Contrastive Knowledge-Augmented Meta-Learning for Few-Shot Classification

Rakshith Subramanyam
Arizona State University, AZ, USA
rsubra17@asu.edu

Mark Heimann, T.S. Jayram, Rushil Anirudh, Jayaraman J. Thiagarajan
Lawrence Livermore National Laboratory, CA, USA
{heimann2, thathachar1, anirudh1, jjayaram}@llnl.gov

Abstract

Model agnostic meta-learning algorithms aim to infer priors from several observed tasks that can then be used to adapt to a new task with few examples. Given the inherent diversity of tasks arising in existing benchmarks, recent methods have resorted to task-specific adaptation of the prior. Our goal is to improve generalization of meta learners when the task distribution contains challenging distribution shifts and semantic disparities. To this end, we introduce CAML (Contrastive Knowledge-Augmented Meta Learning), a knowledge-enhanced few-shot learning approach that evolves a knowledge graph to encode historical experience, and employs a contrastive distillation strategy to leverage the encoded knowledge for task-aware modulation of the base learner. In addition to the standard few-shot task adaptation, we also consider the more challenging multi-domain task adaptation and few-shot dataset generalization settings in our evaluation with standard benchmarks. Our empirical study shows that CAML (i) enables simple task encoding schemes; (ii) eliminates the need for knowledge extraction at inference time; and most importantly, (iii) effectively aggregates historical experience thus leading to improved performance in both multi-domain adaptation and dataset generalization.

1. Introduction

Learning to solve new tasks using only few-shot examples is a long-standing challenge. Meta-learning forms an important class of few-shot learning algorithms that leverages transferable priors from previously observed tasks to learn new tasks quickly. For example, model-agnostic meta-learning (MAML) approaches [2, 21, 6, 4, 3] attempt to learn a single *meta* model (or base learner) on a set of observed tasks, which is assumed to be only a few gradient descent steps away from good task-specific models. Their success hinges on the assumption that the observed tasks are realizations from a common task distribution $p(\mathcal{T})$. Despite its mathematical tractability, the premise of using a single base learner can be insufficient when $p(\mathcal{T})$ is heterogeneous, *i.e.*, the degree of similarity between tasks can be vastly different [18]. This motivates the need for a meta-model to selectively utilize knowledge from its previous experience that is the most relevant for the target task. In this context, task-aware modulation (e.g., MuMo-MAML [18]) is a popular principle to improve MAML on heterogeneous tasks. Conceptually, these approaches use latent task encodings, which characterize realizations from a heterogeneous task distribution, to modulate the base learner and thus improve the adaptation performance on diverse tasks.

In a quest to further improve the performance, recent methods, such as HSML [19] and ARML [20], learn an external knowledge structure for encapsulating information across training episodes and leverage the knowledge to selectively utilize prior experience during adaptation. Though these methods are known to be effective in few-shot adaptation, their generalization under large distribution shifts [11] and semantic disparities [15] can be improved.

In this paper, we introduce Contrastive Knowledge-Augmented Meta Learning (CAML)[1], a task-aware modulation approach, with the goal of improving the generalization of meta-learners. At its core, CAML belongs to the class of MuMo-MAML-style approaches [18]. Though CAML is similar to state-of-the-art ARML [20] in representing few-shot tasks as prototype graphs and using knowledge graphs to encode historical experience, the task encoding scheme, optimization process and the inferencing procedure are entirely different.

Summary of contributions: (i) We propose a contrastive distillation strategy to infuse prior knowledge directly into the image embedding module, which leads to richer task representations and eliminates the need to perform knowl-

[1]CAML codebase: https://github.com/Rakshith-2905/CAML

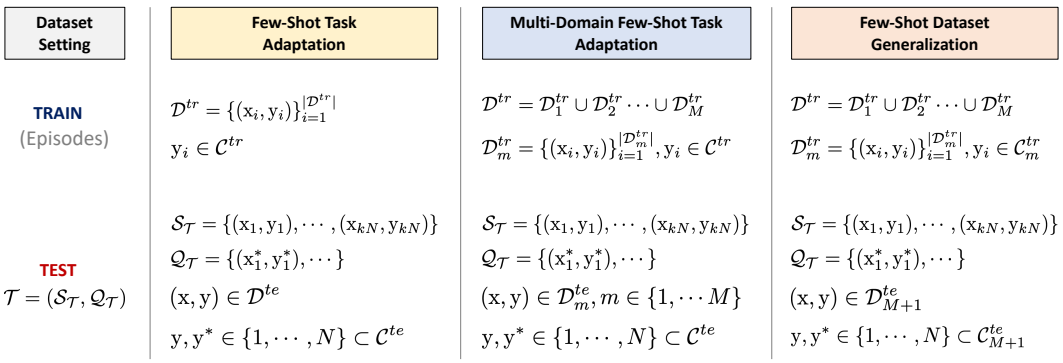


Figure 1. **Few-shot classification tasks**. Here, we formally define the different problem settings considered in this study. As we move from few-shot adaptation to few-shot dataset generalization, the problem becomes increasingly challenging and requires sophisticated task-aware modulation strategies to improve the performance of MAML.

edge extraction during inferencing; (ii) Building upon the improved image embeddings, we adopt a computationally cheap task encoding (average pooling) in lieu of sophisticated architectures (RNN autoencoders in [19, 20]); (iii) We develop an exponential moving average-based update strategy for the knowledge structure, which leads to improved generalization of the meta learner; (iv) Using standard benchmarks (Meta-Dataset, DomainNet), we perform rigorous empirical evaluation of CAML. In particular, we consider the settings of multi-domain task adaptation (we are the first to use this setting) and dataset generalization.

Findings: (i) CAML is a computationally simpler alternative to existing structure-aware meta learners – it uses simple task encoding, is not sensitive to the choice of the image embedding architecture, and does not require knowledge extraction at test-time; (ii) Under larger degrees of heterogeneity (multi-domain), we find that CAML consistently improves upon ARML (2.4% for $1-$shot and and 2.6% for $5-$shot settings); (iii) Even in the challenging dataset generalization setting, CAML provides improvements (across 8 benchmarks from the meta dataset) of 2.1% and 3.3% in $1-$shot and $5-$shot cases.

2. Problem Setup

In this section, we describe the problem settings considered in this study for studying the behavior of different task-aware meta learning approaches. Figure 1 provides an overview of the formulations considered. Broadly, in few-shot classification, training tasks drawn from the distribution $p^{tr}(\mathcal{T})$ are used to learn how to adapt quickly to any of the tasks, and evaluated on previously unseen test tasks from $p^{te}(\mathcal{T})$. Common to all these formulations is that within each of the datasets, the classes seen during training are completely disjoint from those seen during testing.

A. Few-shot Task Adaptation. In this setup, let $\mathcal{D}^{tr} = \{(\mathrm{x}_i, \mathrm{y}_i)\}_{i=1}^{|\mathcal{D}^{tr}|}$ denote the training set comprising samples x_i and their labels y_i, where $\mathrm{y}_i \in \mathcal{C}^{tr}$. In other words, all samples used for training belong to one of the classes from $\mathcal{C}^{tr}$. The goal is to learn an adaptable model using $\mathcal{D}^{tr}$ to support learning new classes with only few examples. For evaluation, we construct a series of few-shot tasks and measure the model's ability to adapt to detect novel classes from $\mathcal{C}^{te}$ (i.e., $\mathcal{C}^{tr} \cap \mathcal{C}^{te} = \emptyset$). More specifically, each $k-$shot $N-$way test episode is represented as the tuple $\mathcal{T} = (\mathcal{S}_\mathcal{T}, \mathcal{Q}_\mathcal{T})$, where the *support* set contains k examples from each of the N classes (selected from $\mathcal{C}^{te}$), i.e., $\mathcal{S}_\mathcal{T} \coloneqq \{(\mathrm{x}_1, \mathrm{y}_1), \cdots, (\mathrm{x}_{kN}, \mathrm{y}_{kN})\}$, $\mathrm{y}_i \in \{1, \cdots, N\}$, and the *query* set $\mathcal{Q}_\mathcal{T} \coloneqq \{(\mathrm{x}_1^*, \mathrm{y}_1^*), \cdots\}$ contains different test examples from the same set of N classes.

B. Multi-Domain Few-shot Task Adaptation. In many practical applications, the training examples $\mathrm{x}_i \in \mathcal{D}^{tr}$ can encompass a variety of distribution shifts. Hence, we consider a new scenario where we represent the training set as a composition of datasets from M different domains, i.e., $\mathcal{D}^{tr} = \mathcal{D}_1^{tr} \cup \mathcal{D}_2^{tr} \cdots \cup \mathcal{D}_M^{tr}$, wherein all samples (regardless of the domain) belong to a common set of classes $\mathcal{C}^{tr}$. The goal here is to learn to adapt to the tasks drawn from any of the M domains. For evaluation, both the support and query sets for a test episode $\mathcal{T}$ are drawn from any domain $m \in \{1, \ldots, M\}$, i.e., $(\mathrm{x}, \mathrm{y}) \in \mathcal{D}_m^{te}$ and the N classes are picked from a disjoint set $\mathcal{C}^{te}$ similar to the previous case.

C. Few-shot Dataset Generalization. In this challenging setting, the training set is defined as a union of M different datasets $\mathcal{D}^{tr} = \mathcal{D}_1^{tr} \cup \mathcal{D}_2^{tr} \cdots \cup \mathcal{D}_M^{tr}$, and more importantly, it is assumed that each dataset contains examples from different sets of classes $\{\mathcal{C}_m^{tr}\}_{m=1}^{M}$. As a result, the goal here is to learn to adapt to completely different semantic concepts corresponding to each of the M datasets. For evaluation, we construct test episodes using novel unseen classes from an entirely different dataset $\mathcal{D}_{M+1}^{te}$. Denoting the set of classes in the novel dataset as $\mathcal{C}_{M+1}^{te}$, we will study how effectively one can leverage the prior to generalize to unseen datasets.

3. Background: Task-Aware Meta Learning

While the few-shot learning literature encompasses a wide variety of approaches, meta-learning is a popular choice [14, 9]. Existing few-shot meta-learning approaches can be broadly categorized into: 1) metric-based meta-learning frameworks [13, 5, 17] that learn a metric or distance function to compare different exemplars; 2) model-based approaches where meta-learning models learn to adjust the model parameters to adapt to new tasks [8, 12]; and 3) gradient-based model agnostic meta-learning models. In particular, our work builds upon model agnostic meta-learning (MAML) [2], which is formulated below.

Given a set of episodes, $\{\mathcal{T}_1^{tr}, \cdots, \mathcal{T}_R^{tr}\}$ comprised of support and query sets ($\mathcal{T}_i^{tr} = (\mathcal{S}_{\mathcal{T}_i^{tr}}, \mathcal{Q}_{\mathcal{T}_i^{tr}})$), from the training set $\mathcal{D}^{tr}$, MAML considers the meta-learner as the initialization of a task network f, *i.e.*, θ_0, and optimizes for a well-generalized initialization θ_0^*. Formally,

$$\theta_0^* = \arg\min_{\bar{\theta}} \sum_{i=1}^{R} \mathcal{L}(f_{\theta_i}; \mathcal{Q}_{\mathcal{T}_i^{tr}}) \quad (1)$$

$$= \arg\min_{\bar{\theta}} \sum_{i=1}^{R} \mathcal{L}(f_{\bar{\theta} - \alpha \nabla_\theta \mathcal{L}(\theta; \mathcal{S}_{\mathcal{T}_i^{tr}})|_{\theta=\bar{\theta}}}; \mathcal{Q}_{\mathcal{T}_i^{tr}}), \quad (2)$$

where the task-specific initialization θ_i is obtained using a gradient step from the meta-initialization θ_0. Note, the notation $\bar{\theta}$ refers to the variables used during the optimization of this bi-level objective function. Here, $\mathcal{L}(f_\theta; \mathcal{S}_{\mathcal{T}_i^{tr}})$ is implemented as the cross entropy loss $\sum_{(\mathrm{x},\mathrm{y}) \in \mathcal{S}_{\mathcal{T}_i^{tr}}} \log P(\mathrm{y}|\mathrm{x}, f_\theta)$.

Task-Aware Modulation. When the tasks used for meta-learning are sampled from a heterogeneous task distribution, inferring a common parameter initialization θ_0 for all tasks can be fundamentally restrictive. Hence, task-aware modulation [18] is a more effective formulation that aims at building a meta-learner which can generalize on heterogeneous task distributions through a set of latent parameters representing task-specific characteristics. For example, MuMo-MAML [18] first uses a task encoder to encode the training episode for a given task into a task embedding vector v_i. The task embedding is then used to obtain modulation vectors that are applied to the global initial parameters θ_0 thereby producing task-aware initialization θ_{0i}. Extending the MAML formulation in (2), the task-aware modulation can be carried out using the support set in the training episode $\mathcal{S}_{\mathcal{T}_i^{tr}}$ and the updated initialization θ_{0i} is used to perform the meta-optimization.

While task-specific initialization can lead to improved generalization on heterogeneous tasks, its effectiveness relies on the ability of the task embeddings to encapsulate all relationships between the large number of observed tasks. Since it is challenging to learn such expressive embeddings, more recent approaches have resorted to storage and retrieval of task-relevant information from historical experience, in order to better balance generalization and customization (task-aware modulation) [19, 20]. For example, hierarchically structured meta learning (HSML) and automated relational meta-learning (ARML) [20] use an external meta knowledge structure to assist the task encoding process. By adopting these knowledge-enhanced representations coupled with a sophisticated task encoder, these approaches often outperform MAML and MuMo-MAML in the standard, few-shot task adaptation setting.

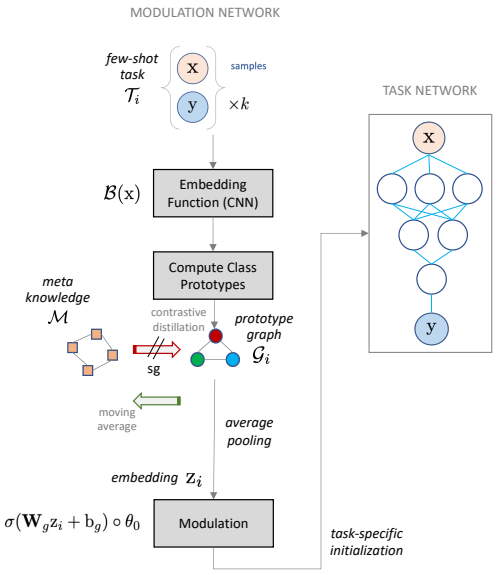


Figure 2. **Approach Overview**. An illustration of the proposed approach for task-aware meta learning. CAML involves four key steps: (i) construct a prototype graph for each training task; (ii) extract knowledge-infused task representation via contrastive distillation; (iii) modulate the base learner based on the task encoding; (iv) update the meta knowledge graph using an exponential moving average strategy. The symbol *sg* denotes the stop gradient operation, *i.e.*, the node features of $\mathcal{M}$ are not directly updated.

4. Proposed Approach

Our goal is to improve the generalization of meta learners under challenging distribution shifts and large semantic disparities. To this end, we develop CAML (see Figure 2), a task-aware modulation approach that uses a meta knowledge graph $\mathcal{M}$ to encapsulate historical experience.

Overview: CAML is comprised of four key steps: (i) *Prototype graph generation*: The first step is to represent each few-shot task as a prototype graph, so that one can incorporate information from the meta knowledge graph and subsequently define a task encoding strategy. The nodes of the prototype graph correspond to class-level centroids computed using features from an image embedding module; (ii) *Knowledge-enhanced task encoding*: In this step, our goal

is to enhance the node features of the prototype graph with relevant information from the knowledge graph. To this end, we propose a novel contrastive training strategy that directly refines the image embedding module by distilling from the knowledge graph. Finally, we define a task encoding based on simple average pooling of prototype node features without any learnable parameters; (iii) *Task-specific modulation*: Next, we will use the inferred task representations to compute a modulation function that can be applied to the base learner and obtain a task-specific initialization; (iv) *Meta knowledge graph update*: The final step is to update the knowledge graph in each training epoch based on the current batch of tasks, which is implemented using an exponential moving average mechanism.

4.1. Algorithm

Step 1: Prototype Graph Generation. Conventionally, feature extractors are used to embed data in low-dimensional latent spaces, where the different classes are easily separable. In task-aware modulation, our goal is to obtain such representations for different few-shot tasks, such that two tasks that are similar in the latent space can use the same task network initialization for effective adaptation. Each $k-$shot $N-$way training episode $\mathcal{T}_i^{tr}$ is comprised of support and query sets $(\mathcal{S}_{\mathcal{T}_i^{tr}} \mathcal{Q}_{\mathcal{T}_i^{tr}})$, wherein there are k samples in each of the N classes randomly selected from $\mathcal{C}^{tr}$. CAML begins by constructing a prototype-based graph [20] with the image embeddings.

Formally, given the support set $\mathcal{S}_{\mathcal{T}_i^{tr}} := \{(\mathrm{x}_j, \mathrm{y}_j), \forall j \in [1, \cdots, kN]\}$ for a training episode, we compute embeddings for each image x_j in the task using an embedding function. While a variety of design choices can be adopted for this, we implement the embedding function using a ResNet-18 architecture. Using the sample-level embeddings, we then compute the prototype vector for each class $n \in [1, \cdots, N]$ by taking the average of the embeddings:

$$\mathrm{v}_i^n = \frac{1}{k} \sum_{\substack{(\mathrm{x}_j, y_j) \in \mathcal{S}_{\mathcal{T}_i^{tr}} \\ y_j = n}} \mathcal{B}(\mathrm{x}_j), \tag{3}$$

where $\mathcal{B}$ denotes the feature extractor that projects an image x_j into $\mathbb{R}^d$. Given the sensitivity of few-shot learning methods to the limited number of examples, operating on the prototype representations reduces the effect of atypical samples. The prototype graph representation is used to both optimize the knowledge-aware task encoding and to update the meta knowledge graph. We also define a simple task encoding function based on the prototype node features:

$$\mathrm{z}_i = \Psi(\mathcal{S}_{\mathcal{T}_i^{tr}}) = \frac{1}{N} \sum_n \mathrm{v}_i^n \tag{4}$$

Node embeddings with high class separability and infused prior knowledge enables the use of this simple feature aggregation strategy in contrast to ARML [20] and HSML [19], which require sophisticated aggregation strategies (e.g., RNN autoencoders).

Step 2: Knowledge-Enhanced Task Encoding. The desideratum of an ideal embedding function in task-aware modulation is to produce expressive task representations that capture the complexity of a given task. Similar to existing structured meta-learning approaches, we adopt a meta knowledge graph structure to encode the historical experience and propose a novel contrastive distillation strategy to produce knowledge-enhanced task encodings. Note that existing approaches update the knowledge structure directly using gradients from the meta update step, which limits its ability to trade-off generalization and customization. Instead, we do not allow gradients to directly alter the knowledge graph (stop gradient or the symbol *sg* in Figure 2).

Formally, let us denote the knowledge graph as $\mathcal{M}$ with randomly initialized node features $\mathcal{H}_{\mathcal{M}} = \{\mathrm{h}_j\}, j = 1, \cdots, M$ and edges $\mathcal{E}_{\mathcal{M}}$. Using the prototype graph, $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, we perform a contrastive distillation from $\mathcal{M}$ to the embedding function $\mathcal{B}$. The edges in both $\mathcal{G}_i$ and $\mathcal{M}$ are parameterized as a function of the absolute difference of the corresponding node features. For example, for any two nodes with features a and b, $Edge(\mathrm{a}, \mathrm{b}) = \sigma(\mathrm{U}^T|\mathrm{a} - \mathrm{b}|)$, where $\mathrm{U} \in \mathbb{R}^{d \times 1}$ is the weight matrix common to all node pairs and σ is the sigmoid function.

In order to extract information for an episode $\mathcal{T}_i$ from $\mathcal{M}$, we construct a super-graph comprising nodes from both $\mathcal{G}_i$ and $\mathcal{M}$. The cross-edges are computed as the softmax of the set of negative Euclidean distances between the pairs. For a pair $\mathrm{v}_i^n \in \mathcal{V}_i$ and $\mathrm{h}_j \in \mathcal{M}$,

$$Edge(\mathrm{v}_i^n, \mathrm{h}_j) = \frac{\exp(-\|(\mathrm{v}_i^n - \mathrm{h}_j)/\gamma\|_2^2/2)}{\sum_{\bar{n},\bar{j}} \exp(-\|(\mathrm{v}_i^{\bar{n}} - \mathrm{h}_{\bar{j}})/\gamma\|_2/2)}. \tag{5}$$

In order to effectively balance between knowledge-enhanced representations and the native representations from the embedding function, we propose a contrastive learning strategy inspired by several existing self-supervised learning approaches such as SimCLR and InfoNCE [1, 10]. Here, we consider the positive pair to be the task encodings from the original prototype representations and the knowledge-enhanced prototype representations obtained via neural message passing on the super-graph. The negatives are node pairs from $\mathcal{G}_i$, which indicate the level of class separability. This objective $\mathcal{L}_{CKD}(\mathcal{T}_i)$ can be expressed as:

$$-\mathbb{E}\left[\log \frac{\exp(\mathtt{sim}(\mathrm{z}_i, \hat{\mathrm{z}}_i))}{\exp(\mathtt{sim}(\mathrm{z}_i, \hat{\mathrm{z}}_i)) + \sum \exp(\mathtt{sim}(\mathrm{v}_i^m, \mathrm{v}_i^n))}\right]. \tag{6}$$

Here, $\hat{\mathrm{z}}_i = \Psi[NMP(\mathcal{G}_i, \mathcal{M})]$ indicates the knowledge-enhanced task representations obtained by first performing neural message passing (NMP) on the super-graph and then

Algorithm 1: Training of CAML

Input: $p^{tr}(\mathcal{T})$ Distribution over training tasks, hyper-parameters α, λ ;
Learnable Parameters: Embedding network $\mathcal{B}$, task network $f(\theta_0)$, modulation parameters Γ, meta knowledge graph $\mathcal{M}$, NMP network;
Initialization: Randomly initialize parameters θ_0, $\mathcal{B}$, $\mathcal{M}$, and NMP network ;
while *not done* **do**
  Sample a batch of tasks $\mathcal{T}_i^{tr} \sim p^{tr}(\mathcal{T})$;
  //meta-train //
  for *each* $\mathcal{T}_i^{tr}$ **do**
    Sample $\mathcal{S}_{\mathcal{T}_i^{tr}}$ and $\mathcal{Q}_{\mathcal{T}_i^{tr}}$ from $\mathcal{T}_i^{tr}$;
    Randomly initialize learnable edges of $\mathcal{M}$;
    Compute prototype vectors $\mathcal{V}_i$ as in (3) ;
    Build prototype graph $\mathcal{G}_i$;
    Construct task representation z_i from (4);
    Compute $\mathcal{L}_{CKD}(\mathcal{T}_i)$ using (6) ;
    Perform task-aware modulation using (7) ;
    Update $\theta_0^* = \theta_0 - \alpha \nabla_\theta \mathcal{L}(\theta; \mathcal{S}_{\mathcal{T}_i^{tr}})$;
  end
  //meta-update//
  Minimize the objective in (8) and update θ_0, $\mathcal{B}$, Γ, edge weights of $\mathcal{M}$, and NMP network;
  for *each* $\mathcal{T}_i^{tr}$ **do**
    Obtain $\hat{\mathcal{H}}_{\mathcal{M}}^i$ using the strategy in Step 4;
  end
  Update $\mathcal{M}$ using $\hat{\mathcal{H}}_{\mathcal{M}}$ averaged over $\mathcal{T}_i^{tr}$;
end

subsequently averaging the updated prototype node representations $\hat{\mathrm{v}}_i^n$. Note that, when performing NMP to obtain knowledge- enhanced task representations, we do not allow the node features in the meta knowledge graph h_j to be changed, and only the prototype representations are updated. Furthermore, the similarity function $\mathtt{sim}$ is implemented using the cosine similarity. In effect, this attempts to modify the embedding function such that the task encoding is consistent with $\mathcal{M}$ while also maximizing the inter-class separability, thus producing rich task representations.

Step 3: Task-Specific Modulation. The next step is to utilize the task encodings for inferring a task-specific meta initialization. To this end, the task representation z_i is used to implement the following modulation function on the global task network initialization θ_0:

$$\theta_{0i} = \Gamma(\theta_0) = \sigma(\mathbf{W}_g \mathrm{z}_i + \mathrm{b}_g) \circ \theta_0, \tag{7}$$

where $\mathbf{W}_g, \mathrm{b}_g$ are learnable parameters. Using a gradient-through-gradient optimization, one can then refine the task-specific initialization θ_{0i}. We incorporate our distillation objective from Step 2 into the meta-update loss function:

$$\min_{\bar{\theta}, \Omega} \sum_{i=1}^{R} \mathcal{L}(f_{\Gamma(\bar{\theta}) - \alpha \nabla_\theta \mathcal{L}(\theta; \mathcal{S}_{\mathcal{T}_i^{tr}})|_{\theta=\Gamma(\bar{\theta})}}; \mathcal{Q}_{\mathcal{T}_i^{tr}}) + \lambda \mathcal{L}_{CKD}(\mathcal{T}_i). \tag{8}$$

Here, Ω corresponds to the parameters of feature extractor $\mathcal{B}$, NMP network and modulation function Γ. The hyper-parameter λ controls the influence of the contrastive distillation term in the overall objective.

Step 4: Meta Knowledge Graph Update. The final step is to update $\mathcal{M}$ with information from the current batch of tasks. By not allowing gradients from the meta update step to alter node features $\mathcal{H}_{\mathcal{M}}$, we are able to better control the historical experience encoded in $\mathcal{M}$. More specifically, using the dataset $\mathcal{T}_i^{tr}$ for each i in parallel, we update the node features $\mathrm{h}_j \in \mathcal{H}_{\mathcal{M}}$ using neural message passing on the super-graph to obtain $\hat{\mathrm{h}}_j^i$. In contrast to the distillation loss computation, during this NMP, we do not allow the prototype node features to be changed and update only the node features of $\mathcal{M}$. Note that, for both the prototype and the knowledge graphs, we use the edges inferred after the meta update in Step 3. Let $\hat{\mathrm{h}}_j$ denote the average of $\hat{\mathrm{h}}_j^i, \forall i$. Finally, we employ an exponential moving average update of the node features of the meta knowledge graph via $\mathrm{h}_j = \alpha \hat{\mathrm{h}}_j + (1-\alpha)\mathrm{h}_j$, where the hyper-parameter α controls the amount of history retained from previous episodes.

5. Results and Findings

Datasets. We consider two large-scale benchmark datasets to evaluate our proposed task-aware modulation approach under the three settings in Figure 1: (i) *Meta-Dataset*: This is a widely adopted benchmark [16] for few-shot image classification and is comprised of multiple image-classification datasets. From this benchmark, we utilize eight datasets for our experiments - (a) CUB-200-2011 (Bird) dataset with 200 classes; (b) describable textures dataset (Texture) with 43 classes; (c) FGVC aircraft (Aircraft) dataset with 100 classes; (d) FGVCx-fungi (Fungi) dataset with 1500 classes; (e) VGG flowers (Flower) dataset containing 102 classes; (f) German traffic signs dataset (Traffic) with 43 classes; (g) Omniglot dataset with 50 classes; (h) Quickdraw dataset with 345 classes; (i) mini-Imagenet with 100 classes. We sampled 5-way few-shot tasks from these datasets for $1-$ and $5-$shot training settings respectively. In each of these datasets, we also constructed disjoint subsets of classes $\mathcal{C}^{tr}$ and $\mathcal{C}^{te}$ for training and testing. For evaluation, we constructed k-shot N-way tasks from the unseen classes $\mathcal{C}^{te}$. Note, for all experiments, the images were resized to $84 \times 84 \times 3$; (ii) *DomainNet*: This popular benchmark [11] for domain adaptation contains images from six different domains (clip-art, info-graph, painting, quick-draw, real,

Table 1. **Few-shot task adaptation**. Performance comparison of the proposed approach against state-of-the-art meta-learning methods. In order to demonstrate that CAML performs competitively in few-shot adaptation, we used 4 different datasets from Meta-Dataset.

Method	Bird	Texture	Aircraft	Fungi	Average
Number of Shots = 1					
Meta-SGD [7]	55.58 ± 1.43	32.38 ± 1.32	52.99 ± 1.36	41.74 ± 1.34	45.67
MAML [2]	53.94 ± 1.45	31.66 ± 1.31	51.37 ± 1.38	42.12 ± 1.36	44.77
MT-Net [6]	58.72 ± 1.43	32.80 ± 1.35	47.72 ± 1.46	43.11 ± 1.42	45.59
B-MAML [21]	54.89 ± 1.48	32.53 ± 1.33	53.63 ± 1.37	42.50 ± 1.33	45.88
HSML [19]	55.99 ± 1.41	32.51 ± 1.35	51.26 ± 1.35	42.86 ± 1.42	45.66
MuMo-MAML [18]	56.82 ± 1.49	33.81 ± 1.36	53.14 ± 1.39	42.22 ± 1.40	46.50
ARML [20]	59.43 ± 1.46	33.30 ± 1.30	56.20 ± 1.34	45.85 ± 1.46	48.70
Proposed	59.71 ± 1.46	35.47 ± 1.38	57.55 ± 1.37	44.97 ± 1.44	49.425
Number of Shots = 5					
Meta-SGD [7]	67.87 ± 0.74	45.49 ± 0.68	66.84 ± 0.70	52.51 ± 0.81	58.18
MAML [2]	68.52 ± 0.79	44.56 ± 0.68	66.18 ± 0.71	51.85 ± 0.85	57.77
MT-Net [6]	69.22 ± 0.75	46.57 ± 0.70	63.03 ± 0.69	53.49 ± 0.83	58.08
B-MAML [21]	69.01 ± 0.74	46.06 ± 0.69	65.74 ± 0.67	52.43 ± 0.84	58.31
HSML [19]	72.07 ± 0.71	44.71 ± 0.66	64.73 ± 0.69	53.38 ± 0.79	58.65
MuMo-MAML [18]	70.49 ± 0.76	45.89 ± 0.69	67.31 ± 0.68	53.96 ± 0.82	59.41
ARML [20]	71.97 ± 0.70	47.18 ± 0.78	73.63 ± 0.64	55.23 ± 0.81	62.00
Proposed	73.09 ± 0.73	48.62 ± 0.69	72.88 ± 0.64	56.11 ± 0.81	62.675

Table 2. **Multi-Domain task adaptation**. Performance comparison of ARML and CAML when the meta-learners were trained using tasks from multiple domains. CAML produces consistently improved generalization in all settings.

Method	ClipArt	InfoGraph	Painting	QuickDraw	Average
Number of Shots = 1					
ARML [20]	47.46 ± 1.48	30.61 ± 1.26	40.26 ± 1.41	65.71 ± 1.33	46.01
Proposed	50.60 ± 1.42	34.13 ± 1.35	43.13 ± 1.44	65.75 ± 1.35	48.40
Number of Shots = 5					
ARML [20]	66.58 ± 0.73	46.19 ± 0.76	56.86 ± 0.72	83.14 ± 0.55	63.19
Proposed	68.47 ± 0.71	50.35 ± 0.75	60.94 ± 0.70	83.47 ± 0.57	65.80

and sketch) belonging to 345 classes. To ensure availability of sufficient data for creating tasks, we ignored classes with less then 50 images and used random splits of 136 and 39 classes for training and evaluation.

Experimental details: For all our experiments we utilized a meta knowledge graph with 4 nodes with 128D features. We leverage a single layer Graph Convolutional Network (GCN) with *tanh* activation for NMP. The base learner uses a 4 layer CNN with 3×3 filters and a single liner classification layer. The 1-shot algorithms were trained for 50K iterations and the 5-shot experiments we trained for 40K iterations, both using a meta batch size 4. We utilized the Adam optimizer for the meta update step and for the inner loop, we performed 5 gradient steps using SGD.

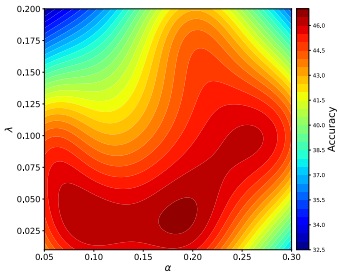
5.1. Findings

CAML performs competitively in standard few-shot adaptation. In our first experiment, we evaluated the ability of CAML to adapt to novel tasks sampled from unseen classes (within the same datasets), and compared against different gradient-based meta learning approaches on the Meta-Dataset benchmark. From the results in Table 1, we clearly notice that approaches that leverage task-aware modulation, e.g., MuMo-MAML, HSML, ARML, CAML etc., consistently outperform vanilla meta-learning approaches such as MAML and Meta-SGD. Among existing task-aware modulation strategies, ARML has been known to produce state-of-the-art results on this benchmark[2]. We find that CAML performs competitively to ARML and HSML in both $1-$ and $5-$ shot training settings, while not requiring knowledge extraction at inference time. This can be attributed to the ability of CAML to capture complex task relations and to effectively distill relevant historical information into the embedding function.

[2]We used the official implementation from the authors (https://github.com/huaxiuyao/ARML) to generate all results for ARML. Even with the prescribed settings, our metrics in Table 1 were lower than those reported in their paper. A few others have also raised this issue on Github, but the authors had not responded at the time of submission.

Table 3. **Dataset Generalization**. The evaluation is carried out using a leave-one-out protocol on the meta-dataset. We find that CAML achieves significantly improved performance over ARML.

Method	Bird	Texture	Aircraft	Fungi	Flower	Traffic	Omniglot	Quickdraw	Imagenet	Average
Number of Shots = 1										
ARML [20]	38.34 ± 1.35	27.13 ± 1.33	27.45 ± 1.23	32.85 ± 1.38	54.79 ± 1.35	39.36 ± 1.33	70.98 ± 1.24	48.02 ± 1.36	32.67 ± 1.32	41.25
Proposed	40.56 ± 1.42	28.75 ± 1.33	28.41 ± 1.24	33.73 ± 1.37	57.89 ± 1.43	44.22 ± 1.39	71.93 ± 1.19	49.62 ± 1.31	34.95 ± 1.34	43.34
Number of Shots = 5										
ARML [20]	55.48 ± 0.80	36.49 ± 0.64	36.39 ± 0.63	44.15 ± 0.73	71.80 ± 0.68	52.69 ± 0.66	89.61 ± 0.44	66.61 ± 0.75	44.63 ± 0.72	55.31
Proposed	58.48 ± 0.72	39.78 ± 0.65	39.45 ± 0.65	45.26 ± 0.75	73.12 ± 0.69	62.18 ± 0.69	91.06 ± 0.42	68.32 ± 0.74	50.39 ± 0.73	58.67



(a) Choice of hyper-parameters

Data Generalization: 1-Shot Training

Image Encoder	Task Encoding	Use KG?	Bird	Texture	Aircraft	Traffic	Average
Shallow CNN	Avg. Pooling	×	38.18 ± 1.34	27.91 ± 1.33	27.77 ± 1.27	44.70 ± 1.32	34.64
Resnet-18	RNN Autoenc.	×	40.88 ± 1.39	28.41 ± 1.28	28.75 ± 1.25	43.44 ± 1.35	35.47
ResNet-18	Avg. Pooling	✔	39.72 ± 1.40	29.55 ± 1.35	27.39 ± 1.24	44.26 ± 1.34	35.23
ResNet-18	Avg. Pooling	×	40.56 ± 1.42	28.75 ± 1.33	28.41 ± 1.24	44.22 ± 1.39	35.48

(b) Impact of different design choices

Figure 3. **Ablations**. We used dataset generalization experiments with $1-$shot training to study the impact of different design choices on the performance of CAML: (a) Sensitivity of α, λ; (b) We explored two architectures for the image encoder (shallow CNN, ResNet18), two task encoding strategies (Average pooling, RNN autoencoder) and the effect of using the inferred knowledge graph at test time.

CAML can handle task heterogeneity in multi-domain adaptation. To further study the performance of CAML on heterogeneous task distributions, in this experiment, we considered DomainNet, a multi-domain benchmark. While both the training and testing tasks were drawn from the same collection of domains (ClipArt, InfoGraph, Painting, QuickDraw), we ensured that the set of classes $\mathcal{C}^{tr}$ and $\mathcal{C}^{te}$ were disjoint. In this setting, the increased complexity of the task distribution makes the modulation process more sensitive, when compared to the previous experiment. For simplicity, we compare CAML with the best performing task-aware modulation baseline, *i.e.*, ARML (our experiments showed CAML was better than MuMo-MAML and HSML as well). As shown in Table 2, CAML achieves performance gaps of 2.4% and 2.6% on average, in $1-$shot and $5-$shot settings respectively.

CAML produces robust task encodings for dataset generalization. Finally, the dataset generalization experiment investigates the ability of CAML to generalize to unseen datasets. The lack of apparent semantic similarity between the classes across different datasets makes this significantly harder. However, improved performance in this problem will be of the most practical value. In this experiment, we evaluated the generalization using a leave-one-out protocol, where we train the meta learner using 8 datasets in Meta-Dataset and evaluate on the ninth dataset. From Table 3, we find that CAML achieves significant performance gains in all training settings – average gains of 2.1% and 3.3% over ARML with the same experimental setup. The observed performance improvements emphasize the efficacy of our meta knowledge construction process, and the robustness of the task representations even for unseen datasets.

5.2. Ablations

We now discuss the impact of different design choices. (i) ***Choice of*** α ***and*** λ: Figure 3(a) illustrates the sensitivity of different choices for α and λ. While α controls the degree to which the history is retained, λ controls the penalty for the distillation cost. These two parameters are used to trade-off generalization (to new tasks) and customization (to observed tasks) of the learner. We find that, when α is very low, i.e., knowledge graphs evolves slowly, using a higher λ hurts the performance. On the other hand, for a reasonably higher $\alpha = 0.2$, the choice of λ becomes less sensitive. In all our experiments, we used $\alpha = 0.2, \lambda = 0.05$;
(ii) ***Choice of feature extractor***: We studied the impact of the choice of architecture for image embedding. In particular, we experimented with (a) ResNet-18; and (b) a shallow CNN (similar to [18]), for the case of dataset generalization. As showed in Figure 3(b), we find that the performance gap between the two models is only $\sim 0.8\%$ on average. This behavior emphasizes the flexibility of implementing CAML, wherein our contrastive distillation strategy is effective with even a shallow CNN model;
(iii) ***Choice of task encoding***: We argued earlier that, through the use of inherently effective image embeddings, CAML can work with a naïve task encoding. To validate this claim, we re-implemented CAML using RNN autoencoder-based task encodings and compared it against the average pooling strategy. Similar to the previous ab-

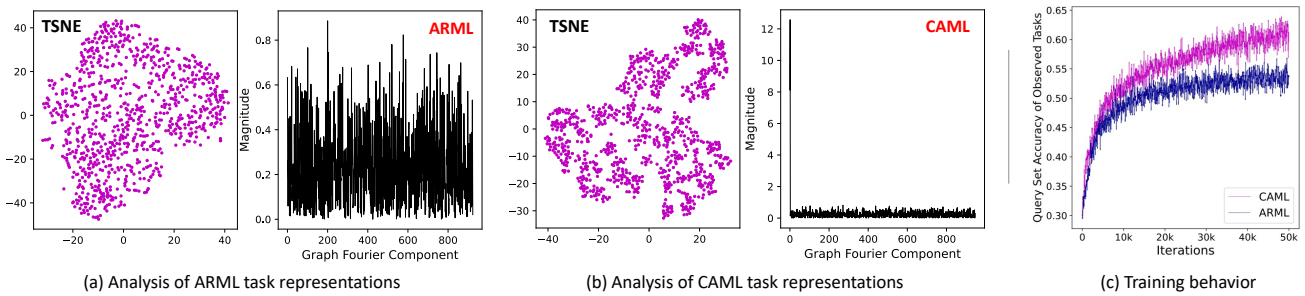


Figure 4. **Analysis.** (a)-(b) Graph Signal Analysis of the task encodings from ARML and CAML for a dataset generalization experiment. For each method, we show the 2-D TSNE embeddings of task encodings for 1000 test tasks and the graph Fourier spectrum of the accuracy score function defined at the nodes of a k-nearest neighbor graphs constructed from the task encodings (k=5); (c) Convergence characteristics of CAML and ARML for a dataset generalization experiment in the $1-$shot training setting.

lation, we used a dataset generalization experiment in the $1-$shot setting (see Figure 3(b)). We find that the RNN autoencoder did not lead to any significant changes in performance (on average the difference was only 0.02%);
(iv) ***Influence of using meta knowledge during adaptation***: Though we used a simple protocol for adaptation, we also experimented a variant, where we performed knowledge infusion (using NMP) at test-time. As showed in Figure 3(b), we found that this did not provide any additional gains (on average 0.25% lower performance), thus implying that the relevant prior information has been effectively distilled into the embedding function;
(v) ***Choice of γ in Eq. (5)***: This parameter was identified using a standard hyper-parameter search. Since the edge weights are learnable (i.e., prototype node features v are updated), we find that the choice of γ is not sensitive. We searched for γ in the range [1,12] and we noticed only marginal variations ($< 0.5\%$ on average) across choices.

6. Analysis

In order to justify the improved behavior of CAML over ARML, we analyzed the expressivity of their corresponding task encodings using tools from graph signal processing. More specifically, we first computed the set of task representations $\mathrm{Z}^{\mathrm{CAML}}$ and $\mathrm{Z}^{\mathrm{ARML}}$ respectively, for a set of 1000 unseen tasks from a dataset generalization experiment (*Traffic* was the unseen dataset). We also obtained the accuracies for all 1000 tasks on the query sets, which are denoted as $\mathrm{f}^{\mathrm{CAML}}$ and $\mathrm{f}^{\mathrm{ARML}}$. Our hypothesis is that if the task representations are robust, two tasks with similar encodings should lead to similar accuracy scores.

To test this hypothesis, we constructed k-nearest neighbor graphs for both CAML and ARML embeddings to obtain the graph adjacency matrices $\mathrm{G}^{\mathrm{CAML}}$ and $\mathrm{G}^{\mathrm{ARML}}$. Next, we computed the graph Fourier basis using the pygsp package (link). Finally, we performed the graph Fourier transform of the signal defined as a vector of accuracy scores. The expectation is that, when the task encodings are robust, the resulting graph Fourier spectrum should concentrate most of the signal's energy at low-frequencies. Figure 4(a)-(b) plots the Fourier spectra obtained for CAML and ARML, when the number of neighbors k was set to 5. Even with such a small neighborhood size, the spectra for ARML contains non-trivial energy at even high frequencies, thus indicating that the task encodings are not consistent with the expected classification performance. In contrast, for CAML, we notice that most of the signal energy is concentrated at low frequencies, thereby demonstrating its improved generalization. This improved behavior is also apparent from the convergence plots in Figure 4(c). The plot shows the accuracy metric measured using the query set of each of the training tasks observed during every iteration.

7. Conclusions

In this work, we presented CAML, a knowledge-enhanced meta-learning approach for few-shot classification. CAML employs a knowledge extraction process that distills prior task information from the learnable knowledge structure to the embedding function using a contrastive objective. This eliminates the need for using the knowledge structure during adaptation and is able to aptly modulate the meta-initialization solely using task encodings obtained via simple average pooling of the prototype embeddings. Using empirical studies on different adaptation settings, we find that CAML consistently outperforms existing baselines. This work motivates the further study of constructing knowledge priors for few-shot adaptation under challenging distribution shifts and semantic discrepancies.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Supported by the LDRD Program under project 21-ERD-012.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

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

[3] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *arXiv preprint arXiv:1710.11622*, 2017.

[4] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

[5] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[6] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2927–2936. PMLR, 2018.

[7] 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.

[8] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, pages 2554–2563. PMLR, 2017.

[9] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[10] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[11] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019.

[12] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[13] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[14] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[15] Eleni Triantafillou, Hugo Larochelle, Richard Zemel, and Vincent Dumoulin. Learning a universal template for few-shot dataset generalization. In *International Conference on Machine Learning*, pages 10424–10433. PMLR, 2021.

[16] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *ICLR*, 2020.

[17] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[18] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Multimodal model-agnostic meta-learning via task-aware modulation. *arXiv preprint arXiv:1910.13616*, 2019.

[19] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *International Conference on Machine Learning*, pages 7045–7054. PMLR, 2019.

[20] Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. Automated relational meta-learning. In *ICLR*, 2020.

[21] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353, 2018.