IMPROVING META-CONTINUAL LEARNING REPRESENTATIONS WITH REPRESENTATION REPLAY

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning often suffers from catastrophic forgetting. Recently, meta-continual learning algorithms use meta-learning to learn how to continually learn. A recent state-of-the-art is online aware meta-learning (OML) Javed & White (2019). This can be further improved by incorporating experience replay (ER) into its meta-testing. However, the use of ER only in meta-testing but not in meta-training suggests that the model may not be optimally meta-trained. In this paper, we remove this inconsistency in the use of ER and improve continual learning representations by integrating ER also into meta-training. We propose to store the samples' representations, instead of the samples themselves, into the replay buffer. This ensures the batch nature of ER does not conflict with the online-aware nature of OML. Moreover, we introduce a meta-learned Predictive Sample Selection to replace the widely used reservoir sampling to populate the replay buffer. This allows the most significant samples to be stored, rather than relying on randomness. Experimental results on a number of real-world meta-continual learning benchmark data sets demonstrate that the proposed method outperforms the state-of-the-art. Moreover, the learned representations have better clustering structures and are more discriminative.

1 INTRODUCTION

Humans can acquire knowledge incrementally throughout their lives. In continual learning (De Lange et al., 2019; Parisi et al., 2019), the machine learning model attempts to imitate this behavior by learning a number of tasks sequentially. This allows for lifelong learning, which acquires knowledge tirelessly and improves continuously over time. It also has diverse real-world applications, from recommendation systems (Yuan et al., 2020), clinical applications (Lee & Lee, 2020) to natural language processing (Biesialska et al., 2020). However, while machine learning models can sometimes surpass humans on specialized tasks, they often suffer from *catastrophic forgetting* (Goodfellow et al., 2015; Kirkpatrick et al., 2017). When new tasks are learned, the old knowledge learned from previous tasks is overwritten, leading to poor performance on these old tasks.

To alleviate this problem, meta-learning (Hospedales et al., 2020) has been recently incorporated into continual learning, leading to meta-continual learning (Caccia et al., 2020). By optimizing models for learning to learn as in meta-learning, meta-continual learned models are meta-trained to avoid catastrophic forgetting. (Riemer et al., 2019) show that the meta-learning objective aligns with the continual learning goals of minimizing interference and maximizing transfer.

A recent state-of-the-art in meta-continual learning is online aware meta-learning (OML) (Javed & White, 2019), which optimizes the model for continual learning through online sequential updates in meta-training. Javed & White (2019) show that the performance of OML can be further improved by incorporating experience replay (ER) (Chaudhry et al., 2019), which is an effective rehearsal strategy for continual learning by replaying tiny episodic memories during the training process, into its meta-testing. However, the use of ER only in meta-testing but not in meta-training suggests that the OML model may not be optimally meta-trained. Moreover, as the effectiveness of ER heavily relies on the samples stored for replay, the use of reservoir sampling (Vitter, 1985) to populate the replay buffer heavily relies on randomness.

In this paper, we improve the continual learning representations and remove inconsistency in the use of ER between meta-training and meta-testing by integrating ER also into meta-training. However, the batch nature of ER training interferes with the online nature of OML during meta-training, resulting in a loss of gradient and Hessian information during the meta-updates. To alleviate this problem, we propose to store the samples' representations, instead of the samples themselves, into the replay buffer. This allows both OML's online training and ER's batch replay to be preserved. Besides, instead of relying on reservoir sampling to populate the replay buffer, we use the meta-learning procedure to select samples that can best help overcome catastrophic forgetting. Experimental results on a number of real-world benchmark data sets demonstrate that the proposed method outperforms the state-of-the-art.

2 Related Work

2.1 Continual Learning

Continual learning aims to learn a series of tasks incrementally (Mai et al., 2021). As data from only the current task is available during training, it is important to overcome catastrophic forgetting. There are three continual learning settings of increasing difficulty (van de Ven & Tolias, 2019): task-incremental, domain-incremental, and class-incremental. In this paper, we focus on the class-incremental setting, which is the most difficult. At test time, the model has to both decide the task identity and solve all the tasks encountered so far. We also use the online (or streaming) setting (Aljundi et al., 2019b; Hayes et al., 2019; Borsos et al., 2020) in which each sample only appears once during training.

A common approach to avoid catastrophic forgetting is by regularization (Zenke et al., 2017; Huszar, 2018; Li & Hoiem, 2018). However, Lesort et al. (2019) show both theoretically and empirically that regularization is insufficient. Knoblauch et al. (2020) also demonstrate that some form of memory or replay buffer, as used in rehearsal methods (Lopez-Paz & Ranzato, 2017; Aljundi et al., 2019a; van de Ven et al., 2020), is necessary.

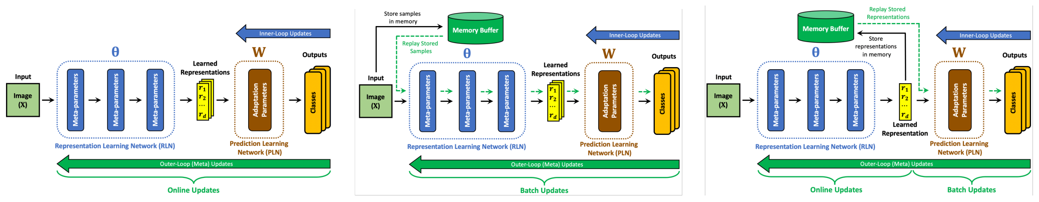
2.2 Meta-Learning

Meta-learning, also known as "learning to learn", aims to enable models to learn more effectively (Hospedales et al., 2020). A well-known example algorithm is model-agnostic meta-learning (MAML) (Finn et al., 2017), which meta-learns a model initialization (meta-parameter θ) that can be quickly adapted to a new task by training on only a few samples. During the training phase of meta-training (*meta-training training*), W_0, a copy of θ, learns from a batch of tasks in the meta-training training set and is updated to W_t after t iterations. During the testing phase of meta-training (*meta-training testing*), θ is updated by back-propagating the losses of each of $W_1, \dots, W_t$ on the meta-training test set. In meta-testing, the *meta-testing training* phase adapts the meta-trained model using the same procedure as meta-training training on each of the unseen tasks. Finally, each adapted model is evaluated on the corresponding task in the meta-testing test set during *meta-testing testing*.

2.3 Meta-Continual Learning

Meta-continual learning meta-learns a model for effective continual learning (i.e., "learning to learn not to forget" (Caccia et al., 2020)). The procedure follows closely that of meta-learning, except that continual learning is performed instead of batch learning. In other words, in each meta-training training iteration, a sequence (instead of a batch) of tasks is sampled, and W_0 is updated on each of the tasks sequentially (thus simulating continual learning).

The state-of-the-art is Online aware Meta-Learning (OML) (Javed & White, 2019). Its model $f_{\theta,W}$ has two components: (i) a representation-learning network (RLN) f_θ with parameter θ, which meta-learns a good representation for continual learning, and (ii) a prediction learning network (PLN) f_W with parameter W, that performs prediction using RLN's representation. Algorithm 1 shows its meta-training procedure (Figure 1(a)). A task sequence T_i is sampled at outer iteration i (step 3). A training set S_{train} (of size k) and a test set S_{test} is constructed from these tasks (step 4). The inner loop (*meta-training training*) online updates the PLN using the loss ℓ on each sample of S_{train} (step 8), as in the online class-incremental continual learning setting (van de Ven & Tolias, 2019). Note that online update, instead of batch update, is key to OML as it simulates continual learning and allows the effect of catastrophic forgetting be taken into account. The RLN is not explicitly updated



(a) OML. (b) Naive approach. (c) Representation replay.

Figure 1: OML and two ways to integrate with experience replay into its meta-training.

during meta-training training. However, it is implicitly updated from $\theta_1 = \theta$ to $\theta_2, \ldots, \theta_{k+1}$ so that the meta-loss gradient (to be computed in meta-training testing) can back-propagate through all k intermediate models and allow the RLN to improve its representation based on each S_{train} sample. Step 10 (*meta-training testing*) updates the RLN by minimizing the meta-loss[1] $\tilde{L}(\theta_{k+1}, W_{k+1}) = \ell(f_{\theta_{k+1},W_{k+1}}(S_{test}[:,0]), S_{test}[:,1])$ with gradient descent (using learning rate β):

$$\theta \leftarrow \theta - \beta \nabla_\theta \tilde{L}(\theta_{k+1}, W_{k+1}). \tag{1}$$

On meta-testing, a sequence T of tasks that have not been seen during meta-training is used. The *meta-testing training* phase uses the training set in T, with the same procedure as in meta-training training (online updating the PLN by each sample in T once). The model is then evaluated in *meta-testing testing* using the test set of T.

Javed & White (2019) demonstrate that OML can also be used in conjunction with experience replay (ER) (Chaudhry et al., 2019) during meta-testing. A replay buffer stores previous tasks' inputs to approximate the previous tasks' distributions. A random batch from the buffer is interleaved with a new sample at each meta-testing training iteration.

Another state-of-the-art is ANML (Beaulieu et al., 2020), which is similar to OML but meta-learns a network that selective gates the PLN outputs. ANML is shown to outperform OML in (Beaulieu et al., 2020). However, Javed & White (2019) later showed that the two have comparable performance after fixing a bug in the meta-gradient computation of OML.[2]

3 Proposed Method

3.1 Using Experience Replay in Meta-Training

In OML, inner updates to the PLN are performed in an online manner so as to simulate continual learning. However, these updates perform simple SGD and do not take explicit measures to combat catastrophic forgetting. Moreover, while Javed & White (2019) demonstrate that using experience replay (ER) in meta-testing alone improves OML's meta-testing testing accuracies, the lack of ER in meta-training creates inconsistency. More extensive experiments in Section 4.1 also shows that it is indeed worse than standard OML on more challenging data sets. To alleviate these problems, we propose integrating ER also into the meta-training phase of OML.

3.1.1 Naive Approach

In this section, we first present a straightforward integration. At inner iteration j of Algorithm 1, the new meta-training sample X_j is simply combined with $b-1$ random samples from the replay buffer (Figure 1(b)). The resultant batch B_j goes through both the RLN and PLN as in standard OML. The PLN is updated by SGD as: $W_{j+1} = W_j - \alpha \nabla_{W_j} \ell(f_{\theta,W_j}(B_j[:,0]), B_j[:,1])$, where $f_{\theta,W_j}(B_j[:,0])$ is the prediction on this batch, and $B_j[:,1]$ is the target output.

By expanding the meta-loss gradient $\nabla_\theta \tilde{L}(\theta_{k+1}, W_{k+1})$ in (1) to leading order as in (Nichol et al., 2018), we obtain the following Proposition. All the proofs are in the appendix.

[1] Here, the inputs of all samples in S is denoted $S[:,0]$, and the corresponding labels denoted $S[:,1]$.
[2] https://github.com/khurramjaved96/mrcl#05-july-2020-major-bug-fix-and-refactoring-log

Proposition 1. *$\nabla_\theta \tilde{L}(\theta_{k+1}, W_{k+1}) = \left[\tilde{g} - \alpha \sum_{j=1}^{k} \tilde{H}'(B_j)\tilde{g} - \alpha \tilde{H} \sum_{j=1}^{k} \tilde{g}'(B_j),\ 0\right] + O(\alpha^2)$, where $\tilde{g} \equiv \nabla_\theta \tilde{L}(\theta, W_1)$ and $\tilde{H} \equiv \nabla_\theta^2 \tilde{L}(\theta, W_1)$ are the gradient and Hessian of the meta-loss respectively, $\tilde{g}'(B_j) \equiv \nabla_\theta \ell(f_{\theta,W_j}(B_j[:, 0]), B_j[:, 1])$, and $\tilde{H}'(B_j) \equiv \nabla_\theta^2 \ell(f_{\theta,W_j}(B_j[:, 0]), B_j[:, 1])$.*

As $\tilde{g}'(B_j)$ and $\tilde{H}'(B_j)$ are computed from all samples in B_j, the loss's gradient and Hessian contributions from X_j may be canceled out by some replay samples in B_j. This loss of information can hinder the RLN from learning representations from all meta-training samples. The benefits of obtaining gradients for each sample to train the RLN, which is crucial in OML, are thus lost.

3.1.2 Representation Replay

To alleviate the problem in Section 3.1.1, the RLN needs to be online updated by only the current sample X_j (as in OML). Instead of storing the raw samples directly, we propose representation replay, which stores the latent representations of the input samples into the replay buffer (Figure 1(c)). Algorithm 2 shows the meta-training training procedure with representation replay (which replaces steps 6-9 of OML's Algorithm 1). At inner iteration j, X_j still passes through the RLN to obtain its representation $f_\theta(X_j)$. A batch of representations R_j is also sampled from the replay buffer. As R_j are not raw samples, they only need to go through the PLN f_{W_j}. Using the combined batch of representations ($\{f_{\theta,W_j}(X_j)\} \cup R_j$), the PLN is updated by SGD as $W_{j+1} = W_j - \alpha \nabla_{W_j}(\ell(f_{\theta,W_j}(X_j), Y_j) + \ell(f_{W_j}(R_j[:, 0]), R_j[:, 1]))$.

After k inner updates, the RLN is updated. As R_j does not depend on θ, only the gradient and Hessian for X_j need to be computed when meta-updating the RLN. The meta-loss gradient can be obtained by the following Proposition.

Proposition 2. *$\nabla_\theta \tilde{L}(\theta_{k+1}, W_{k+1}) = \left[\tilde{g} - \alpha \sum_{j=1}^{k} \tilde{H}'(X_j)\tilde{g} - \alpha \tilde{H} \sum_{j=1}^{k} \tilde{g}'(X_j), 0\right] + O(\alpha^2)$, where $\tilde{g}'(X_j) \equiv \nabla_\theta \ell(f_{\theta,W_j}(X_j), Y_j)$, $\tilde{H}'(X_j) \equiv \nabla_\theta^2 \ell(f_{\theta,W_j}(X_j), Y_j)$, and $\tilde{g}$ and $\tilde{H}$ are as in Proposition 1.*

Compared with Proposition 1, $\tilde{g}'(B_j)$ and $\tilde{H}'(B_j)$ are now replaced by $\tilde{g}'(X_j)$ and $\tilde{H}'(X_j)$, respectively. Thus, information on X_j will no longer be interfered by replay samples in the same batch. This allows the RLN to learn representations for each individual sample while minimizing interference to the previously learned tasks. Note that the proposed method can also be used with other meta-continual representation learning methods, such as ANML (Beaulieu et al., 2020) and La-MAML (Gupta et al., 2020).

A related technique is latent replay (Pellegrini et al., 2019). It tries to reduce the time and space required for rehearsal by storing the activations at some intermediate layer (instead of the samples themselves), so that the forward and backward passes only need to go through the model's top layers but not the lower levels. Though latent replay also stores the representations, it differs fundamentally from the proposed representation replay in the following aspects: (i) Latent replay aims to reduce time and space in continual learning, while representation replay aims at improving the meta-continual learning representations; (ii) In latent replay, one has to tune and decide which particular intermediate layer for storing the representations, while representation replay always stores the outputs of the RLN; (iii) Latent replay performs slow training on the lower layers. With a long training process, the stored feature representations may become unsuitable for current and future training. For representation replay, the RLN is not updated in both meta-training training and meta-testing training, and thus does not suffer from the aforementioned problem.

3.2 Selecting Samples into the Replay Buffer

To select samples into the replay buffer, reservoir sampling (Vitter, 1985) has been commonly used (Robins, 1995; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019). However, reservoir sampling uses random selection which does not reflect how each sample may benefit continual learning. Another simple approach is to implement the replay buffer as a ring buffer and store the last M/T samples of each task (Chaudhry et al., 2021), where M is the buffer size and T is the total number of tasks. However, these last few samples may not be the most representative.

Recently, Aljundi et al. (2019b) propose a greedy sample selection scheme, which encourages diversity in the replay buffer by choosing samples whose gradients are least similar to all gradients of a random batch from the buffer. However, this may be problematic when the task has outlying samples. More recently, Borsos et al. (2020) formulate sample selection as a bilevel optimization problem. However, even on using greedy forward selection, it is still very computationally expensive.

In this section, we propose a novel scheme, called Predictive Sample Selection (PSS), for selecting samples into the replay buffer based on how they affect the loss. When the buffer is full, the buffered sample that leads to the smallest loss increase is replaced. For computational efficiency, the effect on loss is estimated by a simple FC layer. Note that sample selection only takes place during meta-training training and meta-testing training.

The procedure is shown in Algorithm 3. First, the current batch of representations R (containing the replay buffer samples and a new sample $(f_\theta(X_j), Y_j)$) goes through a forward pass of the FC layer. For each sample $(f_\theta(X_r), Y_r) \in R$, the FC outputs a score $FC(f_\theta(X_r))$ which is a prediction of the difference $\ell_{R\backslash\{(f_\theta(X_r),Y_r)\}} - \ell_R$. Here, ℓ_R and $\ell_{R\backslash\{(f_\theta(X_r),Y_r)\}}$ are the model's (PLN's) losses on class Y_r when trained on R with and without $(f_\theta(X_r), Y_r)$ respectively. The loss is measured on all meta-training samples belonging to class Y_r, as (X_r, Y_r) is mainly used to prevent forgetting on Y_r. It thus takes into account of how this sample prevents catastrophic forgetting of its class if it resides in the buffer. Step 3 then selects sample $(f_\theta(\hat{X}), \hat{Y})$ with the lowest score. If this comes from the replay buffer, it will be replaced by the new sample $(f_\theta(X_j), Y_j)$ when the replay buffer is full.

The FC layer is trained during meta-training training, by minimizing the square loss between $FC(f_\theta(\hat{X}))$ and the actual loss increase $\ell_{R\backslash\{(f_\theta(X_r),Y_r)\}} - \ell_R$. To evaluate $\ell_{R\backslash\{(f_\theta(X_r),Y_r)\}}$ (step 5), we need to first update the PLN on $R\backslash\{(f_\theta(X_r), Y_r)\}$. This can be done very efficiently by performing only one SGD step on W_j (step 4). During meta-testing training, the FC layer is not updated, and only steps 1 and 2 in Algorithm 3 need to be performed.

Algorithm 1 Meta-training in OML (Javed & White, 2019).

```
Require: $p(T)$: distribution over task sequences;
 1: randomly initialize $\theta$
 2: while not done do
 3:     sample task sequence $T_i \sim p(T)$
 4:     obtain $S_{train}$, $S_{test}$ from $T_i$
 5:     randomly initialize $W_1$
        // meta-training training
        // (updating PLN on each sample j)
 6:     for $j = 1, 2, \ldots, k$ do
 7:         $(X_j, Y_j) = S_{train}[j]$
 8:         $W_{j+1} = W_j - \alpha\nabla_{W_j}\ell(f_{\theta,W_j}(X_j), Y_j)$
 9:     end for
        // meta-training testing
        // (updating RLN)
10:     $\theta = \theta - \beta\nabla_\theta\tilde{L}(\theta_{k+1}, W_{k+1})$
11: end while
```

Algorithm 2 Representation replay.

```
Require: $M$: the replay buffer
 1: function REP_REPLAY($\theta$, $W_1$, $M$)
 2:     for $j = 1, 2, \ldots, k$ do
 3:         $(X_j, Y_j) = S_{train}[j]$
 4:         sample batch $R$ from $M$
 5:         $R \leftarrow R \cup \{(f_\theta(X_j), Y_j)\}$
        // update PLN
 6:         $W_{j+1} = W_j - \alpha\nabla_{W_j}\ell(f_{W_j}(R[:, 0]), R[:, 1])$
 7:         $(f_\theta(\hat{X}), \hat{Y}) \leftarrow$ PRED_SAMPLE_SEL($\theta, W_j, W_{j+1}, R$)
        // update replay buffer
 8:         $M \leftarrow M \cup \{(f_\theta(X_j), Y_j)\}$
 9:         if $|M| > max$ then
10:             $M \leftarrow M\backslash\{(f_\theta(\hat{X}), \hat{Y})\}$
11:         end if
12:     end for
13:     return $W_{k+1}$, $M$
14: end function
```

4 Experiments

A summary of their meta-training and meta-testing data sets (which have disjoint sets of classes) is shown in Table 1. The experimental setup follows (Javed & White, 2019). The meta-training training set contains task sequences. Each sequence S_{train} randomly samples 3 tasks (classes) from the meta-training data set, and each task has 5 randomly sampled images for each class. The meta-training test set S_{test} randomly samples another 5 images for each of the same 3 classes in S_{train}, as well as a random batch of images from previously seen classes (*Omniglot* uses 15 random

Algorithm 3 Predictive Sample Selection (PSS).

```
Require: $FC$: the FC layer for the Predictive Sample Selection (PSS)
1: function PRED_SAMPLE_SEL($\theta, W_j, W_{j+1}, R$)
2:     $scores = f_{FC}(R)$
3:     $(f_\theta(\hat{X}), \hat{Y}) = R[\arg\min(scores)]$
4:     $W'_{j+1} = W_j - \alpha \nabla_{W_j} \ell(f_{W_j}(R \backslash \{(f_\theta(\hat{X}), \hat{Y})\}))$
5:     compute $\ell_R$ using $W_{j+1}$ and $\ell_{R \backslash \{(f_\theta(\hat{X}), \hat{Y})\}}$ using $W'_{j+1}$
6:     update $FC$ based on actual loss increase $\ell_{R \backslash \{(f_\theta(\hat{X}), \hat{Y})\}} - \ell_R$
7:     return $(f_\theta(\hat{X}), \hat{Y})$
8: end function
```

images, while *mini-ImageNet* and *CIFAR-FS* use 10 random images). The meta-testing training set is a task sequence containing all classes in the meta-testing set, while the meta-testing testing set is a set of unseen samples for all classes in the meta-testing set. Each task (class) randomly samples training and test images from the meta-testing set (15 training and 5 test images for *split-Omniglot*; 30 training and 100 test images for *mini-ImageNet* and *CIFAR-FS*). Further details on the setup can be found in Appendix C.

		#classes	#samples/class
split-Omniglot	meta-train	963	20
	meta-test	600	20
mini-ImageNet	meta-train	64	600
	meta-test	20	600
CIFAR-FS	meta-train	64	600
	meta-test	20	600

Table 1: Summary of the meta-training and meta-testing sets for the data sets used.

	meta-training	meta-testing
ANML	-	-
OML	-	-
OML(ER)	-	ER
OML(REP)	-	REP
OMER	ER	ER
OMREP	REP	REP
OMREP(no-REP)	REP	-

Table 2: Use of replay in each method.

We will experiment with the state-of-the-arts of **OML**[3] and **ANML** (Beaulieu et al., 2020), and the following OML variants.
(i) **OML(ER)**, which combines OML with ER (and reservoir sampling) as in (Javed & White, 2019). Meta-training follows that of standard OML, but ER is used during meta-testing training;
(ii) **OML(REP)**: Meta-training follows OML, but representation replay (REP) is used during meta-testing training;
(iii) **Online aware Meta-Experience Replay (OMER)**: The naive approach for incorporating ER into OML as discussed in Section 3.1.1;
(iv) **Online aware Meta-Representation Replay (OMREP)**: The proposed method (Section 3.1.2);
(v) **OMREP(no-REP)**, an OMREP variant which does not use REP in meta-testing training.
The use of replay in each of these methods is shown in Table 2. Performance is measured by the accuracy on the meta-testing sets from all previously learned tasks.

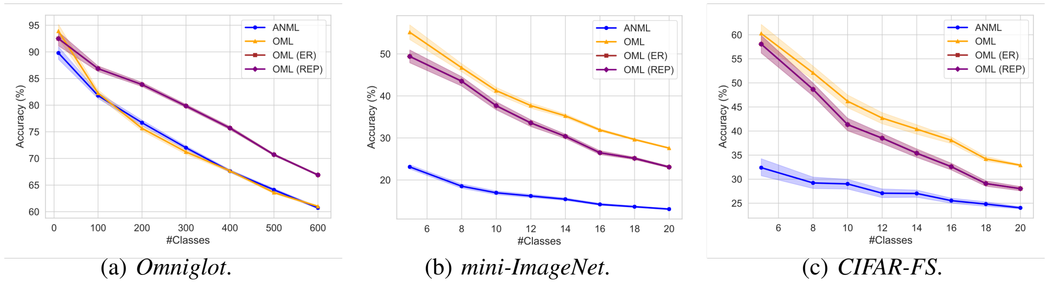
4.1 Using Experience Replay in Meta-Testing Only

We first demonstrate that using experience replay only in meta-testing, as in (Javed & White, 2019), may not be useful. Experiments are performed on **ANML**, **OML** and its variants **OML(ER)** and **OML(REP)**. In OML(ER)/OML(REP), the replay buffer is initially empty and populated by reservoir sampling. Its size is set to 20% of the meta-testing training set (i.e., 1,800 for *Split-Omniglot*, and 120 for *mini-ImageNet* and *CIFAR-FS*). Similar experiments have been performed in (Javed & White, 2019). However, here, we use longer task sequences to further explore the effects of catastrophic forgetting, and use data sets with varied complexities.

Figure 2(a) shows the meta-testing testing accuracies with the number of classes on *Split-Omniglot*. As can be seen, the testing accuracies of ANML and OML are very similar. The curves for OML(ER)

[3]https://github.com/khurramjaved96/mrcl. We used the version after bug fix for correct meta-gradients.

and OML(REP) are exactly the same, as the RLN is not updated during meta-testing. The only difference between them is that the stored representations in OML(REP) do not need to go through the RLN during meta-testing training. Hence, OML(REP) is more efficient than OML(ER) and takes less memory.[4] Both methods significantly outperform ANML and OML.



(a) *Omniglot*. (b) *mini-ImageNet*. (c) *CIFAR-FS*.

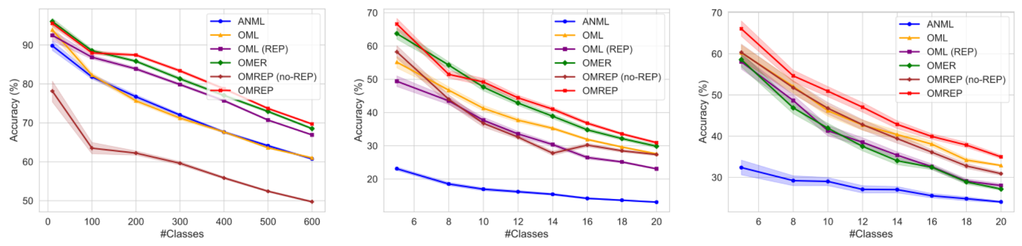
Figure 2: Meta-testing testing accuracies of ANML, OML, OML(ER), OML(REP). Note that the curves for OML(ER) and OML(REP) completely overlap with each other.

As for *mini-ImageNet* and *CIFAR-FS*, they are more difficult than *split-Omniglot*, and the observations in Figures 2(b) and 2(c) are different. First, ANML fails to learn the meta-testing classes. Second, while OML(ER)/OML(REP) has good testing accuracies on *Split-Omniglot* (as also reported in (Javed & White, 2019)), it is now outperformed by OML. This can be attributed to the increased effects of the stability-plasticity tradeoff (Parisi et al., 2019). Though revisiting previous tasks can help maintain the accuracies for learned tasks, this hinders the model's ability to learn new tasks.

4.2 Using Experience Replay in Both Meta-Training and Meta-Testing

In this experiment, experience replay is also used in meta-training. Since OML(ER) is identical to OML(REP) in terms of accuracies, we compare **ANML**, **OML** and **OML(REP)** with the OML variants of **OMER**, **OMREP(no-REP)** and **OMREP**. The setup is the same as in Section 4.1.

Figure 3 shows the meta-testing testing accuracies. OMREP has the best accuracies on all data sets. Thus, by making the meta-training training procedure consistent with that of meta-testing training, OMREP can learn the effects of ER on continual learning during meta-training. On the other hand, OMER (which uses ER in meta-training and meta-testing) does not have good accuracies overall. As discussed in Section 3.1.1, information from some meta-training samples may have been canceled out by replay samples. Thus, the RLN fails to learn good meta-continual learning representations.



(a) *Omniglot*. (b) *mini-ImageNet*. (c) *CIFAR-FS*.

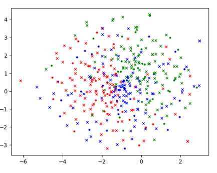
Figure 3: Meta-testing testing accuracies of ANML, OML, OML(REP), OMER, OMREP(no-REP), and OMREP.

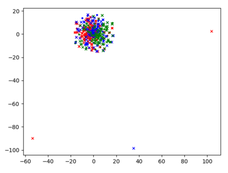
[4] 5.3GB for OML(REP) vs 6.4GB for OML(ER) in meta-training, and 2.4GB for OML(REP) vs 7.1GB for OML(ER) in meta-testing.

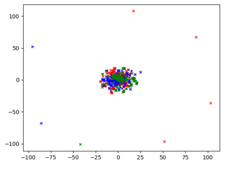
4.3 Better Meta-Continual Learning Representation

In this section, we demonstrate that the RLN representation learned by OMREP has a better clustering structure and is more discriminative. Since the meta-trained RLNs for OML(ER) and OML(REP) are the same as that of OML, we only compare ANML, OML, OMER and OMREP.[5]

Figure 4 shows the visualizations by t-SNE (Maaten & Hinton, 2008) on three random classes from the meta-testing set of *CIFAR-FS*. Visualizations for the meta-training set and other data sets are shown in Figures 8 and 7, respectively, in Appendix D. As can be seen, for ANML, OML, and OMER, their RLN output representations for different classes are mixed together. This indicates the failure in capturing representations useful for classification. Overall, the clustering structure from the OMREP's RLN output representations is the best, with most samples of the same class close to each other. Table 3 quantitatively evaluates the clustering quality by the intra-class to inter-class variance ratio[6] (Goldblum et al., 2020). A lower ratio means better class separation. OMREP yields the best class separation on the more difficult data sets *mini-ImageNet* and *CIFAR-FS*. On *split-Omniglot*, OMREP is comparable with OMER and are better than the others.


(a) ANML.


(b) OML.


(c) OMER.

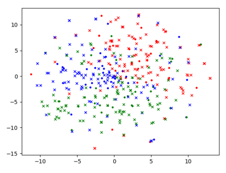

(d) OMREP.

Figure 4: T-SNE visualizations for the RLN output representations learned from the meta-testing sets of *CIFAR-FS*. Circles represent training samples, and crosses represent testing samples (best viewed in color).

	split-Omniglot	*mini-ImageNet*	*CIFAR-FS*
ANML	192.1	8.8	6.0
OML	186.8	8.7	4.4
OMER	**129.5**	5.2	4.2
OMREP	138.0	**4.8**	**3.4**

Table 3: Intra-class to inter-class variance ratios for the learned RLN output representations.

	split-Omniglot	*mini-ImageNet*	*CIFAR-FS*
ANML	70.6	12.8	25.2
OML	60.5	16.8	29.5
OMER	70.6	**32.9**	28.3
OMREP	**71.8**	32.3	**33.3**

Table 4: Meta-testing testing accuracies by a kNN classifier on the learned RLN output representations.

Following (Beaulieu et al., 2020), we use a k-nearest-neighbor (kNN) classifier of training samples in the meta-testing training task sequence for each class on the RLN representations learned by various methods. The kNN classifier is trained on the RLN representations from the meta-testing training set and is evaluated on the RLN representations from the meta-testing testing set. As can be seen from Table 4, OMREP outperforms all the other methods on *split-Omniglot* and *CIFAR-FS*, and has comparable accuracy as OMER on *mini-ImageNet*.

4.4 Sample Selection

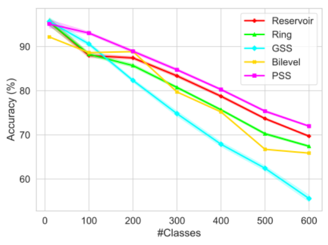
In this experiment, we compare the proposed Predictive Sample Selection (PSS) with the following ways to populate the replay buffer in OMREP: (i) reservoir sampling (Vitter, 1985); (ii) ring buffer (Chaudhry et al., 2021); (iii) greedy sample selection (GSS) (Aljundi et al., 2019b); and (iv) bilevel optimization (Borsos et al., 2020).

Figure 5 shows the meta-testing testing accuracies. As can be seen, PSS outperforms the others. Table 5 shows the time for meta-testing. Reservoir sampling and ring buffer are simple and so have

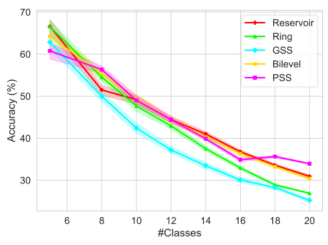
[5]We do not compare with OMREP(no-REP), as it has the same representations as OMREP, and only differ in that it does not use replay in meta-testing.

[6]The ratio is defined as $\frac{\sigma^2_{\text{within}}}{\sigma^2_{\text{between}}} = \frac{C}{N}\frac{\sum_{i,j}\|f_\theta(x_{i,j})-\mu_i\|^2}{\sum_i\|\mu_i-\mu\|^2}$, where $f_\theta(x_{i,j})$ is the feature representation of sample $x_{i,j}$ in class i, μ_i is the mean representation of class i, μ is the mean representation for the whole data set, C is the number of classes, and N is the number of samples per class.

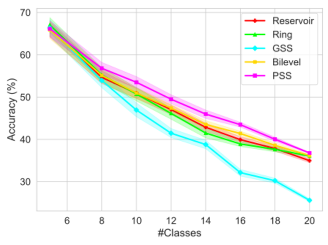
small time requirements. PSS only requires a forward pass of the trained FC layer and thus is also efficient. On the other hand, GSS requires computing the gradients for both the new sample and the replay buffer batches at each step and is much more expensive. Performing bilevel optimization at each step is, as expected, the most expensive.



(a) *Omniglot*.



(b) *mini-ImageNet*.



(c) *CIFAR-FS*.

Figure 5: Meta-testing testing accuracies for the different memory selection schemes in OMREP.

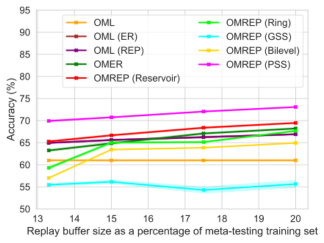
	Omniglot	*mini-ImageNet*	*CIFAR-FS*
reservoir	0.15	0.01	0.01
ring	0.15	0.01	0.01
GSS	7.5	0.5	0.5
bilevel	30.0	2.0	2.0
PSS	0.75	0.05	0.05

Table 5: Meta-testing time (in GPU hours) for different memory selection schemes.

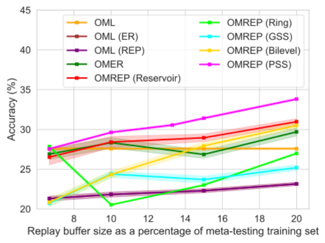
4.5 Effect of Replay Buffer Size

In previous experiments, the replay buffer size is set to 20% of the meta-testing training set. In this experiment, we vary the size from a minimum of 2 samples per class (i.e., 13.33% for *Omniglot* and 6.67% for *mini-ImageNet* and *CIFAR-FS*) to 20%. We consider the most difficult setting where the largest number of tasks (classes) has to be continually learned (i.e., 600 classes of *Omniglot*, and 20 classes of *mini-ImageNet* and *CIFAR-FS*).

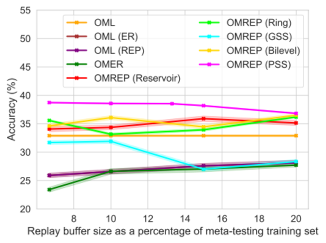
Figure 6 shows the meta-testing testing accuracies. For reference, we also show the accuracies of OML, which does not use a replay buffer. As can be seen, on all data sets, OMREP with PSS consistently outperforms all the baselines.



(a) *Omniglot*.



(b) *mini-ImageNet*.



(c) *CIFAR-FS*.

Figure 6: Meta-testing testing accuracies with different replay buffer sizes. Note that the curves for OML(ER) and OML(REP) completely overlap.

5 Conclusion

In this paper, we proposed representation replay, which improves the meta-continual learning representations while ensuring that the batch nature of ER does not conflict with the online-aware nature of meta-continual learning. We also proposed the Predictive Sample Selection, which selects the most sensitive samples into the replay buffer. Experimental results demonstrate that the proposed method outperforms the state-of-the-art, and the meta-continual learning representations obtained have better clustering structures and are more discriminative.

6 Ethics Statement

We have reviewed the ICLR Code of Ethics and have confirmed that this paper adheres to them. There are no privacy and security issues as all data sets used have been publicly released. The findings and research works do not have any negative social impacts or inappropriate applications.

7 Reproducibility Statement

For all theoretical results in this paper, the assumptions have been explained clearly and the complete proofs of the propositions are included in Appendix A and B. The methodologies used are detailed in Section 3.1. The data sets used and the detailed experimental setups are provided in Section 4 and Appendix C.

References

R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars. Online continual learning with maximally interfered retrieval. Technical report, arXiv:1908.04742, 2019a.

R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. In *NeurIPS*, pp. 11816–11825, 2019b.

S. Beaulieu, L. Frati, T. Miconi, J. Lehman, K. Stanley, J. Clune, and N. Cheney. Learning to continually learn. In *ECAI*, pp. 992–1001, 2020.

L. Bertinetto, J. Henriques, P. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019.

M. Biesialska, K. Biesialska, and M. Costa-jussà. Continual lifelong learning in natural language processing: A survey. Technical report, arXiv:2012.09823, 2020.

Z. Borsos, M. Mutny, and A. Krause. Coresets via bilevel optimization for continual learning and streaming. In *NeurIPS*, 2020.

M. Caccia, P. Rodríguez, O. Ostapenko, F. Normandin, M. Lin, L. Caccia, I. Laradji, I. Rish, A. Lacoste, D. Vázquez, and L. Charlin. Online fast adaptation and knowledge accumulation (OSAKA): A new approach to continual learning. In *NeurIPS*, 2020.

A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. Dokania, P. Torr, and M. Ranzato. On tiny episodic memories in continual learning. Technical report, arXiv:1902.10486, 2019.

A. Chaudhry, A. Gordo, P. Dokania, P. Torr, and D. Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *AAAI*, pp. 6993–7001. AAAI Press, 2021.

M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. Technical report, arXiv:1909.08383, 2019.

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

M. Goldblum, S. Reich, L. Fowl, R. Ni, V. Cherepanova, and T. Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *ICML*, pp. 3607–3616. PMLR, 2020.

I. Goodfellow, M. Mirza, X. Da, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. Technical report, arXiv:1312.6211, 2015.

G. Gupta, K. Yadav, and L. Paull. La-maml: Look-ahead meta learning for continual learning. Technical report, arXiv:2007.13904, 2020.

T. Hayes, N. Cahill, and C. Kanan. Memory efficient experience replay for streaming learning. In *ICRA*, pp. 9769–9776. IEEE, 2019.

T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. Technical report, arXiv:2004.05439, 2020.

F. Huszar. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 115(1):2496–2497, 2018.

K. Javed and M. White. Meta-learning representations for continual learning. In *NeurIPS*, pp. 1818–1828, 2019.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017.

J. Knoblauch, H. Husain, and T. Diethe. Optimal continual learning has perfect memory and is NP-HARD. In *ICML*, 2020.

C. Lee and A. Lee. Clinical applications of continual learning machine learning. *The Lancet Digital Health*, 2(6):e279–e281, 2020.

T. Lesort, A. Stoian, and D. Filliat. Regularization shortcomings for continual learning. Technical report, arXiv:1912.03049, 2019.

Z. Li and D. Hoiem. Learning without forgetting. *IEEE*, 40(12):2935–2947, 2018.

D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *NIPS*, pp. 6467–6476, 2017.

L. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal Machine Learning Research*, 9 (Nov):2579–2605, 2008.

Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner. Online continual learning in image classification: An empirical survey. Technical report, arXiv:2101.10423, 2021.

A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. Technical report, arXiv:1803.02999, 2018.

G. Parisi, R. Kemker, J. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni. Latent replay for real-time continual learning. Technical report, arXiv:1912.01100, 2019.

M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019.

A. Robins. Catastrophic forgetting, rehearsal, and pseudorehearsal. *Connection Science*, 7:123–146, 1995.

G. van de Ven and A. Tolias. Three scenarios for continual learning. Technical report, arXiv:1904.07734, 2019.

G. van de Ven, H. Siegelmann, and A. Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):1–14, 2020.

J. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1): 37–57, 1985.

F. Yuan, G. Zhang, A. Karatzoglou, X. He, J. Jose, B. Kong, and Y. Li. One person, one model, one world: Learning continual user representation without forgetting. Technical report, arXiv:2009.13724, 2020.

F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *ICML*, volume 70, pp. 3987–3995, 2017.

A Proof of Proposition 1

Proof. Let $U_j([\theta_j, W_j]) = [\theta_j, W_j] - \alpha\nabla_{[\theta_j,W_j]}\ell_j$, where $\ell_j \equiv \ell(f_{\theta,W_j}(B_j[:,0]), B_j[:,1])$. As in Nichol et al. (2018), we have

$$\begin{aligned}\nabla_\theta \tilde{L}(\theta_{k+1}, W_{k+1}) &= \nabla_\theta \tilde{L}(U_k(\cdots(U_1([\theta_1, W_1])))) \\ &= (\nabla_\theta(U_1([\theta_1, W_1]))\ldots\nabla_{\theta_k}(U_k([\theta_k, W_k])))(\nabla_{\theta_{k+1}}\tilde{L}(\theta_{k+1}, W_{k+1}))\end{aligned} \tag{2}$$

As $\nabla_{\theta_j}U_j([\theta_j, W_j]) = [I - \alpha\nabla^2_{\theta_j}\ell_j, -\alpha\nabla^2_{\theta_j,W_j}\ell_j]$, $\nabla_\theta\tilde{L}(\theta_{k+1}, W_{k+1})$ in (2) reduces to:

$$\prod_{j=1}^{k}\Big([I - \alpha\nabla^2_{\theta_j}\ell_j, -\alpha\nabla^2_{\theta_j,W_j}\ell_j]\Big)\,\nabla_{\theta_{k+1}}\tilde{L}(\theta_{k+1}, W_{k+1}).$$

By Taylor's theorem,

$$\begin{aligned}\nabla_{\theta_{k+1}}\tilde{L}(\theta_{k+1}, W_{k+1}) &= \nabla_\theta\tilde{L}(\theta, W_1) + \nabla^2_\theta\tilde{L}(\theta, W_1)(\theta_{k+1} - \theta_1) + O(\|\theta_{k+1} - \theta_1\|^2) \\ &= \tilde{g} + \tilde{H}(\theta_{k+1} - \theta_1) + O(\alpha^2) \\ &= \tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\nabla_{\theta_j}\ell_j + O(\alpha^2) \\ &= \tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}(\nabla_\theta\ell_j + O(\alpha)) + O(\alpha^2) \\ &= \tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\tilde{g}'(B_j) + O(\alpha^2).\end{aligned}$$

Expanding this to leading order as in Nichol et al. (2018),

$$\begin{aligned}\nabla_\theta\tilde{L}(\theta_k, W_{k+1}) &= \prod_{j=1}^{k}\Big([I - \alpha\tilde{H}'(B_j), -\alpha\nabla^2_{\theta_j,W_j}\ell_j]\Big)\left(\tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\tilde{g}'(B_j)\right) + O(\alpha^2) \\ &= \left(\left[I - \alpha\sum_{j=1}^{k}\tilde{H}'(B_j) + O(\alpha^2), O(\alpha^2)\right]\right)\left(\tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\tilde{g}'(B_j)\right) + O(\alpha^2) \\ &= \left[\tilde{g} - \alpha\sum_{j=1}^{k}\tilde{H}'(B_j)\tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\tilde{g}'(B_j), 0\right] + O(\alpha^2).\end{aligned}$$

□

B Proof of Proposition 2

Proof. Let $U_j([\theta_j, W_j]) = [\theta_j, W_j] - \alpha\nabla_{[\theta_j,W_j]}\ell_j$, where $\ell_j \equiv \ell(f_{\theta,W_j}(X_j), Y_j)$. Note that ℓ_j is now the loss on the new sample (X_j, Y_j) instead of a batch of samples B_j as in Appendix A. As in Nichol et al. (2018), we have

$$\begin{aligned}\nabla_\theta \tilde{L}(\theta_{k+1}, W_{k+1}) &= \nabla_\theta \tilde{L}(U_k(\cdots(U_1([\theta_1, W_1])))) \\ &= (\nabla_\theta(U_1([\theta_1, W_1]))\ldots\nabla_{\theta_k}(U_k([\theta_k, W_k])))(\nabla_{\theta_{k+1}}\tilde{L}(\theta_{k+1}, W_{k+1}))\end{aligned} \tag{3}$$

As $\nabla_{\theta_j}U_j([\theta_j, W_j]) = [I - \alpha\nabla^2_{\theta_j}\ell_j, -\alpha\nabla^2_{\theta_j,W_j}\ell_j]$, $\nabla_\theta\tilde{L}(\theta_{k+1}, W_{k+1})$ in (3) reduces to:

$$\prod_{j=1}^{k}\Big([I - \alpha\nabla^2_{\theta_j}\ell_j, -\alpha\nabla^2_{\theta_j,W_j}\ell_j]\Big)\,\nabla_{\theta_{k+1}}\tilde{L}(\theta_{k+1}, W_{k+1}).$$

By Taylor's theorem,

$$\begin{aligned}
\nabla_{\theta_{k+1}}\tilde{L}(\theta_{k+1}, W_{k+1}) &= \nabla_{\theta}\tilde{L}(\theta, W_1) + \nabla^2_{\theta}\tilde{L}(\theta, W_1)(\theta_{k+1} - \theta_1) + O(\|\theta_{k+1} - \theta_1\|^2) \\
&= \tilde{g} + \tilde{H}(\theta_{k+1} - \theta_1) + O(\alpha^2) \\
&= \tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\nabla_{\theta_j}\ell_j + O(\alpha^2) \\
&= \tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}(\nabla_{\theta}\ell_j + O(\alpha)) + O(\alpha^2) \\
&= \tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\tilde{g}'(X_j) + O(\alpha^2).
\end{aligned}$$

Expanding this to leading order as in Nichol et al. (2018),

$$\begin{aligned}
\nabla_{\theta}\tilde{L}(\theta_k, W_{k+1}) &= \prod_{j=1}^{k}\left([I - \alpha\tilde{H}'(X_j), -\alpha\nabla^2_{\theta_j, W_j}\ell_j]\right)\left(\tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\tilde{g}'(X_j)\right) + O(\alpha^2) \\
&= \left(\left[I - \alpha\sum_{j=1}^{k}\tilde{H}'(X_j) + O(\alpha^2), O(\alpha^2)\right]\right)\left(\tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\tilde{g}'(X_j)\right) + O(\alpha^2) \\
&= \left[\tilde{g} - \alpha\sum_{j=1}^{k}\tilde{H}'(X_j)\tilde{g} - \alpha\tilde{H}\sum_{j=1}^{k}\tilde{g}'(X_j), 0\right] + O(\alpha^2).
\end{aligned}$$

□

C Experimental Setup

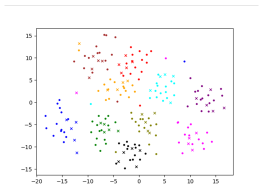
The model architecture used for *Omniglot* follows (Javed & White, 2019), and the architectures for *mini-ImageNet* and *CIFAR-FS* are the same as (Finn et al., 2017) and (Bertinetto et al., 2019) but with more filters and no batch normalization layers as in (Javed & White, 2019). The additional filters are also added by (Javed & White, 2019) because of the added difficulty for continual learning. The model architectures for *Omniglot*, *mini-ImageNet*, and *CIFAR-FS* are shown in Table 6.

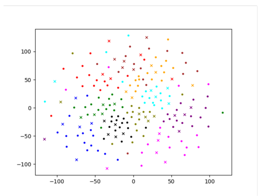
	Split-Omniglot	*mini-ImageNet*	*CIFAR-FS*
Input size	84×84	84×84	32×32
Number of convolutional layers	6	4	4
Number of filters	256	256	256
Kernel size	3×3	3×3	3×3
Stride	[2,1,2,1,2,2]	1	1
Padding	0	0	1
Maxpool kernel size	None	2×2	2×2
Maxpool stride	None	[2,2,2,1]	[2,2,2,1]
Maxpool padding	None	0	0
Number of fully-connected layers	1	1	1
Activation function	ReLU	ReLU	ReLU

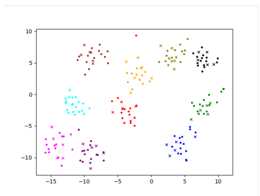
Table 6: Networks used for *Split-Omniglot*, *mini-ImageNet*, and *CIFAR-FS*.

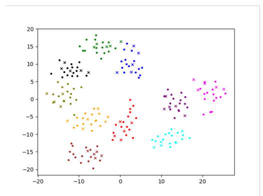
For methods using a replay buffer, the buffer size is set to 15 and the batch size b for replay is 10. The inner learning rate α is 0.03 and the meta-learning rate β is 10^{-4}. The models are meta-updated by the Adam optimizer and are trained for 200,000 epochs.

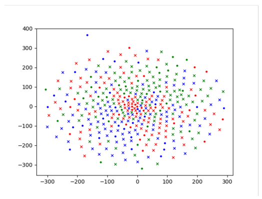
Following the meta-testing setup in (Javed & White, 2019), we perform five validation runs to choose the best learning rate to use from $\{0.03, 0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001\}$. Using the selected learning rate, the meta-testing phase is performed 50 times. The average is shown

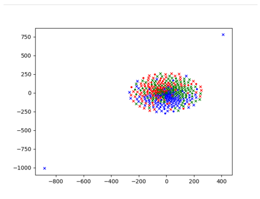

(a) ANML.

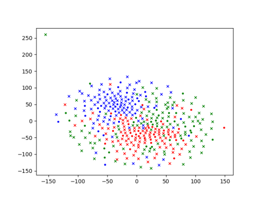

(b) OML.

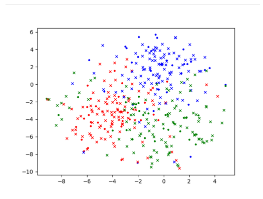

(c) OMER.

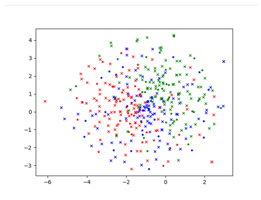

(d) OMREP.

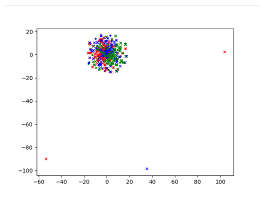

(e) ANML.

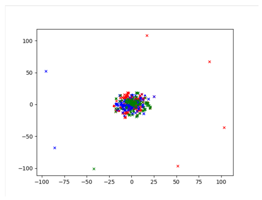

(f) OML.


(g) OMER.


(h) OMREP.


(i) ANML.


(j) OML.


(k) OMER.

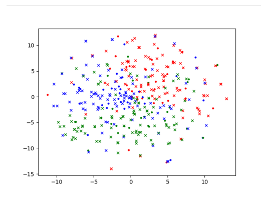

(l) OMREP.

Figure 7: T-SNE visualizations for the RLN output representations learned from the meta-testing sets of *Omniglot* (top), *mini-ImageNet* (middle), and *CIFAR-FS* (bottom). Circles are training samples, and crosses represent testing samples.

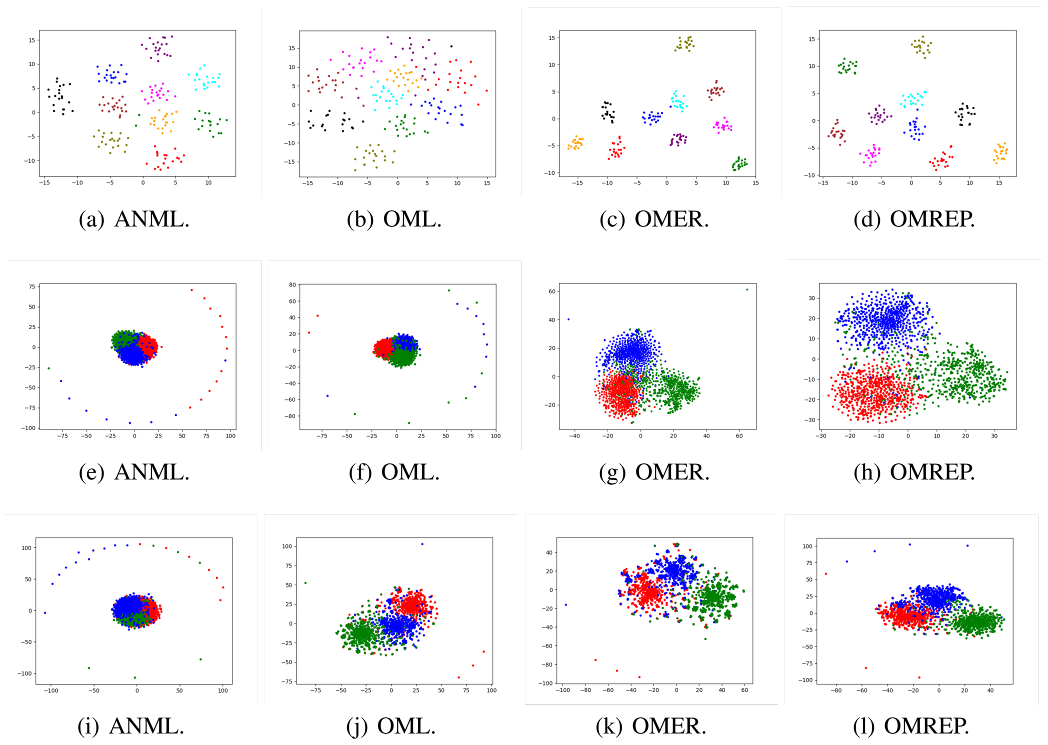
with a 95% confidence interval (drawn using 1,000 bootstraps). For methods using replay strategies, a batch size of 100 is used during meta-testing to adapt to a large number of tasks.

For ANML, the model architecture follows (Beaulieu et al., 2020) for all data sets (the input images are resized accordingly), but the hyperparameters values are the ones provided in the tables.

We use one NVIDIA GeForce RTX 2080Ti GPU to train the models. The meta-training phase takes about 3-4 days, while the meta-testing phase takes one day for *Split-Omniglot* and 2-3 hours for *mini-ImageNet* and *CIFAR-FS*.

D T-SNE Visualizations

Figure 7 (resp. 8) shows the t-SNE visualizations on the meta-testing (resp. meta-training) sets for ten random classes of *Omniglot*, and three random classes of *mini-ImageNet* and *CIFAR-FS*. From the visualizations, we can see that OMER and OMREP generally have better clustering structures, with OMREP having a farther separation between classes and fewer mixups between different classes for both the meta-training and meta-testing sets.



(a) ANML. (b) OML. (c) OMER. (d) OMREP.

(e) ANML. (f) OML. (g) OMER. (h) OMREP.

(i) ANML. (j) OML. (k) OMER. (l) OMREP.

Figure 8: T-SNE visualizations for the RLN output representations learned from the meta-training sets of *Omniglot* (top), *mini-ImageNet* (middle), and *CIFAR-FS* (bottom). Circles are training samples, and crosses represent testing samples.