72% | 29.90% | 49.95% | 43.36% | 39.61% | 42.97% | 40.08% | 36.52% |
| | HSML | 58.15% | 53.20% | 51.09% | 32.01% | 30.21% | 30.17% | 49.98% | 45.79% | 40.87% | 42.58% | 41.29% | 37.01% |
| | ProtoNet | 53.67% | 50.98% | 46.66% | 31.37% | 29.08% | 28.48% | 45.54% | 43.94% | 35.49% | 37.71% | 38.00% | 34.36% |
| | TADAM | 54.76% | 52.18% | 48.85% | 32.03% | 29.90% | 30.82% | 50.42% | 47.59% | 40.17% | 41.73% | 40.09% | 36.27% |
| | ARML | **59.67%** | **54.89%** | **52.97%** | **32.31%** | **30.77%** | **31.51%** | **51.99%** | **47.92%** | **41.93%** | **44.69%** | **42.13%** | **38.36%** |
| 5-way 5-shot | MAML | 71.51% | 68.65% | 63.93% | 42.96% | 39.59% | 38.87% | 64.68% | 62.54% | 49.20% | 54.08% | 52.02% | 46.39% |
| | MetaSGD | 71.31% | 68.73% | 64.33% | 41.89% | 37.79% | 37.91% | 64.88% | 63.36% | 52.31% | 53.18% | 52.26% | 46.43% |
| | MT-Net | 71.18% | 69.29% | 68.28% | 43.23% | 39.42% | 39.20% | 63.39% | 58.29% | 46.12% | 54.01% | 51.70% | 47.02% |
| | MUMOMAML | 71.57% | 70.50% | 64.57% | 44.57% | 40.31% | 40.07% | 63.36% | 61.55% | 52.17% | 54.89% | 52.82% | 47.79% |
| | HSML | 71.75% | 69.31% | 65.62% | 44.68% | 40.13% | 41.33% | 70.12% | 67.63% | 59.40% | 55.97% | 54.60% | 49.40% |
| | ProtoNet | 70.42% | 67.90% | 61.82% | 44.78% | 38.43% | 38.40% | 65.84% | 63.41% | 54.08% | 51.45% | 50.56% | 46.33% |
| | TADAM | 70.08% | 69.05% | 65.45% | 44.93% | **41.80%** | 40.18% | 70.35% | 68.56% | 59.09% | 56.04% | 54.04% | 47.85% |
| | ARML | **73.05%** | **71.31%** | **67.14%** | **45.32%** | 40.15% | **41.98%** | **71.89%** | **68.59%** | **61.41%** | **56.83%** | **54.87%** | **50.53%** |