Clustered Task-Aware Meta-Learning by Learning from Learning Paths (Supplementary Material)

Danni Peng, Sinno Jialin Pan

Appendix A
On the Connection between Learning Path and Conditional Distribution

Since the learning path of a task is directed by the labels (i.e., learning proceeds in the direction that optimizes the loss), unlike the feature embeddings, which if plotted for all the tasks in the latent space will form the marginal distribution $p(\mathbf{x})$, the path embeddings in the latent space will constitute the conditional distribution $p_\theta(y|\mathbf{x})$.

To illustrate this point, consider a simple example of binary classification problem. Suppose each data instance i consists of a 2-D feature $\mathbf{x}^{(i)} = [x_1, x_2]$ and a label $y^{(i)} \in \{0, 1\}$. We apply the logistic model $\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}$, where $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2$, and minimize the log loss $l = -y \log(\hat{y}) + (1 - y) \log(\hat{y})$. To further simplify the case, we assume each task consists of only one instance (that is, each instance is treated as a task). Consider 3 tasks $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$ with exactly the same feature $[x_1 = 2, x_2 = 4]$ but different label y, we can compute the derivatives of l w.r.t. β_0, β_1 and β_2 as follows:

$\mathcal{T}_i$	x_1	x_2	y	$\frac{\partial l}{\partial \beta_0}$	$\frac{\partial l}{\partial \beta_1}$	$\frac{\partial l}{\partial \beta_2}$
1	2	4	0	$\frac{1}{1+e^z}$	$\frac{2}{1+e^z}$	$\frac{4}{1+e^z}$
2	2	4	0	$\frac{1}{1+e^z}$	$\frac{2}{1+e^z}$	$\frac{4}{1+e^z}$
3	2	4	1	$\frac{-1}{1+e^z}$	$\frac{-2}{1+e^z}$	$\frac{-4}{1+e^z}$

Since the learning path follows the gradient descent direction, we can see that $\mathcal{T}_1$ and $\mathcal{T}_2$ will have exactly the same learning paths, while $\mathcal{T}_3$ will have a learning path that points in opposite direction. Hence, the path embeddings generated for $\mathcal{T}_1$ and $\mathcal{T}_2$ will be positioned together in the latent space, while $\mathcal{T}_3$ will be positioned far apart. The overall 3-point distribution in the latent space essentially describes the conditional distribution where $p(y = 0|x_1 = 2, x_2 = 4) = \frac{2}{3}$ and $p(y = 1|x_1 = 2, x_2 = 4) = \frac{1}{3}$. On the other hand, the feature embeddings which are not informed by the labels will position all the 3 tasks together, representing the marginal distribution $p(x_1 = 2, x_2 = 4) = 1$.

• *Danni Peng and Sinno Jialin Pan are with School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail: danni001@e.ntu.edu.sg, sinnopan@ntu.edu.sg*

Appendix B
Datasets and Pre-Processing Details

B.1 Image Datasets

In our experiments, Mixture-Of-Datasets consist of six widely used benchmark datasets. For the CUB-200-2011 (Bird), FGVC-Aircraft (Aircraft), FGVCx-Fungi (Fungi), and DTD (Texture) datasets, we directly acquire the pre-processed datasets released by the authors of HSML[1] [1]. Pre-processing details of these four sub-datasets can be found in the appendix of [1]. We have obtained the authors' consent to use this asset for our research.

For the VGG Flower (Flower)[2] and GTSRB (Traffic Sign)[3] datasets, we download the raw datasets from the respective official websites. We follow the pre-processing protocols in [2] to resize all the images to $84 \times 84 \times 3$. We also try to maintain a similar meta-train/validation/test ratio as in [1] for class splits. For VGG Flower, there are in total 102 flower classes, and each class consists of 40 $\sim$ 258 images. For GTSRB, we use the official training data which contains 43 traffic sign classes, and each class has 30 images.

To facilitate comparisons with the existing meta-learning methods, we also conduct experiments on some common benchmarks. For the task-homogeneous setting, we conduct experiments on 3 popular few-shot benchmarks: miniImageNet [3], tieredImageNet [4] and CIFAR-FS [5]. miniImageNet contains 100 classes from ILSVRC-12 dataset [6], where each class contains 600 images of size $84 \times 84 \times 3$. tieredImageNet is also constructed from ILSVRC-12, containing 608 classes with 600 images per class. It further groups the 608 classes into 34 higher-level categories and splits them into meta-train/validation/test categories by 20/6/8. This ensures that the classes in different splits are sufficiently distinct, making it more challenging for prediction. CIFAR-FS is a few-shot benchmark constructed from CIFAR-100 [7], which contains 100 classes with 600 images of size $32 \times 32 \times 3$ per class. For miniImageNet, we acquire the dataset from [3]. For tieredImageNet and CIFAR-FS, we use the pre-processed datasets released by the authors of TSA-MAML[4] [8]. Table 1 records the meta-train/validation/test splits for Mixture-Of-Datasets and the 3 task-homogeneous benchmarks.

1. https://github.com/huaxiuyao/HSML
2. https://www.robots.ox.ac.uk/~vgg/data/flowers/
3. https://benchmark.ini.rub.de/
4. https://github.com/Carbonaraa/TSA-MAML

TABLE 1: Meta-train/validation/test splits for Mixture-Of-Datasets and the 3 task-homogeneous benchmarks.

Datasets	#Train Class	#Val Class	#Test Class
CUB-200-2011 (Bird) [9]	64	16	20
FGVC-Aircraft (Aircraft) [10]	64	16	20
FGVCx-Fungi (Fungi) [11]	64	16	24
VGG Flower (Flower) [12]	64	16	22
DTD (Texture) [13]	30	7	10
GTSRB (Traffic Sign) [14]	26	7	10
miniImageNet [3]	64	16	20
tieredImageNet [4]	351	97	160
CIFAR-FS [5]	64	16	20

For task-heterogeneous setting, we test our method on Meta-Dataset [15]. Meta-Dataset is a recently introduced large-scale benchmark consisting of 10 sub-datasets: ILSVRC-12 [6], Omniglot [16], FGVC-Aircraft (Aircraft) [10], CUB-200-2011 (Bird) [9], DTD (Texture) [13], Quick Draw [17], FGVCx-Fungi (Fungi) [11], VGG Flower (Flower) [12], GTSDB (Traffic Sign) [18] and MSCOCO [19], where GTSDB (Traffic Sign) and MSCOCO are fully reserved for evaluation. Though the sub-datasets of Mixture-Of-Datasets largely overlap with those of Meta-Dataset, we decided not to rely on Meta-Dataset to perform the main evaluations and comparisons with other relevant MAML-based baselines, as Meta-Dataset adopts a pipeline that constructs tasks with variable number of ways which is unfavorable to MAML-based methods. More details on task sampling can be found in the original paper [15]. The meta-train/validation/test splits of each sub-dataset of Meta-Dataset roughly follow the ratio 70%/15%/15%. For ILSVRC-12 and Omniglot, a class hierarchy is exploited in determining the splits so that the meta-train classes are semantically different from the meta-test classes. More details about the datasets can be found in the paper and their released code[5].

B.2 Recommendation Datasets

B.2.1 MovieLens-1M

MovieLens-1M[6] dataset [20] contains 1,000,209 anonymous ratings on around 3,900 movies made by 6,040 users. We acquire the additional side information released by the authors of MeLU[7] [23], which contains user profile information of gender, age group, occupation, zipcode, and movie profile information of rate, genre, director and actor.

The dataset is split into meta-train/validation/test sets according to the timestamps of ratings by ratio 7/1/2. As a result, the start time is 2000-04-26, the meta-train cut-off time is 2000-11-22, the meta-validation cut-off time is 2000-12-02, and the end time is 2003-03-01. Users with less than 30 samples in each split are removed. In the end, the average sample size for each user is 163. The statistics of the preprocessed MovieLens-1M dataset are recorded in Table 2. The number of meta-train/validation/test users is shown in Table 3.

5. https://github.com/google-research/meta-dataset
6. https://grouplens.org/datasets/movielens/
7. https://github.com/hoyeoplee/MeLU

TABLE 2: Statistics of recommendation datasets.

Datasets	Features	#IDs
MovieLens-1M [20]	User	5,255
	Gender	2
	Age Group	7
	Occupation	21
	Zipcode	3,402
	Movie	3,697
	Rate	6
	Genre	25
	Director	2,186
	Actor	8,030
Yelp [21]	User	25,206
	Business Service	159,611
	Category	1,317
	City	1,023
Amazon-CDs [22]	User	3,876
	CD	132,209
	Category	490
	Brand	49,283

TABLE 3: Number of meta-train/validation/test users for the three recommendation datasets.

Datasets	#Train User	#Val User			#Test User		
		All	Warm	Cold	All	Warm	Cold
MovieLens-1M [20]	4,119	621	122	499	1,229	485	744
Yelp [21]	22,822	1,281	681	600	3,442	1,441	2,001
Amazon-CDs [22]	3,584	202	110	92	355	119	236

B.2.2 Yelp

Yelp[8] dataset [21] contains 8,635,403 reviews for 160,585 business services made by 2,189,457 users. We extract the star ratings from the reviews for our experiments. For each business service, we collect the city it resides in and the categories it belongs to from the business metadata. However, for the user, the personal information is lacking, and we only have access to the user ID.

The dataset is split into meta-train/validation/test sets according to the timestamps of ratings by ratio 7/1/2. As a result, the start time is 2004-10-12, the meta-train cut-off time is 2018-02-07, the meta-validation cut-off time is 2018-09-04, and the end time is 2019-12-13. Users with less than 30 samples in each split are removed. In the end, the average sample size for each user is 89. The statistics of the preprocessed Yelp dataset are recorded in Table 2. The number of meta-train/validation/test users is shown in Table 3.

B.2.3 Amazon-CDs

Amazon-CDs[9] dataset [22] contains 3,749,004 ratings for 544,442 CD and Vinyl products made by 1,578,597 users. For each CD, we collect its brand and category, while for the user, the personal profile information is lacking and we only have access to the user ID.

The dataset is split into meta-train/validation/test sets according to the timestamps of ratings by ratio 7/1/2. As

8. https://www.yelp.com/dataset
9. https://jmcauley.ucsd.edu/data/amazon/

a result, the start time is 1997-09-13, the meta-train cut-off time is 2012-01-13, the meta-validation cut-off time is 2013-03-05, and the end time is 2014-07-23. Users with less than 30 samples in each split are removed. In the end, the average sample size for each user is 82. The statistics of the pre-processed Amazon-CDs dataset are recorded in Table 2. The number of meta-train/validation/test users is shown in Table 3.

APPENDIX C
HYPER-PARAMETERS SETTINGS

C.1 Few-Shot Image Classification

For fair comparisons, we use the same feature extractor and base-learner for all the compared methods. Following [2], the feature extractor adopts a Conv-4 architecture: a 4-layer 3×3 convolutions with 32 filters, followed by a fully-connected layer that maps the extracted features into the task embedding dimension d_e. For the base-learner, we investigate 2 different backbone architectures: Conv-4 and ResNet-12. We follow [24] for the implementation of ResNet-12, which adopts (64-128-256-512) for the number of filters at the 4 blocks, where each block contains 3 convolution layers and a residual layer. Each convolution layer is accompanied by batch normalization, ReLU activation, and 2×2 max-pooling.

For all the task-conditioned methods that generate task embeddings (i.e., MMAML [25], L2F [26], HSML [1], ARML [27]), we set the dimension of task embedding d_e as 128, following [1]. For methods that adopt some form of clustering (i.e., HSML [1], ARML [27], TSA-MAML [8]), we follow their best hyper-parameters settings for the task-homogeneous datasets (i.e., miniImageNet, tieredImagenet and CIFAR-FS). For the task-heterogeneous Mixture-Of-Datasets where the number of underlying distributions is clearly known, we set the number of clusters as 6 for TSA-MAML and our CTML where K-means clustering is adopted (for our CTML, we set the number of clusters for both feature and path representations as 6, i.e., $k_{path} = k_{feat} = 6$). For HSML [1] where hierarchical clustering is adopted, we set the number of clusters in the first layer as 3 and the second layer as 2. For ARML [27] where a meta-relational graph is constructed, we set the number of vertices in the graph as 6. For Meta-Dataset, since 8 out of 10 sub-datasets are involved in meta-training, we set the number of clusters as 8 for our CTML.

For the task-adaptive inner update methods (i.e., ModGrad [28], ALFA [29], MeTAL [30]), we follow the best hyper-parameters settings suggested by the authors. For ModGrad, we set the dimension of the context vector as 300, and the dimension of the attention vector as 5. We apply preconditioning only on the gradients of the last two blocks of the backbone, following the official implementation of ModGrad. For ALFA, we adopt a 3-layer MLP with ReLU as the update rule learner. For MeTAL, we adopt a 2-layer MLP as the meta loss function and another 2-layer MLP as the loss function adapter that learns to output the task-specific affine transformation parameters for the weights of the meta loss network.

C.2 Cold-Start Recommendation

Different from the image dataset, the recommendation dataset typically consists of user-item interactions and profile features for both sides. The feature extractor is implemented as several embedding lookup tables, each corresponding to one categorical feature. We set the embedding dimension as 8 for all features. The base-learner is implemented as a general deep learning-based recommender system model, composed of an embedding layer followed by a 3-layer MLP that models the interaction between different features. The embedding dimension for the base-learner is set to 8, and the hidden sizes for the fully-connected layers are set to be (32-16).

For both MAMO [31] and our CTML, where K-means clustering is adopted to group users/tasks, we tune the number of clusters (in our case, both k_{path} and k_{feat}) in $\{4, 8, 12, 16, 20\}$, and tune the dimension of task embedding d_e in $\{64, 128, 256, 512, 1024\}$. Other hyper-parameters settings specific to the baseline methods are directly adopted from the source codes released by the authors.

APPENDIX D
OTHER PATH LEARNER DESIGNS

In addition to the GRU-based meta path learner design described in (5), we also experiment with 3 other network designs to test their effectiveness in modeling the step-wise geometric quantities.

Recall that the meta path learner is applied coordinate-wise on the base-learner parameters. That is, the same meta path learner is used to process the sequence of step-wise tuples $\mathbf{P}^d_{\mathcal{T}_i} \in \mathbb{R}^{(\tau+1)\times 4}$ independently for all parameters at different dimensions $d \in \{1, 2, ..., D\}$.

The first design linearly transforms the vectorized $\mathbf{P}^d_{\mathcal{T}_i}$ with learnable weight and bias:

$$\texttt{Linear}: \quad \mathbf{h}^d_i = \mathbf{W} \cdot \mathrm{vec}(\mathbf{P}^d_{\mathcal{T}_i}) + \mathbf{b}. \tag{1}$$

The second one employs multiple fully-connected layers with nonlinear activation $\sigma(\cdot)$:

$$\begin{aligned} \texttt{FC}: \quad & \mathbf{u}^0 = \mathrm{vec}(\mathbf{P}^d_{\mathcal{T}_i}), \\ & \mathbf{u}^1 = \sigma(\mathbf{W}^1 \cdot \mathbf{u}^0 + \mathbf{b}^1), \\ & \qquad \vdots \\ & \mathbf{u}^L = \sigma(\mathbf{W}^L \cdot \mathbf{u}^{L-1} + \mathbf{b}^L), \\ & \mathbf{h}^d_i = \mathbf{u}^L. \end{aligned} \tag{2}$$

The third design applies multiple self-attention layers on the sequence with scaled dot-product similarity following [32]. To inform about the sequential order, we apply sinusoid positional encodings at the input level, and also apply the causal mask to prevent interaction with future steps (i.e., set the upper triangular entries of the similarity matrix to $-\infty$ before inputting into softmax). The final representation

$\mathbf{h}_i^d$ is given by the $(\tau+1)$-th row (i.e., the last row) of the final-layer output:

$$
\begin{aligned}
\texttt{Attention}: \quad & \mathbf{U}^0 = \mathbf{P}^d_{\mathcal{T}_i} + positional\ encodings, \\
& \mathbf{U}^1 = \text{softmax}(\frac{\mathbf{Q}^1 \cdot (\mathbf{K}^1)^\top}{\sqrt{d_a}}) \cdot \mathbf{V}^1, \\
& \qquad \vdots \\
& \mathbf{U}^L = \text{softmax}(\frac{\mathbf{Q}^L \cdot (\mathbf{K}^L)^\top}{\sqrt{d_a}}) \cdot \mathbf{V}^L, \\
& \mathbf{h}_i^d = \mathbf{U}^L_{\tau+1,:}, \\
\text{where} \quad & \mathbf{Q}^l = \sigma(\mathbf{U}^{l-1} \cdot \mathbf{W}^l_Q + \mathbf{b}^l_Q) \in \mathbb{R}^{(\tau+1)\times d_a}, \\
& \mathbf{K}^l = \sigma(\mathbf{U}^{l-1} \cdot \mathbf{W}^l_K + \mathbf{b}^l_K) \in \mathbb{R}^{(\tau+1)\times d_a}, \\
& \mathbf{V}^l = \sigma(\mathbf{U}^{l-1} \cdot \mathbf{W}^l_V + \mathbf{b}^l_V) \in \mathbb{R}^{(\tau+1)\times d_a}.
\end{aligned} \tag{3}
$$

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 $\mathcal{T}_i$:

$$\mathbf{e}^{path}_{\mathcal{T}_i} = \text{FC}([\mathbf{h}_i^1, ..., \mathbf{h}_i^D]^\top). \tag{4}$$

Appendix E
Additional Experimental Results

E.1 Additional Learning Path Visualization

E.1.1 Few-Shot Image Classification

Figure 1 shows the learning paths for 6 meta-test tasks (5-way 1-shot) randomly sampled from the other 3 sub-datasets: Bird, Fungi and Texture. From Figure 1b, we see that the overall direction of the learning path plays an important role in determining the task representation. Looking at the GRU gates in Figure 1c, it seems that the initial step and the last few steps are often the most important/informative steps, as they provide useful information about the learning direction. From Figure 1d to 1f, we see that the modulated initializations produced by CTML exhibit a better clustering result than CTML-feat, owing to the inclusion of path information. The modulated initializations and actual learning paths of CTML(approx.) are very similar to that of CTML. Looking at the task modulation on selected layers in Figure 1g to 1i, CTML(approx.) produces very similar modulation to that of CTML, while CTML-feat produces a rather different modulation from the other two.

E.1.2 Cold-Start Recommendation

We also provide visualization of learning paths for cold-start recommendations. Figure 2 and 3 show the visualization of 6 users randomly selected from the MovieLens-1M and Amazon-CDs datasets respectively.

In this case, we notice that the direction of the learning path is still the most important factor in representing tasks. Since the steps are now more evenly spread along the rehearsed paths (Figure 2a and 3a), all the steps are important in determining the path direction and hence the z gates shown in Figure 2b and 3b exhibit more dark-blue blocks among the steps. For visualization of actual learning paths, since the distance between the modulated initializations and the distance spanned by the actual learning paths are now of comparable magnitude, we do not need to zoom in to see the actual learning paths in this case (unlike the image classification examples). The actual learning paths are shown in Figure 2c to 2e and Figure 3c to 3e. For CTML, we can see that tasks with similar rehearsed learning paths are modulated similarly and their initializations are located closely, while for CTML-feat, there is no clear clustering pattern for the task initializations, demonstrating that the features alone are insufficient to distinguish between users. Here, CTML(approx.) again does a good job of approximating the CTML results and producing similar actual learning paths. In Figure 3f to 3h and Figure 2f to 2h, we show the modulation on the weights and biases of the 3 fully-connected layers (i.e., the decision layers) for the first task (i.e., blue circle task). Depending on the dataset, the input dimension of the first layer varies with the number of features available (i.e., the user and item profile features). From the results, we see that the task modulation produced by CTML-feat is distinct from the other two, and CTML(approx.) is able to produce very similar task modulation as CTML.

E.2 Additional Shortcut Approximation Visualization

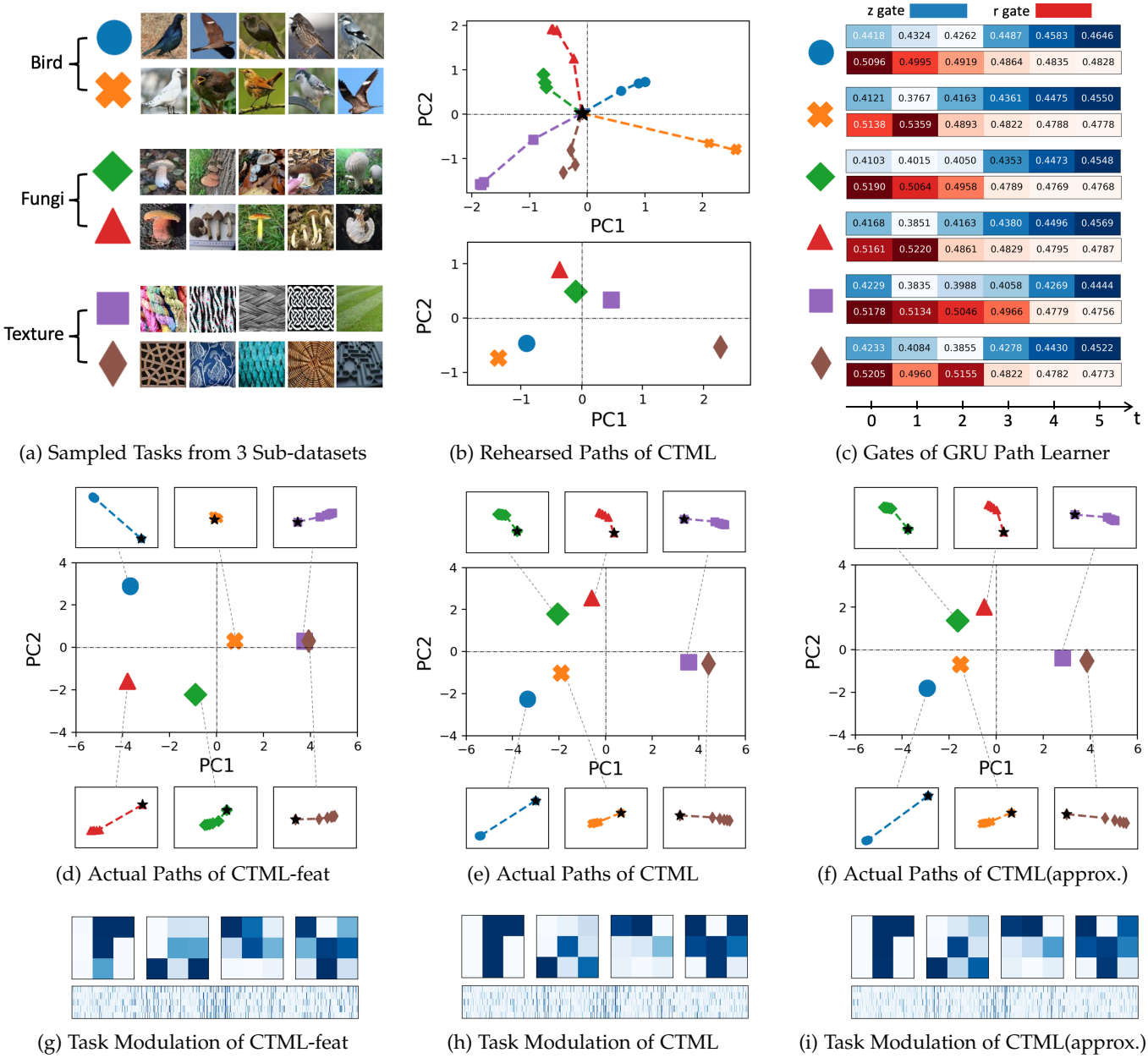
In this section, we give more examples to verify the effectiveness of the shortcut approximation. Figure 4 shows the results of 3 tasks sampled from 3 different image sub-datasets. We can see that the shortcut tunnel is able to learn and accommodate different mappings, providing a guarantee of the performance of the approximated model.

E.3 Comparisons with SOTA Methods on Meta-Dataset

Table 4 shows comparisons of Proto-CTML with 8 state-of-the-art methods on Meta-Dataset. Looking at the average rank, Proto-CTML emerges as the third among these strong baselines, even under the unfavorable condition that the classification layer cannot be meta-learned, demonstrating the effectiveness of incorporating path information in this challenging task-heterogeneous scenario.

E.4 Full Results on Recommendation Datasets

Table 5 presents full disentangled results for warm & cold users on 3 recommendation datasets. Here, we see that the performance improvements of CTML over the baselines for cold users are significant, demonstrating the effectiveness of CTML in generalizing to unseen tasks.

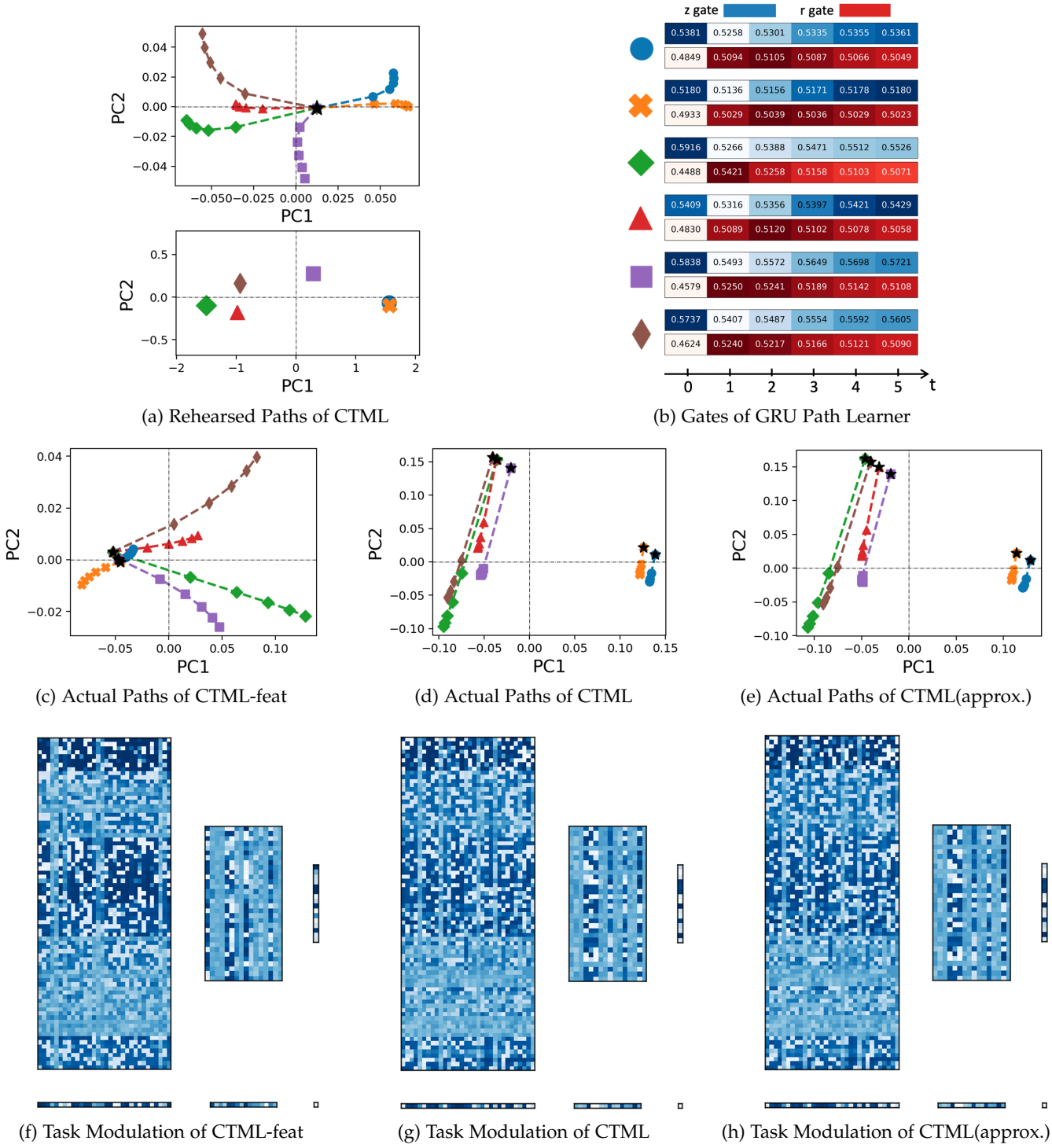


(a) Sampled Tasks from 3 Sub-datasets (b) Rehearsed Paths of CTML (c) Gates of GRU Path Learner

(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. 1: Visualization of learning paths. (a) 6 meta-test tasks (5-way 1-shot) randomly sampled from the **Bird**, **Fungi**, and **Texture** 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. (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.



(a) Rehearsed Paths of CTML

(b) Gates of GRU Path Learner

(c) Actual Paths of CTML-feat

(d) Actual Paths of CTML

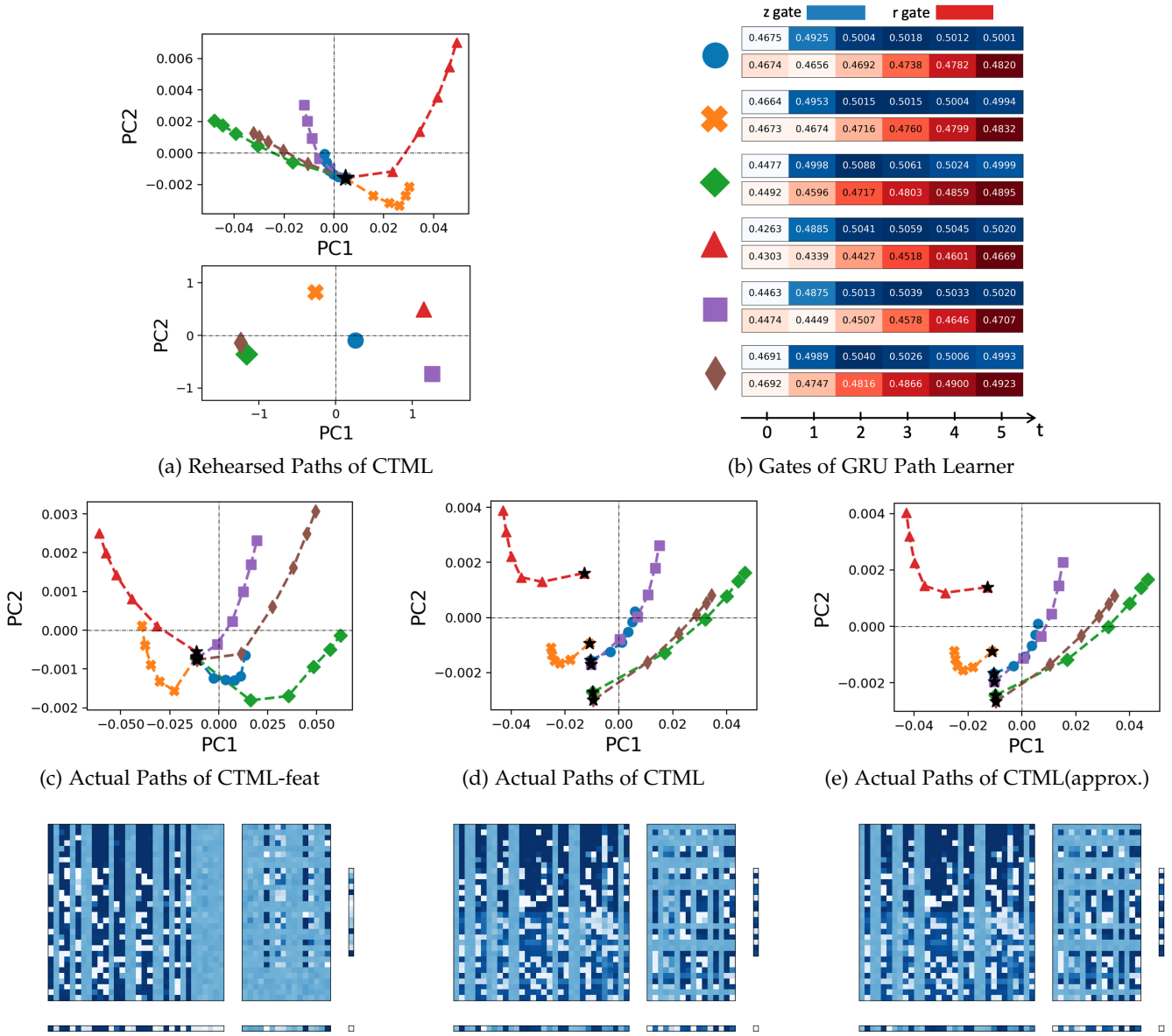
(e) Actual Paths of CTML(approx.)

(f) Task Modulation of CTML-feat

(g) Task Modulation of CTML

(h) Task Modulation of CTML(approx.)

Fig. 2: Visualization of learning paths for 6 meta-test cold users randomly selected from **MovieLens-1M**. (a) 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. (b) 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. (c)-(e) 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. (f)-(h) Visualization of task modulation on the weights and biases of the 3 fully-connected layers (i.e., the decision layers) for the first task (i.e., blue circle task) generated by CTML-feat, CTML, and CTML(approx.) respectively.



(a) Rehearsed Paths of CTML

(b) Gates of GRU Path Learner

(c) Actual Paths of CTML-feat

(d) Actual Paths of CTML

(e) Actual Paths of CTML(approx.)

(f) Task Modulation of CTML-feat

(g) Task Modulation of CTML

(h) Task Modulation of CTML(approx.)

Fig. 3: Visualization of learning paths for 6 meta-test cold users randomly selected from **Amazon-CDs**. (a) 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. (b) 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. (c)-(e) 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. (f)-(h) Visualization of task modulation on the weights and biases of the 3 fully-connected layers (i.e., the decision layers) for the first task (i.e., blue circle task) generated by CTML-feat, CTML, and CTML(approx.) respectively.

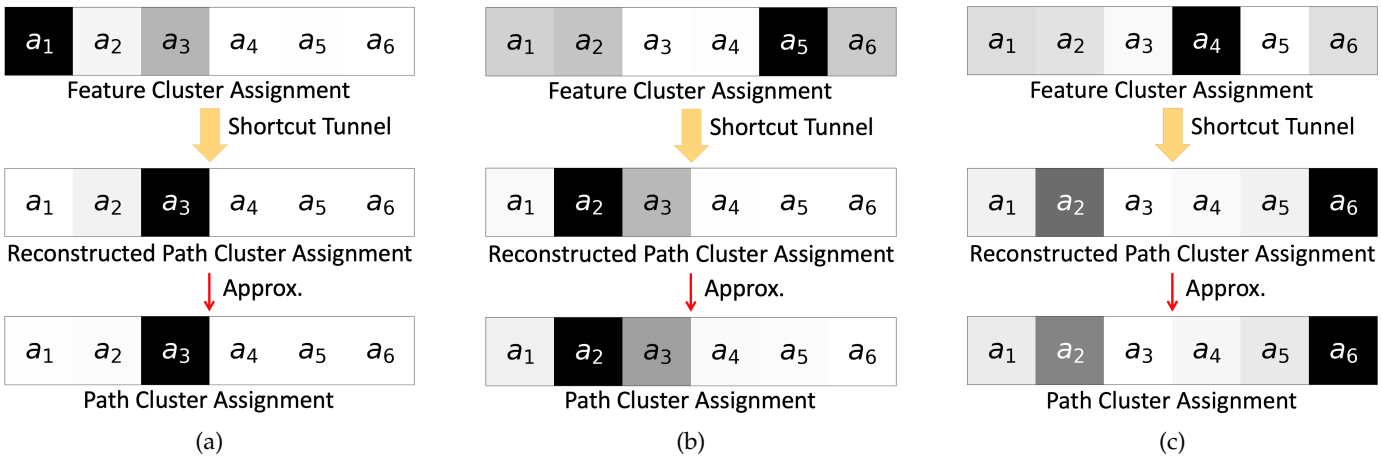


Fig. 4: Shortcut approximation of path cluster assignment from feature cluster assignment for 3 tasks sampled from 3 different image sub-datasets. Here, we set $k_{feat} = k_{path} = 6$.

TABLE 4: Performance on Meta-Dataset in the form of 95% confidence interval of accuracy (%) for state-of-the-art methods. Models are trained on the first 8 sub-datasets and tested on the meta-test splits of all the 10 sub-datasets. Models are ranked based on the average performance over the 10 sub-datasets. The results reported here are directly obtained from [33].

Sub-Datasets	CNAPS [34]	SimpleCNAPS [35]	SUR [36]	URT [37]	FLUTE [38]	tri-M [39]	URL [40]	TSA [33]	Proto-CTML
ILSVRC	50.8 ± 1.1 (9)	56.5 ± 1.1 (5)	54.5 ± 1.1 (7)	55.0 ± 1.1 (6)	51.8 ± 1.1 (8)	58.6 ± 1.0 (1)	57.5 ± 1.1 (2)	57.4 ± 1.1 (3)	56.6 ± 1.1 (4)
Omniglot	91.7 ± 0.5 (9)	91.9 ± 0.6 (8)	93.0 ± 0.5 (6)	93.3 ± 0.5 (4)	93.2 ± 0.5 (5)	92.0 ± 0.6 (7)	94.5 ± 0.4 (2)	95.0 ± 0.4 (1)	93.6 ± 1.4 (3)
Aircraft	83.7 ± 0.6 (7)	83.8 ± 0.6 (6)	84.3 ± 0.5 (5)	84.5 ± 0.6 (4)	87.2 ± 0.5 (3)	82.8 ± 0.7 (8)	88.6 ± 0.5 (2)	89.3 ± 0.4 (1)	79.9 ± 1.0 (9)
Birds	73.6 ± 0.9 (8)	76.1 ± 0.9 (4)	70.4 ± 1.1 (9)	75.8 ± 0.8 (5)	79.2 ± 0.8 (3)	75.3 ± 0.8 (7)	80.5 ± 0.7 (2)	81.4 ± 0.7 (1)	75.6 ± 1.2 (6)
Textures	59.5 ± 0.7 (9)	70.0 ± 0.8 (7)	70.5 ± 0.7 (6)	70.6 ± 0.7 (4.5)	68.8 ± 0.8 (8)	71.2 ± 0.8 (3)	76.2 ± 0.7 (2)	76.7 ± 0.7 (1)	70.6 ± 1.0 (4.5)
Quick Draw	74.7 ± 0.8 (9)	78.3 ± 0.7 (7)	81.6 ± 0.6 (4)	82.1 ± 0.6 (1)	79.5 ± 0.7 (6)	77.3 ± 0.7 (8)	81.9 ± 0.6 (3)	82.0 ± 0.6 (2)	80.3 ± 1.2 (5)
Fungi	50.2 ± 1.1 (7)	49.1 ± 1.2 (8)	65.0 ± 1.0 (3.5)	63.7 ± 1.0 (5)	58.1 ± 1.1 (6)	48.5 ± 1.0 (9)	68.8 ± 0.9 (1)	67.4 ± 1.0 (2)	65.0 ± 0.9 (3.5)
VGG Flower	88.9 ± 0.5 (6)	91.3 ± 0.6 (4)	82.2 ± 0.8 (9)	88.3 ± 0.6 (7)	91.6 ± 0.6 (3)	90.5 ± 0.5 (5)	92.1 ± 0.5 (2)	92.2 ± 0.5 (1)	87.2 ± 0.8 (8)
Traffic Sign	56.5 ± 1.1 (7)	59.2 ± 1.0 (5)	49.8 ± 1.1 (9)	50.1 ± 1.1 (8)	58.4 ± 1.1 (6)	63.0 ± 1.0 (4)	63.3 ± 1.2 (3)	83.5 ± 0.9 (1)	64.8 ± 1.3 (2)
MSCOCO	39.4 ± 1.0 (9)	42.4 ± 1.1 (8)	49.4 ± 1.1 (6)	48.9 ± 1.1 (7)	50.0 ± 1.0 (5)	52.8 ± 1.1 (4)	54.0 ± 1.0 (2)	55.8 ± 1.1 (1)	53.4 ± 1.1 (3)
Avg. rank	8	6.2	6.45	5.15	5.3	5.6	2.1	1.4	4.8

TABLE 5: Full disentangled results for warm & cold users on 3 recommendation datasets in MAE and NDCG@20. We report the mean over 5 trials, where the std devs are all less than 0.001

Metrics	Methods	MovieLens-1M			Yelp			Amazon-CDs		
		Overall	Warm Users	Cold Users	Overall	Warm Users	Cold Users	Overall	Warm Users	Cold Users
MAE ↓	MeLU [23]	0.8164	0.8112	0.8197	0.9464	0.9100	0.9576	0.7216	0.6840	0.7383
	PAML [41]	0.7725	0.7575	0.7879	0.9280	0.9151	0.9336	0.7150	0.6813	0.7317
	MAMO [31]	0.8084	0.8073	0.8091	0.8965	0.9125	0.8840	0.7207	0.7103	0.7260
	MetaHIN [42]	0.7727	0.7490	0.7841	0.8832	0.9021	0.8767	0.6743	0.6675	0.6804
	CTML-feat	0.7560	0.7482	0.7631	0.9023	0.9107	0.8949	0.6804	0.6697	0.6874
	CTML-path	0.7592	0.7474	0.7690	0.8714	0.8912	0.8581	0.6286	0.6207	0.6372
	CTML	**0.7543**	0.7415	**0.7626**	**0.8603**	**0.8700**	**0.8533**	**0.6048**	**0.6080**	0.6011
	CTML(approx.)	0.7555	**0.7411**	0.7652	0.8679	0.8721	0.8607	0.6087	0.6144	**0.6007**
NDCG@20 ↑	MeLU [23]	0.7839	0.7935	0.7716	0.8149	0.8161	0.8139	0.9026	0.9037	0.9020
	PAML [41]	0.8255	0.8391	0.8165	0.8215	0.8230	0.8204	0.9083	0.9096	0.9051
	MAMO [31]	0.8118	0.8274	0.8030	0.8332	0.8296	0.8384	0.9116	0.9067	0.9140
	MetaHIN [42]	0.8264	0.8356	0.8110	0.8375	0.8324	0.8412	0.9111	0.9082	0.9153
	CTML-feat	0.8470	0.8530	0.8384	0.8295	0.8301	0.8259	0.9122	0.9078	0.9179
	CTML-path	0.8493	0.8499	0.8437	0.8401	0.8327	0.8420	0.9201	0.9180	0.9238
	CTML	**0.8562**	**0.8634**	**0.8487**	**0.8467**	**0.8364**	**0.8524**	**0.9279**	**0.9244**	**0.9293**
	CTML(approx.)	0.8531	0.8627	0.8418	0.8433	0.8319	0.8497	0.9244	0.9215	0.9260

REFERENCES

[1] H. Yao, Y. Wei, J. Huang, and Z. Li, "Hierarchically structured meta-learning," in *ICLR*, 2019, pp. 7045–7054.

[2] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, pp. 1126–1135.

[3] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2016.

[4] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," *arXiv:1803.00676*, 2018.

[5] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," *arXiv:1805.08136*, 2018.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[7] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.

[8] 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 *UAI*, 2021, pp. 23–33.

[9] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[10] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," *arXiv:1306.5151*, 2013.

[11] B. Schroeder and Y. Cui, "Fgvcx fungi classification challenge 2018," 2018.

[12] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *ICVGIP*, 2008, pp. 722–729.

[13] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *CVPR*, 2014, pp. 3606–3613.

[14] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.

[15] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol *et al.*, "Meta-dataset: A dataset of datasets for learning to learn from few examples," in *International Conference on Learning Representations*, 2019.

[16] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[17] J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg, "The quick, draw!-ai experiment (2016)," *URL http://quickdraw. withgoogle. com*, vol. 4, 2016.

[18] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," in *The 2013 international joint conference on neural networks (IJCNN)*. Ieee, 2013, pp. 1–8.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[20] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, 2015.

[21] Y. Dataset, "Yelp dataset challenge," 2019.

[22] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *SIGIR*, 2015, pp. 43–52.

[23] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "Melu: meta-learned user preference estimator for cold-start recommendation," in *KDD*, 2019, pp. 1073–1082.

[24] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *CVPR*, 2019, pp. 403–412.

[25] R. Vuorio, S.-H. Sun, H. Hu, and J. J. Lim, "Multimodal model-agnostic meta-learning via task-aware modulation," in *NeurIPS*, 2019.

[26] S. Baik, S. Hong, and K. M. Lee, "Learning to forget for meta-learning," in *CVPR*, 2020, pp. 2379–2387.

[27] H. Yao, X. Wu, Z. Tao, Y. Li, B. Ding, R. Li, and Z. Li, "Automated relational meta-learning," in *ICLR*, 2020.

[28] C. Simon, P. Koniusz, R. Nock, and M. Harandi, "On modulating the gradient for meta-learning," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII*, 2020, pp. 556–572.

[29] S. Baik, M. Choi, J. Choi, H. Kim, and K. M. Lee, "Meta-learning with adaptive hyperparameters," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, pp. 20 755–20 765.

[30] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9465–9474.

[31] M. Dong, F. Yuan, L. Yao, X. Xu, and L. Zhu, "Mamo: Memory-augmented meta-optimization for cold-start recommendation," in *KDD*, 2020, pp. 688–697.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[33] W.-H. Li, X. Liu, and H. Bilen, "Cross-domain few-shot learning with task-specific adapters," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7161–7170.

[34] J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner, "Fast and flexible multi-task classification using conditional neural adaptive processes," in *NeurIPS*, 2019, pp. 7959–7970.

[35] P. Bateni, R. Goyal, V. Masrani, F. Wood, and L. Sigal, "Improved few-shot visual classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 493–14 502.

[36] N. Dvornik, C. Schmid, and J. Mairal, "Selecting relevant features from a multi-domain representation for few-shot classification," in *European Conference on Computer Vision*. Springer, 2020, pp. 769–786.

[37] L. Liu, W. L. Hamilton, G. Long, J. Jiang, and H. Larochelle, "A universal representation transformer layer for few-shot image classification," in *International Conference on Learning Representations*, 2020.

[38] E. Triantafillou, H. Larochelle, R. Zemel, and V. Dumoulin, "Learning a universal template for few-shot dataset generalization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 424–10 433.

[39] Y. Liu, J. Lee, L. Zhu, L. Chen, H. Shi, and Y. Yang, "A multi-mode modulator for multi-domain few-shot classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8453–8462.

[40] W.-H. Li, X. Liu, and H. Bilen, "Universal representation learning from multiple domains for few-shot classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9526–9535.

[41] R. Yu, Y. Gong, X. He, Y. Zhu, Q. Liu, W. Ou, and B. An, "Personalized adaptive meta learning for cold-start user preference prediction," in *AAAI*, vol. 35, no. 12, 2021, pp. 10 772–10 780.

[42] Y. Lu, Y. Fang, and C. Shi, "Meta-learning on heterogeneous information networks for cold-start recommendation," in *KDD*, 2020, pp. 1563–1573.