Université de Montréal

Look-Ahead Meta-Learning for Continual Learning

par

Gunshi Gupta

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en apprentissage automatique

July 26, 2020

Sommaire

Le problème “d’apprentissage continu” implique l’entraînement des modèles profonds avec une capacité limitée qui doivent bien fonctionner sur un nombre inconnu de tâches arrivant séquentiellement. Cette configuration peut souvent résulter en un système d’apprentissage qui souffre de “l’oublie catastrophique”, lorsque l’apprentissage d’une nouvelle tâche provoque des interférences sur la progression de l’apprentissage des anciennes tâches. Les travaux récents ont montré que les techniques de “méta-apprentissage” ont le potentiel de réduire les interférences entre les anciennes et les nouvelles tâches. Cependant, les procédures d’entraînement ont présentement une tendance à être lente ou hors ligne et sensibles à de nombreux hyperparamètres. Dans ce travail, nous proposons “Look-ahead MAML (La-MAML)”, un algorithme de méta-apprentissage rapide basé sur l’optimisation pour l’apprentissage continu en ligne et aidé par une petite mémoire épisodique. Ceci est réalisé en utilisant l’équivalence d’un objectif MAML en plusieurs étapes et un objectif d’apprentissage continu “temps conscient”. L’équivalence résulte au développement d’un algorithme intuitif que nous appelons Continual-MAML (C-MAML), utilisant un méta-apprentissage continu pour optimiser un modèle afin qu’il fonctionne bien sur une série de distributions de données changeantes. En intégrant la modulation des taux d’apprentissage par paramètre dans La-MAML, notre approche fournit un moyen plus flexible et efficace d’atténuer l’oubli catastrophique par rapport aux méthodes classiques basées sur les prieurs. Cette modulation a également des liens avec des travaux sur la métadescendance, que nous identifions comme une direction importante de la recherche pour développer de meilleurs optimiser pour un apprentissage continu. Dans des expériences menées sur des repères de classification visuelle du monde réel, La-MAML atteint des performances supérieures aux autres approches basées sur la relecture, basées sur les prieurs et basées sur le méta-apprentissage pour un apprentissage continu. Nous démontrons également qu’elle est robuste et plus évolutive que de nombreuses approches de pointe.

Mots clés: apprentissage tout au long de la vie, e-learning, méta-apprentissage, modulation du taux d’apprentissage

Summary

The continual learning problem involves training models with limited capacity to perform well on a set of an unknown number of sequentially arriving tasks. This setup can often see a learning system undergo *catastrophic forgetting*, when learning a newly seen task causes interference on the learning progress of old tasks. While recent work has shown that meta-learning has the potential to reduce 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. This is achieved by realising the equivalence of a multi-step MAML objective to a *time-aware* continual learning objective adopted in prior work. The equivalence leads to the formulation of an intuitive algorithm that we call *Continual-MAML (C-MAML)*, employing continual meta-learning to optimise a model to perform well across a series of changing data distributions. By additionally incorporating the modulation of per-parameter learning rates in *La-MAML*, our approach provides a more flexible and efficient way to mitigate *catastrophic forgetting* compared to conventional *prior-based* methods. This modulation also has connections to prior work on *meta-descent*, which we identify as an important direction of research to develop better optimizers for continual learning. In experiments conducted on real-world visual classification benchmarks, *La-MAML* achieves performance superior to other replay-based, prior-based and meta-learning based approaches for continual learning. We also demonstrate that it is robust, and more scalable than many recent state-of-the-art approaches.

Keywords: Continual Learning, Online Learning, Meta-Learning, Learning Rate Modulation

Contents

List of tables

List of figures

Dédicaces

I am grateful to my advisor Prof. Liam Paull for all the freedom, support and guidance he provided throughout the course of my master's program. His philosophy towards research and emphasis on good formulation inspires us to hold our work to high standards and ensure that it advances the field in a meaningful way. I'd like to express my deep gratitude towards my co-author and friend Karmesh Yadav, whose good humor and curiosity made the whole process fun, and who shall be a part of many collaborations to come. I was fortunate to have the company of very talented and welcoming lab mates at the REAL lab, including Krishna, Sai, Dishank, Dhaivat, Vincent, Bhairav, Manfred, Breandan, Florian, Mark and others.

I was also very fortunate to have been around a very talented peer group at Mila including people like Felipe Codevilla, Matt Riemer, Alexandre Piche, Shagun Sodhani, David, Martin, Sanjay Thakur, Rim Assoeul, Samarth Sinha, interactions with whom significantly shaped my thoughts and views in academic matters. Many thanks are also due to the very competent and supportive tech and admin teams, and Olexa who continues to save the lab years of debugging and trouble-shooting everyday. Doing deep learning research was made possible only because of access to the many compute resources that Mila has made available to us.

Montreal would not have become home so quickly if not for my flatmate Disha and our friendly neighbour Julie. Lastly I'm grateful for my friends: Disha, Rithesh, Varsha, Sandeep, Ankit and Max; who make everything better and for my parents who make everything possible.

Chapter 1

Introduction

Deep Learning (DL) has stood the test of time as a machine learning solution to reasoning problems for which massive data and compute can be harnessed. While we have witnessed the great success of learning-based systems for tasks like recognition and translation, significant engineering effort goes into the deployment and maintenance of deep models that underlie these systems. This is partially due to the fact that there is a dissonance between the settings in which models are trained and the settings in which they are deployed. While models are trained on *independent and identically distributed (i.i.d)* data by repeatedly shuffling the training samples of a stationary dataset, a deployed system often sees data arriving in a highly temporally correlated fashion. The conditions required by stochastic gradient descent (SGD) are thus violated in the real-world.

As an example, consider the case of an image tagging system interacting with a user to recommend tags for recurring people in the images of an album. The user provides sparse supervision to the system by tagging some people in the initial images. It is expected that the system should quickly adapt and suggest correct tags for these people if they are present in later photos. However, it is also implied that in this process of adaptation, it should not degrade its tagging performance on other people who might also have been present in some previous albums and for whom this system was already accurate before adaptation. In other words, intelligent agents must possess the ability to incorporate new concepts on-the-go from data that naturally arrives sequentially in the real world. While *fast adaptation* is desired, we are usually interested in adaptation that retains useful priors and knowledge learned in the past. This process is referred to as *Continual Learning* (CL) [**34**], and is essential for a learning system to accumulate knowledge and skills over time. Examples of settings where this style of learning is essential include recommender systems that need to continually adapt to user browsing patterns, perception models that need to recognise an open-set of entities in the real-world, home robots that need to learn an ever-increasing set of behaviours and maneuvers and so on. This applies to most systems deployed and acting online in a non-stationary environment. In this way, continual learning also brings us a step closer to correctly addressing the problems of embodied or situated learning agents when they are deployed in the world.

For an online system to get better over time, it needs to keep updating its model using data seen at every time-step. At present, gradient-based learning algorithms are not sophisticated enough to do this over long time-periods because they experience *Catastrophic Forgetting* [**13, 29**], one of the biggest challenges in this setup. It can be described as the phenomenon where a model undergoes many correlated gradient updates on an incoming stream of data, in the process undoing its learning progress on previously seen data. This degradation is due to the fact that gradient directions on old and new data can clash, i.e., there is *gradient interference* between samples learned at different points in time. In the case of infinite storage, this might not be a problem since we could store all the data that arrives and interleave it with any incoming data at a later time for re-training. Since this might not be a realistic provision, we are interested to develop solutions that can keep learning in real-time with limited compute and storage. This saves us from having to re-train models from scratch every time more data comes in, which is often desirable in many real-world applications for any of the following reasons:

(1) The data seen over the course of deployment of a system may be too large to make available in one location on a central server for retraining at any intermediate time.
(2) Re-training is both storage and compute-intensive, while it takes a negligible amount of compute to process data on-the-fly. On *edge-devices*, even if we could store most of the data seen by an agent on a remote server, there is still the issue of low bandwidth of communication being the limiting factor when the agent tries to request old data.
(3) The application in question may require low latency and/or quick adaptation on the part of the learning system, such that learnt knowledge is expected to be demonstrated in the immediate future and so there is no time to re-train. Additionally, there will always be incoming data in online scenarios and we want to make the most of it to stay up to date in this non-stationary world (as opposed to ignoring it and waiting for batch training).
(4) In the scenario of the data being of a user-sensitive nature, it might be undesirable to pool all user-data on a central server to retrain the model due to privacy concerns. It would be ideal if we could train a common base model from a large common dataset once and then deploy it on user devices, where it could continually learn and adapt to the user's changing behaviours while keeping the data on-device: for example as is attempted in the domain of *federated learning.* This way it benefits from the prior knowledge embedded in a large database, and still gets customised to user preferences without sacrificing privacy.

The problem of continual learning has received considerable attention over the past two decades with efforts focusing on reducing dependence on previously seen data due to storage and privacy concerns [**19, 37, 26, 10, 49**]. Algorithms for continual learning 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. **In this work we look at continual learning through the lens of optimisation, building on the insight that meta-learning algorithms can incentivise gradient alignment across task-wise objectives [38].** Besides aligning gradients, meta-learning algorithms show promise for continual learning since they can directly influence model optimisation to achieve auxiliary objectives like *generalisation* or *transfer.* This avoids having to define heuristic incentives like sparsity [**23**] to combat *catastrophic interference.* The downside is that they can be slow and hard to tune, effectively rendering them more suitable for *offline* continual learning [**16, 38**]. In this work, we overcome these difficulties and develop a gradient-based meta-learning algorithm for *efficient, online* continual learning. We summarise our contributions as follows[1]:

(1) We 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.* This objective allows us to integrate continual and meta-learning into the same loop, as part of a standard meta-learning procedure.

(2) We enhance the capabilities of C-MAML by incorporating a mechanism for the modulation of per-parameter learning rates (LRs) to pace the learning of a model across tasks and time. We refer to this as Look-ahead-MAML (La-MAML).

(3) We draw connections to literature on learning rate modulation through meta-descent and explain how it can be exploited to build more robust algorithms for online learning.

(4) Finally, we show that the algorithm is scalable, robust and achieves favourable performance on several benchmarks of varying complexity.

We start by describing the problem formulation, notation background concepts, in Chapter 2. We also outline common setups adopted by prior work in Continual Learning and give a detailed overview of prior work in this domain. In Chapter 3, we describe our proposed objective and related derivations, and the proposed algorithms. In Chapter 4, the evaluation setup, experiments and quantitative results are described. This is followed by a qualitative

[1]This work was submitted under the title *La-MAML: Look-Ahead Meta-Learning for Continual Learning* to the *Thirty-fourth Conference on Neural Information Processing Systems* (NeurIPS 2020) and is currently under review. Chapters 2, 3 and 4 largely overlap with the submitted work and have been augmented with explanations of background concepts and a more extensive literature review. The authors of the submitted paper are Gunshi Gupta, Karmesh Yadav and Prof. Liam Paull. Gunshi Gupta carried out the literature survey, problem formulation and algorithm development and equally participated in the tasks of implementation, experimentation, tuning, experimental setup and paper-writing for this work.

analysis of the robustness and scalability of the proposed algorithm. Finally, Chapter 5 concludes with a discussion on the future scope of this work, and some interesting directions to explore to get closer to building scalable and efficient algorithms for continual learning.

Chapter 2

Background

We start by formalising the problem definition for continual learning that we adopt in this work in Section 2.1. We also establish notation and review background concepts related to meta-learning that are essential in order to arrive at our proposed algorithm. We then describe the commonly used categorisations used to identify different setups in continual learning in Section 2.2. Finally, we provide an overview of related work in Section 2.3.

2.1. 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. Let τ_i be the i-th task we see, then we define X^i,Y^i $= \{(X^i_n, Y^i_n)\}_{n=0}^{N_i}$ as the set of N_i input-label pairs randomly drawn from it's data distribution D_i. An any time-step j during 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:

$$\sum_{i=1}^{t} \mathbb{E}_{(X^i,Y^i)}\left[\ell_i\left(f_i\left(X^i;\theta_0^j\right),Y^i\right)\right] = \mathbb{E}_{(X^{1:t},Y^{1:t})}\left[L_t\left(f\left(X^{1:t};\theta_0^j\right),Y^{1:t}\right)\right] \tag{2.1.1}$$

where ℓ_i is the loss for task τ_i and f_i is a learnt, mapping from inputs to outputs using parameters θ_0^j. [1] The loss $L_t = \sum_{i=1}^{t} \ell_i$ is the sum of all task-wise losses for tasks $\tau_{1:t}$ where t goes up from 1 to T, with T not known beforehand.

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

$$U\left(\theta_0^j\right) = \theta_1^j = \theta_0^j - \alpha\nabla_{\theta_0^j}\ell(\theta_0^j) = \theta_0^j - \alpha g_0^j \tag{2.1.2}$$

[1]We keep the zero subscript for the parameters here since we will later introduce the concept of *inner* and *outer* updates in meta-learning, and then the subscript and superscript will correspond to the index of the *inner* and *outer* updates made to θ respectively, where we will associate each time-step j with one *outer-update* .

where $g_0^j = \nabla_{\theta_0^j} \ell(\theta_0^j)$. $U\,()$ 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 vector learning-rate. $U\left(\cdot, x_i\right)$ implies gradient updates are computed by evaluating the loss using data x_i:

$$U\left(\theta_0^j, x_i\right) = \theta_1^j = \theta_0^j - \alpha \nabla_{\theta_0^j} \ell(\theta_0^j, x_i) \tag{2.1.3}$$

From here onwards, we will omit the data argument in most places so that $\ell_i(\cdot)$ implies that the loss ℓ_i for task τ_i is evaluated on some data $(x,y) \sim (X^i,Y^i)$ from that task.

2.1.1. Meta-learning

Meta-learning [**41**], or *learning-to-learn* [**46**] has emerged as a popular approach for training models amenable to fast adaptation on limited data. It works by learning algorithmic priors or parameters using the data of training tasks, to enable faster learning on related unseen test tasks. For instance, these priors could be learnt in the form of a parameter initialisation [**11**] or in the form of a learning rule encoded in the weights of a recurrent neural network [**36**].

The process of meta-learning typically involves two phases:

(1) *meta-train phase* : In this phase the algorithm gathers useful inductive biases from training tasks and is analogous to the training phase in standard learning. It involves splitting the training tasks' data into training and testing splits so the model can validate and modulate what it learns on the *meta-train-train* set, based on how well it generalises to the *meta-train-test* set.
(2) *meta-test phase* : This phase involves adapting and testing the learnt parameters on the unseen test tasks and it is analogous to the testing phase in standard learning. The adaptation involves learning from a handful of samples from the unseen task, starting from the solution learnt in the *meta-train* phase. This procedure tests the quality of priors learnt in the *meta-train* phase, reflected in how well the model is able to use them to adapt to a new, but related task.

Optimisation or Gradient-based Meta-Learning (GBML) learns a meta-initialization θ_0^j for a class of parametrized functions $f(\theta) : X \rightarrow Y$ such that a small number (k) of stochastic gradient steps on a new task's samples suffice to learn good task-specific model parameters θ_k^j. Therefore gradient-based learning rules like SGD play the role of the adaptation procedure that happens in the *meta-test phase*. GBML relies on the differentiability of these learning rules to adjust the initialisation θ_0^j at time-step j, to a new initialisation θ_0^{j+1} at time-step $j+1$, based on some application-defined *fitness* measure of the parameters θ_k^j reached after *few-shot adaptation* from θ_0^j.

There is an implicit assumption here that train and test tasks are related enough that their respective optima lie close by, and are reachable in a few gradient steps from some common initialisation θ_0^j. If this is not the case, then the few shot adaptation assumption at meta-test time is too restrictive to reach optimal parameters for the new task. Next, we introduce the gradient-based meta-learning algorithms MAML [**11**] and OML [**16**], that we build upon in Chapter 3.

2.1.1.1. Model-Agnostic Meta-Learning (MAML). MAML [**11**] is one of the simplest GBML algorithms that optimises model parameters to be good at few-shot generalisation on a set of related tasks. It does this by repeatedly unrolling a gradient-based optimiser from an initialisation θ_0^j on samples from a set of tasks, to obtain a new set of parameters. It back-propagates the error of these new parameters on held-out samples from these tasks, all the way through the unrolled updates to obtain a better starting point for the next *unrolling.*

We review some common terminology used in MAML: 1) at a given time-step j during training, model parameters θ_0^j (or θ_0 for simplicity), are often referred to as an *initialisation*, since the aim is to find an ideal starting point for few-shot gradient-based adaptation on unseen data. 2) *Fast* or *inner-updates*, refer to gradient-based updates made to a copy of θ_0, optimising some 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 gradient update (or *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 the *meta-gradient* $\nabla_{\theta_0} L_{meta}(\theta_k)$. MAML thus optimises θ_0^j at time j, to perform optimally on tasks in $\{\tau_{1:t}\}$ after undergoing a few gradient updates on their samples. It optimises in every *meta-update*, the objective [2] :

$$\min_{\theta_0^j} \mathbb{E}_{\tau_{1:t}} \left[L_{meta} \left(U_k(\theta_0^j) \right) \right] = \min_{\theta_0^j} \mathbb{E}_{\tau_{1:t}} \left[L_{meta}(\theta_k^j) \right] \tag{2.1.4}$$

The MAML objective in Eqn 2.1.4 is often approximated by a first-order version in practice, which is referred to as First-Order MAML (FOMAML). This is because computing the hessian for a neural network is expensive and therefore in FOMAML the second-order information is ignored and the hessians are assumed to be zero. The *meta-gradient*, which denotes the gradient of the meta-loss with respect to the initial parameters, is different from the gradient of the meta-loss at the parameters at the end of the inner loop(θ_k^j). However, the first-order version of MAML simply treats both of them as equal, since it assumes that the derivative of the chain of updates in the inner-loop with respect to the initial weights is now equal to the *identity* vector. This often leads to very negligible degradation in performance, since networks with ReLU activations are linear almost everywhere, and already have almost zero hessians in many parts of the optimisation landscape.

[2]In this work we will consistently use the following notation for indices: t denotes the number of tasks seen till any point of time j during training. i denotes the index ranging over these t tasks. We will associate each time-step j with one full *meta-update*, where a *meta-update* is typically compose of multiple *inner-updates* followed by a single *outer-update*. Therefore j also denotes the index iterating over the *outer-updates* made during training. k denotes the number of inner updates we do in each *meta-update* and k' denotes the index iterating over the k inner updates.

The gradient expressions (or *meta-gradients*) of different meta-learning algorithms MAML, FOMAML and Reptile are derived in [**33**]. These terms, denoted as g_{MAML}, g_{FOMAML} and g_{Reptile} (assuming we take k fast-updates before every *slow-update*), can be broken down into the following terms:

$$\begin{aligned}
\mathbb{E}\left[g_{\text{MAML}}\right] &= (1)AvgGrad - (2(k-1)\alpha)AvgGradInner \\
\mathbb{E}\left[g_{\text{FOMAML}}\right] &= (1)AvgGrad - ((k-1)\alpha)AvgGradInner \\
\mathbb{E}\left[g_{\text{Reptile}}\right] &= (k)AvgGrad - \left(\frac{1}{2}k(k-1)\alpha\right)AvgGradInner
\end{aligned} \tag{2.1.5}$$

where the term $AvgGrad$ is the gradient of the meta-loss *at the initialisation parameters*, and responsible for its minimisation *at the initialisation parameters*. The other term $AvgGradInner$ pushes the initialisation parameters in a direction where the gradients of the inner and outer loop losses align more. As we mentioned before in Chapter 1, this is an appealing property to have in a continual learning algorithm as well. This is because increasing alignment across different objectives can go a long way in mitigating catastrophic interference between them in the future.

As mentioned before, meta-learning is typically employed for the task of few-shot generalisation, i.e, how to perform better on a new related task having seen a handful of samples from its training distribution. **The key assumption here is that testing and training distributions are related, and learning priors at meta-train time allows the model to do efficient few-shot adaptation at test time.** In this setup, usually a new set of tasks is seen at test time, *once* (there is no repeated addition of new tasks like in continual learning), and we do not care about the performance on train-time tasks once we have adapted to new tasks. While few-shot generalisation is an appealing property in continual learning systems, it is not directly apparent how few-shot meta-learning formulations can be used here. We will now describe how MER [**38**] takes a step in this direction.

2.1.2. Equivalence of Meta-Learning and Gradient Alignment Objectives

Gradient Episodic Memory (GEM) [**28**] investigated the connection between task-wise parameter sharing and forgetting in continual learning and developed an algorithm to explicitly minimise *gradient interference* between tasks, when learning a new task. This is an objective that meta-learning algorithms can implicitly optimise for. The first-order gradient-based meta learning algorithm Reptile was proposed in [**33**], which derived the gradients of different meta learning objectives and showed how they incentivise alignment (we stated the expressions for these gradients in the equations 2.1.5). *Meta Experience Replay* (MER) [**38**] formalized the transfer-interference trade-off in cotninual learning. It showed that the continual learning objective of minimising loss on tasks $\tau_{1:t}$ seen till time j, while aligning gradients between them, is equivalent to the meta-learning objective of the Reptile algorithm, i.e.:

$$\min_{\theta_0^j}\left(\sum_{i=1}^{t}\left(\ell_i(\theta_0^j)\right)-\alpha\sum_{p,q\leq t}\left(\frac{\partial\ell_p\left(\theta_0^j\right)}{\partial\theta_0^j}\cdot\frac{\partial\ell_q\left(\theta_0^j\right)}{\partial\theta_0^j}\right)\right)=\min_{\theta_0^j}\mathbb{E}_{\tau_{1:t}}\left[L_t\left(U_k(\theta_0^j)\right)\right] \tag{2.1.6}$$

where the loss $L_t = \sum_{i=1}^{t}\ell_i$ indicates the performance on all tasks $\tau_{1:t}$ seen till any time-step j during training.

Given this equivalence, MER proposes a continual learning algorithm that follows the same training procedure as that of Reptile. It does this by maintaining a replay buffer to store a random subset of the incoming data through reservoir sampling. This data is then sampled periodically and interleaved with incoming data to make *meta-updates* to the parameters using Reptile. This influences the optimisation to move towards parameters that decrease the loss on all data samples used in the *fast-updates*, while aligning their gradients with each other.

We will similarly show how a different, more data-efficient version of this objective coincides with the multi-step MAML algorithm we propose. Note that this implies that the procedure to learn an *initialisation* through meta-learning, coincides with the procedure to learn optimal parameters during continual learning. This means that we can directly treat the initialisation as the optimal learned parameters without needing to do any test-time adaptation. Thus we will be employing meta-learning to influence the optimisation of a continually training model.

2.1.3. Online-aware Meta-Learning (OML)

An offline meta-learning algorithm was proposed in **[16]** to pre-train a *Representation-Learning Network (RLN)* to provide a representation suitable for continual learning to a *Task-Learning Network (TLN)*. The RLN's representation is trained using *catastrophic forgetting as the learning signal.* This is done by differentiating through the online updates of the TLN to change the RLN such that its output representation evolves to become suitable to be input to a network that uses it for online updates. Data from a fixed set of tasks (τ_{val}), is repeatedly used to evaluate the RLN and TLN, to test how well they remember or perform on τ_{val}, after the TLN has undergone temporally correlated updates on some new task. 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 a batch sampled from τ_{val} along with the current task's data. This loss is then differentiated to make *slow updates* to the TLN and RLN. **This composition of two losses to simulate online learning in the inner loop and test *forgetting* in**

the outer loop, is referred to as the *OML objective*. It carries out many correlated *fast* updates to the TLN and relies on the *slow updates* to the RLN to eventually provide a better representation to the TLN for continual learning.

The continual learning performance of the method is evaluated at *meta-test* time. This is done by fixing the RLN parameters, and continually training a randomly initialised TLN over sequential tasks to now test how much performance it retains at the end. This work showed that we could potentially use the meta-objective to achieve the continual learning behaviour we desire *directly*. It also empirically showed that by optimising a representation with the OML objective, the emergent property of sparsity is observed in the representation. This was appealing since it did not have to be explicitly incentivised like in [**23**]. While this work showed good results on a toy dataset, this would not directly work in data streams with the complexity of the real world since the fixed representation of the RLN would not be enough to differentiate new concepts.

2.2. Taxonomy

Research in the field of Continual Learning aims to address and bring us closer to the online learning setting that learnt models are eventually deployed in. Therefore there are as many settings studied in prior work as there are deployment scenarios, reflecting a broad range of real-world scenarios with different assumptions and budgets, some more constrained than others. Continual Learning setups and solutions for them are usually roughly categorised based on the following questions:

(1) *What kinds of concepts are being added incrementally?*
   (a) *Task Incremental*: This setup assumes that we want to learn a sequence of mutually exclusive tasks, for example, many separate 5-way classification tasks, and that the data for each task arrives one after another (with the possibility of revisiting tasks seen in the past). This setup is currently simulated by splitting the classes of a dataset into disjoint sets of classification tasks, and processing the data associated with each task in a randomly-picked sequential order.
   (b) *Domain Incremental*: In this setup the model learns a series of datasets or domains, all corresponding to the same task. For instance, consider an object detection system trained to detect a fixed set of classes from crowd sourced imagery. Here, differences in user-cameras with unknown calibration parameters will lead to a non-trivial domain shift problem that is usually not modeled by simple input perturbations. And therefore each time we get a camera-feed from a user, the imagery corresponds to a new domain (associated with the same task).
   (c) *Class Incremental*: This setup, specific to the classification problem, aims at learning a large classifier to which new classes are continually added to construct

a more difficult classification task with more classes to discriminate amongst. Often the major issues here are due to bias in the last layer which rapidly overfits to boost the scores for the classes that were seen most recently.

(2) *What auxiliary information do we have about the data coming in?*
   - (a) *Task-aware*: In this setup, we know when one task ends and when another begins, i.e, task boundaries are known at both train and test time. This could be possible when a system knows what high-level regime it is working within, for example, home-robots could treat different locations or rooms as a new task. This basically implies that we have knowledge of the context that a system is functioning in. A recommender system might have different usage or browsing patterns to learn, conditional on whether it is weekend or weekday, and so this conditional context variable could decide the task identity.
   - (b) *Task-agnostic*: In this setup, the data is seen as one long, continuous stream without any task segmentation. This might be more realistic for certain learning setups, as it comes with fewer assumptions about the information available to an agent in unstructured environments. For instance, a recommender system that might want to show content according to the different moods of the user does not actually know what the real labels corresponding to the user's moods at any time-step are. Therefore, these approaches often have to do context detection (or task inference) as part of the algorithm.

(3) *What constraints are there on how incoming data can be processed?*
   - (a) *Batch-Within-Online* or the *Multiple-Pass* setup: Here, we assume that each task becomes available to us as an *i.i.d.* dataset that we can temporarily store. Therefore we can normally carry out mini-batch training with SGD for many epochs, since the data is locally stationary and i.i.d. *within* a task.
   - (b) *Online-Within-Online* or the *Single-Pass* setup: This is the streaming setting where we assume that the data arrives as a continuous non-stationary stream of a few samples at a time. Here, we would not be able to go back and re-process any data even if it belongs to the same task that we are still seeing. This setup is also sometimes referred by the name *Efficient Lifelong Learning (LLL)*[**8**]. As we move towards developing more data-efficient algorithms, we will want to move to this fully-online setup.

Although all of the above categorisations represent valid and plausible setups, some setups like *batch-within-online* can be adequately handled by algorithms addressing harder setups like *online-within-online*, where we are only allowed to look at the data *once* instead of storing it for repeated training. In general, algorithms that work for harder, more constrained setups can usually work across more privileged setups with bigger budgets as well, and therefore research is gravitating towards the former. From here on, we will directly refer to these setups by their names.

2.3. Related work

Most continual learning approaches can be roughly categorized into one of the following:
(1) *distillation-based approaches*
(2) *replay-based approaches*
(3) *regularisation approaches*
(4) *meta-learning-based approaches*
(5) *expansion-based approaches.*
They differ in how they choose to respond to changes in the data distribution, their mode of optimisation, etc. and come with their own trade-offs between forward and backward transfer, compute, storage overhead and so on. We describe each of these here, and will subsequently compare against some approaches from each of these categories, as baselines in Chapter 4.

2.3.1. Distillation-Based Approaches

Distillation or rehearsal-based approaches [**26, 37**] attempt to distill the knowledge of *older* versions of a model into later versions as it undergoes continual training. This is done by storing snapshots of the model at the end of each task, and using their outputs as pseudo-labels later to supervise future model outputs while training on new tasks. This is the same technique as *knowledge distillation* [**15**], which is used in model compression research. The distinction is that in the continual learning setup, distillation is done only on incoming data instead of old data. This has primarily been studied for supervised classification problems where new classes or sets of classes are being added to a model. Under this scheme, the model is trained using an additional loss based on the discrepancies between the *logits* produced by the old and new models for the classes from the old tasks. This carries out a form of hypothesis-matching since we want the *logit* distribution on old classes to be consistent between the old and new models, even on new data that is unrelated to the old tasks. Newer works such as AGEM [**8**], EWC [**19**], iCarl [**37**] are now trying to avoid this procedure of storing copies of the model since it assumes we have a large storage memory available to us, which is often not the case.

2.3.2. Replay-Based Approaches

In order to circumvent the issue of catastrophic forgetting, *replay-based methods* maintain a small collection of samples from previous tasks in memory. Some approaches utilise an *episodic-buffer* or a *replay-buffer* [**7, 37**] to sample old data points and interleave them with new data to mimic the *i.i.d.* setup within continual learning. Others like GEM [**28**] and A-GEM [**8**] take replay memory samples into account to determine altered *low-interference* gradients for updating parameters while learning new tasks. Some works later studied which samples to keep in the buffer based on their contribution to the sample diversity within their task, or how their gradients align with new tasks [**4**].

Generative-replay [**44**] trains generative models to be able to replay samples from old tasks. The motivation to do this is that generative models might yield more diversity in their replay than is possible by storing just a handful of past samples. However the memory trade-off is currently not favourable for this approach, since generative models themselves rack up a large number of parameters, and don't perform well enough to justify the overhead. This also currently faces scalability concerns arising from the difficulty of modeling complex non-stationary distributions, such as those seen in real-world imagery.

For methods aiming to learn on huge dataset sizes, a large number of tasks, or classes with many samples per class, a replay buffer is often very effective. It is also necessary to achieve any kind of backward transfer.

2.3.3. Regularisation-Based Approaches

Regularisation-based methods avoid using replay at all by constraining the network weights according to heuristics intended to ensure good performance in continual learning scenarios. This involves enforcing weight or representational sparsity [**5, 23**] to ensure that only a subset of neurons remain active and receive gradients at any point of time. For instance, HAT [**43**] learns a task dependent attention mechanism on the parameters of a model, concurrently with parameter learning. The attention masks are conditioned on those of previous tasks so that they can have low overlap amongst them. These mechanisms aim to reduce the possibility of catastrophic interference across tasks.

Prior-focused approaches are a subset of *regularisation-based* approaches that try to preserve the performance on old tasks by either slowing down or penalising the change of parameters deemed important for performance on these tasks [**20**]. This includes many bayesian approaches for continual learning since the bayesian framework lends itself naturally to the online nature of the problem. For instance Variational Continual Learning (VCL) [**32**] simply carries out a bayesian update of the weights by taking the previous time-step's posterior as prior and computing the likelihood over the new task's dataset to get the new posterior. It also depends on *coresets* (in other words, a replay memory) to achieve best results. Bayesian Gradient Descent (BGD) [**49**] proposed a closed-form update of the weights of a bayesian neural network (BNN) in which the learning rate of the means is proportional to the associated variances of the parameters. UCB [**10**], concurrent to BGD, proposed training a BNN where the learning rate for the parameters are explicitly modulated based on the associated variances of the BNN. The key idea amongst these approaches is that for parameters which crucially affect the model's performance on a task, the variances should be lower and therefore can be used to lower their learning rates.

2.3.4. Meta-Learning-Based Approaches

Meta-Learning-based approaches are fairly recent and have shown impressive results on small benchmarks like Omniglot and MNIST [**24, 22**]. MER [**38**], inspired by GEM[**28**], proposes a meta-learning algorithm that utilises replay to incentivise alignment of gradients between all the tasks. Online-Aware Meta Learning (OML) [**16**] introduces a meta-objective for a pre-training algorithm to learn an optimal representation *offline*, which is subsequently frozen and used for continual learning. We will review these in more detail in Chapter 2.1.

Orthogonal setups are investigated in [**2, 12, 30**], 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. This is more commonly referred to as the problem of online learning and is often confused with continual learning because of inconsistent terminology used in prior work. Online learning aims to find the best possible initialisation to start from, to learn a new task by making the best use of old data to find a good initialisation. This is similar to meta-learning, with the added condition that it is tested over many sequential phases where unseen tasks are added, as opposed to a single meta-test phase in meta-learning.

Meta learning approaches themselves can be divided into one of two categories:

(1) *Meta-Continual learning*, where a meta-learning algorithm is used to learn a prior for the model parameters which are subsequently used for adaptation only (no further meta-learning) [**16, 31**].

(2) *Continual-Meta Learning*, where continual learning happens with meta-learning in the loop. In other words, there is lifelong meta-learning of the parameters for continual learning. While [**38**] falls into this category, it is prohibitively slow and therefore almost offline. **In this work we attempt to develop an online algorithm for this setting, that can leverage a small replay memory to achieve positive backward transfer.**

2.3.5. Expansion-Based Approaches

Expansion Based approaches [**39, 3, 47**], orthogonal to the above approaches, avoid forgetting by freezing a subset of parameters that were trained on earlier tasks while adding more capacity dynamically to the model during training. While there usually isn't a possibility of forward or backward transfer since mutually exclusive parts of a model are responsible for different tasks or data, there is also no interference.

Non-parametric Approaches: More recently, some approaches have investigated task-free continual learning where an ensemble of learnt models is grown over time as needed, in a *task-agnostic* setup. These approaches typically start off as one sub-network and rely on Dirichlet Processes (DP) to add capacity by spawning more sub-networks. The DP prior helps to decide when to add another component to the ensemble, based on how well a new sample is *explained* or *fit* by the current ensemble members. They maintain a buffer of samples not explained well by any of the current members and when they reach a threshold, they are assigned to a new sub-network. This is to prevent the tendency to over-add sub-networks, which would otherwise lead to explosion of capacity especially in the beginning of tasks when the losses on most samples are high. This however leads to needing to maintain many hyper-parameters, like thresholds, to decide when it is time to expand. Even in the absence of defined task boundaries, each member of the ensemble can be thought to be responsible for a subset of inter-related samples from the incoming data, i.e, a *concept*.

To decide which sub-network to use to generate the model output and backpropagate loss to for a particular input, they train a classifier over the mixture of models. This classifier essentially does task or concept inference to decide assignment to a member of the ensemble, similar to the EM procedure. This is an approximation of the actual nested optimisation since it only optimises over the new data points whereas EM formulations like clustering optimise over all data in each iteration. Approaches that train a classifier for task inference are essentially training in a *class incremental* fashion, as new tasks keep getting added to the old ones. While there is no forgetting in the sub-networks as they share no parameters across different tasks, this common classifier itself could face catastrophic forgetting. This underscores why forgetting is an omnipresent phenomenon across approaches and needs to be addressed.

CURL [**35**] follows the framework described above. It uses a shared encoder to map inputs to a latent representation, which is used to infer the task by modelling a Gaussian Mixture Model on the latent code. The model dynamically expands its capacity to capture new tasks, and incorporates generative replay to minimise catastrophic forgetting, since they have a shared encoder across tasks, which could experience *forgetting*. CN-DPM [**25**], similar to CURL, also uses generative replay, model expansion and a short-term memory to store unexplained data points. Unlike CURL however, it uses task-wise experts that incorporate a generative model, so there is no shared capacity that could experience forgetting. Some approaches like MOLe [**31**], MOCA [**14**] [**17**] use meta-learning to better initialise the weights of every spawned sub-network so that it is amenable to fast adaptation during continual learning. In contrast, we aim to formulate an algorithm that *continually* meta-learns optimal parameters which become increasingly resistant to *forgetting* over time.

Chapter 3

Proposed approach

In the previous sections, we have described the properties we would like to have in our continual learning algorithm. *We reviewed the OML objective and noted that it is able to directly simulate online behaviour in the inner loop of meta-learning.* It modifies the representation over the course of training by testing whether the model underwent forgetting. It relies on a MAML-derived algorithm to learn a representation fit to bootstrap continual learning, and sees desirable emergent properties like sparsity in the representation, that make it suitable for continual learning.

We also discussed that meta-learning algorithms are able to incentivise learning with gradient alignment across objectives and how MER exploited the similarity between its continual learning objective and that of Reptile. Even though its objective is meant to be optimised *online* for continual learning, it is prohibitively slow in practice.

We want to develop an online algorithm optimising a principled continual learning objective that somehow directly regulates continual learning behaviour like OML, to have similar emergent properties. This naturally leads us to formulate a MAML-based algorithm that optimises the OML objective for *all* the parameters of a network *online*. We derive the equivalence of this objective to a continual learning objective adopted in prior work. It is important to prove this equivalence, because it is otherwise unprincipled to use a few-shot meta-learning formulation like MAML without intending to do any test-time adaptation. This is because the continual learning problem setting typically does not accommodate any assumption about the availability of small numbers of unseen samples for test-time adaptation. We will go about modifying the OML objective for an online setting through the following steps:

(1) We first define a similar OML objective for the *online* optimisation of all the parameters of a model, using k-step MAML. In contrast, the original OML objective **[16]** learns a static representation which is fixed at continual learning time.

(2) We replace the data from τ_{val} with a replay buffer containing samples from old tasks to ensure the signal from the meta-loss is now aimed at remembering old tasks.

(3) We show the equivalence of this objective to a well known objective for continual learning that is *time-aware*, ie, it aligns the current task gradients with those of the past tasks. It thus treats the current task in a special manner as opposed to treating all tasks equally as done in [**38**].

(4) We will later also incorporate a notion of adaptivity in learning by proposing a mechanism for the modulation of per-parameter learning rates.

We start by describing *Continual-MAML* (C-MAML), our proposed base algorithm for online continual learning along with the associated proof for the validity of its objective in Sections 3.1. In Section 3.2, we then propose the learning rate modulation scheme which we augment C-MAML with, and the resulting algorithm is referred to as Look-Ahead MAML (La-MAML). The derivations corresponding to the learning rate updates is detailed in subsection 3.2.1. We then draw connections between our proposed *look-ahead updates* and prior work on *meta-descent* and learning-rate modulation in subsection 3.2.2.

3.1. C-MAML

C-MAML aims to optimise the OML objective *online*, so that learning on the current task does not lead to forgetting on previously seen tasks. We define this objective, adapted to optimise a model's parameters θ_0^j instead of a representation at time-step j, as:

$$\min_{\theta_0^j} \mathrm{OML}(\theta_0^j, t) = \min_{\theta_0^j} \mathbb{E}_{\mathcal{S}_k^j \sim D_t}\left[L_t\left(U_k(\theta_0^j, \mathcal{S}_k^j)\right)\right] = \min_{\theta_0^j} \mathbb{E}_{\mathcal{S}_k^j \sim D_t}\left[L_t\left(\theta_k^j\right)\right] \tag{3.1.1}$$

$$where$$

S_k^j is a small stream of data tuples $(x_{j:j+k}, y_{j:j+k}) \sim (X^t, Y^t)$ from the current task τ_t that is being 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. 2.1.1 for tasks $\tau_{1:t}$. Note that we omit the implied data argument $(x, y) \sim (X^i, Y^i)$ that is the input to each loss ℓ_i in L_t for any task τ_i. We will show in the next section that optimising our objective in Eq. 3.1.1 through the k-step MAML update in C-MAML also coincides with optimising the continual learning objective of AGEM [**8**]:

$$\min_{\theta_0^j} \mathbb{E}_{\mathcal{S}_k^j \sim D_t}\left[L_t\left(U_k(\theta_0^j, \mathcal{S}_k^j)\right)\right] = \min_{\theta_0^j} \mathbb{E}_{\mathcal{S}_k^j \sim D_t}\left[\sum_{i=1}^{t}\left(\ell_i(\theta_0^j) - \alpha \frac{\partial \ell_i\left(\theta_0^j\right)}{\partial \theta_0^j} \cdot \frac{\partial \ell_t\left(\theta_0^j\right)}{\partial \theta_0^j}\right)\right] \tag{3.1.2}$$

This differs from MER's objective in Eq. 3.1.3 by being *asymmetric* or *time-aware*: it focuses on aligning the gradients of the current task τ_t and the average gradient of all tasks

$\tau_{1:t}$ instead of aligning all the pair-wise gradients between tasks $\tau_{1:t}$. It can be argued that the increasing the former kind of alignment also indirectly leads to increase in the latter kind of alignment, but in a more data-efficient manner. We rewrite the objective from MER here in the same format for easier comparison:

$$\min_{\theta_0^j} \mathbb{E}_{\mathcal{S}\sim\tau_{1:t}}\left[L_t\left(U_k(\theta_0^j,\mathcal{S})\right)\right] = \min_{\theta_0^j} \mathbb{E}_{\tau_{1:t}}\left[\sum_{i=1}^{t}\left(\ell_i(\theta_0^j)\right) - \alpha \sum_{p,q\leq t}\left(\frac{\partial \ell_p\left(\theta_0^j\right)}{\partial\theta_0^j}\cdot\frac{\partial \ell_q\left(\theta_0^j\right)}{\partial\theta_0^j}\right)\right] \tag{3.1.3}$$

Here, $\mathcal{S}$ is data sampled from a replay buffer containing old tasks' samples and augmented with a sample from the current task. Therefore MER uses data samples from all the tasks seen so far, to take a single gradient step in such a way that the gradient alignment between all of them is increased (according to a first-order approximation). Later in Section 4.3.1, we will show empirically that gradient alignment amongst old tasks doesn't degrade while a new task is learnt, thus avoiding the need to repeatedly optimise the inter-task alignment between all of them. This results in a drastic speedup over MER's objective (Eq. 3.1.3) which tries to align all $\tau_{1:t}$ equally, thus resampling 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.

During training, a replay-buffer R is populated through *reservoir sampling* on the incoming data stream as in [**38**]. At the start of every meta-update, a batch b is sampled from the current task. Batch b is also combined with a batch sampled from R to form the *meta-batch*, b_m, which reflects samples from both old and new tasks. θ_0^j is updated through k SGD-based *inner-updates* by seeing the current task's samples from b one at a time. The outer-loss or *meta-loss* $L_t(\theta_k^j)$ is evaluated on b_m. It indicates the performance of parameters θ_k^j on all tasks seen till time j. A single *meta-update* of C-MAML is visually depicted in Figure 3.1 and the complete training procedure is described in Algorithm 1.

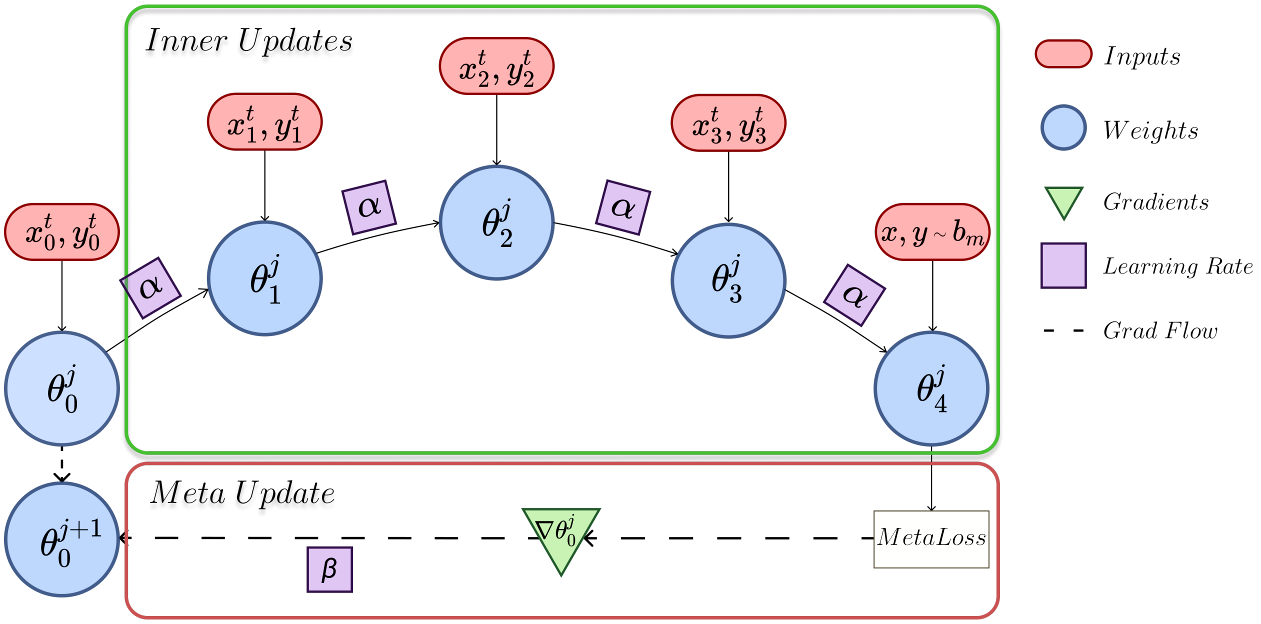


Fig. 3.1. The proposed base **C-MAML** algorithm: For every batch of data from the streaming task, the initial weights θ_0^j undergo a series of k *fast updates* to obtain θ_k^j, which is evaluated against a meta-loss to backpropagate gradients with respect to the weights θ_0^j to get θ_0^{j+1}. Here, $k = 4$. The blue boxes indicate *fast weights* while the green box indicates the gradient for the *slow update*. Purple boxes indicate scalar learning-rates.

Algorithm 1 C-MAML : C-MAML

```
Input: Network weights $\theta_0^0$, inner objective $\ell$, meta objective $L$, Inner learning rate $\alpha$, Outer learning rate $\beta$
$j \leftarrow 0$
$R \leftarrow \{\}$                                          ▷ Initialise replay-buffer
for $t := 1$ to $T$ do
    $(X^t, Y^t) \sim D_t$
    for $ep := 1$ to $num_{epochs}$ do
        for batch $b$ in $(X^t, Y^t)$ do
            $k \leftarrow sizeof(b)$
            $b_m \leftarrow Sample(R) \cup b$                ▷ batch of samples from $\tau_{1:t}$ for meta-loss
            for $k' = 0$ to $k - 1$ do
                Push $b[k']$ to R with some probability based on reservoir sampling
                $\theta_{k'+1}^j \leftarrow \theta_{k'}^j - \alpha \cdot \nabla_{\theta_{k'}^j} \ell_t(\theta_{k'}^j, b[k'])$   ▷ inner-update on each incoming sample
            end for
            $\theta_0^{j+1} \leftarrow \theta_0^j - \beta \cdot \nabla_{\theta_0^j} L_t(\theta_k^j, b_m)$        ▷ outer-update by differentiating meta-loss
            $j \leftarrow j + 1$
        end for
    end for
end for
```

3.1.1. Equivalence of Objectives

When we optimise the OML objective through the k-step MAML update, as proposed in C-MAML in Eq. 3.1.1:

$$\min_{\theta_0^j} \mathbb{E}_{\tau_{1:t}} \left[L_t \left(U_k(\theta_0^j) \right) \right] \tag{3.1.4}$$

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 continual learning :

$$\min_{\theta_0^j} \sum_{i=1}^{t} \left(\ell_i(\theta_0^j) - \alpha \frac{\partial \ell_i \left(\theta_0^j\right)}{\partial \theta_0^j} \cdot \frac{\partial \ell_t \left(\theta_0^j\right)}{\partial \theta_0^j} \right) \tag{3.1.5}$$

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 [**33**] showed that the k-step MAML gradient for the weights θ_0^j at time j, denoted as $g_{\text{MAML}}(\theta_0^j)$ is of the form:

$$\frac{\partial L_{meta}(\theta_k^j)}{\partial \theta_0^j} = \bar{g}_k - \alpha \sum_{k'=0}^{k-1} \bar{g}_{k'} \bar{H}_k - \alpha \sum_{k'=0}^{k-1} \bar{H}_{k'} \bar{g}_k + O\left(\alpha^2\right) \quad (\alpha \text{ is the } \textit{inner-loop} \text{ learning rate})$$

$$= \bar{g}_1 - \alpha \bar{H}_1 \bar{g}_0 - \alpha \bar{H}_0 \bar{g}_1 + O\left(\alpha^2\right) \quad (\text{using } k = 1)$$

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

$$= \frac{\partial L_{meta}\left(\theta_0^j\right)}{\partial \theta_0^j} - \frac{\partial}{\partial \theta_0^j} (\bar{g}_0 \cdot \bar{g}_1)$$

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

$$= \frac{\partial \left(\sum_{i=1}^{t} \ell_i(\theta_0^j) - \alpha \frac{\partial L_{meta}(\theta_0^j)}{\partial \theta_0^j} \frac{\partial \ell_t(\theta_0^j)}{\partial \theta_0^j} \right)}{\partial \theta_0^j}$$

$$= \frac{\partial \left(\sum_{i=1}^{t} \ell_i(\theta_0^j) - \alpha \frac{\partial \sum_{i=1}^{t} \ell_i(\theta_0^j)}{\partial \theta_0^j} \frac{\partial \ell_t(\theta_0^j)}{\partial \theta_0^j} \right)}{\partial \theta_0^j} \quad (\text{expanding } L_{meta})$$

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

which is the same as the gradient of Eq. 3.1.5.

where:

$\bar{g}_k = \frac{\partial L_{meta}\left(\theta_0^j\right)}{\partial \theta_0^j}$ (gradient of the *meta-loss* evaluated at the initialisation)

$\bar{g}_{k'} = \frac{\partial}{\partial \theta_0^j} L_{inner}(\theta_0^j)$ (for $k' < k$) (gradient of the *inner-loss* on data seen at step k', evaluated at the initialisation)

$\theta_{k'+1}^j = \theta_{k'}^j - \alpha g_{k'}$ (sequence of parameter vectors obtained in the inner-loop)

$\bar{H}_k = L''_{meta}\left(\theta_0^j\right)$ (Hessian of the *meta-loss* evaluated at the initialisation)

$\bar{H}_{k'} = L''_{inner}\left(\theta_0^j\right)$ (for $k' < k$) (Hessian of the *inner-loss* on data seen at step k', evaluated at the initialisation)

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

$$L_{inner} = \ell_t$$

Bias in the objective: We can see in Eq. 3.1.5 that the gradient alignment term introduces some bias, which means that the parameters don't exactly converge to the minimiser of the losses on all tasks. This is acceptable in the regime of continual learning since we don't aim to reach the minimiser of some stationary distribution anyway (as also mentioned in Section 3.2.2). 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 and time, ensuring low-interference between tasks will pay off much 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.

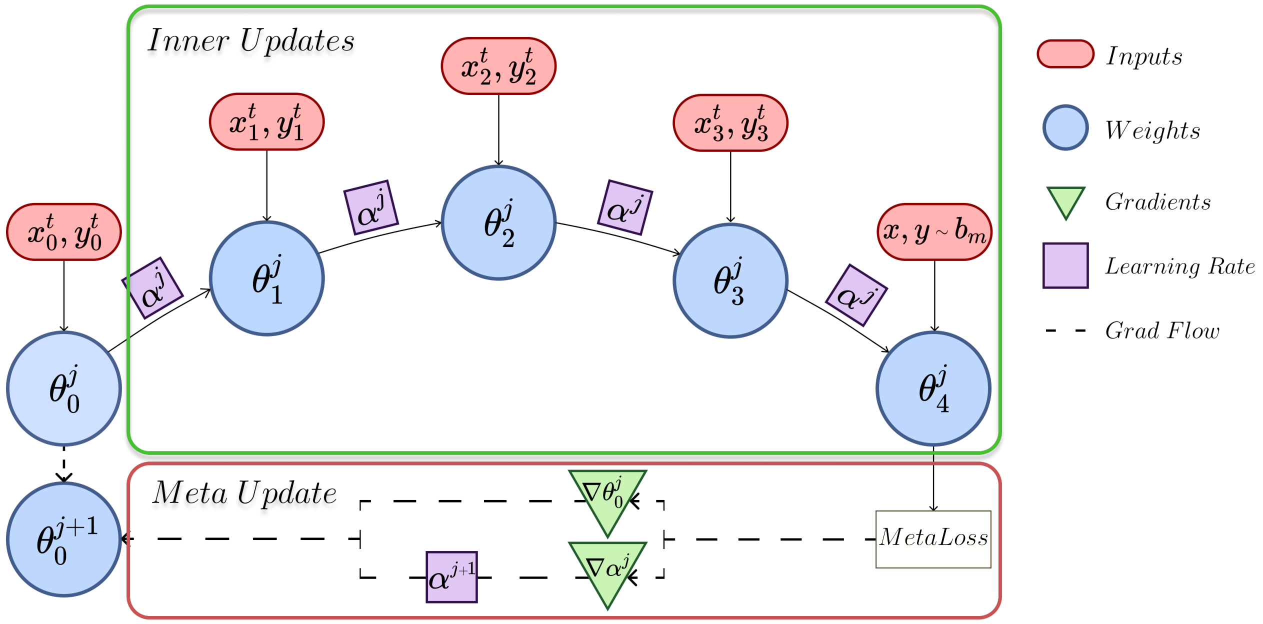


Fig. 3.2. The proposed **La-MAML** algorithm: For every batch of data, the initial weights undergo a series of k *fast updates* to obtain θ_k^j, which is evaluated against a meta-loss to backpropagate gradients with respect to the weights θ_0^j and learning rates α^j to get θ_0^{j+1} and α^{j+1}. Here, $k = 4$. First α^j is updated to α^{j+1} which is then used to update θ_0^j to θ_0^{j+1} The blue boxes indicate *fast weights* while the green boxes indicate gradients for the *slow update*. Purple boxes indicate the vector of learnable learning-rates. Learning rates and weights are updated in an asynchronous manner.

3.2. 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 to be used in the *inner updates*, as depicted in Figure 3.2. This is motivated by our observation that the expression for the gradient of Eq. 3.1.1 with respect to the inner loop's learning rates directly reflects the alignment between the old and new tasks. The augmented learning objective and its gradient with respect to the learning rate vector α^j at time-step j during training, denoted as $g_{MAML}(\alpha^j)$ are then given as:

$$\min_{\theta_0^j, \alpha^j} \sum_{\mathcal{S}_k^j \sim D_t} \left[L_t \left(U_k \left(\alpha^j, \theta_0^j, \mathcal{S}_k^j \right) \right) \right] \tag{3.2.1}$$

$$g_{MAML}(\alpha^j) = \frac{\partial}{\partial \alpha^j} L_t \left(\theta_k^j \right) = \frac{\partial}{\partial \theta_k^j} L_t \left(\theta_k^j \right) \cdot \left(- \sum_{k'=0}^{k-1} \frac{\partial}{\partial \theta_{k'}^j} \ell_t \left(\theta_{k'}^j \right) \right) \tag{3.2.2}$$

We will present the derivation of $g_{MAML}(\alpha^j)$ in the next section, and have simply stated the expression here for a first-order approximation of $g_{MAML}(\alpha^j)$ [**11**]. Here, the first term 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 learning rates 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) learning rate gradients would pull up (down) the learning rate magnitude. We depict this visually in Figure 3.3.

We propose updating the network weights and learning rates *asynchronously* in the meta-update. Let α^{j+1} be the updated learning rate vector obtained by taking an SGD step with the learning rate gradient from Eq. 3.2.1 at time j. We then update the weights as:

$$\theta_0^{j+1} \leftarrow \theta_0^j - max(0, \alpha^{j+1}) \cdot \nabla_{\theta_0^j} L_t(\theta_k^j) \tag{3.2.3}$$

where k is the number of steps taken in the inner-loop. The learning rates 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. Concurrently, the weights for which the gradients align are updated with a higher learning rate. 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 2 illustrates this procedure.

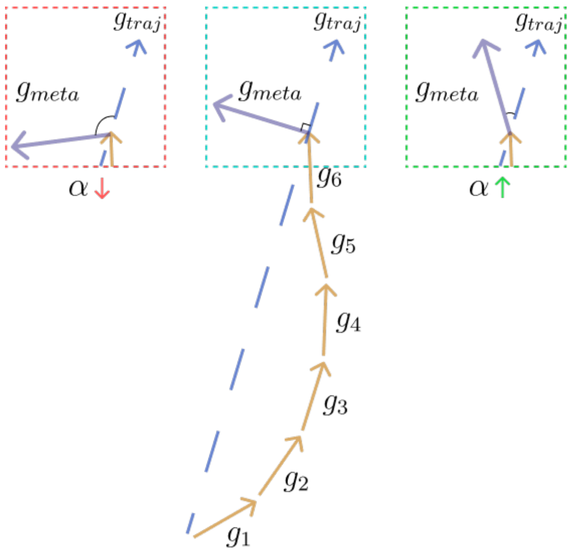


Fig. 3.3. Different scenarios for the alignment of g_{traj} (blue dashed line) and g_{meta}, going from interference (left) to alignment (right). Yellow arrows denote the *inner updates*. The learning rates in α increase (decrease) when gradients align (interfere).

Algorithm 2 La-MAML : Look-ahead MAML

```
Input: Network weights $\theta_0^0$, vector of learning rates $\alpha^0$ (initialised to some positive value $\alpha_0$), inner objective $\ell$, meta objective $L$, scalar learning rate for $\alpha$ : $\eta$
$j \leftarrow 0$
$R \leftarrow \{\}$                                                    ▷ Initialise replay-buffer
for $t := 1$ to $T$ do
    $(X^t,Y^t) \sim D_t$
    for $ep := 1$ to $num_{epochs}$ do
        for batch $b$ in $(X^t,Y^t)$ do
            $k \leftarrow sizeof(b)$
            $b_m \leftarrow Sample(R) \cup b$                ▷ batch of samples from $\tau_{1:t}$ for meta-loss
            for $n = 0$ to $k-1$ do
                Push $b[k']$ to R with some probability based on reservoir sampling
                $\theta^j_{k'+1} \leftarrow \theta^j_{k'} - \alpha^j \cdot \nabla_{\theta^j_{k'}} \ell_t(\theta^j_{k'}, b[k'])$
            end for
            $\alpha^{j+1} \leftarrow \alpha^j - \eta \nabla_{\alpha^j} L_t(\theta^j_k, b_m)$
                              ▷ Update learning rates $\alpha^j$ by differentiating the meta-loss
            $\theta_0^{j+1} \leftarrow \theta_0^j - max(0, \alpha^{j+1}) \cdot \nabla_{\theta_0^j} L_t(\theta^j_k, b_m)$
                    ▷ async outer-update with clipped $\alpha^{j+1}$, by differentiating the meta-loss
            $j \leftarrow j+1$
        end for
    end for
end for
```

Our meta-learning based algorithm incorporates concepts from both prior-based and replay-based approaches. The learning rates 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 learning rates 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 [**20**]. The learnable learning rates 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 [**48**] since the learning rates are evolved in a manner similar to *look-ahead search*. Look-ahead search adjusts step-sizes of parameter updates, based on the *fitness* of parameters that would be obtained after making the proposed parameter updates. Our learning rate update is also analogous to the heuristic uncertainty-based learning rate update schemes of UCB [**10**] or Bayesian Gradient Descent (BGD) [**49**], which we compare to in Section 4.2.3.

In the next section, we provide the derivations of the MAML gradient of the model weights and learning rates, which we have talked about in this section.

3.2.1. Hypergradient Derivation for La-MAML

We derive the gradient of the weights θ_0^j and learning rates 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*:

$$\begin{aligned}
g_{\text{MAML}}(\alpha^j) &= \frac{\partial}{\partial \alpha^j} L_t\left(\theta_k^j\right) = \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \frac{\partial}{\partial \alpha^j}\left(\theta_k^j\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \frac{\partial}{\partial \alpha^j}\left(U\left(\theta_{k-1}^j\right)\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \frac{\partial}{\partial \alpha^j}\left(\theta_{k-1}^j - \alpha^j \frac{\partial \ell_t(\theta_{k-1}^j)}{\partial \theta_{k-1}^j}\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(\frac{\partial}{\partial \alpha^j}\theta_{k-1}^j - \frac{\partial}{\partial \alpha^j}\left(\alpha^j \frac{\partial \ell_t(\theta_{k-1}^j)}{\partial \theta_{k-1}^j}\right)\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(\frac{\partial}{\partial \alpha^j}\theta_{k-1}^j - \frac{\partial \ell_t(\theta_{k-1}^j)}{\partial \theta_{k-1}^j}\right)
\end{aligned}$$

(Taking $\dfrac{\partial \ell_t\left(\theta_{k-1}^j\right)}{\partial \theta_{k-1}^j}$ as a constant w.r.t α^j to get the first-order MAML approximation)

$$\begin{aligned}
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(\frac{\partial}{\partial \alpha^j} U\left(\theta_{k-2}^j\right) - \left(\frac{\partial \ell_t(\theta_{k-1}^j)}{\partial \theta_{k-1}^j}\right)\right) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(\frac{\partial}{\partial \alpha^j}\theta_0^j - \sum_{n=0}^{k-1} \frac{\partial \ell_t(\theta_n^j)}{\partial \theta_n^j}\right) \qquad (a) \\
&= \frac{\partial}{\partial \theta_k^j} L_t\left(\theta_k^j\right) \cdot \left(- \sum_{n=0}^{k-1} \frac{\partial \ell_t(\theta_n^j)}{\partial \theta_n^j}\right) \qquad (b)
\end{aligned}$$

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.

For completeness, we also similarly derive the MAML gradient for the weights θ_0^j, denoted as $g_{\text{MAML}}(\theta_0^j)$ as:

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

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

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

3.2.2. Connection to Work Outside Continual Learning

We outline here the connections of our approach to prior work not pertaining to the field of 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 continual learning 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* [**6, 42**] 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 [**6, 42**] the correlation between two successive stochastic gradients on the same data distribution is used to converge faster. *Hypergradient Descent* [**6**] 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 learning rates, to obtain these learning rate *hypergradients* automatically.

Learning learning rates in meta-learning: Meta-SGD [**27**] 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 [**45**] 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 continual learning domain.

Chapter 4

Evaluation

In this Chapter, we will first describe our experiments and baselines in Section 4.1, and present quantitative results in Section 4.2. Additionally, we will present a qualitative analysis of our algorithm in Section 4.3. This includes an investigation of the robustness and scalability of our algorithm in subsections 4.3.2 and 4.3.3, along with a study of the variation in gradient alignment across tasks over time in subsection 4.3.1. We will also provide a discussion about the merits and demerits of our method against prior work in Section 4.3.4. Finally, we will provide further details about hyper-parameter optimisation and other experimental details towards the end of the chapter.

4.1. Experimental Setup

We evaluate La-MAML in task incremental settings, where the model learns a set of sequentially streaming classification tasks. The loss used is the cross-entropy loss over the *logits* of the classes within a task.

Benchmarks: Experiments are performed on the MNIST, CIFAR and TinyImagenet datasets [**1, 24, 21**]. The large-scale visual classification experiments (on CIFAR and Tiny-Imagenet) are *task-aware* and we use a *multi-headed* network, i.e, the network parameters are shared across tasks except for the last classification layer which is separate for each task. The experiments on MNIST are *task-agnostic* and use a *single-headed* network, since their classification space remains the same (classes $0 - 9$), but the transformation to the images changes with each task.

Metrics: Similar to [**38**], 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.**

4.1.1. Baselines

On the MNIST benchmarks, we follow the experimental setup of MER [**38**], which evaluates the model in the *Single-Pass* setting. We compare our algorithm against the baselines used in MER, which are as follows:

- Online: A simple baseline for the *Single-Pass* setup, where a single network is trained one example at a time with SGD.
- EWC [**20**]: Elastic Weight Consolidation is a *prior-based* method which constraints the model's weights based on how important they are for the previous tasks to avoid catastrophic forgetting. The importance is measured using a *Fisher Information Matrix* for each task.
- GEM [**28**]: Gradient Episodic Memory uses constrained optimisation to solve a quadratic program on the gradients of new and replay samples, trying to make sure that these gradients do not *interfere* with and alter the previous tasks' gradients.
- MER [**38**]: 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 *Single-Pass* setups of all benchmarks.

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

- IID: The model trains on the data from all tasks in an independent and identically distributed manner, thus bypassing the issue of catastrophic forgetting completely. Therefore, IID acts as an upper bound for the RA achievable with the network chosen for these experiments.
- 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.
- iCARL [**37**]: iCarl is a baseline from the family of *distillation-based* class-incremental learners, which learns to classify images in the metric space. We modify it to work in the task-incremental scenario. It prevents catastrophic forgetting by using a small memory of exemplar samples per class, and store copies of the network to perform distillation from these old network weights.
- A-GEM [**8**]: 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 (modification of C-MAML for the purpose of comparison of learning rate-modulation): Bayesian Gradient Descent (BGD) [**49**] proposes training a bayesian

neural network for continual learning where the learning rates for the parameters (the means) are derived from their variances, and thus are modulated during training. We construct this baseline by equipping C-MAML with the above-mentioned bayesian training mechanism, where each parameter in θ_0^j is now sampled from a gaussian distribution with a certain mean and variance. The inner-loop stays the same as C-MAML (with a constant scalar learning rate α), but the magnitude of the meta-update to the parameters in θ_0^j 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.

Table 4. II. 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

4.2. Quantitative Results

In this section we will first present results on the toy MNIST benchmarks for continual learning in subsection 4.2.1, and on larger scale visual classification benchmarks in subsection 4.2.2. We will also investigate and ablate the performance our proposed learning rate-modulation scheme in subsection 4.2.3.

4.2.1. Continual learning benchmarks

First, we carry out experiments on the toy continual learning benchmarks proposed in prior work on continual learning. **MNIST Rotations**, introduced in [**28**], 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

Table 4. I. 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-La-MAML	74.07 ± 0.58	-6.66 ± 0.44	70.54 ± 1.54	-14.02 ± 2.14	44.48 ± 0.76	-24.18 ± 0.65
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**

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 [**38**], 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 [**33**], 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* [**20**], *GEM* [**28**] and *MER* [**38**], detailed in Section 4.1.1. In Table 4. I, we see that La-MAML achieves comparable or better performance than the baselines on all benchmarks. Table 4. II 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.

4.2.2. 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 [**28**]. 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 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 Section 4.4.

Table 4. IV reports the results of these experiments. We consistently observe superior performance of La-MAML as compared to other continual learning baselines on both datasets across setups. While the iCarl baseline attains lower BTI in some setups, it achieves that at the cost of lower accuracy during learning. Among the high-performing approaches, La-MAML has the lowest BTI. Recent work [**9, 38**] 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 learning rate 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 robust and better-performing under both the *Single* and *Multiple-Pass* setups of continual learning which come with different kinds of challenges. Many continual learning

methods [**10, 43**] are suitable for only one of the two setups. As shown in Figure 4.1, our model evolves to become resistant to *forgetting* as training progresses. This means that beyond a point, it can keep taking gradient updates on a small window of incoming samples without needing to do *meta-updates*.

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

To capture the gains from learning the learning rates, we compare La-MAML with our base algorithm, **C-MAML**. We ablate our choice of updating learning rates asynchronously by constructing a version of C-MAML where per-parameter learnable learning rates are used in the inner updates while the meta-update still uses a constant scalar learning rate

Table 4. III. Results on the *Multiple-Pass* and *Single-Pass* setups with CIFAR-100 Dataset. Experiments are run with 3 seeds. * indicates result omitted due to high instability. MER results for Multiple-Pass omitted due to very high training times.

Method	CIFAR-100			
	Multiple		Single	
	RA	BTI	RA	BTI
IID	85.60 ± 0.40	-	-	
ER	59.70 ± 0.75	-16.50 ± 1.05	47.88 ± 0.73	-12.46 ± 0.83
iCarl	60.47 ± 1.09	-15.10 ± 1.04	53.55 ± 1.69	**-8.03 ± 1.16**
GEM	62.80 ± 0.55	-17.00 ± 0.26	48.27 ± 1.10	-13.7 ± 0.70
AGEM	58.37 ± 0.13	-17.03 ± 0.72	46.93 ± 0.31	-13.4 ± 1.44
MER	-	-	51.38 ± 1.05	-12.83 ± 1.44
Meta-BGD	65.09 ± 0.77	-14.83 ± 0.40	57.44 ± 0.95	-10.6 ± 0.45
C-MAML	65.44 ± 0.99	-13.96 ± 0.86	55.57 ± 0.94	-9.49 ± 0.45
La-ER	67.17 ± 1.14	-12.63 ± 0.60	56.12 ± 0.61	-7.63 ± 0.90
Sync-La-MAML	67.06 ± 0.62	-13.66 ± 0.50	58.99 ± 1.40	-8.76 ± 0.95
La-MAML	**70.08 ± 0.66**	**-9.36 ± 0.47**	**61.18 ± 1.44**	-9.00 ± 0.2

Table 4. IV. Results on the *Single-Pass* and *Multiple-Pass* setups with TinyImagenet-200 Dataset. Experiments are run with 3 seeds. * indicates result omitted due to high instability. MER results for Multiple-Pass omitted due to very high training times.

Method	TinyImagenet			
	Multiple		Single	
	RA	BTI	RA	BTI
IID	77.1 ± 1.06	-	-	-
ER	48.23 ± 1.51	-19.86 ± 0.70	39.38 ± 0.38	-14.33 ± 0.89
iCarl	54.77 ± 0.32	**-3.93 ± 0.55**	45.79 ± 1.49	**-2.73 ± 0.45**
GEM	50.57 ± 0.61	-20.50 ± 0.10	40.56 ± 0.79	-13.53 ± 0.65
AGEM	46.38 ± 1.34	-19.96 ± 0.61	38.96 ± 0.47	-13.66 ± 1.73
MER	-	-	44.87 ± 1.43	-12.53 ± 0.58
Meta-BGD	*	*	50.64 ± 1.98	-6.60 ± 1.73
C-MAML	61.93 ± 1.55	-11.53 ± 1.11	48.77 ± 1.26	-7.6 ± 0.52
La-ER	54.76 ± 1.94	-15.43 ± 1.36	44.75 ± 1.96	-10.93 ± 1.32
Sync-La-MAML	65.40 ± 1.40	-11.93 ± 0.55	**52.84 ± 2.55**	-7.3± 1.93
La-MAML	**66.99 ± 1.65**	**-9.13 ± 0.90**	52.59 ± 1.35	**-3.7 ± 1.22**

Table 4. V. Gradient Alignment on CIFAR-100 and IMAGENET dataset (values lie in [-1,1], higher is better)

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

during training. We refer to it as **Sync-La-MAML** since it has synchronously updated learning rates that don't modulate the meta-update. We also construct an ablation referred to as *La-ER*, where the parameter updates are carried out as in ER but the learning rates are modulated using the La-MAML objective's first-order version. This tells us what the gains of learning rate modulation are over ER, when there is no meta-learning to encourage gradient alignment of the model parameters. While only minor gains are seen on the MNIST benchmarks from asynchronous learning rate 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-La-MAML*. This shows that optimising the learning rates aids learning and our *asynchronous* update helps in knowledge consolidation by enforcing conservative updates to mitigate interference.

To test our learning rate modulation against an alternative *bayesian* modulation scheme proposed in BGD [**49**], we define a baseline called Meta-BGD where per-parameter variances are modulated instead of learning rates. This is described in further detail in Section 4.1.1. 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 Section 4.3.2, as well as a discussion of the setups adopted in prior work, in Section 4.3.4.

We also compare the gradient alignment of our three variants along with ER in Table 4. V by calculating the cosine similarity between the gradients of the replay samples and newly arriving data samples. As previously stated, the aim of many continual learning 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.

4.3. Qualitative Analysis

In this section, we focus on a qualitative analysis of our algorithm and its ablations to investigate its robustness and scalability. First, we show how the inter-task gradient alignment varies over time for C-MAML and La-MAML, to provide empirical evidence that the time-aware objective for continual learning that we optimise is sufficient to incentivise

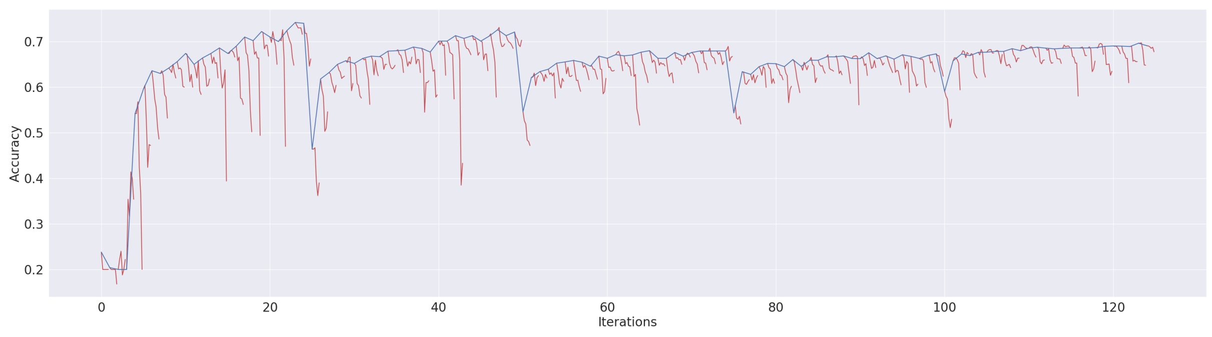


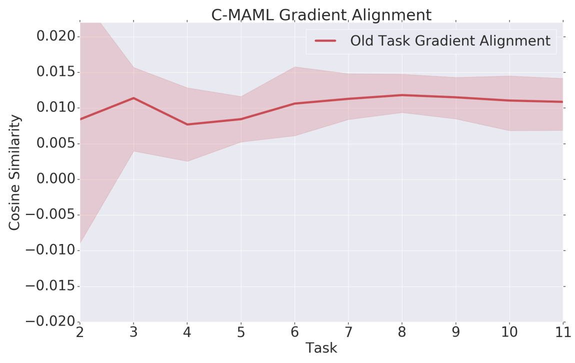
Fig. 4.1. Retained Accuracy (RA) for La-MAML plotted every 25 meta-updates up to Task 5 on CIFAR-100. RA at iteration j (x-axis) denotes accuracy on all tasks seen uptil then. Red denotes the RA computed during the *inner updates* (at θ_k^j). Blue denotes RA computed at θ_0^{j+1} right after a *meta-update*. We see that in the beginning, inner updates lead to catastrophic forgetting (CF) since the weights are not suitable for continual learning 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.

alignment across all tasks. We noted previously that this is partially responsible for our sample-efficiency. Second, we show how the performance of La-MAML, C-MAML and Sync-La-MAML varies across variations in hyper-parameters like inner and outer learning-rates. Lastly, we show a timing comparison of our algorithm v/s MER, our main meta-learning baseline.

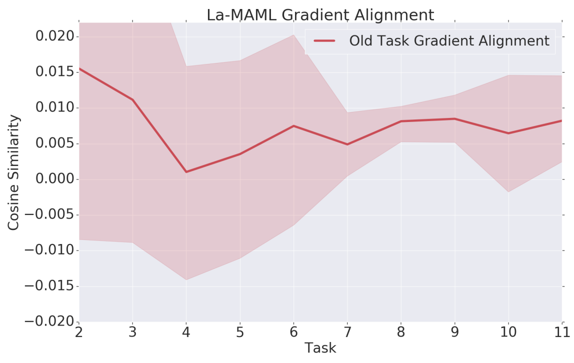
4.3.1. Inter-Task Alignment

We now provide empirical evidence that our more sample-efficient objective (compared to MER) does not come at the cost of any significant gradient interference between *old* tasks, when we try to align the gradients between *old* and *new* tasks. We assume that at time j during training, we are seeing samples from the streaming task τ_t. It is intuitive that incentivising the alignment of all $\tau_{1:t}$ with the common τ_t indirectly also incentivises the alignment amongst $\tau_{1:t-1}$ as well. To demonstrate that this happens, we compute the mean dot product of the gradients amongst the old tasks $\tau_{1:t-1}$ as the new task τ_t is added, for t varying from 2 to 11. We do this for La-MAML and La-MAML on CIFAR-100.

As can be seen in Figures 4.2a and 4.2b, the alignment stays positive and roughly constant even as more tasks are added.



(a) C-MAML



(b) La-MAML

Fig. 4.2. Average dot product amongst gradients as tasks are added, for the C-MAML and La-MAML algorithms. *x-axis* shows the streaming task ID t.

4.3.2. 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 learning rate-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 learning rate (scalar) for the weights (α and β)
- **Sync-La-MAML**: Common initialization value for the vector of inner-loop learning rates (α_0), scalar learning rate for the vector of learning rates (η) and scalar learning rate for the weights in the outer update (β)
- **La-MAML**: Scalar initialization value for the vector learning rates (α_0) and scalar learning rate of learning rates (η)

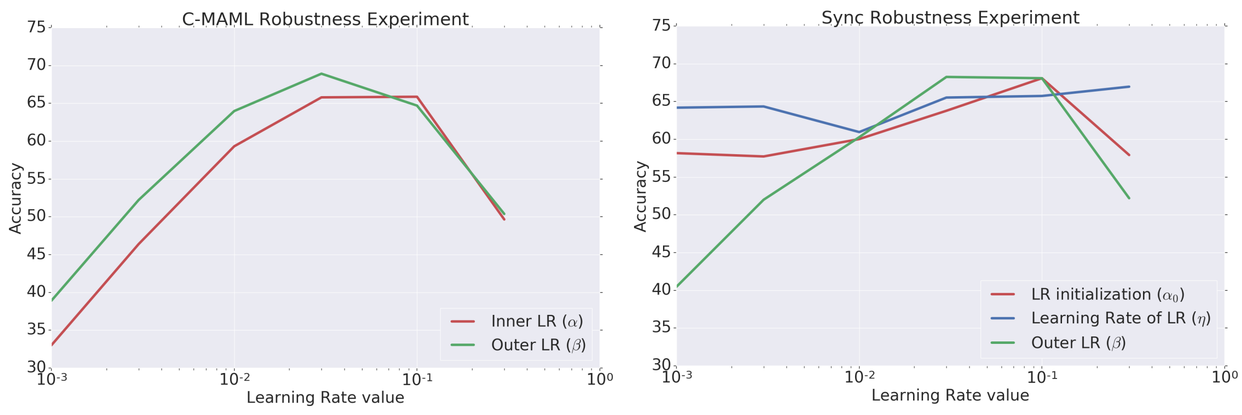
La-MAML is considerably more robust to tuning compared to its variants, as can be seen in Figure 4.3c. We empirically observe that it only requires tuning of the initial value of the learning rate, while being relatively insensitive to the learning rate of the learning rate (η). We see a consistent trend 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 learning rates 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.

As seen in Figure 4.3a, C-MAML sees considerable performance variation with the tweaking of both the inner and outer learning rate. We also see that the effects of the variations of the inner and outer learning rate follow very similar trends and their optimal values 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. Sync-La-MAML (Figure 4.3b), 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.

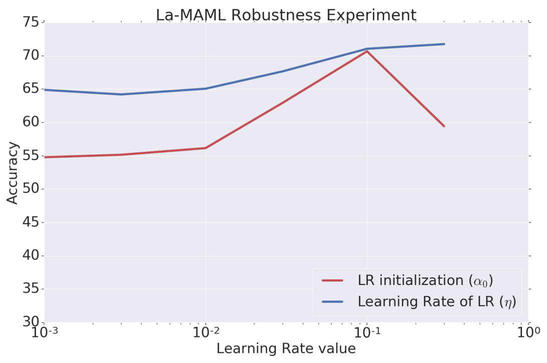
Fig. 4.4 shows the result of 2D grid-searches over sets of the above-mentioned hyper-parameters for C-MAML and La-MAML for a better overview.

4.3.3. Timing Comparisons

In this figure we compare the wall-clock running times (Retained Accuracy (RA) vs Time) of MER and La-MAML in the *Single-Pass* setup, on the same machine. Both methods have a hyper-parameter *glances* which indicates the number of meta-updates made on each data-point, with the performance of the algorithms increasing with the increase in *glances* up to a certain point. Note that in the *Single-Pass* setup, we can only see the data in a single pass, and therefore seeing each point for a number of *glances* is essential, since once we move on to

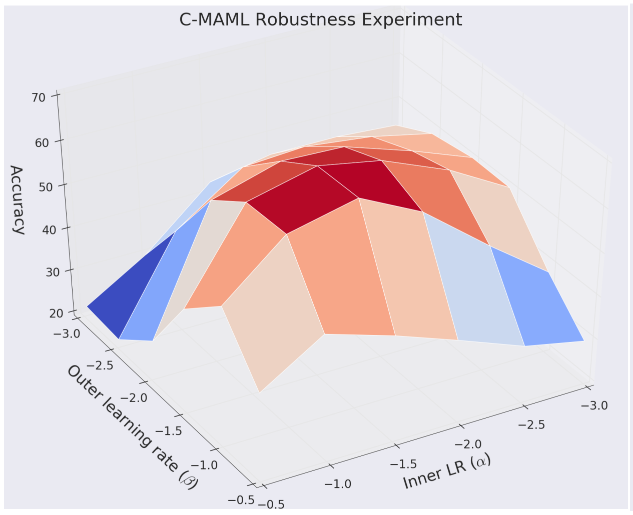


(a) C-MAML: Modulation of α and β **(b)** Sync-La-MAML: Modulation of α_0, η and β

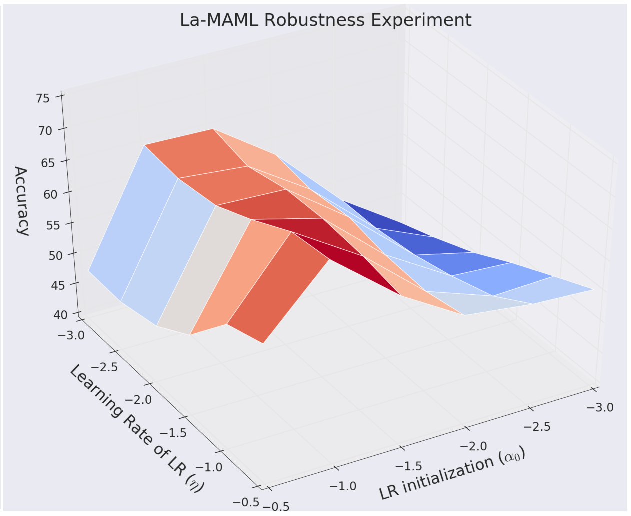


(c) La-MAML: Modulation of α_0 and η

Fig. 4.3. 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].



(a) C-MAML: Modulation of α and β



(b) La-MAML: Modulation of α_0 and η

Fig. 4.4. 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.

the next data point, we can't revisit an old data point again. Experiment was conducted by running our method for one to five *glances*, and MER for [2,4,6,8,10] *glances* while recording the RA and running times of both the methods. The *glances* are chosen according to how many are needed to achieve performance comparable to the other method. In the plots we see that 5 *glances* of La-MAML take the same time in which MER finishes a single *glance* run, making our method suitable for the scale of real world experiments.

4.3.4. Qualitative Comparison with Prior Work

In Table 4. VI, we provide an overview of the limitations and benefits coming from various continual learning methods to situate our work better in the context of prior work.

Prior-focused methods face model capacity saturation as the number of tasks increase. These methods freeze weights to defy forgetting, and so penalise changes to the parameters of the model, even if those changes could potentially improve the model's performance on old tasks. They are also not suitable for the *Single-Pass* setup, since it requires many passes through the data for every task to learn weights that are optimal enough to be frozen. Additionally, the success of weight freezing schemes can be attributed to over-parameterisation in neural networks, leading to sub-networks with sufficient capacity to learn separate tasks. However continual-learning setups are often motivated in resource-constrained settings requiring efficiency and scalability. Therefore solutions that allow light-weight continual learners are desirable. Meta-learning algorithms are able to exploit even small models to learn a good initialization where gradients are aligned across tasks, enabling shared progress on

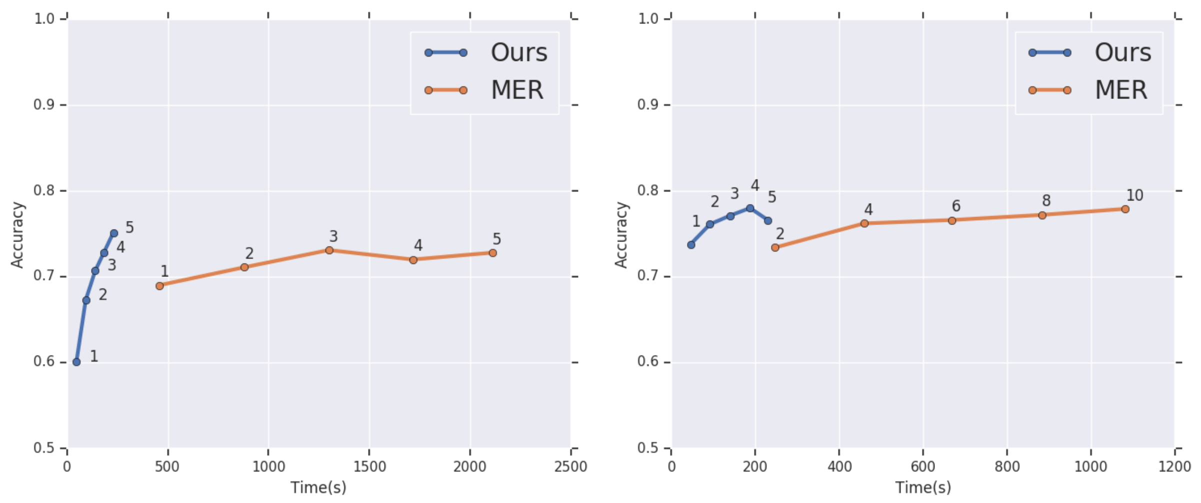


Fig. 4.5. Retained Accuracy vs Running time (seconds) for La-MAML and MER while varying the number of *glances* through the data on MNIST Permutations (left) and Rotations (right). Passes are denoted by the label marked in the plot corresponding to datapoints.

optimisation of task-wise objectives. Our method additionally allows meta-learning to also achieve a prior-focusing affect through the asynchronous meta-update, without necessarily needing over-parameterised models. It allows change, as long as those changes lead to *descending the gradient* on old tasks.

In terms of resources, *meta-learning based methods* require smaller replay memories than traditional methods because they learn to generalise better across and within tasks, thus being sample-efficient. It should be noted that our learning rate modulation involves clipping updates for parameters with non-aligning gradients. In this sense, it is related to methods like GEM and AGEM mentioned before. Where the distinction lies, is that our method takes some of the burden off of the clipping, by ensuring that gradients are more aligned in the first place. This means that there should be less interference and therefore less clipping of updates deemed essential for learning new tasks, on the whole.

Our learnable learning rates incur a memory overhead equal to the parameters of the network. This is comparable to or less than many prior-based methods that store between 1 to T scalars per parameter depending on the approach (T is the number of tasks). Additionally, since we use the scalars as per-parameter learning rates, they can be regarded as non-additional memory since sophisticated optimizers like Adam [**18**] store per-parameter statistics anyway. However, we do need to store the computational graph of the derivatives of the learning rates, which is an added overhead during training.

4.4. Experimental Details

We will now describe the remaining details of the experimental setup we adopted. *Hyper-parameters and Optimisation*: We carry out hyper-parameter tuning for all the approaches by performing a grid-search over the range [0.0001 - 0.3] for hyper-parameters related to the learning-rate. Further experiments are carried out to find the optimal number of *glances* over each sample in the *Single-Pass* setup. Tables 4. VII and 4. VIII lists the optimal hyper-parameters used for all the compared approaches. All setups used the SGD optimiser since it was found to preform better than Adam [**18**] (possibly due to reasons stated in Section 3.2.2 regarding the continual learning 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. IV

Batch Sizes: 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

Table 4. VI. Setups in prior work: We describe the setups and assumptions adopted by prior work, focusing on approaches relevant to our method. FWT and BWT refer to forward and backward transfer as defined in [**28**]. '-' refers to no inductive bias for or against the specific property. Saturation of capacity refers to reduced network plasticity due to weight change penalties gradually making further learning impossible. The *Single-Pass* setup is defined in Section 2.2 and refers to whether a method can work in the setup where training data is only seen in a single stream, without repeatedly shuffling data during training per task. < and > under replay indicate that a method's replay requirements are lesser or more compared to other methods in the table. *Fishers* refers to the Fisher Information Matrix (FIM) computed per task. Each FIM has storage equal to that of the model parameters. Approaches using Bayesian Neural Networks require twice as many parameters (as does La-MAML) to store the mean and variance estimates per parameter.

Approach	Transfer		Capacity	Resources		Algorithm
	FWT	BWT	Saturates	Single-Pass	Storage	
Prior-Focused	-	$\times$	$\surd$	$\times$	T Fishers	EWC [**20**]
Prior Focused	-	$\times$	$\surd$	$\times$	T masks	HAT [**43**]
Prior Focused	-	$\times$	$\surd$	$\surd$	2x params	BGD/UCB [**49**] [**10**]
Replay	-	-	-	$\surd$	> replay	iCarl [**37**]
Replay	-	-	-	$\surd$	> replay	GEM [**28**]
Meta + Replay	$\surd$	$\surd$	-	$\surd$	replay	MER [**38**]
Meta + Replay	$\surd$	$\surd$	-	$\surd$	< replay	Ours
Expansion	$\times$	$\times$	$\times$	$\surd$	N Models	CURL
Expansion	$\times$	$\times$	$\times$	$\surd$	N Models	MOLE
Expansion	$\times$	$\times$	$\times$	$\surd$	N Models	CN-DPM

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.

MAML implementation: We found only very small performance gaps between the first and second-order MAML versions of our proposed algorithms with performance differences being in the range of 1-2% for RA. This means that we could equivalently use the first-order versions for continual learning, when the compute budget is lower. This is in line with the observation cited before, that deep neural networks have near-zero hessians since the ReLU non-linearity is linear almost everywhere [**40**].

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 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 evenly split the validation set into *val* and *test* splits, since the labels in the actual test set are not released. We similarly evenly split the CIFAR-100 test data into *test* and *val* splits since it doesn't contain a validation split in the release. For the training dataset, we randomly subsample and use 2500 samples from each task for training.

Table 4. VII. Final hyperparameters for all compared approaches on the CIFAR and Imagenet benchmarks. LR refers to the learning rate hyper-parameter

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

Table 4. VIII. Final hyperparameters used for our variants on the MNIST benchmarks

Method	Parameter	Permutations	Rotations	Many
C-MAML	α	0.03	0.1	0.03
	β	0.1	0.1	0.15
	Glances	5	5	5
Sync-La-MAML	α_0	0.15	0.15	0.03
	β	0.1	0.3	0.03
	η	0.1	0.1	0.1
	Glances	5	5	10
La-MAML	α_0	0.3	0.3	0.1
	η	0.15	0.15	0.1
	Glances	5	5	10

Chapter 5

Conclusion

We introduced La-MAML, an efficient meta-learning algorithm that leverages replay to avoid *forgetting* while learning new tasks. It favors positive backward transfer by learning the weights and LRs in an asynchronous manner, guided by the interplay across old and new tasks. It is capable of learning online on a non-stationary stream of data and scales to larger-scale vision problems. We conducted experiments on benchmarks of varying complexity and differently constrained setups. The presented results demonstrate superior performance of our algorithm against the state-of-the-art across both the *Single* and *Multiple-Pass* setups, which present a unique set of challenges to a continual learning system. We also showed that our method does not require extensive tuning unlike many meta-learning algorithms and is relatively robust since it is fairly insensitive of one of the two hyper-parameters in the algorithm. As mentioned before, this is desirable in continual or non-stationary learning, since we can't tune learning rate schedules beforehand.

In the future, more work on analysing and producing good optimizers for continual learning is needed, since many of our standard go-to optimizers like Adam [**18**] are primarily aimed at ensuring faster convergence in *stationary* supervised learning setups. An interesting direction is to explore how the connections to *meta-descent* that we described in Section 3.2.2, can lead to more stable training procedures for meta-learning in non-stationary settings.

References

[1] Tiny-imagenet. `https://tiny-imagenet.herokuapp.com/`.

[2] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018.

[3] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. pages 7120–7129, 07 2017.

[4] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11816–11825. Curran Associates, Inc., 2019.

[5] Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless sequential learning. In *International Conference on Learning Representations*, 2019.

[6] Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *International Conference on Learning Representations*, 2018.

[7] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018.

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019.

[9] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On Tiny Episodic Memories in Continual Learning. *arXiv e-prints*, page arXiv:1902.10486, February 2019.

[10] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. In *International Conference on Learning Representations*, 2020.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[12] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1920–1930, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[13] Robert French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 05 1999.

[14] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks, 2020.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. 03 2015.

[16] Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, pages 1818–1828, 2019.

[17] Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9122–9133. Curran Associates, Inc., 2019.

[18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[19] 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*, 114(13):3521–3526, 2017.

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[21] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[22] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[23] Lei Le, Raksha Kumaraswamy, and Martha White. Learning sparse representations in reinforcement learning with sparse coding. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 2067–2073. AAAI Press, 2017.

[24] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[25] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations*, 2020.

[26] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

[27] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[28] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[29] James Mcclelland, Bruce Mcnaughton, and Randall O'Reilly. 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.

[30] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based RL. In *International Conference on Learning Representations*, 2019.

[31] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. *ArXiv*, abs/1812.07671, 2019.

[32] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[33] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[34] German Parisi, Ronald Kemker, Jose Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 02 2018.

[35] Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7647–7657. Curran Associates, Inc., 2019.

[36] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[37] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[38] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

[39] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *ArXiv*, abs/1606.04671, 2016.

[40] Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical Analysis of the Hessian of Over-Parametrized Neural Networks. *arXiv e-prints*, page arXiv:1706.04454, June 2017.

[41] Jürgen Schmidhuber. Evolutionary principles in self-referential learning. 1987.

[42] Nicol Schraudolph. Local gain adaptation in stochastic gradient descent. 06 1999.

[43] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4548–4557, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[44] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2990–2999. Curran Associates, Inc., 2017.

[45] Harkirat Singh Behl, Atılım Güneș Baydin, and Philip H. S. Torr. Alpha MAML: Adaptive Model-Agnostic Meta-Learning. *arXiv e-prints*, page arXiv:1905.07435, May 2019.

[46] Sebastian Thrun and Lorien Pratt, editors. *Learning to Learn*. Kluwer Academic Publishers, USA, 1998.

[47] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong Learning with Dynamically Expandable Networks. *arXiv e-prints*, page arXiv:1708.01547, August 2017.

[48] Ya-xiang Yuan. A review of trust region algorithms for optimization. *ICM99: Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, 09 1999.

[49] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task Agnostic Continual Learning Using Online Variational Bayes. *arXiv e-prints*, page arXiv:1803.10123, Mar 2018.