Meta Learning on a Sequence of Imbalanced Domains with Difficulty Awareness

Zhenyi Wang [1], Tiehang Duan [1], Le Fang[1], Qiuling Suo[1] and Mingchen Gao[1]

[1]Department of Computer Science and Engineering, University at Buffalo, SUNY
[1]{*zhenyiwa, tiehangd, lefang, qiulings, mgao8*}*@buffalo.edu*

Abstract

Recognizing new objects by learning from a few labeled examples in an evolving environment is crucial to obtain excellent generalization ability for real-world machine learning systems. A typical setting across current meta learning algorithms assumes a stationary task distribution during meta training. In this paper, we explore a more practical and challenging setting where task distribution changes over time with domain shift. Particularly, we consider realistic scenarios where task distribution is highly imbalanced with domain labels unavailable in nature. We propose a kernel-based method for domain change detection and a difficulty-aware memory management mechanism that jointly considers the imbalanced domain size and domain importance to learn across domains continuously. Furthermore, we introduce an efficient adaptive task sampling method during meta training, which significantly reduces task gradient variance with theoretical guarantees. Finally, we propose a challenging benchmark with imbalanced domain sequences and varied domain difficulty. We have performed extensive evaluations on the proposed benchmark, demonstrating the effectiveness of our method.

1. Introduction

Learning from a few labeled examples to acquire skills for new task is essential to achieve machine intelligence. Take object recognition in personalized self-driving system as an example [10]. Learning each user's personal driving preference model forms one task. The system is first deployed in the small city, Rochester. The company later extends its market to New York. The user base of New York is much larger than that of Rochester, causing domain imbalance. Also, after adapting to New York users, the learned user behavior from Rochester will be easily forgotten. Similar scenarios occur when learning to solve NLP tasks on a sequence of different languages [13] with imbalanced resources of different languages.

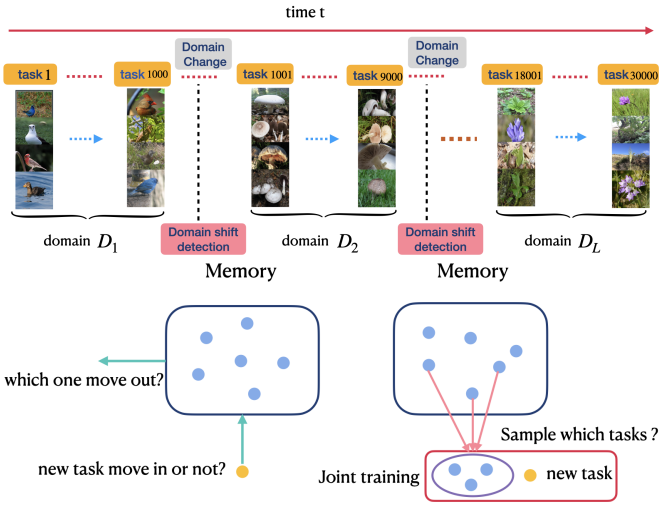


Figure 1: Illustration of meta learning for few shot object recognition on a sequence of imbalanced domains. Our focused problems including domain change detection, how to manage memory and sample memory tasks for joint training with streaming tasks.

Meta learning is a promising approach for solving such few-shot learning problems. One common assumption of current models is that the task distribution is stationary during meta training. However, real world scenarios (such as the above self-driving system) are more complex and often involve learning across different domains (environments), with challenges such as: (1) task distributions change among different domains; (2) tasks from previous domains are usually unavailable when training on a new domain; (3) the number of tasks from each domain could be highly imbalanced; (4) domain difficulty could vary significantly in nature across the domain sequence. An example is shown in Figure 1. Directly applying current meta learning models to such scenarios is not suitable to tackle these challenges, e.g., the object recognition accuracy of meta learned neural networks

generally deteriorates significantly on previous context after adapting to a new environment [23, 46, 63].

In this work, we cope with such challenges by considering a more realistic problem setting that (1) learning on a sequence of domains; (2) task stream contains significant domain size imbalance; (3) domain labels and boundaries remain unavailable during both training and testing; (4) domain difficulty is non-uniform across the domain sequence. We term such problem setup as *Meta Learning on a Sequence of Imbalanced Domains with Varying Difficulty* (MLSID).

MLSID requires the meta learning model both adapting to a new domain and retaining the ability to recognize objects from previous domains. To tackle this challenging problem, we adopt replay-based approaches, i.e., a small number of tasks from previous domains are maintained in a memory buffer. Accordingly, there are two main problems that need to be solved: (1) how to determine which task should be stored into the memory buffer and which to be moved out. To address this problem, we propose an adaptive memory management mechanism based on the domain distribution and difficulty, so that the tasks in memory buffer could maximize the retained knowledge of previous domains; (2) how to determine which tasks to sample from memory during meta training. We propose an efficient adaptive task sampling approach to accelerate meta training and reduce gradient estimation variance according to our derived optimal task sampling distribution. Our intuition is that not all tasks are equally important for joint training at different iterations. It is thus desirable to dynamically determine which tasks to sample and to be jointly trained with current tasks to mitigate catastrophic forgetting at each training iteration.

Our contributions are summarized as following:

- To our best knowledge, this is the first work of meta learning on a sequence of imbalanced domains. For convenient evaluation of different models, we propose a new challenging benchmark consisting of imbalanced domain sequences.
- We propose a novel mechanism, *"Memory Management with Domain Distribution and Difficulty Awareness"*, to maximize the retained knowledge of previous domains in the memory buffer.
- We propose an efficient adaptive task sampling method during meta training, which significantly reduces gradient estimation variance with theoretical guarantees, making the meta training process more stable and boosting the model performance.
- Our method is orthogonal to specific meta learning methods and can be integrated with them seamlessly. Extensive experiments with gradient-based and metric-based meta learning methods on the proposed benchmark demonstrate the effectiveness of our method.

2. Problem Setting

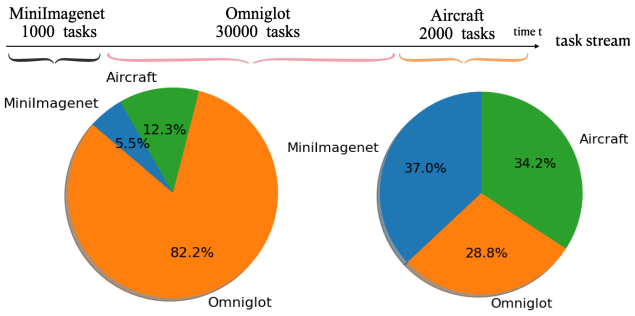
A series of mini-batch training tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N$ arrive sequentially, with possible domain shift occurring in the stream, i.e., the task stream can be segmented by continual latent domains, $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_L$. $\mathcal{T}_t$ denotes the mini-batch of tasks arrived at time t. The domain identity associated with each task remains unavailable during both meta training and testing. Domain boundaries, i.e., indicating current domain has finished and the next domain is about to start, are unknown. This is a more practical and general setup. Each task $\mathcal{T}$ is divided into training and testing data $\{\mathcal{T}^{train}, \mathcal{T}^{test}\}$. Suppose $\mathcal{T}_t^{train}$ consists of K examples, $\{(\boldsymbol{x}^k, \boldsymbol{y}^k)\}_{k=1}^K$, where in object recognition, $\boldsymbol{x}^k$ is the image data and $\boldsymbol{y}^k$ is the corresponding object label. We assume the agent stays within each domain for some consecutive time. Also, we consider a simplified setting where the agent will not return back to previous domains and put the contrary case into future work. Our proposed learning system maintains a memory buffer $\mathcal{M}$ to store a small number of training tasks from previous domains for replay to avoid forgetting of previous knowledge. Old tasks are not revisited during training unless they are stored in the memory $\mathcal{M}$. The total number of tasks processed is much larger than memory capacity. At the end of meta training, we randomly sample a large number of *unseen few-shot tasks* from each latent domain, $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_L$ for meta testing. The model performance is the average accuracy on all the sampled tasks.

3. Methodology

3.1. Conventional Reservoir Sampling and Its Limitations

Reservoir sampling (RS) [57, 15] is a random sampling method for choosing k samples from a data stream in a single pass without knowing the actual value of total number of items in advance. Straightforward adoption of RS here is to maintain a fixed memory and uniformly sample tasks from the task stream. Each task in the stream is assigned equal probability $\frac{n}{N}$ of being moved into the memory buffer, where n is the memory capacity size and N is the total number of tasks seen so far. However, it is not suitable for the practical scenarios previously described, with two major shortcomings: (1) the task distribution in memory can be skewed when the input task stream is highly imbalanced in our setting. This leads to under-representation of the minority domains; (2) the importance of each task varies as some domains are more difficult to learn than others. This factor is also not taken into account with RS.

To address the above issues, we propose to first detect domain change in the input task stream to associate each task with a latent domain label. We then present a new mechanism, called *Memory Management with Domain Distribution and Difficulty Awareness* by utilizing the associated latent



(a) reservoir sampling (b) proposed memory management

Figure 2: An example of (a) reservoir sampling and (b) proposed memory management method jointly considering domain distribution and difficulty when meta learning on task stream from three latent domains.

domain label with each task. For simple illustration, we construct an imbalanced input task stream from Miniimagenet, Omniglot and Aircraft as shown in Figure 2. Evidently, the resulting distribution of stored tasks with RS is highly imbalanced and dramatically influenced by the input task stream distribution. In contrast, our memory management mechanism balances the three domain proportions by jointly considering domain distribution and difficulty.

Model Summary: We first illustrate on our domain change detection component in Section 3.2, which is used for (1) managing and balancing tasks in the memory buffer by incorporating the task difficulty (defined in Section 3.4) to determine whether the new incoming task should be moved into the memory and which old task should be moved out of memory in Section 3.3; (2) adaptive sampling tasks from memory buffer during meta training by dynamically adjusting the sampling probability of each task in the memory according to the task gradient for mitigating catastrophic forgetting in Section 3.4.

3.2. Online Domain Change Detection

Online domain change detection is a difficult problem due to: (1) few shot tasks are highly diverse within a single domain; (2) there are varying degree of variations at domain boundaries across the sequence. In our initial study, we found that it is inadequate to set a threshold on the change of mini-batch task loss value for detecting domain change. We thus construct a low dimensional projected space and perform online domain change detection on this space.

Projected space Tasks $\mathcal{T}_t$ are mapped into a common space $\boldsymbol{o}_t = f_{\boldsymbol{\theta}_t}(\mathcal{T}_t^{train}) = \frac{1}{K}\sum_{k=1}^{K} f_{\boldsymbol{\theta}_t}(\{\boldsymbol{x}^k\})$ where K is the number of training data and $f_{\boldsymbol{\theta}_t}$ is the CNN embedding network. The task embedding could be further refined by incorporating the image labels, e.g., concatenating the word embedding of the image categories with image embedding. We leave this direction as interesting future work. To reduce the variance across different few shot tasks and capture the general domain information, we compute the exponential moving average of task embedding $\boldsymbol{O}_t$ as $\boldsymbol{O}_t = \alpha\boldsymbol{o}_t + (1-\alpha)\boldsymbol{O}_{t-1}$, where the constant α is the weighting multiplier which encodes the relative importance between current task embedding and past moving average. A sliding window stores the past m (m is a small number) steps moving average, $\boldsymbol{O}_{t-1}, \boldsymbol{O}_{t-2}, \cdots, \boldsymbol{O}_{t-m}$, which are used to form the low dimensional projection vector $\mathbf{z}_t$, where the i-th dimensional element of $\mathbf{z}_t$ is the distance between $\boldsymbol{o}_t$ and $\boldsymbol{O}_{t-i}$, $d(\boldsymbol{o}_t, \boldsymbol{O}_{t-i})$. The projected m dimensional vector $\mathbf{z}_t$ captures longer context similarity information spanning across multiple consecutive tasks.

Online domain change detection At each time t, we utilize the above constructed projected space for online domain change detection. Assume we have two windows of projected embedding of previous tasks $\mathcal{U}^B = \{\mathbf{z}_{t-2B}, \mathbf{z}_{t-2B+1}, \cdots, \mathbf{z}_{t-B-1}\}$ with distribution Q and $\mathcal{V}^B = \{\mathbf{z}_{t-B}, \mathbf{z}_{t-B+1}, \cdots, \mathbf{z}_t\}$ with distribution R, where B is the window size. In other words, $\mathcal{V}^B$ represents the most recent window of projection space (test window) and $\mathcal{U}^B$ represents the projection space of previous window (reference window). $\mathcal{U}^B$ and $\mathcal{V}^B$ are non-overlapping windows. For notation clarity and presentation convenience, we use another notation to denote the $\mathcal{U}^B = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_B\}$ and $\mathcal{V}^B = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_B\}$, i.e., $\mathbf{u}_i = \mathbf{z}_{t-2B+i-1}$ and $\mathbf{v}_i = \mathbf{z}_{t-B+i-1}$. Our general framework is to first measure the distance between the two distributions Q and R, $d(Q, R)$; then, by setting a threshold b, the domain change is detected when $d(Q, R) > b$. Here, we use Maximum Mean Discrepancy (MMD) to measure the distribution distance. Following [37], the MMD distance between Q and R is defined as:

$$\text{MMD}[\mathcal{F}, Q, R] := \sup_{f\in\mathcal{F}} \{\mathbb{E}_{\mathbf{u}\sim Q}[f(\mathbf{u})] - \mathbb{E}_{\mathbf{v}\sim R}[f(\mathbf{v})]\} \quad (1)$$

U-statistics [25] can be used for estimating MMD^2:

$$W_t^B = \text{MMD}^2[\mathcal{U}^B, \mathcal{V}^B] = \frac{1}{B(B-1)} \sum_{i\neq j}^{B} h(\mathbf{u}_i, \mathbf{u}_j, \mathbf{v}_i, \mathbf{v}_j) \quad (2)$$

and $h(\cdot)$ is defined as:

$$h(\mathbf{u}_i, \mathbf{u}_j, \mathbf{v}_i, \mathbf{v}_j) = k(\mathbf{u}_i, \mathbf{u}_j) + k(\mathbf{v}_i, \mathbf{v}_j) - k(\mathbf{u}_i, \mathbf{v}_j) - k(\mathbf{u}_j, \mathbf{v}_i) \quad (3)$$

where $k(\cdot, \cdot)$ is RKHS kernel. In this paper, we assume RBF kernel $k(x, x') = exp(-||x - x'||^2/2\sigma^2)$ is used.

The detection statistics at time t is W_t^B. If Q and R are close, W_t^B is expected to be small, implying small proba-

bility of existence of domain change. If Q and R are significantly different distributions, W_t^B is expected to be large, implying higher chance of domain shift. Thus, W_t^B characterizes the chance of domain shift at time t. We then test on the condition of $W_t^B > b$ to determine whether domain change occurs, where b is a threshold. Each task $\mathcal{T}_t$ is associated with a latent domain label L_t, $L_0 = 0$. If $W_t^B > b$, $L_t = L_{t-1} + 1$, i.e., a new domain arrives (Note that the actual domain changes could happen a few steps ago, but for simplicity, we could assume domain changes occur at time t); otherwise, $L_t = L_{t-1}$, i.e., the current domain continues. We leave the more general case with domain revisiting as future work. How to set the threshold is a non-trivial task and is described in the following.

Setting the threshold Clearly, setting the threshold b involves a trade-off between two aspects: (1) the probability of $W_t^B > b$ when there is no domain change; (2) the probability of $W_t^B > b$ when there is domain change. As a result, if the domain similarity and difficulty vary significantly, simply setting a fixed threshold across the entire training process is highly insufficient. In other words, adaptive threshold of b is necessary. Before we present the adaptive threshold method, we first show the theorem which characterizes the property of detection statistics W_t^B in the following.

Algorithm 1 Online Domain Change Detection (ODCD).

Require: stream of detection statistics W_t^B; constant ρ; desired quantile (significance level) δ; Initialize $\mu_0 = 0$ and $\mu_0^{(2)} = 0$

Function ODCD (W_t^B, ρ, δ)
$\mathbf{d} = False$; // indicator of domain shift
$\mu_t = (1-\rho)\mu_{t-1} + \rho(W_t^B)^2$
$\mu_t^{(2)} = (1-\rho)\mu_{t-1}^{(2)} + \rho(W_t^B)^4$
$\sigma_t = \sqrt{\mu_t^{(2)} - \mu_t^2}$
if $W_t^B > \mu_t + \delta\sigma_t$ **then**
  $\mathbf{d} = True$; //there is domain shift at time t
end if
return $\mathbf{d}$
EndFunction

Theorem 1 *Assume* $\mathbf{z}_i$ *are drawn i.i.d. from* Q. *Suppose that* $\mathbb{E}_Q||k(\mathbf{z},\cdot)||^4 < \infty$. *Set* $\mu \overset{def}{=} \mathbb{E}_Q k(\mathbf{z},\cdot)$ *and* $K(\mathbf{z},\mathbf{z}') \overset{def}{=} \langle k(\mathbf{z},\cdot) - \mu, k(\mathbf{z}',\cdot) - \mu\rangle$. *Suppose the eigenvalue* ξ_l *and eigenvectors* ϕ_l^2 *of* K *satisfy* $\xi_l \geq 0$ *and* $\mathbb{E}_Q\phi_l^2 < \infty$ *such that* $K(\mathbf{z},\mathbf{z}') = \sum_{l\geq 1}\xi_l\phi_l(\mathbf{z})\phi_l(\mathbf{z}')$ *and* $\langle\phi_l,\phi_{l'}\rangle = \boldsymbol{I}_{l=l'}$. *Then,*

$$W_t^B \xrightarrow{d} \beta\sum_{l\geq 1}\xi_l Z_l^2 \tag{4}$$

Where $\xrightarrow{d}$ means converge in distribution and $(Z_l)_{l\geq 1}$ is a collection of infinite independent standard normal random variables and β is a constant. The theorem and proof follow from [50, 30]. We can observe that W_t^B asymptotically follows a distribution formed by a weighted linear combination of independent normal distribution. By Lindeberg's central limit theorem [55], it is reasonable to assume W_t^B is approximately Gaussian distribution. The problem is thus reduced to estimate its mean μ_t and σ_t. The adaptive threshold b, following from [30], can be estimated by online approximation, $b = \mu_t + \delta\sigma_t$, where δ is a constant and set to be the desired quantile of the normal distribution. This adaptive method for online domain change detection is shown in Algorithm 1.

3.3. Memory Management with Domain Distribution and Difficulty Awareness

In this section, we design the memory management mechanism for determining which task to be stored in the memory and which task to be moved out. The mechanism, named *Memory Management with Domain Distribution and Difficulty Awareness* (M2D3), jointly considers the difficulty and distribution of few shot tasks in our setting. M2D3 first estimates the probability of the current task $\mathcal{T}_t$ to be moved into the memory. The model will then determine the task to be moved out in the event that a new task move-in happens. To improve efficiency, we utilize the obtained latent domain information associated with each task (as described in previous section) to first estimate this move-out probability at cluster-level before sampling single task, as in Figure 3.

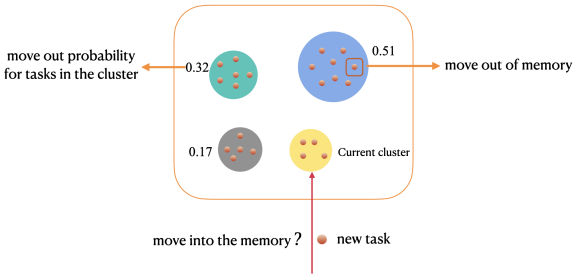


Figure 3: Illustration on the memory management process. Each colored circle represents one cluster in the buffer and each dot denotes one task.

Here we define the notations involved in the following method description. Each task $\mathcal{T}_t$ in the memory is associated with a latent domain label L_t and all the tasks with the same latent domain label form one cluster. $\mathcal{M}_i$ denotes the cluster formed by all the tasks with latent domain label i in memory $\mathcal{M}$, $n_i = |\mathcal{M}_i|$ denotes the number of tasks in $\mathcal{M}_i$ and $n = |\mathcal{M}|$ denotes the total number of tasks in memory, and $\mathcal{I}_i$ denotes the importance score of cluster $\mathcal{M}_i$.

Probability of new task moving into memory When the new task $\mathcal{T}_t$ arrives, the chance of $\mathcal{T}_t$ being stored in memory is estimated, with the basic principle being the

more incremental knowledge is brought by $\mathcal{T}_t$, the higher the probability of $\mathcal{T}_t$ being stored. This depends on the difficulty and prevalence of current latent domain. We propose an approach on top of this principle to estimate this probability. The score function of $\mathcal{T}_t$ is defined as:

$$S_{new} = (1 - \frac{n_{L_t}}{n})\mathcal{I}_t^T \tag{5}$$

Where $\mathcal{I}_t^T$ represents the importance for $\mathcal{T}_t$, which is defined as the task-specific gradient norm in Section 3.4. n_{L_t} denotes the number of tasks of current latent domain cluster in memory buffer. $\frac{n_{L_t}}{n}$ denotes the prevalence of current latent domain in memory. $\mathcal{I}_i$ represents the importance for cluster $\mathcal{M}_i$, which is defined as the cluster-specific gradient norm G_i in Section 3.4 (The computation is shared and corresponding terms are computed only once.). The importance of in-memory tasks is defined as $M_s = \frac{1}{L_t - 1}\sum_{i=1}^{L_t-1}\frac{n_i}{n}\mathcal{I}_i$. The score function of in-memory tasks is defined as:

$$S_{mem} = \frac{n_{L_t}}{n}M_s \tag{6}$$

The probability of moving $\mathcal{T}_t$ into the memory is:

$$P_{in} = \frac{e^{S_{new}}}{e^{S_{new}} + e^{S_{mem}}} \tag{7}$$

This task selection mechanism maximizes the incremental knowledge of each task added into memory.

Algorithm 2 Memory Management with Domain Distribution and Difficulty Awareness (M2D3).

Require: mini-batch training tasks $\mathcal{T}_t$; memory tasks $\mathcal{M}$; domain label L_{t-1}

Function M2D3 $(\mathcal{M}, \mathcal{T}_t)$
calculate the probability $\mathcal{P}_{in}$ to move $\mathcal{T}_t$ into memory as Eq. 7.
calculate detection statistics of W_t^B
$\mathbf{d} = \text{ODCD}(W_t^B, \rho, \delta)$; detect domain change by Alg. 1.
if $\mathbf{d}$ **then**
  $L_t = L_{t-1} + 1$
  $\mathcal{M}_{L_t} = \{\}$
end if
if memory $\mathcal{M}$ is not full **then**
  $\mathcal{M}_{L_t} \leftarrow \mathcal{M}_{L_t} \cup \mathcal{T}_t$
else
  if $\mathcal{T}_t$ is moved into memory by Eq. 7 **then**
    calculate the move-out probability for each cluster $\mathcal{P}_t^i$ and sample cluster j according to Eq. 8 and 9.
    sample task from $\mathcal{M}_j$ to move out of memory.
    move $\mathcal{T}_t$ into memory $\mathcal{M}_{L_t} \leftarrow \mathcal{M}_{L_t} \cup \mathcal{T}_t$
  end if
end if
return updated memory buffer $\mathcal{M}$
EndFunction

Probability of existing tasks moving out of memory To improve the efficiency of removing the tasks currently in memory, we perform a hierarchical sampling approach. We perform sampling first at cluster level before focusing on individual tasks, as shown in Figure 3. The estimated probability is correlated with both the size of each cluster in memory and its importance. The factor for each cluster $\mathcal{M}_i$ is defined as:

$$\mathcal{A}_i \propto -(1 - \frac{n_i}{n})\mathcal{I}_i \tag{8}$$

The moving out probability for each cluster $\mathcal{M}_i$ at time t is then defined as

$$\mathcal{P}_t^i = \frac{e^{\mathcal{A}_i}}{\sum_{i=1}^{i=L_t-1} e^{\mathcal{A}_i}} \tag{9}$$

The complete mechanism is summarized in Algorithm 2.

3.4. Adaptive Memory Task Sampling for Training

During meta training, a mini-batch of tasks are sampled from the memory and are jointly trained with current tasks to mitigate catastrophic forgetting. Direct uniform sampling tasks from memory incurs high variance, and results in unstable training [31, 9]. On the other hand, our intuition for non-uniform task sampling mechanism is that the tasks are not equally important for retaining the knowledge from previous domains. The tasks that carry more information are more beneficial for the model to remember previous domains, and should be sampled more frequently. To achieve this goal, we propose an efficient adaptive task sampling scheme in memory that accelerates training and reduces gradient estimation variance. As shown in Figure 4, the sampling probability of Miniimagenet and Aircraft are adjusted and increased based on the scheme suggesting the importance of these domains are higher than that of Omniglot for retaining knowledge.

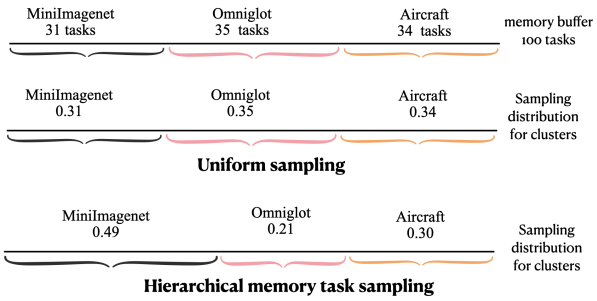


Figure 4: A simple example of uniform task sampling and our adaptive memory task sampling method for sampling tasks from memory buffer during meta training.

With the task specific loss function $\mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_i) = P(\mathcal{T}_i^{test}|\boldsymbol{\theta}, \mathcal{T}_i^{train})$. The optimization objective at time t is defined as minimizing on the loss function of both the new tasks and memory tasks $\mathcal{H}(\boldsymbol{\theta}) = \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_t) + \sum_{\mathcal{T}_i \in \mathcal{M}} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_i)$.

At time t, our proposed adaptive sampling mechanism assigns each task $\mathcal{T}_i \in \mathcal{M}$ a probability q_i^t such that $\sum_{i=1}^{i=n} q_i^t = 1$, we then sample $\mathcal{T}_{i_t}$ based on the distribution $\mathbf{q}_t = (q_1^t, q_2^t, \cdots, q_n^t)$. We temporally omit the subscript (superscript) t for the following theorem for notation clarity.

Theorem 2 *Let $\mathbf{p}(\mathcal{T})$ be the distribution of the tasks in memory $\mathcal{M}$. Then,*

$$\mathbb{E}_{\mathbf{p}(\mathcal{T})} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}) = \mathbb{E}_{\mathbf{q}(\mathcal{T})}[\frac{\mathbf{p}(\mathcal{T})}{\mathbf{q}(\mathcal{T})} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T})] = \Omega \quad (10)$$

Let $\mathbb{V}_{\mathbf{q}}[\Omega]$ denotes the covariance of the above estimator associated with $\mathbf{q}$. Then, the trace of $\mathbb{V}_{\mathbf{q}}[\Omega]$ is minimized by the following optimal $\mathbf{q}^*$

$$\mathbf{q}^*(\mathcal{T}) = \frac{\mathbf{p}(\mathcal{T})||\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T})||_2}{\int \mathbf{p}(\mathcal{T})||\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T})||_2}. \quad (11)$$

In particular, if no prior information is available on task distribution, uniform sampling of tasks from memory is adopted and $\mathbf{p}(\mathcal{T}) = \frac{1}{n}$, $\mathbf{q}^*(\mathcal{T}_i) = \frac{||\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_i)||_2}{\sum_{j=1}^{n} ||\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_j)||_2}$. Thus, $w(\mathcal{T}_i) = \frac{\mathbf{p}(\mathcal{T}_i)}{\mathbf{q}(\mathcal{T}_i)} = \frac{1}{n\,\mathbf{q}(\mathcal{T}_i)}$

Proof is provided in Appendix C. The parameters are updated as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta w_i^t \nabla_{\boldsymbol{\theta}_t} \mathcal{L}_{\boldsymbol{\theta}}(\mathcal{T}_{i_t}) \quad (12)$$

Where η is the learning rate, $w_i^t = \frac{1}{nq_i^t}$. Similar to standard SGD analysis [24], we define the convergence speed of meta training as the shrinkage of distance to optimal parameters $\boldsymbol{\theta}^*$ between two consecutive iterations $C = -\mathbb{E}_{\mathbf{q}_t}[||\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}^*||_2^2 - ||\boldsymbol{\theta}_t - \boldsymbol{\theta}^*||_2^2]$. Following [29, 1], it can be expressed as:

$$C = 2\eta(\boldsymbol{\theta}_t - \boldsymbol{\theta}^*)\Omega - \eta^2 \Omega^T \Omega - \eta^2 \mathrm{Tr}(\mathbb{V}_{\mathbf{q}_t}[\Omega]) \quad (13)$$

Theorem 2 illustrates the optimal task sampling distribution for reducing the gradient variance is proportional to the per-task gradient norm. Minimizing the gradient variance (last term of RHS in Eq.13) as in Theorem 2 also speeds up the convergence (maximize C) as a byproduct. However, it is computationally prohibitive to compute this distribution. We therefore propose efficient approximation to it.

By Section 3.2, each memory task is associated with a latent cluster label. Utilizing this property, we can first sample R (small) tasks from each cluster, then calculate the gradient norm for each cluster as G_i.

By doing so, the computational efficiency of the optimal task sampling distribution will be significantly improved. The sampling probability for each cluster is calculated as:

$$\mathcal{Z}_t^i = \frac{n_i G_i}{\sum_{j=1}^{j=L_t} n_j G_j} \quad (14)$$

The sampling scheme is to first sample cluster indexes from memory according to Eq. 14, then randomly sample tasks from the specified clusters. We name this task sampling scheme as *ada**P**tive m**E**mory **T**ask **S**ampling* (PETS).

Eq. 14 illustrates that the original sampling distribution of each cluster (measured by the frequency of each cluster in the memory buffer) is weighted by the corresponding importance of each cluster measured by the gradient norm G_i. In practice, the computational efficiency can be further improved by computing the sampling distribution every s steps with the same distribution during each time interval. PETS is summarized in Algorithm 3.

Algorithm 3 Adaptive Memory Task Sampling (PETS).

Require: A sequence of mini-batch training tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N$; memory buffer $\mathcal{M}$; model parameters $\boldsymbol{\theta}$;

1: **for** $t = 1$ to N **do**
2: **for** each cluster $\mathcal{M}_j$ in $\mathcal{M}$ **do**
3: sample mini-batch tasks from cluster $\mathcal{M}_j$ and calculate gradient norm G_j for $\mathcal{M}_j$.
4: **end for**
5: calculate the task sampling distribution from each cluster $\mathcal{M}_j$ as in Eq. 14.
6: sample tasks $\mathcal{B}$ from $\mathcal{M}$ according to the distribution $\mathcal{Z}_t$ as in Eq. 14.
7: update $\boldsymbol{\theta}$ by meta training on $\mathcal{T}_t \cup \mathcal{B}$
8: Memory tasks update $\mathcal{M} = \mathrm{M2D3}(\mathcal{M}, \mathcal{T}_t)$
9: **end for**

4. Related Work

Meta Learning: Meta learning [49] focuses on rapidly adapting to unseen tasks by learning on a large number of similar tasks. Representative works include [56, 51, 19, 20, 22, 48, 7, 41, 6, 40, 36, 60, 45, 52, 64], etc. All of these methods work on the simplified setting where task distributions are stationary during meta training. Completely different from these works, we focus on the more challenging setting where task distributions are non-stationary and imbalanced.

Online meta learning [21] stores all previous tasks in online setting to avoid forgetting with small number of tasks. [27] use Dirichlet process mixtures (DPM) to model the latent tasks structure and expand network. By contrast, ours focuses on mitigating catastrophic forgetting with single model when meta learning on imbalanced domain sequences with only limited access to previous domains.

Multi-domain meta learning [53, 54, 58] assume tasks from all domains are available during meta training. We focus on the case that each domain in an imbalanced domain sequence sequentially arrives.

Continual Learning: Continual learning (CL) aims to maintain previous knowledge when learning on sequentially arriving data with distribution shift. Many works focus on

mitigating catastrophic forgetting during the learning process. Representative works include [38, 14, 47, 62, 33, 42, 18, 2, 11, 4], etc. *Continual few-shot learning* [8] (CFSL) focuses on remembering previously learned few-shot tasks in a single domain. To our best knowledge, the replay-based approach to imbalanced streaming setting of continual learning has been only considered in [5, 17, 32]. Different from these works, which focus on learning on *a small number of tasks* and aim to generalize to previous tasks, our work focuses on the setting where the model learns on *a large number of tasks* with domain shift and imbalance, and aims to generalize to the *unseen tasks* from previous domains without catastrophic forgetting instead of remembering on a specific task.

Incremental and Continual Few-shot Learning: Incremental few-shot learning [23, 46, 63] aim to learn new categories while retaining knowledge on old categories within a single domain and assume access to the base categories is unlimited. This paper, by contrast, requires good generalization to *unseen* categories in previous domains and access to previous domains is limited.

Continual-MAML [12] aims for online fast adaptation to new tasks while accumulating knowledge on old tasks and assume previous tasks can be unlimited revisited. MOCA [26] works in online learning and learns the experiences from previous data to improve sequential prediction. In contrast, ours focuses on generalizing to previous domain when learning on a large number of tasks with sequential domain shift and limited access to previous domains.

5. Experiments

Our method is orthogonal to specific meta learning models and can be integrated into them seamlessly. For illustration, we evaluate our method on representative meta learning models including (1) gradient-based meta learning **ANIL** [43], which is a simplified model of MAML [20]; (2) metric-based meta learning **Prototypical Network** (**PNet**) [51]. Extension to other meta learning models is straightforward.

Baselines: (1) **sequential training**, which learns the latent domains sequentially without any external mechanism and demonstrates the model forgetting behavior; (2) **reservoir sampling (RS)** [57]; (3) **joint offline training**, which learns all the domains jointly in a multi-domain meta-learning setting; (4) **independent training**, which trains each domain independently. Among them, **joint offline training** and **independent training** serve as the performance upper bound. In addition, since continual learning (CL) methods only apply to a small number of tasks, directly applying CL methods to our setting with large number of tasks (more than 40K) is infeasible. Instead, we combine several representative CL methods with meta learning base model. We modify and adapt **GSS** [5], **MIR** [3], **AGEM** [14] and **MER** [47] to our setting and combine them with meta learning base models to serve as strong baselines. We denote these baselines as PNet-GSS, ANIL-GSS, etc.

Proposed benchmark To simulate realistic imbalanced domain sequences, we construct a new benchmark and collect 6 domains with varying degree of similarity and difficulty, including **Quickdraw** [28], **AIRCRAFT** [39], **CUB** [61], **Miniimagenet** [56], **Omniglot** [34], **Necessities** from Logo-2K+ [59]. We resize all images into the same size of 84×84. All the methods are compared for 5-way 1-shot and 5-way 5-shot learning. All the datasets are publicly available with more details provided in Appendix A. We calculate the average accuracy on unseen testing tasks from all the domains for evaluation purpose.

Implementation details For ANIL-based [43] baselines, following [7], we use a four-layer CNN with 48 filters and one fully-connected layer as the meta learner. For PNet-based [51] baselines, we use a five-layer CNN with 64 filters of kernel size being 3 for meta learning. Following [51], we do not use any fully connected layers for PNet-based models. Similar architecture is commonly used in existing meta learning literature. We do not use any pre-trained network feature extractors which may contain prior knowledge on many pre-trained image classes, as this violates our problem setting that future domain knowledge is completely unknown. We perform experiments on different domain orderings, with the default ordering being Quickdraw, MiniImagenet, Omniglot, CUB, Aircraft and Necessities. To simulate imbalanced domains in streaming setting, each domain on this sequence is trained on 5000, 2000, 6000, 2000, 2000, 24000 steps respectively. In this setup, reservoir sampling will underrepresent most domains. All experiments are averaged over three independent runs. More implementation details are given in Appendix B. We made our code publicly available at `https://github.com/joey-wang123/Imbalancemeta.git`.

5.1. Comparison to Baselines

We compare our methods to the baselines. The memory maintains 300 batches (2) tasks. Results are shown in Table 1 and 2. We can observe that our method significantly outperforms baselines by a large margin of 5.21% for 5-shot learning and 4.95% for 1-shot learning with PNet-based model. For ANIL-based baselines, our method outperforms baselines by 4.60% for 5-shot learning and 2.19% for 1-shot learning. This shows the effectiveness of our method.

5.2. Effect of Memory Capacity

We explore the effect of memory capacity for the performance of baselines and our method. Table 3 and 4 show the results with memory capacity (batches) of 200, 300 and 500 respectively. Our method significantly outperforms all the baselines in each capacity case.

Table 1: Comparisons with PNet-based baselines

Algorithm	5-Way 1-Shot ACC	5-Way 5-Shots ACC
PNet-Sequential	31.82 ± 0.56	48.21 ± 0.50
PNet-RS	34.68 ± 1.96	53.69 ± 0.76
PNet-GSS	36.15 ± 1.59	55.16 ± 0.72
PNet-AGEM	34.07 ± 1.71	52.61 ± 0.68
PNet-MIR	34.53 ± 1.45	53.91 ± 0.56
PNet-MER	35.82 ± 1.69	54.28 ± 0.61
PNet-Ours	$\mathbf{41.10 \pm 0.42}$	$\mathbf{60.37 \pm 0.32}$
Joint-training	52.96 ± 0.45	68.56 ± 0.37
Independent-training	58.25 ± 0.36	72.23 ± 0.29

Table 2: Comparisons with ANIL-based baselines

Algorithm	5-Way 1-Shot ACC	5-Way 5-Shots ACC
ANIL-Sequential	30.68 ± 0.67	41.39 ± 0.37
ANIL-RS	32.11 ± 0.90	48.72 ± 0.79
ANIL-GSS	31.78 ± 1.08	48.93 ± 0.83
ANIL-AGEM	32.23 ± 1.21	48.56 ± 0.91
ANIL-MIR	31.85 ± 0.97	48.34 ± 0.72
ANIL-MER	32.72 ± 1.06	49.05 ± 0.96
ANIL-Ours	$\mathbf{34.91 \pm 0.73}$	$\mathbf{53.65 \pm 0.56}$
Joint-training	52.37 ± 0.72	66.21 ± 0.61
Independent-training	56.52 ± 0.57	69.67 ± 0.53

Table 3: Effect of memory size for PNet-based baselines

Algorithm	5-Way 1-Shot ACC	5-Way 5-Shots ACC
PNet-RS ($n = 200$)	34.12 ± 1.12	53.29 ± 0.42
PNet-Ours ($n = 200$)	40.11 ± 0.73	59.86 ± 0.27
PNet-RS ($n = 300$)	34.68 ± 1.96	53.69 ± 0.76
PNet-Ours ($n = 300$)	41.10 ± 0.42	60.37 ± 0.32
PNet-RS ($n = 500$)	35.67 ± 0.82	55.95 ± 0.79
PNet-Ours ($n = 500$)	41.82 ± 0.90	61.05 ± 0.60

Table 4: Effect of memory size for ANIL-based baselines

Algorithm	5-Way 1-Shot ACC	5-Way 5-Shots ACC
ANIL-RS ($n = 200$)	31.03 ± 0.97	45.96 ± 0.81
ANIL-Ours ($n = 200$)	32.83 ± 0.71	48.21 ± 0.61
ANIL-RS ($n = 300$)	32.11 ± 0.90	48.72 ± 0.79
ANIL-Ours ($n = 300$)	34.91 ± 0.73	53.65 ± 0.56
ANIL-RS ($n = 500$)	39.35 ± 0.76	53.86 ± 0.68
ANIL-Ours ($n = 500$)	42.79 ± 0.67	59.23 ± 0.49

5.3. Effect of Domain Ordering

We also compare to other two orderings: **Necessities, CUB, Omniglot, Aircraft, MiniImagenet, Quickdraw**; and **Omniglot, Aircraft, Necessities, CUB, Quickdraw, MiniImagenet**. The results are shown in Appendix D. In all cases, our method substantially outperforms the baselines.

5.4. Effect of Different Ratios of Domains

To explore how the different domain ratios affect the model performance, we did another set of experiments with iterations of 4K, 4K, 3K, 4K, 4K, 22K steps on each domain respectively. The results are shown in Table 8 in Appendix.

5.5. Effect of Domain Revisiting

To investigate the effect of domain revisiting on baselines and our method, we perform experiment on the domain sequence with domain revising of Quickdraw. The details and results are shown in Table 7 in Appendix. We currently assume that there is no domain-revising, properly handling domain-revisiting is left as interesting future work.

5.6. Ablation Study

Effect of memory management mechanism To verify the effectiveness of M2D3 proposed in section 3.3, Table 9 in Appendix shows the experiments with simple reservoir sampling without M2D3 (PNet-RS) and with M2D3 (PNet-Ours (without PETS)) respectively. Our method with M2D3 significantly outperforms baseline by 4.1% and 4.2% respectively. The memory proportion for each latent domain is shown in Figure 5 in Appendix. For RS baseline, the memory proportion for each domain is highly imbalanced. On the contrary, our memory management mechanism enables the memory proportion for each domain is relatively balanced, demonstrating the effectiveness of our method.

Effect of PETS To verify the effectiveness of PETS proposed in section 3.4, we compare the gradient variance with uniform sampling and our adaptive task sampling method, the gradient variance during training is shown in Figure 6 in Appendix. We can see that our adaptive task sampling achieves much less gradient variance especially when training for longer iterations. Table 9 in Appendix shows that with PETS, the performance is improved by more than 2.2% and 2.4% for 1-shot and 5-shot learning respectively.

6. Conclusion

This paper addresses the forgetting problem when meta learning on non-stationary and imbalanced task distributions. To address this problem, we propose a new memory management mechanism to balance the proportion of each domain in the memory buffer. Also, we introduce an efficient adaptive memory task sampling method to reduce the task gradient variance. Experiments demonstrate the effectiveness of our proposed methods. For future work, it would be interesting to meta learn the proportional of each domain automatically.

Acknowledgements This research was supported in part by NSF through grants IIS-1910492.

References

[1] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. *https://arxiv.org/abs/1511.06481*, 2016. 6

[2] 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. 7

[3] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *Advances in Neural Information Processing Systems*, 2019. 7

[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. 7

[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. 7

[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. 6

[7] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *International Conference on Learning Representations*, 2019. 6, 7

[8] Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. Defining benchmarks for continual few-shot learning. *https://arxiv.org/abs/2004.11967*, 2020. 7

[9] Sébastien Arnold, Pierre-Antoine Manzagol, Reza Babanezhad Harikandeh, Ioannis Mitliagkas, and Nicolas Le Roux. Reducing the variance in online optimization by transporting past gradients. *Advances in Neural Information Processing Systems*, 2019. 5

[10] I. Bae, J. Moon, J. Jhung, H. Suk, T. Kim, H. Park, J. Cha, J. Kim, D. Kim, and Shiho Kim. Self-driving like a human driver instead of a robocar: Personalized comfortable driving experience for autonomous vehicles. *NeurIPS 2019 Workshop: Machine Learning for Autonomous Driving*, 2020. 1

[11] 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. 7

[12] 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. 7

[13] Giuseppe Castellucci, Simone Filice, Danilo Croce, and Roberto Basili. Learning to solve NLP tasks in an incremental number of languages. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021. 1

[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

[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. 2

[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] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 7

[18] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. *The 2020 European Conference on Computer Vision (ECCV)*, 2020. 7

[19] H. Edwards and A. Storkey. Towards a neural statistician. *International Conference on Learning Representations*, 2017. 6

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, 2017. 6, 7

[21] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *Proceedings of International Conference on Machine Learning*, 2019. 6

[22] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *Advances in Neural Information Processing Systems*, 2018. 6

[23] 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. 2, 7

[24] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. 6

[25] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. 3

[26] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. *Advances in Neural Information Processing Systems*, 2020. 7

[27] 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. 6

[28] Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. The quick, draw! – a.i. experiment. 2016. 7, 12

[29] Angelos Katharopoulos and Francois Fleuret. Not all samples are created equal: Deep learning with importance sampling.

Proceedings of the 35th International Conference on Machine Learning. 6, 12

[30] N. Keriven, D. Garreau, and I. Poli. Newma: A new method for scalable model-free online change-point detection. *IEEE Transactions on Signal Processing*, 68:3515–3528, 2020. 4

[31] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. *Proceedings of the International Conference on Machine Learning*, 2019. 5

[32] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In *European Conference on Computer Vision (ECCV)*, 2020. 7

[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

[34] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. *Conference of the Cognitive Science Society*, 2011. 7, 12

[35] 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

[36] 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. 6

[37] Shuang Li, Yao Xie, Hanjun Dai, and Le Song. Scan *b*-statistic for kernel change-point detection. *Advances in Neural Information Processing Systems*, 2015. 3

[38] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 2017. 7

[39] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *https://arxiv.org/abs/1306.5151*, 2013. 7, 12

[40] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *International Conference on Learning Representations*, 2018. 6

[41] Tsendsuren Munkhdalai and Hong Yu. Meta networks. *Proceedings of the 34th International Conference on Machine Learning*, 2017. 6

[42] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. *Proceedings of the International Conference on Learning Representations*, 2018. 7

[43] 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. 7

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

[45] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 6

[46] 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. 2, 7

[47] 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. 7

[48] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. *Proceedings of the 34th International Conference on Machine Learning*, 2016. 6

[49] J. Schmidhuber. A neural network that embeds its own meta-levels. *IEEE International Conference on Neural Networks*, 1993. 6

[50] Robert J. Serfling. *Approximation theorems of mathematical statistics*. Wiley series in probability and mathematical statistics : Probability and mathematical statistics. Wiley, 1980. 4

[51] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, 2017. 6, 7

[52] Pavel Tokmakov, Yu-Xiong Wang, and Martial Hebert. Learning compositional representations for few-shot recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 6

[53] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. *Proceedings of the International Conference on Learning Representations*, 2020. 6

[54] 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. 6

[55] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998. 4

[56] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 2016. 6, 7, 12

[57] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 1985. 2, 7

[58] 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. 6, 12

[59] 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. 7, 12

[60] Zhenyi Wang, Yang Zhao, Ping Yu, Ruiyi Zhang, and Changyou Chen. Bayesian meta sampling for fast uncertainty adaptation. *International Conference on Learning Representations*, 2020. 6

[61] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. 2010. 7, 12

[62] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *International Conference on Learning Representations*, 2018. 7

[63] 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. 2, 7

[64] Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. Variational few-shot learning. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 6