La-MAML: Look-ahead Meta Learning for Continual Learning

Gunshi Gupta [*1] Karmesh Yadav [*2] Liam Paull [1]

Abstract

The continual learning problem involves training models with limited capacity to perform well on a set of an unknown number of sequentially arriving tasks. While meta-learning shows great potential for reducing interference between old and new tasks, the current training procedures tend to be either slow or offline, and sensitive to many hyper-parameters. In this work, we propose *Look-ahead MAML (La-MAML)*, a fast optimisation-based meta-learning algorithm for *online*-continual learning, aided by a small episodic memory. Our proposed modulation of per-parameter learning rates in our meta-learning update allows us to draw connections to prior work on *hypergradients* and *meta-descent*. This provides a more flexible and efficient way to mitigate *catastrophic forgetting* compared to conventional *prior-based* methods. *La-MAML* achieves performance superior to other replay-based, prior-based and meta-learning based approaches for continual learning on real-world visual classification benchmarks.

1. Introduction

Embodied or interactive agents that accumulate knowledge and skills over time must possess the ability to continually learn. *Catastrophic forgetting* (French, 1999; Mcclelland et al., 1995), one of the biggest challenges in this setup, can occur when the *i.i.d.* sampling conditions required by stochastic gradient descent (SGD) are violated as the data belonging to different tasks to be learnt arrives sequentially. Algorithms for *continual learning* (CL) must also use their limited model capacity efficiently since the number of future tasks is unknown. Ensuring gradient-alignment across tasks is therefore essential, to make shared progress on their objectives. *Gradient Episodic Memory* (GEM) (Lopez-Paz & Ranzato, 2017) investigated the connection between weight sharing and forgetting in CL and developed an algorithm that explicitly tried to minimise *gradient interference*. This is an objective that meta-learning algorithms implicitly optimise for (refer to (Nichol et al., 2018) for derivations of the effective parameter update made in first and second order meta learning algorithms). *Meta Experience Replay* (MER) (Riemer et al., 2019) formalized the transfer-interference trade-off and showed that the gradient alignment objective of GEM coincide with the objective optimised by the first order meta-learning algorithm Reptile (Nichol et al., 2018).

Besides aligning gradients, meta-learning algorithms show promise for CL since they can *directly* use the meta-objective to influence model optimisation and improve on auxiliary objectives like generalisation or transfer. This avoids having to define heuristic incentives like sparsity (Le et al., 2017) for better CL. The downside is that they are usually slow and hard to tune, effectively rendering them more suitable for *offline* continual learning (Javed & White, 2019; Riemer et al., 2019). In this work, we overcome these difficulties and develop a gradient-based meta-learning algorithm for *efficient, online* continual learning. We first propose a base algorithm for continual meta-learning referred to as Continual-MAML (C-MAML) that utilizes a replay-buffer and optimizes a meta-objective that mitigates forgetting. Subsequently, we propose a modification to C-MAML, La-MAML, which incorporates modulation of per-parameter learning rates (LRs) to pace the learning of a model across tasks and time. Finally, we show that the algorithm is scalable, robust and achieves favourable performance on several benchmarks of varying complexity.

2. Related work

Relevant CL approaches can be roughly categorized into *replay-based, regularisation-based (or prior-based)* and *meta-learning-based* approaches.

In order to circumvent the issue of catastrophic forgetting, *replay-based methods* maintain a collection of samples from previous tasks in memory. Approaches utilising an *episodic-buffer* (Castro et al., 2018; Rebuffi et al., 2017) uniformly sample old data points to mimic the *i.i.d.* setup within continual learning. Approaches proposing *Generative-replay* (Shin et al., 2017) train generative models to *replay* the data

[*]Equal contribution [1]Mila, Universite de Montreal [2]Robotics Institute, Carnegie Mellon University. Correspondence to: Gunshi Gupta <gunshigupta9@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

distribution of past samples. However, they currently face scalability issues arising from the difficulty of modeling complex non-stationary distributions. GEM (Lopez-Paz & Ranzato, 2017) and A-GEM (Chaudhry et al., 2019) take memory samples into account to determine altered *low-interference* gradients for updating parameters.

Regularisation-based methods avoid using replay at all by constraining the network weights according to heuristics intended to ensure that performance on previous tasks is preserved. This involves penalising changes to weights deemed important for old tasks (Kirkpatrick et al., 2017) or enforcing weight or representational sparsity (Aljundi et al., 2019) to ensure that only a subset of neurons remain active (and thus receiving gradients) at any point of time.

Meta-Learning-based approaches are fairly recent and have shown impressive results on small benchmarks like Omniglot and MNIST. MER (Riemer et al., 2019), inspired by GEM(Lopez-Paz & Ranzato, 2017), utilises replay to incentivise alignment of gradients between old and new tasks. Online-aware Meta Learning (OML) (Javed & White, 2019) introduces a meta-objective for a pre-training algorithm to learn an optimal representation *offline*, which is subsequently frozen and used for CL. The the orthogonal setup of *online learning* is investigated in (Al-Shedivat et al., 2018; Finn et al., 2019; Nagabandi et al., 2019), where a learning agent uses all previously seen data to adapt quickly to an incoming stream of data, thereby ignoring the problem of catastrophic forgetting. Our motivation lies in developing a *scalable*, *online* algorithm capable of learning from limited cycles through streaming data with reduced interference on old samples. In the following sections, we review background concepts, outline our proposed algorithm, and note connections to prior work not directly pertaining to CL.

3. Preliminaries

We consider a setting where a sequence of T tasks $[\tau_1, \tau_2, ..\tau_T]$ is learnt by observing their training data $[D_1, D_2, ..D_T]$ sequentially. We define $X^i, Y^i = \{(X_n^i, Y_n^i)\}_{n=0}^{N_i}$ as the set of N_i input-label pairs randomly drawn from D_i. An any time-step j during online learning, we aim to minimize the empirical risk of the model on all the t tasks seen so far ($\tau_{1:t}$), given limited access to data (X^i, Y^i) from previous tasks τ_i ($i < t$).

We refer to this objective as the *cumulative risk*, given by:

$$\begin{aligned}&\sum_{t=1}^{T}\sum_{i=1}^{t} E_{(X^i,Y^i)}\left[\ell_i\left(f_i\left(X^i;\theta\right), Y^i\right)\right]\\ &= \sum_{t=1}^{T} E_{(X^{1:t},Y^{1:t})}\left[L_t\left(f\left(X^{1:t};\theta\right), Y^{1:t}\right)\right]\end{aligned} \tag{1}$$

where ℓ_i is the loss on τ_i and f_i is a learnt, possibly task-specific mapping from inputs to outputs using parameters θ. $L_t = \sum_{i=1}^{t} \ell_i$ is the sum of all task-wise losses for tasks $\tau_{1:t}$. T is unknown beforehand and we merely use it to specify an upper bound in our summations.

Let θ_0^j denote the model's parameters at time j, and ℓ denote a loss objective to be minimised. Then the SGD operator acting on parameters θ_0^j, denoted by $U(\theta_0^j)$, is defined as:

$$U\left(\theta_0^j\right) = \theta_1^j = \theta_0^j - \alpha\nabla_{\theta_0^j}\ell(\theta_0^j) = \theta_0^j - \alpha g_0^j \tag{2}$$

where $g_0^j = \nabla_{\theta_0^j}\ell(\theta_0^j)$. f$U$ can be composed for k updates as $U_k\left(\theta_0^j\right) = U... \circ U \circ U(\theta_0^j) = \theta_k^j$. α is a scalar or a vector LR. $U(\cdot, x)$ implies gradient updates are made on data sample x. We now introduce the MAML (Finn et al., 2017) and OML (Javed & White, 2019) algorithms, that we build upon in Section 4.

Model-Agnostic Meta-Learning (MAML): Meta-learning (Schmidhuber, 1987), or *learning-to-learn* (Thrun & Pratt, 1998) has emerged as a popular approach for training models amenable to fast adaptation on limited data. MAML (Finn et al., 2017) proposed optimising model parameters to learn a set of tasks *while* improving on auxiliary objectives like few-shot generalisation within their task distributions. We review some terminology used in gradient-based meta-learning: 1) at time-step j during training, model parameters θ_0^j (or θ_0 for simplicity), are often referred to as an *initialisation*, since MAML finds an ideal *starting point* for few-shot gradient-based adaptation on unseen data. 2) *Fast updates*, refer to gradient-based updates made to a copy of θ_0, optimising an *inner objective* (in this case, ℓ_i for some τ_i). 3) A *meta-update* involves the *trajectory* of fast updates from θ_0 to θ_k, followed by making a permanent *slow-update* to θ_0. This *slow-update* is computed by evaluating an auxiliary objective (or *meta-loss* L_{meta}) on θ_k, and differentiating through the *trajectory* to obtain $\nabla_{\theta_0} L_{meta}(\theta_k)$. MAML thus optimises θ_0^j to perform optimally on tasks in $\{\tau_{1:t}\}$ after undergoing a few gradient updates on their samples by optimsiing the objective:

$$\min_{\theta_0^j} E_{\tau_{1:t}}\left[L_{meta}\left(U_k(\theta_0^j)\right)\right] = \min_{\theta_0^j} E_{\tau_{1:t}}\left[L_{meta}(\theta_k^j)\right] \tag{3}$$

Equivalence of Meta-Learning and Gradient Alignment Objectives: The first-order meta-learning algorithm Reptile has been shown to be approximately equivalence to the MAML objective (Nichol et al., 2018). MER (Riemer et al., 2019) then showed that their CL objective of minimising loss on tasks $\tau_{1:t}$ seen till time j, while aligning gradients between them:

$$\min_{\theta_0^j} \left(\sum_{i=1}^{t} \left(\ell_i(\theta_0^j) \right) - \alpha \sum_{p,q \leq t} \left(\frac{\partial \ell_p \left(\theta_0^j \right)}{\partial \theta_0^j} \cdot \frac{\partial \ell_q \left(\theta_0^j \right)}{\partial \theta_0^j} \right) \right) \tag{4}$$

is equivalent to the Reptile objective i.e.:

$$\min_{\theta_0^j} E_{\tau_{1:t}} \left[L_t \left(U_k(\theta_0^j) \right) \right] \tag{5}$$

where the *meta-loss* $L_t = \sum_{i=1}^{t} \ell_i$ is evaluated on samples from tasks $\tau_{1:t}$. We will show how a version of this objective coincides with our proposed multi-step MAML-based algorithm. Note that this implies that the procedure to learn an *initialisation* through meta-learning, coincides with the procedure to learn optimal parameters for CL.

Online-aware Meta-Learning (OML): (Javed & White, 2019) proposed to meta-learn a *Representation-Learning Network (RLN)* to provide a representation suitable for CL to a *Task-Learning Network (TLN)*. The RLN's representation is learnt in an *offline* phase, where it is trained using *catastrophic forgetting as the learning signal*. Data from a fixed set of tasks (τ_{val}), is repeatedly used to evaluate the RLN and TLN as the TLN undergoes temporally correlated updates. In every *meta-update*'s inner loop, the TLN undergoes *fast updates* on streaming task data with a frozen RLN. The RLN and updated TLN are then evaluated through a *meta-loss* computed on data from τ_{val} along with the current task. This tests how the performance of the model has changed on τ_{val} in the process of trying to learn the streaming task. The meta-loss is then differentiated to get gradients for *slow updates* to the TLN and RLN. This composition of two losses to simulate CL in the inner loop and test *forgetting* in the outer loop, is referred to as the *OML objective*. The *slow updates* to the RLN lead to it eventually providing a better representation to the TLN for CL.

4. Proposed approach

In the previous section, we described how the OML objective can directly regulate CL behaviour, and how MER exploits the equivalence of the objectives of Reptile and CL to construct a CL algorithm that uses Reptile updates. We noted that OML is for offline pre-training of a static representation and that MER's online algorithm is still prohibitively slow. We observe that we can adapt the OML objective to be optimised *continuously* online with respect to all of the model parameters similar to a standard k-step MAML update. We then prove the equivalence of this MAML objective to that of a standard CL objective, to be able to use this algorithm for continual learning in a principled manner.

In this section, we describe *Continual-MAML* (C-MAML), the base algorithm that we propose for online continual learning. We then detail an extension to C-MAML, referred to as Look-Ahead MAML (La-MAML), outlined in Algorithm 1.

4.1. C-MAML

C-MAML aims to optimise the OML objective *online*, so that learning on the current task doesn't lead to forgetting on previously seen tasks. We define this objective, adapted to optimise a model's parameters θ instead of a representation at time-step j, as:

$$\min_{\theta_0^j} \text{OML}(\theta_0^j, t) = \min_{\theta_0^j} \sum_{\mathcal{S}_k^j \sim D_t} \left[L_t \left(U_k(\theta_0^j, \mathcal{S}_k^j) \right) \right] \tag{6}$$

where S_k^j is a stream of data tuples $\left(X_{j+l}^t, Y_{j+l}^t \right)_{l=1}^{k}$ from the current task τ_t that is seen by the model at time j. The meta-loss $L_t = \sum_{i=1}^{t} \ell_i$ is evaluated on $\theta_k^j = U_k(\theta_0^j, S_k^j)$. It evaluates the fitness of θ_k^j for the continual learning prediction task defined in Eq. 1 until τ_t. We omit the implied data argument $(x^i, y^j) \sim (X^i, Y^i)$ that is the input to each loss ℓ_i in L_t for any task τ_i. We will show in Appendix B that optimising our objective in Eq. 6 through the k-step MAML update in C-MAML:

$$\min_{\theta_0^j} E_{\tau_{1:t}} \left[L_t \left(U_k(\theta_0^j) \right) \right] \tag{7}$$

coincides with optimising the CL objective of AGEM (Chaudhry et al., 2019):

$$\min_{\theta_0^j} \sum_{i=1}^{t} \left(\ell_i(\theta_0^j) - \alpha \frac{\partial \ell_i \left(\theta_0^j \right)}{\partial \theta_0^j} \cdot \frac{\partial \ell_t \left(\theta_0^j \right)}{\partial \theta_0^j} \right) \tag{8}$$

This differs from Eq. 4's objective by being *asymmetric*: it focuses on aligning the gradients of τ_t and the average gradient of $\tau_{1:t}$ instead of aligning all the pair-wise gradients between tasks $\tau_{1:t}$. Incentivizing alignment of all $\tau_{1:t}$ with the common τ_t indirectly incentivises alignment amongst $\tau_{1:t}$. This results in a drastic speedup over MER's objective (Eq. 4) which tries to align all $\tau_{1:t}$ equally, and therefore has to resample incoming samples $s \sim \tau_t$ to form a uniformly distributed batch over $\tau_{1:t}$. Since each s then has $\frac{1}{t}$-th the contribution in gradient updates, it becomes necessary for MER to take multiple passes over many such uniform batches including s.

To evaluate the *meta-loss* $L_t(\theta_k^j)$ which indicates performance of θ_k^j on all tasks seen till time j, old tasks' data is sampled from a replay-buffer R. R is populated through *reservoir sampling* on the incoming data stream as in (Riemer et al., 2019). The SGD-based *inner-updates* starting from initial weights θ_0^j are carried out using data from the task being streamed at the time of meta-update step j.

At the start of every meta-update step, a batch b is sampled from the current task and is combined with a batch sampled from R to form the *meta-batch*, b_m. θ_0^j is updated for k steps by seeing samples from b one at a time. The outer loss is evaluated on b_m.

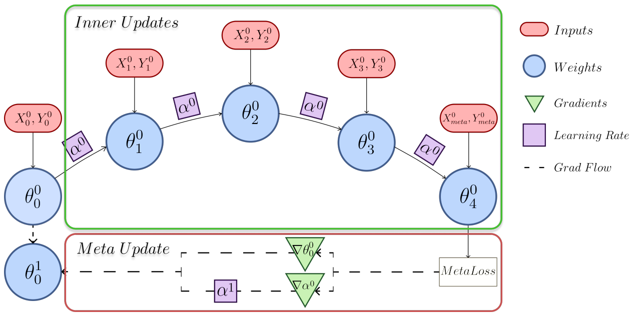


Figure 1. The proposed **La-MAML** algorithm: For every batch of data, the initial weights undergo a series of k *fast updates* to obtain θ_k^j (here $j = 0$), which is evaluated against a meta-loss to backpropagate gradients with respect to the weights θ_0^0 and LRs α^0. First α^0 is updated to α^1 which is then used to update θ_0^0 to θ_0^1 The blue boxes indicate *fast weights* while the green boxes indicate gradients for the *slow updates*. LRs and weights are updated in an asynchronous manner.

4.2. La-MAML

Despite the fact that meta-learning incentivises the alignment of *within-task* and *across-task* gradients, there can still be some interference between the gradients of old and new tasks, $\tau_{1:t-1}$ and τ_t respectively. This would lead to forgetting on $\tau_{1:t-1}$, since its data is no longer fully available to us. This is especially true at the beginning of training a new task, when its gradients aren't necessarily aligned with the old ones. A mechanism is thus needed to ensure that *meta-updates* are conservative with respect to $\tau_{1:t-1}$, so as to avoid negative transfer on them. The magnitude and direction of the *meta-update* needs to be regulated, guided by how the loss on $\tau_{1:t-1}$ would be affected by the update.

In **La-MAML**, we include a set of learnable per-parameter learning rates (LRs) to be used in the *inner updates*, as depicted in Figure 1. This is motivated by our observation that the expression for the gradient of Eq. 6 with respect to the inner loop's LRs directly reflects the alignment between the old and new tasks. The augmented learning objective and its gradient with respect to the LR vector α, denoted as $g_{MAML}(\alpha)$ is then given as:

$$\min_{\theta_0^j, \alpha^j} \sum_{\mathcal{S}_k^j \sim D_t} \left[L_t \left(U \left(\alpha^j, \theta_0^j, \mathcal{S}_k^j \right) \right) \right] \tag{9}$$

$$g_{MAML}(\alpha^j) = \frac{\partial}{\partial \theta_k^j} L_t \left(\theta_k^j \right) \cdot \left(- \sum_{n=0}^{k-1} \frac{\partial}{\partial \theta_n^j} \ell_t \left(\theta_n^j \right) \right) \tag{10}$$

We provide the full derivation in the Appendix A, and simply state the expression for a first-order approximation (Finn et al., 2017) of $g_{MAML}(\alpha)$ here. The first term in $g_{MAML}(\alpha)$ corresponds to the gradient of the meta-loss on batch b_m: g_{meta}. The second term indicates the cumulative gradient from the inner-update trajectory: g_{traj}. This expression indicates that the gradient of the LRs will be negative when the inner product between g_{meta} and g_{traj} is high, ie. the two are aligned. Similarly, it should be zero when the two are orthogonal (not interfering) and positive when there is disagreement between the two, leading to interference. Negative (positive) LR gradients would pull up (down) the LR magnitude. We depict this visually in Figure 2.

Algorithm 1 La-MAML : Look-ahead MAML

```
Input: Network weights $\theta$, LRs $\alpha$, inner objective $\ell$, meta objective $L$, learning rate for $\alpha$ : $\eta$
$R \leftarrow \{\}$                                    ▷ replay-buffer
for $t = 1$ to $T$ do
    $Data_t \leftarrow Sample(D_t)$
    for $ep = 1$ to $num_{ep}$ do
        for $b$ in $Data_t$ do
            $k = sizeof(b)$
            $b_m \leftarrow Sample(R, b)$
            $Initialize\ \theta_0^0, \alpha^0 \leftarrow \theta, \alpha$
            for $n = 0$ to $k - 1$ do
                Push $b[n]$ to R with reservoir sampling
                $\theta_{n+1}^0 = \theta_n^0 - \alpha^0 \cdot \nabla_{\theta_n^0} \ell_t(\theta_n^0, b[n])$
            end for
            $L_{meta} = L_t(\theta_k^0, b_m)$
            $\alpha^1 \leftarrow \alpha^0 - \eta \nabla_{\alpha^0} L_{meta}$                          (a)
            $\theta_0^1 \leftarrow \theta_0^0 - max(0, \alpha^1) \cdot \nabla_{\theta_0^0} L_{meta}$      (b)
            $\alpha, \theta \leftarrow \alpha^1, \theta_0^1$
        end for
    end for
end for
```

We propose updating the network weights and LRs *asynchronously* in the meta-update. Let α^{j+1} be the updated LR vector obtained by taking an SGD step with the LR gradient from Eq. 10 at time j. We then update the weights as:

$$\theta_0^{j+1} \leftarrow \theta_0^j - max(0, \alpha^{j+1}) \cdot \nabla_{\theta_0^j} L_t(\theta_k^j) \tag{11}$$

where k is the number of steps taken in the inner-loop. The LRs α^{j+1} are clipped to positive values to avoid *ascending* the gradient, and also to avoid making *interfering* parameter-updates, thus mitigating catastrophic forgetting. The meta-objective thus conservatively modulates the pace and direction of learning to achieve quicker learning progress on a new task while facilitating transfer on old tasks. Algorithm 1 illustrates this procedure. Lines (a), (b) are the only difference between C-MAML and La-MAML, with C-MAML using a fixed scalar LR α for the meta-update to θ_0^j instead of α^{j+1}.

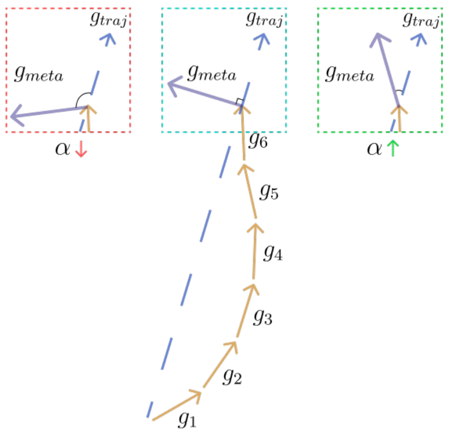


Figure 2. Different scenarios for the alignment of g_{traj} (blue dashed line) and g_{meta}, going from interference (left) to alignment (right). Yellow arrows denote the *inner updates*. The LR α increases (decreases) when gradients align (interfere).

Our meta-learning based algorithm incorporates concepts from both prior-based and replay-based approaches. Our LRs modulate the parameter updates in an entirely data driven manner, guided by the interplay between the gradients of the replay samples and the streaming task. However, since LRs evolve with every meta-update, their decay is temporary. This is unlike many prior-based approaches, where penalties on the change in parameters gradually become so high that the network capacity saturates (Kirkpatrick et al., 2017). Our learnable LRs can be modulated to high and low values as tasks arrive, thus being a simpler, flexible and elegant way to constrain weights. This asynchronous update resembles trust-region optimisation (Yuan, 1999) since the LRs are evolved in a manner similar to *look-ahead search*, which adjusts step-sizes based on the loss incurred on adapted parameters. Our LR update is also analogous to the heuristic uncertainty-based LR update schemes of UCB (Ebrahimi et al., 2020) or BGD (Zeno et al., 2018), the latter of which we will include in our comparative analysis in Section 5.3.

4.3. Connections to Work Outside Continual Learning

Stochastic Meta-Descent (SMD): When learning over a non-stationary data distribution, using decaying learning rate schedules is not common. Strictly diminishing learning rate schedules aim for closer and faster *convergence* to a fixed mimima of a stationary distribution, which is at odds with the goal of online and continual learning. In many Continual Learning scenarios it is impossible to manually tune the schedule since the extent of the data distribution is unknown. However, *adaptivity* in learning rates is still highly desired to better adapt to the optimisation landscape and accelerate learning. Another reason to desire adaptivity in CL is to modulate the degree of adaptation of certain parameters, to reduce catastrophic forgetting. Our adaptive learning rates can be connected to work on *meta-descent* (Baydin et al., 2018; Schraudolph, 1999) in standard *offline* supervised learning (OSL). While several variations of *meta-descent* exist, the core idea behind these variations and our approach is the same: *gain adaptation*, analogous to gain adjustment in a Kalman Filter (KF). In a KF, the *gain* is a quantity that signifies how much trust we place in a proposed belief or parameter update. While in our case, we want to check the correlation between old and new task gradients to adapt the gain so that we can make the most shared progress on old and new tasks, in the case of (Baydin et al., 2018; Schraudolph, 1999) the correlation between two successive stochastic gradients on the same data distribution is used to converge faster. HD (Baydin et al., 2018) proposes analytically, *asynchronously* updating learning rates during optimization. This is done by differentiating the update rule at any time-step j with respect to the learning rates at time-step $j-1$. We instead rely on the meta-objective's differentiability with respect to the LRs, to obtain these learning rate *hypergradients* automatically.

Learning LRs in meta-learning: Meta-SGD (Li et al., 2017) first proposed learning the per-parameter learning rates used within the inner-loop of MAML in the few-shot learning setting. Some notable differences between their update and ours exist. They *synchronously* update the weights and learning rates while our *asynchronous* update to the learning rates serves to carry out a more conservative update to the weights. The intuition for our update stems from the need to mitigate gradient interference and its connection to the transfer-interference trade-off ubiquitous in continual learning. Similarly, α-MAML (Singh Behl et al., 2019) proposed analytically updating the two *scalar* learning rates used in the inner and outer MAML update for more adaptive few-shot learning. In contrast, our *per-parameter* learning rates are modulated implicitly through back-propagation, to regulate change in parameters based on their alignment across tasks, providing our model with a more powerful degree of adaptability in the CL domain.

5. Experiments

In this section we evaluate La-MAML in task incremental settings, where the model learns a set of sequentially streaming classification tasks. Experiments are performed on the MNIST, CIFAR and TinyImagenet (tin) datasets. Similar

Table 1. RA, BTI and their standard deviation on MNIST benchmarks. Each experiment is run with 5 seeds.

METHOD	ROTATIONS		PERMUTATIONS		MANY	
	RA	BTI	RA	BTI	RA	BTI
ONLINE	53.38 ± 1.53	-5.44 ± 1.70	55.42 ± 0.65	-13.76 ± 1.19	32.62 ± 0.43	-19.06 ± 0.86
EWC	57.96 ± 1.33	-20.42 ± 1.60	62.32 ± 1.34	-13.32 ± 2.24	33.46 ± 0.46	-17.84 ± 1.15
GEM	67.38 ± 1.75	-18.02 ± 1.99	55.42 ± 1.10	-24.42 ± 1.10	32.14 ± 0.50	-23.52 ± 0.87
MER	**77.42** ± **0.78**	**-5.60**±**0.70**	73.46± 0.45	-9.96 ± 0.45	47.40 ± 0.35	-17.78 ± 0.39
C-MAML	77.33 ± 0.29	-7.88 ± 0.05	**74.54** ± **0.54**	-10.36 ± 0.14	47.29 ± 1.21	-20.86 ± 0.95
SYNC	70.07 ± 2.07	-15.42 ± 4.19	60.12 ± 1.84	-19.56 ± 2.43	42.73 ± 1.20	-23.96 ± 1.95
LA-MAML	**77.42** ± **0.65**	-8.64 ± 0.403	74.34 ± 0.67	**-7.60** ± **0.51**	**48.46** ± **0.45**	**-12.96** ± **0.073**

to (Riemer et al., 2019), we use the retained accuracy (RA) metric to compare various approaches. RA is the average accuracy of the model across tasks at the end of training. We also report the *backward-transfer and interference* (BTI) values which measure the average change in the accuracy of each task from when it was learnt to the end of the last task. A smaller BTI implies lesser forgetting during training.

Efficient Lifelong Learning (LLL): Formalized in (Chaudhry et al., 2019), the setup of efficient lifelong learning assumes that incoming data for every task has to be processed in only one single pass: once processed, data samples are not accessible anymore unless they were added to a replay memory. We evaluate our algorithm on this challenging *(Single-Pass)* setup as well as the standard *(Multiple-Pass)* setup where offline training-until-convergence is performed for every task, once we have access to its data.

Table 2. Running times for MER and La-MAML on MNIST benchmarks for one epoch

METHOD	ROTATIONS	PERMUTATIONS
LA-MAML	45.95 ± 0.38	46.13 ± 0.42
MER	218.03 ± 6.44	227.11 ± 12.12

5.1. Continual learning benchmarks

First, we carry out experiments on the toy continual learning benchmarks proposed in prior CL works. **MNIST Rotations**, introduced in (Lopez-Paz & Ranzato, 2017), comprises tasks to classify MNIST digits rotated by a different common angle in [0, 180] degrees in each task. In **MNIST Permutations**, tasks are generated by shuffling the image pixels by a fixed random permutation. Unlike Rotations, the input distribution of each task is unrelated here, leading to less positive transfer between tasks. **Many Permutations**, a more complex version of Permutations, has five times more tasks (100 tasks) and five times less training data (200 images per task). We use the same architecture and experimental settings as in MER (Riemer et al., 2019), allowing us to compare directly with their results. We use the cross-entropy loss as the *inner* and *outer* objectives during meta-training. Similar to (Nichol et al., 2018), we see improved performance when evaluating and summing the *meta-loss* at all steps of the inner updates as opposed to just the last one.

We compare our method in the *Single-Pass* setup against multiple baselines including *Online*, *Independent*, *EWC* (Kirkpatrick et al., 2017), *GEM* (Lopez-Paz & Ranzato, 2017) and *MER* (Riemer et al., 2019), as well as ablations detailed in Appendix E. In Table 1, we see that La-MAML achieves comparable or better performance than the baselines on all benchmarks. Table 2 shows that La-MAML matches the performance of MER in less than 20% of the training time, owing to its sample-efficient objective which allows it to make make more learning progress per iteration. This also allows us to scale it to real-world visual recognition problems as described next.

5.2. Real-world classification

While La-MAML fares well on the MNIST benchmarks, we are interested in understanding its capabilities on more complex visual classification benchmarks. We conduct experiments on the **CIFAR-100** dataset in a task-incremental manner (Lopez-Paz & Ranzato, 2017). 20 tasks comprising of disjoint *5-way* classification problems are streamed. We also evaluate on the **TinyImagenet-200** dataset by partitioning its 200 classes into 40 *5-way* classification tasks. Experiments are carried out in both the *Single-Pass* and *Multiple-Pass* settings, where in the latter we allow training for up to a maximum of 10 epochs. Each method has a replay-buffer containing 200 and 400 samples for CIFAR-100 and TinyImagenet respectively. We provide further details about the baselines and evaluation setup in Appendix D.

Table 4 reports the results of these experiments. We consistently observe superior performance of La-MAML as compared to other CL baselines on both datasets across setups. While the iCarl baseline attains lower BTI in some setups, it achieves that at the cost of much lower performance throughout learning. Among the high-performing approaches, La-MAML has the lowest BTI. Recent work (Chaudhry et al., 2019; Riemer et al., 2019) noted that Experience Replay (ER) is often a very strong baseline that closely matches the performance of the proposed algorithms. We highlight the fact that meta-learning and LR modulation combined show an improvement of more than 10 and 18% (as the number of tasks increase from CIFAR to Imagenet) over the ER baseline in our case, with limited replay. Overall, we see that our method is better-performing under both the standard and LLL setups of CL which come with different kinds of challenges. Many CL methods (Ebrahimi et al., 2020; Serra et al., 2018) are suitable for only one of the two setups. As shown in Figure 3, our model evolves to become resistant to *forgetting* as training progresses. This means that beyond a point, it can keep making gradient updates on a small window of incoming samples without needing to do *meta-updates*.

Table 3. Gradient Alignment on CIFAR-100 and IMAGENET dataset (values lie in [-1,1], higher is better)

METHOD	CIFAR-100	IMAGENET
ER	$0.22 \times 10^{-2} \pm 0.0017$	$0.27 \times 10^{-2} \pm 0.0005$
C-MAML	$1.84 \times 10^{-2} \pm 0.0003$	$1.74 \times 10^{-2} \pm 0.0005$
SYNC	$2.28 \times 10^{-2} \pm 0.0004$	$2.17 \times 10^{-2} \pm 0.0020$
LA-MAML	$1.86 \times 10^{-2} \pm 0.0027$	$2.14 \times 10^{-2} \pm 0.0023$

5.3. Evaluation of La-MAML's learning rate modulation

To capture the gains from learning the LRs, we compare La-MAML with our base algorithm, **C-MAML**. We ablate our choice of updating LRs asynchronously by constructing a version of C-MAML where per-parameter learnable LRs are used in the inner updates while the meta-update still uses a constant scalar LR during training. We refer to it as *Sync-La-MAML* or **Sync** since it has synchronously updated LRs that don't modulate the meta-update. While only minor gains are seen on the MNIST benchmarks from asynchronous LR modulation, the performance gap increases as the tasks get harder. On CIFAR-100 and TinyImagenet, we see a trend in the RA of our variants with La-MAML performing best followed by *Sync*. This shows that optimising the LRs aids learning and our *asynchronous* update helps in knowledge consolidation by enforcing conservative updates to mitigate interference.

To test our LR modulation against an alternative *bayesian* modulation scheme proposed in BGD (Zeno et al., 2018), we define a baseline called Meta-BGD where per-parameter variances are modulated instead of LRs. This is described in further detail in Appendix E. Meta-BGD emerges as a strong baseline and matches the performance of C-MAML given enough Monte Carlo iterations m, implying m times more computation than C-MAML. Additionally, Meta-BGD was found to be sensitive to hyperparameters and required extensive tuning. We present a discussion of the robustness of our approach in Appendix C.

We also compare the gradient alignment of our three variants along with ER in Table 3 by calculating the cosine similarity between the gradients of the replay samples and newly arriving data samples. As previously stated, the aim of many CL algorithms is to achieve high gradient alignment across tasks to allow parameter-sharing between them. We see that our variants achieve an order of magnitude higher cosine similarity compared to ER, verifying that our objective promotes gradient alignment.

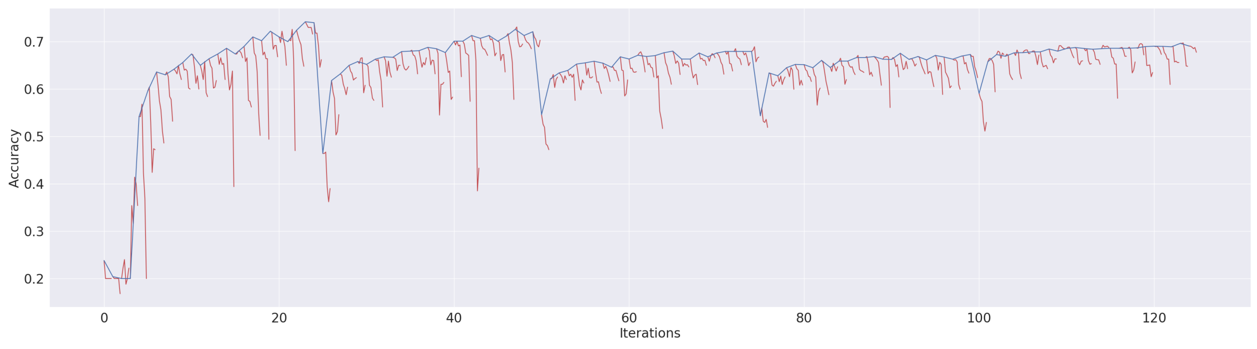


Figure 3. Retained Accuracy (RA) for La-MAML plotted every 25 meta-updates up to Task 5 on CIFAR-100. RA at iteration *j* (with *j* increasing along the x-axis) denotes accuracy on all tasks seen uptil then. Red denotes the RA computed during the *inner updates* (at θ_k^j). Blue denotes RA computed at θ_0^{j+1} right after a *meta-update*. We see that in the beginning, inner updates lead to catastrophic forgetting (CF) since the weights are not suitable for CL yet, but eventually become resistant when trained to retain old knowledge while learning on a stream of correlated data. We also see that RA maintains its value even as more tasks are added indicating that the model is successful at learning new tasks without sacrificing performance on old ones.

Table 4. Results on the standard continual (Multiple) and LLL (Single) setups with CIFAR-100 and TinyImagenet-200. Experiments are run with 3 seeds. * indicates result omitted due to high instability.

METHOD	CIFAR-100				TINYIMAGENET			
	MULTIPLE		SINGLE		MULTIPLE		SINGLE	
	RA	BTI	RA	BTI	RA	BTI	RA	BTI
IID	85.60 ± 0.40	-	-	-	77.1 ± 1.06	-	-	-
ER	59.70 ± 0.75	-16.50 ± 1.05	47.88 ± 0.73	-12.46 ± 0.83	48.23 ± 1.51	-19.86 ± 0.70	39.38 ± 0.38	-14.33 ± 0.89
ICARL	60.47 ± 1.09	-15.10 ± 1.04	53.55 ± 1.69	**-8.03 ± 1.16**	54.77 ± 0.32	**-3.93 ± 0.55**	45.79 ± 1.49	**-2.73 ± 0.45**
GEM	62.80 ± 0.55	-17.00 ± 0.26	48.27 ± 1.10	-13.7 ± 0.70	50.57 ± 0.61	-20.50 ± 0.10	40.56 ± 0.79	-13.53 ± 0.65
AGEM	58.37 ± 0.13	-17.03 ± 0.72	46.93 ± 0.31	-13.4 ± 1.44	46.38 ± 1.34	-19.96 ± 0.61	38.96 ± 0.47	-13.66 ± 1.73
MER	-	-	51.38 ± 1.05	-12.83 ± 1.44	-	-	44.87 ± 1.43	-12.53 ± 0.58
META-BGD	65.09 ± 0.77	-14.83 ± 0.40	57.44 ± 0.95	-10.6 ± 0.45	*	*	50.64 ± 1.98	-6.60 ± 1.73
C-MAML	65.44 ± 0.99	-13.96 ± 0.86	55.57 ± 0.94	-9.49 ± 0.45	61.93 ± 1.55	-11.53 ± 1.11	48.77 ± 1.26	-7.6 ± 0.52
SYNC	67.06 ± 0.62	-13.66 ± 0.50	58.99 ± 1.40	-8.76 ± 0.95	65.40 ± 1.40	-11.93 ± 0.55	**52.84 ± 2.55**	-7.3± 1.93
LA-MAML	**70.08 ± 0.66**	**-9.36 ± 0.47**	**61.18 ± 1.44**	-9.00 ± 0.2	**66.99 ± 1.65**	**-9.13 ± 0.90**	52.59 ± 1.35	**-3.7 ± 1.22**

6. Conclusion

We introduced La-MAML, an efficient meta-learning algorithm that leverages replay to avoid forgetting and favors positive backward transfer by learning the weights and LRs in an asynchronous manner. It is capable of learning online on a non-stationary stream of data and scales to vision tasks. We presented results that showed better performance against the state-of-the-art in the setup of efficient lifelong learning (LLL) (Chaudhry et al., 2019), as well as the standard continual learning setting. In the future, more work on analysing and producing good optimizers for CL is needed, since many of our standard go-to optimizers like Adam (Kingma & Ba, 2014) are primarily aimed at ensuring faster convergence in *stationary* supervised learning setups. Another interesting direction is to explore how the connections to *meta-descent* can lead to more stable training procedures for meta-learning.

Acknowledgements

The authors are grateful to Matt Reimer, Sharath Chandra Raparthy, Alexander Zimin and the anonymous reviewers for proof-reading the paper and suggesting improvements. This research was enabled in part by support provided by Compute Canada (`www.computecanada.ca`).

References

URL `https://tiny-imagenet.herokuapp.com/`.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Sk2u1g-0-`.

Aljundi, R., Rohrbach, M., and Tuytelaars, T. Selfless sequential learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkxbrn0cYX`.

Baydin, A. G., Cornish, R., Rubio, D. M., Schmidt, M., and Wood, F. Online learning rate adaptation with hypergradient descent. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkrsAzWAb`.

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 233–248, 2018.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Hkf2_sC5FX`.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., and Ranzato, M. On Tiny Episodic Memories in Continual Learning. *arXiv e-prints*, art. arXiv:1902.10486, February 2019.

Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. Uncertainty-guided continual learning with bayesian neural networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HklUCCVKDB`.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceed-*

ings of the 34th International Conference on Machine Learning-Volume 70, pp. 1126–1135. JMLR. org, 2017.

Finn, C., Rajeswaran, A., Kakade, S., and Levine, S. Online meta-learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1920–1930, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/finn19a.html.

French, R. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 05 1999. doi: 10.1016/S1364-6613(99)01294-2.

Javed, K. and White, M. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, pp. 1818–1828, 2019.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/content/114/13/3521.

Le, L., Kumaraswamy, R., and White, M. Learning sparse representations in reinforcement learning with sparse coding. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 2067–2073. AAAI Press, 2017. ISBN 9780999241103.

Li, Z., Zhou, F., Chen, F., and Li, H. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

Mcclelland, J., Mcnaughton, B., and O'Reilly, R. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102:419–57, 08 1995. doi: 10.1037/0033-295X.102.3.419.

Nagabandi, A., Finn, C., and Levine, S. Deep online learning via meta-learning: Continual adaptation for model-based RL. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxAfnA5tm.

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

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., , and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1gTShAct7.

Schmidhuber, J. Evolutionary principles in self-referential learning. 1987.

Schraudolph, N. Local gain adaptation in stochastic gradient descent. 06 1999. doi: 10.1049/cp:19991170.

Serra, J., Suris, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4548–4557, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/serra18a.html.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2990–2999. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6892-continual-learning-with-deep-generative-repl pdf.

Singh Behl, H., Güneș Baydin, A., and Torr, P. H. S. Alpha MAML: Adaptive Model-Agnostic Meta-Learning. *arXiv e-prints*, art. arXiv:1905.07435, May 2019.

Thrun, S. and Pratt, L. (eds.). *Learning to Learn*. Kluwer Academic Publishers, USA, 1998. ISBN 0792380479.

Yuan, Y.-x. A review of trust region algorithms for optimization. *ICM99: Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, 09 1999.

Zeno, C., Golan, I., Hoffer, E., and Soudry, D. Task Agnostic Continual Learning Using Online Variational Bayes. *arXiv e-prints*, art. arXiv:1803.10123, Mar 2018.

A. Hypergradient Derivation for La-MAML

We derive the gradient of the weights θ_0^j and LRs α^j at time-step j under the k-step MAML objective, with $L_t = \sum_{i=0}^{t} \ell_i$ as the *meta-loss* and ℓ_t as the *inner-objective*:

$$
\begin{aligned}
g_{\text{MAML}}(\alpha^j) &= \frac{\partial}{\partial \alpha^j} L_t\left(\theta_k^j\right) = \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \frac{\partial}{\partial \alpha^j}\left(\theta_k^j\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \frac{\partial}{\partial \alpha^j}\left(U\left(\theta_{k-1}^j\right)\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \frac{\partial}{\partial \alpha^j}\left(\theta_{k-1}^j - \alpha^j \frac{\partial \ell_t(\theta_{k-1}^j)}{\partial \theta_{k-1}^j}\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(\frac{\partial}{\partial \alpha^j}\theta_{k-1}^j - \frac{\partial}{\partial \alpha^j}\left(\alpha^j \frac{\partial \ell_t(\theta_{k-1}^j)}{\partial \theta_{k-1}^j}\right)\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(\frac{\partial}{\partial \alpha^j}\theta_{k-1}^j - \frac{\partial \ell_t(\theta_{k-1}^j)}{\partial \theta_{k-1}^j}\right) \\
&\quad \text{(Taking } \frac{\partial \ell_t\left(\theta_{k-1}^j\right)}{\partial \theta_{k-1}^j} \text{ as a constant w.r.t } \alpha^j \text{ to get the FO-MAML approximation)} \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(\frac{\partial}{\partial \alpha} U\left(\theta_{k-2}^j\right) - \left(\frac{\partial \ell_t(\theta_{k-1}^j)}{\partial \theta_{k-1}^j}\right)\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(\frac{\partial}{\partial \alpha}\theta_0^j - \sum_{n=0}^{k-1} \frac{\partial \ell_t(\theta_n^j)}{\partial \theta_n^j}\right) \qquad (a) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(- \sum_{n=0}^{k-1} \frac{\partial \ell_t(\theta_n^j)}{\partial \theta_n^j}\right) \qquad (b)
\end{aligned}
$$

Where (a) is obtained by recursively expanding and differentiating the update function $U()$ as done in the step before it. (b) is obtained by assuming that the initial weight in the meta-update at time j : θ_0^j, is constant with respect to α^j.

Similarly we can derive the MAML gradient for the weights θ_0^j, denoted as $g_{\text{MAML}}(\theta_0^j)$ as:

$$
\begin{aligned}
g_{\text{MAML}}(\theta_0^j) &= \frac{\partial}{\partial \theta_0^j} L_t(\theta_k^j) = \frac{\partial}{\partial \theta_k^j} L_t(\theta_k^j) \frac{\partial \theta_k^j}{\partial \theta_0^j} = \frac{\partial}{\partial \theta_k^j} L_t(\theta_k^j) \frac{\partial U_k(\theta_{k-1}^j)}{\partial \theta_0^j} \\
&= \frac{\partial}{\partial \theta_k^j} L_t(\theta_k^j) \frac{\partial}{\partial \theta_{k-1}^j} U(\theta_{k-1}^j) \cdots \frac{\partial}{\partial \theta_0^j} U(\theta_1^j) \\
&\quad \text{(repeatedly applying chain rule and using } \theta_k^j = U(\theta_{k-1}^j) \text{)} \\
&= L_t'(\theta_k^j)\left(I - \alpha \ell_t''(\theta_{k-1}^j)\right) \cdots \left(I - \alpha \ell_t''(\theta_0^j)\right) \\
&\quad \left(\text{ using } U'(\theta_n^j) = I - \alpha \ell_t''(\theta_n^j)\right) \quad (\prime \text{ implies derivative with respect to argument}) \\
&= \left(\prod_{n=0}^{k-1} \left(I - \alpha \ell_t''(\theta_n^j)\right)\right) L_t'(\theta_k^j)
\end{aligned}
$$

Setting all first-order gradient terms as constants to ignore second-order derivatives, we get the first order approximation as:

$g_{\text{FOMAML}}(\theta_0^j) = \left(\prod_{n=0}^{k-1} \left(I - \alpha \ell_t''\left(\theta_n^j\right)\right)\right) L_t'(\theta_k^j) = L_t'(\theta_k^j)$

In Appendix B, we show the equivalence of the C-MAML and CL objectives in Eq. 7 by showing that the gradient of the former ($g_{\text{MAML}}(\theta_0^j)$) is equivalent to the gradient of the latter.

B. Equivalence of Objectives

It is straightforward to show that when we optimise the OML objective through the k-step MAML update, as proposed in C-MAML in Eq. 6:

$$\min_{\theta_0^j} E_{\tau_{1:t}} \left[L_t \left(U_k(\theta_0^j) \right) \right] \tag{12}$$

where the *inner-updates* are taken using data from the streaming task τ_t, and the *meta-loss* $L_t(\theta) = \sum_{i=1}^{t} \ell_i(\theta)$ is computed on the data from all tasks seen so far, it will correspond to minimising the following surrogate loss used in CL :

$$\min_{\theta_0^j} \sum_{i=1}^{t} \left(\ell_i(\theta_0^j) - \alpha \frac{\partial \ell_i \left(\theta_0^j\right)}{\partial \theta_0^j} \cdot \frac{\partial \ell_t \left(\theta_0^j\right)}{\partial \theta_0^j} \right) \tag{13}$$

We show the equivalence for the case when $k = 1$, for higher k the form gets more complicated but essentially has a similar set of terms. Reptile (Nichol et al., 2018) showed that the k-step MAML gradient for the weights θ_0^j at time j, denoted as $g_{\mathrm{MAML}}(\theta_0^j)$ is of the form:

$$\begin{aligned}
\frac{\partial L_{meta}(\theta_k^j)}{\partial \theta_0^j} &= \bar{g}_k - \alpha \bar{H}_k \sum_{i=0}^{k-1} \bar{g}_i - \alpha \sum_{i=0}^{k-1} \bar{H}_i \bar{g}_k + O\left(\alpha^2\right) \\
&= \bar{g}_1 - \alpha \bar{H}_1 \bar{g}_0 - \alpha \bar{H}_0 \bar{g}_1 + O\left(\alpha^2\right) \quad \text{(putting } k = 1\text{)}
\end{aligned}$$

Expressing the terms as derivatives, and using $\frac{\partial}{\partial \theta_0^j}\left(\bar{g}_0 \cdot \bar{g}_1\right) = \bar{H}_1 \bar{g}_0 + \bar{H}_0 \bar{g}_1$, we get :

$$\begin{aligned}
&= \frac{\partial L_{meta}\left(\theta_0^j\right)}{\partial \theta_0^j} - \frac{\partial}{\partial \theta_0^j}\left(\bar{g}_0 \cdot \bar{g}_1\right) \\
&= \frac{\partial \left(\sum_{i=1}^{t} \ell_i(\theta_0^j) - \alpha \bar{g}_1 \cdot \bar{g}_0 \right)}{\partial \theta_0^j} \quad \text{(substituting } L_{meta} = L_t = \sum_{i=1}^{t} \ell_i\text{)} \\
&= \frac{\partial \left(\sum_{i=1}^{t} \ell_i(\theta_0^j) - \alpha \frac{\partial L_{meta}(\theta_0^j)}{\partial \theta_0^j} \frac{\partial \ell_t(\theta_0^j)}{\partial \theta_0^j} \right)}{\partial \theta_0^j} \\
&= \frac{\partial \left(\sum_{i=1}^{t} \ell_i(\theta_0^j) - \alpha \frac{\partial \sum_{i=1}^{t} \ell_i(\theta_0^j)}{\partial \theta_0^j} \frac{\partial \ell_t(\theta_0^j)}{\partial \theta_0^j} \right)}{\partial \theta_0^j} \quad \text{(expanding } L_{meta}\text{)} \\
&= \frac{\partial \left(\sum_{i=1}^{t} \ell_i(\theta_0^j) - \alpha \sum_{i=1}^{t} \frac{\partial \ell_i(\theta_0^j)}{\partial \theta_0^j} \frac{\partial \ell_t(\theta_0^j)}{\partial \theta_0^j} \right)}{\partial \theta_0^j}
\end{aligned}$$

which is the same as the gradient of Eq. 13.

where:

$$\begin{aligned}
\bar{g}_k &= \frac{\partial L_{meta}\left(\theta_0^j\right)}{\partial \theta_0^j} && \text{(gradient of the \textit{meta-loss} evaluated at the initial point)} \\
\bar{g}_i &= \frac{\partial}{\partial \theta_0^j} L_{inner}(\theta_0^j) \quad (\text{for } i < k) && \text{(gradients of the \textit{inner-updates} evaluated at the initial point)} \\
\theta_{i+1}^j &= \theta_i^j - \alpha g_i && \text{(sequence of parameter vectors)}
\end{aligned}$$

$$\bar{H}_k = L''_{meta}\left(\theta_0^j\right) \qquad\qquad \text{(Hessian of the \textit{meta-loss} evaluated at the initial point)}$$

$$\bar{H}_i = L''_{inner}\left(\theta_0^j\right) \quad \text{(for } i < k\text{)} \quad \text{(Hessian of the \textit{inner-objective} evaluated at the initial point)}$$

And, in our case:

$$L_{meta} = L_t = \sum_{i=1}^{t} \ell_i$$

$$L_{inner} = \ell_t$$

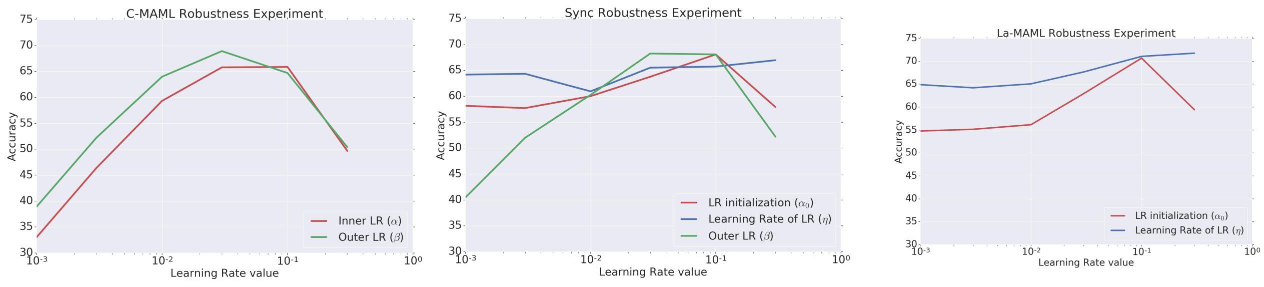
Bias in the objective: We can see in Eq. 13 that the gradient alignment term introduces some bias, which means that we don't exactly converge to the minimiser of the losses on all tasks. This has been acceptable in CL for the following reason: We never aim to reach the minimiser of some stationary distribution anyway in the CL regime (as also mentioned in Section 4.3). If we did converge to the minimiser of say t tasks at some time j, this minimiser would no longer be optimal as soon as we see the new task τ_{t+1}. Therefore, in the limit of infinite tasks or time, gradient alignment between tasks will pay off more as opposed to being able to converge to the exact minima, by allowing us to make shared progress on both previous and incoming tasks.

C. Robustness

Learning rate is one of the most crucial hyper-parameters during training and it often has to be tuned extensively for each experiment. In this section we analyse the robustness of our proposed variants to their LR-related hyper-parameters on the CIFAR-100 dataset. Our three variants have different sets of these hyper-parameters which are specified as follows:

- **C-MAML**: Inner and outer update LR (scalar) for the weights (α and β)
- **Sync La-MAML**: Inner loop initialization value for the vector LRs (α_0), scalar learning rate of LRs (η) and scalar learning rate for the weights in the outer update (β)
- **La-MAML**: Scalar initialization value for the vector LRs (α_0) and scalar learning rate of LRs (η)

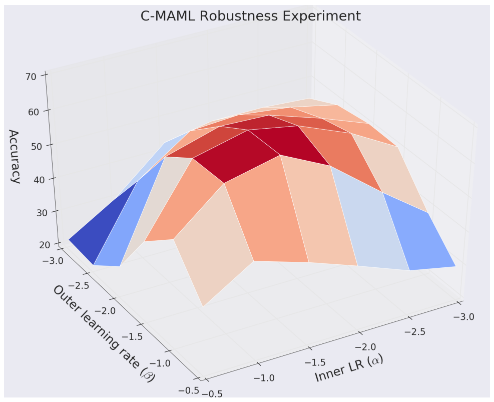
La-MAML is considerably more robust to tuning compared to its variants, as can be seen in Figure 4c. We empirically observe that it only requires tuning of the initial value of the LR, while being relatively insensitive to the learning rate of the LR (η). We see a consistent trend where the increase in η leads to an increase in the final accuracy of the model. The increase is very gradual, across a wide range of LRs varying over 2 orders of magnitude (from 0.003 to 0.3), the difference in RA is only 6%. This means that even without tuning this parameter (η), La-MAML would have outperformed most baselines at their optimally tuned values.



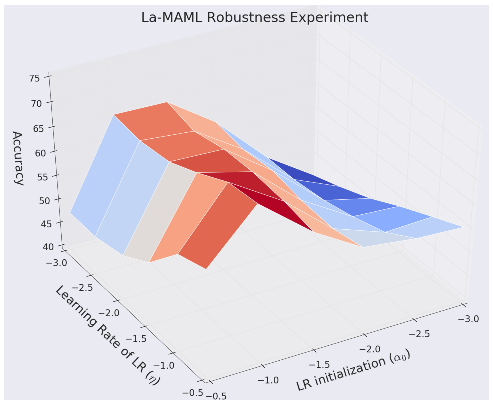
(a) C-MAML: Modulation of α and β (b) Sync: Modulation of α_0, η and β (c) La-MAML: Modulation of α_0 and η

Figure 4. Retained Accuracy vs Learning Rates plot for La-MAML and its variants. Figures are plotted by varying one of the learning rate hyperparameter while keeping the others fixed at their optimal value. The hyperparameter is varied between [0.001, 0.3].

As seen in Figure 4a, C-MAML sees considerable performance variation with the tweaking of both the inner and outer LR. We also see that the effects of the variations of the inner and outer LR follow very similar trends and their optimal values



(a) C-MAML: Modulation of α and β



(b) La-MAML: Modulation of α_0 and η

Figure 5. Plots of Retained Accuracy (RA) across hyper-parameter variation for C-MAML and La-MAML. We show results of the grid search over the learning rate hyperparameters. RA decreases from red to blue. All the hyperparameters are varied between [0.001, 0.3], with the axes being in log-scale.

finally selected are also identical. This means that we could potentially tune them by doing just a 1D search over them together instead of varying both independently through a 2D grid search. The Sync version of La-MAML (Figure 4b), while being relatively insensitive to the scalar initial value α_0 and the η, sees considerable performance variation as the outer learning rate for the weights: β is varied. This variant has the most hyper-parameters and only exists for the purpose of ablation.

Fig. 5 shows the result of 2D grid-searches over sets of the above-mentioned hyper-parameters for C-MAML and La-MAML for a better overview.

D. Experimental

We carry out hyperparameter tuning for all the approaches by performing a search over the range [0.0001 - 0.3] for hyper-parameters related to the learning-rate. Note that in the *Single-Pass* (LLL) setup, since we only see the data in one pass without the option of revisiting, seeing each data-point for multiple *glances* is essential. We define a hyper-parameter called *glances*, which indicates the number of gradient updates made on each data-point, with the performance of the algorithms increasing with the increase in the number of *glances* up to a certain point. This is replaced by the parameter indicating the number of epochs that the model is trained for, for each task in the *Multiple-Pass* setup. Table 5 lists the optimal hyperparameters for all the compared approaches. All setups used the SGD optimiser since it was found to preform better than Adam (Kingma & Ba, 2014) (possibly due to reasons stated in Section 4.3 regarding the CL setup).

To avoid exploding gradients, we clip the gradient values of all approaches at a norm of 2.0. *Class divisions* across different tasks vary with the random seeds with which the experiments were run. Overall, we did not see much variability across different class splits, with the variation being within 0.5-2% of the mean reported result as can be seen from Table 4

For all our baselines, we use a constant batch-size of 10 samples from the streaming task. This batch is augmented with 10 samples from the replay buffer for the replay-based approaches. La-MAML and its variants split the batch from the streaming task into a sequence of smaller disjoint sets to take $k = 5$ gradient steps in the *inner-loop*. In MER, each sample from the incoming task is augmented with a batch of 10 replay samples to form the batch used for the meta-update. We found very small performance gaps between the first and second-order versions of our proposed variants with performance differences in the range of 1-2% for RA. This is in line with the observation that deep neural networks have near-zero hessians since the ReLU non-linearity is linear almost everywhere (**?**).

MNIST Benchmarks: On the MNIST continual learning benchmarks, images of size 28x28 are flattened to create a 1x784 array. This array is passed on to a fully-connected neural network having two layers with 100 nodes each. Each layer uses ReLU non-linearity. These experiments use a modest replay buffer of size 200 for MNIST Rotations and Permutation and size 500 for Many Permutations.

Real-world visual classification: For Cifar and Imagenet we used a CNN having 3 and 4 conv layers respectively with 160

Table 5. Final hyperparameters for all compared approaches on the CIFAR and Imagenet benchmarks

METHOD	PARAMETER	CIFAR-100		IMAGENET	
		SINGLE	MULTIPLE	SINGLE	MULTIPLE
ER	*LR*	0.03	0.03	0.1	0.1
	Epochs/Glances	10	10	10	10
IID	*LR*	-	0.03	-	0.01
	Epochs/Glances	-	50	-	50
ICARL	*LR*	0.03	0.03	0.01	0.01
	Epochs/Glances	2	10	2	10
GEM	*LR*	0.03	0.03	0.03	0.03
	Epochs/Glances	2	10	2	10
AGEM	*LR*	0.03	0.03	0.01	0.01
	Epochs/Glances	2	10	2	10
MER	*LR* α	0.1	-	0.1	-
	LR β	0.1	-	0.1	-
	LR γ	1	-	1	-
	Epochs/Glances	10	-	10	-
META-BGD	η	50	50	50	-
	std-init	0.02	0.02	0.02	-
	β_{inner}	0.1	0.1	0.1	-
	mc-iters	2	2	2	-
	Epochs/Glances	3	10	3	-
C-MAML	α	0.03	0.03	0.03	0.03
	β	0.03	0.03	0.03	0.03
	Epochs/Glances	5	10	2	10
SYNC LA-MAML	α_0	0.1	0.1	0.075	0.075
	β	0.1	0.1	0.075	0.075
	η	0.3	0.3	0.25	0.25
	Epochs/Glances	5	10	2	10
LA-MAML	α_0	0.1	0.1	0.1	0.1
	η	0.3	0.3	0.3	0.3
	Epochs/Glances	10	10	2	10

3x3 filters. The output from the final convolution layer is flattened and is passed through 2 fully connected layers having 320 and 640 units respectively. All the layers are succeeded by ReLU nonlinearity. For CIFAR and Imagenet we allow a replay buffer of size 200 and 400 respectively which implies that each class in these dataset gets roughly about 1-2 samples in the buffer. For Tiny-Imagenet, we split the validation set into *val* and *test* splits, since the labels in the actual test set are not released.

E. Baselines

On the MNIST benchmarks, we compare our algorithm against the baselines used in (Riemer et al., 2019), which are as follows:

- Online: A baseline for the LLL setup, where a single network is trained one example at a time with SGD.
- EWC (Kirkpatrick et al., 2017): Elastic Weight Consolidation is a regularisation based method which constraints the weights important for the previous tasks to avoid catastrophic forgetting.
- GEM (Lopez-Paz & Ranzato, 2017): Gradient Episodic Memory does constrained optimisation by solving a quadratic program on the gradients of new and replay samples, trying to make sure that these gradients do not alter the past tasks' knowledge.
- MER (Riemer et al., 2019): Meta Experience Replay samples i.i.d data from a replay memory to meta-learn model parameters that show increased gradient alignment between old and current samples. We evaluate against this baseline

in the LLL setup.

On the real-world visual classification datasets, we carry out experiments on GEM, MER along with:-

- ER: Experience Replay uses a small replay buffer to store old data using reservoir sampling. This stored data is then replayed again along with the new data samples.
- IID: Network gets the data from all tasks in an independent and identically distributed manner, thus bypassing the issue of catastrophic forgetting completely.
- iCARL (Rebuffi et al., 2017): iCarl is from the family of class incremental learners, which learns to classify images in the metric space. It prevents catastrophic forgetting by using a memory of exemplar samples to perform distillation from the old network weights.
- A-GEM (Chaudhry et al., 2019): Averaged Gradient Episodic Memory proposed to project gradients of the new task to a direction such as to avoid interference with respect to the average gradient of the old samples in the buffer.
- Meta-BGD: Bayesian Gradient Descent (Zeno et al., 2018) proposes training a bayesian neural network for CL where the learning rate for the parameters (the means) are derived from their variances. We construct this baseline by equipping C-MAML with bayesian training, where each parameter in θ is now sampled from a gaussian distribution with a certain mean and variance. The inner-loop stays as in C-MAML(constant LR), but the magnitude of the meta-update to the parameters in θ is now influenced by their associated variances. The variance updates themselves have a closed form expression which depends on m monte-carlo samples of the *meta-loss*, thus implying m forward passes of the inner-and-outer loops (each time with a newly sampled θ) to get m meta-gradients.