LEARNING TO LEARN WITHOUT FORGETTING BY MAXIMIZING TRANSFER AND MINIMIZING INTERFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Lack of performance when it comes to continual learning over non-stationary distributions of data remains a major challenge in scaling neural network learning to more human realistic settings. In this work we propose a new conceptualization of the continual learning problem in terms of a trade-off between transfer and interference. We then propose a new algorithm, Meta-Experience Replay (MER), that directly exploits this view by combining experience replay with optimization based meta-learning. This method learns parameters that make interference based on future gradients less likely and transfer based on future gradients more likely. We conduct experiments across continual lifelong supervised learning benchmarks and non-stationary reinforcement learning environments demonstrating that our approach consistently outperforms recently proposed baselines for continual learning. Our experiments show that the gap between the performance of MER and baseline algorithms grows both as the environment gets more non-stationary and as the fraction of the total experiences stored gets smaller.

1 THE STABILITY-PLASTICITY DILEMMA AND CONTINUAL LEARNING

A long-held goal of the AI community is to build agents capable of operating autonomously for long periods. Such agents must incrementally learn and adapt to a changing environment while maintaining memories of what they have learned before, a setting known as lifelong, or continual learning (Thrun, 1996). Over the years, much research in this domain has focused on the stability-plasticity dilemma (Carpenter & Grossberg, 1987), also referred to as the catastrophic forgetting problem (McCloskey & Cohen, 1989), where an agent must learn a current task without degrading its performance on the previous tasks (Goodrich, 2015). It is often assumed that the the learner is exposed to a sequence of tasks, where each task is a sequence of (labeled) samples from the same distribution, and the distributions the tasks come from are different (Lopez-Paz & Ranzato, 2017). Given a potentially infinite number of such tasks/distributions in continual learning problem, it is important to consider stability-plasticity trade-off in both forward and backward directions in time (Figure 1).

Indeed, previously proposed solutions to the catastrophic forgetting problem are not well-suited to addressing the full continual learning problem. To be specific, many of those approaches focus on the preservation of past learning by reducing the degree of neural network weight-sharing across the past and current tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Lee et al., 2017). By only partially sharing weights across tasks, semi-distributed representations are able to overcome the natural recency bias in neural nets and more cleanly segregate past from current learning (French, 1991; Goodrich, 2015; Rosenbaum et al., 2018). However, it is still possible that the reduction in weight sharing may worsen future performance on unseen tasks by limiting the networks capacity for forward transfer.

Here we seek to find a solution to the more general continual learning problem that also performs well in the more limited stability-plasticity context. More specifically, we propose a new conceptualization of the continual lifelong learning problem that generalizes the stability-plasticity dilemma to a transfer-interference trade-off that extends into the past and, critically, to the future as well (section 2). The key difference in perspective with past conceptualizations of continual learning is that

we are not just concerned with current interference and transfer with respect to past examples (i.e., the classical stability-plasticity dilemma), but also concerned with the dynamics of transfer and interference moving forward as we learn. This perspective inspires a meta-learning formulation of the continual learning problem where on top of learning to explain all examples seen, we also regularize our learning so that it learns to make transfer more likely and interference less likely in the future. To the extent that our meta-learning into the future generalizes, this should effectively make it easier for our model to effectively perform continual learning in non-stationary settings.

2 THE TRANSFER-INTERFERENCE TRADE-OFF FOR CONTINUAL LEARNING

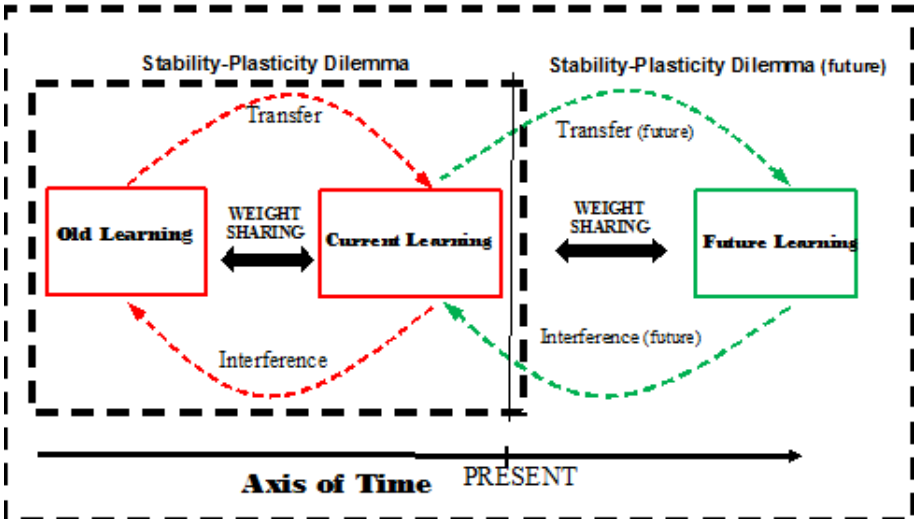

Figure 1: In the stability-plasticity dilemma, current learning can degrade old learning (red lines). It is important to address this well-known problem without reducing the networks ability to perform in the future (green lines). The transfer-interference trade-off is the stability-plasticity dilemma considered in both forward and backward directions. This bi-directional view is crucial since solutions to the stability-plasticity dilemma that reduce the degree of weight-sharing are unlikely to function well into the future.

Throughout our paper we will discuss ideas in terms of the supervised learning problem formulation. Extensions to the reinforcement learning formulation are straightforward and we provide more details in the appendix. In typical offline supervised learning with neural networks, we can express our optimization objective in terms of the stationary distribution of x, y pairs within the dataset D. Here we define the continual learning problem as sampling from a never ending stream of x, y pairs with no knowledge or assumptions about the underlying distributions from which the data are drawn. The expectation of the loss at any point in time is simply taken over past history and current sample, and we optimize it for all points in the future.

$$\frac{\partial L(x_i, y_i)}{\partial \theta} \cdot \frac{\partial L(x_j, y_j)}{\partial \theta} > 0. \tag{1}$$

where $\cdot$ is the dot product operator. This implies that learning example i will without repetition improve performance on example j and vice versa. On the other hand we can say that *interference* occurs when:

$$\frac{\partial L(x_i, y_i)}{\partial \theta} \cdot \frac{\partial L(x_j, y_j)}{\partial \theta} < 0. \tag{2}$$

This implies in contrast that learning example i will lead to unlearning (i.e. forgetting) of example j and vice versa.

Past solutions described in the previous section for the Stability-Plasticity Dilemma in continual learning operate in a simplified temporal context whereby learning is divided into two phases: all

past experiences are lumped together as *old memories* and the data that is currently being learned qualifies as *new learning*. In this setting, the goal is to minimize the interference projecting backward in time, and many of the approaches that succeed in this respect do so by reducing the degree of weight sharing in one form or another. However, one need not worry about performance on the current task, because that is what is currently being learned. Therefore, solving the stability-plasticity dilemma reduces solely to mitigating interference on the old memories. In the appendix we explain how our baseline approaches (Kirkpatrick et al., 2017; Lopez-Paz & Ranzato, 2017) each fit within this paradigm. As one example, the approach EWC in (Kirkpatrick et al., 2017) promotes less sharing of parameters for new learning that were deemed to be important for performance on old memories.

The important issue with this perspective, however, is that the system is not done with learning that will occur in the future, and what the future may bring is largely unknown. This makes it incumbent upon us to do nothing to potentially undermine the networks ability to effectively learn in an uncertain future. This consideration makes us extend the temporal horizon of the stability-plasticity problem forward, turning it, more generally, into a continual learning problem that we label as solving the Transfer-Interference Trade-off (Figure 1). Specifically, it is important not only to reduce backward interference from our current point in time, but we must do so in a manner that does not limit our ability to learn in the future. This more general perspective makes the problem more difficult to solve, because the issue of weight sharing across tasks arises both backward and forward in time. When viewed from this temporally symmetric perspective, the transfer-interference trade-off becomes clear: the weight sharing across tasks that enables transfer to improve future performance must not disrupt performance on what has come previously. This brings us to the central point of the paper: if we mitigate interference by reducing weight-sharing, we are likely to worsen the networks capacity for transfer into the future, since transfer depends precisely on weight sharing: what is the future will eventually become the past, and performance must be good at all points in time.

We build off past work on experience replay (Murre, 1992; Lin, 1992; Robins, 1995) that has been a mainstay for solving non-stationary problems with neural networks (i.e. leading to human level performance on many Atari games (Mnih et al., 2015)). We argue that this is because of replays ability to stabilize learning without sacrificing weight-sharing. In section 3.2 we motivate a particularly effective variant of experience replay for continual learning. Then in section 3.3 we describe how we augment this variant of experience replay with meta-learning to derive a new Meta-Experience Replay (MER) algorithm that optimizes for the transfer-interference trade-off. In our experiments we demonstrate that MER is quite general, achieving superior sample efficient continual learning performance both in a many task supervised learning setting and within highly non-stationarity RL tasks. In contrast to past work on meta-learning for few shot learning (Santoro et al., 2016; Vinyals et al., 2016; Ravi Larochelle, 2016; Finn et al., 2017) and reinforcement learning across successive tasks (Al-Shedivat et al., 2018) we are not trying to simply improve the speed of learning on new data. In this paper we are trying to improve the speed of learning on new data in a way that also preserves knowledge of past data. Critically, our approach also is not reliant on any notion of tasks being provided by humans to the model. In fact, in most of the settings we explore, we consider the more general case where the model must detect the concept of new tasks on its own without supervision.

3 A System for Learning to Learn without Forgetting

In this section we outline the motivation and details of our approach Meta-Experience Replay (MER) meant to enable general purpose continual learning while efficiently solving for the transfer-interference trade-off outlined in the last section.

3.1 Motivation

In typical offline supervised learning with neural networks, we can express our optimization objective in terms of the stationary distribution of x, y pairs within the dataset D:

$$\theta = arg \min_{\theta} \mathbb{E}_{(x,y)\in D}[L(x, y)], \tag{3}$$

where L is the loss function, which can be selected to fit the problem. If we would like to maximize transfer and minimize interference, we can imagine it would be useful to add an auxiliary loss to the objective to bias the learning process in that direction. One obviously beneficial choice would be to also directly consider the gradients with respect to the loss function evaluated at randomly chosen datapoints within the dataset. If we could maximize the dot products between gradients at these different points, it would directly encourage the network to share parameters where gradient directions align and keep parameters separate that interfere with each other that have gradients in opposite directions. So, in an ideal case we would like to optimize for the following objective:

$$\theta = arg\min_{\theta} \mathbb{E}_{(x_i,y_i)\&(x_j,y_j)\in D}[L(x_i,y_i) + L(x_j,y_j) - \alpha \frac{\partial L(x_i,y_i)}{\partial \theta} \cdot \frac{\partial L(x_j,y_j)}{\partial \theta}], \quad (4)$$

where (x_i, y_i) and (x_j, y_j) are randomly sampled unique data points. In our work we will attempt to make a continual learning system that optimizes for this objective. However, there are multiple problems that must be addressed to implement this kind of learning process in practical settings. The first problem is that continual learning deals with learning over a continuous non-stationary stream of data. We address this by following past work and implementing an experience replay module that augments online learning so that we can approximately optimize over the stationary distribution of all examples seen so far.

Algorithm 1 Meta-Experience Replay (MER)

procedure TRAIN($D, \theta, \alpha, \beta, \gamma, k$)
  $M \leftarrow \{\}$
  for $t = 1, ..., T$ **do**
    for (x, y) `in` D_t **do**
      // Draw batches from buffer:
      $B_1, ..., B_s \leftarrow sampleBatches(x, k, M)$
      $\theta_0^A \leftarrow \theta$
      for $i = 1, ..., s$ **do**
        $\theta_{i0}^W \leftarrow \theta$
        for $j = 1, ..., k$ **do**
          $x_c, y_c \leftarrow B_i[j]$
          $\theta_{ij}^W \leftarrow SGD(x_c, y_c, \theta_{j-1}^W, \alpha)$
        end for
        // Within batch Reptile meta-update:
        $\theta \leftarrow \theta_{i0}^W + \beta(\theta_{ik}^W - \theta_{i0}^W)$
        $\theta_i^A \leftarrow \theta$
      end for
      // Across batch Reptile meta-update:
      $\theta \leftarrow \theta_0^A + \gamma(\theta_s^A - \theta_0^A)$
      // Update memory with reservoir sampling:
      $M \leftarrow M \cup \{(x, y, t)\}$
    end for
  end for
  return θ, M
end procedure

Another practical problem is that the gradients of this loss depend on the second derivative with respect to the loss function, which is expensive to compute. We address this issue by indirectly approximating this objective to a first order Taylor series expansion using an online meta-learning algorithm with minimal computational overhead.

3.2 EXPERIENCE REPLAY

As mentioned above, the continual lifelong learning setting poses a challenge for the optimization of neural networks as examples come one by one in a non-stationary stream. We would like to instead have our neural network optimize over the stationary distribution of all example seen so far. Experience replay (Lin, 1992; Murre, 1992) is an old technique that remains a central component of deep learning systems attempting to learn in non-stationary settings. In this paper we adopt conventions from recent work (Zhang & Sutton, 2017; Riemer et al., 2017) leveraging this approach. The central feature of experience replays is keeping a memory of examples seen M that is interleaved with the training of the current example with the goal of making the neural network training more stable. As a result experience replay approximates the objective in equation 3 to the extent that M approximates D:

$$\theta = arg\min_{\theta} \mathbb{E}_{(x,y)\in M}[L(x,y)], \quad (5)$$

where M has a current size M_{size} and maximum size M_{max}. In our work, we update the buffer with reservoir sampling. This ensures that at every timestep the probability that any of the N examples

seen has of being in the buffer is equal to M_{size}/N. This implies that the contents of the buffer resembles a stationary distribution over all examples seen to the extent that the items stored captures the variation of past examples. Following the standard practice in offline learning, we train by randomly sampling a batch B of experiences from the stored distribution. However, we differ in that the current example has a special role ensuring that it is always interleaved with the examples sampled from the replay buffer.

Before we proceed to the next example, we want to make sure we provide our algorithm with the ability to optimize for the current example (particularly if it is not added to the memory). Over N examples seen, this still implies that we have trained with each example as the current example with probability per step of $1/N$. We provide a detailed algorithm further explaining how experience replay is used in this work in the appendix. Obviously, it is not scalable to store every experience seen in memory. As such, in this work we focus on showing that we can achieve greater performance than baseline techniques when each approach is provided with only a small memory buffer.

3.3 Meta-Learning as a bias for Experience Replay

One of the most popular meta-learning algorithms to date is Model Agnostic Meta-Learning (MAML) (Finn et al., 2017). MAML requires training and validation splits from a set of tasks to perform meta-training, optimizing the network to find a set of initial parameters θ_0 that can easily adapt to a new unseen task when trained on training data from the task. This is done by optimizing the initial parameters with respect to its generalization to the task validation data after training has completed on the training data. One aspect of MAML that limits its scalability is the need to explicitly compute second derivatives. The authors proposed a variant called first-order MAML (FOMAML), which is defined by ignoring the second derivative terms to address this issue and surprisingly found that it achieved very similar performance. Recently, this phenomenon was explained by (Nichol & Schulman, 2018) who noted through Taylor expansion that the two algorithms were approximately optimizing for the same loss function. Nichol & Schulman (2018) also proposed an algorithm, Reptile, that efficiently optimizes for approximately the same objective while not requiring that the data be split into training and testing splits for each task learned. Reptile is implemented by optimizing across k batches of data sequentially with an SGD based optimizer and learning rate α. After training on these batches, we take the initial parameters before training θ_0 and update them to $\theta_0 \leftarrow \theta_0 + \beta * (\theta_k - \theta_0)$ where β is the learning rate for the meta-learning update. The process repeats for each series of k batches. As shown in terms of gradients in (Nichol & Schulman, 2018), Reptile approximately optimizes for the following objective within a set of k batches:

$$\theta = arg\min_{\theta} \mathbb{E}_{B_1,...,B_k \in D}[2\sum_{i=1}^{k} L(B_i) - \sum_{i=1}^{k}\sum_{j=1}^{i-1} \alpha \frac{\partial L(B_i)}{\partial \theta} \cdot \frac{\partial L(B_j)}{\partial \theta}], \tag{6}$$

where $B_1, ..., B_k$ are batches within the total dataset. This objective is quite appealing and similar to the objective in equation 4 to the extent that gradients produced on these batches approximate samples from the stationary distribution of all examples seen in the past. In this work, we modify the Reptile algorithm to properly integrate it with an experience replay module, facilitating continual learning while maximizing transfer and minimizing interference. We describe in more detail during the derivation in the appendix that achieving the Reptile objective in an online setting where examples are provided sequentially is non-trivial and is in part only achievable because of our random sampling strategies for both the buffer and batch. Following our remarks about experience replay from the prior section, this allows us to optimize for the following objective in a continual learning setting using our proposed Meta-Experience Replay (MER) algorithm:

$$\theta = arg\min_{\theta} \mathbb{E}_{(x_1,y_1),...,(x_k,y_k) \in M}[2\sum_{i=1}^{k} L(x_i, y_i) - \sum_{i=1}^{k}\sum_{j=1}^{i-1} \alpha \frac{\partial L(x_i, y_i)}{\partial \theta} \cdot \frac{\partial L(x_j, y_j)}{\partial \theta}]. \tag{7}$$

This can be achieved by using MER which maintains an experience replay memory M with reservoir sampling and at each time step draws $k-1$ random samples from the buffer to be trained alongside the current example. Each of the k examples is treated as its own Reptile batch of size 1. In light of our ideal objective in equation 4, we can see that using a batch size of 1 has an advantage over larger

batches because it allows for the second derivative information conveyed to the algorithm to be fine grained on the example level. Another reason it makes sense to use sample level effective batches is because for a given number of samples drawn from the buffer we maximize k. In equation 7 the typical offline learning loss has a weighting proportional to k and the regularizer term to maximize transfer and minimize interference has a weighting proportional to $\alpha k(k-1)/2$. This implies that by maximizing k we can put more weight on the regularization term. We demonstrate in our experiments that for a fixed number of examples drawn from the buffer, we consistently achieve better performance by converting it to a long list of individual samples for one update than we do by using proper batches as in the original Reptile paper targeting few shot learning.

In the final piece of the MER algorithm we samples s separate batches from the buffer M that are processed sequentially and each interleaved with the current example we then apply the Reptile algorithm again in an outer loop across the s batches. In the appendix we demonstrate that the terms differentiating this strategy from just constructing one big batch consisting of the same examples are small enough that they approximate the same objective. However, it is an effective means of prioritizing the current example further when scaling to larger effective k values. We provide further details for MER in algorithm 1. As we explain in the appendix, we always set $\beta = 1$ to ensure we approximate the desired objective function. It is important to note that while we use some of the machinery of past work on meta-learning, the functional regularization on learning is totally different. Meta-learning applied to few shot learning aims to facilitate fast learning and large movements through the parameter space when adapting initial parameters to a new task by learning to encourage gradient agreement across minibatches. In contrast, for continual learning we are regularizing the objective to encourage the maximization of knowledge transfer and the minimization of interference between the learning of any two examples.

4 Empirical Evaluation for Supervised Continual Lifelong Learning

To test the efficacy of our Meta-Experience Replay (MER) approach we compare it to relevant baselines for continual learning of many supervised tasks (see the appendix for in depth descriptions):

- **Online:** represents online learning performance of a model trained straightforwardly one example at a time on the incoming non-stationary training data by simply applying SGD.
- **Independent:**[1] is the performance of an independent predictor per task that can be possibly initialized by cloning the last trained predictor if it leads to improved performance.
- **Task Input:**[1] has the same architecture as Online, but with a dedicated input layer per task.
- **EWC:** Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) is an algorithm that modifies online learning where the loss is regularized to avoid catastrophic forgetting.
- **GEM:** Gradient Episodic Memory (GEM) (Lopez-Paz & Ranzato, 2017) is an approach for making efficient use of episodic storage by following gradients on incoming examples to the maximum extent while altering them so that they do not interfere with past memories.

In our experiments we follow (Lopez-Paz & Ranzato, 2017) and consider final retained accuracy across all tasks after training sequentially on all tasks as our main metric for comparing approaches. Moving forward we will refer to this metric as *retention*.

Question 1 *How does MER perform on supervised continual learning benchmarks?*

To address this question we consider two continual learning benchmarks from (Lopez-Paz & Ranzato, 2017). **MNIST Permutations** is a variant of MNIST first proposed in (Kirkpatrick et al., 2017) where each task is transformed by a fixed permutation of the MNIST pixels. As such, the input distribution of each task is unrelated. **MNIST Rotations** is another variant of MNIST proposed in (Lopez-Paz & Ranzato, 2017) where each task contains digits rotated by a fixed angle between 0 and 180 degrees. We follow the standard benchmark setting from (Lopez-Paz & Ranzato, 2017) using a modest memory buffer of size 5120 to learn 1000 sampled examples across each of 20 tasks.

[1] originally reported in (Lopez-Paz & Ranzato, 2017)

Model	MNIST Rotations	MNIST Permutations
Online	47.5%	55.9%
Independent	62.4%	45.2%
Task Input	75.9%	75.6%
EwC	59.7%	63.9%
GEM	86.7%	82.7%
MER	**89.5%**	**85.6%**

Table 1: Final retention performance on continual lifelong learning 20 tasks benchmarks from (Lopez-Paz & Ranzato, 2017).

Model	Buffer Size	MNIST Rotations	MNIST Permutations
GEM	5120	86.7%	82.7%
	500	73.9%	69.3%
	200	65.2%	56.3%
MER	5120	**89.5%**	**85.6%**
	500	**82.1%**	**77.8%**
	200	**77.2%**	**73.3%**

Table 2: Final retention performance as a function of the allocated buffer size limit on continual lifelong learning 20 tasks benchmarks from (Lopez-Paz & Ranzato, 2017).

Following conventions, we use a two layer MLP architecture for all models with 100 hidden units in each layer. We also model our hyperparameter search after (Lopez-Paz & Ranzato, 2017). We provide detailed information about the parameters we searched for and picked in the appendix.

In Table 1 we report results on these benchmarks in comparison to our baseline approaches. Clearly GEM outperforms our other baselines, but our approach adds significant value over GEM in terms of final retention accuracy on both benchmarks. EWC and using a task specific input layer both also lead to significant gains over standard online learning. However, they are quite far below the performance of approaches that make usage of episodic storage. Although EWC does not directly store any examples, we should note that in storing the Fisher information for each task it accrues more incremental resources than any of the episodic storage approaches.

Question 2 *How do the performance gains from MER vary as a function of the buffer size?*

To really make progress towards the greater goals of lifelong learning, we would like our algorithm to make the most use of even a modest buffer. This is because in extremely large scale settings it is unrealistic to assume a system can feasibly store a large percentage of previous examples in memory. As such, we would like to compare MER to GEM, which is known to perform well in this setting, in terms of how it can perform with an extremely small memory buffer. We consider a buffer size of 500, that is over 10 times smaller than the standard setting on these benchmarks. Additionally, we also consider a buffer size of 200, matching the smallest setting explored in (Lopez-Paz & Ranzato, 2017). For these tasks this setting corresponds to an average storage of 1 example for each combination of task and class. We report results for this setting in Table 2. We can see that the benefits of using MER over GEM grow as the buffer size becomes smaller. In the smallest setting, MER is providing more than a 10% boost in retention performance on both benchmarks.

Question 3 *How effective is MER at dealing with increasingly non-stationary settings?*

Another larger goal of lifelong learning is to enable continual learning with only relatively few examples per task. This setting is particularly difficult because we have less data to characterize each class to learn from and our distribution is increasingly non-stationary over a fixed amount of training. We would like to explore how various models perform in this kind of setting. To do this we consider two new benchmarks. **Many Permutations** is a variant of *MNIST Permutations* that has 5 times more tasks (100 total) and 5 times less training examples per task (200 each). Meanwhile

Model	Buffer Size	Many Permutations	Omniglot
Online	0	31.9%	4.9%
EWC	0	33.2%	5.2%
GEM	5120	57.3%	18.4%
	500	32.5%	-
MER	5120	**61.9%**	**76.0%**
	500	**47.3%**	**33.0%**

Table 3: Final retention performance on many task non-stationary continual lifelong learning benchmarks.

in **Omniglot** we also explore the Omniglot (Lake et al., 2011) benchmark treating each of the 50 alphabets to be a task. Following multi-task task conventions 90% of the data is used for training and 10% is used for testing. Overall there are 1623 different character classes. We learn each class and task sequentially. We follow (Vinyals et al., 2016) in scaling down the images to 28x28. Moreover, we also use a CNN that consists of a stack of 4 modules before a fully connected softmax layer where each module includes a 3x3 convolution with 64 filters, a Relu non-linearity and 2x2 max-pooling.

We report continual learning results using these new datasets in Table 3. We can see in the case of the *Many Permutations* dataset that the effect of efficiently using episodic storage becomes even more pronounced when the setting becomes more non-stationary. GEM and MER both achieve nearly double the performance of EWC and online learning. We also see that increasingly non-stationary settings lead to a larger performance gain for MER over GEM. Gains are quite significant for *Many Permutations* and remarkable for *Omniglot*. *Omniglot* is even more non-stationary including slightly fewer examples per task and MER nearly quadruples the performance of baseline techniques. Considering the poor performance of online learning and EWC it is natural to question whether or not examples were learned in the first place. We experiment with using as many as 100 gradient descent steps per incoming example to ensure each example is learned when first seen. However, do to the extremely non-stationary setting no variant we tried surpassed 5.5% retained accuracy. GEM also has major deficits for learning on Omniglot that are resolved by MER. GEM maintains a buffer using a recent item based sampling strategy and thus can not deal with non-stationarity within the task nearly as well as reservoir sampling. Additionally, we found that the optimization based on the buffer was significantly less effective and less reliable as the quadratic program fails for many hyperparameter values that lead to non-positive definite matrices. Unfortunately, we could not get GEM to consistently converge on Omniglot for a memory size of 500, meanwhile we see MER handles it well. MER greatly outperforms GEM with an order of magnitude smaller buffer.

5 Empirical Evaluation for Continual Reinforcement Learning

Question 4 *Can MER improve a DQN with experience replay in continual reinforcement learning settings?*

We considered the evaluation of MER in a continual reinforcement learning setting where the environment is highly non-stationary. In order to produce these non-stationary environments in a controlled way suitable for our experimental purposes, we utilized different arcade games provided by Tasfi (2016). In our experiments we used Catcher and FlappyBird, two simple but interesting enough environments. For the purposes of our explanation, we will call each set of fixed game-dependent parameters a task [2]. The multi-task setting is then built by introducing changes in these parameters, resulting in non-stationarity across tasks.[3] As in our other experiments, each agent is evaluated based on its retention of all tasks. Therefore, the reported performances provide a measurement of adaptation to the current task as well as of retention of knowledge acquired from other

[2]Our agents function in a standard setting with no provided task information, forcing it to identify these switches in game play on its own. This procludes standard usage of baselines like GEM and EWC.

[3]Note that while these changes in these game-dependent parameters that alter the behavior of the game results in a strong non-stationarity, switching effectively from one task to the next keeps the same many other elements of the game, such as the overall appearance. It is in this sense that our setting differs significantly from playing different levels in multi-level games.

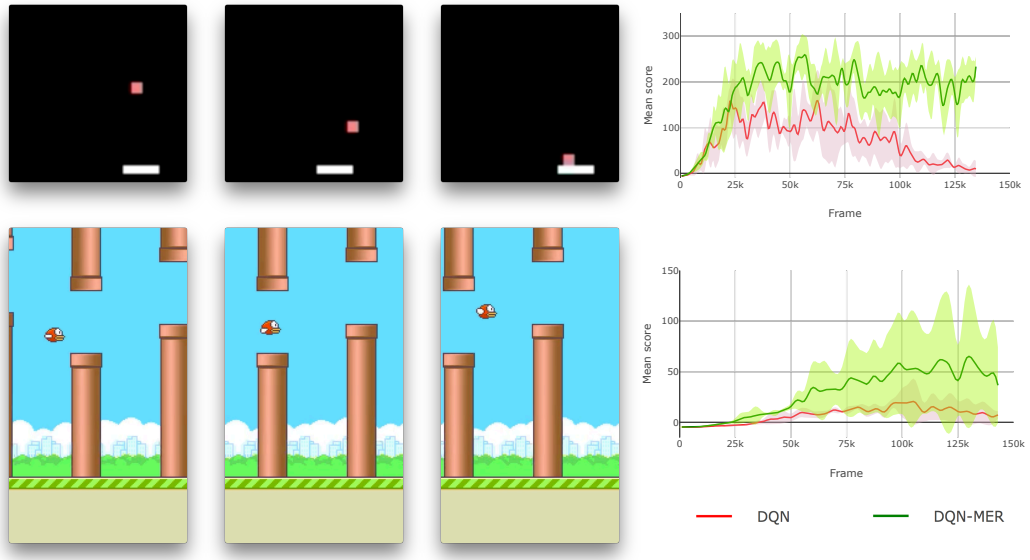

Figure 2: *Right, above and below*: sequence of frames for the game Catcher and Flappy Bird respectively. Non-stationarity is introduced in Catcher by changing the velocity with which the fruit falls from the top of the screen to the bottom. In Flappy Bird, the changing parameter is the gap between upper and lower pipes, which is decreased when the task switches. *Left, above and below*: average score in Catcher and Flappy Bird respectively for the evaluation on the first task, corresponding to slower fruit velocity in Catcher and larger pipe gap in Flappy Bird.

tasks. Our model uses a standard DQN setting originally developed for Atari (Mnih et al., 2015): a CNN that consist of a stack of 3 layers, the first with 32 filters with kernel 8 by 8 and stride 4, a second layer with 64 filters with kernel 4 by 4 and stride 2, and a final layer with 64 filters with kernel 3 by 3 and stride 1, followed by two fully connected layers. A ReLU non-linearity was applied after all the hidden layers. We limited the memory buffer size for our models to 50,000 transitions, which is roughly the same proportion of the total memories used in the benchmark setting four our supervised learning tasks.

We illustrate the continual reinforcement learning scenario with these two games. In Catcher, the agent controls a segment that lies horizontally in the bottom of the screen, i.e. a basket, and can move right or left, or stay still. The goal is to move the basket to catch as many pellets, or fruits, as possible. Missing a fruit results in losing one of the three available lives. Fruits emerge one by one from the top of the screen, and have a descending velocity that is fixed for each task. In this game, the different tasks are then obtained by incrementally increasing the fruit velocity a total of 6 times during training. In the case of the very popular game FlappyBird, the agent has to navigate a bird in an environment full of pipes by deciding whether to flap or not flap its wings. The pipes appear always in pairs, one from the bottom of the screen and one from the top of the screen, and have a gap that allows the bird to pass through them. Flapping the wings results in the bird ascending, otherwise the bird descends to ground naturally. Both ascending and descending velocities are presets by the physics engine of the game. The goal is to pass through many pairs of pipes as possible without hitting a pipe, as this results in losing the game. The scoring scheme in this game awards a point each time a pipe is crossed.[4] In our setting the different tasks are obtained by incrementally reducing the separation between upper and lower pipes a total of 6 times during training.

In Figure 3, we show the performance in Catcher when trained sequentially on 6 different tasks for 25,000 frames each to a maximum of 150k frames, evaluated at each point in time in all 6 tasks. Under these non-stationary conditions, a DQN using MER performs consistently better than the

[4]Although with very simple mechanics, FlappyBird has proven to be somehow challenging for many humans. According to the original game scoring scheme, players with a score of 10 receive a Bronze medal; with 20 points, a Silver medal; 30 results in a Gold medal, and any score better than 40 is rewarded with a Platinum medal.

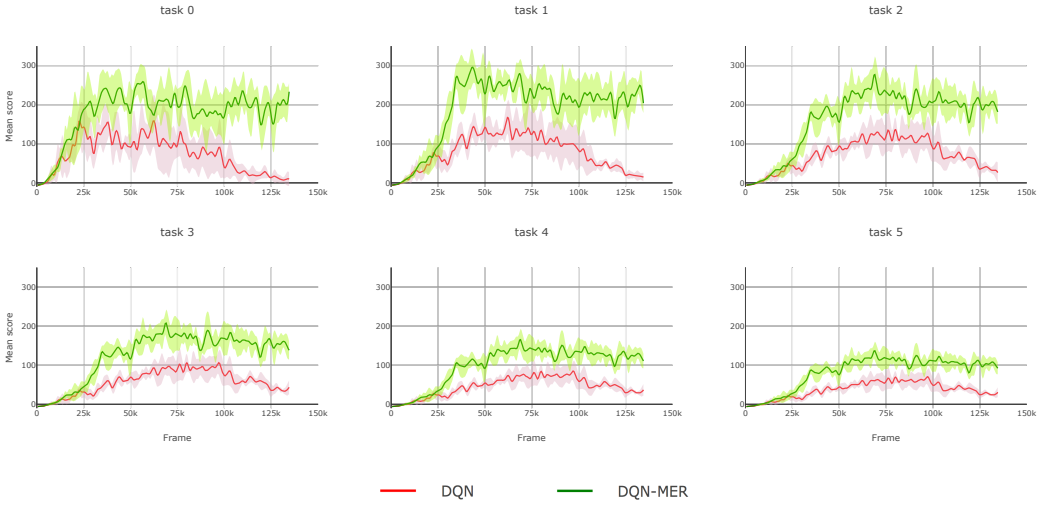

Figure 3: Performance during training in continual learning for a non-stationary version of Catcher. Graphs show averaged values over ten validation episodes across five different seed initializations. Vertical grid lines on the frame axis indicate task switch.

regular DQN version that utilizes an experience replay buffer. Figure (3.1) shows the average score for the first task, If we take as inspiration how humans perform, in the last stages of training we hope that a player that obtains good results in later tasks will also obtain good results in the first tasks, insofar the first tasks are subsumed in the latter ones. For example, in Catcher, later tasks have the fruit move faster, and thus we expect to be able to do well in the first task. However, DQN forgets significantly to perform the first task, i.e., get slowly moving fruits. In marked contrast, DQN-MER exhibits minimal or no forgetting for the first set of episodes after being trained in the rest of the tasks. This behavior is intuitive in the sense previously exposed that we would expect forward transfer to happen naturally in this setting, as it seems to be the case with human players. Similar behavior appears in Flappy Bird. DQN-MER becomes a Platinum player for the first task environment during the period in which is learning the third task, a more difficult environment in which the pipe gap is noticeably smaller than in the first task (see Appendix E.2). Then, DQN-MER keeps this level forward for the first and second task environments.

6 Further Analysis of the Approach

In this section we would like to dive deeper into how MER works. To do so we run additional detailed experiments across our three MNIST based continual learning benchmarks.

Question 5 *Does MER lead to a shift in the distribution of gradient dot products?*

We would like to directly verify that MER achieves our motivation in equation 7 and results in significant changes in the distribution of gradient dot products between new incoming examples and past examples over the course of learning when compared to experience replay (ER). For these experiments, we maintain a history of all examples seen that is totally separate from our notion of memory buffers that only include a partial history of examples. Every time we receive a new example we use the current model to extract a gradient direction and we also randomly sample five examples from the previous history. We save the dot products of the incoming example gradient with these five past example gradients and consider the mean of the distribution of dot products seen over the course of learning for each model. We run this experiment on the best hyperparamater setting for both our ER model and our MER model with one batch per example for fair comparison. Each model is evaluated five times over the course of learning. We report mean and standard deviations

Model	MNIST Permutations	MNIST Rotations	Many Permutations
ER	-0.569 ($\pm$ 0.077)	-1.652 ($\pm$ 0.082)	-1.280 ($\pm$ 0.078)
MER	+0.042 ($\pm$ 0.017)	+0.017 ($\pm$ 0.007)	+0.131 ($\pm$ 0.027)

Table 4: Analysis of the mean dot product across the period of learning between gradients on incoming examples and gradients on randomly sampled past examples across 5 runs on MNIST based benchmarks.

of the mean gradient dot product across runs in Table 4. We can thus verify that a very significant and reproducible difference in the mean gradient encountered is seen for MER in comparison to ER alone. This difference alters the learning process making incoming examples on average result in slight transfer rather than significant interference. This analysis confirms the desired effect of the objective function in equation 7.

Question 6 *What components of MER are most important?*

We would like to further analyze our proposed MER model to understand what components add the most value and when. More specifically we would like to understand how powerful our variant of experience replay described in section 3.2 is on its own and how much is added by adding meta-learning to experience replay both within a batch and across multiple batches per example. In the appendix we provide detailed results considering ablated baselines for our experiments on the MNIST lifelong learning benchmarks. We consistently see that our version of experience replay provides gains over GEM on its own. Clearly the value of optimizing over an approximation of the stationary distribution is hard to beat in these problem settings. However, we see that both performing meta-learning within the batch and across multiple batches per example consistently create additional value in our experiments. Meta-learning appears to provide increasing value as the buffer size becomes smaller.

7 Conclusion

In this paper we have cast a new perspective on the problem of continual learning in terms of a fundamental trade-off between transfer and interference. Exploiting this perspective, we have in turn developed a new algorithm Meta-Experience Replay (MER) that is well suited for application to general purpose continual learning problems. We have demonstrated that MER regularizes the objective of experience replay so that gradients on incoming examples are more likely to have transfer and less likely to have interference with respect to past examples. The result is a general purpose solution to continual learning problems that outperforms strong baselines for both supervised continual learning benchmarks and continual learning in non-stationary reinforcement learning environments. In this paper we argue that techniques for continual learning have been largely driven by different conceptualizations of the fundamental problem encountered by neural networks and as such we hope that the transfer-interference trade-off can be a useful problem view for future work to exploit with MER as a first successful example.

References

Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1): 54–115, 1987.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Robert M French. Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. 1991.

Benjamin Frederick Goodrich. Neuron clustering for mitigating catastrophic forgetting in supervised and reinforcement learning. 2015.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, pp. 201611835, 2017.

Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Cognitive Science Society*, volume 33, 2011.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*, pp. 4652–4662, 2017.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continuum learning. *NIPS*, 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Jacob MJ Murre. Learning and categorization in modular neural networks. 1992.

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2018.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. icarl: Incremental classifier and representation learning. *CVPR*, 2017.

Matthew Riemer, Tim Klinger, Michele Franceschini, and Djallel Bouneffouf. Scalable recollections for continual lifelong learning. *arXiv preprint arXiv:1711.06761*, 2017.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *ICLR*, 2018.

Norman Tasfi. Pygame learning environment. `https://github.com/ntasfi/PyGame-Learning-Environment`, 2016.

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, pp. 640–646, 1996.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995, 2017.

Shangtong Zhang and Richard S Sutton. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.

A Further Descriptions and Comparisons with Baseline Algorithms

Independent: originally reported in (Lopez-Paz & Ranzato, 2017) is the performance of an independent predictor per task which has the same architecture but with less hidden units proportional to the number of tasks. The independent predictor can be initialized randomly or clone the last trained predictor depending on what leads to better performance.

EWC: Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) is an algorithm that modifies online learning where the loss is regularized to avoid catastrophic forgetting by considering the importance of parameters in the model as measured by their fisher information. EWC follows the catastrophic forgetting view of the continual learning problem by promoting less sharing of parameters for *new learning* that were deemed to be important for performance on *old memories*. We utilize the code provided by (Lopez-Paz & Ranzato, 2017) in our experiments. The only difference in our setting is that we provide the model one example at a time to test true continual learning rather than providing a batch of 10 examples at a time.

GEM: Gradient Episodic Memory (GEM) (Lopez-Paz & Ranzato, 2017) is an algorithm meant to enhance the effectiveness of episodic storage based continual learning techniques by allowing the model to adapt to incoming examples using SGD as long as the gradients do not interfere with examples from each task stored in a memory buffer. If gradients interfere leading to a decrease in the performance of a past task, a quadratic program is used to solve for the closest gradient to the original that does not have negative gradient dot products with the aggregate memories from any previous tasks. GEM is known to achieve superior performance in comparison to other recently proposed techniques that use episodic storage like (Rebuffi et al., 2017), making superior use of small memory buffer sizes. GEM follows similar motivation to our approach in that it also considers the intelligent use of gradient dot product information to improve the use case of supervised continual learning. As a result, it a very strong and interesting baseline to compare with our approach. We modify the original code and benchmarks provided by (Lopez-Paz & Ranzato, 2017). Once again the only difference in our setting is that we provide the model one example at a time to test true continual learning rather than providing a batch of 10 examples at a time.

We can consider the GEM algorithm as tailored to the stability-plasticity dilemma conceptualization of continual learning in that it looks to preserve performance on past tasks while allowing for maximal plasticity to the new task. To achieve this, GEM (Lopez-Paz & Ranzato, 2017) solves a quadratic program to find an approximate gradient g_{new} that closely matches $\frac{\partial L(x_{new}, y_{new})}{\partial \theta}$ while ensuring that the following constraint holds:

$$g_{new} \cdot \frac{\partial L(x_{old}, y_{old})}{\partial \theta} > 0. \tag{8}$$

B Deriving the Effective Objective of MER

We would like to derive what objective our proposed Meta-Experience Replay algorithm (1) approximates. We follow conventions from (Nichol & Schulman, 2018) and first demonstrate what happens to the effective gradients computed by the algorithm in the most trivial case. As in (Nichol & Schulman, 2018), this allows us to extrapolate an effective gradient that is a function of the number of steps taken. We can then consider the effective loss function that results in this gradient. Before we begin, let us define the following terms from (Nichol & Schulman, 2018):

$$g_i = \frac{\partial L(\theta_i)}{\partial \theta_i} \text{ (gradient obtained during SGD)} \tag{9}$$

$$\theta_{i+1} = \theta_i - \alpha g_i \text{ (sequence of parameter vectors)} \tag{10}$$

$$\bar{g}_i = \frac{\partial L(\theta_i)}{\partial \theta_0} \text{ (gradient at initial point)} \tag{11}$$

$$g_i^j = \frac{\partial L(\theta_i)}{\partial \theta_j} \text{ (gradient evaluated at point i with respect to parameters j)} \tag{12}$$

$$\bar{H}_i = \frac{\partial^2 L(\theta_i)}{\partial \theta_0^2} \text{ (Hessian at initial point)} \tag{13}$$

$$H_i^j = \frac{\partial^2 L(\theta_i)}{\partial \theta_j^2} \text{ (Hessian evaluated at point i with respect to parameters j)} \tag{14}$$

In (Nichol & Schulman, 2018) they consider the effective gradient across one loop of reptile with size $k = 2$. As we have both an outer loop of Reptile applied across batches and an inner loop applied within the batch, we start with a setting where the number of batches $s = 2$ and the number of examples per batch $k = 2$. Let's recall from the original paper that the gradients of Reptile with $k = 2$ was:

$$g_{Reptile,k=2,s=1} = g_0 + g_1 = \bar{g}_0 + \bar{g}_1 - \alpha \bar{H}_1 \bar{g}_0 + O(\alpha^2) \tag{15}$$

So, we can also consider the gradients of Reptile if we had 4 examples in one big batch as opposed to 2 batches of 2 examples:

$$\begin{aligned} g_{Reptile,k=4,s=1} &= g_0 + g_1 + g_2 + g_3 \\ = \bar{g}_0 + \bar{g}_1 + \bar{g}_2 + \bar{g}_3 - \alpha \bar{H}_1 \bar{g}_0 - \alpha \bar{H}_2 \bar{g}_0 - \alpha \bar{H}_2 \bar{g}_1 - \alpha \bar{H}_3 \bar{g}_0 - \alpha \bar{H}_3 \bar{g}_1 - \alpha \bar{H}_3 \bar{g}_2 + O(\alpha^2) \end{aligned} \tag{16}$$

Now we can consider the case for MER where we define the parameter values as follows extending algorithm 1:

$$\theta_0 = \theta_0^A = \theta_{00}^W \tag{17}$$

$$\theta_{01}^W = \theta_{00}^W - \alpha g_{00} \tag{18}$$

$$\theta_{02}^W = \theta_{01}^W - \alpha g_{01} \tag{19}$$

$$\theta_1^A = \theta_0^A + \beta \frac{(\theta_{02}^W - \theta_0^A)}{\alpha} = \theta_0 + \beta \frac{(\theta_{02}^W - \theta_0)}{\alpha} = \theta_{10}^W \tag{20}$$

$$\theta_{11}^W = \theta_{10}^W - \alpha g_{10} \tag{21}$$

$$\theta_{12}^W = \theta_{11}^W - \alpha g_{11} \tag{22}$$

$$\theta_2^A = \theta_1^A + \beta \frac{(\theta_{12}^W - \theta_1^A)}{\alpha} \tag{23}$$

$$\theta = \theta_0^A + \gamma \beta \frac{(\theta_2^A - \theta_0^A)}{\beta} = \theta_0^A + \gamma(\theta_2^A - \theta_0^A) \tag{24}$$

g_{MER} the gradient of Meta-Experience Replay can thus be defined analogously to the gradient of Reptile as:

$$g_{MER} = \frac{\theta_0^A - \theta_2^A}{\beta} = \frac{\theta_0 - \theta_2^A}{\beta} \tag{25}$$

By simply applying Reptile from equation 15 we can derive the value of the parameters θ_1^A after updating with Reptile within the first batch in terms of the original parameters θ_0:

$$\theta_1^A = \theta_0 - \beta \bar{g}_{00} - \beta \bar{g}_{01} + \beta \alpha \bar{H}_{01} \bar{g}_{00} + O(\beta \alpha^2) \tag{26}$$

By subbing equation 26 into equation 23 we can see that:

$$\theta_2^A = \theta_0 - \beta\bar{g}_{00} - \beta\bar{g}_{01} + \beta\alpha\bar{H}_{01}\bar{g}_{00} - \beta g_{10} - \beta g_{11} + O(\beta\alpha^2) \tag{27}$$

We can express g_{10} in terms of the initial point, by considering a Taylor expansion following the Reptile paper:

$$g_{10} = \bar{g}_{10} + \alpha\bar{H}_{10}(\theta_{10}^W - \theta_0) + O(\alpha^2) \tag{28}$$

Then substituting in for θ_{10}^W we express g_{10} in terms of θ_0 :

$$g_{10} = \bar{g}_{10} - \alpha\beta\bar{H}_{10}\bar{g}_{00} - \alpha\beta\bar{H}_{10}\bar{g}_{01} + O(\alpha^2) \tag{29}$$

We can then rewrite g_{11} by taking a Taylor expansions with respect to θ_{10}^W:

$$g_{11} = g_{11}^{10} - \alpha H_{11}^{10} g_{10} + O(\alpha^2) \tag{30}$$

Taking another Taylor expansion we find that we can transform our expression for the Hessian:

$$H_{11}^{10} = \bar{H}_{11} + O(\alpha) \tag{31}$$

We can analogously also transform our expression our expression for g_{11}^{10}:

$$g_{11}^{10} = \bar{g}_{11} + \alpha\bar{H}_{11}(\theta_{10}^W - \theta_0) + O(\alpha^2) \tag{32}$$

Substituting for θ_{10}^W in terms of θ_0

$$g_{11}^{10} = \bar{g}_{11} - \alpha\beta\bar{H}_{11}\bar{g}_{00} - \alpha\beta\bar{H}_{11}\bar{g}_{01} + O(\alpha^2) \tag{33}$$

We then substitute equation 31, equation 33, and equation 29 into equation 34:

$$g_{11} = \bar{g}_{11} - \alpha\beta\bar{H}_{11}\bar{g}_{00} - \alpha\beta\bar{H}_{11}\bar{g}_{01} - \alpha\bar{H}_{11}\bar{g}_{10} + O(\alpha^2) \tag{34}$$

Finally, we have all of the terms we need to express θ_2^A and we can then derive an expression for the MER gradient g_{MER}:

$$\begin{aligned} g_{MER} = \bar{g}_{00} + \bar{g}_{01} + \bar{g}_{10} + \bar{g}_{11} \\ -\alpha\bar{H}_{01}\bar{g}_{00} - \alpha\bar{H}_{11}\bar{g}_{10} - \alpha\beta\bar{H}_{10}\bar{g}_{00} - \alpha\beta\bar{H}_{10}\bar{g}_{01} - \alpha\beta\bar{H}_{11}\bar{g}_{00} - \alpha\beta\bar{H}_{11}\bar{g}_{01} + O(\alpha^2) \end{aligned} \tag{35}$$

This equation is quite interesting and very similar to equation 16. As we would like to approximate the same objective, we remove one hyperparameter from our model by setting $\beta = 1$. This yields:

$$\begin{aligned} g_{MER} = \bar{g}_{00} + \bar{g}_{01} + \bar{g}_{10} + \bar{g}_{11} \\ -\alpha\bar{H}_{01}\bar{g}_{00} - \alpha\bar{H}_{11}\bar{g}_{10} - \alpha\bar{H}_{10}\bar{g}_{00} - \alpha\bar{H}_{10}\bar{g}_{01} - \alpha\bar{H}_{11}\bar{g}_{00} - \alpha\bar{H}_{11}\bar{g}_{01} + O(\alpha^2) \end{aligned} \tag{36}$$

Indeed, with β set to equal 1, we have shown that the gradient of MER is the same as one loop of Reptile with a number of steps equal to the total number of examples in all batches of MER. The primary difference is that in MER we can effectively leverage a greater number of effective Reptile steps k while also shifting up the priority of the current example within the batch.

Now we would like to link the gradient being optimized for to the expected approximate objective function. It was noted in the Reptile paper that the following equality holds if the examples and order are random.

$$\mathbb{E}[\bar{H}_1\bar{g}_0] = \mathbb{E}[\bar{H}_0\bar{g}_1] = \frac{1}{2}\mathbb{E}[\frac{\partial}{\partial\theta_0}(\bar{g}_0 \cdot \bar{g}_1)] \tag{37}$$

In our work to make sure this equality holds in an online setting, we must take multiple precautions as noted in the main text. The issue is that examples are received in sequence so when applied in an online setting the order is not totally random or arbitrary as in the original Reptile work. We address by maintaining our buffer using reservoir sampling, which ensures that any example seen before has a probability $\frac{1}{N}$ of being a particular element in the buffer. We also randomly select over these elements to form a batch. As this makes the order largely arbitrary to the extent that our buffer includes all examples seen, we are approximating the random offline setting from the original Reptile paper. As such we can view the gradients in equation 16 and equation 36 as leading to approximately the following objective function:

$$\theta = arg\min_{\theta} \mathbb{E}_{(x_1,y_1),\ldots,(x_k,y_k)\in M}[2\sum_{i=1}^{k} L(x_i, y_i) - \sum_{i=1}^{k}\sum_{j=1}^{i-1} \alpha \frac{\partial L(x_i, y_i)}{\partial\theta} \cdot \frac{\partial L(x_j, y_j)}{\partial\theta}] \tag{38}$$

This is precisely equation 7 in the main text.

C Hyperparameter Search

Here we report the hyper-parameter grids that we searched over in our experiments. The best values for the MNIST Rotations (ROT) at each buffer size (ROT-5120, ROT-500, ROT-200), MNIST Permutations (PERM) at each buffer size (PERM-5120, PERM-500, PERM-200), Many Permutations (MANY) at each buffer size (MANY-5120, MANY-500), and Omniglot (OMNI) at each buffer size (OMNI-5120, OMNI-500) are noted accordingly in parenthesis.

- Online Learning
  - learning rate: [0.0001, 0.0003, 0.001 (PERM, ROT), 0.003 (MANY), 0.01, 0.03, 0.1 (OMNI)]
- EWC
  - learning rate: [0.001 (OMNI), 0.003 (MANY), 0.01, 0.03 (ROT, PERM), 0.1, 0.3, 1.0]
  - regularization: [1, 3 (MANY), 10 (OMNI), 30, 100, 300 (ROT, PERM), 1000, 3000, 10000, 30000]
- GEM
  - learning rate: [0.001, 0.003 (MANY-500), 0.01 (ROT, PERM, OMNI, MANY-5120), 0.03, 0.1, 0.3, 1.0]
  - memory strength (γ): [0.0 (PERM-500, MANY-500), 0.1 (PERM-200, MANY-200), 0.5 (OMNI), 1.0 (ROT-5120, ROT-500, ROT-200, PERM-5120)]
- Experience Replay
  - learning rate: [0.00003, 0.0001, 0.0003, 0.001, 0.003 (MANY), 0.01 (ROT, PERM), 0.03 (OMNI), 0.1]
  - batch size (k-1): [5, 10 (ROT-500, PERM-200), 25 (ROT-5120, PERM-5120, PERM-500), 50 (OMNI, MANY-5120, ROT-200), 100, 250]
- Meta-Experience Replay
  - learning rate (α): [0.01 (OMNI-5120), 0.03 (ROT, PERM, MANY), 0.1 (OMNI-500)]
  - within batch meta-learning rate (β): 1.0
  - across batch meta-learning rate (γ): [0.03 (ROT-200, PERM-5120, MANY-5120, MANY-500), 0.1 (ROT-5120, ROT-500, PERM-500, PERM-200), 0.3, 1.0 (OMNI)]
  - batch size (k-1): [5 (MANY-500, OMNI-500), 10, 25 (PERM-500, PERM-200, MANY-5120, OMNI-5120), 50 (ROT-5120, ROT-500, PERM-5120), 100 (ROT-200)]
  - number of batches per example: [1, 2 (ROT-500, PERM-500, PERM-200, OMNI-500), 5 (ROT-5120, ROT-200, OMNI-5120), 10 (PERM-5120, MANY-5120, MANY-500)]

Model	Buffer Size	Rotations	Permutations	Many Permutations
ER (Section 3.2)	5120	88.4	83.8	60.2
	500	77.6	75.0	45.1
	200	71.2	68.5	–
MER (One Batch)	5120	89.0	85.0	61.2
	500	80.3	76.1	45.1
	200	73.2	71.2	-
MER (Multiple Batches)	5120	**89.5**	**85.6**	**61.9**
	500	**82.1**	**77.8**	**47.3**
	200	**77.2**	**73.3**	-

Table 5: Ablation experiments on MNIST based learning lifelong learning benchmarks.

D ABLATION EXPERIMENTS

E CONTINUAL REINFORCEMENT LEARNING

E.1 PARAMETERS FOR CONTINUAL REINFORCEMENT LEARNING EXPERIMENTS

For the continual reinforcement learning setting we set the parameters using results from the experiments in the supervised setting as a guidance. Both Catcher and Flappy Bird used the same hyper parameters as detailed below with the obvious exception of the game-dependent parameter that defines each task. Models were trained with a maximum number of frames of 150k and 6 total tasks, switching every 25k frames. Runs used different random seeds for the initialization as stated in the figures.

- Game Parameters
  - Catcher: Δ: 0.03 (vertical velocity of pellet increased from default 0.608).
  - Flappy Bird: Δ: -10 (pipe gap decreased 10 from default 100).
- Experience Replay
  - learning rate: 0.0001
  - batch size (k-1): 16
- Meta-Experience Replay
  - learning rate (α): 0.0001
  - batch meta-learning rate (γ): 0.3
  - batch size (k-1): 16
  - number of batches per example: 1
  - buffer size: 50000

E.2 CONTINUAL REINFORCEMENT LEARNING EVALUATION FOR FLAPPY BIRD

Performance during training in continual learning for a non-stationary version of Flappy Bird is shown in Figure (4). Graphsshow averaged values over three validation episodes across three different seed initializations. Vertical grid lineson the frame axis indicate task switch

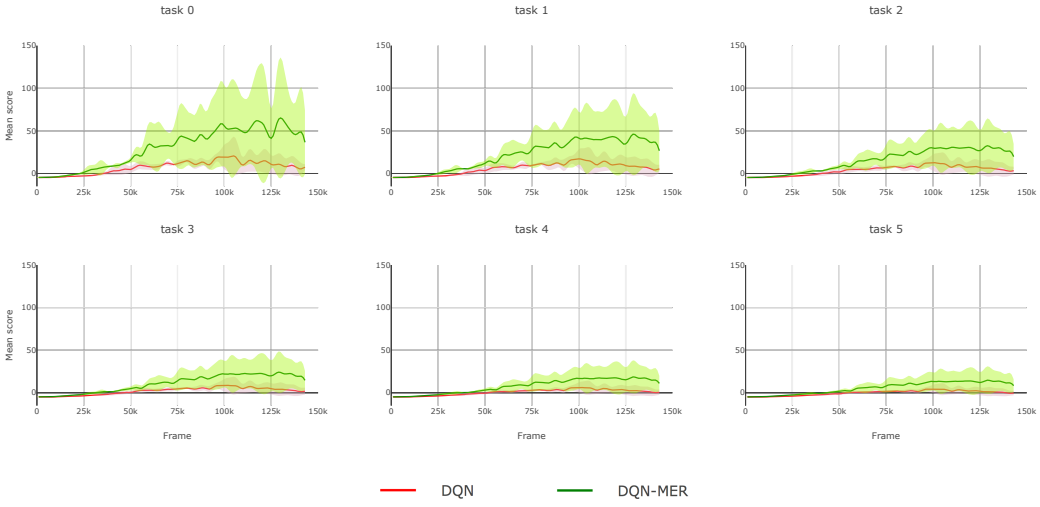

Figure 4: Continual learning for a non-stationary version of Flappy Bird.