COMBINING PARTITION-TREE WEIGHTING AND MAML FOR CONTINUAL AND ONLINE LEARNING

Anna Koop [*]
University of Alberta and Google DeepMind Canada
akoop@ualberta.ca

Michael Bradley Johanson
Canada
michael.johanson@gmail.com

Michael H. Bowling
University of Alberta
Canada
mbowling@ualberta.ca

ABSTRACT

Learning from experience requires adapting and responding to errors over time. However, gradient-based deep learning can fail dramatically in the continual, online setting. In this work, we address this shortcoming by combining two meta-learning methods: the purely online Partition Tree Weighting (PTW) mixture-of-experts algorithm, and a novel variant of the Model-Agnostic Meta-Learning (MAML) initialization-learning procedure. We demonstrate our approach, Replay-MAML PTW, in a piecewise stationary classification task in which the task distribution is unknown and the context changes are unobserved and random. We refer to this continual, online, task-agnostic setting as *experiential learning*. In this setting, Replay-MAML PTW matches and even outperforms an augmented learner that is allowed to train offline from the environment's task distribution and is given explicit notification when the environment context changes. Replay-MAML PTW thus provides a base learner with the benefits of offline training, access to the true task distribution, and direct observation of context-switches, but requires only a $O(\log T)$ increase in computation and memory.

1 INTRODUCTION

Supervised learning is the automated discovery of a mapping from inputs to desired outputs, based on explicitly labeled examples of input-output pairs. While classic supervised learning approaches tend to assume the existence of a large labeled dataset, the field of *continual learning* considers a more general case where the data is presented sequentially and may change in the input distribution or the correct output for a given input (Verwimp et al., 2023; Hadsell et al., 2020; Aljundi, 2019). One model for this continual learning setting is the introduction of a *task*, or fixed i.i.d. (independent and identically distributed) distribution of input-outputs, and the challenge is to learn as the current task continues to change. With this idea of a task, we also introduce *context* to be any information provided to the agent about the current task.

While continual learning embraces the possibility of the data changing during training time, it is still most common for evaluation to be *offline* (separate from the agent's sequence of training examples), and *task-dependent* (conditioned on specific task contexts) (Verwimp et al., 2023; Hadsell et al., 2020). In offline evaluation, training is periodically paused, possibly after completing training epochs or even training to convergence (Lange et al., 2022). Error is then measured on a held-out test-set for each task (Wang et al., 2024; van de Ven et al., 2022; Lange et al., 2022). This approach places emphasis on avoiding *catastrophic forgetting* (French, 1999; Aljundi, 2019) as the learner is being evaluated on all past tasks, not just the one generating the current data. An alternative is to focus on *online accuracy*, the learner's performance on the very sequence it is training on, which is the approach taken in this paper. This approach places emphasis on *fast remembering* (e.g. He et al., 2020; Caccia et al., 2020), *endless adaptation* (e.g. Abel et al., 2023), and *tracking* (e.g. Sutton et al., 2007).

Continual learning algorithms can be grouped according to the assumptions about how the task distribution changes and what context is available to the agent (Wang et al., 2024; Lange et al., 2022). At one extreme is *task-incremental learning*, where the tasks are explicitly known and identified in advance, although presented sequentially during training. At the opposite end — often associated with *online learning* — data is only available sequentially and the task

[*] anna.koop@gmail.com

may change arbitrarily with no context available to the learning agent(Wang et al., 2024; Shalev-Shwartz, 2012). In this setting, which we focus on in this work, the task distributions are not known in advance, and even changes in the task are not signaled. We call this setting *experiential learning* because the learner only has access to the data it experiences: both training and testing occur *online* through interaction with sequential data.

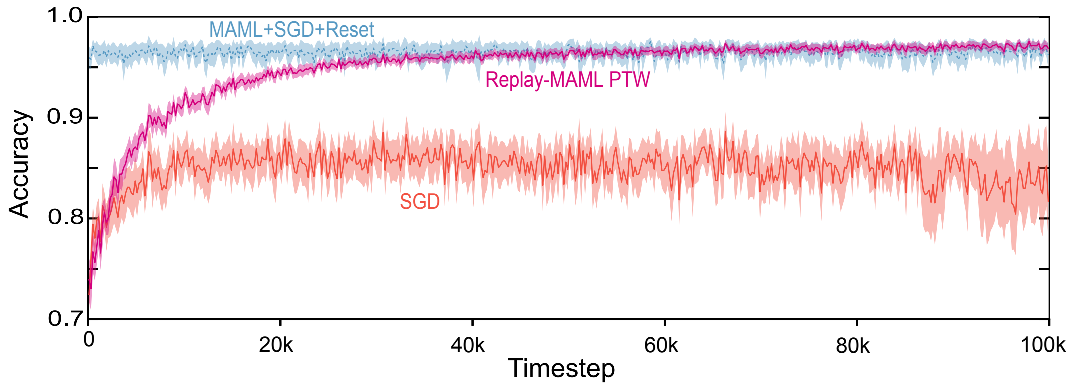


Figure 1: Classification accuracy in the Switching MNIST environment: online and continual Stochastic Gradient Descent (SGD); offline and non-continual Model-Agnostic Meta-Learning (MAML+SGD+Reset); and the online and continual Replay-MAML PTW. The data are quantized into intervals of 200 timesteps and averaged across 30 independent trials. Shaded regions indicate the 95% confidence interval.

We present a novel algorithm for this experiential setting. Our key result is shown in Figure 1 with accuracy shown on a continually-switching variation of MNIST digit classification. Our novel experiential learning algorithm Replay-MAML PTW (in pink) learns to perform as well as a learning algorithm MAML+SGD+Reset (in blue) that is given the advantage of pretraining on the task distribution and being explicitly informed of task changes. Replay-MAML PTW achieves this result by using a relatively small ensemble of models and combining two powerful algorithms: Partition Tree Weighting (PTW) for online adaptation in task-switching environments and a new online variant of Model-Agnostic Meta-Learning (MAML) for re-initializing models in the ensemble.

The rest of the paper will progressively introduce Replay-MAML PTW, building it up from its components. We will first introduce the Switching MNIST testbed that forms the basis for experimental comparison. We then build up from a simple stochastic gradient descent algorithm to introduce PTW, then MAML and approaches to combining them, before finally reaching Replay-MAML PTW. With each step we evaluate the components on our Switching MNIST testbed, providing both benchmark comparisons while also illustrating each component's role in the final algorithm.

2 Experimental Testbed for Continual Supervised Learning

We are focused on continual supervised learning problems. Example inputs, possibly in small batches, are presented to the learner from some distribution $p(\mathbf{x})$ and after the learner makes a prediction $\hat{y}$, it is shown the correct class from the distribution $p(y|\mathbf{x})$. In classical supervised learning, the data distribution $p(\mathbf{x}, y)$ — *i.e.*, the task — is independent and identically distributed (i.i.d.) and stationary throughout time. In this regime, the performance of the model is often measured with *offline testing*, where a subset of the data not used during training is used to evaluate model performance. In the continual learning setting, the observed data distribution $p(\mathbf{x})$ can change over time, and even the conditional probability $p(y|\mathbf{x})$ of the correct class can shift. The learning algorithm tries to predict the class of each batch of inputs before seeing the correct class, even as the task may change over time, creating a situation of *endless adaptation* (Abel et al., 2023). If the learning algorithm is given no additional context about the tasks or when they change, we call this an experiential learning problem.

We create an experimental testbed that embodies this experiential learning setting. We use a modification of the standard MNIST benchmark (Deng, 2012), which we call Switching MNIST. Switching MNIST is explicitly constructed to provide an unending stream of tasks from a finite data source that we know can be well-represented by standard learning techniques. It is defined as an experiential learning problem but allows for task-awareness and offline training for comparison between experiential and non-experiential learners.

2.1 SWITCHING MNIST

MNIST is a stationary k-way classification problem where images of digits from the MNIST dataset are assigned their correct class label: $k = 10$. Switching MNIST uses the MNIST dataset to create a non-stationary environment that stochastically changes tasks while an agent interacts with it. A *task* in Switching MNIST is a classification problem using a subset of the 10 digits sampled at uniform random without replacement, where each of the k classes contains n digits. For example, if $k = 3$ and $n = 2$, then one task may involve the agent learning to map images of 7 and 2 to class A, 9 and 6 to class B, and 1 and 5 to class C, and images of the other four digits are not observed. The input distribution $p(\mathbf{x})$ is uniform random over the $k \cdot n$ digits currently assigned to a class, with the label $p(\mathrm{y}|\mathbf{x})$ given by its assigned class.

On each timestep, the environment samples a batch of images from the current task, and the agent predicts their class labels. The agent's accuracy is computed and the batch's true labels are revealed for the agent to learn from. At the start of each timestep, the environment may stochastically switch to a new task with a fixed *switch probability*, without informing the agent. This creates a piecewise-stationary, task-agnostic sequence. The environment has periods of stability in which the learner experiences both repeated and novel data distributions. Furthermore, over sufficiently long learning trajectories each label will appear in all classes equally often. Thus *catastrophic label interference*, or the complete reversal of previously learned class mappings, is an unavoidable reality. There is no single comprehensive dataset over all tasks on which a trained learning agent is evaluated. Instead, the agent is evaluated on its online behavior.

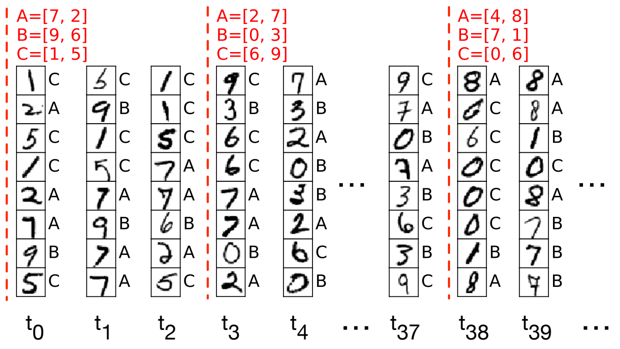


Figure 2: A sample sequence of experience from a Switching MNIST experience. Stochastic task switches are marked with a dashed red line. The current task is shown in red, though the agent sees only the image-class pairings and not the mappings from digit labels to classes. At each time-step, eight images are drawn from the current input distribution, then assigned a class according to the current mapping.

An example stream of experience from Switching MNIST is illustrated in Figure 2. The agent sees only the batch of image-class pairings, shown in black. The environment information is shown in red, annotating the task-specific class mapping, with the dashed line indicating the beginning of a new task. The agent sees a random batch of images from the digits assigned to classes in the current task, predicts the class of each image, and its accuracy is measured. The agent then updates its model given the correct labels. At the start of the next timestep, the environment may stochastically switch to a new task. In general this is a fully experiential setting: although the environment is piecewise stationary with discrete task changes, the learning task is online, evaluation is interleaved with learning, and no additional context is observed.

Switching MNIST satisfies our experiential criteria while allowing for comparison with non-experiential learners. Labels may appear in any class, so learning must be embedded in time: the input-output mapping is tied to the current task context. However, because the environment is piecewise-stationary and selects its tasks from a well-defined distribution, it is possible to train non-experiential learners offline, sampling according to the same task distribution. This also allows us to provide context information, such as when tasks change, to non-experiential learners for direct comparison to their experiential counterparts. Note that throughout this paper, we will use + in algorithm names to denote non-experiential counterparts to our basic learning algorithms: +Reset indicates the algorithm is notified when the environment changes context.

Evaluation is in terms of per-step average accuracy on the current batch, thus it is online and consistent with our *experiential evaluation* criteria. The random assignment of labels also allows us to turn the static MNIST dataset into a continual environment without being restricted by the size of the dataset. Using a simple classification setting, using two classes consisting of a single label, tasks are quite likely to repeat. In more complicated settings, tasks are unlikely to repeat at all. Although the size of the dataset ensures that inputs will recur, their context is changing. In the experimental results presented throughout this paper, we use the constants $k = 3$ and $n = 2$ with a batch size of 8 and a switching probability of 0.02 on each timestep (so that tasks switch every 50 timesteps on average) with experiments lasting 100,000 timesteps. This creates a relatively difficult learning problem with low probability of exact repeats between the 18,900 possible tasks to sample from. We expect an agent's accuracy to range from 33.3% (uniform random) to 98.6% (random on the first timestep of a new task and perfect thereafter).

3 Building an Experiential Learner

We present our algorithm by describing each of its component parts individually. In each case, we will show the performance of the component part on our Switching MNIST task. These components will act as baseline comparisons for our final algorithm, some of which will themselves be experiential learning algorithms, while others will require a context signal or use the task distribution for pretraining. This incremental approach will also serve to demonstrate the advantage that each component part serves in our final algorithm.

3.1 Stochastic Gradient Descent

The underlying model for all learning agents in this paper is a standard stochastic gradient descent (SGD) learner that uses a simple yet sufficient neural network to minimize cross-entropy loss. The network has one convolutional layer (64 channels, 3x3 kernel, stride of 1) and 2 fully-connected linear layers (128 wide and 64 wide) with ReLU activation functions. Images are classified with a softmax over the output layer. With the standard MNIST 10-class classification and simple SGD optimization, this network is complex enough to achieve 100% training accuracy in the standard MNIST training set.

Before we discuss further specifics of SGD, we first discuss a recently explored challenge that arises when using neural networks for endless adaptation: the *catastrophic loss of plasticity*, where deep neural networks demonstrate an abrupt and potentially complete failure to adapt, even when future data is drawn from a consistent distribution (Dohare et al., 2022; 2024). Recent work by Lyle et al. (2023) has identified that this problem cannot be explained by simple saturation or gradient pathologies, and both parametrization and optimization methods influence this. Furthermore, loss of plasticity seems to be hastened by non-stationary learning tasks, such as the testbed being used in this work. Indeed, we observed this phenomenon with vanilla SGD in long training runs.

Two versions of the SGD learner form the basis for our future comparisons, SGD and SGD+Reset. As a reminder, the + signals non-experiential components in the learning algorithm. In both agents, the network weights are randomly initialized and updated with one α-weighted gradient step per timestep. The gradient is with respect to the cross-entropy loss, with α tuned independently for each experiment setting. We did not use momentum, gradient clipping, or normalization as these techniques often hastened the onset of catastrophic loss of plasticity. SGD is initialized at the start of the experiment and a gradient step is taken on each timestep, and thus matches the criteria for an experiential learner. It may be able to transfer some knowledge between tasks, but may also suffer from instability and loss of plasticity due to optimizing for past tasks. The non-continual version, SGD+Reset, receives an extra signal from the environment when the task has changed and randomly re-initializes the weights of its network. This means it cannot transfer knowledge across task boundaries, but its weights are only updated with respect to the current task. This provides a baseline for perfectly plastic learning.

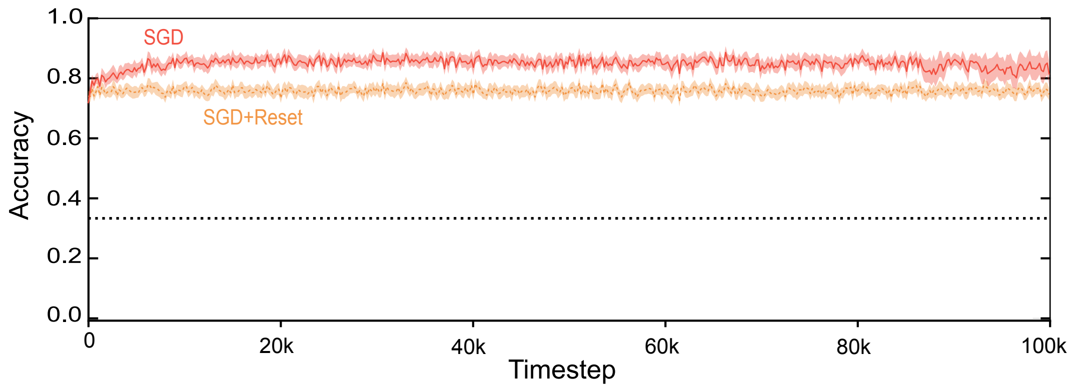


Figure 3: Classification accuracy of SGD and SGD+Reset, averaged over 30 random seeds. The dotted black line at 33.3% indicates the expected accuracy of uniform random classification.

SGD Results. In all graphs, the agent's performance is averaged over 30 trials. The random seeds are consistent across learners, so all learners experienced the same sequence of tasks, and (when relevant) the same network initialization. For clarity in the graph, data is quantized into intervals of 200 timesteps, and each data point on the graph is the average interval accuracy over the 30 trials. The shaded region indicates the 95% confidence interval.

Both SGD learners significantly outperform the uniform random baseline of 33.3% accuracy. In spite of the short task length in which to learn, SGD+Reset quickly recovers from the random re-initialization to classify labels correctly ($75.8 \pm 0.1\%$). The performance of SGD ($85.0 \pm 0.3\%$) improves on SGD+Reset, indicating that the experiential learner is benefiting from cross-task transfer, in spite of the non-i.i.d. setting and catastrophic label interference inherent in Switching MNIST. The increase in SGD's variance towards the end of the trial indicates early signs of loss of plasticity as some trials begin failing to recover following task switches. See Appendix B for a discussion of the increased variance and catastrophic loss of plasticity that occurs beyond the 100,000 timestep horizon.

3.2 Partition-Tree Weighting

Partition-Tree Weighting (PTW; Veness et al., 2013) provides a mechanism for generalizing a base learning algorithm to non-stationary problems, by maintaining copies of the base learner's models that are automatically re-initialized at set points in time. Using $\lceil \log_2(T) \rceil$ models through time T, PTW creates a mixture model that closely approximates the performance of the base learner on the best possible partition of the timesteps into piecewise stationary models (see Veness et al. (2013) for theoretical details). When learning across all experience is the best course of action, as in a typical i.i.d. learning task, PTW places most of its mixture weight on the longest-running model. When a model can benefit from training only on recent data, as in some piecewise-stationary tasks, PTW places more weight on recently initialized models. It is able to do this without any awareness of task distributions or switching times.

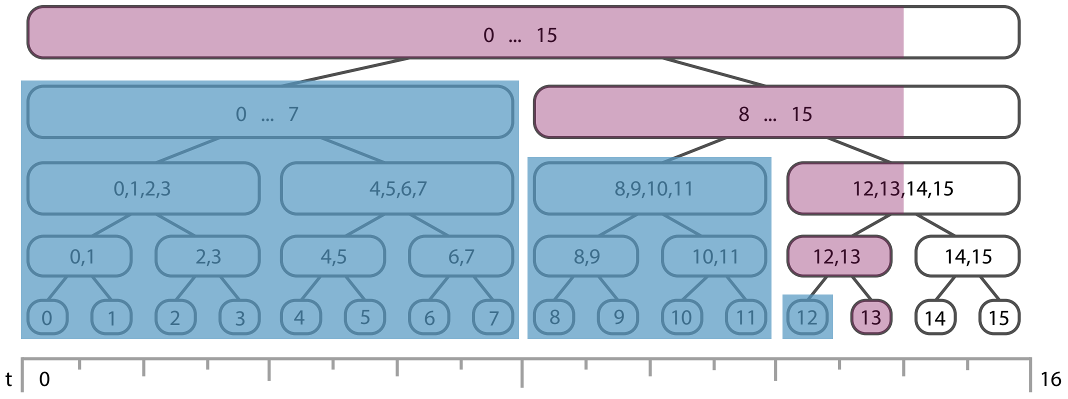


Figure 4: An illustration of PTW's internal structure. Each node (rounded black rectangle) in the binary tree represents one particular instantiation of the base learner that is updated during the timesteps indicated within the box. Purple highlights indicate active models at timestep 13, which are updated and used for inference. The blue boxes represent older models that are no longer used, and are instead represented by summary statistics for their subtree.

PTW's internal structure is illustrated in Figure 4. The black rounded rectangles that are nodes in a tree represent particular instantiations of the base learner, and the label indicates the span of time on which it was trained. A model at height d in the tree observes and is updated on 2^d batches of data; a leaf model is updated only once, and the root model is updated on all data. Figure 4 represents the state of the learner at timestep 13. Although the tree must contain 31 different nodes to cover all possible binary partitions of the 16-timestep sequence, only the 5 models highlighted in purple that contain the current timestep are stored in memory, updated, or used for inference. The older "completed" models, highlighted in blue, do not need to be stored or used for inference. This binary partition of the 16 timesteps requires at most $\lceil \log_2(T) \rceil$ active models to approximate, within a small bound, the best-partition-in-hindsight for experiments of length T, which otherwise would require a doubly-exponential number of models in T. PTW efficiently computes the loss incurred by all binary partitions by summarizing each subtree as its time interval is completed. See Veness et al. (2013) for details.

PTW Results. We apply PTW with SGD as the base learner to our Switching MNIST task. As can be seen in Figure 5, PTW achieves ($87.3 \pm 0.3\%$) accuracy, which improves on SGD's ($85.0 \pm 0.3\%$). Note that in this case, we do not see the increased variance in performance that SGD suffers that signals a loss of plasticity; see Appendix B for further results of SGD and PTW with a longer horizon.

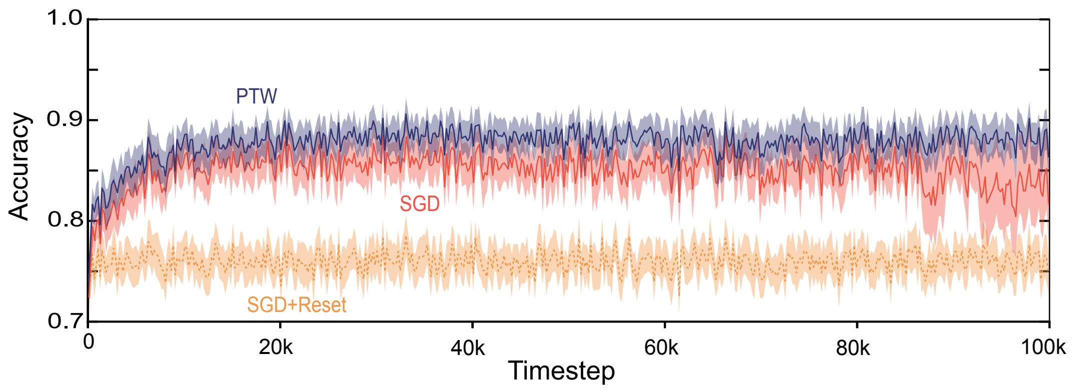


Figure 5: Classification accuracy of PTW on randomized Switching MNIST. By using a mixture of long-term and short-term SGD models, PTW outperforms both SGD (another experiential learner) and SGD+Reset, which uses context information to re-initialize.

3.3 Model-Agnostic Meta-Learning

To improve the initial performance of short-running models, we use Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017). MAML was designed as an offline meta-learning procedure for discovering useful weight initializations for deep networks. Its meta-training defines the loss in terms of performance **after** some steps of adaptation, rather than the model's current performance. MAML training does this by embedding an update step in the meta-loss calculation, and calculating a change to the initial weights with respect to the updated model. This requires training data to be grouped by task, with carefully controlled sampling from distinct tasks. Before we look into adapting MAML for the experiential case, we will evaluate its performance when trained offline and with the context of task changes. We will then show how the offline-trained MAML initialization improves both SGD and PTW's performance.

MAML is a gradient-descent learning algorithm with a meta-loss function split into an inner and outer loop. The inner loop executes task-specific tuning of the network weights, using the gradient of the loss on a sample from the current task(s). With another sample from the same task, the outer loop then uses the post-adaptation loss (*i.e.*, the gradient of the loss with respect to the fine-tuned weights) to adjust the pre-adaptation weights. This training process repeats offline on tasks randomly sampled according to the experimental distribution, resulting in a set of network weights ($\text{MAML}_{\text{init}}$) that provide fast adaptation for few-shot learning (Finn et al., 2017).

Although Switching-MNIST is defined as an experiential learning problem, we can exploit its piecewise-stationary structure to construct an offline training regime. For every training step, we sample a random task and corresponding training batch, then split it into a *tuning batch* for the inner loop and *validation batch* for the outer. We chose to split the data evenly between the update and validation batch. This even split was the most consistent in our hyperparameter sweep. Experiments with different training batch sizes did not show any consistent improvement in results, so we used a training batch size of 16 so that the tuning batch matched the batch used in the experiential setting.

The MAML model has a standard construction with cross-entropy loss, a simple SGD update rule in the inner loop and Adam optimiser in the outer loop. The meta-loss function uses the tuning batch to execute a single gradient step from the initial network weights, and then calculates the cross-entropy error of the updated weights on the validation batch. For offline MAML training, we sample from 100,000 random tasks, which we found had the best experimental performance without risk of overtraining. As others have noted, MAML can be sensitive to hyperparameter settings and we found a larger training set greatly increased the variance across seeds without significantly improving performance (Antoniou et al., 2018; Nichol et al., 2018).

For the actual experiment, the pretrained weights ($\text{MAML}_{\text{init}}$) are used by two SGD learning algorithms as in Section 3.1. MAML+SGD is continual and task-agnostic: after initializing the network with $\text{MAML}_{\text{init}}$, MAML+SGD uses SGD to fine-tune continually, without resetting for the duration of the experiment. MAML+SGD+Reset is task-aware: at every task switch, the SGD network weights are re-initialized to $\text{MAML}_{\text{init}}$.

MAML Results. The initial weights that MAML learns provide a clear benefit for a non-experiential SGD learner, as shown in Figure 6. Not only is MAML+SGD+Reset's accuracy ($96.5 \pm 0.6\%$) much higher than SGD+Reset

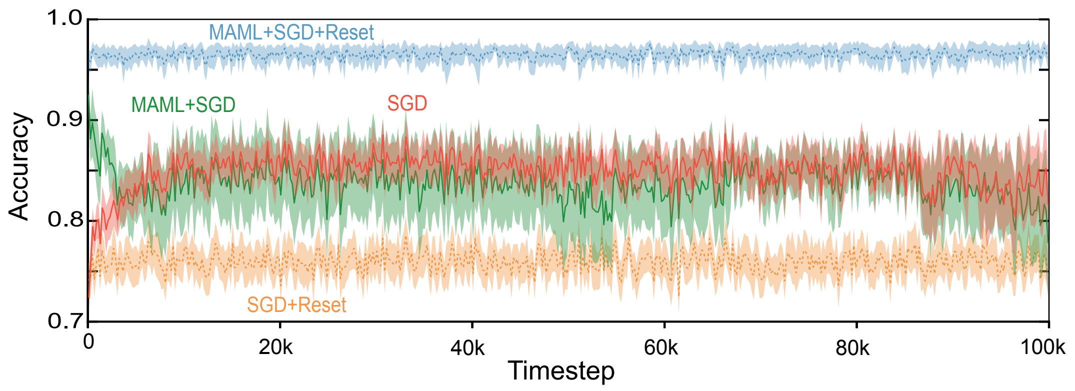


Figure 6: Effect of MAML pretraining compared to random initialization in Switching MNIST. The offline-trained, task-aware MAML+SGD+Reset significantly outperforms the more experiential learners. Although MAML+SGD uses the same offline-trained initialization, without explicit reset it only briefly outperforms SGD and has higher variance.

$(75.8 \pm 0.1\%)$, it also outperforms SGD $(85.0 \pm 0.3\%)$. On the other hand, when not explicitly reset when the task changes, MAML+SGD averages $(83.6 \pm 1.7\%)$ accuracy, and only briefly benefits over random initialization.

The benefits of starting from a MAML initialization are clear, but as it must be trained offline, observe context switches, and explicitly reset, it cannot be applied directly to the experiential learning problem. The next step is to combine a MAML initialization with PTW for continual learning, then to explore possibilities for training MAML online.

3.4 MAML AND PTW

The combination of MAML and PTW for continual, task-agnostic learning is a simple matter of training a MAML initialization (MAML$_{\text{init}}$) and providing it to a PTW learner to use in place of random initialization. We will use MAML+PTW to refer to this combination.

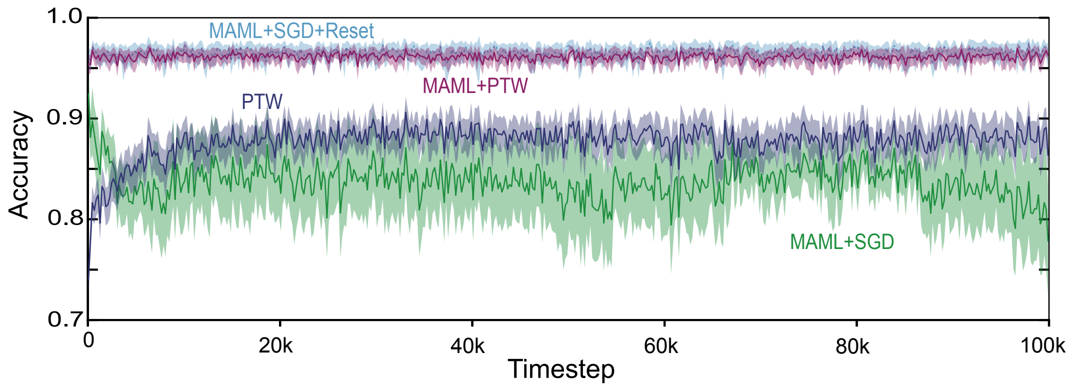


Figure 7: MAML+PTW performs almost as well as MAML+SGD+Reset without requiring observable context. It also maintains that performance, where MAML+SGD experiences the same collapse as SGD

MAML+PTW Results. Figure 7 shows that MAML provides a dramatic improvement to PTW's performance. The average accuracy of MAML+PTW $(96.2 \pm 0.5\%)$ is significantly greater than that of PTW $(87.3 \pm 0.3\%)$. It also nearly matches the accuracy of the non-continual MAML+SGD+Reset[1] $(96.5 \pm 0.6\%)$, even though MAML+SGD+Reset

[1]Although the confidence intervals overlap due to the noisy task, MAML+SGD+Reset's per-timestep accuracy is consistently slightly above MAML+PTW.

is notified of task changes and MAML+PTW is not. It is important to note that all of these methods, including MAML+PTW, are relying on offline pre-training using explicit knowledge of the task distribution. To use MAML for the fully experiential case, this must be trained online as well.

3.5 Replay-MAML

MAML+PTW is not an experiential learner due to the pretraining required by MAML, and so our final step in creating an experiential learner is to remove this requirement. To do so, we propose a simple MAML extension that uses a relatively small replay buffer and is trained entirely online. On every timestep we execute a standard MAML update using a random contiguous block from the circular replay buffer. Our tests show this works in our experiential setting even with a small buffer, a single gradient step per timestep, and training batches that require no context.

The Replay-MAML buffer is a circular buffer where each index contains a single $(\mathbf{x}, \mathrm{y})$ pair. At each timestep, the input-class pairs from the current timestep are written into contiguous indices in the buffer. When the write index reaches the end of the buffer, it is moved back to the start so that new samples overwrite the oldest. This simple structure has several important benefits: the meta-training does not have to use the same batch size as the online experience, and the meta-learned weights are updated on many different tasks even though the buffer may be small and holds only the most recent tasks. However, this also means that when sampling from the buffer, Replay-MAML provides no guarantees that the entire sample is drawn from the same distribution: it is entirely possible that the training sample straddles one or more task switches. In our tests, discussed below, this proved to have surprisingly little effect on performance.

Replay-MAML can be seen as a variant of Follow the Meta Leader (FTML; Finn et al., 2019). FTML uses a different experimental setup where data is identified by task and split into training and evaluation data. However, the procedure for training in an online fashion is identical to Replay-MAML except they assume an unbounded replay buffer that contains all past task data, and there training batches for the meta-learning update are guaranteed to be from the same task. Replay-MAML demonstrates neither of those qualities are needed to learn a good weight initialization.

The online meta-training procedure is simple: starting from a random index into the buffer, take a contiguous block of examples as the meta-batch. Shuffle these, and split them into update and validation sets for the normal MAML update. Note that regardless of the total size of the buffer, the first meta-training update happens as soon as the buffer holds enough samples for a single training batch. Thereafter, Replay-MAML does one meta-training update on each timestep. Because only one update is performed and the replay buffer has a constant length, an agent using Replay-MAML performs constant time and memory additional work per timestep.

For online learning, the meta-trained weights can be used directly in place of the offline-trained $\mathrm{MAML}_{\mathrm{init}}$. Whenever the online learner needs to initialize network weights, it queries Replay-MAML to retrieve the current $\mathrm{MAML}_{\mathrm{init}}$ weights. On the first few timesteps the Replay-MAML $\mathrm{MAML}_{\mathrm{init}}$ weights are no worse than random initialization, and this starting point quickly improves.

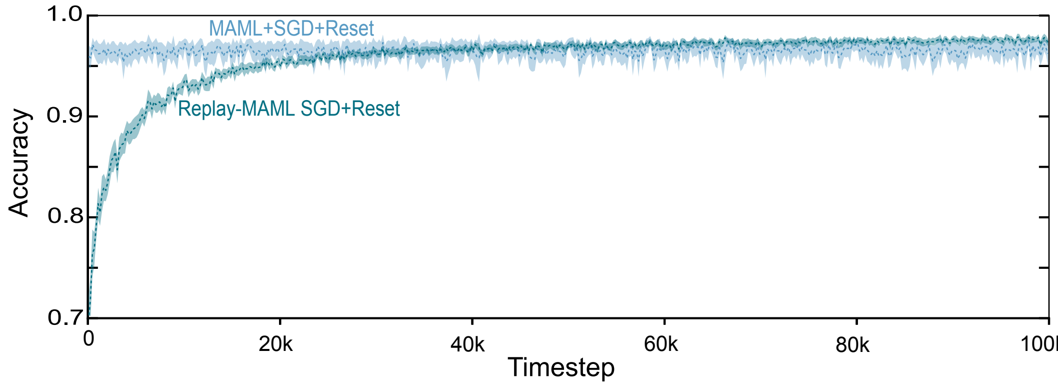


Figure 8: Replay-MAML SGD+Reset compared to MAML+SGD+Reset. The online Replay-MAML SGD+Reset quickly reaches and then surpasses the performance of the offline MAML+SGD+Reset. Though online, Replay-MAML SGD+Reset is not an experiential learners as it requires observable context switches.

Replay-MAML results. The experiments shown here use a meta-training batch size of 16 and split the data evenly between inner and outer loss calculations. This means the meta-training batch may span data from different tasks 3.7% of the time. Our buffer size includes 1,250 timesteps of data and so contains on average 25 tasks. See Appendix C.1 for experiments varying the buffer size.

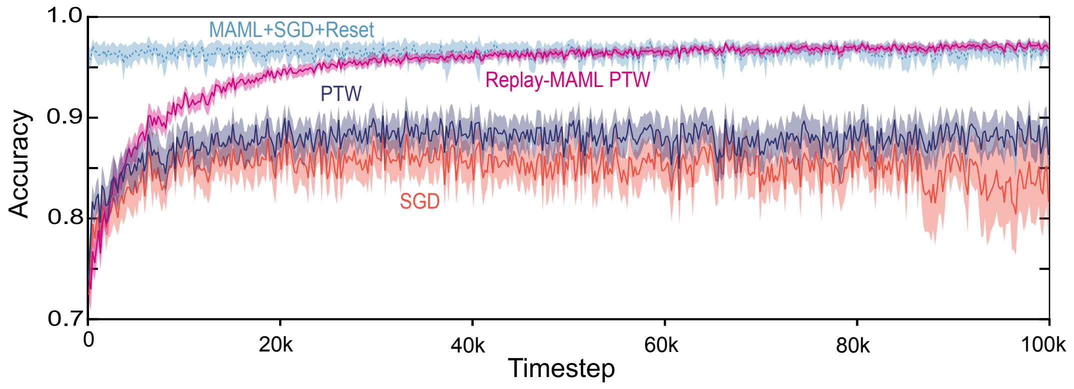


Figure 9: Replay-MAML PTW performance compared to the other experiential learners and MAML+SGD+Reset. Within the first 5,000 timesteps Replay-MAML PTW surpasses both PTW and SGD. Furthermore, within 100,000 timesteps it matches the performance of MAML+SGD+Reset.

As can be seen in Figure 8, Replay-MAML SGD+Reset quickly improves its random initialization. Within 25,000 timesteps it has reached MAML+SGD+Reset's average accuracy ($96.5 \pm 0.6\%$), while only having been updated a fraction of the time with samples from 500 different tasks (on expectation) rather than 100,000. By 100,000 timesteps, the online Replay-MAML SGD+Reset ($97.8 \pm 0.0\%$) outperforms the offline MAML+SGD+Reset ($96.5 \pm 0.6\%$). In spite of sampling from a relatively small replay buffer and task-agnostic training, Replay-MAML SGD+Reset produces a successful MAML initialization, completely online.

We found it surprising that Replay-MAML could surpass offline MAML. One reason appears to be that training online naturally gives rise to repeated samples from the same task. Rather than being a problem, especially for long-horizon tasks, it appears to be a benefit. Recall that our offline MAML training procedure samples one batch per task, whereas various suggested modifications of the offline MAML training allow for more batches per task (e.g. Gupta et al., 2020; Collins et al., 2022; Nichol et al., 2018; Finn et al., 2017). We experimented with more batches per task for offline MAML training, but could not find hyperparameters that gave stable training for our task. A second reason might be a regularizing effect caused by the 3.7% probability that a training batch might be noisy due to straddling task boundaries.

3.6 Replay-MAML PTW

Finally we have all the pieces we need for our fully experiential learner, with no offline training or additional context information required: PTW for forming a mixture of models reset at different frequencies to track task changes and Replay MAML to learn good model initializations online. We call this combination Replay-MAML PTW, as a fully experiential learning algorithm.

Replay-MAML PTW is a powerful combination of its component parts, and returns us to Figure 1. Within 5,000 timesteps Replay-MAML PTW has surpassed the accuracy of both PTW ($87.3 \pm 0.3\%$) and SGD ($85.0 \pm 0.3\%$), our two fully experiential baselines. By 50,000 timesteps Replay-MAML PTW has matched the performance of MAML+SGD+Reset ($96.5 \pm 0.6\%$), which requires both knowledge of the true task distribution and observable context switches. By 100,000 timesteps, Replay-MAML PTW ($97.5\% \pm 0.1\%$) has surpassed the best task-aware learner while operating entirely within the experiential constraints.

4 Other Datasets

We also apply Replay-MAML PTW to Fashion-MNIST (Xiao et al., 2017) and CIFAR-10 (Krizhevsky, 2009), two more challenging classification datasets. We used the same switching regime where each task has 3 classes each made up of 2 labels from the underlying dataset with a 2% probability of a task switch after every batch of 8 instances.

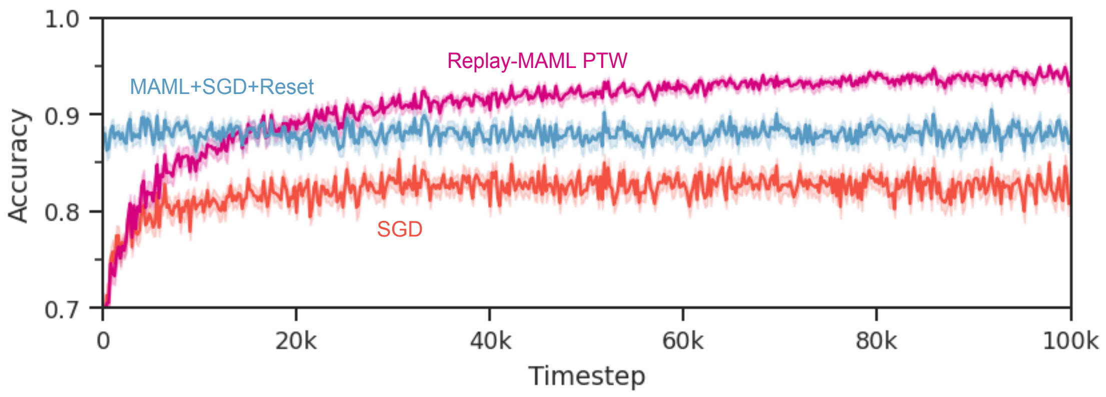


Figure 10: Replay-MAML PTW performance compared to SGD and MAML+SGD+Reset on the Switching Fashion-MNIST task.

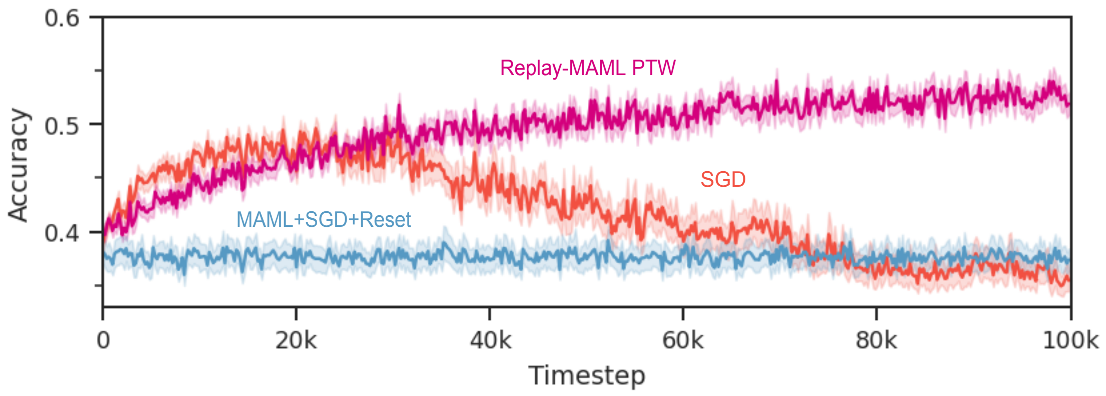


Figure 11: Replay-MAML PTW performance SGD and MAML+SGD+Reset on the Switching version of CIFAR10.

We used the same base neural network architecture and sweep of hyperparameters for all algorithms. Figures 10 and 11 shows the final comparison on these datasets mirroring Figure 1. We see qualitatively similar results on Fashion-MNIST, where Replay-MAML PTW significantly outperforms SGD, while matching and exceeding the performance of MAML+SGD+Reset, which is allowed to pre-train with the underlying task distribution and resets weights on signals of task changes. In CIFAR-10, a similar trend holds, but in this case MAML+SGD+Reset with offline training struggled to find a good weight initialization, which might be due to known instability issues and high sensitivity to hyperparameters (Antoniou et al., 2018). Meanwhile, SGD starts to exhibit plasticity loss even within the span of 100,000 timesteps, whereas Replay-MAML PTW shows continual improvement with no signs of plasticity loss. In these datasets, the improvement over MAML+SGD+Reset is more notable than with MNIST, and we believe this is due to the increased difficulty of the task highlighting fast learning after a task switch. As discussed in Section 3.5, Replay-MAML's online training procedure seems to give some training advantages through repeated batch training on the same task or regularization due to a small fraction of the training data spanning multiple tasks.

5 Conclusion

In this paper we introduced Replay-MAML PTW, an approach to experiential learning that combines two meta-learning techniques: PTW for online learning in non-stationary settings, and MAML for learning to initialize a model that adapts quickly. We demonstrated our approach in a piecewise-stationary classification task. We show it is able to continually adapt, even outperforming methods that make use of offline sampling from the task distribution and explicit signaling of task changes. The benefits of Replay-MAML PTW does come at a $O(\log T)$ computation and memory cost. In the future, we hope to adapt our approach to the continual reinforcement learning setting, and explore reducing the additional small computation cost by restricting PTW's restarts to subsets of the networks.

REFERENCES

David Abel, Andre Barreto, Benjamin Van Roy, Doina Precup, Hado P. van Hasselt, and Satinder Singh. A Definition of Continual Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 50377–50407, December 2023.

Rahaf Aljundi. *Continual Learning in Neural Networks*. PhD thesis, arXiv, October 2019.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your MAML. In *International Conference on Learning Representations*, September 2018.

Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Page-Caccia, Issam Hadj Laradji, Irina Rish, Alexandre Lacoste, David Vázquez, and Laurent Charlin. Online Fast Adaptation and Knowledge Accumulation (OSAKA): A New Approach to Continual Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 16532–16545. Curran Associates, Inc., 2020.

Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. How Does the Task Landscape Affect MAML Performance? In *Proceedings of The 1st Conference on Lifelong Learning Agents*, pp. 23–59. PMLR, November 2022.

Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6):141–142, November 2012. ISSN 1558-0792. doi: 10.1109/MSP.2012.2211477.

Shibhansh Dohare, A Rupam Mahmood, and Richard S Sutton. Continual Backprop: Stochastic Gradient Descent with Persistent Randomness. In *Multi-Disciplinary Conference on Reinforcement Learning and Decision Making*, 2022.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, August 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07711-7.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1126–1135. PMLR, July 2017.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1920–1930, 2019.

R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, April 1999. ISSN 1879-307X. doi: 10.1016/s1364-6613(99)01294-2.

Gunshi Gupta, Karmesh Yadav, and Liam Paull. Look-ahead Meta Learning for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11588–11598. Curran Associates, Inc., 2020.

Raia Hadsell, Dushyant Rao, Andrei A. Rusu, and Razvan Pascanu. Embracing Change: Continual Learning in Deep Neural Networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, December 2020. ISSN 1879-307X. doi: 10.1016/j.tics.2020.09.004.

Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei Alex Rusu, Yee Whye Teh, and Razvan Pascanu. Task Agnostic Continual Learning via Meta Learning. In *4th Lifelong Machine Learning Workshop at ICML 2020*, July 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Matthias De Lange, Gido M. van de Ven, and Tinne Tuytelaars. Continual evaluation for lifelong learning: Identifying the stability gap. In *The Eleventh International Conference on Learning Representations*, September 2022.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding Plasticity in Neural Networks. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 23190–23211. PMLR, July 2023.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms, October 2018.

Shai Shalev-Shwartz. *Online Learning and Online Convex Optimization*, volume 4. Now Publishers, 2012. doi: 10.1561/2200000018. URL http://dx.doi.org/10.1561/2200000018.

Richard S. Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 871–878, New York, NY, USA, June 2007. Association for Computing Machinery. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273606.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, December 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00568-3.

Joel Veness, Martha White, Michael Bowling, and András György. Partition Tree Weighting. In *2013 Data Compression Conference*, pp. 321–330, March 2013. doi: 10.1109/DCC.2013.40.

Eli Verwimp, Rahaf Aljundi, Shai Ben-David, Matthias Bethge, Andrea Cossu, Alexander Gepperth, Tyler L. Hayes, Eyke Hüllermeier, Christopher Kanan, Dhireesha Kudithipudi, Christoph H. Lampert, Martin Mundt, Razvan Pascanu, Adrian Popescu, Andreas S. Tolias, Joost van de Weijer, Bing Liu, Vincenzo Lomonaco, Tinne Tuytelaars, and Gido M. van de Ven. Continual Learning: Applications and the Road Forward. *Transactions on Machine Learning Research*, November 2023. ISSN 2835-8856.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(08):5362–5383, August 2024. ISSN 0162-8828. doi: 10.1109/TPAMI.2024.3367329.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, September 2017.

A Hyperparameters

Hyperparameters were determined with a grid search, based on the optimal average performance over 100,000 online updates.

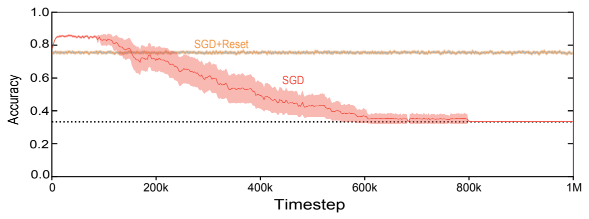
Dataset	Algorithm	α	outer-loop α
MNIST	SGD	0.05	-
	SGD+Reset	0.01	-
	MAML+SGD+Reset	0.05	3e-4
	MAML+SGD	0.05	1e-4
	Replay-MAML SGD+Reset	0.01	3e-4
	PTW	0.05	-
	MAML+PTW	0.01	3e-4
	Replay-MAML PTW	0.01	3e-4
Fashion MNIST	SGD	0.05	-
	MAML+SGD+Reset	0.01	7e-5
	Replay-MAML PTW	0.005	1e-3
CIFAR-10	SGD	0.05	-
	MAML+SGD+Reset	0.005	1e-4
	Replay-MAML PTW	0.005	1e-4

Table 1: Learning rate parameters used in experiment results.

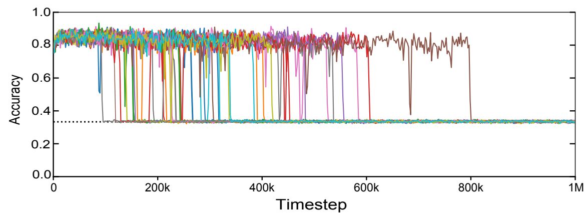
B Catastrophic loss of plasticity

Even in the case of SGD, if we extend the life of our continual learner, as shown in Figure 12, a surprising issue emerges. The accuracy of SGD+Reset naturally remains consistent, as reset makes its performance on each task independent. SGD has a dramatic collapse, and falls to accuracy of uniform random classification. This is not, as Figure 12a might suggest, due to any gradual decay in accuracy, but an abrupt and permanent plummet on each individual trial from peak to random performance. Within our 30 trials the earliest collapse happened after just 92,000 updates, and by one million updates all 30 learners had collapsed, as can be seen in Figure 12b.

Our experiments confirm that catastrophic loss of plasticity is a consistent phenomenon and surfaced in all settings we tried, if we just ran the online evaluation for long enough. While SGD improves on the completely plastic SGD+Reset for sufficiently short experiments, the catastrophic loss of plasticity means SGD alone is not a solution to experiential learning.



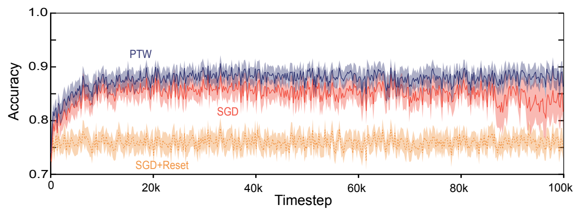
(a) Accuracy averaged over 30 trials. The performance of the experiential SGD learner, initially best, falls to uniform random.



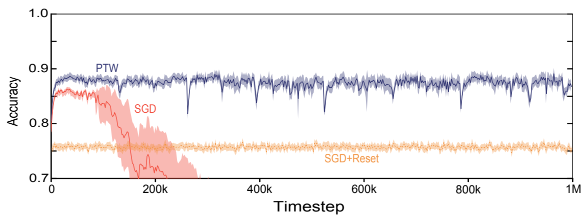
(b) In each of the 30 trials, the SGD learner eventually suffers a catastrophic loss of plasticity.

Figure 12: SGD accuracy over one million timesteps.

Furthermore, Figure 13b shows how over the full million timesteps PTW $87.4, \pm\, 0.2\%$ does not suffer the same collapse in performance. In fact, none of its 30 seeds experienced the catastrophic loss of plasticity that hit all SGD learners.



(a) Accuracy over the first 100,000 timesteps. Using a mixture of long- and short-term SGD models PTW outperforms the other experiential learner, SGD, and SGD+Reset.



(b) Long-term performance of PTW compared to SGD and SGD+Reset. While every SGD run stops tracking within a million timesteps, PTW maintains good performance across every seed.

Figure 13: Accuracy of PTW compared to SGD and SGD+Reset.

C Effect of buffer size on Replay-MAML

C.1 Replay-MAML results in Switching MNIST

Combining Replay-MAML with another learner thus introduces a computational cost on each timestep for the MAML update, and a memory cost for the replay buffer to store previously observed samples and tasks. The computational cost of Replay-MAML is roughly equal to the cost of an update for SGD+Reset, which we consider an acceptable tradeoff for eliminating the need for MAML offline training and *a priori* knowledge of the task distribution that will be encountered during the online phase. The memory cost of Replay-MAML depends on the size of the replay buffer used. A large replay buffer is expensive, but random samples drawn from it will more closely approximate the stream of new tasks used for training MAML. A small replay buffer is cheap, but random samples from it will be more likely to repeat the most recently observed tasks, or even the current task. We will investigate this tradeoff empirically, to discover how small the replay buffer can get while still providing good performance.

As can be seen in Figure 14, Replay-MAML quickly improves its random initialization, reaching and even exceeding offline MAML's average accuracy of $96.5 \pm 0.6\%$ (indicated on the graph with the dotted line). The red line illustrates our default buffer size. Note that within 2500 update steps, having seen on average only 50 different tasks, it is performing as well as MAML trained on 100,000 independent samples. In fact, we see that while larger buffers improve the early performance of the learner, all four of the learners match and then slightly surpass the accuracy of MAML+SGD+Reset, while training on about half of the 100,000 batches used for MAML pretraining. Further, relatively small replay buffers are sufficient: the 1250 buffer size learner stores just 1.25% of the data observed online, with no loss in accuracy compared to the 12500 buffer size learner.

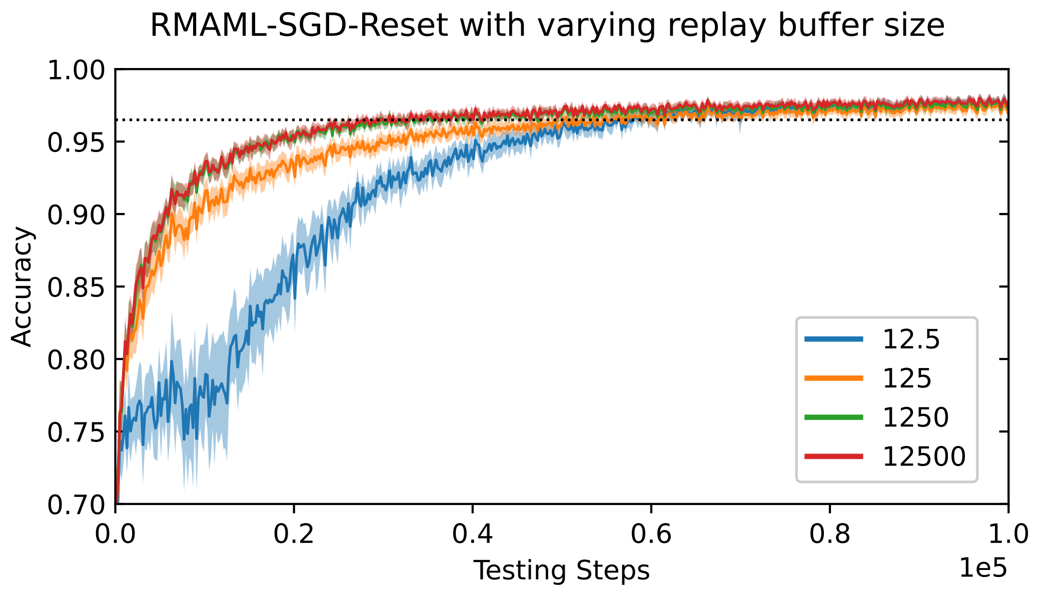


Figure 14: Average online accuracy of Replay-MAML SGD+Reset trained with four replay buffer sizes, averaged across 30 random seeds. The legend indicates the timesteps spanned by the buffer.

C.2 Online-MAML results in Switching MNIST

We found Replay-MAML's resilience to small buffer sizes surprising. We had expected when the buffer was smaller than the average task length, Replay-MAML's performance would suffer dramatically. And although larger buffer sizes perform better, the fact that meta-training is executed almost exclusively on the current data distribution does not stop Replay-MAML from learning a very good initialization. Naturally, we wondered if we could do away with the buffer entirely.

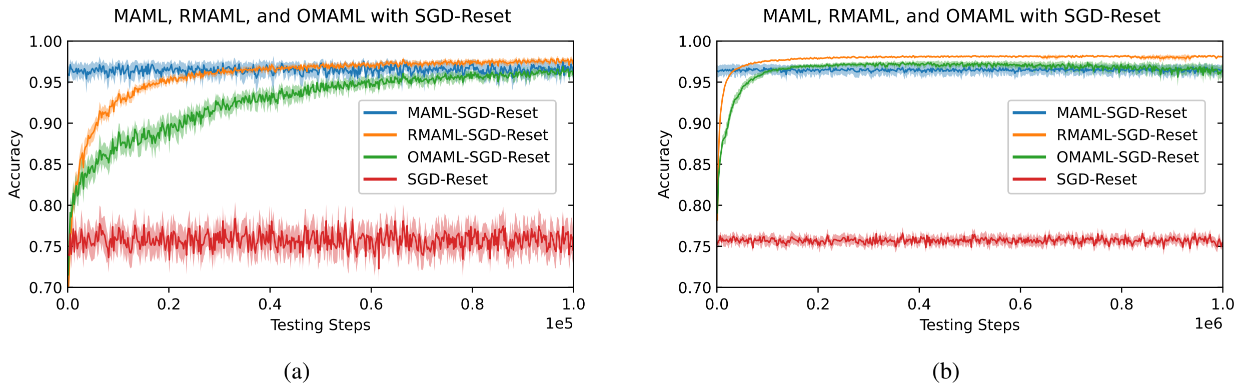


Figure 15: Average online accuracy of 30 independent runs of offline (MAML), replay-buffer (Replay-MAML), and online (OMAML) MAML learners.

With no replay buffer, our online MAML update (OMAML) is the incremental version of the update described previously: execute the meta-training MAML update using only the current batch of data. For simplicity we simply divided each batch in half, leaving us with 4 samples for the inner SGD step and 4 for outer meta-update. When a context switch is observed, the online SGD+Reset learner pulls the most recent parameter set from Replay-MAML.

The performance of OMAML is illustrated in Figure 15. The average performance of the best offline-trained MAML-SGD+Reset is shown in blue. Although OMAML does not exceed offline MAML's average performance within the first 100,000 updates, it is able to match it in the long run.