Clustered Task-Aware Meta-Learning by Learning From Learning Paths

Danni Peng and Sinno Jialin Pan

***Abstract*—To enable effective learning of new tasks with only a few examples, meta-learning acquires common knowledge from the existing tasks with a globally shared meta-learner. To further address the problem of task heterogeneity, recent developments balance between customization and generalization by incorporating task clustering to generate task-aware modulation to be applied to the global meta-learner. However, these methods learn task representation mostly from the features ofinput data, while the task-specific optimization process with respect to the base-learner is often neglected. In this work, we propose a Clustered Task-Aware Meta-Learning (CTML) framework with task representation learned from both features and learning paths. We first conduct rehearsed task learning from the common initialization, and collect a set of geometric quantities that adequately describes this learning path. By inputting this set of values into a meta path learner, we automatically abstract path representation optimized for downstream clustering and modulation. Aggregating the path and feature representations results in an improved task representation. To further improve inference efficiency, we devise a shortcut tunnel to bypass the rehearsed learning process at a meta-testing time. Extensive experiments on two real-world application domains: few-shot image classification and cold-start recommendation demonstrate the superiority of CTML compared to state-of-the-art methods. We provide our code at https://github.com/didiya0825.**

***Index Terms*—Task clustering, task representation based on learning path, task-aware meta-learning.**

I. Introduction

THE astonishing performance of deep learning relies on large amounts of data, which are not always available. Humans, on the other hand, are able to learn new tasks much more quickly, leveraging prior experience to relate knowledge among tasks. Inspired by this property of human intelligence, meta-learning (also known as learning to learn) [1] acquires transferable knowledge from existing tasks in the form of embedding functions [2], [3], [4], initial parameters [5], [6], optimization strategies [7], [8], or models that directly map training samples to network parameters [9], [10]. Recent developments adopt more advanced techniques like transductive inference [11], [12] and causal intervention [13] to achieve further improvements. Although meta-learning has shown success in fields like few-shot image classification and cold-start recommendation, most of them typically assume that all the tasks are drawn from a single distribution and face the challenge of handling tasks that come from different underlying distributions, a problem known as *task heterogeneity* [14], [15], [16].

To overcome this challenge, many recently developed methods leverage task-specific information to customize the global meta-learner [17], [18], [19], [20], [21], [22], [23]. To further consider generalization among related tasks, methods that perform various types of task clustering are proposed, including K-means clustering, hierarchical clustering or relational graph to accommodate the cluster information, with parameterization learned from meta-training tasks or from an external knowledge base [15], [16], [24], [25], [26], [27]. Despite their effectiveness in improving over the globally shared meta-learning algorithms, these methods learn task representations only based on the input distribution in (original or projected) feature space, while the interaction between data and the base-learner is often neglected.

The amount of information contained in the task-specific data about a network responsible for performing the task can be seen as a good representation of the task itself. This can be manifested as the gradients of the network parameters with respect to the task-specific loss, or Fisher Information Matrix (FIM) which indicates the importance of different network parameters in solving the task. [28] introduce a task embedding method based on FIM of a pre-trained model. [29] propose to incorporate the gradients as features when adapting a pre-trained network to a specific task. To alleviate the conflict issue of the global initialization methods, [30] utilize task gradients at the initialization as task representation to produce attenuation. However, these methods represent tasks based on gradients at only a single point in parameter space (e.g., at a fixed pre-trained model or at the initialization), while the potential of exploiting a wider range on the task manifold remains unexplored.

Considering the parameter initialization approach, the key of success lies in that the task adaptation process is accounted for when training the meta-learner. This task-specific learning may involve multiple gradient descent steps, and thus constitute a learning trajectory on the loss surface [31], [32]. To better characterize the task optimization behaviors, it is more beneficial

Manuscript received 21 May 2022; revised 11 January 2023; accepted 14 February 2023. Date of publication 6 March 2023; date of current version 30 June 2023. This work was supported in part by Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Singapore Joint Research Institute (JRI), Nanyang Technological University, Singapore. Recommended for acceptance by T. M. Hospedales. *(Corresponding author: Sinno Jialin Pan.)*

Danni Peng is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: danni001@e.ntu.edu.sg).

Sinno Jialin Pan is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, and also with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: sinnopan@cuhk.edu.hk).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TPAMI.2023.3250323.

Digital Object Identifier 10.1109/TPAMI.2023.3250323

to look at the complete learning trajectory as opposed to only the first gradient step at the initialization, as it is likely that tasks with similar gradients at first will have their learning paths diverged as the update proceeds. With that in mind, we are motivated to leverage the entire learning path for better task representation, which will then be used to condition the global initialization.

In particular, we propose a **C**lustered **T**ask-aware **M**eta-**L**earning (CTML) framework, building upon the well-known Model-Agnostic Meta-Learning (MAML) [5]. To address the problem of task heterogeneity with a good balance between customization and generalization, we modulate the common initialization based on task representation learned from both its local task-specific information and global clustering results. In addition to using the input features to represent the task, we further leverage the task learning path with respect to the base-learner to characterize task from the perspective of optimization. To facilitate clustering among similar learning paths, we devise a GRU-based meta path learner to abstract path representations from the step-wise geometric quantities along the learning path. We realize that it may be too costly to rehearse the entire learning process for task representation. Hence, we further propose a shortcut tunnel to bypass the rehearsed task learning during meta-testing and predict path cluster assignment directly from the feature cluster assignment. We carefully study the effectiveness of CTML in two real-world application domains: few-shot image classification and cold-start recommendation, and show that our method is able to outperform the baselines with comparable inference time.

II. Related Work

A. Task-Aware Meta-Learning

Our work mainly focuses on enhancing the task representation to handle task heterogeneity in meta-learning. Recent developments address task heterogeneity by tailoring the shared knowledge to tasks with task-specific information. [20] and [21] model the uncertainty exists in task distribution with probabilistic framework. [18], [33] and [19] condition the base network on task-specific data by designing a meta adaptation network. To enable more robust training, [22] propose to learn a lower-dimensional latent space specific to each task to generate the base network parameters. [23] introduce a non-parametric approach of task-conditioning based on the task's similarities with other meta-training tasks. Relevant to our work, [17] build upon MAML and modulate the global parameter initialization with task representation based on input features. [34] generate preconditioning on the inner update gradients. [35] and [36] meta-learn the inner update rule and inner update loss function respectively conditioned on the task-specific information. [37] further incorporate ensemble of the inner-loop updated models to reduce variance. [12] adopt the transductive setting, utilizing the query set to generate synthetic gradient steps from a task-adaptive initializaiton.

However, customizing the common knowledge to individual tasks without considering the relations among similar tasks may lead to poor generalization. In regard to this, [25] and [26] apply K-means clustering on users (treated as tasks) based on their profile information to address the cold-start problem in recommender systems. [24] employ a hierarchical structure to model the nested relationships inherent in domains with clear taxonomy, such as image classification. [15] further develop an automatic relational graph method by constructing a meta-knowledge graph to preserve and propagate the global structural information. [16] take advantages of an external knowledge base to facilitate task clustering. Despite their effectiveness in exploiting the global clustering structure to generalize across tasks, they rely solely on the features of input data to represent task identity, while the interaction between data and the base-learner (e.g., gradients) which can be highly informative for task representation is neglected.

B. Gradient-Based Task Representation

Generally, the use of gradient-based features has been shown to have great potential for deep network adaptation to specific tasks [29], [38]. In continual learning, [32], [39], [40] explicitly use gradients for task representation to fight catastrophic forgetting. In the context of task-aware meta-learning, [28] propose to use the FIM of a pre-trained network to represent tasks and assist the meta task of selecting the best pre-trained model. [30] leverage gradients of the globally initialized parameters to generate task-specific attenuation. However, these methods utilize gradients only at a single point in parameter space (e.g., at a fixed pre-trained network or at the initialization), while the potential of exploiting the entire learning trajectory is under-explored.

Meta-learning typically incorporates the entire learning process of individual tasks when learning the meta-learner. [41] propose to conduct task clustering based on gradients at each inner-update step and aggregate the gradients for more stable task adaptation, while [27] perform clustering at the end of the learning trajectory (i.e., based on the adapted parameters of individual tasks), and then compute a set of new cluster-specific initializations for a second-stage meta-training to further boost performance. [31] highlight the benefits of leveraging the entire learning paths for deriving the common knowledge, but the proposed algorithm only relies on one geometric quantity – the length of the path, which can be limited in terms of characterizing the learning paths, and the information is not used to tailor the initialization to specific task. Our work aims to address the above-mentioned limitations, whereby higher-order behaviors of the learning path are also taken into account to learn a better task representation for conditioning the global initialization.

III. Preliminaries

In a task-heterogeneous setting, tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ are sampled from a mixture of task distributions $\{p_1(\mathcal{T}), p_2(\mathcal{T}), \ldots\}$, where the number of underlying distributions may not be known. The goal of meta-learning is to learn sharable knowledge by training the meta-learner on a set of meta-training tasks $\mathbb{T}^{tr} = \{\mathcal{T}_i\}_{i=1}^{N^{tr}}$, and test it on a set of meta-testing tasks $\mathbb{T}^{te} = \{\mathcal{T}_i\}_{i=N^{tr}+1}^{N}$. For each task $\mathcal{T}_i \in \mathbb{T}^{tr} \cup \mathbb{T}^{te}$, the samples are further divided into a training set (also termed support set) $\mathcal{D}_{\mathcal{T}_i}^{tr} = \{(x_{i,j}, y_{i,j})\}_{j=1}^{n_{\mathcal{T}_i}^{tr}}$ and a test

set (also termed query set) $\mathcal{D}^{te}_{\mathcal{T}_i} = \{(x_{i,j}, y_{i,j})\}_{j=n^{tr}_{\mathcal{T}_i}+1}^{n_{\mathcal{T}_i}}$, which together form an "episode" [2]. This episodic scheme allows us to train tasks to learn fast during meta-training, and test the learning performance of new tasks in the same manner during meta-testing. For few-shot learning, the size of the training set $n^{tr}_{\mathcal{T}_i}$ is usually small.

Our work builds upon Model-Agnostic Meta-Learning (MAML) [5]. It implements the meta-learner as an initialization of parameters $\theta_0 \in \mathbb{R}^D$ of the base-learner f_θ responsible for the prediction task. During meta-training, the global initialization θ_0 is first adapted to each meta-training task $\mathcal{T}_i \in \mathbb{T}^{tr}$ by learning on the respective training set $\mathcal{D}^{tr}_{\mathcal{T}_i}$, which yields the task-specific parameters $\theta_{\mathcal{T}_i}$. After that, loss is computed on the test set $\mathcal{D}^{te}_{\mathcal{T}_i}$ based on $\theta_{\mathcal{T}_i}$ and propagated backwards to update θ_0. Taking one-step adaptation as an example, the meta-optimization is as follows:

$$\begin{aligned}\theta_0^* &= \arg\min_{\theta_0} \sum_{\mathcal{T}_i \in \mathbb{T}^{tr}} \mathcal{L}(f_{\theta_{\mathcal{T}_i}}, \mathcal{D}^{te}_{\mathcal{T}_i}) \\ &= \arg\min_{\theta_0} \sum_{\mathcal{T}_i \in \mathbb{T}^{tr}} \mathcal{L}\left(f_{\theta_0 - \alpha \nabla_\theta \mathcal{L}(f_{\theta_0}, \mathcal{D}^{tr}_{\mathcal{T}_i})}, \mathcal{D}^{te}_{\mathcal{T}_i}\right), \end{aligned} \quad (1)$$

where α is the adaptation rate, $\mathcal{L}(f_\theta, \mathcal{D})$ can be mean square error loss for regression task (i.e., $\frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}(y - f_\theta(x))^2$), or cross-entropy loss for classification task (i.e., $-\frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} y \log f_\theta(x)$). During meta-testing, the learned global initialization θ_0^* is adapted to each meta-testing task $\mathcal{T}_i \in \mathbb{T}^{te}$ using $\mathcal{D}^{tr}_{\mathcal{T}_i}$, and the learning performance is evaluated on $\mathcal{D}^{te}_{\mathcal{T}_i}$.

However, with a globally shared initialization, MAML is not capable of handling task heterogeneity. Though task-adaptive methods have been developed to tailor the global initialization, they lack explicit modeling of the global clustering structure. Hence, a framework considering both task-specific information and global structure is desired.

IV. Methodology

Grounded on MAML, our CTML framework modulates the common initialization based on task representation learned from two different perspectives enhanced with the clustering information. Specifically, given the training set of a specific task, we first conduct rehearsed learning of the task from the common initialization and compute a set of step-wise quantities along the learning path. This set of values will be inputted into a meta path learner to generate the task-specific path embedding. On the other hand, input features of the given task will also be extracted to form the feature embedding. Both feature and path embeddings will then undergo a soft K-means clustering in their respective latent space. Finally, the two embeddings enriched with cluster information will be aggregated to produce the task-aware modulation to be applied on the global initialization. From this modulated initialization, standard MAML follows, which conducts the actual task learning on the training set and performs meta-update across tasks by optimizing the test sets. To further improve the inference efficiency, a shortcut tunnel is simultaneously learned during meta-training, which at meta-testing time can be used to bypass the rehearsed learning process by generating path cluster assignment directly from the feature cluster assignment. Fig. 1 shows an overview of the CTML framework.

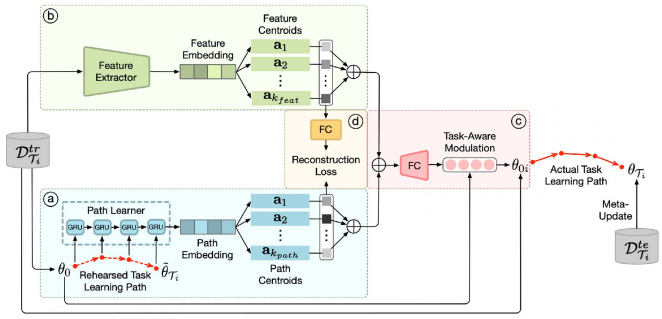


Fig. 1. Overview of CTML. For each incoming task $\mathcal{T}_i$, part (a) extracts path representation and performs clustering in the path representation space; part (b) extracts feature representation and performs clustering in the feature representation space; part (c) generates a task-aware modulation to be applied on the global initialization; part (d) reconstructs path cluster assignment from feature cluster assignment.

In the following sections, we first elaborate on task representation learning based on learning path and features respectively, and then introduce the task-aware modulation and the shortcut tunnel.

A. Task Representation Based on Learning Path

1) Path Construction: To obtain the representation of task $\mathcal{T}_i$ based on learning path, we first conduct a τ-step rehearsed learning from the global initialization θ_0 on training set $\mathcal{D}^{tr}_{\mathcal{T}_i}$. Applying the same gradient descent update as in MAML, we obtain the updated parameters at each step $t \in \{1, 2, \ldots, \tau\}$ by:

$$\tilde{\theta}^t_{\mathcal{T}_i} = \tilde{\theta}^{t-1}_{\mathcal{T}_i} - \alpha \nabla_\theta \mathcal{L}(f_{\tilde{\theta}^{t-1}_{\mathcal{T}_i}}, \mathcal{D}^{tr}_{\mathcal{T}_i}), \quad (2)$$

where $\tilde{\theta}^0_{\mathcal{T}_i} = \theta_0$. The overhead $\tilde{\cdot}$ is used to differentiate between the parameters updated from the rehearsed learning and from the actual learning.

Joining the discrete points $\{(\tilde{\theta}^t_{\mathcal{T}_i}, \tilde{\mathcal{L}}^t_{\mathcal{T}_i})\}_{t=0}^{\tau}$ constitutes the rehearsed learning path of $\mathcal{T}_i$ (we write $\mathcal{L}(f_{\tilde{\theta}^t_{\mathcal{T}_i}}, \mathcal{D}^{tr}_{\mathcal{T}_i})$ as $\tilde{\mathcal{L}}^t_{\mathcal{T}_i}$ for brevity) on the unique task manifold $M_{\mathcal{T}_i} \in \mathbb{R}^{D+1}$, characterized by the base-learner function and the task-specific data distribution. We collect the coordinates of point $(\tilde{\theta}^t_{\mathcal{T}_i}, \tilde{\mathcal{L}}^t_{\mathcal{T}_i})$ at each update step $t \in \{0, 1, 2, \ldots, \tau\}$ to indicate the exact learning trajectory. To account for higher-order behaviors of the learning path, we further incorporate the gradients $\nabla_\theta \tilde{\mathcal{L}}^t_{\mathcal{T}_i}$ and the Fisher Information Matrix (FIM) $\tilde{F}^t_{\mathcal{T}_i}$ at each step, specifying the direction and curvature respectively. The FIM provides a measure of the amount of information that task $\mathcal{T}_i$ contains about the parameters θ. We use FIM to transform second-order derivative to the square of first-order derivative:

$$\begin{aligned} F(\theta) &= -\mathbb{E}_{x,y\sim\tilde{p}(x)p_\theta(y|x)}[\nabla^2_\theta \log p_\theta(y|x)] \\ &= \mathbb{E}_{x,y\sim\tilde{p}(x)p_\theta(y|x)}[\nabla_\theta \log p_\theta(y|x) \nabla_\theta \log p_\theta(y|x)^\top]. \end{aligned} \quad (3)$$

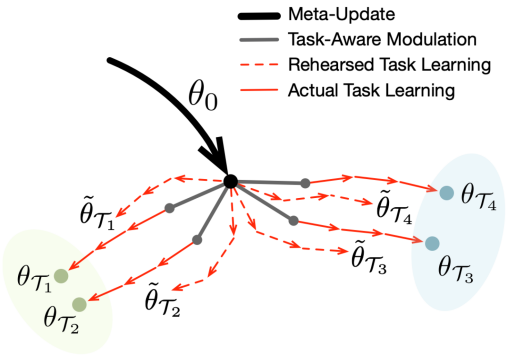


Fig. 2. Illustration of task learning under the CTML framework. Tasks with similar rehearsed learning paths will produce similar modulations on the global initialization θ_0, resulting in more informed task-adaptive initializations that facilitate the actual task learning.

To avoid analytically computing the expectation of the complicated density function, we adopt the common approach to approximate the expectation with empirical Fisher based on the training samples [42], [43]. That is, for a task $\mathcal{T}_i$ with the cross-entropy loss $\mathcal{L}$, we compute its empirical FIM $\tilde{F}^t_{\mathcal{T}_i}$ at step t as follows:

$$\tilde{F}^t_{\mathcal{T}_i} = \mathbb{E}_{x,y\sim\mathcal{T}_i}[\nabla_\theta \mathcal{L}(f_{\tilde{\theta}^t_{\mathcal{T}_i}}, x, y)\nabla_\theta \mathcal{L}(f_{\tilde{\theta}^t_{\mathcal{T}_i}}, x, y)^\top]. \quad (4)$$

Following [28], we assume negligible correlation between different parameters and consider only the diagonal entries of $\tilde{F}^t_{\mathcal{T}_i}$, denoted by $\tilde{\mathcal{F}}^t_{\mathcal{T}_i}$. As a result, we obtain a set of step-wise quantities $\{(\tilde{\theta}^t_{\mathcal{T}_i}, \tilde{\mathcal{L}}^t_{\mathcal{T}_i}, \nabla_\theta \tilde{\mathcal{L}}^t_{\mathcal{T}_i}, \tilde{\mathcal{F}}^t_{\mathcal{T}_i})\}_{t=0}^{\tau}$ that adequately describes the rehearsed learning path.

As opposed to looking at only a single point in the parameter space (e.g., use gradients at the initialization to represent task [30]), it is beneficial to take into account the entire learning path traversed, which gives a more complete picture of the optimization process. Take for instance, two tasks $\mathcal{T}_a$ and $\mathcal{T}_b$ may have gradients forming a small angle $\psi^0_{a,b}$ at the initialization, i.e., $\cos(\psi^0_{a,b}) > 0$. However, it is likely that their learning paths will diverge as the gradient update proceeds, i.e., the accumulated angle $\sum_{t=0}^{\tau} \cos(\psi^t_{a,b}) < 0$ (see dotted paths of $\mathcal{T}_2$ and $\mathcal{T}_3$ in Fig. 2). Conversely, the tasks may have gradients pointing in different directions at first, but eventually converge towards the same direction as the learning proceeds (see dotted paths of $\mathcal{T}_1$ and $\mathcal{T}_2$). Considering a single step only can be restrictive for task representation from the optimization perspective. Looking further down the path allows better characterization of the learning behavior and even the flexibility to determine the "important" steps (as shown in our experiments later), producing a more informative task representation.

2) Path Representation: Having collected the step-wise quantities $\{(\tilde{\theta}^t_{\mathcal{T}_i}, \tilde{\mathcal{L}}^t_{\mathcal{T}_i}, \nabla_\theta \tilde{\mathcal{L}}^t_{\mathcal{T}_i}, \tilde{\mathcal{F}}^t_{\mathcal{T}_i})\}_{t=0}^{\tau}$ along the rehearsed learning path, the problem now is how to evaluate "similarity" among different task learning paths for clustering. Previous works with single-step gradients usually compute dot product or cosine similarity between gradients of two tasks [30], [44], [45]. With multiple steps, a straightforward modification will be to simply sum up the similarity scores at all steps. However, this human-defined rule may not be the best way of measuring path similarity. Instead, we propose to employ a meta path learner to automatically learn a path embedding from the step-wise quantities, and then measure the similarity between these vector representations. In other words, the meta path learner induces a latent space on which the distance metric best characterizes what is considered as "similar" (or "dissimilar") among task learning paths.

Before we delve into the design of the path learner, we first elaborate on how we construct the input to the path learner from the step-wise quantities. Recall that at each step t, we have $\tilde{\theta}^t_{\mathcal{T}_i}, \nabla_\theta \tilde{\mathcal{L}}^t_{\mathcal{T}_i}, \tilde{\mathcal{F}}^t_{\mathcal{T}_i} \in \mathbb{R}^D$ and $\tilde{\mathcal{L}}^t_{\mathcal{T}_i} \in \mathbb{R}$. To obtain a regularly shaped input, we duplicate $\tilde{\mathcal{L}}^t_{\mathcal{T}_i}$ to form a D-dimensional vector and stack it together with the other 3 components to form a matrix $\mathbf{P}^t_{\mathcal{T}_i} \in \mathbb{R}^{4\times D}$. Further stacking the $\tau+1$ steps forms the overall 3-D matrix $\mathbf{P}_{\mathcal{T}_i} \in \mathbb{R}^{(\tau+1)\times 4\times D}$. To avoid high model complexity, we apply the meta path learner coordinate-wise on the base-learner parameters (i.e., over the D dimensions). That is to say, the same path learner is used to process the matrix $\mathbf{P}^d_{\mathcal{T}_i} \in \mathbb{R}^{(\tau+1)\times 4}$ independently for all $d \in \{1, 2, \ldots, D\}$. With that, the size of the path learner will be independent of the size of the base-learner, allowing for good scalability to deeper & wider backbones. The capacity of the path learner network can be adjusted by tuning its hidden size.

For the path learner design, we propose to leverage the Gated Recurrent Units (GRUs) to model the sequential dependencies among steps.[1] Specifically, the hidden state $\mathbf{h}^{d,t}_i$ at step t is obtained by inputting the t-th row vector $\mathbf{P}^d_{t,:} \in \mathbb{R}^4$ of $\mathbf{P}^d_{\mathcal{T}_i}$ and the hidden state $\mathbf{h}^{d,t-1}_i$ at the previous step $t-1$ into the GRU cell via $\mathbf{r}^{d,t}_i = \sigma(\mathbf{W}_r \cdot [\mathbf{h}^{d,t-1}_i, \mathbf{P}^d_{t,:}])$, where $[\cdot,\cdot]$ denotes concatenation, $\sigma(\cdot)$ is the sigmoid activation, $\mathbf{h}^{d,t}_i = (1-\mathbf{z}^{d,t}_i) \odot \mathbf{h}^{d,t-1}_i + \mathbf{z}^{d,t}_i \odot \tilde{\mathbf{h}}^{d,t}_i$, $\tilde{\mathbf{h}}^{d,t}_i = \tanh(\mathbf{W}_{\tilde{h}} \cdot [\mathbf{r}^{d,t}_i \odot \mathbf{h}^{d,t-1}_i, \mathbf{P}^d_{t,:}])$, and $\mathbf{z}^{d,t}_i = \sigma(\mathbf{W}_z \cdot [\mathbf{h}^{d,t-1}_i, \mathbf{P}^d_{t,:}])$. Note that $\mathbf{r}^{d,t}_i, \mathbf{z}^{d,t}_i \in (0,1)$ are gates that control how much of past and present information to be retained, and $\mathbf{W}_r, \mathbf{W}_z, \mathbf{W}_{\tilde{h}}$ are learnable weights shared across all steps. For step $t=0$, we use a zero-initialized input hidden state.

The path representation $\mathbf{h}^d_i$ at the d-th dimension is obtained from the final step hidden state $\mathbf{h}^{d,\tau}_i$ via a linear transformation: $\mathbf{h}^d_i = \mathbf{W}_o \cdot \mathbf{h}^{d,\tau}_i + \mathbf{b}_o$. Concatenating the path representations at all D dimensions and passing it through a fully-connected layer gives the final path embedding $\mathbf{e}^{path}_{\mathcal{T}_i} \in \mathbb{R}^{d_e}$ for task $\mathcal{T}_i$:

$$\mathbf{e}^{path}_{\mathcal{T}_i} = \mathrm{FC}\left([\mathbf{h}^1_i, \ldots, \mathbf{h}^D_i]^\top\right). \quad (5)$$

Inspired by [16], [25], [26], we handle task heterogeneity without jeopardizing generalization among similar tasks by employing a simple yet effective soft K-means clustering on the path embeddings.[2] Specifically, we maintain k_{path} learnable cluster centroids $\{\mathbf{a}^{path}_j | \forall j \in [1, k_{path}]\} \in \mathbb{R}^{k_{path}\times d_e}$ for path embeddings. The cluster-enhanced path embedding $\tilde{\mathbf{e}}^{path}_{\mathcal{T}_i} \in$

[1]Other network designs are possible for the meta path learner. We compare their efficacy in ablation study.

[2]Other clustering methods are possible. We leave it to future work.

$\mathbb{R}^{d_e}$ is the weighted sum of the cluster centroids:

$$\tilde{\mathbf{e}}_{\mathcal{T}_i}^{path} = \sum_{j=1}^{k_{path}} q_{ij}^{path} \mathbf{a}_j^{path}, \tag{6}$$

where $q_{ij}^{path} = \frac{(1+||\mathbf{e}_{\mathcal{T}_i}^{path} - \mathbf{a}_j^{path}||^2)^{-1}}{\sum_{j'}(1+||\mathbf{e}_{\mathcal{T}_i}^{path} - \mathbf{a}_{j'}^{path}||^2)^{-1}}$ is the probability of assigning $\mathbf{e}_{\mathcal{T}_i}^{path}$ to cluster j, computed using the Student's t -distribution as a kernel, following [46].

B. Task Representation Based on Features

Task representation based on learning path can be interpreted as encoding the conditional distribution[3] $p_\theta(y|x)$, whereas task is best described by the joint distribution $p(x, y) = p_\theta(y|x)p(x)$. Therefore, we further incorporate the marginal distribution of input features $p(x)$ by learning another representation solely based on features, as what has been done in most of the existing task-aware meta-learning methods [15], [17], [24], [26]. Another significance of including feature-based representation is that we can create a mapping between the path and feature cluster assignments to bypass the rehearsed learning during meta-testing (details will be elaborated in Section IV-D).

Generally, the design of the feature extractor may vary for different application domains. Let $\mathcal{E}(\cdot)$ denote an arbitrary feature extractor, the feature embedding $\mathbf{e}_{\mathcal{T}_i}^{feat} \in \mathbb{R}^{d_e}$ for task $\mathcal{T}_i$ is obtained by aggregating the extracted features of all samples in the training set:

$$\mathbf{e}_{\mathcal{T}_i}^{feat} = \frac{1}{n_{\mathcal{T}_i}^{tr}} \sum_{j=1}^{n_{\mathcal{T}_i}^{tr}} (\mathcal{E}(x_{i,j})). \tag{7}$$

Similar to the path embeddings, we also employ a soft K-means clustering to promote generalization among related feature embeddings. Specifically, given k_{feat} cluster centroids $\{\mathbf{a}_j^{feat} | \forall j \in [1, k_{feat}]\} \in \mathbb{R}^{k_{feat} \times d_e}$, the cluster-enhanced feature embedding $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{feat} \in \mathbb{R}^{d_e}$ is obtained by:

$$\tilde{\mathbf{e}}_{\mathcal{T}_i}^{feat} = \sum_{j=1}^{k_{feat}} q_{ij}^{feat} \mathbf{a}_j^{feat}, \tag{8}$$

where $q_{ij}^{feat} = \frac{(1+||\mathbf{e}_{\mathcal{T}_i}^{feat} - \mathbf{a}_j^{feat}||^2)^{-1}}{\sum_{j'}(1+||\mathbf{e}_{\mathcal{T}_i}^{feat} - \mathbf{a}_{j'}^{feat}||^2)^{-1}}$ is the probability of assigning $\mathbf{e}_{\mathcal{T}_i}^{feat}$ to cluster j.

C. Task-Aware Modulation

We aggregate the path and feature embeddings via a learnable weight vector $\lambda \in \mathbb{R}^{d_e}$ to generate the final task representation. This task representation will then be used to produce a modulation to tailor the global initialization θ_0 to specific task. The modulated initialization θ_{0i} for task $\mathcal{T}_i$ is obtained by:

$$\theta_{0i} = \sigma(\mathbf{W} \cdot (\lambda \tilde{\mathbf{e}}_{\mathcal{T}_i}^{path} \oplus (1-\lambda) \tilde{\mathbf{e}}_{\mathcal{T}_i}^{feat}) + \mathbf{b}) \odot \theta_0, \tag{9}$$

where λ, $\mathbf{W}$, $\mathbf{b}$ are learnable parameters, $\sigma(\cdot)$ is the sigmoid function, $\oplus$ and $\odot$ denote element-wise addition and multiplication respectively.

From this modulated initialization, task $\mathcal{T}_i$ will undergo regular task adaptation learning for τ steps. After that, the meta-learner ϕ will be updated by optimizing loss across all the test sets, the same procedure as in MAML. The meta-optimization objective is:

$$\min_{\phi} \sum_{\mathcal{T}_i \in \mathbb{T}^{tr}} \mathcal{L}\left(f_{\theta_{0i} - \alpha \sum_{t=0}^{\tau-1} \nabla_\theta \mathcal{L}(f_{\theta_{\mathcal{T}_i}^t}, \mathcal{D}_{\mathcal{T}_i}^{tr})}, \mathcal{D}_{\mathcal{T}_i}^{te}\right), \tag{10}$$

where ϕ includes all the meta-learned parameters: the global initialization of the base-learner θ_0, the meta path learner $\{\mathbf{W}_r, \mathbf{W}_z, \mathbf{W}_{\tilde{h}}, \mathbf{W}_o, \mathbf{b}_o\}$, the feature extractor $\mathcal{E}(\cdot)$, the path and feature cluster centroids $\{\mathbf{a}_j^{path}\}_{j=1}^{k_{path}}$ and $\{\mathbf{a}_j^{feat}\}_{j=1}^{k_{feat}}$, and the final modulation $\{\lambda, \mathbf{W}, \mathbf{b}\}$.

D. Improving Meta-Testing Efficiency Via Shortcut

The need to conduct rehearsed learning before the actual learning of each task inevitably leads to twice the inference time compared to the vanilla MAML. Though it is not possible to bypass the rehearsed learning during meta-training, it is possible to improve the inference efficiency during meta-testing with the well-trained cluster centroids. Note that the cluster-enhanced path embedding $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{path}$ used to generate the modulation is obtained solely based on the path cluster centroids and the soft assignment (see (6)). Hence, if we can estimate the path cluster assignment without actually going through the rehearsed learning process, we will be able to cut the inference time by half.

Inspired by this, we devise a shortcut tunnel to predict the path cluster assignment directly from the feature cluster assignment of the same task. The assumption behind is that there exists a relation between the two that can be captured by a mapping function. Imagine an extreme case where two tasks having the exact same feature assignment (i.e., exact same feature embedding), they most probably consist of the same set of samples (e.g., the images), which should correspond to only one set of labels. This will lead to the same learning path and hence, the same path assignment.

Hence, most of the time, similar feature assignments should also correspond to similar path assignments. However, it is possible that the converse may occur, where two tasks with very similar feature embeddings (i.e., the visual features of the images from the two tasks are very similar) are actually drawn from different classes. In that case, the two tasks will be clustered closely in the feature representation space, but positioned far apart in the path representation space. The mapping function is expected to capture this complex relationship and incorporate the new information brought by the paths simply inferring from the corresponding feature assignments.

The mapping between the feature and path cluster assignments can be linear or non-linear. For better expressivity, we employ a two-layer fully connected network to approximate the mapping. The reconstruction of the path assignment from the

[3]See Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2023.3250323, for a detailed explanation of this.

Algorithm 1: Meta-Training of CTML.

Required: Meta-training tasks $\mathbb{T}^{tr} = \{\mathcal{T}_i\}_{i=1}^{N^{tr}}$; Number of steps τ; Adaptation rate α; Meta-update rate β; Number of clusters k_{path}, k_{feat}

Randomly initialize all learnable parameters ϕ
while *not converged* **do**
  Sample a batch of tasks $\mathcal{B}$ from $\mathbb{T}^{tr}$
  for $\mathcal{T}_i \in \mathcal{B}$ **do**
    Extract $\mathbf{e}_{\mathcal{T}_i}^{feat}$ and perform clustering to obtain $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{feat}$
    Extract $\mathbf{e}_{\mathcal{T}_i}^{path}$ from the τ-step rehearsed learning and perform clustering to obtain $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{path}$
    Aggregate $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{feat}$ and $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{path}$ to generate θ_{0i}
    Evaluate $\mathcal{L}(f_{\theta_{\mathcal{T}_i}}, \mathcal{D}_{\mathcal{T}_i}^{te})$ and $\mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr})$
  Meta-update
  $\phi \leftarrow \phi - \beta \nabla_\phi \sum_{\mathcal{T}_i \in \mathcal{B}} [\mathcal{L}(f_{\theta_{\mathcal{T}_i}}, \mathcal{D}_{\mathcal{T}_i}^{te}) + \zeta \mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr})]$

Algorithm 2: Meta-Testing of CTML.

Required: Meta-testing tasks $\mathbb{T}^{te} = \{\mathcal{T}_i\}_{i=N^{tr}+1}^{N}$; Number of steps τ; Adaptation rate α; Number of clusters k_{path}, k_{feat}; Trained ϕ

for $\mathcal{T}_i \in \mathbb{T}^{te}$ **do**
  Extract $\mathbf{e}_{\mathcal{T}_i}^{feat}$ and perform clustering to obtain $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{feat}$
  if *use shortcut approximation* **then**
    Generate path assignment via shortcut tunnel to obtain $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{path}$
  else
    Extract $\mathbf{e}_{\mathcal{T}_i}^{path}$ from the τ-step rehearsed learning and perform clustering to obtain $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{path}$
  Aggregate $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{feat}$ and $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{path}$ to generate θ_{0i}
  Evaluate $\mathcal{L}(f_{\theta_{\mathcal{T}_i}}, \mathcal{D}_{\mathcal{T}_i}^{te})$

feature assignment of $\mathcal{T}_i$ is as follows:

$$\hat{\mathbf{q}}_i^{path} = \text{softmax}(\text{FCs}(\mathbf{q}_i^{feat})), \tag{11}$$

where $\mathbf{q}_i^{feat} = [q_{i1}^{feat}, \ldots, q_{ik_{feat}}^{feat}]^\top \in \mathbb{R}^{k_{feat}}$ and $\hat{\mathbf{q}}_i^{path} = [\hat{q}_{i1}^{path}, \ldots, \hat{q}_{ik_{path}}^{path}]^\top \in \mathbb{R}^{k_{path}}$. The mapping allows for cases where $k_{path} \neq k_{feat}$.

To align the reconstructed and the actual assignment distributions, we use the Jensen-Shannon (JS) divergence (a symmetrized version of Kullback-Leibler (KL) divergence) as the reconstruction loss:

$$\mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr}) = \text{JS}(\hat{\mathbf{q}}_i^{path} || \mathbf{q}_i^{path}) = \frac{1}{2}\text{KL}(\hat{\mathbf{q}}_i^{path} || \mathbf{p}_i) + \frac{1}{2}\text{KL}(\mathbf{q}_i^{path} || \mathbf{p}_i),$$

where $\mathbf{p}_i = \frac{1}{2}(\hat{\mathbf{q}}_i^{path} + \mathbf{q}_i^{path})$, and $\text{KL}(\mathbf{q}||\mathbf{p}) = \sum_j q_j \log \frac{q_j}{p_j}$ is the KL divergence.

During meta-training, the reconstruction loss is optimized together with the loss in (10), resulting in the following overall objective:

$$\min_{\phi} \sum_{\mathcal{T}_i \in \mathbb{T}^{tr}} \mathcal{L}(f_{\theta_{0i} - \alpha \sum_{t=0}^{\tau-1} \nabla_\theta \mathcal{L}(f_{\theta_{\mathcal{T}_i}^t}, \mathcal{D}_{\mathcal{T}_i}^{tr})}, \mathcal{D}_{\mathcal{T}_i}^{te}) + \zeta \mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr}),$$

where ζ controls the weight of $\mathcal{L}_r(\mathcal{D}_{\mathcal{T}_i}^{tr})$, and ϕ now further includes the shortcut tunnel parameters.

To apply the shortcut tunnel during meta-testing, we simply replace $\mathbf{q}_i^{path}$ with $\hat{\mathbf{q}}_i^{path}$ in (6) to obtain the cluster-enhanced path embedding, i.e., $\tilde{\mathbf{e}}_{\mathcal{T}_i}^{path} = \sum_{j=1}^{k_{path}} \hat{q}_{ij}^{path} \mathbf{a}_j^{path}$. The overall meta-training and meta-testing procedures of CTML are summarized in Algorithms 1 and 2.

V. Experiments

We conduct experiments on two application domains: few-shot image classification and cold-start recommendation.

A. Few-Shot Image Classification

In few-shot image classification, each task is defined as assigning images to N classes after training with K samples (i.e., N-way K-shot) [2]. To simulate task heterogeneity, we construct a Mixture-of-Datasets consisting of 6 widely used datasets: CUB-200-2011 (Bird) [47], FGVC-Aircraft (Aircraft) [48], FGVCx-Fungi (Fungi) [49], VGG Flower (Flower) [50], Describable Textures (Texture) [51], and GTSRB Traffic Signs (Traffic Sign) [52]. Following [24], we create each task by sampling N classes from one of the 6 sub-datasets, such that tasks are drawn from different underlying distributions.

To facilitate comparisons with the existing meta-learning methods, we also conduct experiments on some common benchmarks. For task-homogeneous setting, we conduct experiments on 3 popular few-shot benchmarks: miniImageNet [7], tieredImageNet [53] and CIFAR-FS [54], where for each experiment, tasks are drawn from only one underlying distribution. We adopt the pipeline of TSA-MAML [27] to conduct experiments on these 3 datasets.[4]

For task-heterogeneous setting, we test our method on Meta-Dataset [55], a recently introduced large-scale benchmark consisting of 10 sub-datasets. This benchmark simulates a realistic setting by constructing tasks with variable number of ways (i.e., ranging from 5 to 50). Apart from being more challenging, this strategy also makes the benchmark less suitable for MAML-based methods, as without knowing the number of ways in advance, the initialization of the final classification layer cannot be meta-learned during the meta-training phase and this seriously affects the performance of the MAML-based methods. According to [55], in order to implement MAML-based methods on Meta-Dataset, the final classification layer is either zero-initialized (i.e., fo-MAML) or constructed manually from the prototypes (i.e., fo-Proto-MAML). More details will be discussed in Section V-A5. Adopting their strategies, we apply CTML on top of fo-MAML and fo-Proto-MAML on Meta-Dataset to test its effectiveness even with this unfavorable condition. We adopt the data pipeline of Meta-Dataset[5] to test CTML.

More details on data pre-processing and meta-train/validation/test splits are included in Appendix B.1., available in the online supplemental material.

1) Baselines and Our Method: We compare the performance of CTML against 9 other MAML-based methods with

[4]https://github.com/Carbonaraa/TSA-MAML
[5]https://github.com/google-research/meta-dataset

TABLE I
FEW-SHOT CLASSIFICATION PERFORMANCE OF MAML-BASED METHODS ON MIXTURE-OF-DATASETS WITH CONV-4 AS THE BACKBONE. WE SAMPLED 1000 TASKS FOR META-TESTING. THE RESULTS ARE REPORTED IN THE FORM OF MEAN ACCURACY (%) ± STD OVER 8 TRIALS. WE ALSO REPORT THE INFERENCE TIME (IN MILLISECONDS) PER TASK FOR THE 5-WAY 1-SHOT SCENARIO DURING META-TESTING (THE INFERENCE TIME FOR THE 5-SHOT SCENARIO IS AROUND THE SAME AS THE 1-SHOT SCENARIO, HENCE OMITTED FROM THE TABLE)

	Methods	Bird	Aircraft	Fungi	Flower	Texture	Traffic Sign	Avg.	Runtime
5-way 1-shot	MAML [5]	50.06 ± 1.63	48.26 ± 1.32	37.22 ± 1.60	61.08 ± 1.59	30.04 ± 1.68	90.32 ± 1.42	52.83 ± 1.54	82.9
	MMAML [17]	51.59 ± 1.42	50.13 ± 1.04	39.19 ± 1.24	62.27 ± 0.89	31.49 ± 1.36	91.01 ± 1.28	54.28 ± 1.21	83.2
	L2F [30]	55.50 ± 1.47	52.43 ± 1.09	41.09 ± 1.31	64.07 ± 1.34	33.87 ± 1.21	92.72 ± 0.81	56.61 ± 1.21	117.9
	HSML [24]	54.37 ± 1.21	52.04 ± 1.33	41.75 ± 1.10	63.51 ± 1.39	**34.43 ± 1.12**	91.81 ± 0.78	56.32 ± 1.16	101.3
	ARML [15]	56.75 ± 1.78	54.11 ± 1.51	42.83 ± 1.67	64.42 ± 1.62	33.01 ± 1.91	92.14 ± 1.58	57.21 ± 1.68	93.0
	TSA-MAML [27]	53.71 ± 1.38	48.72 ± 1.25	42.43 ± 1.12	63.05 ± 1.42	33.12 ± 1.52	91.20 ± 1.09	55.37 ± 1.30	193.6
	ModGrad [34]	54.87 ± 0.92	47.64 ± 1.29	41.44 ± 1.38	62.73 ± 1.65	33.46 ± 1.33	93.11 ± 1.57	55.54 ± 1.36	136.1
	ALFA [35]	55.13 ± 1.53	53.39 ± 1.32	40.90 ± 1.44	65.00 ± 0.96	29.60 ± 1.71	92.00 ± 0.62	56.00 ± 1.26	113.4
	MeTAL [36]	56.10 ± 1.37	50.67 ± 1.69	42.42 ± 1.03	66.52 ± 1.87	31.86 ± 1.96	93.28 ± 1.12	56.81 ± 1.51	128.6
	CTML-feat	57.68 ± 1.50	53.73 ± 1.33	41.68 ± 1.42	65.10 ± 1.03	32.74 ± 1.62	91.88 ± 1.37	57.14 ± 1.38	84.3
	CTML-path	58.27 ± 0.98	53.64 ± 0.60	42.81 ± 1.13	65.81 ± 0.87	32.79 ± 1.06	93.12 ± 0.62	57.74 ± 0.88	171.2
	CTML	**60.51 ± 0.87**	**55.12 ± 0.54**	**43.61 ± 1.01**	67.12 ± 0.92	34.05 ± 0.98	93.78 ± 0.74	**59.03 ± 0.84**	181.0
	CTML(approx.)	60.12 ± 1.32	54.71 ± 0.72	43.19 ± 1.26	**67.52 ± 1.03**	33.95 ± 1.34	**93.95 ± 0.93**	58.91 ± 1.10	85.7
5-way 5-shot	MAML [5]	69.84 ± 0.89	63.64 ± 0.48	52.11 ± 0.52	75.35 ± 0.55	41.58 ± 0.83	96.72 ± 0.37	66.54 ± 0.61	-
	MMAML [17]	70.59 ± 0.72	65.29 ± 0.33	53.10 ± 0.29	76.90 ± 0.87	43.01 ± 0.86	98.04 ± 0.49	67.82 ± 0.59	-
	L2F [30]	71.23 ± 0.55	65.68 ± 0.55	54.79 ± 0.61	77.03 ± 0.44	44.32 ± 0.77	97.65 ± 0.30	68.45 ± 0.54	-
	HSML [24]	71.40 ± 0.59	67.31 ± 0.27	54.23 ± 0.29	76.32 ± 0.38	45.00 ± 0.67	98.12 ± 0.26	68.73 ± 0.41	-
	ARML [15]	71.13 ± 0.78	69.18 ± 0.27	54.39 ± 0.31	78.01 ± 0.45	44.93 ± 0.39	98.07 ± 0.28	69.29 ± 0.41	-
	TSA-MAML [27]	71.82 ± 0.62	71.65 ± 0.49	55.85 ± 0.35	80.98 ± 0.68	45.36 ± 0.44	98.54 ± 0.17	70.70 ± 0.46	-
	ModGrad [34]	72.64 ± 1.04	68.32 ± 0.72	52.77 ± 0.45	78.81 ± 0.91	43.26 ± 0.85	98.31 ± 0.64	69.02 ± 0.77	-
	ALFA [35]	71.01 ± 0.84	66.43 ± 0.62	54.01 ± 0.38	79.87 ± 0.79	45.97 ± 0.91	98.00 ± 0.47	69.22 ± 0.67	-
	MeTAL [36]	69.64 ± 0.51	64.36 ± 0.88	55.14 ± 0.71	78.48 ± 0.40	44.75 ± 0.52	97.02 ± 0.29	68.23 ± 0.55	-
	CTML-feat	71.60 ± 0.63	66.83 ± 0.31	53.71 ± 0.63	77.24 ± 0.77	45.07 ± 0.59	97.83 ± 0.17	68.71 ± 0.52	-
	CTML-path	72.43 ± 0.65	69.11 ± 0.46	56.14 ± 0.78	80.87 ± 0.80	46.24 ± 0.79	**98.97 ± 0.57**	70.63 ± 0.68	-
	CTML	**75.21 ± 0.35**	72.07 ± 0.40	**57.92 ± 0.57**	**81.78 ± 0.39**	**47.39 ± 0.62**	98.71 ± 0.19	**72.18 ± 0.42**	-
	CTML(approx.)	74.31 ± 0.62	**72.81 ± 0.73**	57.13 ± 0.34	80.78 ± 0.69	46.73 ± 0.83	98.40 ± 0.44	71.69 ± 0.61	-

different task-conditioning techniques: (1) common initialization method: MAML [5]; (2) task-adaptive initialization methods: MMAML (feature-based customization) [17] and L2F (gradient-based customization) [30]; (3) cluster-enhanced initialization methods: HSML (feature-based hierarchical clustering) [24], ARML (feature-based relational graph) [15] and TSA-MAML (optimization-based K-means clustering) [27]; (4) task-adaptive inner update methods: ModGrad (preconditioning on the inner gradients) [34], ALFA (meta-learning the inner update rule) [35] and MeTAL (meta-learning the inner loss function) [36]. For our proposed CTML, we report both the original meta-testing performance (CTML) and the one with shortcut approximation (CTML(approx.)). We further implement 2 variants to compare with the baselines: CTML-feat which only uses features for task representation and CTML-path which only uses path for task representation. Following [5], we adopt a 4-layer 3×3 convolutions with 32 filters (referred to as Conv-4) for both the feature extractor $\mathcal{E}(\cdot)$ and the base-learner f_θ for all the methods.

It was shown recently that, with a deeper backbone, the simple training paradigm of pre-training + fine-tuning can achieve very competitive performance [56], [57]. Hence, we further conduct experiments on ResNet-12 to compare our method against 3 non-MAML-based baselines: (1) Finetune: A method that simply pre-trains a feature extractor on the entire training dataset and then finetunes the classifier for each task; (2) Finetune-cosine: A modification of Finetune which employs cosine distance between the feature vector and the class vectors [57]; (3) Finetune-distill: A modification of Finetune that employs sequential self-distillation to improve the pre-training of feature extractor [56]. We follow [58] for the ResNet-12 implementation, which adopts (64-128-256-512) for the number of filters at the 4 blocks.

For the MAML-based methods, we follow the hyper-parameters settings in MAML [5], where the adaptation learning rate α is set to be 0.01, the meta-update learning rate β is 0.001, the number of adaptation steps τ is 5, the meta batch size $|\mathcal{B}|$ is 4, and the size of test set $|\mathcal{D}_{\mathcal{T}_i}^{te}|$ is 15. The number of meta-update iterations is set to be 60,000 for Mixture-of-Datasets and miniImageNet, 80,000 for tieredImageNet and Meta-Dataset, and 40,000 for CIFAR-FS. More details on hyper-parameters settings can be found in Appendix C.1, available in the online supplemental material.

2) Results on Mixture-of-Datasets: Table I presents the few-shot classification performance of the MAML-based methods on Mixture-of-Datasets. First, we see that all the task-adaptive methods outperform the vanilla MAML in this task-heterogeneous setting. L2F with gradient-based conditioning performs better than the feature-based MMAML. After considering generalization across similar tasks with some clustering techniques, HSML and ARML further improve over MMAML. TSA-MAML which uses the adapted model as task representation for clustering emerges as the strongest baseline in the 5-shot scenario. For the task-adaptive inner update methods (i.e., ModGrad, ALFA, MeTAL), their performance is only slightly better than L2F, which simply adjusts the initialization based on task gradients, demonstrating the importance of adapting the initialization as opposed to the subsequent update steps. For our proposed CTML, CTML-feat with K-means clustering on features exhibits comparable performance as HSML and ARML. For CTML-path, the improvements over the

TABLE II
FEW-SHOT CLASSIFICATION PERFORMANCE (MEAN ACCURACY (%) ± STD OVER 8 TRIALS) INCLUDING THE NON-MAML-BASED BASELINES FOR TWO BACKBONES: CONV-4 AND RESNET-12

Methods	Mixture-of-Datasets			
	5-way 1-shot		5-way 5-shot	
	Conv-4	ResNet-12	Conv-4	ResNet-12
Finetune	49.01 ± 0.68	56.19 ± 0.71	65.08 ± 0.52	71.11 ± 0.58
Finetune-cosine [57]	50.64 ± 0.98	57.59 ± 1.23	67.42 ± 0.82	72.23 ± 0.79
Finetune-distill [56]	51.03 ± 1.49	58.82 ± 0.91	68.17 ± 1.27	72.20 ± 0.77
MAML [5]	52.83 ± 1.54	56.84 ± 1.38	66.54 ± 1.02	69.84 ± 0.98
MMAML [17]	54.28 ± 1.21	59.12 ± 1.61	67.82 ± 0.56	71.92 ± 0.81
L2F [30]	56.61 ± 1.21	60.69 ± 1.28	68.45 ± 0.84	72.07 ± 0.60
HSML [24]	56.32 ± 1.16	61.57 ± 0.89	68.73 ± 0.72	73.35 ± 1.03
ARML [15]	57.21 ± 1.68	61.06 ± 1.19	69.29 ± 0.69	73.56 ± 0.70
TSA-MAML [27]	55.37 ± 1.30	61.67 ± 0.96	70.70 ± 0.46	74.10 ± 0.86
ModGrad [34]	55.54 ± 1.36	60.91 ± 1.37	69.02 ± 0.77	72.69 ± 0.75
ALFA [35]	56.00 ± 1.26	61.37 ± 1.54	69.22 ± 0.67	73.11 ± 0.69
MeTAL [36]	56.81 ± 1.51	60.98 ± 1.22	68.23 ± 0.55	73.80 ± 0.82
CTML	**59.03** ± 0.84	**63.18** ± 1.02	**72.18** ± 0.64	**75.03** ± 0.51
CTML(approx.)	58.91 ± 1.10	62.69 ± 1.32	71.69 ± 1.05	74.81 ± 0.92

feature-based baselines are more significant, especially for the 5-shot scenario where the learning path becomes more reliable with more training data for each task. The final CTML combining the benefits of both the feature and path representations achieves further improvements and outperforms the strongest baselines (ARML for 1-shot and TSA-MAML for 5-shot) by a significant margin.[6]

Due to the additional rehearsed learning, the meta-testing time of CTML and CTML-path is about twice that of the baselines (except for TSA-MAML, which also requires performing the adaptation twice for task representation). After applying the shortcut approximation, we are able to cut the inference time of CTML by half with just a small compromise on performance. Note that for ModGrad, ALFA and MeTAL, since they require additional procedures at each inner step, their extra time costs increase linearly with the number of inner steps, while for the other methods that only focus on the initialization, the extra time cost is fixed.

Table II presents the few-shot classification performance including the non-MAML-based baselines on Mixture-of-Datasets[7] for 2 backbones: Conv-4 and ResNet-12. We can see that for ResNet-12, the Finetune methods outperform MAML for both scenarios. This may be due to that the deeper model has a larger capacity to accommodate the transferable knowledge obtained from pre-training. Nevertheless, the task-adaptive MAML-based methods still yield better performance than the Finetune methods for ResNet-12, demonstrating the effectiveness of task-conditioning on a deeper backbone under the task-heterogeneous setting. Our method incorporating the path representation yields the best result. Besides, we note that a major drawback of the Finetune methods is that they often require a large number of steps to adapt the classifier to new tasks (i.e., the fine-tuning phase). In our experiments, it was found that the Finetune methods generally take more than 200 steps to converge for task adaptation, while for the MAML-based methods, only a few steps will suffice (within 10 steps). This phenomenon was also revealed in [55].

[6]Based on t-test, the improvements of CTML over all the baselines are considered significant at a level of 0.025

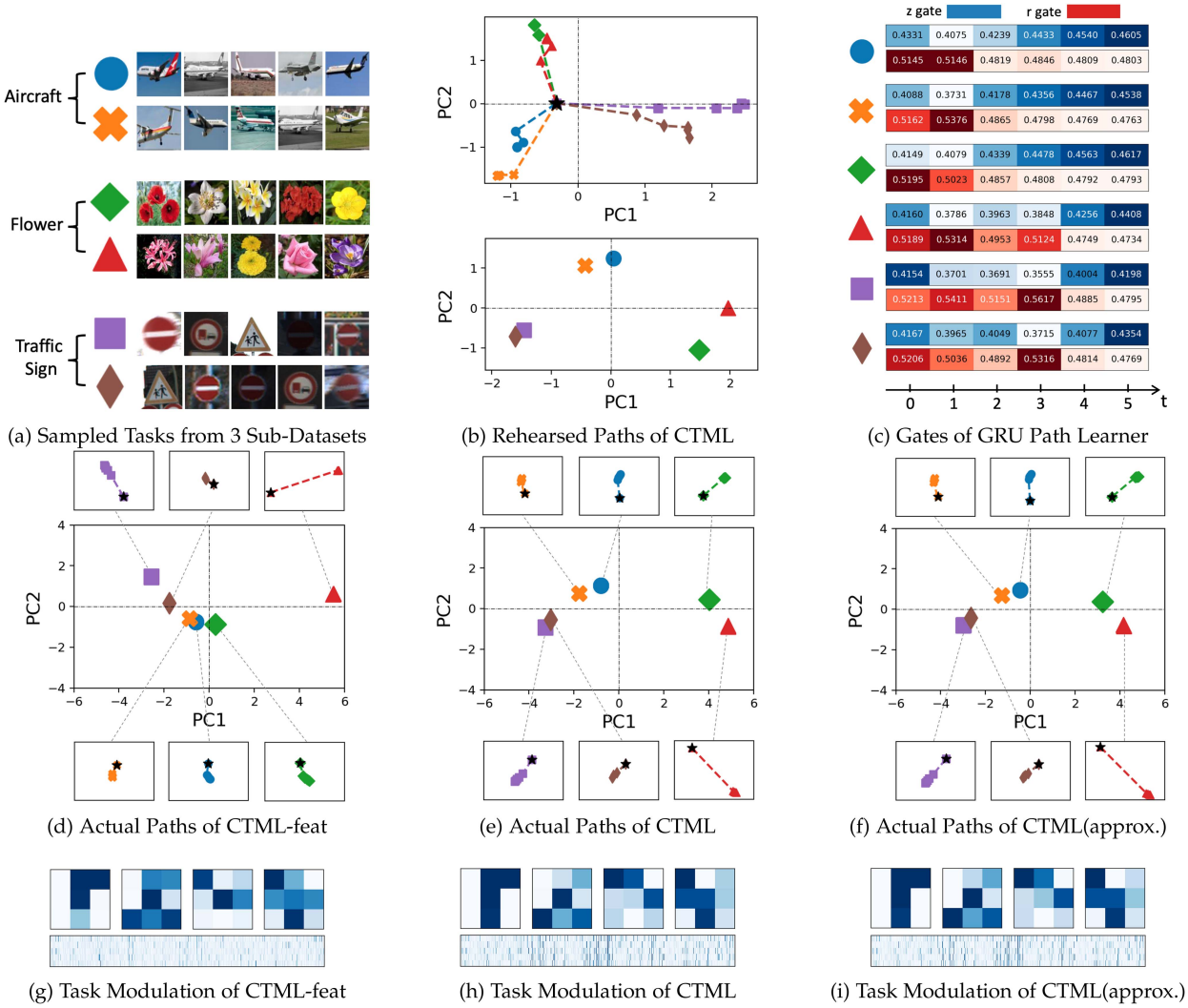
[7]Due to the limit of space, the performance of individual sub-datasets is omitted from Tables II and III

TABLE III
FEW-SHOT CLASSIFICATION PERFORMANCE (MEAN ACCURACY (%) ± STD OVER 8 TRIALS) OF VARIANTS OF CTML

Variants	Mixture-of-Datasets	
	5-way 1-shot	5-way 5-shot
Remove $\tilde{\theta}^t_{\mathcal{T}_i}$	56.74 ± 1.03	71.38 ± 0.52
Remove $\tilde{\mathcal{L}}^t_{\mathcal{T}_i}$	57.24 ± 1.20	71.50 ± 0.71
Remove $\nabla_\theta \tilde{\mathcal{L}}^t_{\mathcal{T}_i}$	57.89 ± 1.33	71.76 ± 0.61
Remove $\tilde{\mathcal{F}}^t_{\mathcal{T}_i}$	58.13 ± 1.12	71.88 ± 0.67
Linear Path Learner	57.52 ± 1.09	71.44 ± 0.47
FC Path Learner	58.37 ± 1.02	71.62 ± 0.39
Attention Path Learner	58.67 ± 1.40	71.83 ± 0.56
No Clustering	57.29 ± 1.24	71.06 ± 0.62
CTML	**59.03** ± 0.84	**72.18** ± 0.42

3) Ablation Study: We further conduct an ablation study on the Mixture-of-Datasets to test the efficacy of various CTML designs: the 4 step-wise input components, the GRU path learner design, and task clustering. Table III presents the results of different variants. First of all, we see that the lower-order components like $\tilde{\theta}^t_{\mathcal{T}_i}$ and $\tilde{\mathcal{L}}^t_{\mathcal{T}_i}$ seem to be more important than their higher-order counterparts. Second, to test the effectiveness of different path learner designs, we introduce 3 other design alternatives: a linear network, a non-linear fully-connected network, and an attention-based network. Detailed formulation of the 3 network designs can be found in Appendix D, available in the online supplemental material. From the results, we see that our proposed CTML with GRU path learner performs the best. This can be attributed to the ability of GRU to capture the sequential dependencies among the steps. Attention-based model is also a competitive candidate, as the positional encodings and the causal mask serve to inform about the sequential order. Lastly, we see that task clustering indeed helps boost performance, demonstrating the importance of explicitly considering the global structure for better generalization.

4) Visualization of Learning Paths: To understand the path learning mechanism, we randomly sample 2 meta-test tasks (5-way 1-shot) from each of the Aircraft, Flower, and Traffic Sign sub-datasets. The sampled tasks are shown in Fig. 3(a). In Fig. 3(b), we plot the 5-step rehearsed learning paths of these tasks and the corresponding path embeddings generated by the GRU path learner. We see that the learning paths of tasks from the same sub-dataset generally move along the same direction, and the corresponding path embeddings are able to encode this information and positioned closely for similar paths in the latent space. In Fig. 3(c), we further visualize the z and r gates produced by GRU at each rehearsed step t, controlling how much to retain for the current step and how much to retain for the previous step respectively. We can see that the GRU path learner is able to identify the "important" steps (darker-blue z gate blocks) and absorbs a larger portion of the inputs at these steps into the final embedding.



(a) Sampled Tasks from 3 Sub-Datasets (b) Rehearsed Paths of CTML (c) Gates of GRU Path Learner

(d) Actual Paths of CTML-feat (e) Actual Paths of CTML (f) Actual Paths of CTML(approx.)

(g) Task Modulation of CTML-feat (h) Task Modulation of CTML (i) Task Modulation of CTML(approx.)

Fig. 3. Visualization of learning paths. (a) 6 meta-test tasks (5-way 1-shot) randomly sampled from the **Aircraft**, **Flower**, and **Traffic Sign** sub-datasets. (b) PCA visualization (on the first two principal components) of the 5-step rehearsed learning paths (upper subplot) and the corresponding path embeddings (lower subplot) for the 6 tasks. (c) Visualization of z gate (how much to retain for the current step input) and r gate (how much to retain for the previous step memory) of GRU at each rehearsed step t for the 6 tasks. (d)-(f) PCA visualization of the modulated initializations and the 5-step actual learning paths for the 6 tasks generated by CTML-feat, CTML and CTML(approx.) respectively. The initial point of the path in each mini-plot is indicated by the "black star". (g)-(i) Visualization of task modulation on 4 randomly selected filters (size of 3×3) from the 4 convolutional blocks and the final read-out layer (size of 800×5) for the first task (i.e., blue circle task) generated by CTML-feat, CTML, and CTML(approx.) respectively.

From Fig. 3(d) to (f), we plot the actual learning paths of the 6 tasks with initializations modulated by CTML-feat, CTML and CTML(approx.) respectively. Since the distance spanned by the actual learning paths is much shorter than the separation between the modulated initializations, we zoom in for each task (to the same scale for all the tasks in all the plots for easier comparisons) to visualize the actual learning paths. The initial point of the path in each mini-plot is indicated by the "black star". First, we see that the initializations of CTML exhibit a finer clustering pattern than CTML-feat, signifying the benefits of incorporating the path information. Second, CTML(approx.) produces very similar modulated initializations and actual learning paths as CTML, showing the efficacy of the shortcut approximation. In Fig. 3(g) to (i), we extract the modulations on 4 randomly selected filters (size of 3×3) and the final read-out layer (size of 800×5) for the first task (i.e., blue circle task). It is evident that the task modulation of CTML(approx.) highly resembles that of CTML, while CTML-feat produces a rather different task modulation from the other two. More visualization examples can be found in Appendix E.1, available in the online supplemental material.

5) Results on Public Benchmarks: **Task-Homogeneous Benchmarks** We further conduct experiments to compare the MAML-based methods on miniImageNet [7], tieredImageNet [53] and CIFAR-FS [54], which are public benchmarks following task-homogeneous setting. In this section, we further include the more challenging 10-way 1-shot scenario following [27]. From Table IV, we see that the improvements of the task-adaptive methods over the common initialization methods (i.e., MAML, fo-MAML, Reptile) are less significant here as compared to those on Mixture-of-Datasets (Table I). Nevertheless, even under this task-homogeneous setting where the

TABLE IV
FEW-SHOT CLASSIFICATION PERFORMANCE OF MAML-BASED METHODS ON MINIIMAGENET, TIEREDIMAGENET, AND CIFAR-FS. THE RESULTS ARE REPORTED IN THE FORM OF 95% CONFIDENCE INTERVAL OF ACCURACY (%) BASED ON 600 META-TESTING TASKS. ALL RESULTS ARE ADOPTED DIRECTLY FROM [27], EXCEPT FOR THOSE DENOTED WITH *, WHICH ARE OBTAINED FROM OUR REPRODUCTION

Methods	miniImageNet			tieredImageNet			CIFAR-FS		
	5-way 1-shot	5-way 5-shot	10-way 1-shot	5-way 1-shot	5-way 5-shot	10-way 1-shot	5-way 1-shot	5-way 5-shot	10-way 1-shot
MAML [5]	48.26 ± 1.84	63.11 ± 0.92	31.31 ± 0.52	50.48 ± 1.81	68.06 ± 1.75	34.25 ± 1.19	57.46 ± 0.90	72.75 ± 0.71	39.97 ± 0.56
fo-MAML [5]	48.07 ± 1.75	63.15 ± 0.91	23.16 ± 0.94	50.12 ± 1.82	67.43 ± 1.80	31.53 ± 1.08	49.30 ± 1.18	66.96 ± 1.27	37.83 ± 1.06
Reptile [6]	49.97 ± 0.32	65.99 ± 0.58	30.41 ± 0.83	51.06 ± 0.45	66.30 ± 0.78	33.79 ± 0.29	54.03 ± 0.92	72.60 ± 0.83	38.41 ± 0.97
MMAML [17]	49.83 ± 1.83	63.73 ± 0.91	31.25 ± 0.45	48.52 ± 0.47	64.39 ± 0.47	33.69 ± 0.35	45.16 ± 0.58	58.56 ± 0.51	27.30 ± 0.25
L2F* [30]	50.38 ± 1.72	64.06 ± 0.89	32.33 ± 0.46	50.91 ± 0.87	67.40 ± 0.75	35.21 ± 0.56	54.60 ± 0.97	73.05 ± 0.71	42.29 ± 0.60
HSML [24]	50.38 ± 1.85	64.03 ± 0.90	31.84 ± 0.46	48.82 ± 0.86	66.74 ± 0.76	34.63 ± 0.55	54.71 ± 1.50	69.62 ± 1.01	38.49 ± 1.22
ARML* [15]	50.42 ± 1.79	64.12 ± 0.90	31.72 ± 0.46	50.21 ± 0.82	68.79 ± 0.71	35.33 ± 0.56	54.18 ± 0.94	73.67 ± 0.70	42.13 ± 0.59
TSA-MAML [27]	49.22 ± 0.92	**66.26 ± 0.92**	32.13 ± 0.44	52.03 ± 0.86	68.97 ± 0.74	35.78 ± 0.58	58.21 ± 0.93	73.52 ± 0.72	42.18 ± 0.58
ModGrad* [34]	48.48 ± 0.86	63.90 ± 0.65	30.60 ± 0.45	51.36 ± 0.87	66.53 ± 0.79	33.80 ± 0.55	53.53 ± 0.89	73.72 ± 0.71	40.89 ± 0.74
ALFA* [35]	48.88 ± 0.81	62.87 ± 0.69	31.59 ± 0.45	**52.25 ± 0.85**	**70.46 ± 0.75**	36.08 ± 0.58	57.10 ± 0.92	72.00 ± 0.75	42.25 ± 0.62
MeTAL* [36]	48.76 ± 0.79	63.65 ± 0.67	31.50 ± 0.45	50.00 ± 0.81	67.55 ± 0.74	34.20 ± 0.56	55.54 ± 0.95	73.45 ± 0.73	40.62 ± 0.57
CTML	**50.83 ± 1.83**	66.13 ± 0.92	**32.87 ± 0.47**	51.84 ± 0.90	69.62 ± 0.74	**36.14 ± 0.57**	**58.58 ± 0.92**	**74.32 ± 0.71**	**42.74 ± 0.63**

TABLE V
PERFORMANCE ON META-DATASET IN THE FORM OF 95% CONFIDENCE INTERVAL OF ACCURACY (%) FOR FO-MAML AND FO-PROTO-MAML BEFORE AND AFTER APPLYING CTML. MODELS ARE TRAINED ON THE FIRST 8 SUB-DATASETS AND TESTED ON THE META-TEST SPLITS OF ALL THE 10 SUB-DATASETS. COMPARISONS WITH THE STATE-OF-THE-ART METHODS CAN BE FOUND IN APPENDIX E.3, AVAILABLE IN THE ONLINE SUPPLEMENTAL MATERIAL

Sub-Datasets	Meta-Dataset			
	fo-MAML		fo-Proto-MAML	
		+ CTML		+ CTML
ILSVRC	37.83 ± 1.01	**52.33 ± 1.18**	46.52 ± 1.05	**56.64 ± 1.08**
Omniglot	83.92 ± 0.95	**87.16 ± 1.46**	82.69 ± 0.97	**93.60 ± 1.35**
Aircraft	76.41 ± 0.69	**76.71 ± 1.19**	75.23 ± 0.76	**79.88 ± 1.03**
Birds	62.43 ± 1.08	**69.68 ± 1.31**	69.88 ± 1.02	**75.59 ± 1.21**
Textures	**64.14 ± 0.83**	61.67 ± 1.08	68.25 ± 0.81	**70.56 ± 0.96**
Quick Draw	59.73 ± 1.10	**69.00 ± 1.10**	66.84 ± 0.94	**80.25 ± 1.19**
Fungi	33.54 ± 1.11	**38.48 ± 0.97**	41.99 ± 1.17	**64.99 ± 0.94**
VGG Flower	79.94 ± 0.84	**83.85 ± 0.81**	**88.72 ± 0.67**	87.18 ± 0.75
Traffic Sign	42.91 ± 1.31	**58.49 ± 1.52**	52.42 ± 1.08	**64.78 ± 1.31**
MSCOCO	29.37 ± 1.08	**29.98 ± 1.02**	41.74 ± 1.13	**53.39 ± 1.07**

advantages of task-conditioning are rendered less prominent, our proposed CTML still emerges as a strong candidate among the state-of-the-art MAML-based methods, achieving the best performance in 6 out of 9 scenarios.

Task-Heterogeneous Benchmark. Meta-Dataset [55] is recently introduced to simulate the more realistic task-heterogeneous setting. This benchmark involves 10 sub-datasets, where the model is trained on 8 sub-datasets and tested on all 10 sub-datasets to enable both in-distribution and out-distribution evaluations. Though it is a less suitable benchmark for evaluating MAML-based methods (for the reason mentioned earlier), to have a rough idea of how CTML is positioned among the state-of-the-art meta-learning methods on this task-heterogeneous benchmark, we apply CTML on top of fo-MAML and fo-Proto-MAML proposed in [55]. That is, we do not meta-learn the initialization of the classification layer and only meta-learn the modulation on the initialization obtained by fo-MAML or fo-Proto-MAML.

Table V shows the results of fo-MAML and fo-Proto-MAML before and after applying CTML. We see that for both cases, applying CTML significantly boosts the model performance, demonstrating its effectiveness in this challenging setup where the classification layer cannot be meta-learned. In Appendix E.3, available in the online supplemental material, we include a table to compare fo-Proto-MAML + CTML (in short, Proto-CTML) with 8 latest state-of-the-art methods on Meta-Dataset. Proto-CTML emerges as the third among these strong candidates.

B. Cold-Start Recommendation

In the field of recommender systems, meta-learning has been applied to solve the cold-start problem, which refers to making recommendations related to new items or new users [59], [60], [61]. Here, we focus on the user cold-start problem, where making recommendations for each user is treated as a task. This setting is task-heterogeneous in nature as users may belong to different preference groups. We conduct experiments on 3 public benchmark datasets: MovieLens-1M [62], Yelp [63], and Amazon-CDs [64], consisting of user ratings on movies, business services, and CD products respectively. We split each dataset into meta-train/validation/test by a ratio of 7/1/2 according to the timestamps of ratings. Note that it is possible for meta-train users to appear in meta-validation and meta-test sets, we refer to these users as **warm users**, and those have never appeared in the meta-train set as **cold users**. For each user, we use the first 10 samples as the training set (i.e., support set), and the rest as the test set (i.e., query set). More details about datasets can be found in Appendix B.2, available in the online supplemental material.

1) Baselines and Our Method: We choose four baselines that specifically tackle the user cold-start problem in recommender systems developed based on MAML. MeLU [61] is the first to adopt MAML in recommender systems with locally adapted decision layers; MetaHIN [65] is a MAML-based method that further incorporates Heterogeneous Information Networks (HIN) for data augmentation and multi-faceted adaptations; MAMO [25] involves a global memory module to group users based on profile information and personalizes the initial bias; PAML [66] personalizes the adaptation learning rate to allow for better fitting of the minor users. Following the baselines, we implement the feature extractor $\mathcal{E}(\cdot)$ as several embedding lookup matrices for different features, and the base-learner f_θ as a general recommender system model consists of an embedding layer followed by multiple fully-connected layers (i.e., the decision layers) [61].

TABLE VI
RECOMMENDATION PERFORMANCE ON 3 DATASETS IN MAE AND NDCG@20. WE REPORT THE MEAN OVER 5 TRIALS, WHERE THE STD DEVS ARE ALL LESS THAN 0.001. APPENDIX E.4, AVAILABLE IN THE ONLINE SUPPLEMENTAL MATERIAL, INCLUDES THE FULL DISENTANGLED RESULTS FOR WARM & COLD USERS

Metrics	Methods	MovieLens-1M	Yelp	Amazon-CDs
MAE ↓	MeLU [61]	0.8164	0.9464	0.7216
	PAML [66]	0.7725	0.9280	0.7150
	MAMO [25]	0.8084	0.8965	0.7207
	MetaHIN [65]	0.7727	0.8832	0.6743
	CTML-feat	0.7560	0.9023	0.6804
	CTML-path	0.7592	0.8714	0.6286
	CTML	**0.7543**	**0.8603**	**0.6048**
	CTML(approx.)	0.7555	0.8679	0.6087
NDCG@20 ↑	MeLU [61]	0.7839	0.8149	0.9026
	PAML [66]	0.8255	0.8215	0.9083
	MAMO [25]	0.8118	0.8332	0.9116
	MetaHIN [65]	0.8264	0.8375	0.9111
	CTML-feat	0.8470	0.8295	0.9122
	CTML-path	0.8493	0.8401	0.9201
	CTML	**0.8562**	**0.8467**	**0.9279**
	CTML(approx.)	0.8531	0.8433	0.9244

For all the compared methods, we follow the hyper-parameters settings in MeLU [61], where the adaptation learning rate α is set to be 5e-3, the meta-update learning rate β is 5e-5, the number of adaptation steps τ is 5, and the number of meta-training epochs as 20. Detailed hyper-parameters settings can be found in Appendix C.2, available in the online supplemental material.

2) Results: We evaluate the recommendation performance of different methods in terms of two metrics: (1) Mean Absolute Error (MAE): A measure of rating prediction errors; (2) Normalized Discounted Cumulative Gain (NDCG): A measure of ranking quality based on the predicted ratings. NDCG is computed at the top 20 items in the predicted ranking (termed NDCG@20). Note that smaller MAE and larger NDCG@20 indicate better performance.

From the results in Table VI, we can see that PAML and MAMO with personalized adaptation rate and initialization achieve better performance than MELU. Our CTML surpasses all the baselines, including MetaHIN which leverages HIN for data augmentation. Furthermore, we notice that even CTML-path is able to outperform all the baselines by a significant margin. This suggests that the learning process of users toward better prediction can be more useful than the profile features in terms of representing user preference. Further including features on top of learning path representations results in marginal improvements (compare CTML-path with CTML). This finding is meaningful as in reality, user profile information may not always be available. Learning from learning path becomes a promising alternative for user representation to address the cold-start problem when the side information is seriously lacking or inaccurate.

3) Number of Clusters: For recommendation problem, there is no ground-truth grouping of users. In Fig. 4(a), we investigate the effect of varying the number of clusters k_{path} and k_{feat} for MovieLens-1 M dataset. We see that the best performance occurs at smaller k_{feat} values and larger k_{path} values, which implies greater complexity for clustering the learning paths.

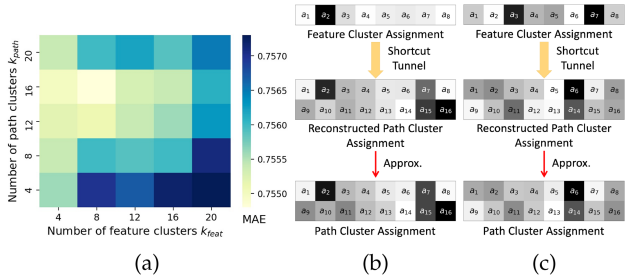


Fig. 4. (a) Effect of varying k_{feat} and k_{path} for MovieLens-1 M. (b)-(c) Shortcut approximation for 2 users randomly sampled from MovieLens-1 M ($k_{feat} = 8$ and $k_{path} = 16$).

4) Shortcut Approximation: Our design of the shortcut tunnel allows flexibility to map between different numbers of clusters (i.e., $k_{path} \neq k_{feat}$). In Fig. 4, we visualize the shortcut approximation of path assignment (over 16 clusters) from feature assignment (over 8 clusters) for 2 users randomly sampled from the MovieLens-1 M dataset. We can see that the shortcut tunnel is rather reliable as the reconstructed path assignment highly resembles the actual path assignment. More examples of few-shot image classification are provided in Appendix E.2, available in the online supplemental material.

VI. CONCLUSION

In this work, we introduce a CTML framework that leverages both features and learning path to enhance the task representation. A GRU-based meta path learner is employed to automatically extract the relevant knowledge from path and generate the path representation. To speed up the inference process, we introduce a shortcut tunnel to bypass the rehearsed learning during meta-testing. Extensive experiments demonstrate the effectiveness of CTML. In the future, we will extend the framework to other meta-learning algorithms and explore different designs of meta path learner to further improve the effectiveness of path modeling.

REFERENCES

[1] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Learning to Learn*. Berlin, Germany: Springer, 1998, pp. 3–17.
[2] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3637–3645.
[3] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4080–4090.
[4] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1199–1208.
[5] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[6] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018, *arXiv:1803.02999*.
[7] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. Int. Conf. Learn. Representations*, 2016.
[8] M. Andrychowicz et al., "Learning to learn by gradient descent by gradient descent," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3988–3996.
[9] M. Garnelo et al., "Conditional neural processes," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1704–1713.

[10] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *Proc. Int. Conf. Learn. Representations*, 2018.
[11] M. Boudiaf, Z. I. Masud, J. Rony, J. Dolz, P. Piantanida, and I. B. Ayed, "Transductive information maximization for few-shot learning," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 2445–2457.
[12] S. X. Hu et al., "Empirical bayes transductive meta-learning with synthetic gradients," in *Proc. Int. Conf. Learn. Representations*, 2019.
[13] Z. Yue, H. Zhang, Q. Sun, and X.-S. Hua, "Interventional few-shot learning," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 2734–2746.
[14] A. Devos and Y. Dandi, "Model-agnostic learning to meta-learn," 2020, *arXiv:2012.02684*.
[15] H. Yao et al., "Automated relational meta-learning," in *Proc. Int. Conf. Learn. Representations*, 2020.
[16] Q. Suo, J. Chou, W. Zhong, and A. Zhang, "Tadanet: Task-adaptive network for graph-enriched meta-learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1789–1799.
[17] R. Vuorio, S.-H. Sun, H. Hu, and J. J. Lim, "Multimodal model-agnostic meta-learning via task-aware modulation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–12.
[18] B. N. Oreshkin, P. R. López, and A. Lacoste, "TADAM: Task dependent adaptive metric for improved few-shot learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 719–729.
[19] J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner, "Fast and flexible multi-task classification using conditional neural adaptive processes," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 7957–7968.
[20] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 9537–9548.
[21] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn, "Bayesian model-agnostic meta-learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 7343–7353.
[22] A. A. Rusu et al., "Meta-learning with latent embedding optimization," in *Proc. Int. Conf. Learn. Representations*, 2018.
[23] R. Wang, Y. Demiris, and C. Ciliberto, "Structured prediction for conditional meta-learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 2587–2598.
[24] H. Yao, Y. Wei, J. Huang, and Z. Li, "Hierarchically structured meta-learning," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 7045–7054.
[25] M. Dong, F. Yuan, L. Yao, X. Xu, and L. Zhu, "MAMO: Memory-augmented meta-optimization for cold-start recommendation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 688–697.
[26] X. Lin, J. Wu, C. Zhou, S. Pan, Y. Cao, and B. Wang, "Task-adaptive neural process for user cold-start recommendation," 2021, *arXiv:2103.06137*.
[27] P. Zhou, Y. Zou, X.-T. Yuan, J. Feng, C. Xiong, and S. Hoi, "Task similarity aware meta learning: Theory-inspired improvement on MAML," in *Proc. 37th Conf. Uncertainty Artif. Intell.*, 2021, pp. 23–33.
[28] A. Achille et al., "Task2Vec: Task embedding for meta-learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6430–6439.
[29] F. Mu, Y. Liang, and Y. Li, "Gradients as features for deep representation learning," 2020, *arXiv:2004.05529*.
[30] S. Baik, S. Hong, and K. M. Lee, "Learning to forget for meta-learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2379–2387.
[31] S. Flennerhag, P. G. Moreno, N. D. Lawrence, and A. Damianou, "Transferring knowledge across learning processes," in *Proc. Int. Conf. Learn. Representations*, 2018.
[32] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3987–3995.
[33] H. Li, W. Dong, X. Mei, C. Ma, F. Huang, and B.-G. Hu, "LGM-net: Learning to generate matching networks for few-shot learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3825–3834.
[34] C. Simon, P. Koniusz, R. Nock, and M. Harandi, "On modulating the gradient for meta-learning," in *Proc. 16th Eur. Conf. Comput. Vis.*, Glasgow, U.K., 2020, pp. 556–572.
[35] S. Baik, M. Choi, J. Choi, H. Kim, and K. M. Lee, "Meta-learning with adaptive hyperparameters," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 20 755–20 765.
[36] S. Baik, J. Choi, H. Kim, D. Cho, J. Min, and K. M. Lee, "Meta-learning with task-adaptive loss function for few-shot learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9465–9474.
[37] Y. Liu, B. Schiele, and Q. Sun, "An ensemble of epoch-wise empirical bayes for few-shot learning," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 404–421.
[38] M. A. Zinkevich, A. Davies, and D. Schuurmans, "Holographic feature representations of deep networks," in *Proc. 37th Conf. Uncertainty Artif. Intell.*, 2017.
[39] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, 2017.
[40] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 139–154.
[41] J. R. Garcia et al., "Meta-learning with MAML on trees," 2021, *arXiv:2103.04691*.
[42] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Rev.*, vol. 60, no. 2, pp. 223–311, 2018.
[43] H. Park, S.-I. Amari, and K. Fukumizu, "Adaptive natural gradient learning algorithms for various stochastic models," *Neural Netw.*, vol. 13, no. 7, pp. 755–764, 2000.
[44] M. Riemer et al., "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *Proc. Int. Conf. Learn. Representations*, 2018.
[45] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," 2020, *arXiv:2001.06782*.
[46] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.
[47] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-UCSD birds-200–2011 dataset," California Institute of Technology, 2011. [Online]. Available: https://www.vision.caltech.edu/datasets/cub_200_2011/
[48] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," 2013, *arXiv:1306.5151*.
[49] B. Schroeder and Y. Cui, "FGVCx fungi classification challenge 2018," 2011. [Online]. Available: https://github.com/visipedia/fgvcx_fungi_comp
[50] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis. Graph. Image Process.*, 2008, pp. 722–729.
[51] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3606–3613.
[52] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man versus. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Netw.*, vol. 32, pp. 323–332, 2012.
[53] M. Ren et al., "Meta-learning for semi-supervised few-shot classification," 2018, *arXiv:1803.00676*.
[54] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," 2018, *arXiv:1805.08136*.
[55] E. Triantafillou et al., "Meta-dataset: A dataset of datasets for learning to learn from few examples," in *Proc. Int. Conf. Learn. Representations*, 2019.
[56] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: A good embedding is all you need?," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 266–282.
[57] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *Proc. Int. Conf. Learn. Representations*, 2018.
[58] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 403–412.
[59] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6907–6917.
[60] M. Volkovs, G. Yu, and T. Poutanen, "DropoutNet: Addressing cold start in recommender systems," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4964–4973.
[61] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "MeLU: Meta-learned user preference estimator for cold-start recommendation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 1073–1082.
[62] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. InterAct. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2015.
[63] Y. Dataset, "Yelp dataset challenge," 2019. [Online]. Available: https://www.yelp.com/dataset
[64] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 43–52.
[65] Y. Lu, Y. Fang, and C. Shi, "Meta-learning on heterogeneous information networks for cold-start recommendation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1563–1573.
[66] R. Yu et al., "Personalized adaptive meta learning for cold-start user preference prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 10 772–10 780.

Danni Peng received the BEng (Hons.) degree in industrial systems engineering from the National University of Singapore, in 2019. She is currently working towards the PhD degree in alibaba talent programme with Alibaba-NTU Joint Research Institute, Nanyang Technological University, Singapore. Her research interests include meta-learning and its applications.

Sinno Jialin Pan received the PhD degree in computer science from the Hong Kong University of Science and Technology (HKUST), in 2011. He is a professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong (CUHK), Hong Kong. Prior to joining CUHK, he was a Provost's chair professor of computer science with NTU Singapore. He serves as an associate editor for *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *Artificial Intelligence Journal*, and *ACM Transactions on Intelligent Systems and Technology*.