# Learning to Learn and Remember Super Long Multi-Domain Task Sequence

Zhenyi Wang[1], Li Shen[2], Tiehang Duan[3], Donglin Zhan[4], Le Fang[1], Mingchen Gao[1]
[1] State University of New York at Buffalo, USA [2]JD Explore Academy, Beijing, China
[3]Meta, Seattle, WA, USA [4]Columbia University, New York, NY, USA
{zhenyiwa, lefang, mgao8}@buffalo.com, {mathshenli, tiehang.duan}@gmail.com, dz2478@columbia.edu

## Abstract

*Catastrophic forgetting (CF) frequently occurs when learning with non-stationary data distribution. The CF issue remains nearly unexplored and is more challenging when meta-learning on a sequence of domains (datasets), called sequential domain meta-learning (SDML). In this work, we propose a simple yet effective learning to learn approach, i.e., meta optimizer, to mitigate the CF problem in SDML. We first apply the proposed meta optimizer to the simplified setting of SDML, domain-aware meta-learning, where the domain labels and boundaries are known during the learning process. We propose dynamically freezing the network and incorporating it with the proposed meta optimizer by considering the domain nature during meta training. In addition, we extend the meta optimizer to the more general setting of SDML, domain-agnostic meta-learning, where domain labels and boundaries are unknown during the learning process. We propose a domain shift detection technique to capture latent domain change and equip the meta optimizer with it to work in this setting. The proposed meta optimizer is versatile and can be easily integrated with several existing meta-learning algorithms. Finally, we construct a challenging and large-scale benchmark consisting of 10 heterogeneous domains with a super long task sequence consisting of 100K tasks. We perform extensive experiments on the proposed benchmark for both settings and demonstrate the effectiveness of our proposed method, outperforming current strong baselines by a large margin.*

## 1. Introduction

Catastrophic forgetting (CF) [47] frequently occurs when learning with data distribution shift. The CF issue is largely overlooked in the more challenging problem setting, i.e., meta-learning on a sequence of domains, where domain shift occurs sequentially when the model meta-learns on *a large number* of tasks and aims to generalize to the unseen tasks from previous domains. This has significant implications for real-world applications, for example:

- Robot learns on many visual recognition tasks, where each task may consist of only a small number of labeled image data. It may sequentially go through numerous environments as illustrated in Fig. 1. When adapting to a new environment, the skills learned in previous environments may be easily forgotten.
- For a personalized dialogue/recommendation system [44, 50], where learning the personal model for each user is viewed as an individual task, the user base may shift over time, e.g., the system is first deployed for Canadian users, then the company extends its market to Europe. While learning about European users, the system may quickly forget previous Canadian users' habits.

We generalize and formulate the above problem setting as *sequential domain meta-learning* (SDML), where a model is required to make proper decisions based on only a few training examples with the underlying environments/domains constantly changing. Recent work reveals that catastrophic forgetting often occurs when transferring a meta-learning model to a new context [55, 79]. We expect that adjustments to a new environment/domain should not erase the learned knowledge from old ones. On the other hand, most existing works of continual learning [58, 61] can only mitigate the forgetting on a short sequence of (typically less than 50) tasks. These continual learning methods are infeasible to be directly applied in SDML with a super long task sequence consisting of (at least) 100K tasks, which is our main focus.

We propose to learn a meta optimizer to mitigate the catastrophic forgetting issue during the learning process. Intuitively, more important parameters for previous domains should be updated more slowly to avoid forgetting and less important parameters could be updated faster for efficient learning of the current domain. To achieve this goal, we store a small number of tasks in a memory buffer and calculate the gradient of the meta loss for the memory tasks with respect to the learnable learning rates at each iteration.

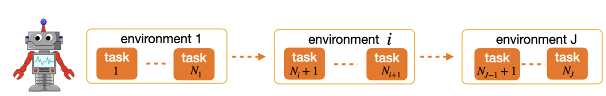



Figure 1. Demonstration of SDML learning scenario

The gradient corresponds to the degree of catastrophic interference between current tasks and previous memory tasks. The meta optimizer dynamically adjusts the learning rates according to this gradient. Next, we apply the proposed optimizer to the simplified setting of SDML, *domain-aware meta-learning*, where the domain labels and boundaries are known during the learning process. To incorporate the fact of heterogeneous domain nature (where different domains do not share categories) in SDML, we propose to dynamically freeze the network and integrate it with the proposed meta optimizer during meta training. In addition, we extend the meta optimizer to the more general setting of SDML, *domain-agnostic meta-learning*, where domain labels and boundaries are unknown during the learning process. We propose a domain shift detection technique to capture latent domain change and equip the meta optimizer with it.

Most existing meta-learning benchmarks are designed for the stationary setting and are not suitable for evaluating the CF issue in SDML. To evaluate the proposed methods, we construct a large-scale and challenging dataset consisting of a sequence of 10 heterogeneous domains for the SDML setting. We integrate the proposed methods with both representative metric-based and gradient-based meta-learning approaches. Results on both *domain-aware* and *domain-agnostic* meta-learning demonstrate that our method significantly outperforms related strong baselines by a large margin. Our contributions can be summarized as the following:

- To our best knowledge, we are the first to tackle the CF issue when learning on *a super long task sequence of at least 100K tasks with sequential domain shift.*
- We propose a meta optimizer to address the catastrophic forgetting issue of SDML, a more challenging problem than existing continual learning methods trying to address.
- We apply the proposed meta optimizer to the domain-aware and domain-agnostic meta-learning setting of SDML. The proposed method is versatile and can be easily integrated into both metric-based and gradient-based meta-learning approaches.
- To verify the effectiveness of the proposed method, we construct a challenging and large-scale dataset consisting of 10 heterogeneous domains. Comprehensive experiments demonstrate that our method outperforms related strong baselines by a large margin.

# 2. Related Work

## 2.1. Continual Learning

Continual learning (CL) [3, 9, 14, 33, 43, 47, 58, 78] focuses on learning a sequence of tasks without forgetting previous ones. CL merely sequentially learns on a small number of tasks (typically less than 50 tasks) and aims to *generalize to the testing data from all the previous tasks*. Continual few-shot learning (CFSL) [8] is an application of CL to few-shot learning, usually within a single domain, and focuses on remembering previously learned few-shot tasks when learning on the current one. The purpose of [8] is to evaluate existing meta-learning methods under the conditions of CFSL. SDML is significantly different from CL and CFSL due to the high variability underlying a large number of dynamically formed few-shot tasks (more than 100K tasks) with domain shift. Thus, it is infeasible for a CL or CFSL model to remember so large number of tasks during the learning process. In addition, CL can also be applied on a sequence of datasets (domains) [63], however, whose goal is to generalize to the testing data from all the previous (small number of) tasks. By contrast, in SDML, the goal is to generalize to the unseen tasks from all the previous domains by training on a large number of tasks with significant sequential domain shift, which makes our SDML distinct from existing works.

Task/domain/class incremental learning [69] are three common scenarios in task-aware CL. Later on, more general cases of CL, i.e., task-free CL [4, 27, 52], focuses on the case that task identities and boundaries are both unknown during both training and testing. These learning scenarios focus on task-level data distribution shifts, and each class has a large amount of data. They aim to generalize to *seen* task. In contrast, SDML focuses on: 1) task-level data shift; 2) domain-level task distribution shift; 3) few-shot learning challenges. The goal is to generalize to *unseen* testing tasks.

Continuous domain adaptation [42] is a recent application of continual learning to domain adaptation. The difference discussion compared to SDML is presented in Appendix G.

## 2.2. Meta Learning

Most existing works of meta-learning [6, 19, 21, 29, 38, 65, 70, 75, 81] focus on stationary task distributions. In contrast, SDML focuses on non-stationary task distributions with sequential domain shifts. Directly applying these meta-learning methods to SDML would incur significant forgetting of previous knowledge without additional mechanisms. Online meta-learning (OML) [22], assumes tasks arrive sequentially and aims to achieve better performance on future tasks. SDML is fundamentally different from and more challenging than OML since OML *ignores* the CF issue during meta-learning by *storing the data from all the previous tasks in memory in their small-scale* problem setting. However, we consider a more practical setting by storing a small number of tasks in memory in our *large-scale* setting. Jerfel et al. [30] extend MAML and use Dirichlet process mixtures to group similar training tasks together but cannot scale to our large-scale setting. MOCA [26] focuses on meta-learning in online learning, i.e., utilizing more context from previous data to improve future sequential prediction;they are entirely different from SDML. CAVIA [82] uses a separate context vector for fast task adaptation, while SDML focuses on domain-level task distribution remembering and adaptation.

Continual meta-learning [1, 13, 54, 74] is to apply the meta-learning techniques for continual learning. They either depends on context switch [13], fixed-size state-vector [1], or encoding the recent context by RNN [54]. These would be highly insufficient to address the CF issue in our very long task sequence.

Incremental few-shot learning (IFSL) [24, 55, 79] aims to learn new categories while retaining knowledge on old categories within a single domain and assumes *unlimited* access to the base categories. SDML is substantially different from IFSL. Detailed discussion is provided in Appendix G.

### 2.3. Learning Rate Adaptation

Dynamically updating the learning rates in meta-learning is not new. Meta-SGD [41] learns per parameter learning rates for MAML to accelerate the training process. Lee and Choi [39] and flennerhag et al. [23] propose to learn the gradient update rule for meta-learning. Similar to Meta-SGD [41], Gupta et al. [25] apply meta-learning for task parameters adaptation to mitigate forgetting in CL. Different from these works, which operate on *task parameters*, our work operates on domain level *meta parameters*.

## 3. Problem Setup

For SDML (Figure 1), we first provide some definitions.

**Definition 1.** ***non-stationary heterogeneous domains***. *A sequence of domains, $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_J$, arrive sequentially. Each domain $\mathcal{D}_i$ is represented as a labeled dataset $\{(\boldsymbol{x}^k, \boldsymbol{y}^k)\}_{k=1}^{I_i}$ with $I_i$ labeled datapoints; where $\boldsymbol{x}^k$ are the datapoints and $\boldsymbol{y}^k$ are the labels. All the domains do not share class labels. $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_J$ are called non-stationary heterogeneous domains.*

**Definition 2.** ***non-stationary task sequence***. *From time 1 to $N_1$, we randomly sample mini-batch tasks $\mathcal{T}_t$ at each time $t$ from task distribution $P(\mathcal{D}_1)$; from time $N_1 + 1$ to $N_2$, we randomly sample mini-batch tasks $\mathcal{T}_t$ at each time $t$ from task distribution $P(\mathcal{D}_2)$; from time $N_{i-1} + 1$ to $N_i$, we randomly sample mini-batch tasks $\mathcal{T}_t$ at each time $t$ from task distribution $P(\mathcal{D}_i)$, where $P(\mathcal{D}_i)$ is the collection of a large number of tasks in domain $\mathcal{D}_i$. This learning procedure continues until domain $\mathcal{D}_J$. The time steps $\{N_i, i = 1, 2, \cdots, J-1\}$ are the time when domain shift happens. $\mathcal{T}_1, \cdots, \mathcal{T}_t, \cdots, \mathcal{T}_{N_J}$ are called non-stationary task sequence.*

The agent stays within each domain for a long time, i.e., $|N_i - N_{i-1}|$ is a large number, to learn on a super long task sequence. Each task $\mathcal{T}$ is divided into support set $\mathcal{S}$ (training data, consisting of $K$ data examples, $\{(\boldsymbol{x}^k, \boldsymbol{y}^k)\}_{k=1}^{K}$ and query set $\mathcal{Q}$ (testing data). Our goal is to *online* meta-learn a model $f_{\boldsymbol{\theta}}$ for each arriving domain while not forgetting all previous domains, where $\boldsymbol{\theta}$ denotes the network parameters. At the end of meta training, the performance is evaluated on *many unseen tasks* sampled from $P(\mathcal{D}_1), \cdots, P(\mathcal{D}_J)$, respectively.

To this end, our framework allows allocating a small memory buffer $\mathcal{M}$ to store a small number of training tasks from previous domains. We maintain and update the memory with reservoir sampling (RS) [71], which assigns equal probability for each incoming task of being stored in $\mathcal{M}$. RS works by maintaining a reservoir of size $V$ to maintain a maximal number of $V$ tasks in memory. More details for maintaining memory buffer is provided in Appendix B.

## 4. Learning to Mitigate Forgetting in SDML

To address the CF issue in SDML, we present the proposed meta optimizer in section 4.1. In section 4.2, we apply the meta optimizer to the simplified setting of SDML, *domain-aware meta-learning*. In section 4.3, we apply the meta optimizer to the more general setting of SDML, *domain-agnostic meta-learning*.

### 4.1. Learning meta optimizer for SDML

Standard meta-learning methods, such as Prototypical Networks (PNet) [65] and MAML [21], are mostly widely studied in meta-learning literature. Given the task-specific data $\mathcal{T}_t = \{\mathcal{S}, \mathcal{Q}\}$, the task-specific loss function is $\mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_t) = P(\mathcal{Q}|\boldsymbol{\theta}, \mathcal{S})$. They update the *meta parameters* $\boldsymbol{\theta}$ by learning on current task $\mathcal{T}_t$, which we denote as the update $\boldsymbol{\theta}' = \boldsymbol{\theta} - \boldsymbol{\lambda}\frac{\partial \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_t)}{\partial \boldsymbol{\theta}}$, where $\boldsymbol{\lambda}$ are the learning rates.

In standard meta training on a single domain (dataset) in a *stationary* setting, the learning rates $\boldsymbol{\lambda}$ for the meta parameters are usually set to be *constant and equal* for all parameters during the training process. However, this would incur significant forgetting of previous knowledge if meta-learning on a sequence of domains in a *non-stationary* setting. Therefore, we propose to adaptively and separately adjust the learning rates for each meta parameter to balance between remembering previous domains and learning the current domain. Intuitively, more important parameters for previous domains should be updated slower to avoid forgetting, and less important parameters could be updated faster for efficient learning of the current domain. We store a small number of tasks from previous domains in memory $\mathcal{M}$ to meta-learn the importance, which equals the degree of interference between current tasks and memory tasks $\mathcal{M}$. We first define the concepts of transfer and catastrophic interference.

We propose a versatile framework that does not depend on which specific meta-learning algorithm to be used. It can be integrated into these standard meta-learning methods to mitigate the CF problem by dynamically adjusting the learning rates $\boldsymbol{\lambda}$ for the *meta parameters*. $\nabla_{\boldsymbol{\theta}}^i = \frac{\partial \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_i)}{\partial \boldsymbol{\theta}}$ denotes the gradient of $\mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_i)$ with respect to meta parameters. $\nabla_{\boldsymbol{\theta}}^i \cdot \nabla_{\boldsymbol{\theta}}^j = \frac{\partial \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_i)}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_j)}{\partial \boldsymbol{\theta}}$ is the dot product between a pair of task gradients. For any pair of tasks $\mathcal{T}_i$ and $\mathcal{T}_j$, *catastrophic interference* occurs between tasks $\mathcal{T}_i$ and $\mathcal{T}_j$ if $\nabla_{\boldsymbol{\theta}}^i \cdot \nabla_{\boldsymbol{\theta}}^j < 0$; *transfer* occurs between tasks $\mathcal{T}_i$ and $\mathcal{T}_j$ if $\nabla_{\boldsymbol{\theta}}^i \cdot \nabla_{\boldsymbol{\theta}}^j > 0$. The concepts of *catastrophic interference* and *transfer* are used for explaining why the proposed meta optimizer can mitigate the CF issue in SDML. Our idea for mitigating the CF in SDML is to use memory task loss as signal guidance for learning rate adjustment. The objective for training the model to avoid catastrophic forgetting becomes

$$\min_{\boldsymbol{\theta}}[\mathcal{F}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\mathcal{T}\in\mathcal{M}} \mathcal{L}_{\boldsymbol{\theta}'}(\mathcal{T})], \text{where } \boldsymbol{\theta}' = \boldsymbol{\theta} - \boldsymbol{\lambda}\frac{\partial \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_t)}{\partial \boldsymbol{\theta}},$$

where $\boldsymbol{\theta}'$ are the updated parameters by standard meta training on tasks $\mathcal{T}_t$ with gradient descent and $\boldsymbol{\lambda}$ are the learnable learning rates. $\mathcal{F}(\boldsymbol{\theta})$ is the meta loss which optimizes the generalization on memory tasks $\mathcal{M}$. The derivative of $\mathcal{F}(\boldsymbol{\theta})$ with respect to the learning rates $\boldsymbol{\lambda}$ (by chain rule) is

$$\frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\lambda}} = \frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}'}\frac{\partial \boldsymbol{\theta}'}{\partial \boldsymbol{\lambda}} = -\frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}'} \cdot \frac{\partial \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_t)}{\partial \boldsymbol{\theta}}. \tag{1}$$

Based on above estimated gradient for $\boldsymbol{\lambda}$, the learning rates $\boldsymbol{\lambda}$ are updated as:

$$\boldsymbol{\lambda} = \boldsymbol{\lambda} - \eta\frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\lambda}}. \tag{2}$$

**Algorithm 1** Meta Optimizer for SDML.

1: **REQUIRE:** A sequence of mini-batch training tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_{N_1}; \ldots; \mathcal{T}_{N_i+1}, \ldots, \mathcal{T}_{N_{i+1}}; \ldots; \mathcal{T}_{N_{J-1}+1}, \ldots, \mathcal{T}_{N_J}\}$; where $\{N_i, i = 1, 2, \cdots, J-1\}$ are the time steps when domain shift happens; Initialize learning rates $\boldsymbol{\lambda}_0$ and model parameters $\boldsymbol{\theta}$; $\eta$ is step size for updating learning rate.
2: **for** $t = 1$ to $N_J$ **do**
3: update parameters $\boldsymbol{\theta}_t$ by meta training on $\mathcal{T}_t$ $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{\lambda}_t \frac{\partial \mathcal{L}_{\boldsymbol{\theta}_t}(\mathcal{T}_t)}{\partial \boldsymbol{\theta}_t}$
4: $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t - \eta\frac{\partial \mathcal{F}(\boldsymbol{\theta}_t)}{\partial \boldsymbol{\lambda}_t}$
5: Reservoir sampling to update task memory $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{T}_t$ if decided to store the task $\mathcal{T}_t$
6: **end for**

On the RHS (right hand side) of Eq. (1), $\frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}'}$ is the meta gradient on memory tasks and $\frac{\partial \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_t)}{\partial \boldsymbol{\theta}}$ is current task gradient. In other words, $\frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\lambda}}$ reflects the catastrophic interference (or transfer) between current task and memory tasks. If $\frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}'}$ aligns with $\frac{\partial \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_t)}{\partial \boldsymbol{\theta}}$ (dot product is positive, i.e., transfer occurs), $\frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\lambda}}$ is then negative and learning rates are increased in Eq. (2); otherwise, catastrophic interference occurs and learning rates are decreased. Eq. (2) adaptively mitigate catastrophic forgetting by encouraging less catastrophic interference between current task and previous memory tasks. On the other hand, our method can be interpreted as approximately optimizing the following objective by adding additional gradient dot product regularization:

$$\min_{\boldsymbol{\theta}} \left[\mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_t) + \mathop{\mathbb{E}}_{\mathcal{T}_j\sim\mathcal{M}} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_j) - \rho \mathop{\mathbb{E}}_{\mathcal{T}_j\sim\mathcal{M}} (\nabla_{\boldsymbol{\theta}}^t \cdot \nabla_{\boldsymbol{\theta}}^j)\right] \tag{3}$$

where $j$ is the task index in memory buffer $\mathcal{M}$ and $\rho$ weighs the relative importance of the dot product term. Adaptive learning rate optimizes the third regularization term. Maximizing this term encourages parameter updates towards directions where task gradient directions align between current task gradient and memory task gradients. More discussion of this interpretation is provided in Appendix D.

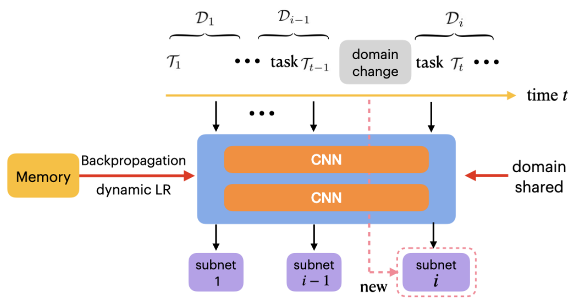
Below, we apply the proposed meta optimizer to domain-aware and domain-agnostic settings on the network illustrated in Fig. 2(a) and 2(b). We assume all domains share the same CNN-based structure for feature extraction, while the model also has the flexibility to expand a small subnet on top of domain-shared layers for newly arriving domains as a domain-specific unit. When training on the domain $\mathcal{D}_i$, only the domain-shared layers and subnet $i$ are used for meta training; other subnets $1, 2, \cdots, i-1$ are fixed to avoid forgetting of previous domain knowledge. The meta optimizer for mitigating CF in SDML is described in Algorithm 1 and the testing algorithm is provided in Appendix F.

**Remark** The proposed dynamic architecture shares some similarity with existing methods, e.g., PNN [60], DEN [78], PathNet [20] and PDEN [40]. PNN duplicates the network for each domain and grows the number of parameters quadratically. DEN expands network in *neuron level*. PathNet needs a pre-defined set of modules to learn the paths. PDEN uses a similar network as ours but aims to improve domain generalization. By contrast, ours *share and fix* a common backbone across different domains, thus significantly reducing the number of parameters and does not need pre-defined modules.

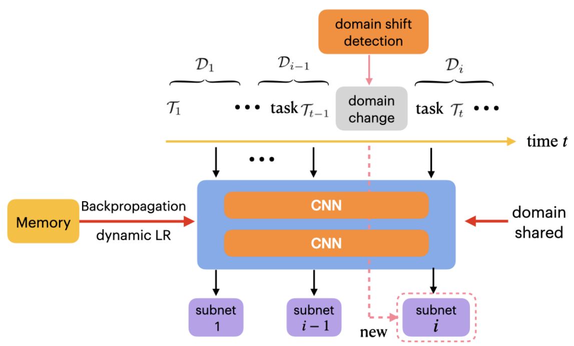
Gradient dot product information has been applied on various machine learning problem, including domain generalization [46, 64], multi-task learning [80] and continual learning [58]. These methods use *gradient product/projection* to adjust *parameters* for multi-task and domain generalization. In contrast, our method uses *task gradient* instead of data gradient. We use gradient product to adjust *learning rate* to *mitigate forgetting* in SDML.

### 4.2. Meta-optimizer for domain-aware setting

In this section, we consider a simplified setting of SDML, domain-aware meta-learning, where the domain identity associated with each task is known. Also, the time steps when domain shift happens $\{N_i, i = 1, 2, \cdots, J-1\}$ are known during meta training. Although directly applying the proposed meta optimizer in this setup can mitigate forgetting, it largely neglects the domain difficulty, which varies across different domains during meta training. For example, in SDML, suppose a complex domain comes first, followed by a simple and very dissimilar domain; much fewer iterations on the second domain is then sufficient to achieve near-best performance. The issue is that continuous training on the second domain could gradually lose the knowledge on previ-




(a) domain-aware setting model architecture




(b) domain-agnostic setting model architecture

Figure 2. Model Architecture Overview

ous domains since they share a common network structure as in Fig. 2(a). However, if we freeze the domain-shared part at some proper time, the knowledge forgetting on previous domains could be largely mitigated. This mechanism is especially beneficial when training on long domain sequences. We propose an online adaptive freeze mechanism on top of the meta optimizer to ensure a trade-off between obtaining decent performance on the current domain and preventing forgetting on previous domains.

We approximate the true posterior distribution $P(\boldsymbol{\theta}|\{\mathcal{T}_t, \mathcal{M}\})$ with approximated posterior distribution $q(\boldsymbol{\theta})$ by $\min_{\boldsymbol{\theta}} \mathbb{KL}(q(\boldsymbol{\theta})|P(\boldsymbol{\theta}|\{\mathcal{T}_t, \mathcal{M}\}))$. The variational lower bound (ELBO) can be estimated as:

$$\log P(\{\mathcal{T}_t, \mathcal{M}\}) \geq -\underset{\mathcal{T}_j \in \mathcal{M}}{\mathbb{E}} \mathbb{E}_{q(\boldsymbol{\theta})} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_j) - \mathbb{E}_{q(\boldsymbol{\theta})} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_t) + H(q(\boldsymbol{\theta}))$$
$$= \mathrm{ELBO}(\boldsymbol{\theta}),$$

where $H(q(\boldsymbol{\theta})) = -\mathbb{E}_{q(\boldsymbol{\theta})} \log q(\boldsymbol{\theta})$ is Shannon entropy of $q(\boldsymbol{\theta})$. On the RHS, the first term corresponds to the likelihood of memory tasks (measuring the forgetting on previous domains), the second term corresponds to the likelihood of current tasks, and $H(q(\boldsymbol{\theta}))$ measures the convergence and uncertainty of $q(\boldsymbol{\theta})$ on current domain. $H(q(\boldsymbol{\theta}))$ will generally decrease with gradual convergence. It encourages the posterior over $\boldsymbol{\theta}$ to have wider support and avoids fitting to current domain too much. $H(q(\boldsymbol{\theta}))$ generally does not have closed-form, and is approximated with Gaussian mean-field for simplicity. With mean $\mu$ and standard deviation $\sigma$, the entropy of Gaussian is $\log(\sigma\sqrt{2\pi e})$.

Therefore, this ELBO reflects the trade-off between forgetting on previous domains and fitting on the current domain. $\mathrm{argmax}_{\boldsymbol{\theta}} \mathrm{ELBO}(\boldsymbol{\theta})$ corresponds to reasonable freeze point. When combining with the proposed meta optimizer, the network is frozen when the ELBO does not increase for a fixed number of iterations. Interestingly, our proposed method does not need any hold-out validation set, which is desirable for our setting. Our online ELBO calculation method within finite time interval is shown in Appendix F.

### 4.3. Meta-optimizer for domain-agnostic setting

In this section, we extend the proposed meta optimizer to the more general setting of SDML, domain-agnostic meta-learning, i.e., the time steps when domain shift happens $\{N_i, i = 1, 2, \cdots, J-1\}$ are unknown during meta training. The domain-aware setting is relatively straightforward as we know when the new domain comes and the domain identity associated with each task. We thus know when to add the small subnet for the new arriving domain as shown in Figure 2(a). By contrast, in the domain-agnostic setting, when the domain shift happens is completely unknown, thus when to expand the network and add subnet is unknown. Our idea is that if we equip the meta optimizer with a domain shift detection component and a domain shift is detected, a small subnet will be added on top of the domain-shared layers, as shown in Fig. 2(b). This offers necessary flexibility in the net with the potential to learn a varying number of domains instead of fixing the network in advance. However, domain shift detection is a rather challenging problem due to (1) the highly volatile nature of few-shot tasks; (2) varying degrees of similarity across different domains. Thus, simply setting a threshold on the loss value to detect domain shift does not work well in our preliminary study. To solve this problem, we construct a latent space and enable Bayesian online changepoint detection (BOCPD) [2] to operate on it for effective domain shift detection.

**Latent space.** The few-shot task $\mathcal{T}_t$ arriving at time $t$ are converted to a task embedding $\boldsymbol{e}_t = f_{\boldsymbol{\theta}_t}(\mathcal{S})$, (also could be $f_{\boldsymbol{\theta}_t}(\mathcal{Q})$). Suppose $\mathcal{S}$ consisting of $K$ data examples, $\{(\boldsymbol{x}^k, \boldsymbol{y}^k)\}_{k=1}^K$, they are then embedded by $\boldsymbol{e}_t = f_{\boldsymbol{\theta}_t}(\{\boldsymbol{x}^k\}_{k=1}^K)$. A series of moving average embedding $\boldsymbol{E}_t$ is computed in the form of $\boldsymbol{E}_t = \alpha \boldsymbol{e}_t + (1-\alpha)\boldsymbol{E}_{t-1}$ to reduce the variance across different few-shot tasks, the constant $\alpha$ is the smoothing factor which weighs the relative importance of current task embedding and past moving average. We keep track of the past $m$ steps $\boldsymbol{E}_{t-1}, \boldsymbol{E}_{t-2}, \cdots, \boldsymbol{E}_{t-m}$ and utilize them to form the distance metric vector $\boldsymbol{d}_t = (d(\boldsymbol{e}_t, \boldsymbol{E}_{t-1}), d(\boldsymbol{e}_t, \boldsymbol{E}_{t-2}), \cdots, d(\boldsymbol{e}_t, \boldsymbol{E}_{t-m})))$, which encodes the generalized domain information spanning across previous tasks. Each element $d(\boldsymbol{e}_t, \boldsymbol{E}_{t-i})$ denotes the Euclidean distance from current task embedding $\boldsymbol{e}_t$ to the moving average of $i$ steps ago, $\boldsymbol{E}_{t-i}$.

**Domain shift detection.** We then use the constructed latent space $\boldsymbol{d}_t$ for domain shift detection since the latent space captures the abrupt changes at the time when domain shift happens. We denote $Z_t$ ($Z_0 = 0$) as the latent domain label at time $t$ and $\boldsymbol{d}_{1:t}$ as all the latent space vector from $t = 1$ until time $t$. BOCPD is originally designed for detecting the abrupt changes (changepoints) in data stream in online setting. It estimates the posterior distribution over run lengths $l_t$, which are the number of time steps since the last time of domain shift. $l_t = 0$ corresponds to the case that domain shift happens, and $l_t = \tau > 0$ indicates the continuation of current domain and past $\tau$ batches (steps) of tasks all belong to current domain. Our goal is to estimate the posterior of $l_t$ given $\boldsymbol{d}_{1:t}$, i.e., $P(l_t|\boldsymbol{d}_{1:t})$, which can be efficiently computed by using the recursive relation of run length posterior:

$$P(l_t|\boldsymbol{d}_{1:t}) \propto \sum_{l_{t-1}} \underbrace{P(l_t|l_{t-1})}_{\text{prior}} \underbrace{P(\boldsymbol{d}_t|l_{t-1}, \boldsymbol{d}_{1:t-1})}_{\text{UPM}} P(l_{t-1}, \boldsymbol{d}_{1:t-1}). \tag{4}$$

The underlying predictive model (UPM) is modeled as exponential family. The changepoint prior is defined as:

$$P(l_t|l_{t-1}) = \begin{cases} U(l_{t-1}+1), & l_t = 0 \\ 1 - U(l_{t-1}+1), & l_t = l_{t-1}+1 \end{cases}$$

where the first case is the probability of domain shift, the second case corresponds to the probability that domain shift does not occur, i.e., the current domain continues. $U(\boldsymbol{.})$ is constant function.

Plugging the defined prior and UPM into Eq. (4), $l_t$ is inferred from $P(l_t|\boldsymbol{d}_{1:t})$. There are two cases: (1) if $l_t = 0$, a domain shift happens at time $t$ and a small subnet is appended to the domain-shared layer; the latent domain label is updated as $Z_{t+1} = Z_t + 1$. (2) $l_t = l_{t-1} + 1$, there is no domain shift; the latent domain label and network keep unchanged. We also store the average of all the task embedding in one domain as $\mathcal{E}_q$, which is used for domain identity inference during meta testing.

**Meta testing.** During meta testing, the domain identity is unknown for each testing task, thus we need to infer the domain identity to choose which subnet to use for testing. The domain inference of an unseen task $\mathcal{T}$ is performed by (1) feeding task data $\mathcal{T}$ into the domain-shared layers first, then feeding through each subnet $1, 2. \cdots, Z$ to obtain the task embedding $\boldsymbol{e}_q, q = 1, 2. \cdots, Z$, as shown in Figure 2(b); (2) inferring the domain identity with $q_o = \text{argmin}_{q\in\{1,\ldots,Z\}}\, d(\boldsymbol{e}_q, \mathcal{E}_q)$; (3) evaluating the performance on $\mathcal{T}$ with the subnet $q_o$.

The meta training algorithm is shown in Algorithm 2 and the testing algorithm is provided in Appendix F. For Algorithm 2, line 3-4 is the proposed meta optimizer for mitigating CF, and line 5-15 is used for detecting domain shift in the task stream. Specifically, line 5-7 is used for calculating the latent space $\boldsymbol{d}_t$, line 8-9 detect domain shift in latent space, line 10-14 is for updating the latent domain label and expand the network with subnet accordingly.

**Algorithm 2** Domain-agnostic meta training.

1: **REQUIRE:** A sequence of mini-batch training tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_{N_1}; \ldots; \mathcal{T}_{N_i+1}, \ldots, \mathcal{T}_{N_{i+1}}; \ldots; \mathcal{T}_{N_{J-1}+1}, \ldots, \mathcal{T}_{N_J}\}$; the time steps when domain shift happens $\{N_i, i = 1, 2, \cdots, J-1\}$ are unknown; initial learning rates $\boldsymbol{\lambda}$; size of moving window $m$; latent domain label initialized with $Z_0 = 0$; initialize moving average $\boldsymbol{E}_0 = 0$; $\mathcal{E}_0 = 0$; $\eta$ is step size for update the learning rate; weight of moving average $\alpha$; memory buffer $\mathcal{M} = \{\}$
2: **for** $t = 1$ to $N_J$ **do**
3: tasks $\mathcal{T}_t$ arrive, $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{\lambda}_t \frac{\partial \mathcal{L}_{\boldsymbol{\theta}_t}(\mathcal{T}_t)}{\partial \boldsymbol{\theta}_t}$
4: $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t - \eta \frac{\partial \mathcal{F}(\boldsymbol{\theta}_t)}{\partial \boldsymbol{\lambda}_t}$
5: calculate the task embedding $\boldsymbol{e}_t$ of $\mathcal{T}_t$ using $f_{\boldsymbol{\theta}_t}$
6: calculate moving average of $\boldsymbol{e}_t$ as $\boldsymbol{E}_t = \alpha \boldsymbol{e}_t + (1-\alpha)\boldsymbol{E}_{t-1}$
7: calculate $\boldsymbol{d}_t$
8: calculate $P(l_t|\boldsymbol{d}_{1:t})$ via Eq. (4)
9: $l^c = \underset{l_t}{\text{argmax}}\, P(l_t|\boldsymbol{d}_{1:t})$
10: **if** $l^c = 0$ **then**
11: $\mathcal{E}_{L_t} = E_t$
12: $Z_{t+1} = Z_t + 1$
13: add new small subnet to domain-shared part
14: **end if**
15: update parameters of UPM
16: Reservoir sampling to update task memory $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{T}_j$ if decided to store the task
17: **end for**

## 5. Experiments

In this section, we evaluate the efficacy of the proposed meta optimizer by applying it to solve the CF issue in SDML, in both the domain-aware and domain-agnostic settings. Our method is versatile and can be seamlessly integrated with existing meta-learning methods to mitigate the CF issue. For illustration, we evaluate the meta optimizer on current most widely used meta-learning models including **ANIL** [51] and **Prototypical Network** (**PNet**) [65]. The former is a simplified version of MAML. Below, we construct a new benchmark to simulate the domain shift in SDML.

**Benchmark with 100K tasks construction.** We construct a *large-scale* benchmark and collect 10 datasets with varying degree of similarity and difficulty, with default domain arrival order of *Quickdraw* [31], *AIRCRAFT* [45], *CUB* [77], *Miniimagenet* [70], *Omniglot* [35], *Plantae* [28], *Electronic* from Logo-2K+ [73], *CIFARFS* [10], *Fungi* [62], *Necessities* from Logo-2K+ [73]. We also provide detailed analysis by varying the domain order of the 10 datasets, and results are shown in Appendix C.

Each dataset is divided into meta-training, meta-validation and meta-testing classes subset. The subsets for each dataset are disjoint, e.g., the meta-testing classes are not seen during meta-training. More details about datasets and split are available in Appendix A. The non-stationary episodes construction are described in Section 3, the meta training episodes are sampled from the meta training classes of each dataset. The meta testing episodes are sampled from the meta testing classes to form the unseen testing tasks. We randomly sample 10K tasks from each dataset, with a total of 100K training tasks. We can sample more tasks, e.g., 20K tasks, from each dataset, thus more training iterations on each dataset; SDML becomes more challenging for 20K tasks each dataset than 10K tasks each dataset, and there will be more forgetting but with longer training time. The meta-learning model is required to sequentially meta-learn on one sequence of datasets without forgetting previous knowledge. We compare to different methods on 5-way 1-shot and 5-shot learning. More implementation details are given in Appendix B. The dataset and code are available at https://github.com/joey-wang123/SDML.git.

### 5.1. Experiments on domain-aware setting

**CL baselines.** For domain-aware case, we combine the above meta-learning base models with related strong CL baselines, including Elastic Weight Consolidation (**EWC**) [33], Hard Attention Mask (**HAT**) [63]), **UCB** [18], **A-GEM** [14], Experience Replay (Reservoir Sampling (**RS**)) [15], Meta Experience Replay (**MER**) [58], **DEGCL** [12] and **GPM** [61]. These baselines are originally developed for standard continual learning which operates on a small-scale task sequence. It is thus infeasible to directly apply these CL baselines for each task in the large-scale setting of SDML with a super long task sequence. We then instead extend these methods to SDML by making these CL baselines operate on the *meta parameters*. For convenience, we denote these combination methods by prefixing with PNet- and ANIL-, such as PNet-EWC, ANIL-EWC, etc. We also include (i) **Joint training**, which learns all the domains jointly in a multi-domain meta-learning setting and provides the performance upper bound; and (ii) **Sequential training**, which trains on each domain sequentially without any external memory and provides the degree of model forgetting.

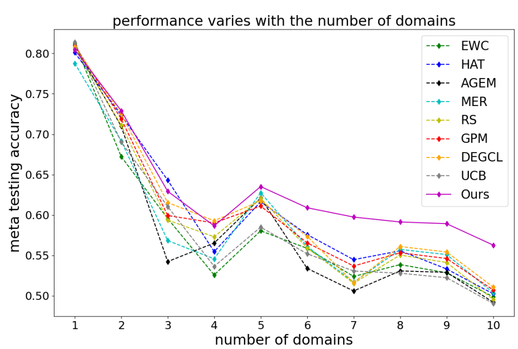



Figure 3. 5-way 5-shot meta testing performance varies with different number of training domains.

**Evaluation metrics.** ACC (accuracy) is defined as the average testing accuracy of many unseen episodes sampled from the meta testing classes of all the datasets. BWT (backward transfer) measures the amount of positive backward transfer or catastrophic forgetting on all the previous datasets evaluated at the end of meta training. Formally, ACC and BWT are defined as ACC $= \frac{1}{N}\sum_{i=1}^{N} a_{N,i}$ and BWT $= \frac{1}{N-1}\sum_{i=1}^{N-1} a_{N,i} - a_{i,i}$, respectively; where $a_{j,i}$ is defined as the average testing accuracy of many unseen episodes sampled from the meta testing subset of dataset $i$ after meta training on dataset $j$. BWT is negative indicates catastrophic forgetting of the previous domains when meta-learning on the new domain. BWT is positive indicates that learning on the new domain will improve the performance of the previous domains. Thus, the larger, the better.

Table 1. Domain-aware SDML results (PNet-based methods)

| Algorithm | 5-way 5-shot ACC | 5-way 5-shot BWT |
|---|---|---|
| PNet-Sequential | $46.83 \pm 0.10$ | $-22.95 \pm 0.12$ |
| PNet-EWC | $49.88 \pm 0.15$ | $-14.51 \pm 0.14$ |
| PNet-HAT | $50.25 \pm 0.26$ | $-16.32 \pm 0.28$ |
| PNet-UCB | $49.06 \pm 0.22$ | $-15.83 \pm 0.20$ |
| PNet-A-GEM | $49.21 \pm 0.31$ | $-20.01 \pm 0.39$ |
| PNet-RS | $49.56 \pm 0.18$ | $-18.87 \pm 0.19$ |
| PNet-MER | $50.38 \pm 0.24$ | $-15.10 \pm 0.24$ |
| PNet-DEGCL | $50.79 \pm 0.37$ | $-13.82 \pm 0.45$ |
| PNet-GPM | $49.73 \pm 0.51$ | $-14.91 \pm 0.58$ |
| Ours | $\mathbf{55.28 \pm 0.19}$ | $\mathbf{-11.15 \pm 0.27}$ |
| Joint-training | $66.32 \pm 0.18$ | N/A |

Table 2. Domain-aware SDML results (ANIL-based methods)

| Algorithm | 5-way 5-shot ACC | 5-way 5-shot BWT |
|---|---|---|
| ANIL-Sequential | $45.85 \pm 0.46$ | $-23.47 \pm 0.43$ |
| ANIL-EWC | $45.45 \pm 0.29$ | $-21.99 \pm 0.34$ |
| ANIL-HAT | $40.58 \pm 0.19$ | $-28.89 \pm 0.24$ |
| ANIL-UCB | $47.21 \pm 0.28$ | $-20.18 \pm 0.22$ |
| ANIL-A-GEM | $48.08 \pm 0.33$ | $-20.30 \pm 0.35$ |
| ANIL-RS | $46.97 \pm 0.27$ | $-21.37 \pm 0.33$ |
| ANIL-MER | $47.96 \pm 0.52$ | $-19.25 \pm 0.50$ |
| ANIL-DEGCL | $47.91 \pm 0.45$ | $-18.57 \pm 0.53$ |
| ANIL-GPM | $47.73 \pm 0.53$ | $-19.76 \pm 0.46$ |
| Ours | $\mathbf{51.56 \pm 0.21}$ | $\mathbf{-16.07 \pm 0.20}$ |
| Joint-training | $68.16 \pm 0.11$ | N/A |

**Comparisons to baselines.** Table 1 and 2 show the 5-way 5-shot learning results. Results of 5-way 1-shot classification are shown in Appendix C. We observe that our method significantly outperforms best performing baselines ranging from 3.2 % to 4.5 % for both PNet-based and ANIL-based approaches, demonstrating the effectiveness of the proposed mechanism. This performance improvement is attributed to two factors: (1) the adaptive meta optimizer to adaptively mitigate forgetting of previous domains; (2) online adaptive freeze mechanism, which properly trade-off between retraining the knowledge of previous domains and effectively learning on current domain.

**How the performance changes with different number of training domains.** Figure 3 shows how the average meta testing accuracy changes with a different number of training domains for 5-way 5-shot learning. The accuracy is evaluated at the end of training on each dataset in the dataset sequence. We find that the proposed method outperforms comparison baselines in most cases, especially when the domain sequence becomes longer.

### 5.2. Experiments on domain-agnostic setting

Since adapting most of the above CL methods to the domain-agnostic setting needs the domain identity associated with each task during meta training and testing. In contrast, for the domain-agnostic setting, the domain identity is unavailable during both training and testing. Thus, the compared baselines include: (1) Experience Replay (reservoir sampling (**RS**)) [15]; (2) **A-GEM** [14]; (3) Gradient-based Sample Selection (**GSS**) [5]. Note that GSS is originally developed for online continual learning to promote the diversity of *stored examples*. We adapt it to SDML by replacing data gradient with task gradient to encourage the diversity of *stored tasks* in memory. Results for domain-agnostic setting are shown in Table 3 and 4. Our method achieves substantial improvement ranging from 3.0 % to 5.1% compared to other models for PNet and ANIL-based methods.

Table 3. Domain-agnostic SDML results (PNet-based methods)

| Algorithm | 5-way 5-shot ACC | BWT |
|---|---|---|
| PNet-Sequential | $46.83 \pm 0.10$ | $-22.95 \pm 0.12$ |
| PNet-RS | $49.56 \pm 0.18$ | $-18.87 \pm 0.19$ |
| PNet-A-GEM | $49.21 \pm 0.31$ | $-20.01 \pm 0.39$ |
| PNet-GSS | $49.64 \pm 0.27$ | $-18.29 \pm 0.31$ |
| Ours | $\mathbf{54.67 \pm 0.20}$ | $\mathbf{-11.67 \pm 0.28}$ |
| Joint-training | $66.32 \pm 0.18$ | N/A |

Table 4. Domain-agnostic SDML results (ANIL-based methods)

| Algorithm | 5-way 5-shot ACC | BWT |
|---|---|---|
| ANIL-Sequential | $45.85 \pm 0.46$ | $-23.47 \pm 0.43$ |
| ANIL-RS | $46.97 \pm 0.27$ | $-21.37 \pm 0.33$ |
| ANIL-A-GEM | $48.08 \pm 0.33$ | $-20.30 \pm 0.35$ |
| ANIL-GSS | $47.96 \pm 0.42$ | $-20.91 \pm 0.42$ |
| Ours | $\mathbf{51.18 \pm 0.31}$ | $\mathbf{-16.85 \pm 0.29}$ |
| Joint-training | $68.16 \pm 0.11$ | N/A |

**Latent space independence analysis for BOCPD**. Since BOCPD assumes the data before and after the changepoints are independent, we analyze and evaluate the correlation (independence) before and after the domain shift with maximal information coefficient (MIC) [56] and total information coefficient (TIC) [57]. These two metrics are based on mutual information and can test the nonlinear dependency between two random variables. We put the evaluation results in Appendix C.3 and further verify that the contexts before and after the domain shift are more independent with our proposed latent space $\boldsymbol{d}_t$ than the raw task embedding $\boldsymbol{e}_t$.

**Domain shift detection accuracy**. We perform analysis for performance of domain shift detection in Appendix C, and the method can accurately detect the domain shift.

### 5.3. More results

**Compared with continual meta-learning [1, 13, 54].** The goal of continual meta-learning [1, 13, 54] is to mitigate catastrophic forgetting during meta training. Althogh [1, 13, 54] do not target for SDML, to show the effectiveness of our method, we compare to the state-of-art continual meta-learning methods, including Continual-MAML [13], MOML [1] and CPM [54]. The results are shown in Table 5.

Our method substantially outperforms these baselines. We believe that CPM uses the most recent context by RNN to remember some past knowledge. The RNN can only handle short-term remembering but cannot handle the forgetting issue in a very long-term context in SDML. MOML only focuses on a small number of tasks within a single domain by encoding previous task instances with a fixed-size state vector. However, this vector is insufficient for remembering past knowledge in SDML with a much larger number of tasks and sequential domain shift. Thus, these baselines do not perform well in SDML.

Table 5. comparisons to continual meta-learning

| Algorithm | 5-Way 1-Shot ACC | 5-Way 5-Shot ACC |
|---|---|---|
| MOML | $34.57 \pm 1.16$ | $47.29 \pm 0.73$ |
| CPM | $33.41 \pm 1.05$ | $48.72 \pm 0.81$ |
| Continual-MAML | $36.36 \pm 1.12$ | $49.81 \pm 0.89$ |
| Ours | $40.23 \pm 0.32$ | $55.28 \pm 0.19$ |

**Ablation study and analysis**. These include: 1) effectiveness of each component; 2) effect of different domain order; 3) effect of different hyperparameters, etc. Due to limited space, detailed results are placed in Appendix C.

**Limitations Discussion.** Our current memory buffer update only relies on random sampling without considering the informativeness of each incoming task. Future work includes online selecting the most informative coreset [49] from the online task stream.

## 6. Conclusion

In this work, we perform extensive studies on the challenging problem of SDML. We propose a meta optimizer to dynamically adjust the learning rate to avoid forgetting during the learning process in SDML. We adapt the meta optimizer to both domain-aware setting and domain-agnostic setting. Experiments on real-word datasets show that our proposed method significantly outperforms related strong baselines by integrating the proposed methods with PNet and MAML. Future work includes designing methods for mitigating the CF issues in SDML without memory buffer.

## References


[1] Durmus Alp Emre Acar, Ruizhao Zhu, and Venkatesh Saligrama. Memory efficient online meta learning. *Proceedings of the 38th International Conference on Machine Learning*, 2021. 3, 8, 12

[2] Ryan Prescott Adams and David J. C. MacKay. Bayesian online changepoint detection. *https://arxiv.org/abs/0710.3742*, 2007. 5

[3] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. *The 2018 European Conference on Computer Vision (ECCV)*, 2018.

[4] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[5] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems 30*, 2019. 8

[6] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, 2016. 2

[7] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *International Conference on Learning Representations*, 2019. 12

[8] Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. Defining benchmarks for continual few-shot learning. *https://arxiv.org/abs/2004.11967*, 2020. 2

[9] E. Belouadah and A. Popescu. Il2m: Class incremental learning with dual memory. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 583–592, 2019.

[10] Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *International Conference on Learning Representations*, 2019. 6, 12

[11] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. Multiwoz – a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *2018 Conference on Empirical Methods in Natural Language Processing*, 2020. 17

[12] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *34th Conference on Neural Information Processing Systems*, 2020. 7, 17

[13] Massimo Caccia, P. Rodríguez, O. Ostapenko, Fabrice Normandin, Min Lin, L. Caccia, Issam H. Laradji, I. Rish, Alexande Lacoste, D. Vázquez, and Laurent Charlin. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *Advances in neural information processing systems*, 2020. 3, 8

[14] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *Proceedings of the International Conference on Learning Representations*, 2019. 7, 8, 13, 17

[15] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *https://arxiv.org/abs/1902.10486*, 2019. 7, 8, 13, 17

[16] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019. 12

[17] Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. Torchmeta: A Meta-Learning library for PyTorch. *https://arxiv.org/abs/1909.06576*, 2019. Available at: https://github.com/tristandeleu/pytorch-meta. 13

[18] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. *Proceedings of the International Conference on Learning Representations*, 2020. 7, 13

[19] H. Edwards and A. Storkey. Towards a neural statistician. *ArXiv*, abs/1606.02185, 2017. 2

[20] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *https://arxiv.org/abs/1701.08734*, 2017. 4

[21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, 2017. 2, 3

[22] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *Proceedings of International Conference on Machine Learning*, 2019. 2, 12

[23] Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. *International Conference on Learning Representations*, 2020. 3

[24] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 19

[25] Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-maml: Look-ahead meta learning for continual learning. *Advances in Neural Information Processing Systems*, 2020. 3

[26] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. *Advances in Neural Information Processing Systems*, 2020. 2

[27] Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A. Rusu, Yee Whye Teh, and Razvan Pascanu. Task agnostic continual learning via meta learning. *https://arxiv.org/abs/1906.05201*, 2019. 2

[28] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 12

[29] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[30] Ghassen Jerfel, Erin Grant, Thomas L. Griffiths, and Katherine Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. *Advances in Neural Information Processing Systems*, 2019. 2

[31] Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. The quick, draw! – a.i. experiment. *quickdraw.withgoogle.com*, 2016. 6, 12

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014. 13

[33] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017. 7, 13

[34] Tze Leung Lai and Haipeng Xing. Sequential change-point detection when the pre- and post-change parameters are unknown. *Sequential Analysis*, 2(29):162–175, 2010. 18

[35] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. *Conference of the Cognitive Science Society*, 2011. 6, 12

[36] Ilaria Lauzana, Nadia Figueroa, and Jose Medina. Bayesian online multivariate changepoint detection algorithm. *https://github.com/epfl-lasa/changepoint-detection*, 2015. 13

[37] Hae Beom Lee, Hayeon Lee, Donghyun Na, Saehoon Kim, Minseop Park, Eunho Yang, and Sung Ju Hwang. Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks. In *International Conference on Learning Representations*, 2020. 12

[38] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[39] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. *International Conference on Machine Learning*, 2018. 3

[40] Lei Li, Ke Gao, Juan Cao, Ziyao Huang, Yepeng Weng, Xiaoyue Mi, Zhengze Yu, Xiaoya Li, and Boyang xia. Progressive domain expansion network for single domain generalization. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 4

[41] Zhenguo Li, Fengwei Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few shot learning. *ArXiv*, abs/1707.09835, 2017. 3

[42] Hong Liu, Mingsheng Long, Jianmin Wang, and Yu Wang. Learning to adapt to evolving domains. *Advances in Neural Information Processing Systems*, 2020. 2, 19

[43] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 2017.

[44] Yuanfu Lu, Yuan Fang, and Chuan Shi. Meta-learning on heterogeneous information networks for cold-start recommendation. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020. 1

[45] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *https://arxiv.org/abs/1306.5151*, 2013. 6, 12

[46] Lucas Mansilla, Rodrigo Echeveste, Diego H. Milone, and Enzo Ferrante. Domain generalization via gradient surgery. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 4

[47] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 1989. 1, 2

[48] Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. Continual learning for natural language generation in task-oriented dialog systems. *2020 Conference on Empirical Methods in Natural Language Processing*, 2020. 17

[49] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. *International Conference on Machine Learning*, 2020. 8

[50] Kun Qian and Zhou Yu. Domain adaptive dialog generation via meta learning. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. 1, 17

[51] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *International Conference on Learning Representations*, 2020. 6, 12, 13

[52] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml : An incremental task-agnostic meta-learning approach. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[53] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017. 12

[54] Mengye Ren, Michael L. Iuzzolino, Michael C. Mozer, and Richard S. Zemel. Wandering within a world: Online contextualized few-shot learning. *International Conference on Learning Representations*, 2021. 3, 8

[55] Mengye Ren, Renjie Liao, Ethan Fetaya, and Richard S. Zemel. Incremental few-shot learning with attention attractor networks. *Advances in Neural Information Processing Systems*, 2019. 1, 3, 19

[56] David N. Reshef, Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean McVean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011. 8, 14

[57] Yakir A. Reshef, David N. Reshef, Hilary K. Finucane, Pardis C. Sabeti, and Michael Mitzenmacher. Measuring dependence powerfully and equitably. *Journal of Machine Learning Research*, 17(211):1–63, 2016. 8, 14

[58] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *International Conference on Learning Representations*, 2019. 1, 4, 7, 13

[59] Ritov, Y. Decision theoretic optimality of the cusum procedure. *Annals of Statistics*, 3:1464–1469, 1990. 18

[60] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *https://arxiv.org/abs/1606.04671*, 2016. 4

[61] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *International Conference on Learning Representations*, 2021. 1, 7, 17

[62] Brigit Schroeder and Yin Cui. Fgvcx fungi classification challenge 2018. 2018. 6, 12

[63] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *Proceedings of the 35th International Conference on Machine Learning*, 2018. 2, 7, 13

[64] Yuge Shi, Jeffrey Seely, Philip Torr, Siddharth N, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. *International Conference on Learning Representations*, 2022. 4

[65] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, 2017. 2, 3, 6, 12, 13

[66] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation. *Proceedings of the International Conference on Learning Representations*, 2020. 12

[67] Tze Leung Lai. Asymptotic optimality of invariant sequential probability ratio tests. *Annals of Statistics*, 2:318–333, 1981. 18

[68] Tze Leung Lai. Information bounds and quick detection of parameter changes in stochastic systems. *IEEE Transactions on Information Theory*, 44(7):2917–2929, 1998. 18

[69] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. 2019. 2

[70] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *https://arxiv.org/pdf/1606.04080.pdf*, 2016. 2, 6, 12

[71] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 1985. 3

[72] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J. Lim. Multimodal model-agnostic meta-learning via task-aware modulation. *Proceedings of the Advances in Neural Information Processing Systems*, 2019. 12

[73] Jing Wang, Weiqing Min, Sujuan Hou, Shengnan Ma, Yuanjie Zheng, Haishuai Wang, and Shuqiang Jiang. Logo-2k+: A large-scale logo dataset for scalable logo classification. *AAAI Conference on Artificial Intelligence*, 2019. 6, 12

[74] Zhenyi Wang, Tiehang Duan, Le Fang, Qiuling Suo, and Mingchen Gao. Meta learning on a sequence of imbalanced domains with difficulty awareness. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8947–8957, October 2021. 3

[75] Zhenyi Wang, Yang Zhao, Ping Yu, Ruiyi Zhang, and Changyou Chen. Bayesian meta sampling for fast uncertainty adaptation. *International Conference on Learning Representations*, 2020. 2

[76] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962. 19

[77] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. 2010. 6, 12

[78] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *International Conference on Learning Representations*, 2018. 4

[79] Sung Whan Yoon, Do-Yeon Kim, Jun Seo, and Jaekyun Moon. Xtarnet: Learning to extract task-adaptive representation for incremental few-shot learning. *International Conference on Machine Learning*, 2020. 1, 3, 19, 20

[80] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 2020. 4

[81] Yufan Zhou, Zhenyi Wang, Jiayi Xian, Changyou Chen, and Jinhui Xu. Meta-learning with neural tangent kernels. *International Conference on Learning Representations*, 2021. 2

[82] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. *Proceedings of the 36th International Conference on Machine Learning*, 97:7693–7702, 09–15 Jun 2019. 2