Model-Agnostic Learning to Meta-Learn

Arnout Devos[1*] ARNOUT.DEVOS@EPFL.CH and **Yatin Dandi**[1,2*] YATIND@IITK.AC.IN
[1]*School of Computer and Communication Sciences, EPFL, Switzerland*
[2]*Department of Computer Science and Engineering, IIT Kanpur, India*

Abstract

In this paper, we propose a learning algorithm that enables a model to quickly exploit commonalities among related tasks from an unseen task distribution, before quickly adapting to specific tasks from that same distribution. We investigate how learning with different task distributions can first improve adaptability by meta-finetuning on related tasks before improving goal task generalization with finetuning. Synthetic regression experiments validate the intuition that learning to meta-learn improves adaptability and consecutively generalization. Experiments on more complex image classification, continual regression, and reinforcement learning tasks demonstrate that learning to meta-learn generally improves task-specific adaptation. The methodology, setup, and hypotheses in this proposal were positively evaluated by peer review before conclusive experiments were carried out.

Keywords: Pre-registration, Machine Learning

1. Introduction

Recent years have seen encouraging developments in meta-learning based approaches for deep neural networks and their successful application to various domains (Finn et al., 2017; Rajeswaran et al., 2019; Nichol et al., 2018). These approaches typically assume a distribution over tasks and aim to exploit the shared properties across tasks to learn a model that can adapt to unknown tasks from this distribution using only a few training data points. Unfortunately, their adaptive capabilities do not generalize well to unseen tasks from related but different task distributions (Chen et al., 2019).

A number of recent works have proposed addressing the presence of different sets of related tasks by explicitly factoring in the heterogeneous nature of the task distribution in the design of the architecture and update rule (Requeima et al., 2019; Yao et al., 2020, 2019; Vuorio et al., 2019). However, these approaches still assume a fixed task distribution, such as tasks sampled from a fixed set of families of functions or a multi-modal distribution arising out of a fixed set of task datasets. We argue that generalizing across unknown datasets and task distributions is a fundamentally more difficult problem than fixed distribution meta-learning. With a new task distribution or dataset, it is unrealistic to expect the model to quickly adapt to any arbitrary task from such a distribution. Instead, we expect the model to quickly learn to adapt to any task from the new task distribution after being exposed to only a few tasks of it. This generalizes the notion of few-shot learning to *few-task* (few-shot) learning. Changes in task distributions might also arise due to natural or artificial transformations of the data. With different task distributions arising from a "distribution

[*] Denotes equal contribution.

over task distributions", it is not only desirable to "quickly adapt" to unseen tasks but also "quickly learn to adapt" to unseen task distributions.

We propose a general framework for adapting to unseen task distributions by "learning to meta-learn" on different task distributions during training. Thus, the heterogeneity of tasks in our approach is not fixed but flexibly modeled through hierarchical sampling from a distribution over task distributions. Similar to MAML (Finn et al., 2017), we propose a general framework to learn a suitable initialization for a single set of parameters. Unlike MAML, which only trains a model to quickly adapt the parameters on a new task using few task-specific gradient steps, our model is also trained to quickly adapt its initialization to a new task distribution using few meta gradient steps on this unseen task distribution (see Figure 1). We hypothesize that our approach would allow models to transfer learn capabilities across datasets in supervised and unsupervised learning, and new environments in reinforcement learning as well as quickly adapt to unseen augmentations and distortions at test time.

2. Related Work

Model-Agnostic Meta-Learning (MAML) by Finn et al. (2017) is a seminal work in few-shot meta-learning which seeks a common model initialization that allows the model to perform well on any goal task from the training task distribution with few gradient steps (and samples). Multimodal MAML (MMAML) by Vuorio et al. (2018, 2019) extends MAML with the capability to identify tasks sampled from a multimodal task distribution and adapt quickly through gradient updates. Yao et al. (2019) proposed the hierarchically structured meta-learning (HSML) algorithm that explicitly tailors the transferable knowledge to different clusters of tasks. Automated Relational Meta-Learning (ARML) by Yao et al. (2020) extracts the cross-task relations and constructs a meta-knowledge graph. When a new task arrives, it can quickly find the most relevant structure and tailor the learned structure knowledge to the meta-learner. Still, MMAML, HSML, and ARML only learn to learn from a *fixed* task distribution. Unlike our approach, they are not expected to generalize to new task distributions.

Research on improving meta-learning algorithms is vast, and we will highlight work most related to our approach. Following MAML, Li et al. (2017) and Antoniou et al. (2019) learn the inner learning rate in the outer loop to improve performance, while reducing the hyperparameter tuning requirement. Finn et al. (2017) proposed a first-order approximation of MAML (fo-MAML) to scale to larger models, which was subsequently improved upon by Nichol et al. (2018) with a first-order method called Reptile. Reptile can naturally be extended to our proposed approach, making it scalable and efficient. Chen et al. (2020) found that with increasingly deep architectures, common pre-training and transfer learning can outperform meta-learning from scratch in the visual classification domain. Based on this, they proposed to combine regular pre-training with subsequent meta-learning, which empirically gives a further performance improvement. Raghu et al. (2020) found that feature reuse is a dominant factor in MAML, and proposed a variant called Almost no Inner Loop (ANIL) which learns to only fine-tune the last layer linear classifier. On the contrary, Oh et al. (2021) came to the opposite conclusion that fast learning is crucial, and proposed a

Body Only update in Inner Loop (BOIL) algorithm. These works motivate several of our research questions for the experiments.

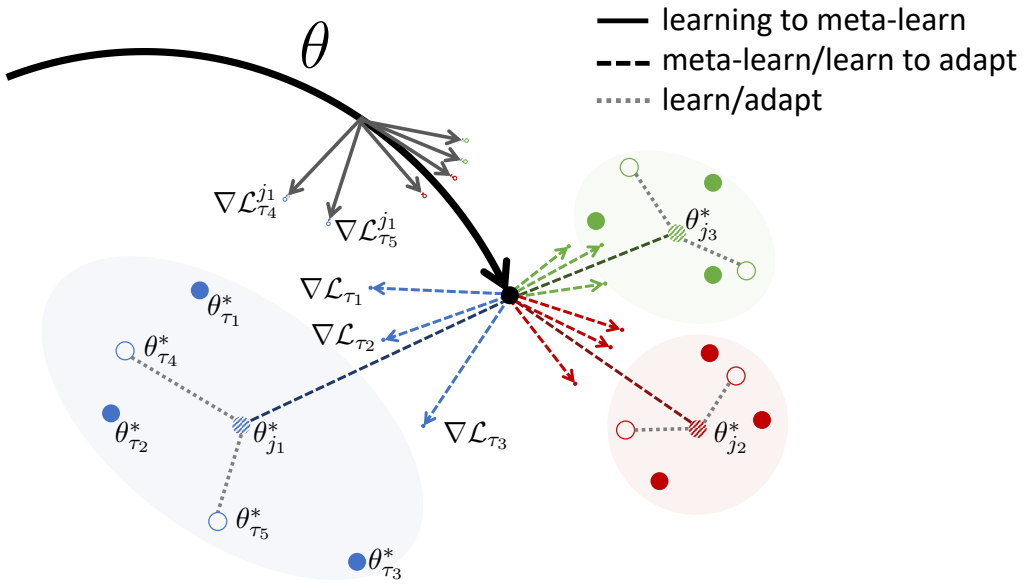


Figure 1: Illustration of our model-agnostic learning to meta-learn algorithm (MALTML), which optimizes for a representation θ that can quickly adapt to new task distributions j and consecutively to their tasks τ. Illustrated with single gradient steps for the meta-learning and learn/adapt phases.

We aim to train models that can quickly change their adaptability before rapid adaptation. We formalize this setting as few-task few-shot learning. In this section, we will clarify the problem setup and we will formalize this learning to meta-learn problem setting for supervised learning, but it can easily be generalized to unsupervised and reinforcement learning (RL).

3.1. Learning to Meta-Learn Problem Setup

In our learning to meta-learn scenario, we consider a distribution $p(j)$ over task distributions such as families of related functions or datasets with similarities. Our goal is to allow the model to adapt to unseen task distributions as well as specific tasks within such distributions. In the L-task K-shot setting, the model is trained to meta-learn a new task distribution j_d from only L tasks with only K examples each for task-learning and Q examples for meta-learning, before learning a single goal task $\mathcal{T}_i$ drawn from $p_{j_d}(\mathcal{T})$ from only K samples. The model f is then improved by considering how the test error on new (validation) data from $\mathcal{T}_i$ changes with respect to the original parameters. This test error on the final goal tasks serves as the training error of the learning to meta-learn process. At the end of training, new families are sampled from $p(j_d)$ and the model's learning to meta-learn performance is

measured by the model's performance after meta-finetuning on L tasks with $K+Q$ examples and finetuning on one or multiple goal tasks from the same family with K examples.

3.2. A Model-Agnostic Learning to Meta-Learn Algorithm

We propose a method that can learn the parameters of any model via learning to meta-learn in such a way as to prepare that model to first quickly change its adaptation capability (initialization) and consecutively adapt quickly to a goal task. The intuition behind this approach is that some internal representations are more transferable to meta-learn with. For example, a neural network could learn features that are broadly applicable to all tasks in the distribution over task distributions $p(j)$ and can then specialize to an individual task distribution $p_{j_d}(\mathcal{T})$, rather than to a single task distribution or task. We assume no specific form of the model, other than that it is parametrized by some parameter set θ, and that the loss functions are sufficiently smooth in θ such that we can employ gradient-based learning techniques.

Formally, we consider a model represented by a parametrized function f_θ with parameters θ. When adjusting the adaptability to a new task *family* j_d, the model's parameters θ become θ'_{j_d}, and consecutively when adapting (finetuning) to a new task $\mathcal{T}_c \sim p_{j_d}(\mathcal{T})$ the model's parameters θ'_{j_d} become θ'_c. In our approach, the updated parameter vector θ'_{j_d} is obtained by few (inner) *meta*-learning steps on few tasks from dataset j_d, also called *meta-finetuning*. Each meta-finetuning step is taken across few task-finetuning steps. A single ($r=1$) *task* gradient step with step size α is:

$$\theta'_{\mathcal{T}_i} = U_{\mathcal{T}_i}^{r=1,\alpha}(\theta) = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta). \tag{1}$$

Then, the family-specific meta-finetuning update, with one ($r=1$) task-level parameter update as in Equation (1) and one ($m=1$) meta-level update across tasks from j_d with step size β, is:

$$\theta'_{j_d} = V_{j_d}^{m=1,\beta}(\theta) = \theta - \beta \nabla_\theta \left(\sum_{\mathcal{T}_i \sim p_{j_d}(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{U_{\mathcal{T}_i}^{r=1,\alpha}(\theta)}) \right). \tag{2}$$

Consecutively, the updated task-specific parameter vector θ'_c is obtained by taking few gradient descent steps, with learning rate γ, on tasks $\mathcal{T}_c \sim p_{j_d}(\mathcal{T})$, starting from θ'_{j_d}. For example, using Equation (1), with 1 gradient step: $\theta'_c = U_{\mathcal{T}_c}^{r=1,\gamma}(\theta'_{j_d})$. Note that using the same r for meta-finetuning and goal task finetuning is a natural choice, but can be deviated from. The step-sizes α, β, and γ may be fixed as hyperparameters or learned on the outer learning loop (see below).

Finally, the global model parameters θ are optimized to perform well after this two-step (meta-learn, then learn) process. Specifically, the model parameters are trained by optimizing the performance of every $f_{\theta'_c}$ on its task $\mathcal{T}_c$. This is done across task distributions sampled from $p(j)$ and tasks sampled from them ($\mathcal{T}_c \sim p_{j_d}(\mathcal{T})$). Concretely, the *learning-to-meta-learn* objective is:

Algorithm 1 Model-Agnostic Learning to Meta-Learn

Require: $p(j)$: distribution over task distributions parameter j
Require: α, β, γ, η: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of task distributions $j_d \sim p(j)$
  for all j_d **do**
    Sample L tasks $\mathcal{T}_i \sim p_{j_d}(\mathcal{T})$
    for all $\mathcal{T}_i$ **do**
      Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
      Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    end for
    Update $\theta'_{j_d} \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ using Q examples per task
    Sample batch of tasks $\mathcal{T}_c \sim p_{j_d}(\mathcal{T})$
    for all $\mathcal{T}_c$ **do**
      Evaluate $\nabla_{\theta'_{j_d}} \mathcal{L}_{\mathcal{T}_c}(f_{\theta'_{j_d}})$ with respect to K examples
      Compute adapted parameters with gradient descent: $\theta'_c = \theta'_{j_d} - \gamma \nabla_{\theta'_{j_d}} \mathcal{L}_{\mathcal{T}_c}(f_{\theta'_{j_d}})$
    end for
  end for
  Update $\theta \leftarrow \theta - \eta \nabla_\theta \sum_{j_d \sim p(j)} \sum_{\mathcal{T}_c \sim p_{j_d}(\mathcal{T})} \mathcal{L}_{\mathcal{T}_c}(f_{\theta'_c})$
end while

$$\min_\theta \sum_{j_d \sim p(j)} \sum_{\mathcal{T}_c \sim p_{j_d}(\mathcal{T})} \mathcal{L}_{\mathcal{T}_c}(f_{\theta'_c}) = \min_\theta \sum_{j_d \sim p(j)} \sum_{\mathcal{T}_c \sim p_{j_d}(\mathcal{T})} \mathcal{L}_{\mathcal{T}_c}\left(U^{r,\gamma}_{\mathcal{T}_c}(V^{m,\beta}_{j_d}(\theta))\right)$$

Note that the learning to meta-learn optimization is performed over the model parameters θ, whereas the learning to meta-learn objective itself is computed using the updated model parameters θ'_c.

The outer optimization across task distributions is performed via stochastic gradient descent (SGD), such that the model parameters θ are updated as follows:

$$\theta \leftarrow \theta - \eta \nabla_\theta \sum_{j_d \sim p(j)} \sum_{\mathcal{T}_c \sim p_{j_d}(\mathcal{T})} \mathcal{L}_{\mathcal{T}_c}(f_{\theta'_c}) \tag{3}$$

where η is the outer step size. The full algorithm, in the general case, is outlined in Algorithm 1.

The MALTML outer gradient update yields a third-order gradient with respect to θ. To make MALTML computationally usable for high-dimensional models, we propose a first-order approximation. Concretely, following Nichol et al. (2018), for every family j_d we employ multiple *first-order* meta-learning updates $\theta'_{j_d} = \widetilde{V}^{m>1}_{j_d}(\theta)$, before updating the model parameters with $\theta \leftarrow \theta + \eta \sum_{j_d}(\theta'_{j_d} - \theta)$. Note that in this case, due to the nature of our two-level first-order approximation, goal task finetuning is not required anymore.

4. Experimental Protocol

The goal of our experimentals is to get conclusive results in different learning domains on whether MALTML can enable a quick and significant change of adaptability to goal tasks from new task distributions. Moreover, we wish to examine whether fast learning (to meta-learn) (Oh et al., 2021) or feature reuse (Raghu et al., 2020) is the dominant factor in the performance.

All the learning to meta-learn problems we consider require some form of change in adaptability and subsequent adaptation to new tasks at test time. When possible, we will compare our results to an oracle that receives the identities of the family and goal task as an additional input or a MAML oracle that is able to meta-finetune on a large number of tasks from the new task distribution, as upper bounds on the performance of the models. Regarding model architecture and optimization, we will follow Finn et al. (2017). We will use insights from Antoniou et al. (2019) to stabilize training where applicable and follow its hierarchical hyperparameter search methodology. We will carry out the experiments in PyTorch (Paszke et al., 2019), using the torchmeta package (Deleu et al., 2019). The code will be available online.

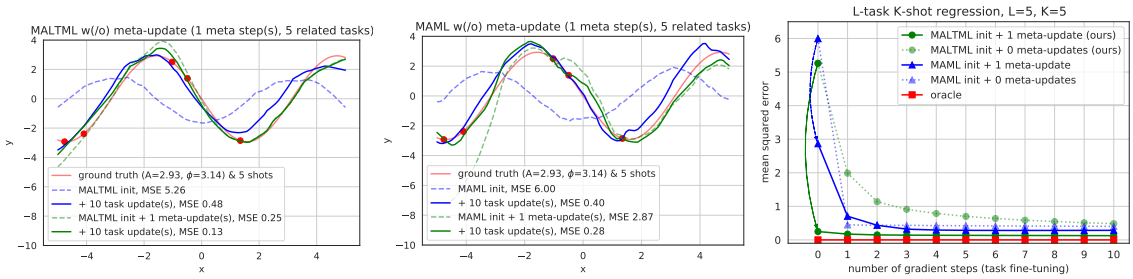
4.1. Illustrative preliminary experiment: regression

We start with a toy regression problem which illustrates the experimental protocol of few-task few-shot learning and the basic principles of MALTML.

Each task distribution (family) consists of sinusoid regression tasks with a specific phase. Thus, $p(j)$ is continuous, where the phase j varies uniformly within $[0, \pi]$. Each task involves regressing from the input to the output of a sine wave, where the amplitude is varied between tasks. Thus, $p_{j_d}(\mathcal{T})$ is continuous, where the amplitude varies within $[0.1, 5.0]$ and the input and output both have a dimensionality of 1. During training and testing, datapoints x are sampled uniformily from $[-5, 5]$. The loss is the mean squared error between the prediction $f(x)$ and the true value. The model is a neural network with 2 hidden layers of size 40 with ReLU nonlinearities. When training with MALTML, we use single gradient updates with fixed step sizes of $\alpha = 0.001$, $\beta = 0.01$, $\gamma = 0.001$, and use Adam (Kingma and Ba, 2015) with an initial learning rate of $\eta = 0.001$. The baselines are also trained with Adam, and an inner learning rate of $\alpha = 0.001$ for MAML. For the 5-task 5-shot regression experiment, we train for 70,000 outer steps with a family batch size of 10, $Q = 5$, and 2 validation tasks per family.

For this preliminary experiment we only contrast with a MAML baseline, which disregards the family structure, and an oracle receiving the true amplitude and phase of the goal task as additional input. In general, we intend to compare to the other oracles described before and another baseline: pretraining on all tasks, which in this case involves training a model to regress random sinusoid functions.

The toy results in Figures 7(b) and 2(c) show that the learned MALTML model is able to quickly change its adaptability to the new family's phase on 5 related 5-shot tasks. Due to this, it reaches a better fit than MAML, which benefits less from the meta-adaptation of its initialization (Figures 7(b) and 2(c)).



(a) (b) (c)

Figure 2: Effect of the meta-update and task fine-tuning on (a) MALTML (b) MAML (c) Test error vs task fine-tuning steps (the meta-update improvement is indicated by the arrows at 0 steps).

4.2. Specifics of main experiments

Besides the toy experiment in section 4.1, we propose to test the effectiveness of MALTML for:

Classification. We propose to apply our method to modified versions of the Omniglot (Lake et al., 2015) and ImageNet (Russakovsky et al., 2015) datasets. For ImageNet, we will propose a few-task few-shot dataset, making use of its hierarchical structure to generate a sufficient amount of training families. Given the finite number of alphabets in Omniglot, which will serve as families, we will use data augmentations similar to the ones used in Khodadadeh et al. (2019) to generate a large number of training families. To arrive at a realistic setting where the (imposed) hierarchical structure from the supervised case is lacking, we will use a hierarchically augmented version of unsupervised meta-learning (Khodadadeh et al., 2019). Specifically, a subset of data augmentation parameters will be sampled per family, before applying them randomly (with the remaining subset) on samples to generate tasks from each family. Note that in this case, on a (family) meta-level the method needs to be sensitive to augmentations, whereas on a task-level it should aim to be invariant to them.

2D Navigation. We propose to evaluate MALTML on a set of families of RL tasks where a point agent must move to different goal positions in 2D, while being given related tasks from the same family. Every family constitutes of a random crop of the unit square, and every task is randomly chosen from within that rectangle. The crops are bounded by 25% to 75% of the original unit length.

Continual Regression We propose to evaluate a continual learning extension of MALTML on incremental sine wave learning as described in Javed and White (2019). Different families of continual learning prediction problems will correspond to different frequencies of the sine waves. The inner meta-learning objective for this setting will be replaced by the Online-aware Meta-Learning objective from Javed and White (2019).

Continual Reinforcement Learning Besides the sine waves continual regression problem, we will aim to evaluate a more challenging and real-world setting of continuous control inspired by Kaplanis et al. (2020).

5. Future Work

Based on the results of our main experiments, future work could involve a multi-task setting corresponding to meta-learning on different categories of tasks such as classification, segmentation, and depth estimation on a single dataset or a set of related datasets. This could also involve augmenting our objective for enforcing cross-task (distribution) consistency (Zamir et al., 2020).

6. Results

6.1. Experimental Setup and Hyperparameters

For the tasks included in Finn et al. (2017), i.e., Omniglot classification, 2D Navigation and Half-Cheetah goal velocity tasks, we use the same topmost optimizer and learning rate, fine-tuning step size, number of gradient steps, and number of tasks for the meta-step for the MAML baselines and the inner task specific updates for MATML. For the other baselines and tasks, we borrow the architectures and hyperparameters from works that compare with MAML's original setup, i.e., Raghu et al. (2020) for ANIL and Oh et al. (2021) for BOIL and tiered-ImageNet. The used architectures and hyperparameters are further described in Appendix A and B. For the inner meta-step size, we performed a grid search over the set of values $\beta \in \{2, 1, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001\}$. Due to computational constraints on our Tesla V100-SXM2 32GB GPU, for tiered-ImageNet and continual regression, we sample only one family per outer update. Other details specific to the tasks are discussed in the corresponding sections and the Appendix.

6.2. Classification

6.2.1. Omniglot

Although the original Omniglot dataset (Lake et al., 2015) provided in characters from separated alphabets for training and testing, it did not provide in a pre-defined validation set as is common in few-shot learning (Vinyals et al., 2017). In Vinyals et al. (2017) different sets of characters for training, validation and testing are sampled, disregarding the alphabet structure. Needing to separate alphabets across these sets for learning to meta-learn, we created new sets. The details of the adapted dataset are described in Appendix A. Since we use a different dataset organization than most other literature, we reproduce results for the baselines as well. Specifically, we reproduce results for MAML (Finn et al., 2017), ANIL (Raghu et al., 2020) and BOIL (Oh et al., 2021) for an honest comparison of rapid learning and feature reuse. Our equivalent learning to meta-learn methods are MALTML, MALTML-ANIL, and MALTML-BOIL, respectively. Appendix Table 5 lists the hyperparameters used for Omniglot. Table 1 shows the results on Omniglot.

6.2.2. Tiered-ImageNet

In a more challenging setting, we evaluate MATLML and the baselines on the Tiered-Imagenet dataset with the families corresponding to different categories of classes, as defined in Ren et al. (2018). Following Ren et al. (2018), the 34 categories are divided into 20 training, 6

Model	1-shot	5-task 1-shot	5-shot	5-task 5-shot
MAML (Finn et al., 2017)	97.86 ± 0.29%	98.04 ± 0.30%	99.32 ± 0.17%	99.26 ± 0.19%
ANIL (Raghu et al., 2020)	96.50 ± 0.41%	96.86 ± 0.38%	98.62 ± 0.27%	98.60 ± 0.26%
BOIL (Oh et al., 2021)	97.39 ± 0.35%	97.57 ± 0.32%	99.10 ± 0.22%	99.19 ± 0.18%
MALTML (ours)	95.82 ± 0.41%	96.49 ± 0.35%	98.65 ± 0.21%	98.77 ± 0.19%
MALTML-ANIL (full, ours)	91.30 ± 0.69%	92.73 ± 0.65%	94.92 ± 0.56%	95.12 ± 0.58%
MALTML-BOIL (full, ours)	97.04 ± 0.35%	97.28 ± 0.32%	98.85 ± 0.23%	99.20 ± 0.17%

Table 1: Accuracies of 5-way (5-task) Few-shot learning on 16 held-out Omniglot alphabets, before and after adaptation to the test families. Includes 95% confidence intervals across 600 test tasks from the 16 test alphabets.

validation, and 8 test categories. We visualize a selection of categories (families) and their tasks in Appendix B. Table 2 shows the results on tiered-ImageNet.

Model	1-shot	4-task 1-shot	5-shot	2-task 5-shot
MAML (Finn et al., 2017)	46.3 ± 2.4%	46.5 ± 2.8%	62.1 ± 2.6%	62.6 ± 3.2%
ANIL (Raghu et al., 2020)	47.2 ± 1.3%	46.4 ± 1.9%	62.8 ± 2.2%	63.1 ± 2.4%
BOIL (Oh et al., 2021)	48.4 ± 1.2%	48.9 ± 2.4%	65.7 ± 2.4%	66.4 ± 3.1%
MALTML (ours)	45.4 ± 1.6%	47.2 ± 3.2%	60.8 ± 3.4%	63.9 ± 4.1%
MALTML-Reptile (ours)	32.9 ± 2.4%	33.6 ± 2.9%	48.3 ± 3.2%	49.1 ± 3.5%
MALTML-ANIL (partial, ours)	34.4 ± 1.4%	38.3 ± 2.4%	51.5 ± 1.7%	54.7 ± 2.2%
MALTML-ANIL (full, ours)	34.1 ± 1.9%	39.5 ± 2.6%	52.3 ± 2.0%	52.8 ± 2.9%
MALTML-BOIL (partial, ours)	37.6 ± 1.8%	39.2 ± 2.5%	55.4 ± 2.3%	57.3 ± 3.7%
MALTML-BOIL (full, ours)	37.3 ± 2.0%	38.8 ± 3.1%	56.1 ± 2.4%	58.8 ± 3.8%

Table 2: 5-way Few-shot and 5-way Few-task Few-shot classification on held-out tiered-ImageNet families, before and after adaptation to the test families. The ± shows 95% confidence intervals over families. Here MALTML-Reptile refers to the model utilizing the first-order approximation for the inner meta-steps. MALTML-ANIL (partial) and MALTML-ANIL (full) refer to the models updating only the head in task specific adaptation and task as well as family specific adaptation, respectively. Similarly, MALTML-BOIL (partial) and MALTML-BOIL (full) refer to updating only the body in the corresponding settings. The size of the confidence intervals is relatively large due to the limited number of 8 test families.

6.3. 2D Navigation

Inspired by the 2D navigation problem in Finn et al. (2017), we evaluate an equivalent family-tuning before task-tuning setting. Specifically, before goal task fine-tuning, we give

the agent few related tasks to meta-tune on. All tasks are sampled from a box region of $x \in (-.5, .5)$ and $y \in (-.5, .5)$. A family with lower bound a and upper bound b constitutes of tasks with goals within the area $x \in (a, b)$ and $y \in (a, b)$. a and b are randomly sampled from $[-0.5, 0.5]$. We use the same hyperparameters as in Finn et al. (2017), a meta-learning rate of 0.01, an outer training batch size of 1, and 20 support tasks and 20 validation tasks. To allow large changes in the policy due to meta updates, we use policy gradient for both inner task specfic updates and inner meta-steps, while the topmost updates are obtained using TRPO (Schulman et al., 2015). Appendix C provides illustrations. Table 3 shows the results on the 2D navigation problem.

Model	**init(ialization)**	**1 task update**	**1 meta-update init**	**1 task update**
MAML (Finn et al., 2017)	-12.6	-11.7	-13.6	-11.3
MALTML (ours)	-38.4	-27.7	-13.3	-10.2

Table 3: Average return for (meta-)adapting the 2D Navigation RL policy. Higher return is better, with 0 being the maximal reward.

6.4. Continual Regression

Following Javed and White (2019), we extend the few-shot regression task to continual learning over trajectories of tasks consisting of sequences of sinusoid functions of randomly sampled frequencies, amplitudes, and phases. The families (task distributions) are constructed by using the same randomly sampled frequency for each task (trajectory) within a family. Further details about the architecture, hyperparameters, and the sampling procedure are provided in Appendix . We refer to the online extensions of MALTML and MAML as OMALTML and OML, respectively. To ensure fair comparison, we utilize the same architecture and inner loop updates as Javed and White (2019), who split the training network into representation learning and prediction learning modules, with only the prediction learning modules being updated in the inner loop. Figures 3, 4(a), and 4(b) demonstrate the effectiveness of OMALTML in adapting to unseen families of trajectories to improve the subsequent task specific adaptation on the given family.

6.5. Continual Reinforcement Learning

We evaluate MALTML's adaptation capabilities on continual Reinforcement Learning through the half-cheetah locomotion task with oscillating gravity recently introduced by Kaplanis et al. (2020). Each family is constructing by adding an oscillating function to a baseline gravity value of -12. The oscillating function corresponds to a sinusoid function with a phase, frequency and amplitude sampled uniformly from the ranges $[0, 2.0], [0.1, 5.0], [0, \pi]$ which describes the evolution of the environment's gravity with time. While obtaining the gravity's value, the timesteps are divided by the horizon (set to 200), to scale the input range to $[0, 1]$. Within each family, different tasks correspond to different goal velocities. Following

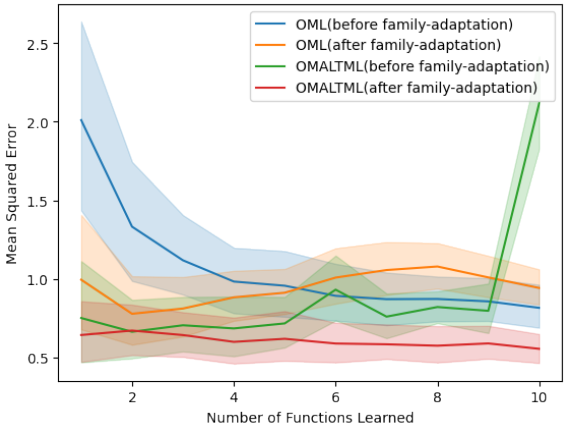


Figure 3: Mean squared error for OMALTML and OML averaged over 50 sinusoid families (frequencies) with 95% confidence interval drawn by 1,000 bootstraps. The x-axis denotes the number of functions in a given trajectory having being utilized to provide task(trajectory) specific updates to the model.

Kaplanis et al. (2020), the gravity's value is appended to the input state at each timestep. Table 4 shows the continual RL results.

Model	**init(ialization)**	**1 task update**	**1 meta-update init**	**1 task update**
MAML (Finn et al., 2017)	-145.5	-74.8	-113.1	-70.0
MALTML (ours)	-156.9	-87.8	-113.9	-62.8

Table 4: Continual Reinforcement Learning: average return for Half-Cheetah with Oscillating Gravity RL policy. Higher return is better.

7. Findings

7.1. Rapid Meta-Learning

7.1.1. MALTML

As demonstrated through the results in Figure 3, and Tables 3,4, MALTML successfully learns to adapt to new task distributions using a meta step on a few tasks for continual regression. However, the improvements due to test time adaptation to new families is significantly limited for classification tasks (Tables 1, 2). We hypothesize that this is primarily due to the limited number of training families in realistic image classification datasets unlike the continuous distribution over families available for continual regression. Moreover, we found that the

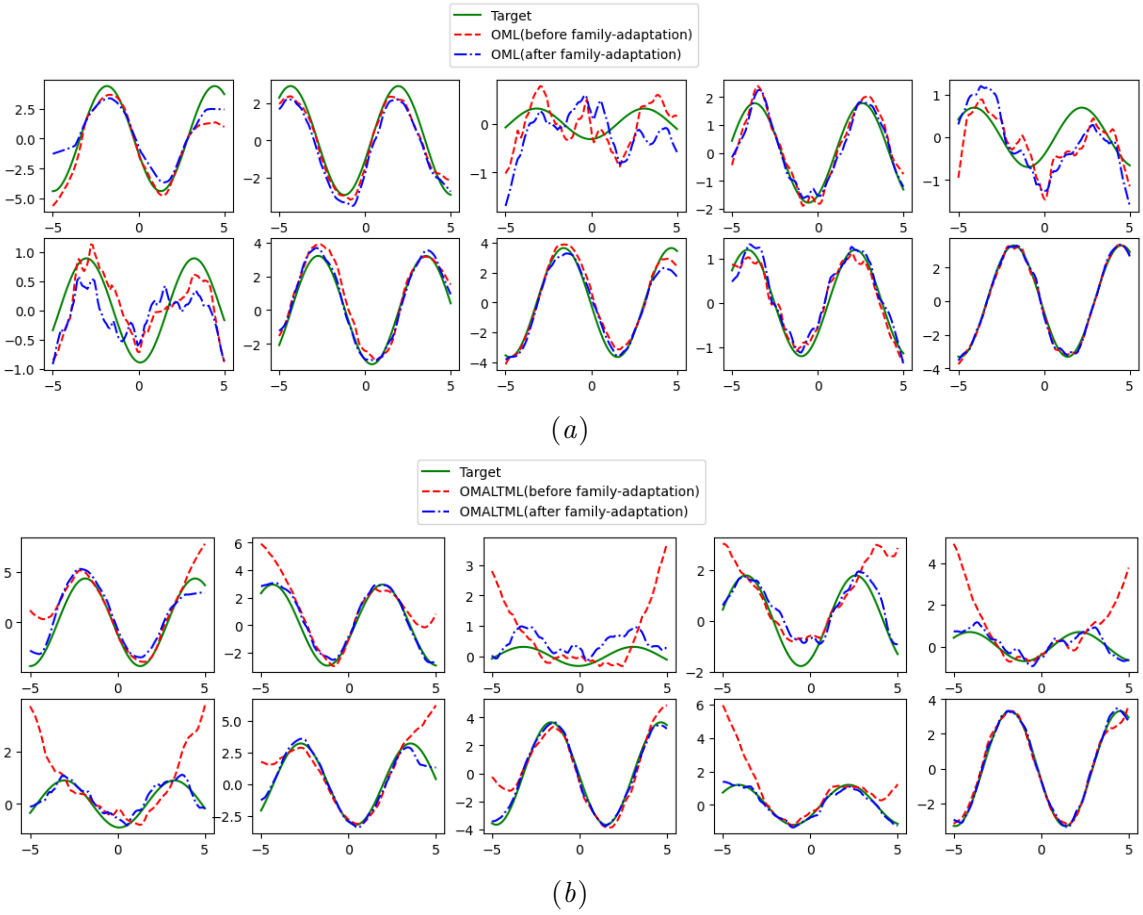


Figure 4: Few-Task Few-shot adaptation in a continual regression task for a randomly sampled trajectory before and after adaptation to another trajectory from the same family: (a) MAML (b) MALTML. Note that MALTML's ability to adapt to the trajectory signficantly improves due to the meta-step while MAML's output function demonstrates negligible change.

magnitude of improvement due to the meta-step is quite sensitive to hyperparameters such as the step size used for the meta-step during training.

7.1.2. MAML

Through the results in Tables 1,2,3,4, and Figure 3, we observe that unlike MALTML, MAML is unable to leverage the meta-steps on unseen tasks to improve adaptability on test task distributions. Thus training MALTML to quickly adapt through meta-steps is beneficial for adaptation to unseen task distributions.

7.2. Rapid (Meta-)Learning is More Important than Feature Reuse

The Omniglot results in Table 1 and the tiered-ImageNet results in Table 2 show that *rapid learning* is a dominant factor in achieving good classification accuracy. Specifically, for learning to meta-learn on Omniglot, there is a clear gap in performance of 2% to 3% between the setting where only the features are trained to rapidly meta-learn and learn (MALTML-BOIL) and the setting where only the classifier is trained to rapidly meta-learn and learn (MALTML-ANIL). Note however, that although *rapid meta-learning* (using few related tasks) does seem to bring an overall performance improvement, it is not significant on Omniglot. For meta-learning, the performance improvement of BOIL over ANIL is not significant, but still consistently present, in agreement with the experiments in Oh et al. (2021).

7.3. Overfitting

Through the tiered-ImageNet results in Table 2, we observe that even though MALTML achieves significant improvement through the meta-step, the mean accuracy on test families can still be significantly lower than training task distribution top accuracies. We hypothesize that this occurs due to the model overfitting on the small number of training families (20 for tiered-ImageNet). For a continuous distribution over families, such as in continual regression, MALTML obtains significant improvement over the baselines in adapting to unseen families.

7.4. Reptile

As shown in Table 2, using the first order Reptile (Nichol et al., 2018) approximation for the inner meta-steps leads to a significant drop in the performance. This suggests a need to design more effective first order approaches for the few-task few-shot learning task.

8. Documented Modifications

1. Instead of creating a custom hierarchical dataset from all the classes defined in Imagenet, we directly utilized the hierarchy introduced in tiered-ImageNet (Ren et al., 2018) based on 34 categories defined on 608 classes (779,165 images). This was done to ensure computational feasibility of the experiments and the absence of any baselines for meta-learning on the full Imagenet dataset.

2. Contrary to the originally proposed oracle baselines, we primarily used MAML as a baseline, since we found that the number of families and the number of tasks within each family were insufficient to obtain reliable family specific or adaptation based oracles. Thus, following the literature, e.g., (Raghu et al., 2020; Oh et al., 2021), we don't include the oracles, and focus on contrasting our approach with real-world baselines.

3. We adjusted the 2D navigation task to have families defined by a single upper and lower bound. This eased the sampling of families and tasks.

4. Self-Supervised Learning: upon deeper investigation based on the proposal, it has been determined that few-shot self-supervised learning using augmentations is not much

closer to a real-life setting than the fully supervised case. Namely, its performance is very dependent on the augmentations used. These augmentations can only be tuned on a validation set with labels. To get to an adequate set of augmentations, one still needs a validation set consisting of (many) labeled validation families and tasks. This can be seen, e.g., in Khodadadeh et al. (2019), where the augmentations are tuned on the Omniglot test set. Hence, we have not conducted this set of experiments.

Acknowledgements

We would like to express gratitude to the anonymous reviewers, as well as Matthias Grossglauser for insightful comments. Arnout Devos acknowledges funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 754354.

References

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your MAML. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJGven05Y7.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020.

Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. Torchmeta: A Meta-Learning library for PyTorch, 2019. URL https://arxiv.org/abs/1909.06576. Available at: https://github.com/tristandeleu/pytorch-meta.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, 2019.

Christos Kaplanis, Claudia Clopath, and Murray Shanahan. Continual reinforcement learning with multi-timescale replay, 2020.

Siavash Khodadadeh, Ladislau Boloni, and Mubarak Shah. Unsupervised meta-learning for few-shot image classification. In *Advances in Neural Information Processing Systems*, pages 10132–10142, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

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

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. {BOIL}: Towards representation change for few-shot learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=umIdUL8rMH.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgMkCEtPB.

Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, 2019.

Mengye Ren, Sachin Ravi, Eleni Triantafillou, Jake Snell, Kevin Swersky, Josh B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJcSzz-CZ.

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems*, pages 7957–7968, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/schulman15.html.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning, 2017.

Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J. Lim. Toward multimodal model-agnostic meta-learning, 2018.

Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Multimodal model-agnostic meta-learning via task-aware modulation. In *Advances in Neural Information Processing Systems*, pages 1–12, 2019.

Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2019.

Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. Automated relational meta-learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Amir Zamir, Alexander Sax, Teresa Yeo, Oğuzhan Kar, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas Guibas. Robust learning through cross-task consistency, 2020.

Appendix A. Omniglot

Omniglot consists of 1623 handwritten characters from 50 alphabets and 20 examples per character. Identical to Vinyals et al. (2017), the grayscale images are resized to 28x28. However, to not have overlapping alphabets between training and testing sets, we resample the characters and alphabets according to Section A.2.

A.1. Hyperparameters

Table 5 provides the hyperparamters used for Omniglot training and testing.

Table 5: Omniglot hyperparameter summary.

	Few-shot			Few-task Few-shot		
Hyperparameter	MAML	ANIL	BOIL	MALTML	MALTML-ANIL	MALTML-BOIL
Model architecture	Conv-4	Conv-4	Conv-4	Conv-4	Conv-4	Conv-4
Image input size	28×28	28×28	28×28	28×28	28×28	28×28
Outer optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Outer step size	0.001	0.001	0.001	0.001	0.001	0.001
Query examples/task	15	15	15	15	15	15
Support Tasks/family	n.a.	n.a.	n.a.	5	5	5
Query Tasks/family	n.a.	n.a.	n.a.	15	15	15
Training batch size	16 tasks	16 tasks	16 tasks	4 families	4 families	4 families
Meta-tuning optimizer	n.a.	n.a.	n.a.	SGD	SGD	SGD
Meta-tuning step size	n.a.	n.a.	n.a.	0.4	0.1	0.4
Meta-tuning steps	n.a.	n.a.	n.a.	1	1	1
Meta-tune last layer	n.a.	n.a.	n.a.	✓	✓	
Meta-tune backbone	n.a.	n.a.	n.a.	✓		✓
Fine-tuning optimizer	SGD	SGD	SGD	SGD	SGD	SGD
Fine-tuning step size	0.1	0.1	0.1	0.4	0.1	0.4
Fine-tuning steps	1	5	1	1	5	1
Fine-tune last layer	✓	✓		✓	✓	
Fine-tune backbone	✓		✓	✓		✓

A.2. training (30), validation (4), test (16) alphabets

training *Anglo-Saxon_Futhorc, Armenian, Atlantean, Aurek-Besh, Balinese, Bengali, Braille, Burmese_(Myanmar), Cyrillic, Early_Aramaic, Ge_ez, Grantha, Gujarati, Inuktitut_(Canadian_Aboriginal_Syllabics), Japanese_(hiragana), Japanese_(katakana), Kannada, Keble, Korean, Latin, Malay_(Jawi_-_Arabic), Malayalam, Manipuri, Mkhedruli_(Georgian), Ojibwe_(Canadian_Aboriginal_Syllabics), Sanskrit, Sylheti, Syriac_(Estrangelo), Tagalog, Tifinagh*

validation *Asomtavruli_(Georgian), Futurama, Oriya, ULOG*

testing *Alphabet_of_the_Magi, Angelic, Arcadian, Atemayar_Qelisayer, Avesta, Blackfoot_(Canadian_Aboriginal_Syllabics), Glagolitic, Greek, Gurmukhi, Hebrew, Mongolian, N_Ko, Old_Church_Slavonic_(Cyrillic), Syriac_(Serto), Tengwar, Tibetan*

Appendix B. Tiered-ImageNet

B.1. Hyperparameters

Table 6 provides the hyperparameters used for tiered-ImageNet training and testing.

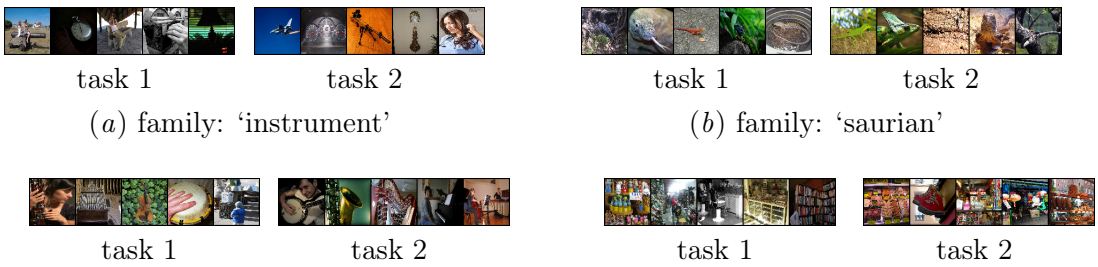
B.2. training (20), validation (6), test (8) categories

training *'game equipment', 'electronic equipment', 'snake, serpent, ophidian', 'tool', 'establishment', 'passerine, passeriform bird', 'aquatic bird', 'primate', 'garment', 'terrier', 'saurian', 'ungulate, hoofed mammal', 'feline, felid', 'restraint, constraint', 'building, edifice', 'musical instrument, instrument', 'instrument', 'protective covering, protective cover, protect', 'hound, hound dog', 'craft'*

validation *'motor vehicle, automotive vehicle', 'furnishing', 'machine', 'durables, durable goods, consumer durables', 'mechanism', 'sporting dog, gun dog'*

testing *'working dog', 'aquatic vertebrate', 'vessel', 'geological formation, formation', 'obstruction, obstructor, obstructer, impedimen', 'solid', 'substance', 'insect'*

B.3. Family and task visualization



(*a*) family: 'instrument' (*b*) family: 'saurian'

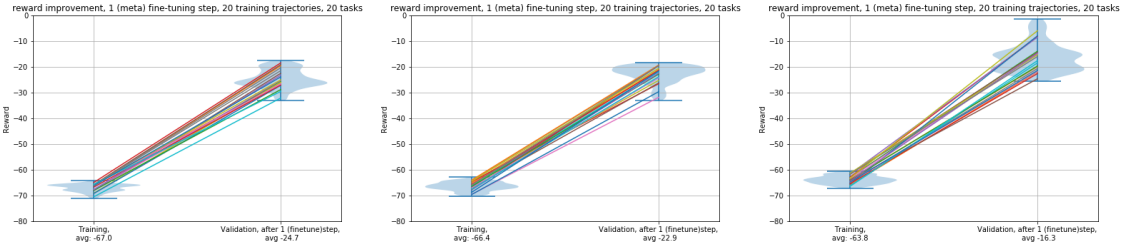
(*c*) family: 'musical instrument, instrument' (*d*) family: 'establishment'

Figure 5: Visualization of the tiered-ImageNet family distribution

Appendix C. 2D navigation

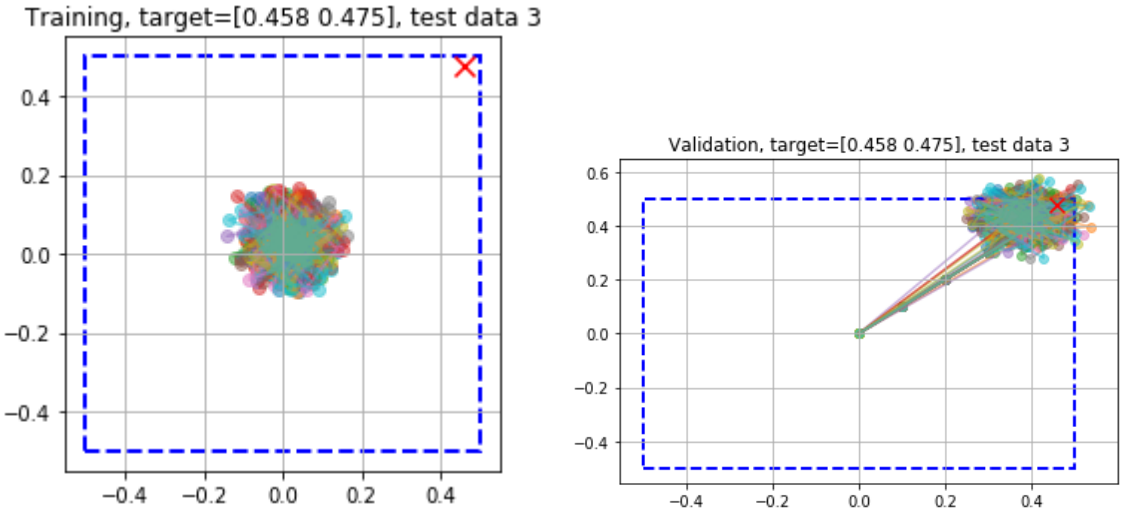
	Few-shot			Few-task Few-shot		
Hyperparameter	MAML	MALTML	MALTML-Reptile	MALTML	MALTML-ANIL (partial)	MALTML-ANIL (full)
Model architecture	Conv-4	Conv-4	Conv-4	Conv-4	Conv-4	Conv-4
Image input size	84×84	84×84	84×84	84×84	84×84	84×84
Outer optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Outer step size	0.001	0.001	0.001	0.001	0.001	0.001
Query examples/task	15	15	15	15	15	15
Support Tasks/family *(shots)*	n.a.	4(1), 2(5)	4(1), 2(5)	4(1), 2(5)	4(1), 2(5)	4(1), 2(5)
Query Tasks/family *num(shot)*	n.a.	8(1), 4(5)	8(1), 4(5)	8(1), 4(5)	8(1), 4(5)	8(1), 4(5)
Training batch size *num(shot)*	8(1), 4(5)	1 family	1 family	1 family	1 family	1 family
Meta-tuning optimizer	n.a.	SGD	SGD	SGD	SGD	SGD
Meta-tuning step size	n.a.	0.05	0.05	0.05	0.05	0.05
Meta-tuning steps	n.a.	1	1	1	1	1
Meta-tune entire network	✓	✓	✓	✓	×	✓
Fine-tuning optimizer	SGD	SGD	SGD	SGD	SGD	SGD
Fine-tuning step size	0.5	0.5	0.5	0.5	0.5	0.5
Fine-tuning steps	1	1	1	1	1	1
Fine-tune entire network	✓	✓	✓	✓	×	×

Table 6: Tiered-ImageNet summary.



(a) (b) (c)

Figure 6: Effect on MAML init of (a) task-finetuning (b) meta-update (TRPO) + task-finetuning (c) meta-update (grad) + task-finetuning.



(a) (b)

Figure 7: Effect on MAML init of (a) 1 gradient meta-tuning step (b) additionally a goal task fine-tuning step.

Appendix D. Continual Regression

D.1. Hyperparameters

We borrow the architecture and inner loop hyperparameters from Javed and White (2019). For the inner meta-steps, we use a step size of 0.2. Due to computational constraints, we use only one trajectory each as query and support task and sample one family for each outer update. For each trajectory, we use the same number of tasks (10) and minibatches (40) per task as Javed and White (2019). The frequencies, amplitudes, and phases are sampled uniformly from the ranges $[1.0, 3.0]$,$[0.1, 5.0]$, $[0.0, \pi]$, respectively.