LOCAL VS GLOBAL CONTINUAL LEARNING

Giulia Lanzillotta[1,2], **Sidak Pal Singh**[2], **Benjamin F. Grewe**[3], **and Thomas Hofmann**[2]

[1]ETH AI Center, Switzerland
[2]Department of Computer Science, ETH Zurich, Switzerland
[3]Institute of Neuroinformatics - University of Zurich and ETH Zurich, Switzerland
Correspondence address: giulia.lanzillotta@ai.ethz.ch

ABSTRACT

Continual learning is the problem of integrating new information in a model while retaining the knowledge acquired in the past. Despite the tangible improvements achieved in recent years, the problem of continual learning is still an open one. A better understanding of the mechanisms behind the successes and failures of existing continual learning algorithms can unlock the development of new successful strategies. In this work, we view continual learning from the perspective of the *multi-task loss approximation*, and we compare two alternative strategies, namely *local* and *global* approximations. We classify existing continual learning algorithms based on the approximation used, and we assess the practical effects of this distinction in common continual learning settings. Additionally, we study optimal continual learning objectives in the case of local polynomial approximations and we provide examples of existing algorithms implementing the optimal objectives.

1 INTRODUCTION

Given the present trend toward training gigantic, foundational models (Achiam et al., 2023), effective continual learning solutions hold the key to reducing the computational costs of ever integrating new information into the model without forgetting older information. The alternative of retraining on the entire data to update the model is not an option at this scale, especially in the case where the models are expected to adapt quickly to a dynamic environment.

McCloskey & Cohen (1989) were the first to observe that neural networks trained on a sequence of tasks perform poorly on inputs from temporally antecedent tasks, a phenomenon termed *catastrophic forgetting*. Several algorithms have since been developed to limit catastrophic forgetting in deep neural networks (De Lange et al., 2021; Khetarpal et al., 2022). However the solutions developed so far struggle in real world scenarios, where satisfying either compute or memory constraints is crucial (Kontogianni et al., 2024; Garg et al., 2023). We believe that a better understanding of the problem of continual learning is needed to design algorithms that can address catastrophic forgetting effectively. Moreover, studying the failure cases of existing continual learning algorithms can point the way towards principled solutions.

In this work, we view continual learning as the problem of *approximating the multi-task loss* and we study existing continual learning algorithms in this light. In particular, we are interested in distinguishing between *local* and *global* approximations. Said simply, local approximations rely on information about the state of the network after learning each task, whereas global approximations don't. This characteristic implies that algorithms employing local approximations need to satisfy a restrictive assumption, which we bring to the surface. We classify continual learning algorithms as either *local* or *global*, based on the properties of the underlying approximation, and we evaluate experimentally the practical consequences of the two approximation schemes.

Contributions and paper structure. After covering some background (Section 2) we introduce the formulation of continual learning from the perspective of loss approximation (Section 3.1). In Section 3.2 we discuss two mutually exclusive approximation strategies, namely *local* and *global* approximations and we define the locality assumption, which is unavoidable for any algorithm employing a local approximation. Next (Section 4), we study the case of continual learning algorithms using polynomial local approximations, and we derive optimal objectives for the quadratic case. In Section 5 we consider a few classic examples from the literature and we classify them as either local or global algorithms. Further, we show formally that orthogonal gradient descent (Farajtabar et al., 2020) implements the optimal objective for local quadratic approximations derived in Section 4. Finally, we provide experimental evidence of our claims in Section 6, and we evaluate the practical implications of local and global approximations in common continual learning settings.

2 BACKGROUND

Consider a set of supervised learning tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$, with the data associated to the t-th task $\mathcal{T}_t$ being $D_t = \{(x, y) \in \mathcal{X}_t \times \mathcal{Y}_t\}$. Continual learning algorithms learn the tasks *sequentially*, whereby at each *learning step* they utilize the present task data to update the model, which is typically a neural network with parameters $\boldsymbol{\theta}$. In order to avoid forgetting the old tasks while learning a new one, the algorithm can often access an external memory, which is also updated after every task. The content of this memory may differ across algorithms. The performance on each task is measured by a task-specific loss function l_t, and the objective of each task is to minimise the expected loss: $\mathcal{L}_t(\boldsymbol{\theta}) = \langle l_t(x, y, \boldsymbol{\theta}) \rangle_{\mathcal{P}_t}$, where $\langle \cdot \rangle_{\mathcal{P}_t}$ denotes the average over the task distribution $\mathcal{P}_t$. In practice, the expectation is approximated by an average over the dataset, which is called the task empirical loss $L_t(\boldsymbol{\theta}) = \langle l_t(x, y, \boldsymbol{\theta}) \rangle_{D_t}$. The continual learning problem is to minimise the *multi-task loss*:

$$\mathcal{L}_t^{MT}(\boldsymbol{\theta}) = \frac{1}{t} \left(\mathcal{L}_1(\boldsymbol{\theta}) + \cdots + \mathcal{L}_t(\boldsymbol{\theta}) \right) \tag{1}$$

while only having direct access to the current task data D_t and the external memory. Other objectives may be considered instead of the multi-task loss, such as the average lifetime performance. We choose to use the multi-task loss objective in order to conform with the historically prevalent practice in the literature.

Notation & Metrics. We use $\boldsymbol{\theta} \in \Theta_t$ to refer to a generic vector in the parameter space and $\boldsymbol{\theta}_t$ to the value of the network parameters after t learning steps. We index the current task with t and any single old task with o. Given a current parameter vector $\boldsymbol{\theta}$ *catastrophic forgetting* on task $\mathcal{T}_o$ may be measured by the signed difference in expected loss $\mathcal{E}_o(\boldsymbol{\theta}) = \mathcal{L}_o(\boldsymbol{\theta}) - \mathcal{L}_o(\boldsymbol{\theta}_o)$. Similarly, the empirical approximation of $\mathcal{E}_o(\boldsymbol{\theta})$ is the signed change in the empirical task loss $E_o(\boldsymbol{\theta}) = L_o(\boldsymbol{\theta}) - L_o(\boldsymbol{\theta}_o)$. Alternatively, forgetting can be measured in terms of the task test accuracy as $\mathcal{E}_o^{acc}(\boldsymbol{\theta}) = \mathrm{ACC}_o(\boldsymbol{\theta}_o) - \mathrm{ACC}_o(\boldsymbol{\theta})$. Notice that in both cases a lower value of forgetting is better. Additionally, we denote the *average forgetting* after t steps by $E(t) = \frac{1}{t} \sum_{o=1}^{t} \mathcal{E}_o(\boldsymbol{\theta}_t)$ and the *average empirical forgetting* as $E(t)$. Likewise, the *average accuracy* after t steps by $\mathrm{ACC}(t) = \frac{1}{t} \sum_{o=1}^{t} \mathrm{ACC}_o(\boldsymbol{\theta}_t)$. We refer the reader to Section 9 (Appendix A) for a full overview of the notation used.

3 LOCAL AND GLOBAL APPROXIMATIONS IN CONTINUAL LEARNING

3.1 PROBLEM FORMULATION

In this work, we view continual learning algorithms from the perspective of *loss approximation*, which we introduce in this section. Consider the problem of minimizing the multi-task expected loss in Equation (1). If hypothetically all the data were available, one could approximate the distribution multi-task loss $\mathcal{L}_t^{MT}(\boldsymbol{\theta})$ by its *empirical version* $L_t^{MT}(\boldsymbol{\theta})$ and use it as the optimization objective, as is common practice in machine learning. However, a continual learning algorithm can only access the current task data D_t and the content of its external memory M_t, therefore, its objective should be, strictly speaking, a function of D_t and M_t alone: $\mathrm{obj}_t(D_t, M_t)$.

In this work, we are interested in the way in which the (explicit or implicit) objective of a continual learning algorithm $\mathrm{obj}_t(D_t, M_t)$ approximates the true objective, i.e. the multi-task loss. In particular, we consider the relation between $L_t^{MT}(\boldsymbol{\theta})$ and $\mathrm{obj}_t(D_t, M_t)$ in order to focus on the locality (or non-locality) of the approximation, and we set aside a discussion of the generalization gap (i.e., whether $L_t^{MT}(\boldsymbol{\theta}) \approx \mathcal{L}_t^{MT}(\boldsymbol{\theta})$) . Accordingly, we view the continual learning problem as follows:

$$\min_{\boldsymbol{\theta} \in \Theta} \ \mathrm{obj}_t(D_t, M_t)(\boldsymbol{\theta}) \qquad s.t. \ \ \mathrm{obj}_t(D_t, M_t)(\boldsymbol{\theta}) \approx L_t^{MT}(\boldsymbol{\theta}) \tag{2}$$

To conform with existing algorithms, we hereafter study approximate objectives $\mathrm{obj}_t(D_t, M_t)$ of the form:

$$\mathrm{obj}_t(D_t, M_t) = \hat{L}_t^{MT}(\boldsymbol{\theta}) := \frac{1}{t} \left(\hat{L}_1(\boldsymbol{\theta}) + \cdots + \hat{L}_{t-1}(\boldsymbol{\theta}) + L_t(\boldsymbol{\theta}) \right) \tag{3}$$

In short, each previous task loss is approximated separately as $\hat{L}_i(\boldsymbol{\theta}) \approx L_i(\boldsymbol{\theta})$. We want to stress that such an approximation $\hat{L}_i(\boldsymbol{\theta})$ need not always be explicit. For some cases, we infer the approximation used from the objective which the algorithm effectively minimizes at each learning step. This distinction will become clear in Section 5, where we review examples from the literature.

3.2 LOCAL AND GLOBAL APPROXIMATIONS

We are now ready to introduce the central point of our discussion. Starting from the formulation of continual learning introduced in the previous section (Equation (2)), we are interested in asking whether the task loss approximation $\hat{L}_t(\boldsymbol{\theta})$ is *local* or *global*.

In general, a *local* approximation of a function $f(x)$ makes use of information about the function at a particular point x_0 to produce a good approximation of f in a neighborhood of x_0. Correspondingly, we say that the task loss approximation $\hat{L}_t(\boldsymbol{\theta})$ is *local* when it uses information about the function at the task solution $\boldsymbol{\theta}_t$ and it is accurate in a neighborhood of $\boldsymbol{\theta}_t$. Conversely, we say that the approximation is *global* when it is independent of $\boldsymbol{\theta}_t$, meaning that modifying $\boldsymbol{\theta}_t$ would not change the approximation. We formalise the notion of local and global approximations in Definition 3.1.

Definition 3.1 (Local and global task loss approximation.)**.** Let $I(X;Y)$ denote the mutual information of the pair of random variables (X, Y). We say that the task loss approximation $\hat{L}_t(\boldsymbol{\theta})$ is *local* when $I(\hat{L}_t(\boldsymbol{\theta}); \boldsymbol{\theta}_t) > 0 \; \forall \; \boldsymbol{\theta} \in \Theta$, and that it is *global* when $I(\hat{L}_t(\boldsymbol{\theta}); \boldsymbol{\theta}_t) = 0 \; \forall \; \boldsymbol{\theta} \in \Theta \setminus \{\boldsymbol{\theta}_t\}$.

For illustrative purposes, consider the constant function approximation $\hat{L}_t(\boldsymbol{\theta}) = C$. A local approximation could be $\hat{L}_t(\boldsymbol{\theta}) = L_t(\boldsymbol{\theta}_t)$, and a global approximation could be $\hat{L}_t(\boldsymbol{\theta}) = 0$. Although it is intuitively clear that the former approximation relies on information of the task solution $\boldsymbol{\theta}_t$, in general one could evaluate $\mathcal{D}_{KL}(\mathcal{P}(L_t(\boldsymbol{\theta}_t), \boldsymbol{\theta}_t) \| \mathcal{P}(L_t(\boldsymbol{\theta}_t)) \cdot \mathcal{P}(\boldsymbol{\theta}_t)) > 0$, given any distribution $\mathcal{P}(\boldsymbol{\theta}_t)$.

Let $\xi_t(\boldsymbol{\theta}) = |\hat{L}_t(\boldsymbol{\theta}) - L_t(\boldsymbol{\theta})|$ be the approximation error in $\boldsymbol{\theta}$. We define the *ϵ-region* of the approximation as $\Omega_t^\epsilon = \{\boldsymbol{\theta} \in \Theta : \xi_t(\boldsymbol{\theta}) < \epsilon\}$. For local approximations Ω_t^ϵ is always a neighborhood of $\boldsymbol{\theta}_t$ (this may be another definition of local approximations), while it may be a disjoint set of points for a global approximation. Notice that the global approximation is not necessarily more accurate than a local approximation. Depending on the case, the ϵ-region of a local approximation may have more volume than the ϵ-region of a global approximation, and thus be 'less wrong' on average.

This brings us to state the main assumption for continual learning algorithms using local approximations. Hereafter, we say a continual learning algorithm is *local* if it uses a local approximation of the task loss function, and *global* if it uses a global approximation instead. It follows that local continual learning algorithms effectively reduce forgetting only with the additional assumption that *learning is localised*, which we dub *the locality assumption*.

Assumption 3.2 (Locality assumption)**.** Given a local approximation with ϵ-region Ω_t^ϵ for task t and arbitrary ϵ, the following condition holds while learning task $t+1$:

$$\boldsymbol{\theta} \in \Omega_1^\epsilon \cap \cdots \cap \Omega_t^\epsilon \tag{4}$$

Simply put, for a continual learning algorithm producing the sequence of solutions $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t$, the locality assumption requires that all the solutions lie relatively close to each other. The error of a local task loss approximation is higher the farther away from the task solution. Thus, for a given error tolerance ϵ the locality assumption is broken when the distance between the task solutions is "too high". In Section 6, we break the locality assumption by artificially increasing the distance between task solutions, and we show that local algorithms struggle in this setting.

4 Case study: local polynomial approximations

A general example of local approximation of $L_t(\boldsymbol{\theta})$ is the Taylor expansion around $\boldsymbol{\theta}_t$:

$$\hat{L}_t(\boldsymbol{\theta}) = L_t(\boldsymbol{\theta}_t) + (\boldsymbol{\theta} - \boldsymbol{\theta}_t)^\intercal \boldsymbol{\nabla} L_t(\boldsymbol{\theta}_t) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_t)^\intercal \frac{\partial^2 \mathcal{L}_t(\boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\intercal} (\boldsymbol{\theta} - \boldsymbol{\theta}_t) + \ldots \tag{5}$$

The approximation error for a p-order approximation is upper bounded by $O(\|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^p)$, and thus the ϵ-region is simply a p-norm ball around $\boldsymbol{\theta}_t$ with radius proportional bounded by ϵ: $\Omega_t^\epsilon = \{\boldsymbol{\theta} : C \cdot \|\boldsymbol{\theta} - \boldsymbol{\theta}_t\|^p < \epsilon\}$. A convenient property of this approximation is that it allows us to express forgetting in terms of the loss derivatives in $\boldsymbol{\theta}_t$. Indeed forgetting is simply $E_t(\boldsymbol{\theta}) = \hat{L}_t(\boldsymbol{\theta}) - L_t(\boldsymbol{\theta}_t)$.

4.1 Quadratic local approximations

A wealth of studies have shown that for over-parameterized networks the loss tends to be very well-behaved and almost convex in a reasonable neighborhood around the minima (Saxe et al., 2013; Choromanska et al., 2015; Jacot et al., 2020). Therefore in such cases a quadratic approximation of $L_t(\boldsymbol{\theta})$ might be accurate.

If we write the Hessian matrix $\frac{\partial^2 \mathcal{L}_t(\boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\intercal}$ as $\mathbf{H}_t^\star$ and denote the change in parameters due to learning a new task $\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}$ as $\boldsymbol{\Delta}_t$, following Equation (5) we can write the forgetting on task o after learning t as:

$$E_o(\boldsymbol{\theta}_t) = (\boldsymbol{\theta}_t - \boldsymbol{\theta}_o)^\intercal \boldsymbol{\nabla} L_o(\boldsymbol{\theta}_o) + \frac{1}{2} (\boldsymbol{\theta}_t - \boldsymbol{\theta}_o)^\intercal \mathbf{H}_o^\star (\boldsymbol{\theta}_t - \boldsymbol{\theta}_o) \tag{6}$$

In order to highlight the contribution of the latest parameter update to the average forgetting $E(t) = \frac{1}{t-1}\sum_{o=1}^{t-1} E_o(\boldsymbol{\theta}_t)$, we can formulate it in a recursive fashion:

$$E(t) = \frac{t-1}{t}\cdot E(t-1) + \underbrace{\frac{1}{t}\cdot \boldsymbol{\Delta}_t^\intercal \left(\sum_{o=1}^{t-1} \boldsymbol{\nabla} L_o(\boldsymbol{\theta}_o)\right) + \frac{1}{2t}\cdot \boldsymbol{\Delta}_t^\intercal \left(\sum_{o=1}^{t-1} \mathbf{H}_o^\star\right)\boldsymbol{\Delta}_t + \frac{1}{t}\boldsymbol{v}^\intercal \boldsymbol{\Delta}_t}_{\text{Additional forgetting due to learning task } t\text{, i.e., } \boldsymbol{\Delta}_t} \tag{7}$$

where $\boldsymbol{v}^\intercal$ denotes the vector $\sum_{o=1}^{t-2}(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_o)^\intercal \mathbf{H}_o^\star$.

Example. Let us consider the case of $t = 2$ as an example, for which $\boldsymbol{v} = \mathbf{0}$. The forgetting of the first task after learning the second is simply: $E(2) = \boldsymbol{\Delta}_2^\intercal \boldsymbol{\nabla} L_1(\boldsymbol{\theta}_1) + \boldsymbol{\Delta}_2^\intercal \mathbf{H}_1^\star \boldsymbol{\Delta}_2$. Putting everything together, the objective when learning the second task is:

$$\min_{\boldsymbol{\Delta}_2 \in \Theta} \{L_2(\boldsymbol{\theta}_1 + \boldsymbol{\Delta}_2) + \boldsymbol{\Delta}_2^\intercal \boldsymbol{\nabla} L_1(\boldsymbol{\theta}_1) + \boldsymbol{\Delta}_2^\intercal \mathbf{H}_1^\star \boldsymbol{\Delta}_2\} \tag{8}$$

Notice that minimizing this objective with respect to $\boldsymbol{\Delta}_2$ is equivalent to minimizing the multi-task loss $L_2 + L_1$. If we now assume that $\boldsymbol{\theta}_1$ is a *local minima* of L_1, we get $\boldsymbol{\nabla} L_1(\boldsymbol{\theta}_1) = 0$ and positive semi-definite Hessian $\mathbf{H}_1^\star \succcurlyeq 0$, from which it follows that $E(2) \geq 0$. In this case the objective Equation (8) can be rewritten as:

$$\min_{\boldsymbol{\Delta}_2 \in \Theta} L_2(\boldsymbol{\theta}_1 + \boldsymbol{\Delta}_2) \quad s.t. \;\; \boldsymbol{\Delta}_2^\intercal \mathbf{H}_1^\star \boldsymbol{\Delta}_2 = 0 \tag{9}$$

Repeating the same procedure for every following task ($t > 2$) we can write the optimal learning objective for any new task under a quadratic local approximation of the loss. We state our result in Theorem 4.1.

Theorem 4.1 (Optimal quadratic local continual learning)**.** *For any continual learning algorithm producing a sequence of parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t$ such that $\boldsymbol{\theta}_i$ is a local minima of L_i and $\sup_{\boldsymbol{\theta}_i, \boldsymbol{\theta}_k} \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_k\|^3 < \epsilon$ the following relationship holds:*

$$E(1), \ldots, E(t-1) = 0 \implies E(t) = \frac{1}{2}\boldsymbol{\Delta}_t^\intercal \left(\frac{1}{t}\cdot \sum_{o=1}^{t-1} \mathbf{H}_o^\star\right)\boldsymbol{\Delta}_t \geq 0 \tag{10}$$

Moreover, if $E(1), \ldots, E(t-1) = 0$ the optimal learning objective for task t is:

$$\min_{\boldsymbol{\Delta}_t \in \Theta} L_t(\boldsymbol{\theta}_{t-1} + \boldsymbol{\Delta}_t) \quad s.t. \;\; \boldsymbol{\Delta}_t^\intercal \left(\frac{1}{t}\cdot \sum_{o=1}^{t-1} \mathbf{H}_o^\star\right)\boldsymbol{\Delta}_t = 0 \tag{11}$$

There is an intuitive explanation of Theorem 4.1: when a quadratic approximation to the loss is accurate enough and each task solution is a local minimum, the parameter updates must be taken along directions where the multi-task loss landscape is (absolutely) flat in order to prevent forgetting. More formally, the constraint in Equation (11) is forcing the update $\boldsymbol{\Delta}_t$ to lie in the *null-space* of the *average Hessian matrix* $\overline{\mathbf{H}}_{<t}^\star := \frac{1}{t}\sum_{o=1}^{t-1} \mathbf{H}_o^\star$.

At this point, it is natural to ask whether there exists a solution to the objective given by Theorem 4.1. Previous studies (Sagun et al., 2016; 2017) have observed that the loss landscape of deep neural networks is mostly degenerate around local optima, implying that most of the eigenvalues of the loss Hessian lie near zero. In other words, the rank of $\mathbf{H}_o^\star$ is rather small, and as shown theoretically in the case of deep linear networks (Singh et al., 2021), this is of the order square-root the number of parameters. In the Appendix (Figure 4) we plot the effective rank of the multi-task loss Hessian matrix as a function of t and we find in practice that the rank of $\overline{\mathbf{H}}_{<t}^\star$ grows sub-linearly in t.

Nevertheless, there may be settings for which the condition in Equation (11) is unsatisfiable (e.g. the multi-task loss Hessian matrix is full rank) or leads to poor task solutions (thereby violating the assumptions of Theorem 4.1). Equation (7) describes forgetting in a more general case with no other assumptions than locality. Generally, for quadratic local approximations of the loss, the optimal learning objective for a given task is:

$$\min_{\boldsymbol{\Delta}_t \in \Theta} \{\; L_t(\boldsymbol{\theta}_{t-1} + \boldsymbol{\Delta}_t) + \boldsymbol{\Delta}_t^\intercal (\overline{\boldsymbol{\nabla} L}_{<t} + \frac{1}{t}\boldsymbol{v}) + \frac{1}{2}\boldsymbol{\Delta}_t^\intercal \overline{\mathbf{H}}_{<t}^\star \boldsymbol{\Delta}_t \;\} \tag{12}$$

where we have introduced the abbreviation $\overline{\boldsymbol{\nabla} L}_{<t} := \frac{1}{t}\sum_{o=1}^{t-1} \boldsymbol{\nabla} L_o(\boldsymbol{\theta}_o)$. Notice that Equation (12) also favours parameter updates which lie in the null space of the multi-task loss Hessian matrix. However while Equation (11) employs a hard constraint effectively reducing the space of solutions, Equation (12) employs a soft constraint, potentially trading-off forgetting with better performance on the task. In Section 5, we review examples of existing algorithms that implement the hard and soft constraints.

The role of the multi-task loss Hessian matrix in the local learning objectives (Equations (11) and (12)) explains the observation in the literature that *"flatter local minima"* (Keskar et al., 2016) result in reduced forgetting (Deng et al., 2021; Mirzadeh et al., 2020a). Under a local quadratic approximation, the flatter the previous task minima, the larger the space of solutions to the current task where forgetting is 0 or close to 0. Therefore *continual learning algorithms favouring flatter landscapes such as (Deng et al., 2021) implicitly rely on a local approximation of the task loss*, and thereby they are successful strategies only when learning is localised.

5 Local and Global algorithms in the literature

We now review some existing continual learning algorithms, classifying them into local and global algorithms. We select a few well known exemplars, representing different families of algorithms according to popular taxonomies of the literature (De Lange et al., 2021).

5.1 Global algorithms

Experience Replay (ER) is one of the oldest (Robins, 1995) and still one of the most effective (Buzzega et al., 2021) algorithms for continual learning. Although several variants have been proposed (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017; Shin et al., 2017; Van de Ven et al., 2020) for now we consider its simplest form. For each task, a random subset of the dataset $S_t \subset D_t$ is stored in an external bufferto approximate the task loss $L_t(\boldsymbol{\theta})$ as follows:

$$\hat{L}_t(\boldsymbol{\theta}) = \frac{1}{|S_t|} \sum_{x,y \in S_t} l_t(x, y, \boldsymbol{\theta}) \tag{13}$$

The objective of each learning step is that of Equation (3). As long as the buffer sampling strategy does not depend on the network parameters after learning the task, the approximation used by experience replay is global. The approximation error is a function of the buffer size and it has been observed that in most cases the algorithm is effective also when the buffer size is small (Buzzega et al., 2020; 2021).

Gradient Episodic Memory (GEM) (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018) employs the same task loss approximation in a different way. Instead of directly minimizing the multi-task loss, the objective of GEM is:

$$\min_{\boldsymbol{\Delta}_t \in \Theta} L_t(\boldsymbol{\theta}_{t-1} + \boldsymbol{\Delta}_t) \qquad s.t.\ \hat{L}_o(\boldsymbol{\theta}_{t-1} + \boldsymbol{\Delta}_t) \leq \hat{L}_o(\boldsymbol{\theta}_{t-1})\ \ \forall o < t \tag{14}$$

In other words, the parameter update $\boldsymbol{\Delta}_t$ does not minimise the approximate task loss $\hat{L}_o$ but it does not increase it (thus avoiding catastrophic forgetting). In order to enforce the constraint in Equation (14), GEM uses a local linearization of the old task loss, which is updated after each optimization step. The linearization may be inaccurate when learning with large gradient step sizes, and result in reduced performance. Nevertheless, the task loss approximation is based on the current state of the network rather than its state after learning the task, which makes this algorithm global.

Synaptic Intelligence (SI) (Zenke et al., 2017) uses a quadratic approximation of the old task loss, centered around the previous task solution. For a current task t the approximation of L_o is:

$$\hat{L}_o(\boldsymbol{\theta}) = (\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1})^\intercal \hat{H}_o (\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1}) \tag{15}$$

where $\hat{H}_o$ is a diagonal matrix updated at each gradient step, which roughly estimates the task Hessian matrix evaluated at $\boldsymbol{\theta}_{t-1}$. The objective of each learning step is that of Equation (3). The approximation of L_o is updated after each task based on the current state of the network $\boldsymbol{\theta}_t$, therefore SI also belongs to the group of global algorithms. Similarly to GEM, SI is sensitive to large step sizes, as the quadratic approximation may be inaccurate if the distance travelled in the parameter space is large.

Progressive Neural networks (PNN) (Rusu et al., 2016) belong to the category of *network expansion* methods, which dynamically allocate new parameters of the neural network to each task. Specifically, PNN subsequently adds '*columns*' (parametrized by feed-forward networks) to the neural network with unilateral connections between them. Although the whole network is used to produce outputs, only the parameters of the last column are trained on the current task, while the others are *frozen*. In more general terms, let $\Theta^1, \ldots, \Theta^t$ denote the parameter subspace associated to each task. By design the modified derivative of the task loss L_t is:

$$\frac{\partial \tilde{L}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} = \begin{cases} \dfrac{\partial L_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} & \text{if } \boldsymbol{\theta}_i \in \Theta^t \\ 0 & \text{if } \boldsymbol{\theta}_i \notin \Theta^t \end{cases} \tag{16}$$

For all parameters Θ^b with $b > t$, both the modified derivative and the true derivative of L_t are 0 (because the parameters do not enter the output computation). However for all parameters Θ^b , $b < t$, the true derivative might be non zero. As a consequence, the approximation $\tilde{L}_t$ is more inaccurate as t increases. However, forgetting is fixed to zero because the parameters are split among tasks. More precisely, a change in the parameters $\boldsymbol{\theta}_i \in \Theta_t$ has no effect on the forgetting on task $o < t$: $\partial E_o(\boldsymbol{\theta})/\partial \boldsymbol{\theta}_i = \partial(L_o(\boldsymbol{\theta}) - L_o(\boldsymbol{\theta}_o))/\partial \boldsymbol{\theta}_i = 0$. As a consequence, it is not only impossible to forget but it is also impossible to *improve* the performance on previous tasks (when this happens we say there is *backward transfer*). Finally, note that the PNN approximation uses the current state of the network to compute the modified derivative, therefore the algorithm is global.

Other algorithms effectively partitioning the parameter space into task-specific subspaces (such as (Mallya & Lazebnik, 2018; Van Etten et al., 2018; Serra et al., 2018)) are functionally equivalent to PNN. In particular, this family of approaches avoids catastrophic forgetting by confining the parameter update to a task-specific subspace, at the cost of potentially suboptimal task solutions , and no backward transfer. From this point of view, an interesting parallel emerges between this family of algorithms and the general algorithm described by Theorem 4.1, with the critical difference that the algorithms of Theorem 4.1 are local, while methods like PNN are global.

5.2 Local algorithms

Second-order regularization. Yin et al. (2020) have recently demonstrated that a large group of continual learning algorithms which use quadratic regularizers to prevent forgetting effectively employ a second order local approximation of the tasks loss function (discussed in Section 4), which shows that these algorithms are local. More specifically, Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) and Structured Laplace Approximation Ritter et al. (2018) minimize the optimal objective of Equation (12), and differ in their assumptions on the hessian matrix.

iCarl (Rebuffi et al., 2017) uses experience replay to avoid catastrophic forgetting. There are several elements contributing to the final form of the algorithm, such as the use of network-generated targets instead of labels to approximate the task loss, and the non-parametric classifier based on the network features. Crucially, the samples in the task replay buffer are selected after the task has been learned in order to best approximate the true feature class mean. The selection procedure, based on herding, inevitably depends on the value of the network parameters after learning the task, as the network features are determined by the parameters. Substantial transformations of the feature map will impact the task-loss approximation accuracy. Therefore, iCarl belongs to the set of local algorithms due to this prioritized buffer selection procedure. In Section 6 we experimentally verify our claim by changing the algorithm's sampling strategy.

Orthogonal gradient descent (OGD) (Farajtabar et al., 2020) avoids catastrophic forgetting by enforcing orthogonality between the parameter update and the previous tasks output-gradients. In order to do so efficiently, a set of the task output-gradient vectors is stored in a buffer at the end of the task. By doing so, the algorithm uses a local approximation of the task loss function. In Theorem 5.1 we show that OGD implements the optimal continual learning objective under local quadratic approximations of the loss (Theorem 4.1).

Theorem 5.1. *Let $\boldsymbol{\Delta}_1, \dots, \boldsymbol{\Delta}_t$ be the sequence of updates produced by Orthogonal Gradient Descent on a sequence of t tasks and let $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_t$ be the corresponding parameters. If $\sup_{\boldsymbol{\theta}_i, \boldsymbol{\theta}_k} \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_k\|^3 < \epsilon$, then $\boldsymbol{\Delta}_t$ satisfies the constraint in Equation (11) and $E_t = 0$ for all learning steps t.*

This result establishes a direct link between OGD and second-order regularization methods such as EWC: in settings where a quadratic approximation of the task loss is accurate, both methods minimize the optimal learning objective, with the difference that OGD relies on a hard constrain (Equation (11)) and regularization methods use soft constraints (Equation (12)). As a consequence, OGD implements a more conservative approach, potentially sacrificing performance on new tasks in order to maintain performance on old tasks, whereas EWC and the like strike a balance between new and old tasks performance which is determined by the regularization strength hyperparameter.

6 Experiments

Experimental setup. With our experiments we want to evaluate the practical side of the theoretical distinction between local and global algorithms. We take several classic algorithms in the literature which are representative of the different families of algorithms, namely: Expericence replay (ER), Averaged GEM (A-GEM, a computationally cheap variant of GEM), Elastic Weight Consolidation (EWC), Synaptic Intelligence (SI), iCarl and Orthogonal Gradient Descent (OGD), which were discussed in Section 5. We do not include Progressive Neural networks (PNN) in the experiments because we are interested in forgetting, which by design is always 0 for PNN.

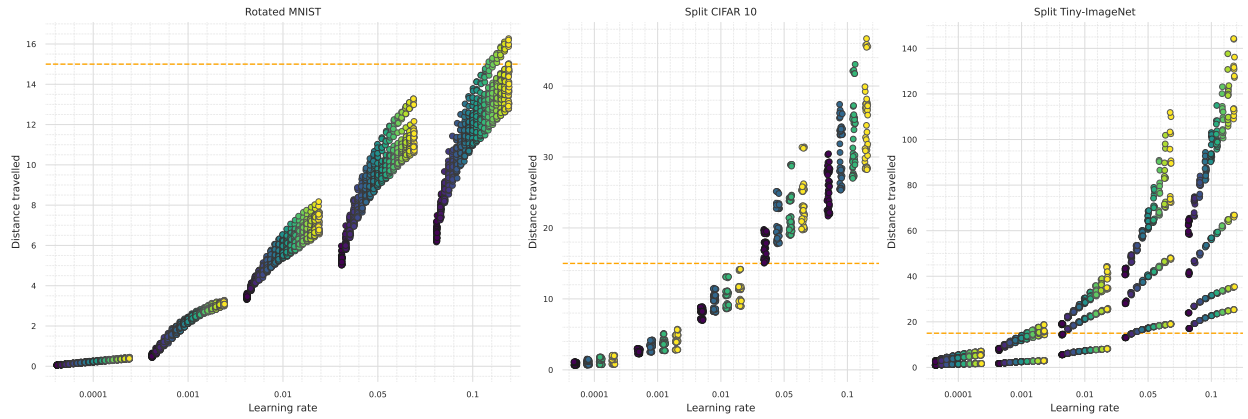

Figure 1: Distance travelled in the parameter space as a function of the optimizer learning rate and the task. We use a color coding of the tasks (a brighter color corresponding to a later task) . For each task, we measure the Euclidean distance between $\boldsymbol{\theta}_t$ and the initialization $\boldsymbol{\theta}_0$. We plot results over all algorithms and random seeds (for a total of 5). Finally, the yellow dashed line is provided as a reference of the relative scale of the y-axes across datasets.

We employ the benchmarking codebase developed by Boschini et al. (2022); Buzzega et al. (2020)[1]. In particular, we carry out our experiments on three popular continual learning challenges: **split CIFAR-10** (Krizhevsky et al., 2009), **split Tiny-ImageNet** (Wu et al., 2017) and **Rotated-MNIST** (LeCun et al., 1998; Lopez-Paz & Ranzato, 2017). In line with the *task-* and *class-incremental* settings, the first two challenges consist in splitting the original dataset into 5 and 10 tasks, each introducing 2 and 20 new classes, respectively. The Rotated-MNIST challenge belongs instead to the *domain-incremental* setting, and it consists of 20 subsequent classification tasks on the same 10 MNIST classes, where the inputs to each task are all rotated by an angle in the interval $[0, \pi)$. For a review of the difference between task-incremental, class-incremental and domain-incremental settings we refer the reader to (Hsu et al., 2018; Van de Ven & Tolias, 2019). Since iCarl cannot be applied to the domain-IL setting, we do not include it in the Rotated-MNIST experiments.

For each continual learning challenge, our choice of network architectures conforms to the standard practice in the literature. Specifically, for the MNIST-based challenge we employ a fully-connected network with two hidden layers, each one comprising of 100 ReLU units. For the challenges based on CIFAR-10 and Tiny ImageNet, we employ a ResNet18 (He et al., 2016). We adopt the standard hyper-parameter configuration and network training setup in (Buzzega et al., 2020), which has been selected by grid-search. We refer the reader to Section 10 in the Appendix for a detailed characterization of the experiment configurations and hyperparameters.

6.1 Local vs Global

Our main experiment consists in comparing local and global algorithms in settings where the locality assumption (Assumption 3.2) holds and settings where it doesn't. Recall that for a local approximation of the task loss L_t, the ϵ-region is a neighborhood of the task solution $\boldsymbol{\theta}_t$. For example, for a polynomial approximation of degree p, the ϵ-region is a p-norm ball around $\boldsymbol{\theta}_t$ of radius ϵ. Roughly, the higher the distance travelled between tasks in the parameter space, the higher the loss approximation error for local approximations. And we say that the locality assumption is broken when this error is higher than a given tolerance of ϵ.

We break the locality assumption by artificially increasing the Euclidean distance between task solutions, i.e. $\|\boldsymbol{\Delta}_t\|_2$. There are a number of factors which determine $\|\boldsymbol{\Delta}_t\|_2$, including the learning rate, the number of epochs, the batch size and the dataset size. We choose to adjust the learning rate while keeping all the other factors constant. More explicitly, we repeat all our experiments for different learning rates, which are approximately equidistant on a logarithmic scale. By varying the learning rate in a wide range, we smoothly interpolate between a *local learning* setting and a *non-local learning* setting.

We verify our methodological choice by plotting the distance travelled in the parameter space as a function of the learning rate in Figure 1. For all our configurations we see that higher learning rates indeed result in a higher distance from initialization, and, consequently higher $\|\boldsymbol{\Delta}_t\|_2$. Interestingly, for low learning rates the curves look homogeneous

[1]The codebase is publicly available at https://github.com/aimagelab/mammoth

across algorithms, whereas they diverge as the learning rate increases, suggesting that the learning dynamics between algorithms are relatively similar in the local learning setting.

Table 1: **Main experiments results.** We mark by – non-existent experiment configurations. The * symbol on the OGD-TinyImagenet experiments indicates the use of a lower memory size (see Section 10 for details).

ALGORITHM	LR	S-CIFAR-10		ROT-MNIST		S-TINY-IMAGENET	
		ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)	ACC (↑)	FGT (↓)
A-GEM	10^{-4}	78.56 ± 2.59	13.39 ± 3.48	-	-	24.65 ± 1.29	**38.25** ± 1.21
	10^{-3}	83.65 ± 1.94	13.6 ± 2.626	74.22 ± 5.75	**12.23** ± 6.01	18.12 ± 0.78	58.41 ± 0.83
	0.005	-	-	79.79 ± 5.21	14.07 ± 5.47	-	-
	10^{-2}	**86.45** ± 2.664	**12.50** ± 3.54	81.21 ± 4.84	14.41 ± 5.12	24.63 ± 0.73	58.36 ± 0.36
	0.05	83.27 ± 2.46	16.96 ± 3.364	82.01 ± 4.97	15.98 ± 5.26	26.65 ± 0.09	57.06 ± 1.02
	0.1	80.36 ± 1.95	20.55 ± 2.50	**82.35** ± 5.46	16.07 ± 5.72	**29.12** ± 0.69	54.08 ± 0.90
ER	10^{-4}	78.61 ± 1.599	10.42 ± 1.94	-	-	34.74 ± 0.44	**19.75** ± 0.66
	10^{-3}	84.97 ± 1.46	11.08 ± 1.68	78.88 ± 0.68	**2.57** ± 0.54	30.58 ± 0.39	40.44 ± 0.31
	0.005	-	-	85.79 ± 0.44	5.69 ± 0.41	-	-
	10^{-2}	90.42 ± 0.69	7.35 ± 1.08	87.03 ± 0.74	6.76 ± 0.76	42.69 ± 0.72	35.49 ± 0.65
	0.05	**91.98** ± 0.93	**6.20** ± 0.92	87.99 ± 1.00	8.91 ± 1.07	**50.78** ± 0.43	30.33 ± 0.64
	0.1	91.74 ± 0.70	6.48 ± 0.82	**88.57** ± 1.04	9.05 ± 1.10	49.20 ± 0.52	31.76 ± 0.61
SI	10^{-4}	68.30 ± 2.67	**27.19** ± 3.26	-	-	14.62 ± 1.21	48.41 ± 1.04
	10^{-3}	**71.03** ± 3.05	29.27 ± 3.28	70.42 ± 3.15	16.09 ± 3.45	13.82 ± 0.93	61.08 ± 1.19
	0.005	-	-	72.19 ± 2.54	21.80 ± 2.68	-	-
	10^{-2}	63.91 ± 1.71	40.29 ± 2.24	73.83 ± 2.76	21.78 ± 2.88	17.84 ± 1.64	58.53 ± 1.10
	0.05	66.66 ± 4.77	35.99 ± 5.97	81.26 ± 2.57	15.41 ± 2.57	45.35 ± 2.30	20.89 ± 2.85
	0.1	68.20 ± 3.94	33.75 ± 5.08	**85.08** ± 2.22	**10.76** ± 2.14	**52.83** ± 3.96	**12.91** ± 4.11
EWC	10^{-4}	61.78 ± 3.31	**32.63** ± 4.47	-	-	13.08 ± 1.47	**39.33** ± 2.10
	10^{-3}	61.40 ± 3.41	39.90 ± 4.85	70.91 ± 2.96	**15.42** ± 3.23	11.96 ± 0.51	58.29 ± 0.99
	0.005	-	-	72.40 ± 2.65	21.52 ± 2.79	-	-
	10^{-2}	62.29 ± 3.09	40.73 ± 4.56	73.18 ± 3.53	22.54 ± 3.72	13.99 ± 0.53	64.59 ± 0.75
	0.05	**62.50** ± 7.20	**28.69** ± 14.03	75.18 ± 4.42	22.88 ± 4.59	16.17 ± 1.29	62.88 ± 1.14
	0.1	54.01 ± 1.99	43.56 ± 11.95	**75.89** ± 3.76	22.57 ± 3.89	**17.18** ± 1.71	58.76 ± 1.77
ICARL	10^{-4}	78.92 ± 0.68	2.49 ± 0.64	-	-	15.91 ± 1.19	1.60 ± 0.73
	10^{-3}	92.45 ± 0.36	**0.97** ± 0.30	-	-	31.05 ± 0.70	**1.23** ± 0.30
	0.005	-	-	-	-	-	-
	10^{-2}	**93.46** ± 0.55	3.31 ± 0.56	-	-	54.13 ±0.20	6.08 ±0.32
	0.05	91.25 ± 0.78	5.64 ± 1.17	-	-	59.45 0.36	11.71 0.33
	0.1	89.76 ± 1.00	5.05 ± 1.56	-	-	**59.93** 0.68	13.76 0.44
OGD	10^{-4}	**65.87** ± 4.28	**27.26** ± 5.37	-	-	12.04* ± 1.97	**40.82*** ± 0.54
	10^{-3}	64.99 ± 4.46	35.99 ± 4.98	**70.43** ± 3.22	**16.09** ± 3.53	12.45* ± 0.81	59.29* ± 1.23
	0.005	-	-	70.23 ± 3.60	24.01 ± 3.78	-	-
	10^{-2}	62.64 ± 4.23	41.90 ± 5.40	69.78 ± 4.65	26.36 ± 4.86	16.01* ± 0.67	64.84* ± 0.23
	0.05	61.02 ± 7.35	40.87 ± 7.91	68.66 ± 5.71	30.02 ± 5.98	17.45* ± 0.18	64.01* ± 0.32
	0.1	64.66 ± 6.45	37.70 ± 7.57	69.06 ± 5.28	30.04 ± 5.51	**19.97*** ± 1.11	59.56* ± 1.56

Main experiment. We are now ready to look at the result of our main experiment. In Table 1 we report forgetting and average accuracy as a function of the learning rate for different algorithms. Forgetting is measured after the last task (T) has been learned, in terms of the task test accuracy: $\mathcal{E}^{acc}(T) = \frac{1}{T}\sum_o \text{ACC}_o(\boldsymbol{\theta}_o) - \text{ACC}_o(\boldsymbol{\theta}_T)$.

Our hypothesis is that *forgetting is higher for local algorithms in the non-local learning setting*, which corresponds to higher learning rates in our experiments. Further, in (Section 5) we claim that algorithms such as OGD, EWC, iCarl (marked in red) are local whereas algorithms such as A-GEM, ER and SI (marked in blue) are global. From our results (Table 1) we observe that local algorithms always achieve the lowest forgetting within the the lower end of the learning rate range considered (i.e. $lr \in \{10^{-4}, 10^{-3}\}$) across all configurations. On the other hand, for global algorithms, there is no discernible correlation between the learning rate and forgetting. Alternatively, take the difference in average forgetting between the lowest two learning rates and the highest two learning rates, for each challenge. For local algorithms, the difference is always negative, whereas for global algorithms the difference can be both positive and negative. Overall, the experiment results confirm our expectations on the functional aspects of

the algorithms, namely that forgetting is higher for local algorithms when the locality assumption is not met. This demonstrates that the difference between local and global approximations is not merely a theoretical one, as it has practical implications, observable with carefully designed experiments.

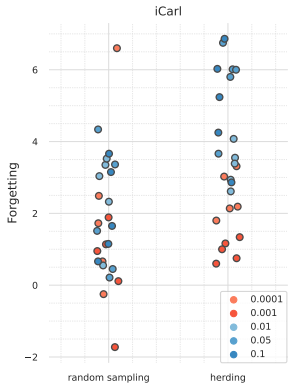

Figure 2: Comparison of random sampling (ours) and herding (standard) buffer selection strategies in iCarl. Higher learning rates, associated with non-local learning are shown in shades of blue, while learning rates associated with local learning are shown in shades of red. Each experiment is repeated over 5 random seeds (plotted as different points).

In addition to these immediate conclusions, from Table 1 we also observe that average accuracy and forgetting may be at odds with each other. This contrast reflects the tension between *stability and plasticity* in learning, termed the *stability-plasticity dilemma* in the literature (Mermillod et al., 2013), which may be summarised as follows: highly plastic but unstable models can quickly achieve high accuracy on any new task but forget as quickly the old ones; conversely, stable but rigid models avoid forgetting at the cost of lower average accuracy on all tasks. Higher learning rates (with no decay) render the network more plastic, leading to higher accuracy overall. In Table 1 we see that global algorithms especially benefit from higher plasticity, as forgetting is homogeneous across learning settings. The definition of the right balance between stability and plasticity is still an open question, and the extent to which accuracy on any past task may be sacrificed for higher average accuracy certainly depends on the application. Conceptual distinctions such as local and global algorithms can help continual learning practitioners in choosing the algorithm which is better suited to their problem.

iCarl. In Section 5 we explain that the buffer selection criteria of iCarl, based on a herding strategy, relies on the information of the task solution, which makes the loss approximation sensitive to big changes in the parameter space. In this section, we test our claims with a simple experiment. We change the buffer selection strategy of iCarl to *uniformly random sampling* from the task dataset, leaving the rest of the algorithm the same. We compare the two selection strategies across multiple learning rates, following the previous experiment design. In Figure 2 we plot the resulting forgetting as a function of the learning rate. We see that the two buffer selection strategies lead to visibly different outcomes when using high learning rates, confirming our thesis. More specifically, the herding strategy used by iCarl results in higher forgetting on average for non-local learning (shades of blue) compared to local-learning settings (shades of red). Instead using a random sampling strategy the forgetting distributions of local and non-local learning are hardly distinguishable. This experiment demonstrates the potential of the loss-approximation viewpoint for understanding continual learning algorithms, which is instrumental to address their failures.

6.2 Robustness of local approximations

Finally, we take a closer look at the geometry of the parameter space around the task optima. We wish to understand how in practice the ϵ-region of a task loss approximation behaves.

To this end, we evaluate the size of the area around a local minima where a second order approximation of the loss is accurate. Recall that around a local minima the second order approximation of the loss (Equation (5)) mostly depends on the hessian matrix, evaluated at the minima. The spectrum of this matrix is often dominated by a few, principal eigenvalues in practice. This means that, when close to the minima, moving along a principal eigenvector direction leads to the sharpest increase in the loss. However, far from the minima, where the second order approximation of the loss is no longer accurate, moving along the same direction will be no different than moving along any other direction, on average.

We observe the transition in the correctness of the second order approximation by looking at the loss function as we move away from the minima along the principal eigenvectors. We obtain the task local optima $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_T$ by training on each task sequentially with an SGD optimizer. We then evaluate the effect of perturbing the optima along a principal eigenvector through the following score:

$$\mathfrak{s}(r) = \mathbb{E}_{\boldsymbol{\mu}'} \left| \frac{L_t(\boldsymbol{\theta}_t + r \cdot \boldsymbol{v}_i) - L_t(\boldsymbol{\theta}_t)}{L_t(\boldsymbol{\theta}_t + r \cdot \boldsymbol{\mu}') - L_t(\boldsymbol{\theta}_t)} \right| \tag{17}$$

In Equation (17) $\boldsymbol{v}_i$ represents the i-th principal eigenvector, and $\boldsymbol{\mu}'$ is a Gaussian random vector scaled to have unit norm. The scalar r controls the distance from the optima. In Figure 3 we plot $\mathfrak{s}(r)$ and the loss $L_t(\boldsymbol{\theta}_t + r \cdot \boldsymbol{v}_i)$ as we vary r in the range $[10^{-3}, 10^6]$ for all 5 tasks of the Split CIFAR-10 challenge. We see that the score increases

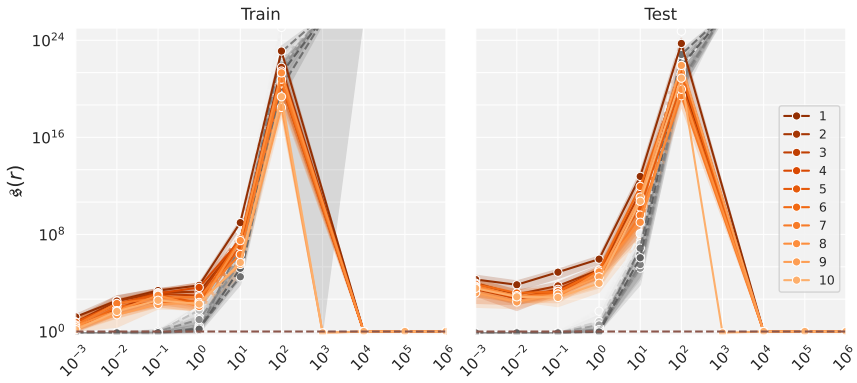

Figure 3: In orange, the perturbation score $\mathfrak{s}(r)$ (Equation (17)) and in gray, the task loss, evaluated on train and test data on the Split CIFAR 10 tasks. The shaded area around the curves reflects standard deviation across tasks. Different lines correspond to different perturbation directions (the first 10 eigenvectors of the the corresponding loss). We evaluate the curves for multiple values of r on a logarithmic scale in the range $[10^{-3}, 10^{6}]$. The shape of the curve is remarkably stable across tasks. Also, notice that the score $\mathfrak{s}(r)$ on the test data is large even for $r = 0$, which indicates that the test loss is not 0 at the local optima.

monotonically with the perturbation strength r for low values of r, and it drops to 1 after $\approx 10^3$. We can deduce that, for this specific case, the quadratic approximation is reasonably accurate in a large neighborhood of the tasks optima. Moreover, we observe that the shape of the curve is remarkably stable across tasks, which suggests that the different task optima have a similar quadratic geometry.

7 Related work

We are not the first to view the continual learning problem from the loss approximation perspective. Several existing algorithms (Zenke et al., 2017; Lee et al., 2016; Yoon et al., 2021) have been devised in this perspective. Moreover, Yin et al. (2020); Kong et al. (2023) have already studied regularization-based algorithms using local quadratic loss approximations. Differently from Yin et al. (2020); Kong et al. (2023), we analyze local polynomial approximations with the aim of deriving optimal continual learning objectives in local-learning setting, rather than analyzing existing algorithms. Similarly, our results on Orthogonal Gradient Descent are also different from those of Bennani et al. (2020), who establish rigorous guarantees for OGD under the Neural Tangent Kernel (Jacot et al., 2020) regime: our results regard instead the equivalence between OGD and the optimal objective under local quadratic approximations.

This work is also related to existing surveys of the continual learning literature, such as (De Lange et al., 2021; Parisi et al., 2019; Khetarpal et al., 2022; Qu et al., 2021; Awasthi & Sarawagi, 2019), which catalogue the existing continual learning into different 'families' and rank them according to different metrics. However, our classification of the literature into local and global algorithms reflects the implicit limitations of the algorithms rather than their surface-level characteristics, such as the presence of regularisation or rather the storing of input-output pairs in an external memory bank.

Finally, our work shares the spirit of existing theoretical studies of catastrophic forgetting and continual learning algorithms, such as (Mirzadeh et al., 2020b;a; Farquhar & Gal, 2019; Ramasesh et al., 2020; Verwimp et al., 2021). To the best of our knowledge, we are the first to focus on the effects of the locality assumption in continual learning algorithms.

8 Conclusion

In summary, in this work, we view continual learning from the point of view of the multi-task loss approximation and we study the differences between local and global approximations. Based on this analysis, we provide a classification of existing algorithms into local and global algorithms, and we evaluate the practical consequences of our theoretical distinction through extensive experiments. We believe our results are of interest to both empirical and theoretical research in continual learning. In general, we find that the loss approximation viewpoint has not been sufficiently developed in the literature, and with this work, we aim to demonstrate that it offers powerful abstractions of continual learning algorithms.

ACKNOWLEDGEMENTS

GL is supported by a fellowship from the ETH AI Center. SPS would like to acknowledge the financial support from Max Planck ETH Center for Learning Systems.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Abhijeet Awasthi and Sunita Sarawagi. Continual learning with neural networks: A review. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pp. 362–365, 2019.

Mehdi Abbana Bennani, Thang Doan, and Masashi Sugiyama. Generalisation guarantees for continual learning with orthogonal gradient descent. *arXiv preprint arXiv:2006.11942*, 2020.

Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15920–15930. Curran Associates, Inc., 2020.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2180–2187. IEEE, 2021.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pp. 192–204. PMLR, 2015.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Danruo Deng, Guangyong Chen, Jianye Hao, Qiong Wang, and Pheng-Ann Heng. Flattening sharpness for dynamic gradient projection memory benefits continual learning. *Advances in Neural Information Processing Systems*, 34: 18710–18721, 2021.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3762–3773. PMLR, 2020.

Sebastian Farquhar and Yarin Gal. A unifying bayesian view of continual learning. *arXiv preprint arXiv:1902.06494*, 2019.

Saurabh Garg, Mehrdad Farajtabar, Hadi Pouransari, Raviteja Vemulapalli, Sachin Mehta, Oncel Tuzel, Vaishaal Shankar, and Fartash Faghri. Tic-clip: Continual training of clip models. *arXiv preprint arXiv:2310.16226*, 2023.

Noah Golmant, Zhewei Yao, Amir Gholami, Michael Mahoney, and Joseph Gonzalez. pytorch-hessian-eigenthings: efficient pytorch hessian eigendecomposition, October 2018. URL https://github.com/noahgolmant/pytorch-hessian-eigenthings.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pp. 1832–1841. PMLR, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks, 2020.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

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.

Yajing Kong, Liu Liu, Huanhuan Chen, Janusz Kacprzyk, and Dacheng Tao. Overcoming catastrophic forgetting in continual learning by exploring eigenvalues of hessian matrix. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Theodora Kontogianni, Yuanwen Yue, Siyu Tang, and Konrad Schindler. Is continual learning ready for real-world challenges? *arXiv preprint arXiv:2402.10130*, 2024.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

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

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *arXiv preprint arXiv:2010.04495*, 2020a.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33:7308–7320, 2020b.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Haoxuan Qu, Hossein Rahmani, Li Xu, Bryan Williams, and Jun Liu. Recent advances of continual learning in computer vision: An overview. *arXiv preprint arXiv:2109.11369*, 2021.

Vinay V Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. *arXiv preprint arXiv:2007.07400*, 2020.

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*, pp. 2001–2010, 2017.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pp. 4548–4557. PMLR, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Sidak Pal Singh, Gregor Bachmann, and Thomas Hofmann. Analytic insights into structure and rank of neural network hessian maps. *Advances in Neural Information Processing Systems*, 34:23914–23927, 2021.

Samuel L Smith, Benoit Dherin, David GT Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. *arXiv preprint arXiv:2101.12176*, 2021.

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):4069, 2020.

Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*, 2018.

Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9385–9394, 2021.

Jiayu Wu, Qixiang Zhang, and Guoxi Xu. Tiny imagenet challenge. *Technical report*, 2017.

Peng Xu, Bryan He, Christopher De Sa, Ioannis Mitliagkas, and Chris Re. Accelerated stochastic power iteration. In *International Conference on Artificial Intelligence and Statistics*, pp. 58–67. PMLR, 2018.

Zhewei Yao, Amir Gholami, Daiyaan Arfeen, Richard Liaw, Joseph Gonzalez, Kurt Keutzer, and Michael Mahoney. Large batch size training of neural networks with adversarial training and second-order information. *arXiv preprint arXiv:1810.01021*, 2018.

Dong Yin, Mehrdad Farajtabar, Ang Li, Nir Levine, and Alex Mott. Optimization and generalization of regularization-based continual learning: a loss approximation viewpoint. *arXiv preprint arXiv:2006.10974*, 2020.

Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. *arXiv preprint arXiv:2106.01085*, 2021.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

APPENDIX

9 PROOFS

Notation

$\mathcal{T}_1, \dots, \mathcal{T}_T$	A sequence of tasks
$\mathcal{X}$	Input space
$\mathcal{Y}_t$	Task t output space (may vary or be shared across tasks)
D_t	The dataset of task $\mathcal{T}_t$: $\{(x_1, y_1), \dots, (x_{n_t}, y_{n_t})\}$
$\Theta \subseteq \mathbb{R}^P$	The neural network parameters
a	A scalar
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\boldsymbol{\theta}$	A generic network parameters vector
$\boldsymbol{\theta}_t$	The network parameters after training on task t
$\boldsymbol{\Delta}_t := \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}$	Parameter update due to the training on task t
$\boldsymbol{\theta}_0$	Network initialization
$l_t(x, y, \boldsymbol{\theta})$	Task t loss function
$\mathcal{L}_t(\boldsymbol{\theta})$	Expected loss on the task t distribution: $\langle l_t(x, y, \boldsymbol{\theta}) \rangle_{\mathcal{P}_t}$
$L_t(\boldsymbol{\theta})$	Empirical loss on the task t dataset: $l_t(x, y, \boldsymbol{\theta}) \rangle_{D_t}$
$L_t^\star$	$L_t(\theta_t)$
$\boldsymbol{\nabla} \mathcal{L}_t(\boldsymbol{\theta})$	Loss gradient vector: $\dfrac{\partial \mathcal{L}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$
$\mathbf{H}_t(\boldsymbol{\theta})$	Loss hessian matrix: $\dfrac{\partial^2 \mathcal{L}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2}$
$\mathbf{H}_t^\star$	$\mathbf{H}_t(\boldsymbol{\theta}_t)$
$\overline{\boldsymbol{\nabla} L}_{<t}$	Average gradient: $\frac{1}{t} \sum_{o=1}^{t-1} \boldsymbol{\nabla} L_o(\boldsymbol{\theta}_o)$
$\overline{\mathbf{H}}_{<t}^\star$	Average hessian: $\frac{1}{t} \sum_{o=1}^{t-1} \mathbf{H}_o^\star$
$\mathcal{E}_o(\boldsymbol{\theta})$	Forgetting: $\mathcal{L}_o(\boldsymbol{\theta}) - \mathcal{L}_o(\boldsymbol{\theta}_o)$
$E_o(\boldsymbol{\theta})$	$L_o(\boldsymbol{\theta}) - L_o(\boldsymbol{\theta}_o)$
$\mathcal{E}_o^{acc}(\boldsymbol{\theta})$	$\mathrm{ACC}_o(\boldsymbol{\theta}_o) - \mathrm{ACC}_o(\boldsymbol{\theta})$
$E(t)$	Average forgetting: $\frac{1}{t} \sum_{o=1}^{t} \mathcal{E}_o(\boldsymbol{\theta}_t)$
$\mathrm{ACC}(t)$	Average accuracy: $\frac{1}{t} \sum_{o=1}^{t} \mathrm{ACC}_o(\boldsymbol{\theta}_t)$
$\mathcal{L}_t^{MT}(\boldsymbol{\theta})$	Expected multi-task loss: $\frac{1}{t} \left(\mathcal{L}_1(\boldsymbol{\theta}) + \dots + \mathcal{L}_t(\boldsymbol{\theta})\right)$
$L_t^{MT}(\boldsymbol{\theta})$	Empirical multi-task loss: $\frac{1}{t} \left(L_1(\boldsymbol{\theta}) + \dots + L_t(\boldsymbol{\theta})\right)$
$\hat{L}_t^{MT}(\boldsymbol{\theta})$	Approximate multi-task loss: $\frac{1}{t} \left(\hat{L}_1(\boldsymbol{\theta}) + \dots + \hat{L}_{t-1}(\boldsymbol{\theta}) + L_t(\boldsymbol{\theta})\right)$

9.1 QUADRATIC LOCAL LOSS APPROXIMATIONS (SECTION 4)

9.1.1 EQUATION (7)

In the following, we show the steps leading to Equation 7.

We start from the formula of forgetting given by a quadratic Taylor expansion of the task loss around $\boldsymbol{\theta}_t$ (Equation 6):

$$E_o(\boldsymbol{\theta}_t) = (\boldsymbol{\theta}_t - \boldsymbol{\theta}_o)^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{2}(\boldsymbol{\theta}_t - \boldsymbol{\theta}_o)^\intercal \mathbf{H}_o^\star (\boldsymbol{\theta}_t - \boldsymbol{\theta}_o)$$

$$\begin{aligned}
E_o(t) &= (\boldsymbol{\theta}_t - \boldsymbol{\theta}_o)^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{2}(\boldsymbol{\theta}_t - \boldsymbol{\theta}_o)^\intercal \mathbf{H}_o^\star (\boldsymbol{\theta}_t - \boldsymbol{\theta}_o) \\
&= \left(\sum_{\tau=o+1}^{t} \boldsymbol{\Delta}_\tau\right)^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{2}\left(\sum_{\tau=o+1}^{t} \boldsymbol{\Delta}_\tau\right)^\intercal \mathbf{H}_o^\star \left(\sum_{\tau=o+1}^{t} \boldsymbol{\Delta}_\tau\right) \\
&= \underbrace{\left(\sum_{\tau=o+1}^{t-1} \boldsymbol{\Delta}_\tau\right)^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{2}\left(\sum_{\tau=o+1}^{t-1} \boldsymbol{\Delta}_\tau\right)^\intercal \mathbf{H}_o^\star \left(\sum_{\tau=o+1}^{t-1} \boldsymbol{\Delta}_\tau\right)}_{E_o(t-1)} + \boldsymbol{\Delta}_t^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{2}\boldsymbol{\Delta}_t^\intercal \mathbf{H}_o^\star \boldsymbol{\Delta}_t + \\
&\quad + \frac{1}{2}\left(\sum_{\tau=o+1}^{t-1} \boldsymbol{\Delta}_\tau\right)^\intercal \mathbf{H}_o^\star \boldsymbol{\Delta}_t + \frac{1}{2}\boldsymbol{\Delta}_t^\intercal \mathbf{H}_o^\star \left(\sum_{\tau=o+1}^{t-1} \boldsymbol{\Delta}_\tau\right) \\
&= E_o(t-1) + \boldsymbol{\Delta}_t^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{2}\boldsymbol{\Delta}_t^\intercal \mathbf{H}_o^\star \boldsymbol{\Delta}_t + \left(\sum_{\tau=o+1}^{t-1} \boldsymbol{\Delta}_\tau\right)^\intercal \mathbf{H}_o^\star \boldsymbol{\Delta}_t
\end{aligned}$$

Based on this formulation, we characterize the average forgetting as follows:

$$\begin{aligned}
E(t) &= \frac{1}{t}\sum_{o=1}^{t} E_o(t) = \frac{1}{t}\underbrace{E_t(t)}_{=0} + \frac{1}{t}\sum_{o=1}^{t-1} E_o(t) \\
&= \frac{1}{t}\sum_{o=1}^{t-1}\left[E_o(t-1) + \boldsymbol{\Delta}_t^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{2}\boldsymbol{\Delta}_t^\intercal \mathbf{H}_o^\star \boldsymbol{\Delta}_t + \left(\sum_{\tau=o+1}^{t-1} \boldsymbol{\Delta}_\tau\right)^\intercal \mathbf{H}_o^\star \boldsymbol{\Delta}_t\right] \\
&= \frac{1}{t}\sum_{o=1}^{t-1}[E_o(t-1)] + \frac{1}{t}\boldsymbol{\Delta}_t^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{t}\sum_{o=1}^{t-1}\left[\frac{1}{2}\boldsymbol{\Delta}_t^\intercal \mathbf{H}_o^\star \boldsymbol{\Delta}_t\right] + \frac{1}{t}\sum_{o=1}^{t-1}\left[\left(\sum_{\tau=o+1}^{t-1} \boldsymbol{\Delta}_\tau\right)^\intercal \mathbf{H}_o^\star \boldsymbol{\Delta}_t\right] \\
&= \frac{t-1}{t}\cdot E(t-1) + \frac{1}{t}\boldsymbol{\Delta}_t^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{2t}\boldsymbol{\Delta}_t^\intercal \left[\sum_{o=1}^{t-1} \mathbf{H}_o^\star\right]\boldsymbol{\Delta}_t + \frac{1}{t}\left[\underbrace{\sum_{o=1}^{t-1}(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_o)^\intercal \mathbf{H}_o^\star}_{\boldsymbol{v}^\intercal}\right]\boldsymbol{\Delta}_t \\
&= \frac{1}{t}\left((t-1)\cdot E(t-1) + \boldsymbol{\Delta}_t^\intercal \boldsymbol{\nabla}\mathcal{L}_o(\boldsymbol{\theta}_o) + \frac{1}{2}\boldsymbol{\Delta}_t^\intercal (\sum_{o=1}^{t-1} \mathbf{H}_o^\star)\boldsymbol{\Delta}_t + \boldsymbol{v}^\intercal \boldsymbol{\Delta}_t\right)
\end{aligned}$$

9.1.2 THEOREM 4.1 (OPTIMAL QUADRATIC LOCAL CONTINUAL LEARNING)

The first part of our proof is by induction. We prove the base case for $t = 2$ and $t = 3$, since some terms trivially cancel out for $t = 1$.

The theorem makes the following assumptions: (1) every task solution $\boldsymbol{\theta}_i$ is a local minima of L_i and (2) $\sup_{\boldsymbol{\theta}_i, \boldsymbol{\theta}_k} \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_k\|^3 < \epsilon$. From the second assumption we deduce that the error of a local quadratic approximation of L_i is bounded by an arbitrary ϵ and may therefore be safely ignored. The first assumption instead lets us state $\boldsymbol{\nabla} L_i(\boldsymbol{\theta}_i) = 0$ and the Hessian $\mathbf{H}_i^\star \succcurlyeq 0$ is p.s.d.

Base case 1: $E(1) = 0 \implies E(2) = \frac{1}{2}\boldsymbol{\Delta}_2^\intercal \boldsymbol{H}_1^\star \boldsymbol{\Delta}_2$. Notice that by definition, $E(1) = \mathcal{E}_1(1) = 0$.

Using Equation 7 we can write $E(2)$:

$$\begin{aligned}
E(2) &= E(1) + \frac{1}{2}\boldsymbol{\Delta}_2^\intercal (\boldsymbol{H}_1^\star)\boldsymbol{\Delta}_2 + (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_1)^\intercal \boldsymbol{H}_1^\star \boldsymbol{\Delta}_2 \\
&= 0 + \frac{1}{2}\boldsymbol{\Delta}_2^\intercal \boldsymbol{H}_1^\star \boldsymbol{\Delta}_2 + 0
\end{aligned}$$

Base case 2: $E(1) = 0, E(2) = 0 \implies E(3) = \frac{1}{2}\boldsymbol{\Delta}_3^\intercal(\frac{1}{3}\boldsymbol{H}_1^\star + \frac{1}{3}\boldsymbol{H}_2^\star)\boldsymbol{\Delta}_3$.
Using the last result from case 1, we have that :

$$E(2) = \frac{1}{2}\boldsymbol{\Delta}_2^\intercal \boldsymbol{H}_1^\star \boldsymbol{\Delta}_2 = 0$$

The latter equation implies that $\boldsymbol{\Delta}_2^\intercal \boldsymbol{H}_1^1 = \mathbf{0}$. Plugging it into the value of $E(3)$ given by Equation 7:

$$\begin{aligned} E(3) &= \frac{1}{3}\left(2 \cdot E(2) + \frac{1}{2}\boldsymbol{\Delta}_3^\intercal(\boldsymbol{H}_2^\star + \boldsymbol{H}_1^\star)\boldsymbol{\Delta}_3 + \sum_{t=1}^{2}(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_t)^\intercal \boldsymbol{H}_t^\star \boldsymbol{\Delta}_3\right) \\ &= 0 + \frac{1}{2}\boldsymbol{\Delta}_3^\intercal(\frac{1}{3}\boldsymbol{H}_2^\star + \frac{1}{3}\boldsymbol{H}_1^\star)\boldsymbol{\Delta}_3 + \frac{1}{3}\underbrace{\boldsymbol{\Delta}_2^\intercal \boldsymbol{H}_1^1}_{=\mathbf{0}}\boldsymbol{\Delta}_3 + \frac{1}{3}\underbrace{(\boldsymbol{\theta}_2 - \boldsymbol{\theta}_2)^\intercal}_{=\mathbf{0}}\boldsymbol{H}_t^\star \boldsymbol{\Delta}_3 \end{aligned}$$

Induction step: *if* $E(\tau) = 0\ \forall \tau < t$ *then* $E(t) \geq 0$.
We start by writing out $E(t)$.

$$\begin{aligned} E(t) &= \frac{1}{t}\left((t-1)(t-1) + \frac{1}{2}\boldsymbol{\Delta}_t^\intercal(\sum_{\tau=1}^{t-1}\boldsymbol{H}_\tau^\star)\boldsymbol{\Delta}_t + \sum_{\tau=1}^{t-1}(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_\tau)^\intercal \boldsymbol{H}_\tau^\star \boldsymbol{\Delta}_t\right) \\ &= 0 + \frac{1}{2}\boldsymbol{\Delta}_t^\intercal(\sum_{\tau=1}^{t-1}\boldsymbol{H}_\tau^\star)\boldsymbol{\Delta}_t + \sum_{\tau=1}^{t-1}(\boldsymbol{\Delta}_{\tau+1} + \cdots + \boldsymbol{\Delta}_{t-1})^\intercal \boldsymbol{H}_\tau^\star \boldsymbol{\Delta}_t \end{aligned}$$

The induction step is accomplished if $\sum_{\tau=1}^{t-1}(\boldsymbol{\Delta}_{\tau+1} + \cdots + \boldsymbol{\Delta}_{t-1})^\intercal \boldsymbol{H}_\tau^\star = \sum_{\tau=1}^{t-2}(\boldsymbol{\Delta}_{\tau+1} + \cdots + \boldsymbol{\Delta}_{t-1})^\intercal \boldsymbol{H}_\tau^\star = \mathbf{0}$.
Consider now the case in which $\sum_{\tau=1}^{t-1}(\boldsymbol{\Delta}_{\tau+1} + \cdots + \boldsymbol{\Delta}_{t-1})^\intercal \boldsymbol{H}_\tau^\star \neq \mathbf{0}$. It follows that:

$$\begin{aligned} &\sum_{\tau=1}^{t-2}(\boldsymbol{\Delta}_{\tau+1} + \cdots + \boldsymbol{\Delta}_{t-1})^\intercal \boldsymbol{H}_\tau^\star \neq \mathbf{0} \\ &\sum_{\tau=1}^{t-2}\boldsymbol{\Delta}_{t-1}^\intercal \boldsymbol{H}_\tau^\star + \sum_{\tau=1}^{t-2}(\boldsymbol{\Delta}_{\tau+1} + \cdots + \boldsymbol{\Delta}_{t-2})^\intercal \boldsymbol{H}_\tau^\star \neq \mathbf{0} \end{aligned}$$

Multilying by $\boldsymbol{\Delta}_{t-1}$ on the right we get:

$$\sum_{\tau=1}^{t-2}\boldsymbol{\Delta}_{t-1}^\intercal \boldsymbol{H}_\tau^\tau \boldsymbol{\Delta}_{t-1} + \sum_{\tau=1}^{t-3}(\boldsymbol{\Delta}_{\tau+1} + \cdots + \boldsymbol{\Delta}_{t-2})^\intercal \boldsymbol{H}_\tau^\tau \boldsymbol{\Delta}_{t-1} \neq \mathbf{0}^\intercal \boldsymbol{\Delta}_{t-1}$$

We compare this result with the value of $\mathcal{E}(t-1)$ given by Equation 7:

$$\begin{aligned} E(t-1) =& \frac{1}{t-1}\left((t-2) \cdot \mathcal{E}(t-2) + \frac{1}{2}\boldsymbol{\Delta}_{t-1}^\intercal(\sum_{o=1}^{t-2}\mathbf{H}_o^\star)\boldsymbol{\Delta}_{t-1} + \sum_{\tau=1}^{t-2}(\boldsymbol{\Delta}_{\tau+1} + \cdots + \boldsymbol{\Delta}_{t-1})^\intercal \boldsymbol{H}_\tau^\star \boldsymbol{\Delta}_{t-1}\right) \\ =& \frac{1}{t-1}\left(0 + \frac{1}{2}\boldsymbol{\Delta}_{t-1}^\intercal(\sum_{\tau=1}^{t-2}\boldsymbol{H}_\tau^\star)\boldsymbol{\Delta}_{t-1} + \sum_{\tau=1}^{t-2}(\boldsymbol{\Delta}_{\tau+1} + \cdots + \boldsymbol{\Delta}_{t-1})^\intercal \boldsymbol{H}_\tau^\star \boldsymbol{\Delta}_{t-1}\right) \neq 0 \end{aligned}$$

We arrived at a contradiction, which proves the induction step and concludes that if $E(1), \ldots, E(t-1) = 0$ then

$$E(t) = \frac{1}{2}\boldsymbol{\Delta}_t^\intercal\left(\frac{1}{t} \cdot \sum_{o=1}^{t-1}\mathbf{H}_o^\star\right)\boldsymbol{\Delta}_t \geq 0 \tag{18}$$

The second part of the proof follows from the simple observation that minimizing the multi-task loss L_t^{MT} with respect to $\boldsymbol{\Delta}_t$ is equivalent to minimizing $L_t + E(t)$ with respect to $\boldsymbol{\Delta}_t$. Morover, notice that, by definition, $E(1) = 0$ for any continual learning algorithm, from which it follows by Equation (18) that $E(2) \geq 0$. Thus, if there exist a

parameter update such that $L_2 + E(2)$ is minimized, then $E(2) = 0$ and therefore the objective can be written as $\min_{\boldsymbol{\Delta}_2} L_2 \; s.t. \; E(2) = 0$. Recursively applying this argument it holds:

$$E(2) = \boldsymbol{\Delta}_2^\intercal (\sum_{o=1}^{1} \mathbf{H}_o^\star) \boldsymbol{\Delta}_2 = 0$$

$$\dots$$

$$E(t-1) = \boldsymbol{\Delta}_{t-1}^\intercal (\sum_{o=1}^{T-1} \mathbf{H}_o^\star) \boldsymbol{\Delta}_{t-1} = 0$$

Then the optimal objective for the t task can be written as Equation (11).

9.2 Local and Global algorithms in the literature (Section 5)

9.2.1 Theorem 5.1 (OGD implements the local quadratic optimal objective)

We start by recalling the OGD algorithm. Let $D_o = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_{n_o}, y_{n_o}))\}$ be the dataset associated with task $\mathcal{T}_o$ and $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^K$ be the network output corresponding to the input $\boldsymbol{x}$. The gradient of the network output with respect to the network parameters is the $P \times K$ matrix:

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \left[\nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}^1(\boldsymbol{x}), \cdots, \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}^K(\boldsymbol{x})\right]$$

The standard version of OGD imposes the following constraint on any parameter update $\boldsymbol{u}$:

$$\langle \boldsymbol{u}, \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}_o}^k(\boldsymbol{x}_i) \rangle = 0 \tag{19}$$

for all $k \in [1, K]$, $\boldsymbol{x}_i \in D_o$ and $o \leq t$, t being the number of tasks solved so far. If SGD is used, for example, the update $\boldsymbol{u}$ is the gradient of the new task loss for a data batch. Notice that the old task gradients are evaluated at the minima $\boldsymbol{\theta}_o$.

In the first part of the proof we want to show that the OGD constraint (Equation (19)) is equivalent to the constraint in Equation (11), namely $\boldsymbol{\Delta}_t^\intercal \left(\frac{1}{t} \cdot \sum_{o=1}^{t-1} \mathbf{H}_o^\star \right) \boldsymbol{\Delta}_t = 0$.

In Theorem 5.1 we make the assumption that the task solutions $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_t$ are local minima of the respective losses, from which it follows that $\boldsymbol{\nabla} L_t(\boldsymbol{\theta}_t) = 0$ and the Hessian $\mathbf{H}_t^\star \succcurlyeq 0$.

Recall the definition of the average loss:

$$EL_t(\boldsymbol{\theta}) = \frac{1}{n_t} \sum_{i=0}^{n_t} l_t(\boldsymbol{x}_{i,t}, y_{i,t}, \boldsymbol{\theta})$$

The Hessian matrix of the loss $\boldsymbol{H}_t(\boldsymbol{\theta})$ can be decomposed as a sum of two other matrices (Schraudolph, 2002): the *outer-product* Hessian and the *functional* Hessian.

$$\boldsymbol{H}_t(\boldsymbol{\theta}) = \frac{1}{n_t} \sum_{i=1}^{n_t} \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \left[\nabla_{\boldsymbol{f}}^2 \ell_i\right] \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\intercal + \frac{1}{n_t} \sum_{i=1}^{n_t} \sum_{k=1}^{K} \left[\nabla_{\boldsymbol{f}} \ell_i\right]_k \nabla_{\boldsymbol{\theta}}^2 \boldsymbol{f}_{\boldsymbol{\theta}}^k(\boldsymbol{x}_i) \; , \tag{20}$$

where $\ell_i = l(\boldsymbol{x_i}, y_i)$ and $\nabla_{\boldsymbol{f}}^2 \ell_i$ is the $K \times K$ matrix of second order derivatives of the loss ℓ_i with respect to the network output $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x_i})$. At the local minimum $\boldsymbol{\theta}_t$ the contribution of the functional Hessian is negligible (Singh et al., 2021). We rewrite the first goal of the proof using these two facts:

$$\boldsymbol{\Delta}_t^\intercal \mathbf{H}_o^\star \boldsymbol{\Delta}_t = 0$$

$$\boldsymbol{\Delta}_t^\intercal \left(\frac{1}{n_o} \sum_{i=1}^{n_o} \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \left[\nabla_{\boldsymbol{f}}^2 \ell_i\right] \nabla_{\boldsymbol{\theta}} \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\intercal \right) \boldsymbol{\Delta}_t = 0 \tag{21}$$

Farajtabar et al. (2020) apply the OGD constraint (Equation (19)) to the batch gradient vector $\boldsymbol{g}_B = \nabla \mathcal{L}_t^B$. Following this choice $\boldsymbol{\Delta}_t = \sum_{s=1}^{S_t} -\eta \boldsymbol{g}_s$, where η is the learning rate. Clearly, if, for all s, $\boldsymbol{g}_s$ satisfies Equation (19), then $\boldsymbol{\Delta}_t$

satisfies it. Hereafter, we ignore the specific form of $\boldsymbol{\Delta}_t$, proving the result for a broader class of algorithms for which $\boldsymbol{\Delta}_t$ satisfies the OGD constraint. Continuing from Equation 21:

$$\frac{1}{n_o}\left(\sum_{i=1}^{n_o}\nabla_{\boldsymbol{\theta}}\boldsymbol{\Delta}_t^\intercal \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\left[\nabla^2_{\boldsymbol{f}}\ell_i\right]\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)^\intercal\boldsymbol{\Delta}_t\right)$$

$$\frac{1}{n_o}\left(\sum_{i=1}^{n_o}\nabla_{\boldsymbol{\theta}}\left[\boldsymbol{\Delta}_t^\intercal\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{f}^1_{\boldsymbol{\theta}}(\boldsymbol{x}_i),\cdots,\nabla_{\boldsymbol{\theta}}\boldsymbol{f}^K_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right]\right]\left[\nabla^2_{\boldsymbol{f}}\ell_i\right]\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)^\intercal\boldsymbol{\Delta}_t\right)$$

$$\frac{1}{n_o}\left(\sum_{i=1}^{n_o}\nabla_{\boldsymbol{\theta}}\left[\underbrace{\boldsymbol{\Delta}_t^\intercal\nabla_{\boldsymbol{\theta}}\boldsymbol{f}^1_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}_{=0},\cdots,\underbrace{\boldsymbol{\Delta}_t^\intercal\nabla_{\boldsymbol{\theta}}\boldsymbol{f}^K_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}_{=0}\right]\left[\nabla^2_{\boldsymbol{f}}\ell_i\right]\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)^\intercal\boldsymbol{\Delta}_t\right)=0,$$

where $\boldsymbol{\Delta}_t^\intercal\nabla_{\boldsymbol{\theta}}\boldsymbol{f}^k_{\boldsymbol{\theta}}(\boldsymbol{x}_i)=0$ is the OGD constraint, which holds for any k, i and $o<t$. This concludes the first part of the proof.

In order to conclude the proof we simply apply Theorem 4.1 to OGD, and see that, if the algorithm objective is minimized at each learning step, then necessarily it follows that $E(t)=0\ \forall\, t\in[T]$.

9.2.2 On the validity of OGD-GTL (Additional result)

Instead of considering all the function gradients with respect to all the outputs $\{\nabla_{\boldsymbol{\theta}}\boldsymbol{f}^k_{\boldsymbol{\theta}}(\boldsymbol{x})\,|\,k\in[1,K],\boldsymbol{x}\in D_o\}$, Farajtabar et al. (2020) also consider a cheaper approximation where they impose orthogonality only with respect to the index corresponding to the true ground truth label (GTL). We show this can be understood via fairly mild assumptions, if the loss function is cross-entropy.

For a cross-entropy loss, $\boldsymbol{f}^k_{\boldsymbol{\theta}}(\boldsymbol{x_i})$ is the log-probability (or logit) associated with class k for input $\boldsymbol{x_i}$. The probability $p(y_i=k|\boldsymbol{x_i};\boldsymbol{\theta})$ is then defined as $(\boldsymbol{p_i})_j=\mathrm{softmax}(\boldsymbol{f_\theta}(\boldsymbol{x_i}))_k$. Recall, from the proof of Theorem 5.1, that the theorem statement can be equivalently written as:

$$\boldsymbol{\Delta}_t^\intercal\left(\frac{1}{n_o}\sum_{i=1}^{n_o}\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\left[\nabla^2_{\boldsymbol{f}}\ell_i\right]\nabla_{\boldsymbol{\theta}}\boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)^\intercal\right)\boldsymbol{\Delta}_t=0$$

For a cross-entropy loss the Hessian of the loss with respect to the network output is given by:

$$\nabla^2_{\boldsymbol{f}}\ell_i=\mathrm{diag}(\boldsymbol{p_i})-\boldsymbol{p_i}\boldsymbol{p_i}^\intercal$$

Without loss of generality, assume index $k=1$ corresponds to the GTL. Towards the end of the usual training, the softmax output at index 1, i.e., $p_1\approx 1$. Let us assume that the probabilities for the remaining output coordinates is equally split between them. More precisely, let

$$p_2,\cdots,p_K=\frac{1-p_1}{K-1}.$$

Hence, in terms of their scales, we can regard $p_1=\mathcal{O}(1)$ while $p_2,\cdots,p_K=\mathcal{O}\left(K^{-1}\right)$. Furthermore, $(1-p_i)=\mathcal{O}(1)\quad\forall\, i\in[2\cdots K]$. Then, a simple computation of the inner matrix in the Hessian outer product, i.e., $\nabla^2_{\boldsymbol{f}}\ell=\mathrm{diag}(\boldsymbol{p})-\boldsymbol{p}\boldsymbol{p}^\top$, will reveal that *diagonal entry corresponding to the ground truth label is $\mathcal{O}(1)$, while rest of the entries in the matrix are of the order $\mathcal{O}\left(K^{-1}\right)$ or $\mathcal{O}\left(K^{-2}\right)$* — thereby explaining this approximation. Also, we can see that this approximation would work well only when $K\gg 1$ and does not carry over to other loss functions, e.g. Mean Squared Error (MSE). In fact, $K=2$ all the entries of this matrix are of the same magnitude $|p_1(1-p_1)|$, and for MSE simply $\nabla^2_{\boldsymbol{f}}\ell_i=\boldsymbol{I}_K$.

10 Details on the experimental setup

10.1 Experiments configuration

We follow the same experiment configurations as (Buzzega et al., 2020). In Table 2 we summarise the configurations used in the main experiment (Section 6.1).

Table 2: **Overview of the experiments configurations.**

DATA	CHALLENGE TYPE	TASKS	ARCHITECTURE (P)	ALGORITHMS
ROTATED-MNIST	DOMAIN-IL	20	MLP (89K)	A-GEM, ER, SI, EWC, OGD
SPLIT CIFAR-10	TASK-IL, CLASS-IL	5	RESNET18 (11M)	A-GEM, ER, SI, ICARL, EWC, OGD
SPLIT TINY-IMAGENET	TASK-IL, CLASS-IL	20	RESNET18 (11M)	A-GEM, ER, SI, ICARL, EWC, OGD

10.2 TRAINING SETTING AND HYPERPARAMETERS

Hyperparameters In our experiments we use the optimal hyperparameters provided by Buzzega et al. (2020). They select the hyperparameters for each experiment configuration by performing a grid-search on a validation set, the latter obtained by sampling 10% of the training set. In Table 3 we report all the value of the optimal hyperparameters, which were used in our experiments. We mark by an asterisk the hyper-parameter which we modified in our experiment configuration compared to (Buzzega et al., 2020). In particular, we increased the number of epochs from 1 to 5 in the Rotated MNIST challenge in order to train effectively with very low learning rates.

Table 3: **Overview of hyperparameters.**

DATA	ALGORITHM	BUFFER SIZE	BATCH SIZE	EPOCHS	OTHER HP
ROTATED-MNIST	A-GEM	500	128	5*	-
	ER	500	128	5*	-
	SI	500	128	5*	$c = 1, \xi = 1$
	EWC	-	128	5*	$\lambda = 0.7, \gamma = 1.0$
	OGD	500	128	5*	'GTL' VARIANT
SPLIT CIFAR-10	A-GEM	200	32	50	-
	ER	200	32	50	-
	SI	200	32	50	$c = 0.5, \xi = 1$
	EWC	-	32	50	$\lambda = 10, \gamma = 1.0$
	OGD	200	32	50	'GTL' VARIANT
	ICARL	200	32	50	WEIGHT DECAY $10e^{-6}$
SPLIT TINYIMAGENET	A-GEM	500	32	100	-
	ER	500	32	100	-
	SI	500	32	100	$c = 0.5, \xi = 1$
	EWC	-	32	100	$\lambda = 25, \gamma = 1.0$
	OGD	200*	32	100	'GTL' VARIANT
	ICARL	500	32	100	WEIGHT DECAY $10e^{-6}$

We repeat all our experiments for 5 different seeds, namely $11, 13, 33, 21, 55$, and 5 different learning rates $[0.0001, 0.001, 0.01, 0.05, 0.1]$. For Rotated-MNIST 0.0001 is too small to produce any meaningful results and we add 0.005 to the list instead. Finally, following (Buzzega et al., 2020) we apply we apply random crops and horizontal flips to both stream and buffer examples for CIFAR-10 and Tiny ImageNet.

The same hyperparameters used for iCarl are carried over to the variant of iCarl using random sampling which we discuss in Section 6.1.

Training settings. All networks are trained with Stochastic Gradient Descent (SGD), with a constant learning rate (as in (Buzzega et al., 2020)). Unless otherwise stated, we do not use weight decay or momentum.

10.3 MAIN EXPERIMENT (SECTION 6.1): METRICS

In Table 1 and Figure 2 we report average forgetting after learning the last task in the sequence. This is measured as the signed difference in test accuracy, averaged over tasks:

$$E^{acc}(T) = \frac{1}{T}\sum_{o=1}^{T} \mathrm{ACC}_o(\boldsymbol{\theta}_o) - \mathrm{ACC}_o(\boldsymbol{\theta}_T)$$

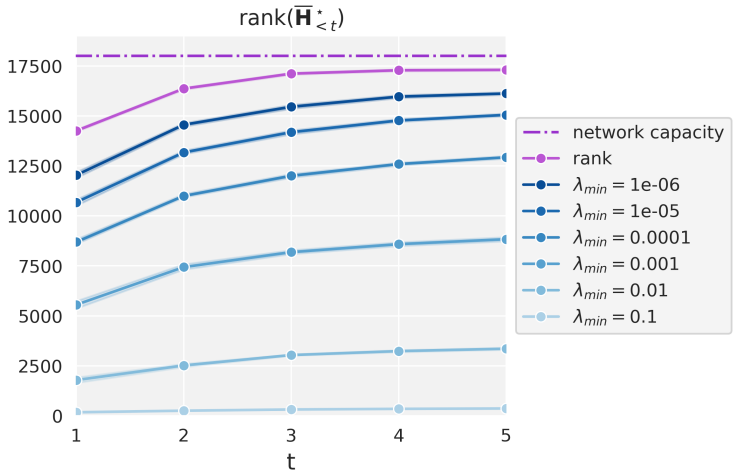

Figure 4: Rank and effective rank for various threshold λ values on a tiny Rotated MNIST challenge. The values are averaged over 5 seeds.

Moreover, we report the average accuracy, over all tasks after learning the last task in the sequence:

$$ACC(T) = \frac{1}{T}\sum_{o=1}^{T} \text{ACC}_o(\boldsymbol{\theta}_T)$$

10.4 Robustness of local approximations (Section 6.2): Hessian eigenvectors computation

In order to compute the perturbation score (Section 6.2) we have to obtain the Hessian matrix first 10 eigenvectors. We do so by using the *Lanczos method*, a cheap iterative method, and the Hessian-vector product (Yao et al., 2018; Xu et al., 2018). Golmant et al. (2018) has publicly provided an implementation of these methods for Pytorch neural networks, which we have adapted to our scope. We randomly sample 2000 inputs from the dataset to compute the Hessian and/or its eigenvectors.

11 Additional results

11.1 Average Hessian rank

In Section 4 we argue that the Hessian matrix has a central role in local approximations of the task loss. In particular, the rank of the average task hessian $\overline{\mathbf{H}}^{\star}_{<t} = \frac{1}{t}\sum_{o=1}^{t-1}\mathbf{H}^{\star}_{o}$ indicates the size of the parameter subspace where forgetting is 0. This quantity is especially important for algorithms, such as Orthogonal Gradient Descent, which effectively constrain the parameter update to the null-space of $\overline{\mathbf{H}}^{\star}_{<t}$ (Theorem 5.1 Theorem 4.1).

In Figure 4 we plot the evolution of $\text{rank}(\overline{\mathbf{H}}^{\star}_{<t})$ over t for a tiny version of the Rotated-MNIST challenge (with only 5 tasks), for which we use a toy MLP network of 18K parameters. We have to reduce the size of the network to compute the full Hessian matrix. We use simply SGD to learn the tasks in a sequential fashion. Additionally, we evaluate the effective rank of the average Hessian for multiple threshold λ values, which better captures the effective dimensionality of the matrix column space.

11.2 Perturbation score on Rotated-MNIST

We repeat the experiment in Section 6.2 for the Rotated-MNIST challenge with the standard network configuration. Similarly to the Split CIFAR-10 setting, we train the network with SGD on the 20 tasks sequentially. We evaluate the perturbation score for the learning rate which achieves the lowest task loss for all tasks, such that the gradients are approximately 0.

We observe two main differences with the Split CIFAR-10 results. First, the curve does not converge to 1 as we increase the radius, which indicates that the loss landscape of the MLP network, compared to the ResNet18 is more close to convex around the tasks local minima. Second, the absolute magnitude of the perturbation score is significantly lower

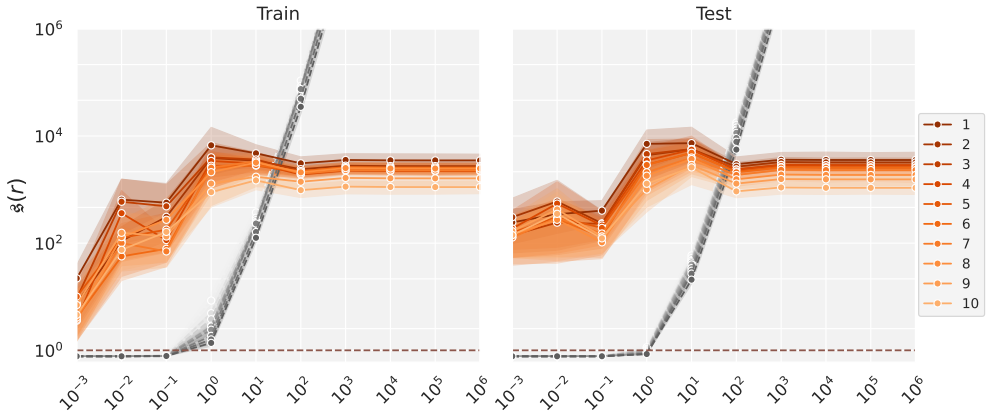

Figure 5: In orange, the perturbation score $\mathfrak{s}(r)$ (Equation (17)) and in gray, the task loss, evaluated on train and test data on the Rotated-MNIST 20 tasks. The shaded area around the curves reflects standard deviation across tasks. Different lines correspond to different perturbation directions (the first 10 eigenvectors of the the corresponding loss). We evaluate the curves for multiple values of r on a logarithmic scale in the range $[10^{-3}, 10^6]$. The shape of the curve is remarkably stable across tasks.

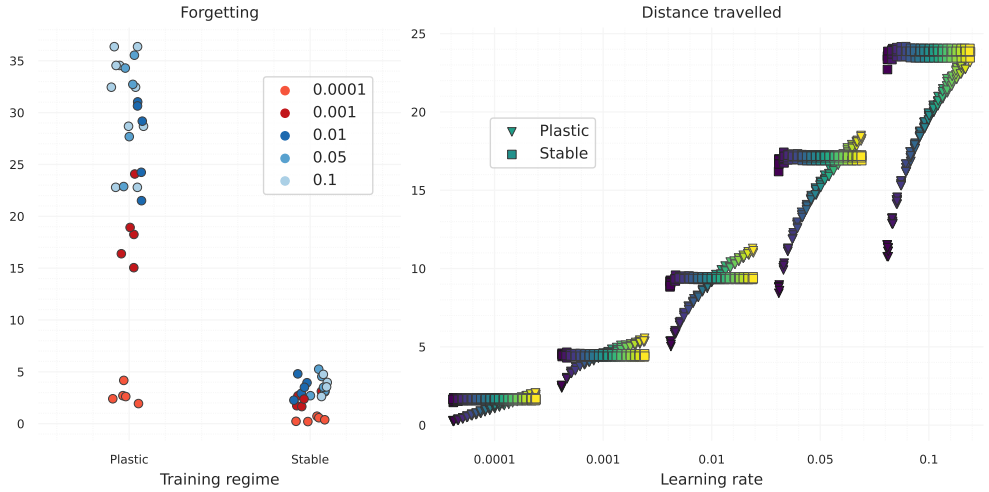

Figure 6: Comparison of the *plastic* and *stable* versions of SGD in terms of average forgetting and distance travelled in the parameter space.

on the Rotated-MNIST benchmark. We believe this is an effect of the difference in network size (80K parameters for Rotated-MNIST and 11M parameters for Split CIFAR-10). Similarly to the Split-CIFAR10 results, we see a remarkable similarity in the curves shape between tasks, which again suggests that the loss landscape geometry across local minima is similar across tasks.

11.3 Evaluation of Stable SGD

Finally, we use our experimental setting to compare SGD under the *Platic* and *Stable* training regimes, studied by Mirzadeh et al. (2020b). In the latter case the learning hyperparameters are adjusted to achieve the best tradeoff between plasticity and stability, according to their analysis of the geometry of the task local minima. In particular, stable SGD uses dropout, large initial learning rate and small batch size in order to converge to flatter minima, and limit the distance travelled in the parameter space. We repeat their experiment on Rotated-MNIST, copying their exact hyperparameter setup (except for the number of learning rate and number of epochs, which we increase to 5). In Figure 6 we compare the *Plastic* and *Stable* in terms of forgetting and distance traveled in the parameter space for

multiple learning rates. We observe a lower forgetting in the stable regime across all learning rates, which suggests that the beneficial properties of the stable configuration are robust to the specific learning rate value. Additionally, we observe that the trajectory in the parameter space converges quickly to a fixed region, limiting the movement from one task to the next. Overall, our observations confirm the results of Mirzadeh et al. (2020b).

We may argue that SGD is a global algorithm, as its objective coincides with the the current task loss $L_t(\boldsymbol{\theta})$, thus implicitly using the constant approximation $\hat{L}_i(\boldsymbol{\theta}) = 0$ for any $i < t$. However, it displays the behaviour of a local algorithm (Figure 6), with higher learning rates yielding higher forgetting in both the stable and plastic regimes. We speculate that this effect might be due to the implicit regularisation of gradient descent (Smith et al., 2021), which in the linear case can be shown to pull the task solution towards its initialisation (Gunasekar et al., 2018), coinciding with the previous task solution. In this way, we have shown how to use the characterisation of the forgetting patterns of local and global algorithms in order to capture the implicit biases of existing algorithms, such as SGD.