Beyond multitask learning: a conceptual analysis of continual learning objectives

Anonymous authors
Paper under double-blind review

Abstract

Continual Learning solutions have historically treated multitask learning as an upper-bound for what a learning process can achieve. This is a natural assumption, given that the multitask objective directly addresses the catastrophic forgetting problem — a central focus in early works. However, this perspective overlooks other critical aspects of the continual learning problem which are essential in real-world applications. One such aspects is adaptation efficiency — the ability to rapidly learn from and respond to new data. In scenarios where the response time is critical to success, prioritizing adaptation speed can be as important as combating forgetting. In this work, we draw on principles from online learning to formalize the limitations of the multitask objective, which is at the core of many existing algorithms, with respect to training time performance. In particular, we track the *average lifelong error* of agents optimizing by gradient descent a multitask objective with optimal memory. Our theoretical results show that the performance of the multitask objective depends on a quantity which we name *instability* of the sequence of tasks. We provide empirical evidence backing our theoretical findings, and argue that, in settings where real- time performance is important, the optimal objective is inherently data dependent. Moreover, we show preliminary results of how incorporating information regarding the distributional shift could be exploited by a simple replay based method, therefore *moving beyond multitask learning*.

1 Introduction

Real-world scenarios are often dynamically evolving, and therefore learning algorithms which are able to adapt to changes in the environment can provide better performance. Continual learning (CL) (e.g. Ring, 1994; Thrun & Mitchell, 1995; Silver et al., 2013; Parisi et al., 2019; Hadsell et al., 2020; Lesort et al., 2020), sometimes referred to as lifelong learning, directly aims to address the problem of how to construct a model that continuously adapts. In some applications, such as autonomous driving and robotic or language assistants, the response time is crucial to a successful interaction with the environment. Typically, the CL problem definition — or rather the solution definition — comes down to a list of desiderata that is expected from the system (e.g Schwarz et al., 2018; Hadsell et al., 2020; Mundt et al., 2023), such as *forward and backward transfer*, and resource efficiency. Historically, addressing *catastrophic forgetting* has been seen as the first step towards solving continual learning, under the premise that retaining *some* information is essential in order to exhibit transfer. Consequently, most research has focused on resolving this specific aspect. However, this perspective largely overlooks the role of (computational) time, as catastrophic forgetting is typically measured after training. In this work *we formally introduce training time in the evaluation of the continual learner*. We consider the —arguably realistic— case in which the agent's performance on the current task matters, and hence we focus on the error accumulated *while learning the task*. In this context, we revisit the longstanding goal of minimizing catastrophic forgetting and assess its impact on training-time performance. While as argued before, in principle remembering skills learned in previous tasks can speed up learning new tasks — a core aim of continual learning — and hence reduce the metric we considered in this paper, it is not guaranteed to do so. Therefore we ask whether and when solving catastrophic forgetting may lead to higher learning time error, thus defying the other desiderata of continual learning (e.g. Wołczyk et al., 2021; Wu et al., 2023; Mundt et al., 2023).

To understand this trade-off we start by arguing that most methods aimed at solving catastrophic forgetting rely, implicitly or explicitly, on the assumption that a *multi-task* objective is optimal and effectively employ objectives which approximate the multi-task objective. Drawing on the principles of the online learning literature, we quantify training time performance using the *average lifelong error*, which aligns closely with the concept of dynamic regret, as further elaborated below. To study the optimality of the multi-task objective we design two gradient-descent agents: *single-task* (ST) and *multi-task* (MT). The ST agent forgets everything after each task, while the MT agent minimizes the multi-task objective, i.e., the average loss over all previous tasks, and represents a CL agent with minimal catastrophic forgetting. We present a theoretical and empirical study of the difference between ST and MT agents at training time. *Our key contribution is to show that there are scenarios in which an MT agent, despite preserving knowledge between tasks, can accumulates higher lifetime loss than a naive forgetting (ST) agent.* Although this finding may initially seem biased against MT agents, our analysis carefully isolates the impact of forgetting from other factors. In particular, we prove that agents minimizing forgetting can sometimes lead to higher error during training and, in certain cases, that *forgetting can actually be beneficial for adapting to a changing environment.* Finally, we demonstrate the extent of this phenomenon across a range of popular supervised learning and reinforcement learning benchmarks.

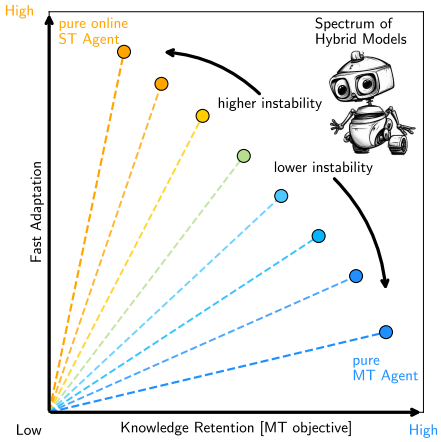

Figure 1: Both knowledge retention and fast adaptation to new data are desirable in continual learning. We show that, from optimization point of view, the current focus on multi-task (MT) performance comes at the cost of training time performance and that the optimum in the spectrum of possible optimization objectives depends on *the data instability.*

Ultimately, this paper underscores *the importance of considering the specific properties of the data stream* when selecting learning strategies in CL, or even to try to estimate these properties and adapt the CL algorithm online as data becomes available. Our results indicate that, in settings where real-time performance is important, the optimal type of agent is inherently data dependent.

2 Background

Multitask learning (Caruana, 1997) refers to a learning process that averages the losses incurred on multiple tasks. The original goal was to promote sharing of features and therefore speed up learning and resulting in solutions that generalize better. Within the Continual Learning literature the multitask objective comes about when analyzing the ability of systems to prevent *catastrophic forgetting* (McCloskey & Cohen, 1989; French, 1999) and it is consequently incorporated into several existing algorithms, either explicitly or implicitly.

CL methods and the implicit reliance on Multi-Task objectives. Traditionally — see e.g. Parisi et al. (2019) — continual learning methods tend to be grouped into three categories, according to how they approach the catastrophic forgetting problem, though this categorization is not without fault (see e.g. Titsias et al., 2019). The first category encompasses regularization based methods, such as Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017). EWC explicitly assumes the multitask solution as optimal[1], and builds the method as an approximation of this objective when one does not have access to other tasks. This multitask approximation is prevalent, even if sometimes implicitly, in many other regularization methods (e.g. Zenke et al., 2017a; Maltoni & Lomonaco, 2018; Swaroop et al., 2019; Li & Hoiem, 2017, etc.) as recently argued by Yin et al. (2020) and Lanzillotta et al. (2024). The second category of methods consist of replay methods (e.g. Robins, 1995; Shin et al., 2017), where the replay is effectively emulating the multi-task objective by representing the task not currently available.[2] The third category, dynamic architecture methods (e.g. Zhou et al., 2012; Rusu et al., 2016; Mallya & Lazebnik, 2018) avoid catastrophic forgetting by learning each task

[1]See equation (2) of their derivation.
[2]Replay emulates a weighted average objective, where the weight of each task may change with time.

with a separate subnetwork. Although the resulting capacity constraint may yield suboptimal performance on any task, the partitioning guarantees good performance on each task. We formalize these arguments in Appendix Section B in the interest of space. In general, most continual learning algorithms do not employ a multitask objective; however they can be interpreted as biased estimates thereof. *In this work we choose to look at the multitask objective as an abstraction of any specific continual learning algorithm, in order to provide a high level intuition and formalism which can be useful more broadly for the CL community.*

Training time and learning efficiency in CL One of the stated desiderata for CL systems is learning with a finite amount of computational resources (Mundt et al., 2023). This naturally calls for measuring the system *adaptation efficiency*, together with offline metrics of *knowledge retention*. In *Online Continual Learning* (OCL) (Cai et al., 2021; Lopez-Paz & Ranzato, 2017; Aljundi et al., 2019; Buzzega et al., 2020), continual learning algorithms are often evaluated using an online metric. For instance, Cai et al. (2021) argue that the *average online accuracy* a_o –directly related to the average lifelong error v – must be measured in addition to knowledge retention in order to get a full picture of the agent's performance. Intuitively, a higher average online accuracy corresponds to faster adaptation. They empirically find that learning efficacy and information retention are conflicting objectives from an optimization perspective. This is in line with the theoretical findings of Kumar et al. (2023), who show, within a information theoretic framework, that an agent with limited capacity must dynamically compromise between retaining old information and acquiring new information in order to maximise its *lifelong performance* (formalised in Section 3). Inspired by these results, our work analyses the effects of a focus on knowledge-retention on the agent's adaptation efficiency. To the best of our knowledge, we are the first to formalize the tension between these two objectives in gradient-based optimization, introducing new conceptual tools to evaluate the adaptation efficiency of multi-task objectives.

Online Learning We take inspiration from the Online Learning (OL) literature (Cesa-Bianchi & Lugosi, 2006; Hoi et al., 2018; Orabona, 2019) to evaluate the performance of an agent at training time. In this work we study a common metric in OL known as the *Dynamic Regret* (Herbster & Warmuth, 1998; Zinkevich, 2003) which compares, at each step of the learning, the current expected cost (or reward) of the agent with the minimal achievable cost (or maximal reward). This metric is particularly relevant in slowly-drifting or piecewise stationary settings such as those typically arising in CL (e.g. Hadsell et al., 2020). More precisely, we ignore the comparator and study instead the *average lifelong error* without loss of generality[3]. An important difference between our work and the standard online learning setting is that we do not assume the data to be online, i.e., that at each time step the agent processes a new data point. Instead, for us time corresponds to optimization steps, which can be computed on on datapoints that have been seen before. Specifically, in practice, the data stream is piecewise stationary, when each stationary piece is constructed from a dataset that the model might visit multiple times before moving to the next stationary piece. Also as described below, this allows us to evaluate the performance on each stationary piece by employing a validation set, in contrast to the typical setup in online learning.

3 Setup: the Average Lifelong Error

In the typical continual learning setting, the agent has to solve a *sequence of tasks*. We consider learning tasks including a target, which broadly covers supervised learning (targets are labels) and reinforcement learning (targets are actions and rewards). For each learning task $\kappa \in \{1, .., K\}$, the agent receives a dataset $D_\kappa = \{(x_1, y_1), \cdots, (x_{N_\kappa}, y_{N_\kappa})\} \sim \mathcal{D}_\kappa$ and learns to predict $Y|X$ through the parametric map $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$. For a task κ the train error is $R_\kappa(\boldsymbol{\theta}) = {}^1/_{N_\kappa} \sum_{(x,y)\in D_\kappa} \ell(\boldsymbol{\theta}; \boldsymbol{x}, y)$ and the *test error*, $\mathcal{R}_\kappa = \mathbb{E}_{(x,y)\sim\mathcal{D}_\kappa} [\ell(\boldsymbol{\theta}; x, y)]$. We consider iterative agents with h update steps in each task, such that its *lifetime*[4] is $T = hK$ and we track the (discrete) parameters dynamics $\theta(t)$ along the trajectory.

Our work proposes to compare two types of agents, a *Single Task* (ST) and a *Multi Task* (MT) agent with associated parameter dynamics $\theta_{ST}(t), \theta_{MT}(t)$. An ST agent optimizes the present task loss R_κ, and is *reset* after completing each task, effectively *forgetting everything*. It serves as a baseline for evaluating performance without employing any continual learning strategies. In contrast, the MT (Multi-Task) agent optimizes the

[3]Our derivations can equivalently be applied to dynamic regret.
[4]Our analysis can be extended without difficulty to tasks of various lengths $h_1, ..., h_K$.

average error across all tasks encountered up to the current point [5], $1/\kappa\,(R_1 + \cdots + R_\kappa)$, without considering future tasks $[\kappa+1, K]$).

Concretely, our goal is to compare the performance of these two types of agents by evaluating the differences in their respective *average lifelong error*:

$$\mathsf{v}_T = \frac{1}{T}\sum_{i=1}^{K}\sum_{t=(i-1)h+1}^{ih} \mathcal{R}_i\left(\theta(t)\right) \tag{1}$$

To do so, we define $\Delta_T = \mathsf{v}_T^{ST} - \mathsf{v}_T^{MT}$, as the difference in average lifelong error of the two agents. This quantity is central to our study.

Informally, Δ_T measures the difference in the rate at which the risk on the current task decreases during training. An agent that *adapts quickly* to the new task will have a lower average lifelong error compared to one that achieves a better final performance but at a slower pace. The retained knowledge offers the MT agent an initial advantage whenever the new task is sufficiently close to the average history. On the other hand a clean slate can benefit the ST agent when there is significant "variation" in the task sequence. In the following analysis we formalize this intuition.

Gradient Descent agents. In our theoretical analysis, we consider ST and MT agents that update their parameters sequentially using gradient descent (GD) on their respective objectives, with a fixed learning rate η. In line with the setting described above, the ST agent is reset to some $\boldsymbol{\theta}_0$ at the first step of each task, while the MT agent is not reset, although its objective is updated. Crucially, we do not assume that gradient descent is run to convergence. Instead, we are interested in modeling the effect the number of update steps per task, h, on the tradeoff between plasticity and stability.

4 Multitask is not always optimal

The primary result of this section demonstrates that, for sufficiently long tasks, the ST agent accumulates less error than MT agent while training on non-stationary task sequences where interference between tasks occurs. We formalize this finding in the specific context of convex losses for a linear regression task.

4.1 Instability and Critical Task Duration

Let $\boldsymbol{\theta}_i^\star$ and $\boldsymbol{\theta}_{[1,i]}^\star$ represent the minimizers of the respective ST and MT objectives during task i, and define $t_0^i := h(i-1)$ (see Appendix A.1.2 for an exact formula of $\boldsymbol{\theta}_i^\star$ and $\boldsymbol{\theta}_{[1,i]}^\star$ in linear regression.) Notably, our metric of interest can be expressed as:

$$\Delta_T = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{h}\sum_{t=t_0^i+1}^{ih} \underbrace{\left(\mathcal{R}_i(\theta_{ST}(t)) - \mathcal{R}_i(\boldsymbol{\theta}_i^\star)\right)}_{\Delta_T^{ST}} - \underbrace{\left(\mathcal{R}_i(\,\theta_{MT}(t)\,) - \mathcal{R}_i(\boldsymbol{\theta}_{[1:i]}^\star)\right)}_{\Delta_T^{MT}} - \underbrace{\left(\mathcal{R}_i(\boldsymbol{\theta}_{[1,i]}^\star) - \mathcal{R}_i(\boldsymbol{\theta}_i^\star)\right)}_{\Delta_T^{I}}$$

Here, we conveniently added and subtracted the risk at the optimal values that these respective agents seek. This introduces an agent-independent term, Δ_T^I, which is unaffected by the choice of agents and instead quantifies the non-stationarity of the learning problem. We refer to this term as *instability*.

We aim to identify the key factors influencing the forgetting vs. no-forgetting trade-off by establishing conditions under which $\Delta_T < 0$, i.e., $\Delta_T^{ST} < \Delta_T^{MT} + \Delta_T^I$. Note that $\Delta_T < 0$ indicates that the single-task agent has a lower *average lifelong error* (i.e., better online performance) than the multitask agent.

A critical observation is that the multitask agent benefits from a long sequence of tasks, as evidenced by the fact that $\|\boldsymbol{\theta}_{[1,\kappa-1]}^\star - \boldsymbol{\theta}_{[1:\kappa]}^\star\|_{\boldsymbol{\Sigma}_x^\kappa}^2$ decreases with increasing κ, so in convex settings [6], $\frac{1}{K}\Delta_T^{MT} \in o(1)$ (see Lemma 9 for a formal proof). Thus, for $K \gg 1$, it is both sufficient and efficient to focus on scenarios where $\Delta_T^{ST} < \Delta_T^I$.

[5] Our MT agent does not have access to future tasks as opposed to traditional MT approaches.

[6] It can be verified in experiments that Δ_T^{MT} decreases with K. Please see Table 2.

Proposition 1 defines the minimum task duration required for the single-task agent to match or outperform the multitask learner. In the convex case, using linear models we can prove that such a task duration exists and is finite (see Theorem 4), as long as the instability of the sequence is strictly positive. While the non-linear case can not be approached theoretically,we will later demonstrate that this concept remains empirically useful in such scenarios.

Proposition 1 (Critical task duration)**.** *The* critical task duration $\bar{h}$ *is the minimum task duration such that* $\Delta_T^{ST} \leq \Delta_T^I$ *for all* $h > \bar{h}$*, where* $T = hK$*.*

4.2 Linear Prediction with Convex Loss

We model each task as a noiseless linear regression problem, where for each task we have $y = {\boldsymbol{\theta}_\kappa^\star}^\top \boldsymbol{x}$. The loss function used is the *squared error*, defined as $\ell_2(\boldsymbol{\theta}; \boldsymbol{x}, y) = (\boldsymbol{\theta}^\top \boldsymbol{x} - y)^2$. Consequently, the train and test errors are expressed as follows:

$$\begin{aligned} R_\kappa(\boldsymbol{\theta}) &= (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star)^\top \hat{\boldsymbol{\Sigma}}_x^\kappa (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star) \\ \mathcal{R}_\kappa(\boldsymbol{\theta}) &= (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star)^\top \boldsymbol{\Sigma}_x^\kappa (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star) \end{aligned} \tag{2}$$

where $\boldsymbol{\Sigma}_x^\kappa = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_\kappa(X)}[\boldsymbol{x}\,\boldsymbol{x}^\top]$ and $\hat{\boldsymbol{\Sigma}}_x^\kappa = \frac{1}{N_\kappa} \sum \boldsymbol{x}_i\,\boldsymbol{x}_i^\top$ are respectively the true and empirical (uncentered) covariance matrices.

Assumption 2 (Strictly convex losses)**.** For any $\kappa \in [1, K]$ and $M > m > 0$ the spectrum of the covariance matrix satisfies the following condition: $m\,\boldsymbol{I} \preccurlyeq \hat{\boldsymbol{\Sigma}}_x^\kappa \preccurlyeq M\,\boldsymbol{I}$.

Under Assumption 2, GD is known to converge exponentially fast (Boyd & Vandenberghe, 2004). See Lemma 5 for a formal statement. In this case, the ST learner admits the following closed-form expression: within task i, the parameter update is given by $\theta_{ST}(t) = \boldsymbol{\theta}_i^\star + (\boldsymbol{I} - \eta \hat{\boldsymbol{\Sigma}}_x^i)^{t - t_0^i} (\boldsymbol{\theta}_0 - \boldsymbol{\theta}_i^\star)$. Since the number of steps per task h is limited, we can *tightly bound* the total error of the ST agent using Assumption 2 and the closed-form formula of geometric series:

$$\Delta_T^{ST} = \frac{1}{K} \sum_{i=1}^{K} \frac{1}{h} \sum_{t=1}^{h} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}_i^*\|_{\boldsymbol{\Sigma}_x^\kappa}^2 (1 - \eta \hat{\boldsymbol{\Sigma}}_x^i)^{2t} \in \Theta\left(\frac{1}{K} \sum_{i=1}^{K} \frac{\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}_i^*\|_{\boldsymbol{\Sigma}_x^\kappa}^2}{h} \frac{1 - \epsilon^h}{1 - \epsilon} \right)$$

where $\epsilon = (1 - \eta m)^2$ in the upper bound and $\epsilon = (1 - \eta M)^2$ in the lower bound. This expression leads to a tight bound on the lifelong error difference Δ_T, as presented in Theorem 13 in the appendix. Consequently, we establish a crucial first result of our study.

Corollary 3 (Monotonic dependence on task duration)**.** *For a suitable choice of learning rate and a fixed task duration* h*, gradient descent on the ST and MT convex objectives described in Section 4.2 gives rise to two parameter dynamics,* $\theta_{ST}(t)$ *and* $\theta_{MT}(t)$*, such that* Δ_T *decreases monotonically with the task duration.*

As a consequence of Corollary 3, increasing the training time will necessarily decrease the difference Δ_T. Note that (Corollary 12 in the Appendix) it is not granted that increasing h will ever result in $\Delta_T < 0$, i.e. that a critical task duration exists in general. Our main result guarantees the existence of a critical task duration, when the instability of the sequence is *strictly positive*. An informal version of the theorem is stated here, with the formal treatment detailed in the Appendix.

Theorem 4 (Existence result, informal)**.** *In the same setting as Corollary 3, if the instability of the sequence is positive then there exists a finite critical task duration* $\bar{h}$*.*

This result arises from solving for h in the bound for Δ_T, yielding a threshold value $\hat{h} < \infty$, with $\bar{h} \leq \hat{h}$ by definition. In other words, Theorem 4 proves that the MT objective is not *always* optimal with respect to training time performance. Instead, long tasks or highly non-stationary problems may be better solved with some degree of forgetting. Conversely, our study also proves that there are cases where the ST agent is not optimal either, specifically when $\Delta_T > 0$. *As a consequence, there is not a single objective which guarantees optimal training performance, instead it is determined by the data .*

While we have the full extent of our study provided in the Appendix, we summarize the key findings here: (1) that Δ_T decreases monotonically as the task duration h increases (Corollary 3); (2) if the instability $\Delta_T^I > 0$,

then there exists a finite critical task duration (Theorem 4); (3) increasing Δ_T^I decreases the critical task duration (Theorem 16).

The last point is particularly interesting, as the inversely proportional relationship between $\bar{h}$ and Δ_T^I indicates that in high instability settings the multi-task objective might always be detrimental for training time performance, and some degree of forgetting is *necessary* to adapt to the new data. These results should prompt continual learning research to move beyond the multitask objective (at least when real-time performance is of the essence): better objectives must trade-off knowledge retention and knowledge adaptation in a data-dependent fashion.

In the remainder of the paper, we assess to what extent these findings extend to the more complex setting of neural network training, evaluating the behaviour of ST and MT agents on popular supervised learning and reinforcement learning benchmarks.

A note on overparametrization. Assumption 2 implies that the system is not overparametrized, i.e. $p < N_\kappa$ for all κ. In order to deal with the overparametrized case it is sufficient to add a norm regularizer $\lambda\|\boldsymbol{\theta}\|^2$ to the loss in our derivations. This minor modification can be seamlessly integrated into our derivations without affecting the results, as we show in Section A.4.

4.3 Illustration on a simple setting

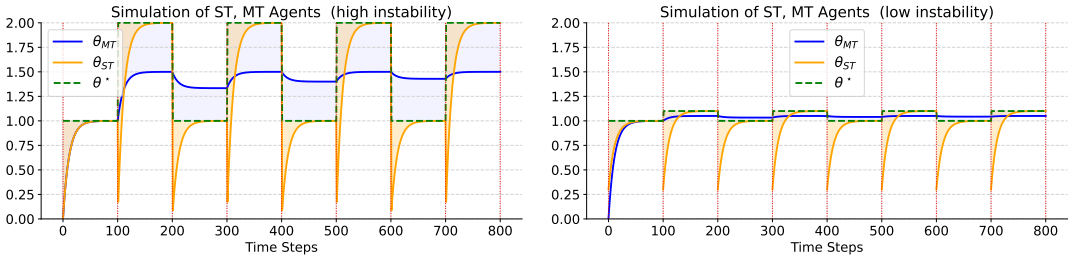

Figure 2: Toy Settings comparisons. $\theta^\star$ oscillates between 1 and 2 for each task on the left, and between 1 and 1.1 for each task on the right. There are 8 tasks (with start marked by dashed red lines) with $h = 100$ each, and $\eta = 0.01$. Both agents are initialized with $\theta_0 = 0$. The shaded area corresponds to the lifelong error of the agent.

In order to build a concrete intuition for the theoretical results we look into two toy settings, depicted in Figure 2. The tasks in the figure are one-dimensional and two different tasks with optimal solutions $\theta_1^\star$ and $\theta_2^\star$ (in green) occur repeatedly in alternating fashion. In the first case (on the left) the difference between the two solutions is 1 and in the second case (on the right) the difference is only 0.1. For the convex least-square setting, i.e $\forall i, \mathcal{R}_i(\theta) = \sigma^2\,(\theta - v_i)^2$, the instability is a function of the difference between the two solutions (full derivations in Section A.3):

$$\frac{1}{K}\,\Delta_T^I = \frac{\sigma^2}{2}\,(\theta_1^\star - \theta_2^\star)^2 \tag{3}$$

As expected, the instability is higher when the difference between task solutions is more pronounced, as seen in the left-hand figure. According to Theorem 4, a critical task duration exists for both tasks, given that instability remains strictly positive in both scenarios. From Corollary 3, it follows that, with all other factors held constant, the critical task duration is expected to be lower in the left-hand toy setting. This is evident as, despite using the same task duration of $h = 100$ in both cases, the ST agent accumulates less error over time in the first scenario, whereas the MT agent demonstrates better average performance in the second. In Section A.3, we simulate the evolution of Δ_T by varying the duration h, and confirm empirically that the critical task duration is approximately half in the first toy setting.

5 Empirical analysis

Our empirical analysis is structured into three main parts. First, we validate our theoretical framework on complex continual learning benchmarks, encompassing both supervised learning and reinforcement learning tasks. Next, we turn to two toy benchmarks, Permuted-CIFAR10 and Shuffled-CIFAR10, where we can control the task sequence's instability by adjusting the permutation strength and the fraction of shuffled labels, respectively. This setup enables us to test our theoretical predictions regarding the relationship between instability and task duration. Finally, we showcase the practical applicability of our framework in continual learning by implementing a simple variant of experience replay, with an objective tailored to the instability of the data stream.

5.1 Single task versus multi task in the wild

Benchmarks We present results for supervised learning and reinforcement learning benchmarks. For supervised learning we take two different benchmarks: the first is based on the CLEAR dataset (Lin et al., 2021), a collection of images of 10 different classes spanning the years 2004-2014. We split the collection into 10 tasks, one for each year. The second benchmark is a sequence of 5 different open source classification datasets, with no semantic overlap between them. In the interest of space we leave the details regarding the experimental setup in Appendix Section C. Hereafter we refer to this as the "MULTIDATASET" (MD5) benchmark. We have chosen these two benchmarks because they represent different types of distribution shifts. While the transitions from one task to the next in CLEAR are arguably smooth (the tasks differ in input resolution but semantically are equivalent), in MD5 they are sharp, changing the semantics of the task altogether. For reinforcement learning we rely on the Meta-World (MW) benchmark (Yu et al., 2020), which is a collection of 50 distinct simulated robotic manipulation environments. We train our agents on a sub-collection of 10 environments called ML10 and we evaluate their average lifelong reward on the same environments in an online fashion. We chose this environment due to it being previously used to highlight interference in continual learning (Wołczyk et al., 2021).

Metrics. We denote by h the number of parameter updates performed. After each update, the performance of the agent is evaluated on a separate test set, in the case of supervised learning, or on new interactions with the environment. The evaluation is always performed on the training task. Additionally, we measure the agent's knowledge retention with *multitask (offline) accuracy* ACC_{agent} or *multitask (offline) reward* R_{agent}, which consists in the average performance across all tasks after training, and is a typical CL metric (Lopez-Paz & Ranzato, 2017; Powers et al., 2022). To aid interpretability and comparison with the offline performance we report the average lifelong accuracy a_o which is more common in the literature (Cai et al., 2021). More details regarding our experimental choices in Section C.

	h	$\mathsf{a}_{o\,ST}$	$\mathsf{a}_{o\,MT}$	Δ_T	ACC_{ST}	ACC_{MT}
MD5	3000	47.0 ± 0.002	43.0 ± 0.005	-0.004 ± 0.005	19.9 ± 0.007	62.8 ± 0.007
CLEAR	3000	46.5 ± 0.0004	68.1 ± 0.0005	+0.216 ± 0.002	65.2 ± 0.012	76.8 ± 0.004
		$\mathsf{r}_{o\,ST}$	$\mathsf{r}_{o\,MT}$	Δ_T	R_{ST}	R_{MT}
ML10	500	1.15 ± 0.21	0.77 ± 0.30	-0.38 ± 0.19	1.007 ± 0.09	1.029 ± 0.14

Table 1: Average lifelong accuracy (a_o) / reward (r_o) and multitask offline accuracy (ACC) / reward (R) in the wild. Higher is better. We report the difference in performance Δ_T in the original metric, e.g $\Delta_T = \mathsf{v}_{ST} - \mathsf{v}_{MT}$ and $\Delta_T = -(\mathsf{r}_{ST} - \mathsf{r}_{MT})$. The lower ACC (R), the higher the forgetting in supervised (RL) benchmarks.

Table 1 shows the performance of the ST and MT agent on the three benchmarks. *The ST agent shows faster adaptation than the MT agent* (see $\mathbf{a}_o/\mathbf{r}_o$) in the MD5 and ML10 benchmarks, while the opposite is true in the CLEAR benchmark. This confirms our intuition that the interference between the tasks is lower in CLEAR, making the multitask objective advantageous. In Section 5.2 we quantify this statement by measuring the amount of instability Δ_T^I in all our benchmarks. Notice that *the MT agent always outperforms*

the ST agent on the multitask performance metrics (ACC/R), indicating – as expected – successful knowledge retention.

	h	$\mathsf{a}_{o\,ST}$	$\mathsf{a}_{o\,MT}$	Δ_T	ACC_{ST}	ACC_{MT}
CLEAR	3000	46.5 ± 0.0004	68.1 ± 0.0005	0.216 ± 0.002	65.2 ± 0.012	76.8 ± 0.004
	6000	56.2 ± 0.0001	71.3 ± 0.004	0.151 ± 0.004	75.9 ± 0.017	78.4 ± 0.003
	9000	61.0 ± 0.0004	72.0 ± 0.001	0.11 ± 0.001	78.1 ± 0.009	78.9 ± 0.010
	12000	64.1 ± 0.0002	73.2 ± 0.0006	**0.10** ± 0.001	76.9 ± 0.0008	79.3 ± 0.001
		$\mathsf{r}_{o\,ST}$	$r_{o\,MT}$	Δ_T	R_{ST}	R_{MT}
ML10	50	1.07 ± 0.10	0.62 ± 0.08	-0.45 ± 0.06	1.593 ± 0.12	0.355 ± 0.08
	500	1.15 ± 0.21	0.77 ± 0.30	-0.38 ± 0.19	1.007 ± 0.09	1.029 ± 0.14

Table 2: Increasing the task duration h in CLEAR and ML10, closes the gap in average lifelong performance.

Next, we ask whether increasing the task duration h would reduce the advantage of the multitask agent in CLEAR and ML10, as predicted by the theory. Table 2 shows the behaviour of our performance metrics as the task duration h is increased. *In accordance with the theory, on CLEAR we observe the error difference* Δ_T *decaying with* h, although it does not fall below 0 —suggesting that the critical task duration may be way above the range of h tested. Interestingly, we also observe that knowledge retention of both ST and MT improve as h is increased. The reason is that the similarity of the tasks grants positive transfer between them, and thus improving performance on one task by training for longer has the additional effect of increasing the performance on all the other tasks. In contrast, on the ML10 benchmark —where the ST agent consistently outperforms the MT agent at training time— the reward gap does not decay with h. We hypothesize that this may be a result of the inherent noisiness of the reward signal used as a performance metric.

Finally, we evaluate the effect of increasing K, the number of tasks in the sequence, on the trade-off between forgetting and memorizing. We perform this experiment on the ML10 benchmark, where the tasks are known to be adversarial in nature. We train the ST and MT agents on a sequence of 3, 6 or 10 tasks presented with the same ordering. If there are more difficult tasks later in the sequence increasing the number of tasks should lead to increased instability in ML10 experiments. In Table 11 we report the average reward on each task: we observe a marked difference in difficulty between the tasks, with easier tasks appearing later in the sequence. The observed increase in average lifelong rewards in Table 3 reflects the distribution of the difficulty in the task ordering. Tasks that yield higher rewards on average, boost the overall performance. Even though there is no clear monotonic trend of Δ_T, we observe that ST globally outperforms MT on average lifelong reward, which is in line with the fact that the first $K = 3$ tasks have relatively high interference and difficulty.

K	$\mathsf{r}_{o\,ST}$	$\mathsf{r}_{o\,MT}$	Δ_T	R_{ST}	R_{MT}
3	0.90 ± 0.37	0.70 ± 0.34	-0.21 ± 0.24	0.58 ± 0.06	0.81 ± 0.38
6	1.02 ± 0.30	0.92 ± 0.48	-0.10 ± 0.23	0.48 ± 0.10	1.03 ± 0.78
10	1.15 ± 0.21	0.77 ± 0.30	-0.38 ± 0.19	1.007 ± 0.09	1.029 ± 0.14

Table 3: Increasing the number of tasks in ML10. The sequence order is fixed, and the number of tasks K observed is chosen between $3, 6, 10$ (10 corresponds to the full sequence).

5.2 Empirical study on the critical task duration and instability

We move on to study empirically the critical task duration and instability in non-convex settings. According to the theory, the critical task duration depends on the sequence instability Δ_T^I, which is by definition a property of the data, independent of the agents: $\Delta_T^I = {}^1/_K \sum_{\kappa=1}^K \mathcal{R}_\kappa(\boldsymbol{\theta}^\star_{[1,\kappa]}) - \mathcal{R}_\kappa(\boldsymbol{\theta}^\star_\kappa)$. In convex settings this quantity can be directly measured (see Equation (14) for a precise formula). However when using non-linear models such as neural networks, the task minimizer $\boldsymbol{\theta}_i^\star$ is not known nor easy to discover. Additionally, when using

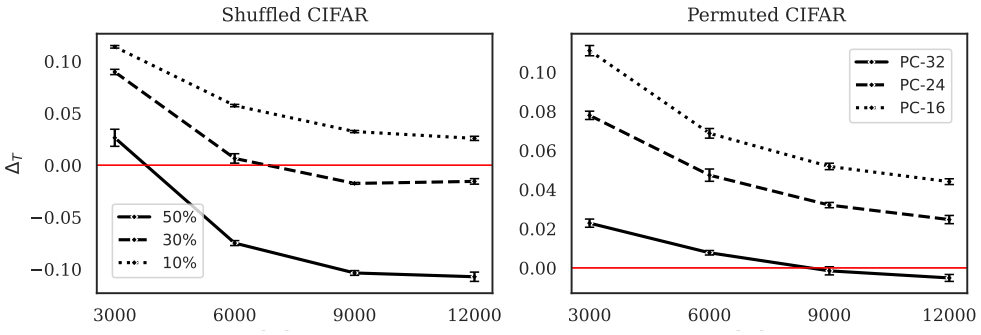

Figure 3: Δ_T as a function of task duration h in Shuffled CIFAR (left) Permuted CIFAR (right) experiments, varying levels of instability. The red horizontal line marks $\Delta_T = 0$. Confidence intervals plotted over 5 seeds.

neural networks the notion of task similarity is inherently model dependent since the features representing the data are.

Hence the question: *how can the instability be estimated in non-convex settings?*

Option 1 We propose to approximate Δ_T^I by training a neural network on the ST and MT objectives, obtaining respectively $\tilde{\boldsymbol{\theta}}_i^\star$ and $\tilde{\boldsymbol{\theta}}_{[1,i]}^\star$ and measure $\tilde{\Delta}_T^I = \frac{1}{K}\sum_{i=1}^{\kappa}(\mathcal{R}_i(\tilde{\boldsymbol{\theta}}_{[1,i]}^\star) - \mathcal{R}_i(\tilde{\boldsymbol{\theta}}_i^\star))$. Note that this quantity is *dependent* on initialization, optimizer and hyperparameters of the experimental setup.

Option 2 Intuitively, Δ_T^I should be higher when there is more interference between the tasks and lower when the tasks have more in common. Thus, we propose to measure directly the transfer between tasks as a proxy for instability. More specifically, we produce a *transfer matrix* $\mathcal{Q}$ whose i, j entry is $\mathcal{R}_j(\tilde{\boldsymbol{\theta}}_i^\star)$ and we compare the average of the diagonal to that of the off-diagonal. In practice, this second option is cheaper to compute, as it does not require to train two separate models and it can be estimated online (provided the agent has access to the full sequence).

Data	**Option 1**	**Option 2**
CLEAR	$-0.024_{\pm 0.003}$	$0.007_{\pm 0.001}$
MD5	$0.017_{\pm 0.008}$	$0.35_{\pm 0.01}$
ML10	$0.407_{\pm 0.002}$	$0.139_{\pm 0.009}$
PC-16	$-0.0213_{\pm 0.0024}$	$0.03_{\pm 0.002}$
PC-32	$0.0014_{\pm 0.004}$	$0.30_{\pm 0.005}$
SC-10%	$0.014_{\pm 0.004}$	$0.007_{\pm 0.006}$
SC-30%	$0.047_{\pm 0.009}$	$0.008_{\pm 0.001}$
SC-50%	$0.137_{\pm 0.005}$	$0.0083_{\pm 0.007}$

Table 4: Measures of instability. The higher the measure the higher the instability. The range of values is not the same for supervised and RL benchmarks. We highlight in gray the toy benchmarks.

In Table 4 we report the measurements of instability with both options. In the supervised learning benchmarks we take $\mathcal{R}(\boldsymbol{\theta})$ to be the test error (thus a quantity between 0 and 1) and in ML10 we use $\mathcal{R}(\boldsymbol{\theta}) = -\mathsf{r}(\boldsymbol{\theta})$, which is generally unbounded. Overall, we observe that the first option can be negative (because $\mathcal{R}_i(\tilde{\boldsymbol{\theta}}_i^\star) \neq 0$ for our choice of $\mathcal{R}$) and the second option is always positive (because training on a task necessarily results in a higher performance on the task, thus $\mathcal{Q}_{ii} < \mathcal{Q}_{ij}\ \forall\, j \neq i$). *Both metrics confirm the intuition that the instability is lower in the CLEAR dataset, and higher in the MD5 and ML10 datasets, which aligns with the observed* Δ_T.

Next, we wish to explore empirically how Δ_T^I impacts $\bar{h}$, by controlling Δ_T^I in two toy experimental settings. More specifically, we build the benchmarks from the CIFAR 10 data Krizhevsky & Hinton (2009) by applying transformations to the inputs or the labels which introduce instability. In particular we work with 'CIFAR10 Permuted - 16/32' (PC-16/32) – where we permute all or a part of the input image pixels – and 'CIFAR10 Shuffled - 10/30/50' (SC-10/30/50) – where we randomly draw a fixed subset of the labels for each task.

The instability measures introduced above (Table 4) validates our methodology. In Figure 3 we visualize the average lifelong error v of the ST and MT agents as we increase the task duration h. The critical task performance corresponds to the value of h where Δ_T is predicted to drop below 0. Since Δ_T is always positive in PC-16 and SC-10, we infer that the critical task duration lays beyond the explored range. However, the

critical task duration for PC-32, SC-30 and SC-50 is within the range: as predicted by the theory, higher Δ_T^I corresponds to lower $\bar{h}$.

5.3 Demo: a data-dependent objective for replay

One of the main takeaway messages of this work is that, when real-time performance is important, the optimization objective in continual learning should be a data-dependent quantity: broadly speaking, the objective should reflect the instability of the sequence, enabling forgetting when it is high and avoiding it when it is low.

We design a simple variant of the experience replay (ER) algorithm (Lin, 1992; Zhang & Sutton, 2020), which we call *Selective Replay* (SR) that does not replay from previous tasks when there is high instability in the sequence. Generally, one could rely on any heuristical measure of Δ_T^I and adapt to the current stream, trading forgetting for forward transfer. In practice, in this simple experiment we create a new controlled benchmark from CIFAR10, which we call 'C10 mixed', where we change the permutation size every 2 tasks (from 16 to 32 and viceversa) – for a total of 10 tasks . We know from Figure 3 that forgetting is beneficial when the permutation size is 32, since the instability is very high (we choose $h = 6000$ such that $\Delta_T < 0$). Intuitively, in this benchmark memory is useful only on the short sections at low instability, where there is positive transfer between the tasks. Thus, both the MT (ER) and ST agent are suboptimal, as the former is forced to remember irrelevant information -which affects its capacity to fit the new data- and the latter fails to remember any useful information. SR is designed to remember the relevant information and discard irrelevant one. We take advantage of the knowledge of the sequence, and simply change the objective from ER to the ST objective when the instability is increased.

Agent	a_o	*ACC*
ER	54 ± 0.3	**50** ± 0.4
ST	58 ± 0.1	46 ± 2
SR	**59** ± 0.05	49 ± 2

Figure 4: Cifar 10 mixed results. (Left) Test error trajectory and test error accumulated through training, evaluated on the current task, for ER (blue), ST (yellow) and SR (green). The vertical lines indicate the switch between regions of lower and higher instability. (Right) Training-time accuracy and offline accuracy for ER, ST and SR. SR has high training-time accuracy and high offline accuracy.

In Figure 4 we plot the test error over the training trajectory of the three agents: ST, ER and SR. Observe the switch in behaviour at every every two tasks (12K steps): in the low instability sections of the sequence the ER agent outperforms the ST agent, while the opposite is true in the high instability ones. Because of its dynamic objective, the SR agent is able to always adhere to the best performing behaviour, thereby accumulating less error during training. Finally, in Figure 4 (right) we record the average online accuracy and final offline accuracy: SR not only exhibits the best adaptation performance but also achieves the best tradeoff between adaptation (a_o) and retention (ACC).

Clearly, crucial to the success of selective replay, and any kind of adaptive objective, is the information regarding the tasks sequence instability -which in the case of this experiment is assumed to be known. Thus, the question becomes how to estimate Δ_T^I in an online fashion, as the data stream is being processed. We believe that this is an exciting avenue for future research, together with the study of data-dependent objectives.

6 Discussion and Conclusion

Multitask objectives arise as a natural target to address *catastrophic forgetting*. However, as was highlighted in previous works as well, the multitask objective overlooks other critical aspects of the problem, such as adaptation efficiency. In this work we formally ask whether and when an emphasis on knowledge retention naturally endows knowledge adaptation in gradient-based optimization.

Crucially we believe our work highlights at least three different observations. Firstly, while it is known that avoiding catastrophic forgetting is not the sole goal of continual learning, most methods are still developed with this goal in mind. *We argue that this is not necessary.* Indeed, we showed that the multitask objective does not always achieve knowledge retention and knowledge adaptation, while different objectives might. We argue that more continual learning methods should remove the reliance on multitask objective or at least reason explicitly about the assumptions being made. Secondly, we argue that without making assumptions on data stream, one cannot achieve fast adaptation. Thus, it should be common for continual methods to exploit the structure of the data stream, either estimating online or assuming it as initial condition. Third, in order to do the above, further formalization of the continual learning problem and *theoretical tools to describe data non-stationarity* are needed. In particular, connecting the field with related topics, such as online learning, but also others like invariances, causality, can provide a rich source to borrow from and adapt mathematical constructs.

Acknowledgements

This research was enabled in part by support provided by (Calcul Québec) (https://www.calculquebec.ca/en/) and the Digital Research Alliance of Canada (https://alliancecan.ca/en). The authors acknowledge the material support of NVIDIA in the form of computational resources. GL would like to acknowledge the support of the AI Center PhD fellowship. Additionally, MS was supported by Fonds de recherche du Québec – Nature et Technologies (FRQNT) #2023-2024-B2X-336394. We are grateful to David Abel for his generous and insightful feedback on earlier results of this work, which have made this paper better in many ways. CV is funded by the Deutsche Forschungsgemeinschaft (DFG) under both the project 468806714 of the Emmy Noether Programme and under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645. CV also thanks the international Max Planck Research School for Intelligent Systems (IMPRS-IS).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning and in particular Continual Learning. A solution to the problem of Continual Learning has many potential societal consequences, supplying Machine Learning models with a fundamentally higher degree of independence. However, this paper does not present any new solution to the problem. We discuss and prove the limitations of current approaches, which may serve as a starting point to design better solutions.

References

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. Task-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11254–11263, 2019.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Cai, Z., Sener, O., and Koltun, V. Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8281–8290, 2021.

Caruana, R. Multitask learning. *Machine learning*, 28:41–75, 1997.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge university press, 2006.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., , and Vedaldi, A. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. ISSN 13646613. doi: 10.1016/S1364-6613(99)01294-2. URL https://www.sciencedirect.com/science/article/abs/pii/S1364661399012942.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pp. 1832–1841. PMLR, 2018.

Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040, 2020. doi: 10.1016/j.tics.2020.09.004. URL https://doi.org/10.1016/j.tics.2020.09.004.

Herbster, M. and Warmuth, M. K. Tracking the best regressor. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 24–31, 1998.

Hoi, S. C. H., Sahoo, D., Lu, J., and Zhao, P. Online learning: A comprehensive survey, 2018. URL https://arxiv.org/abs/1802.02871.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 4th ieee workshop on 3d representation and recognition, at iccv 2013 (3drr-13). In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, Sydney, Australia, Dec. 8 2013.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

Kumar, S., Marklund, H., Rao, A., Zhu, Y., Jeon, H. J., Liu, Y., and Van Roy, B. Continual learning as computationally constrained reinforcement learning. *arXiv preprint arXiv:2307.04345*, 2023.

Lanzillotta, G., Singh, S. P., Grewe, B. F., and Hofmann, T. Local vs global continual learning. *arXiv preprint arXiv:2407.16611*, 2024.

Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., and Díaz-Rodríguez, N. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52 – 68, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2019.12.004. URL http://www.sciencedirect.com/science/article/pii/S1566253519307377.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. URL https://arxiv.org/abs/1606.09282.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

Lin, Z., Shi, J., Pathak, D., and Ramanan, D. The clear benchmark: Continual learning on real-world imagery. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1 (NeurIPS Datasets and Benchmarks 2021)*, 2021.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Maji, S., Rahtu, E., Kannala, J., Blaschko, M. B., and Vedaldi, A. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013. URL http://arxiv.org/abs/1306.5151.

Mallya, A. and Lazebnik, S. 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.

Maltoni, D. and Lomonaco, V. Continuous learning in single-incremental-task scenarios. *arXiv:1806.08568*, 2018.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Mundt, M., Hong, Y., Pliushch, I., and Ramesh, V. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*, 160: 306–336, 2023.

Orabona, F. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. startpage–endpage.

Powers, S., Xing, E., Kolve, E., Mottaghi, R., and Gupta, A. Cora: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents. In *Conference on Lifelong Learning Agents*, pp. 705–743. PMLR, 2022.

Ring, M. B. *Continual learning in reinforcement environments*. PhD thesis, USA, 1994.

Ritter, H., Botev, A., and Barber, D. Online structured laplace approximations for overcoming catastrophic forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

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

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv:1606.04671*, 2016.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress amp; compress: A scalable framework for continual learning. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4528–4537. PMLR, 10–15 Jul 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, Long Beach, CA, 2017.

Silver, D. L., Yang, Q., and Li, L. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*, 2013.

Swaroop, S., Nguyen, C. V., Bui, T. D., and Turner, R. E. Improving and understanding variational continual learning. *ArXiv*, abs/1905.02099, 2019. URL https://arxiv.org/abs/1905.02099.

Thrun, S. and Mitchell, T. M. Lifelong robot learning. In *The biology and technology of intelligent autonomous agents*, pp. 165–196. Springer, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.3723&rep=rep1&type=pdf.

Titsias, M. K., Schwarz, J., Matthews, A. G. d. G., Pascanu, R., and Teh, Y. W. Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1901.11356.

Vapnik, V. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

Wołczyk, M., Zając, M., Pascanu, R., Kuciński, Ł., and Miłoś, P. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Wu, Z., Tran, H., Pirsiavash, H., and Kolouri, S. Is multi-task learning an upper bound for continual learning? In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

Yin, D., Farajtabar, M., Li, A., Levine, N., and Mott, A. Optimization and generalization of regularization-based continual learning: a loss approximation viewpoint. *arXiv preprint arXiv:2006.10974*, 2020.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 1094–1100. PMLR, 2020. doi: 10.48550/arXiv.1910.10897. URL http://proceedings.mlr.press/v119/yu20a.html.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 2017a.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pp. 3987–3995. PMLR, 2017b.

Zhang, S. and Sutton, R. S. A deeper look at experience replay. *Journal or Conference Name*, Volume Number:Page Numbers, 2020.

Zhou, G., Sohn, K., and Lee, H. Online incremental feature learning with denoising autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pp. 1453–1461, 2012.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.

Appendices

A Theoretical proofs

Notation

$\mathcal{D}_1, \ldots, \mathcal{D}_K$	A sequence of tasks: K datasets
$\mathcal{X}$	Input space
$\mathcal{Y}_i$	Task $i \in [K]$ output space (may vary or be shared across tasks)
$\boldsymbol{\theta} \subseteq \mathbb{R}^P$	The neural network parameters
$_n$	Identity matrix with n rows and n columns
$\boldsymbol{\theta}$	A generic network parameters vector
$\boldsymbol{\theta}_{\texttt{Agent}}(t)$	Dynamics of the network parameters of `Agent` (ST,MT)
$\boldsymbol{\theta}_0$	Network initialization
$\ell_i(x, y, \boldsymbol{\theta})$	Task i loss function
$\mathcal{R}_i(\boldsymbol{\theta})$	Expected loss on the task i distribution $\mathcal{D}_i$
$R_t(\boldsymbol{\theta})$	Empirical loss on the task i dataset
t_0^κ	First time step of task κ, equal to $(\kappa - 1)\, h$ since all tasks last h time steps
$\|x\|_{\boldsymbol{\Sigma}} = x^\top \boldsymbol{\Sigma} x$	Elliptic norm of vector x for PSD matrix $\boldsymbol{\Sigma}$

A.1 Recall the Setup

Average lifelong error:

$$\frac{1}{T} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{x_t, y_t} \ell(\, \theta(t); x_t, y_t \,) = \frac{1}{T} \sum_{i=1}^{K} \sum_{t=1}^{h_i} \mathcal{R}_i(\, \theta(t_0^i + t) \,) \tag{4}$$

Average lifelong error difference:

$$\frac{1}{T} \sum_{i=1}^{K} \sum_{t=1}^{h_i} \left(\mathcal{R}_i(\, \theta_{ST}(t_0^i + t) \,) - \mathcal{R}_i(\, \theta_{MT}(t_0^i + t) \,) \right) \tag{5}$$

Agents' objectives:

$$\Omega_{ST}(\boldsymbol{\theta}, \kappa) = R_\kappa(\boldsymbol{\theta}) \qquad \Omega_{MT}(\boldsymbol{\theta}, \kappa) = \frac{1}{\kappa} \sum_{i=1}^{\kappa} R_i(\boldsymbol{\theta}) \tag{6}$$

A.1.1 Linear regression model

We define each task as a linear regression problem:

$$y = \boldsymbol{\theta}_\kappa^{\star\top} \boldsymbol{x} + \xi \tag{7}$$

where ξ is a noise term sampled independently for each input $\boldsymbol{x}$ with mean 0 and variance $\boldsymbol{\Sigma}^2$. In the paper we treat the noiseless case, i.e. assume $\xi = 0$. For completeness, we keep the setting formulation more general.

Let $\mathcal{D}_\kappa(X)$ the marginal distribution on the input space $\mathcal{X}$ and D_κ a dataset of size N_κ sampled i.i.d. from $\mathcal{D}_\kappa$. We denote by $\mathbf{\Sigma}_x^\kappa = \mathbb{E}_{x\sim\mathcal{D}_\kappa(X)}[\boldsymbol{x}\boldsymbol{x}^\top]$ the uncentred *population or true covariance* matrix of the inputs x. Given a training dataset of size N_κ for task κ we define the *empirical covariance* matrix as $\hat{\mathbf{\Sigma}}_x = \frac{1}{N_\kappa}\sum_{i=1}^{N_\kappa} x_i x_i^\top$.

With a squared error $\ell_2(\boldsymbol{\theta}; \boldsymbol{x}, y) = (\boldsymbol{\theta}^\top \boldsymbol{x} - y)^2$ the risk or test error $\mathcal{R}_\kappa(\boldsymbol{\theta})$ of the predictor $f_{\boldsymbol{\theta}} = \boldsymbol{\theta}^\top \boldsymbol{x}$ is:

$$\begin{aligned}\mathcal{R}_\kappa(\boldsymbol{\theta}) &= \mathbb{E}_{x,\xi}\left[\langle \boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta}, \boldsymbol{x}\rangle - \xi\right]^2 \\ &= (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star)^\top \mathbf{\Sigma}_x (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star) \;+\; \sigma^2\end{aligned} \tag{8}$$

Similarly, the training error is simply:

$$\begin{aligned}R_\kappa(\boldsymbol{\theta}) &= \frac{1}{N_\kappa}\sum_{i=1}^{N_\kappa}\left[(\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})^\top \boldsymbol{x}_i - \xi_i\right]^2 \\ &= (\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})^\top \left(\frac{1}{N_\kappa}\sum_{i=1}^{N_\kappa} \boldsymbol{x}_i \boldsymbol{x}_i^\top\right)(\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta}) - \frac{2}{N_\kappa}\sum_{i=1}^{N_\kappa}\xi_i\, \boldsymbol{x}_i^\top(\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta}) \\ &= (\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})^\top \hat{\mathbf{\Sigma}}_x^\kappa (\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta}) - \frac{2}{N_\kappa}\sum_{i=1}^{N_\kappa}\xi_i\, \boldsymbol{x}_i^\top(\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta}) \\ &\overset{\xi_i=0\forall i}{=} (\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})^\top \hat{\mathbf{\Sigma}}_x^\kappa (\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta}) \\ &= (\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})^\top \mathbf{\Sigma}_x (\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta}) - \left((\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})^\top (\mathbf{\Sigma}_x - \hat{\mathbf{\Sigma}}_x^\kappa)(\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})\right) \\ &= \mathcal{R}_k(\boldsymbol{\theta}) + \left((\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})^\top (\mathbf{\Sigma}_x - \hat{\mathbf{\Sigma}}_x^\kappa)(\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})\right)\end{aligned} \tag{9}$$

where in the last line, we highlight that in this simple setting, the training error is equal to the test error up to a vanishing error term that goes to 0 as N_κ grows large. This result is standard and typical of empirical risk minimization (Vapnik, 1991). More precisely, the norm of the error decreases in $O(1/\sqrt{N_k})$ (with a hidden constant factor that depends on the spectrum of $\mathbf{\Sigma}_x$).

Finally, notice that in the noiseless case $\mathcal{R}_\kappa(\boldsymbol{\theta}_\kappa^\star) = 0$ by Equation (8).

Assumption 2. *For any $\kappa \in [1, K]$ and $M > m > 0$ the spectrum of the covariance matrix satisfies the following condition:*

$$m\, \boldsymbol{I} \preccurlyeq \hat{\mathbf{\Sigma}}_x^\kappa \preccurlyeq M\, \boldsymbol{I}$$

A.1.2 Minimizers

Given a sequence of K tasks we can resolve for the minimizers of, respectively, the MT and ST objectives. Trivially, $\operatorname{argmin}_{\boldsymbol{\theta}} \Omega_{ST}(\boldsymbol{\theta}, \kappa) = \boldsymbol{\theta}_\kappa^\star$. For MT we have:

$$\begin{aligned}\boldsymbol{\theta}_{[1,\kappa]}^\star &:= \operatorname*{argmin}_{\boldsymbol{\theta}} \Omega_{MT}(\boldsymbol{\theta}, \kappa) \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{i\le\kappa} (\boldsymbol{\theta}_i^\star - \boldsymbol{\theta})^\top \hat{\mathbf{\Sigma}}_x^i (\boldsymbol{\theta}_i^\star - \boldsymbol{\theta}) = \Big(\sum_{i\le\kappa} \hat{\mathbf{\Sigma}}_x^i\Big)^{-1} \Big(\sum_{i\le\kappa} \hat{\mathbf{\Sigma}}_x^i\, \boldsymbol{\theta}_i^\star\Big)\end{aligned}$$

For simplicity, we denote $\sum_{i\le\kappa} \hat{\mathbf{\Sigma}}_x^i$ by $\bar{\mathbf{\Sigma}}_x^{\le\kappa}$ and $\sum_{i\le\kappa} \hat{\mathbf{\Sigma}}_x^i\, \boldsymbol{\theta}_i^\star$ by $\bar{\boldsymbol{\theta}}_{[1,\kappa]}^\star$.

A.1.3 Gradient descent dynamics

The ST agent and MT agent update their parameters by gradient descent on their respective objectives with a learning rate η. We here consider the case of full batch gradient descent. One iteration during task κ takes the form:

$$
\begin{aligned}
\theta_{ST}(t) &\leftarrow \theta_{ST}(t-1) - \eta \nabla_{\theta_{ST}(t-1)} R_\kappa(\boldsymbol{\theta}) \\
&= \theta_{ST}(t-1) - \eta\, \hat{\boldsymbol{\Sigma}}_x^\kappa \left(\theta_{ST}(t-1) - \boldsymbol{\theta}_\kappa^\star\right) \\
\theta_{MT}(t) &\leftarrow \theta_{MT}(t-1) - \frac{\eta}{\kappa} \sum_{i=1}^{\kappa} \nabla_{\theta_{MT}(t-1)} R_i(\boldsymbol{\theta}) \\
&= \theta_{MT}(t-1) - \frac{\eta}{\kappa} \sum_{i \leq \kappa} \hat{\boldsymbol{\Sigma}}_x^i \left(\theta_{MT}(t-1) - \boldsymbol{\theta}_i^\star\right)
\end{aligned}
\tag{10}
$$

Let t_0^κ be the beginning of task κ and t the absolute time step. Solving the recursion we have:

$$
\begin{aligned}
\theta_{ST}(t) &= \boldsymbol{\theta}_\kappa^\star + (\boldsymbol{I} - \eta \hat{\boldsymbol{\Sigma}}_x^\kappa)^{(t-t_o^\kappa)} \left(\boldsymbol{\theta}_0 - \boldsymbol{\theta}_\kappa^\star\right) \\
\theta_{MT}(t) &= \boldsymbol{\theta}_{[1,\kappa]}^\star + (\boldsymbol{I} - \frac{\eta}{\kappa} \bar{\boldsymbol{\Sigma}}_x^{\leq \kappa})^{(t-t_o^\kappa)} \left(\theta_{MT}(t_0^\kappa) - \boldsymbol{\theta}_{[1,\kappa]}^\star\right)
\end{aligned}
\tag{11}
$$

Note that applying Assumption 2 we directly have $\eta m\, \boldsymbol{I} \preccurlyeq \frac{\eta}{\kappa} \bar{\boldsymbol{\Sigma}}_x^{\leq \kappa} \preccurlyeq \eta M\, \boldsymbol{I}$, which allows us to use the same convergence statements for the ST and MT objectives.

The ST agent is reset after every task, and thus $\theta_{ST}(t_0^\kappa) = \boldsymbol{\theta}_0\, \forall \kappa$. In contrast, the MT agent is never reset and therefore it starts the new task from where it ended the last one $\theta_{MT}(t_0^\kappa) = \boldsymbol{\theta}_0 \iff t_0^\kappa = 0$. The task initialization $\theta_{MT}(t_0^\kappa)$ admits a closed-form expression:

$$
\begin{aligned}
\theta_{MT}(t_0^{\kappa+1}) &= \sum_{j=0}^{\kappa} \Big[\prod_{i=j+1}^{i} \underbrace{(\boldsymbol{I} - \frac{\eta}{i} \bar{\boldsymbol{\Sigma}}_x^{\leq i})}_{P_i}{}^{h_i} \Big] \left(\boldsymbol{I} - (\boldsymbol{I} - \frac{\eta}{j} \bar{\boldsymbol{\Sigma}}_x^{\leq j})^{h_j} \right) \boldsymbol{\theta}_{[1,j]}^\star \\
&= \sum_{j=0}^{\kappa} \Big[\prod_{i=j+1}^{i} P_i{}^{h_i} \Big] \left(\boldsymbol{I} - P_j^{h_j} \right) \boldsymbol{\theta}_{[1,j]}^\star
\end{aligned}
\tag{12}
$$

where, with an abuse of notation we denote $P_0 = 0$ and $\boldsymbol{\theta}_{\leq 0}^\star = \boldsymbol{\theta}_0$.

A.2 Average lifelong error difference

Lemma 5. *For any strictly convex loss R, i.e., there exists $m, M > 0$ such that $m\boldsymbol{I} \leq \nabla^2 R(\boldsymbol{\theta}) \leq M\boldsymbol{I}$ for all $\boldsymbol{\theta}$, the convergence of (full-batch) discrete time gradient descent with learning rate η is geometric and we have:*

$$
\begin{aligned}
\|\theta(k) - \boldsymbol{\theta}^*\|_2 &\leq (1 - \eta m)^k \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2 \qquad & R(\theta(k)) - R(\boldsymbol{\theta}^*) &\leq (1 - \eta m)^{2k} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_{\boldsymbol{\Sigma}_x}^2 \\
\|\theta(k) - \boldsymbol{\theta}^*\|_2 &\geq (1 - \eta M)^k \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2 \qquad & R(\theta(k)) - R(\boldsymbol{\theta}^*) &\geq (1 - \eta M)^{2k} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_{\boldsymbol{\Sigma}_x}^2
\end{aligned}
$$

where $\boldsymbol{\Sigma}_x = \nabla^2 R(\boldsymbol{\theta})$.

Assumption 6 (Learning rate)**.** The learning rate is chosen such that gradient descent converges to a minimum. If Assumption 2 is satisfied this is simply: $\eta < \frac{1}{M}$.

Definition 7 (Decomposition of Δ_T)**.** We identify three separate elements which contribute independently to the average lifelong error difference Δ_T, namely:

$$
\begin{aligned}
\Delta_T^I &= \sum_{i=1}^{K} \mathcal{R}_i(\boldsymbol{\theta}_{[1,i]}^\star) - \mathcal{R}_i(\boldsymbol{\theta}_i^\star) \\
\Delta_T^{MT} &= \sum_{i=1}^{K} \left(\frac{1}{h} \sum_{t=t_0^i+1}^{ih} \mathcal{R}_i(\, \boldsymbol{\theta}_{MT}^{(i)}(t)\,) - \mathcal{R}_i(\boldsymbol{\theta}_{[1:i]}^\star) \right) \\
\Delta_T^{ST} &= \sum_{i=1}^{K} \left(\frac{1}{h} \sum_{t=t_0^i+1}^{ih} \mathcal{R}_i(\, \boldsymbol{\theta}_{ST}^{(i)}(t)\,) - \mathcal{R}_i(\boldsymbol{\theta}_i^\star) \right)
\end{aligned}
$$

Further,

$$
\Delta_T = \frac{1}{K} \Delta_T^{ST} - \frac{1}{K} \Delta_T^{MT} - \frac{1}{K} \Delta_T^I
$$

Theorem 8 (General upper bound on Δ_T)**.** *For clarity in the notation, we fix $h_\kappa = h$ for all tasks, and denote $\epsilon_m = (1-\eta m)^2$ where η is the GD step size and m is from Assumption 2. If Assumption 6 and Assumption 2 are satisfied then the difference in average lifelong error of the ST and MT agents with dynamics described by Equation (11) admits the following upper bound:*

$$\Delta_T \leq \frac{1}{K}\sum_{\kappa=1}^{K}\left(\frac{1}{h}\cdot\frac{1-\epsilon_m^h}{1-\epsilon_m}\|\boldsymbol{\theta}_0-\boldsymbol{\theta}_\kappa^*\|^2_{\boldsymbol{\Sigma}_x^\kappa}+O(1/N_\kappa)\right)-\frac{1}{K}\Delta_T^I \tag{13}$$

Proof We start from the general decomposition of Definition 7. We bound each task term $\Delta_T^{ST}(t_0^\kappa+t), \Delta_T^{MT}(t_0^\kappa+t)$ separately. $\Delta_T^I \geq 0$ cannot be bounded further since it is not agent dependent. However we can rewrite is as follows:

$$\frac{1}{K}\Delta_T^I = \frac{1}{K}\sum_{\kappa=1}^{K}\left(\mathcal{R}_\kappa(\boldsymbol{\theta}^\star_{[1,\kappa]})-\mathcal{R}_\kappa(\boldsymbol{\theta}^\star_\kappa)\right) = \frac{1}{K}\sum_{\kappa=1}^{K}\|\boldsymbol{\theta}^\star_{[1,\kappa]}-\boldsymbol{\theta}^\star_\kappa\|^2_{\boldsymbol{\Sigma}_x^\kappa} \tag{14}$$

Both ST and MT are gradient descent agents that optimize a convex objective. By Lemma 5 and Assumption 2 we have that the train error with respect to the minimum will converge to 0 at a geometric rate. Using a generic concentration argument to upper bound the difference between the empirical risk on the train set and the test error: $R_\kappa(\boldsymbol{\theta}_\kappa^*)-\mathcal{R}_\kappa(\boldsymbol{\theta}_\kappa^*)$ (the train and test set being identically distributed) we get:

$$\Delta_T^{ST}(t_0^\kappa+t) \leq (1-\eta m)^{2t}\|\boldsymbol{\theta}_0-\boldsymbol{\theta}_\kappa^*\|^2_{\boldsymbol{\Sigma}_x^\kappa}+O(1/N_\kappa)$$

and

$$\Delta_T^{MT}(t_0^\kappa+t) \geq (1-\eta M)^{2t}\|\boldsymbol{\theta}_{\mathrm{MT}}(t_0^\kappa)-\boldsymbol{\theta}^\star_{[1:\kappa]}\|^2_{\boldsymbol{\Sigma}_x^\kappa}+O(1/N_\kappa)$$

First note that for $\kappa \gg 0$,

$$\Delta_T^{MT}(t_0^\kappa+t) \geq (1-\eta M)^{2t}\|\boldsymbol{\theta}_{\mathrm{MT}}(t_0^\kappa)-\boldsymbol{\theta}^\star_{[1:\kappa]}\|^2_{\boldsymbol{\Sigma}_x^\kappa}+O(1/N_\kappa) \gtrsim 0$$

because $\boldsymbol{\theta}_{\mathrm{MT}}(t_0^\kappa) \approx \boldsymbol{\theta}^\star_{[1:\kappa-1]} \approx \boldsymbol{\theta}^\star_{[1:\kappa]}$ is close to the minimum at the previous task, which is itself similar to the current minimum. So in general, we can grossly lower bound $\Delta_T^{MT}(t_0^\kappa+t) > 0$ without making a large error (see Lemma 9 for a formal proof).

Recognising that $(1-\eta m)^{2t}$ forms a geometric series with base $\epsilon_m = (1-\eta m)^2$, we can write :

$$\Delta_T \leq \frac{1}{K}\sum_{\kappa=1}^{K}\frac{1}{h}\left(\frac{1-\epsilon_m^h}{1-\epsilon_m}\|\boldsymbol{\theta}_0-\boldsymbol{\theta}_\kappa^*\|^2_{\boldsymbol{\Sigma}_x^\kappa}\right)+O(1/N_\kappa)-\frac{1}{K}\Delta_T^I \tag{15}$$

This concludes the proof.

Lemma 9. *The error term due to the MT agent is negligible:*

$$\frac{1}{K}\sum_{\kappa=1}^{K}\|\boldsymbol{\theta}_{MT}(t_0^\kappa)-\boldsymbol{\theta}^\star_{[1:\kappa]}\|^2_{\boldsymbol{\Sigma}_x^\kappa} \in o(1)$$

Proof Using Equation (12) we can write

$$\begin{aligned}
\boldsymbol{\theta}_{\mathrm{MT}}(t_0^\kappa)-\boldsymbol{\theta}^\star_{[1:\kappa]} &= \sum_{j=0}^{\kappa}\Big[\prod_{i=j+1}^{i}P_i^{h_i}\Big]\left(\boldsymbol{I}-P_j^{h_j}\right)\boldsymbol{\theta}^\star_{[1,j]}-\boldsymbol{\theta}^\star_{[1:\kappa]}\\
&= \left(\boldsymbol{I}-P_{\kappa-1}^h\right)\boldsymbol{\theta}^\star_{[1,\kappa-1]}\\
&+ P_{\kappa-1}^h\left(\boldsymbol{I}-P_{\kappa-2}^h\right)\boldsymbol{\theta}^\star_{[1,\kappa-2]}\\
&+ P_{\kappa-1}^h P_{\kappa-2}^h\left(\boldsymbol{I}-P_{\kappa-3}^h\right)\boldsymbol{\theta}^\star_{[1,\kappa-3]}\\
&+ \dots\\
&+ P_{\kappa-1}^h\dots P_2^h\left(\boldsymbol{I}-P_1^h\right)\boldsymbol{\theta}_1^\star + P_{\kappa-1}^h\dots P_1^h\,\boldsymbol{\theta}_0-\boldsymbol{\theta}^\star_{[1:\kappa]}
\end{aligned}$$

By Assumption 2 we know $P_i^h \preccurlyeq (1-\eta m)^h\boldsymbol{I}$ for all the tasks i and thus we can ignore the contribution of all the terms $j < \kappa-1$ in the norm:

$$\|\boldsymbol{\theta}_{\mathrm{MT}}(t_0^\kappa)-\boldsymbol{\theta}^\star_{[1:\kappa]}\|^2_{\boldsymbol{\Sigma}_x^\kappa} \leq \|\boldsymbol{\theta}^\star_{[1,\kappa-1]}-\boldsymbol{\theta}^\star_{[1:\kappa]}\|^2_{\boldsymbol{\Sigma}_x^\kappa}$$

As κ increases the average will converge to the final average $\boldsymbol{\theta}^{\star}_{[1,K]}$, and $\|\boldsymbol{\theta}^{\star}_{[1,\kappa-1]} - \boldsymbol{\theta}^{\star}_{[1:\kappa]}\|^2_{\boldsymbol{\Sigma}_x^\kappa} \to 0$. In general we can say that $\|\boldsymbol{\theta}_{\mathrm{MT}}(t_0^\kappa) - \boldsymbol{\theta}^{\star}_{[1:\kappa]}\|^2_{\boldsymbol{\Sigma}_x^\kappa}$ decreases with κ and thus:

$$\frac{1}{K}\sum_{\kappa=1}^{K} \|\boldsymbol{\theta}_{\mathrm{MT}}(t_0^\kappa) - \boldsymbol{\theta}^{\star}_{[1:\kappa]}\|^2_{\boldsymbol{\Sigma}_x^\kappa} \in o(1)$$

Corollary 10. *Consider the setting where Assumption 2 and Assumption 6 are satisfied. If the instability of the sequence is null, i.e. $\Delta_T^I = 0$, then the upper bound in Theorem 8 is always positive.*

This result is a direct consequence of the general upper bound above. In particular, Lemma 9 shows that in such setting the error of the MT agents goes to 0 geometrically fast so it is the optimal type of agent.

Theorem 11 (General lower bound on Δ_T)**.** *In the same setting as Theorem 8, using $\epsilon_M = (1-\eta M)^2$, if Assumption 6 and Assumption 2 are satisfied then the difference in average lifelong error of the ST and MT agents with dynamics described by Equation (11) admits the following upper bound:*

$$\Delta_T \geq \frac{1}{K}\sum_{\kappa=1}^{K} \frac{1}{h}\left(\frac{1-\epsilon_M^h}{1-\epsilon_M}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*_\kappa\|^2_{\boldsymbol{\Sigma}_x^\kappa} - \frac{1-\epsilon_m^h}{1-\epsilon_m}\right) - \frac{1}{K}\Delta_T^I$$

Proof The proof is similar to Theorem 11.

Again, we start from the general decomposition of Definition 7. Both ST and MT are gradient descent agents that optimize a convex objective. By Lemma 5 and Assumption 2 we have that the train error with respect to the minimum will converge to 0 at a geometric rate. Using a generic concentration argument to upper bound the difference between the empirical risk on the train set and the test error: $R_\kappa(\boldsymbol{\theta}^*_\kappa) - \mathcal{R}_\kappa(\boldsymbol{\theta}^*_\kappa)$ (the train and test set being identically distributed) we get:

$$\Delta_T^{ST}(t_0^\kappa + t) \geq (1-\eta M)^{2t}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*_\kappa\|^2_{\boldsymbol{\Sigma}_x^\kappa} s$$

and

$$\Delta_T^{MT}(t_0^\kappa + t) \leq (1-\eta m)^{2t}\|\boldsymbol{\theta}_{\mathrm{MT}}(t_0^\kappa) - \boldsymbol{\theta}^{\star}_{[1:\kappa]}\|^2_{\boldsymbol{\Sigma}_x^\kappa} + O(1/N_\kappa)$$

Recognizing that $(1-\eta m)^{2t}$ and $(1-\eta M)^{2t}$ form a geometric series with base $\epsilon_m = (1-\eta m)^2$ and $\epsilon_M = (1-\eta M)^2$ respectively, we can write :

$$\Delta_T \geq \frac{1}{K}\sum_{\kappa=1}^{K} \frac{1}{h}\left(\frac{1-\epsilon_M^h}{1-\epsilon_M}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*_\kappa\|^2_{\boldsymbol{\Sigma}_x^\kappa} - \frac{1-\epsilon_m^h}{1-\epsilon_m}\|\boldsymbol{\theta}_{MT}(t_0^\kappa) - \boldsymbol{\theta}^*_\kappa\|^2_{\boldsymbol{\Sigma}_x^\kappa}\right) + O(1/N_\kappa) - \frac{1}{K}\Delta_T^I$$

Applying Lemma 9 we know that the second term vanishes with K:

$$\frac{1}{K}\sum_{\kappa=1}^{K} \|\boldsymbol{\theta}_{MT}(t_0^\kappa) - \boldsymbol{\theta}^*_\kappa\|^2_{\boldsymbol{\Sigma}_x^\kappa} \in o(1) \tag{16}$$

and thus

$$\Delta_T \geq \frac{1}{K}\sum_{\kappa=1}^{K} \frac{1}{h}\left(\frac{1-\epsilon_M^h}{1-\epsilon_M}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*_\kappa\|^2_{\boldsymbol{\Sigma}_x^\kappa} - \frac{1-\epsilon_m^h}{1-\epsilon_m}\right) + O(1/N_\kappa) - \frac{1}{K}\Delta_T^I$$

This concludes the proof.

Corollary 12. *Consider the setting where Assumption 2 and Assumption 6 are satisfied. Let $V_K = \sum_{\kappa=1}^K \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*_\kappa\|^2_{\boldsymbol{\Sigma}_x^\kappa}$ measure a quantity measuring the 'spread' of the task solution vectors, with respect to initialization, and further let $\omega_M = \frac{1-\epsilon_M^h}{1-\epsilon_M}$ and $\omega_m = \frac{1-\epsilon_m^h}{1-\epsilon_m}$. The lower bound in Theorem 11 is positive if the following is true:*

$$LB > 0 \iff V_K > \frac{\omega_m}{\omega_M} + \frac{h}{\omega_M}\Delta_T^I \tag{17}$$

And thus if the instability of the sequence is null, i.e. $\Delta_T^I = 0$ then the lower bound in Theorem 11 is positive only if $V_K > \frac{\omega_m}{\omega_M}$.

Proof Let LB denote the lower bound on Δ_T of Theorem 11:

$$LB = \frac{1}{K}\sum_{\kappa=1}^{K}\frac{1}{h}\left(\frac{1-\epsilon_M^h}{1-\epsilon_M}\|\boldsymbol{\theta}_0-\boldsymbol{\theta}_\kappa^*\|^2_{\boldsymbol{\Sigma}_x^\kappa}-\frac{1-\epsilon_m^h}{1-\epsilon_m}\right)-\frac{1}{K}\Delta_T^I$$

$$LB>0 \iff \frac{1}{K}\sum_{\kappa=1}^{K}\frac{1}{h}\left(\frac{1-\epsilon_M^h}{1-\epsilon_M}\|\boldsymbol{\theta}_0-\boldsymbol{\theta}_\kappa^*\|^2_{\boldsymbol{\Sigma}_x^\kappa}\right)>\frac{1}{h}\frac{1-\epsilon_m^h}{1-\epsilon_m}+\frac{1}{K}\Delta_T^I$$

$$\frac{1-\epsilon_M^h}{1-\epsilon_M}\left(\frac{1}{K}\sum_{\kappa=1}^{K}\|\boldsymbol{\theta}_0-\boldsymbol{\theta}_\kappa^*\|^2_{\boldsymbol{\Sigma}_x^\kappa}\right)>\frac{1-\epsilon_m^h}{1-\epsilon_m}+h\,\frac{1}{K}\Delta_T^I$$

which concludes the proof.

Corollary 12 highlights the role of the task duration h in the balance between ST and MT agents. As h increases it becomes harder for the MT agent to match the performance of the ST agent. Another consequence of Corollary 12 is that a positive instability does not imply a positive Δ_T. For instance, if the solutions are all δ-close ($\delta = o(\frac{\omega_m}{\omega_M})$) to the initialization (e.g. by being of low norm) then the ST agent may still outperform the MT agent.

Moreover, since both the upper and lower bound on Δ_T vary as h^{-1} we can say that $\Delta_T \in \Omega(h^{-1})$, which confirms that increasing the task duration will always lead to lower Δ_T.

Theorem 13 (Asymptotically tight bounds for Δ_T)**.** *Let $V_K = \sum_{\kappa=1}^{K}\|\boldsymbol{\theta}_0-\boldsymbol{\theta}_\kappa^*\|^2_{\boldsymbol{\Sigma}_x^\kappa}$ as in Corollary 12. In the same setting as Theorems 8 and 11 if Assumption 2 and Assumption 6 are satisfied then the difference in average lifelong error described by Equation (11) can be tightly bounded as follows:*

$$\Delta_T \in \Theta\left(\frac{1}{K}\left(\tfrac{1}{h}V_K-\Delta_T^I\right)+\frac{1}{N_\kappa}+C\right) \tag{18}$$

where C is hiding a constant which depends only on the spectrum of the covariance matrices.

Proof The theorem is a direct consequence of Theorem 8 and Theorem 11.

Interestingly, Theorem 13 highlights the nature of the dependence of Δ_T on h, which is essentially monotonic. The following corollary formalizes this observation.

Corollary 3. *In the same setting as Theorems 8 and 11 if Assumption 2 and Assumption 6 are satisfied then the difference in average lifelong error described by Equation (11) decreases monotonically with the task duration.*

Corollary 3 provides fundamental insight for our study, and has high practical relevance. The task duration h is typically under the control of the agent designer. By Corollary 3 we know that increasing the task duration will necessarily decrease the difference Δ_T. However (Corollary 12) it is not granted that Δ_T will in general be negative, i.e. that a critical task duration exists in general.

In order to prove the existence of a critical task duration we need to consider the worst case scenario, i.e. the upper bound on Δ_T. We are thus looking for cases where the instability is not 0, i.e. $\Delta_T^I > 0$. This is what the next set of results looks at.

Theorem 14 (Negative Δ_T with positive instability)**.** *Consider the setting where Assumption 2 and Assumption 6 are satisfied. If the instability of the sequence is strictly positive, then the upper bound in Theorem 8 is strictly negative if:*

$$h > \frac{\sum_{\kappa=1}^{K}\|\boldsymbol{\theta}_0-\boldsymbol{\theta}_\kappa^*\|^2_{\boldsymbol{\Sigma}_x^\kappa}}{(1-\epsilon_m)\sum_{\kappa=1}^{K}\|\boldsymbol{\theta}^\star_{[1,\kappa]}-\boldsymbol{\theta}_\kappa^\star\|^2_{\boldsymbol{\Sigma}_x^\kappa}} := \hat{h} \tag{19}$$

Proof We simply solve for $\Delta_T < 0$ in Theorem 8:

$$\Delta_T < 0 \Leftarrow h > \frac{\sum_{\kappa=1}^{K}\|\boldsymbol{\theta}_0-\boldsymbol{\theta}_\kappa^*\|^2_{\boldsymbol{\Sigma}_x^\kappa}}{(1-\epsilon_m)\sum_{\kappa=1}^{K}\|\boldsymbol{\theta}^\star_{[1,\kappa]}-\boldsymbol{\theta}_\kappa^\star\|^2_{\boldsymbol{\Sigma}_x^\kappa}}$$

Theorem 15 (Existence of the critical task duration.)**.** *In the setting where Assumption 2 and Assumption 6 are satisfied, if the instability of the sequence is strictly positive, gradient descent on the ST and MT convex objectives described in Section 4.2 gives rise to two parameter dynamics $\theta_{ST}(t)$ and $\theta_{MT}(t)$, such that there exists a finite critical task duration $\bar{h}$.*

Proof The result follows directly from Theorem 14. By definition (Proposition 1), the critical task duration is the minimal value of h such that $\Delta_T < 0$. Since we know by Theorem 14 that $\Delta_T < 0 \; \forall \, h < \hat{h}$ then we know that $\bar{h} \leq \hat{h}$. Noticing that $\hat{h}$ is finite if the instability is strictly positive, then necessarily so is $\bar{h}$.

Theorem 16 (Order of magnitude of the critical task duration)**.** *With all the conditions of Theorem 13, ignoring the constants C and N_κ we know that the critical task duration admits the following asymptotic expression:*

$$\bar{h} \in \Theta\left(\frac{V_K}{\Delta_T^I}\right) \tag{20}$$

Proof Solving for h in Theorem 13 and ignoring the terms depending on N_κ or C leads to the theorem statement.

Theorem 16 provides interesting insights. In particular, at higher instability in general the critical task duration is lower, which means that the multi-task solutions is more likely to achieve worse lifelong performance. At the same time, the norm of the solutions with respect to the initialization V_K influences the balance between the two agents too. With the norm of the solutions tending to 0, the ST agent may still be more performing even in very stable environments.

A.3 Toy settings

For the toy settings in Figure 2 we can obtain explicit expressions by computing Δ_T^I and Δ_T^{ST} exactly.

In a one-dimensional problem the risk is simply $\mathcal{R}_i(\theta) = \sigma^2 \, (\theta - v_i)^2$, where w.l.o.g. we use $\Sigma_x = \sigma^2$. The MT objective minimizer after κ tasks is:

$$\begin{aligned}
\theta^\star_{[1,\kappa]} &= (\sum_{i \leq \kappa} \hat{\mathbf{\Sigma}}_x)^\dagger \underbrace{(\sum_{i \leq \kappa} \hat{\mathbf{\Sigma}}_x \theta_i^\star)}_{\text{all average but the last one if odd}} \\
&= (\kappa\sigma^2)^{-1}(\sigma^2 \left\lfloor \frac{\kappa}{2} \right\rfloor (\theta_1^\star + \theta_2^\star) + \mathbf{1}_{\{\kappa \text{ odd}\}} \sigma^2 \theta_1^*) \\
&= \begin{cases} \mu & \text{if } \kappa \text{ even} \\ \frac{\kappa-1}{\kappa}\mu + \frac{1}{\kappa}\theta_1^\star & \text{if } \kappa \text{ odd} \end{cases}
\end{aligned} \tag{21}$$

where $\mu = \frac{1}{2}(\theta_1^\star + \theta_2^\star)$ is the average solution. Thus, we can easily compute Δ_T^I:

$$\begin{aligned}
\sigma^2 \|\theta^\star_{[1,\kappa]} - \theta^\star_\kappa\|^2 &= \begin{cases} \frac{\sigma^2}{2}(\theta_1^\star - \theta_2^\star)^2 & \text{if } \kappa \text{ even} \\ \frac{\sigma^2}{2} \cdot \frac{\kappa-1}{\kappa}(\theta_1^\star - \theta_2^\star)^2 & \text{if } \kappa \text{ odd} \end{cases} \quad \to_{\kappa \to \infty} \frac{\sigma^2}{2}(\theta_1^\star - \theta_2^\star)^2 \\
\Delta_T^I &= \sum_{k=1}^K \sigma^2 \|\theta^\star_{[1,\kappa]} - \theta^\star_\kappa\|^2 = K \frac{\sigma^2}{2}(\theta_1^\star - \theta_2^\star)^2
\end{aligned} \tag{22}$$

Further, in Figure 2 we use $\theta_0 = 0$, therefore we have:

$$V_K = \sum_{\kappa=1}^K \sigma^2 (\theta_0 - \theta_\kappa^\star)^2 = K \frac{\sigma^2}{2}(\theta_1^{\star 2} + \theta_2^{\star 2}) \tag{23}$$

Finally by Theorem 14 we know that the critical task duration is at most:

$$\bar{h} \leq \frac{V_K}{\Delta_T^I} = \frac{\theta_1^{\star 2} + \theta_2^{\star 2}}{(1-\epsilon)(\theta_1^\star - \theta_2^\star)^2} \tag{24}$$

where $\epsilon = (1 - \sigma\eta)^2$.

In our toy example in Figure 2, we chose $\eta = 0.01$ and $\sigma^2 = 9$, $\theta_1^\star = 1$ and $\theta_2^\star = 2$ in the left plot and $\theta_2^\star = 1.1$ in the right plot. So we can solve for $\bar{h}^{left} \leq 29.09$ and $\bar{h}^{right} \leq 7478.9$.

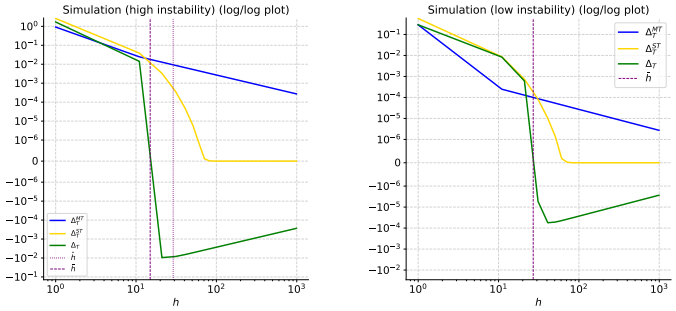

Figure 5: Simulation of Δ_T as a function of T for the two toy settings of Figure 2.

Simulations. In order to get the precise value of $\bar{h}$ for our two toy settings we run 100 simulations as we vary $h \in [1, 1000]$, keeping all the other variables fixed. We plot the average lifelong error of ST and MT, and the respective Δ_T as a function of T in Figure 5. The observed $\bar{h}$ is way lower than its predicted upper bound $\hat{h}$, however the critical task duration is lower for higher instability -as expected. Also notice that when $\Delta_T^{ST} \approx 0$ the Δ_T grows less negative as h is increased. This is a case that is not covered by the theory, since we work with the approximation $\frac{1}{K}\Delta_T^{MT} \approx 0$, whereas at very high h, the effect of $\frac{1}{K}\Delta_T^{MT}$ is much more pronounced compared to $\frac{1}{K}\Delta_T^{ST}$.

A.4 Overparametrization

Assumption 2 implies that the number of data points for each task N_κ is at least equal to the number of parameters of the model p, i.e. $\min_\kappa N_\kappa \geq p$. If this condition is not satisfied, there exist infinitely many vectors which minimize the loss. It is known that gradient descent has an implicit bias towards minimum norm solutions (Gunasekar et al., 2018; Zhang & Sutton, 2020). Therefore, without changing the characteristics of the solution, we can augment the task loss with a regularizer. Denoting the overparametrized case with the ^{o} superscript:

$$\begin{aligned} \mathcal{R}_\kappa^o(\boldsymbol{\theta}) &= \mathbb{E}_{x,\xi}\left[\langle \boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta}, \boldsymbol{x}\rangle\right]^2 + \lambda \left\|\boldsymbol{\theta}\right\|^2 \\ &= (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star)^\top \boldsymbol{\Sigma}_x (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star) + \lambda \left\|\boldsymbol{\theta}\right\|^2 \\ R_\kappa^o(\boldsymbol{\theta}) &= \frac{1}{N_\kappa}\sum_{i=1}^{N_\kappa}\left[(\boldsymbol{\theta}_\kappa^\star - \boldsymbol{\theta})^\top \boldsymbol{x}_i\right]^2 + \lambda \left\|\boldsymbol{\theta}\right\|^2 \\ &= (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star)^\top \hat{\boldsymbol{\Sigma}}_x (\boldsymbol{\theta} - \boldsymbol{\theta}_\kappa^\star) + \lambda \left\|\boldsymbol{\theta}\right\|^2 \end{aligned} \tag{25}$$

Next, we reproduce some key steps of our analysis with this modified loss in order to show that our derivations are readily extended to the overparametrized case.

The new minimizers of the ST and MT objectives are:

$$\begin{aligned} \boldsymbol{\theta}_\kappa^{o,\star} &= \text{argmin}_{\boldsymbol{\theta}}\, \Omega_{ST}(\boldsymbol{\theta}, \kappa) = (\lambda I + \boldsymbol{\Sigma}_x^\kappa)^{-1} \boldsymbol{\Sigma}_x^\kappa\, \boldsymbol{\theta}_\kappa^\star \\ \boldsymbol{\theta}_{[1,\kappa]}^{o,\star} &= \text{argmin}_{\boldsymbol{\theta}}\, \Omega_{MT}(\boldsymbol{\theta}, \kappa) = (\kappa\lambda I + \boldsymbol{\Sigma}_x^{\leq\kappa})^{-1}\, \bar{\boldsymbol{\theta}}_{[1,\kappa]}^\star \end{aligned} \tag{26}$$

And the gradient descent dynamics for the two agents take the form:

$$\begin{aligned} \theta_{ST}^o(t) &\leftarrow \theta_{ST}^o(t-1) - \eta \nabla_{\theta_{ST}^o(t-1)} R_\kappa^o(\boldsymbol{\theta}) \\ &= \theta_{ST}^o(t-1) - \eta\left(\hat{\boldsymbol{\Sigma}}_x^\kappa\left(\theta_{ST}^o(t-1) - \boldsymbol{\theta}_\kappa^\star\right) + \lambda \theta_{ST}^o(t-1)\right) \\ &= (1 - \eta\lambda)\,\theta_{ST}^o(t-1) - \eta\,\hat{\boldsymbol{\Sigma}}_x^\kappa\left(\theta_{ST}(t-1) - \boldsymbol{\theta}_\kappa^\star\right) \end{aligned} \tag{27}$$

$$\begin{aligned}\theta^o_{MT}(t) &\leftarrow \theta^o_{MT}(t-1) - \frac{\eta}{\kappa}\sum_{i=1}^{\kappa} \nabla_{\theta^o_{MT}(t-1)} R^o_i(\boldsymbol{\theta}) \\ &= (1-\eta\lambda)\,\theta^o_{MT}(t-1) - \frac{\eta}{\kappa}\sum_{i\leq\kappa} \hat{\boldsymbol{\Sigma}}^i_x\,(\theta_{MT}(t-1) - \boldsymbol{\theta}^\star_i)\end{aligned} \tag{28}$$

Let $\lambda' = 1-\eta\lambda$. Solving the recursion we have:

$$\begin{aligned}\theta^o_{ST}(t) &= \boldsymbol{\theta}^{o,\star}_\kappa + (\lambda' \boldsymbol{I} - \eta\hat{\boldsymbol{\Sigma}}^\kappa_x)^{(t-t^\kappa_o)}\,(\boldsymbol{\theta}_0 - \boldsymbol{\theta}^{o\star}_\kappa) \\ \theta^o_{MT}(t) &= \boldsymbol{\theta}^{o,\star}_{[1,\kappa]} + (\lambda' \boldsymbol{I} - \frac{\eta}{\kappa}\bar{\boldsymbol{\Sigma}}^{\leq\kappa}_x)^{(t-t^\kappa_o)}\,(\theta_{MT}(t^\kappa_0) - \boldsymbol{\theta}^{o,\star}_{[1,\kappa]})\end{aligned} \tag{29}$$

We now propose an adapted version of Lemma 5, which crucially does not require the empirical covariance to be full rank, thus guaranteeing convergence in the overparametrized regime.

Lemma 17 (Overparametrized convergence under regularization.)**.** *For any convex loss R with added norm regularizer $\lambda\|\boldsymbol{\theta}\|^2$, such that $m\boldsymbol{I} \leq \nabla^2 R(\boldsymbol{\theta}) \leq M\boldsymbol{I}$ for $m, M \in \mathbb{R}^+$ and $0 < \lambda < \eta^{-1} - M$, the convergence of (full-batch) discrete time gradient descent with learning rate η is geometric and we have:*

$$\begin{aligned}\|\theta(k) - \boldsymbol{\theta}^*\|_2 &\leq (1-\eta m')^k \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2 \qquad & R(\theta(k)) - R(\boldsymbol{\theta}^*) &\leq (1-\eta m')^{2k}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2_{\boldsymbol{\Sigma}_x} \\ \|\theta(k) - \boldsymbol{\theta}^*\|_2 &\geq (1-\eta M')^k \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2 \qquad & R(\theta(k)) - R(\boldsymbol{\theta}^*) &\geq (1-\eta M')^{2k}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2_{\boldsymbol{\Sigma}_x}\end{aligned}$$

where $m' = m + \lambda$ and $M' = M + \lambda$, $\boldsymbol{\Sigma}_x = \nabla^2 R(\boldsymbol{\theta})$ and $\boldsymbol{\theta}^\star$ is the minimizer of the regularized objective.

Proof Let us consider the ST agent case, as the proof for the MT agent is similar. By **??** we know that the GD estimate converges to the minimizer $\boldsymbol{\theta}^{o,\star}_\kappa$ exponentially fast:

$$\begin{aligned}\|\theta(k) - \boldsymbol{\theta}^{o,\star}_\kappa\|_2 &\leq (1-\eta m')^k \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^{o,\star}_\kappa\|_2 \\ \|\theta(k) - \boldsymbol{\theta}^{o,\star}_\kappa\|_2 &\geq (1-\eta M')^k \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^{o,\star}_\kappa\|_2\end{aligned}$$

The resulting estimation error is:

$$\begin{aligned}\mathcal{R}_\kappa(\theta(k)) &= (\theta(k) - \boldsymbol{\theta}^\star_\kappa)^\top \boldsymbol{\Sigma}^\kappa_x (\theta(k) - \boldsymbol{\theta}^\star_\kappa) \\ &= (\theta(k) - \boldsymbol{\theta}^{o,\star}_\kappa)^\top \boldsymbol{\Sigma}^\kappa_x (\theta(k) - \boldsymbol{\theta}^{o,\star}_\kappa) + (\boldsymbol{\theta}^{o,\star}_\kappa - \boldsymbol{\theta}^\star_\kappa)^\top \boldsymbol{\Sigma}^\kappa_x (\boldsymbol{\theta}^{o,\star}_\kappa - \boldsymbol{\theta}^\star_\kappa) \\ &\leq (1-\eta m')^{2k}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^{o,\star}_\kappa\|^2_{\boldsymbol{\Sigma}^\kappa_x} + \|\boldsymbol{\theta}^{o,\star}_\kappa - \boldsymbol{\theta}^\star_\kappa\|^2_{\boldsymbol{\Sigma}^\kappa_x}\end{aligned} \tag{30}$$

What is left to prove is that $\|\boldsymbol{\theta}^{o,\star}_\kappa - \boldsymbol{\theta}^\star_\kappa\|^2_{\boldsymbol{\Sigma}^\kappa_x} = 0$. We start by using the definition of $\boldsymbol{\theta}^{o,\star}_\kappa$:

$$\begin{aligned}\boldsymbol{\theta}^{o,\star}_\kappa - \boldsymbol{\theta}^\star_\kappa &= (\lambda I + \boldsymbol{\Sigma}^\kappa_x)^{-1}\boldsymbol{\Sigma}^\kappa_x\,\boldsymbol{\theta}^\star_\kappa - \boldsymbol{\theta}^\star_\kappa \\ &= ((\lambda I + \boldsymbol{\Sigma}^\kappa_x)^{-1}\boldsymbol{\Sigma}^\kappa_x - I)\boldsymbol{\theta}^\star_\kappa\end{aligned} \tag{31}$$

Clearly, the difference is 0 if the regularizer strength is 0:

$$\begin{aligned}&(\lambda I + \boldsymbol{\Sigma}^\kappa_x)^{-1}\boldsymbol{\Sigma}^\kappa_x - I = 0 \\ \iff\ &\boldsymbol{\Sigma}^\kappa_x = \lambda I + \boldsymbol{\Sigma}^\kappa_x \\ \iff\ &\lambda = 0\end{aligned} \tag{32}$$

In practice, $\lambda \to 0$ corresponds to the case where the population risk R has a much stronger weight than the regularization strength in the objective (up to rescaling). Therefore, we may equivalently describe t$\boldsymbol{\theta}^{o,\star}_\kappa$ as the solution to the following constrained minimization problem:

$$\min \|\boldsymbol{\theta}\|^2 \qquad s.t.\ \ R_\kappa(\boldsymbol{\theta}) = 0 \tag{33}$$

Notice that this is the precise definition of the gradient descent solution in overparametrized settings.

In the overparametrized setting the condition $R_\kappa(\boldsymbol{\theta}) = 0$ is satisfied by any $\boldsymbol{\theta} = \boldsymbol{\theta}'_\kappa + \boldsymbol{P}_x \boldsymbol{v}$, where $\boldsymbol{v}$ is any vector in the parameter space, $\boldsymbol{P}_x$ is a projection operator on the orthogonal complement of the data space, i.e. $\boldsymbol{P}_x = \boldsymbol{I} - \boldsymbol{X}_\kappa^\dagger \boldsymbol{X}_\kappa$, and $\boldsymbol{\theta}'_\kappa$ is a solution to the task, i.e. $\boldsymbol{Y}_\kappa = \boldsymbol{\theta}'_\kappa \boldsymbol{X}_\kappa$. Thus picking $\boldsymbol{\theta}_\kappa^{o,\star} \in \{\boldsymbol{\theta} \,|\, \boldsymbol{\theta} = \boldsymbol{\theta}'_\kappa + \boldsymbol{P}_x \boldsymbol{v}\}$ necessarily $\|\boldsymbol{\theta}_\kappa^{o,\star} - \boldsymbol{\theta}_\kappa^\star\|^2_{\boldsymbol{\Sigma}_x^\kappa} = 0$.

Lemma 17 bridges the regularized objective and the population risk, showing that convergence in one is necessarily linked to convergence in the other. Applying this lemma instead of Lemma 5, the results obtained in the underparametrized case can be extended to the overparametrized case without assumptions on the spectrum of the empirical covariance matrix.

A.5 Measuring the instability with the NTK

A key takeaway of our theoretical analysis is that the optimal objective depends on the instability of the sequence. Thus, it is crucial to devise efficient and pragmatic, albeit precise, measures of instability. The two methods which we mention in Section 5.2 introduce noise in the estimate of Δ_T^I due to randomness in the optimization process, and in addition they both have high computational costs.

In what follows we explore a way to get rid of this noise using the Neural Tangent Kernel (NTK) (Jacot et al., 2018). The results are still in a preliminary form and thus they are not included in the main discussion, however they demonstrate potential in this direction of research.

Consider a linearization of the network around its initialization using the Neural Tangent Kernel (NTK) (Jacot et al., 2018):

$$f^{lin}(\boldsymbol{x}; \boldsymbol{\theta}_t) = f_0(\boldsymbol{x}) + \phi(\boldsymbol{x})^\top (\boldsymbol{\theta}_t - \boldsymbol{\theta}_0) \tag{34}$$

where $\phi(\boldsymbol{x}) = \partial_{\boldsymbol{\theta}_0} f_0(\boldsymbol{x})$ are the tangent kernel features. Minimizing a quadratic cost $R = \mathbb{E}_{(x,y)} \left[f^{lin}(x; \boldsymbol{\theta}) - y\right]^2$ averaged over a dataset $(\boldsymbol{X}, \boldsymbol{Y})$ in this new convex space we get the optimal weights:

$$\boldsymbol{\theta}^\star = \boldsymbol{\theta}_0 - \phi(\boldsymbol{X})^\top K(\boldsymbol{X}, \boldsymbol{X})^{-1} \left(f_0(\boldsymbol{X}) - \boldsymbol{Y}\right) \tag{35}$$

where $K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})\phi(\boldsymbol{x}')^\top$ is the neural tangent kernel. In our continual learning setting, let $(\boldsymbol{X}_\kappa, \boldsymbol{Y}_\kappa)$ denote the dataset of task κ and $(\boldsymbol{X}_{[1,\kappa]}, \boldsymbol{Y}_{[1,\kappa]})$ the concatenation of all the datasets $1, \ldots, \kappa$. Using Equation (35), and given a common initialization $\boldsymbol{\theta}_0$, the minimizers of the ST and MT objectives for task κ are:

$$\begin{aligned} \boldsymbol{\theta}_\kappa^\star &= \boldsymbol{\theta}_0 - \phi(\boldsymbol{X}_\kappa)^\top K(\boldsymbol{X}_\kappa, \boldsymbol{X}_\kappa)^{-1} \left(f_0(\boldsymbol{X}_\kappa) - \boldsymbol{Y}_\kappa\right) \\ \boldsymbol{\theta}_{[1,\kappa]}^\star &= \boldsymbol{\theta}_0 - \phi(\boldsymbol{X}_{[1,\kappa]})^\top K(\boldsymbol{X}_{[1,\kappa]}, \boldsymbol{X}_{[1,\kappa]})^{-1} \left(f_0(\boldsymbol{X}_{[1,\kappa]}) - \boldsymbol{Y}_{[1,\kappa]}\right) \end{aligned} \tag{36}$$

The instability is the average error of the MT minimizer compared to the average error of the ST minimizer. Suppose that the ST minimizer is optimal, i.e. that $y = f_0(\boldsymbol{x}) + \phi(\boldsymbol{x})^\top \boldsymbol{\theta}_\kappa^\star$, then we can measure the instability as the average error of the MT minimizer:

$$\begin{aligned} \Delta_T^I &= \sum_{\kappa=1}^K \mathbb{E}_{(x,y)} \left[f^{lin}(\boldsymbol{x}; \boldsymbol{\theta}_{[1,\kappa]}^\star) - y\right]^2 \\ &= \sum_{\kappa=1}^K \mathbb{E}_{(x,y)} \left[\phi(\boldsymbol{x})^\top \boldsymbol{\theta}_{[1,\kappa]}^\star - \phi(\boldsymbol{x})^\top \boldsymbol{\theta}_\kappa^\star\right]^2 \\ &= \sum_{\kappa=1}^K (\boldsymbol{\theta}_{[1,\kappa]}^\star - \boldsymbol{\theta}_\kappa^\star)^\top \mathbb{E}_{\boldsymbol{x}} \left[K(\boldsymbol{x}, \boldsymbol{x})\right] (\boldsymbol{\theta}_{[1,\kappa]}^\star - \boldsymbol{\theta}_\kappa^\star) \\ &= \sum_{\kappa=1}^K \|\boldsymbol{\theta}_{[1,\kappa]}^\star - \boldsymbol{\theta}_\kappa^\star\|^2_{\Sigma_{K_x}^\kappa} \end{aligned}$$

where $\Sigma_{K_x}^\kappa = \mathbb{E}_{\boldsymbol{x}} \left[K(\boldsymbol{x}, \boldsymbol{x})\right]$ is the data covariance matrix in the kernel feature space. Let $\boldsymbol{\Xi}_\kappa = f_0(\boldsymbol{X}_\kappa) - \boldsymbol{Y}_\kappa$ denote the residuals at initialization. Then:

$$\boldsymbol{\theta}^{\star}_{[1,\kappa]} - \boldsymbol{\theta}^{\star}_{\kappa} = \phi(\boldsymbol{X}_{\kappa})^{\top} K(\boldsymbol{X}_{\kappa}, \boldsymbol{X}_{\kappa})^{-1} \boldsymbol{\Xi}_{\kappa} - \phi(\boldsymbol{X}_{[1,\kappa]})^{\top} K(\boldsymbol{X}_{[1,\kappa]}, \boldsymbol{X}_{[1,\kappa]})^{-1} \boldsymbol{\Xi}_{[1,\kappa]}$$

This quantity can be measured directly at initialization, and is exact in the infinite width limit, i.e. $\lim_{width \to \infty} \Delta_I^T \to \Delta_T^{I,\infty}$.

B Review of continual learning algorithms and the link to the multi-task objective

In this section we replicate some of the findings in the literature regarding the connection between existing CL algorithms and the multi-task objective. The discussion is mainly based on Yin et al. (2020) and Lanzillotta et al. (2024). We proceed by algorithm families, following the categorization of Parisi et al. (2019).

B.1 Regularization methods

Let Ω_{CL} be the objective of a general CL algorithm. Yin et al. (2020) consider Ω_{CL} of the form:

$$\Omega_{CL}(\boldsymbol{\theta}, \kappa) = \frac{1}{\kappa} \sum_{i=1}^{\kappa} \hat{R}_i(\boldsymbol{\theta}) \tag{37}$$

where $\hat{R}_i(\boldsymbol{\theta})$ is an approximation of $R_i(\boldsymbol{\theta})$ based on a second order Taylor expansion centered at the task minimizer $\boldsymbol{\theta}_i^\star$. Thus in practice $\Omega_{CL}(\boldsymbol{\theta}, \kappa)$ approximated the MT objective $\Omega_{MT}(\boldsymbol{\theta}, \kappa)$. In Section 4 (Yin et al., 2020) it is shown how two popular regularization based methods implement Ω_{CL}. We loosely follow their arguments here.

Elastic Weight Consolidation. Kirkpatrick et al. (2017) use the approximation

$$\hat{R}_i(\boldsymbol{\theta}) = (\boldsymbol{\theta}_i^\star - \boldsymbol{\theta})^\top F_i (\boldsymbol{\theta}_i^\star - \boldsymbol{\theta})$$

where F_i is the Fisher information matrix computed at $\boldsymbol{\theta}_i^\star$ (Equation 3, Kirkpatrick et al., 2017). For computational reasons, they approximate F_i by zeroing the off diagonal entries. If the loss function is the negative log-likelihood, and we obtained the ground truth probabilistic model, then the Fisher information matrix is equivalent to the Hessian matrix, and $\hat{R}_i(\boldsymbol{\theta})$ coincides with the second order Taylor expansion when the gradient at $\boldsymbol{\theta}_i^\star$ is null.

Kronecker factored Laplace approximation. Ritter et al. (2018) essentially refine the approximation of the Hessian matrix in EWC by considering a more sophisticated approximation of the fisher information matrix through a kronecker product rather than the diagonal approximation (Equations 5 and 9, Ritter et al., 2018).

Synaptic Intelligence Zenke et al. (2017b) explicitly introduce an approximation of the task loss of the following form (Equation 4 and 6, Zenke et al., 2017b):

$$\hat{R}_i(\boldsymbol{\theta}) = R_i(\boldsymbol{\theta}_{old}) + (\boldsymbol{\theta}_{old} - \boldsymbol{\theta})^\top \boldsymbol{\Omega}_i (\boldsymbol{\theta}_{old} - \boldsymbol{\theta}) \tag{38}$$

where $\boldsymbol{\theta}_{old}$ is the value of the model parameters after training on the previous task and $\boldsymbol{\Omega}_i$ is a diagonal matrix which is an estimate of the parameter importance for the task i. In Section 4 (Zenke et al., 2017b) they demonstrate that under certain stability assumptions $\boldsymbol{\Omega}_i$ is directly related to the Hessian computed at $\boldsymbol{\theta}_{old}$. Thus also the SI method enters the general characterization of (Yin et al., 2020), with the difference that the Taylor approximation is not centered in $\boldsymbol{\theta}_i^\star$ but in $\boldsymbol{\theta}_{old}$. Lanzillotta et al. (2024) argue that this choice results in higher performance under long learning sequences.

In general, the conjecture proposed by Yin et al. (2020) is that many second order regularization based methods implicitly build an approximation of the form Equation (37) which is based on a second order Taylor expansion. A full review of the literature is out of the scope of this work and in general infeasible, without which the conjecture cannot be proven. Nonetheless, we believe this conjecture to be true for most existing regularization methods, and we do not make any claims on the ones which escape this characterization.

B.2 Replay methods

Since Experience Replay was first introduced (Robins, 1995), several variants thereof have been proposed. In general, many replay-based algorithms optimize the same objective Ω_{CL} Equation (37), approximating the

task loss R_i through the use of a buffer:

$$\hat{R}_i(\boldsymbol{\theta}) = \sum_{(x,y)\in B_i} \ell(\boldsymbol{\theta}; x, y) \approx \sum_{(x,y)\in D_i} \ell(\boldsymbol{\theta}; x, y) \tag{39}$$

Importantly, often the samples from the buffer have an overall lower weight than the sample from the current task, e.g. by taking a gradient step on each. Thus, more accurately we say that many replay methods optimize the following objective:

$$\Omega_{rep}(\boldsymbol{\theta}, \kappa) = \frac{1}{\kappa} \sum_{i=1}^{\kappa} \alpha_i \, \hat{R}_i(\boldsymbol{\theta}) \tag{40}$$

where the task weight α_i is determined by the specific implementation of the algorithm. Our analysis of the MT objective can be easily extended to weighted average objectives, and we believe this conceptual framework to be an essential contribution of this work. In general, we demonstrate how to evaluate the optimality of any objective against a very simple baseline.

Next, we discuss other famous algorithms which belong to the replay category yet do not fall under the characterization of Equation (40). In doing so we mostly follow the arguments of Lanzillotta et al. (2024).

Orthogonal Gradient Descent. Orthogonal gradient descent (OGD) enforces orthogonality between the parameter update and the previous tasks output gradients (which are stored in the replay buffer). In order to see the connection to multi-task learning we must consider gradient-based updates. For an MT objective the gradients take the form:

$$\partial_{\boldsymbol{\theta}} \Omega_{MT}(\boldsymbol{\theta}, \kappa) = \frac{1}{\kappa} \sum_{i=1}^{\kappa} \partial_{\boldsymbol{\theta}} R_i(\boldsymbol{\theta}) \tag{41}$$

By a first order Taylor expansion, updating the parameters is the direction $-\partial_{\boldsymbol{\theta}} \Omega_{MT}(\boldsymbol{\theta}, \kappa)$ should decrease the objective by:

$$\Omega_{MT}(\boldsymbol{\theta}', \kappa) \approx \Omega_{MT}(\boldsymbol{\theta}, \kappa) - \eta \, \|\partial_{\boldsymbol{\theta}} \Omega_{MT}(\boldsymbol{\theta}, \kappa)\|^2 \tag{42}$$

The OGD condition enforcing orthogonality between the parameter update and the previous tasks output gradients instead modifies the MT loss as follows:

$$\Omega_{MT}(\boldsymbol{\theta}', \kappa) \approx \Omega_{MT}(\boldsymbol{\theta}, \kappa) - \eta \, \beta \| {}^1\!/\!_\kappa \, \partial_{\boldsymbol{\theta}} R_\kappa \|^2 \tag{43}$$

where $\beta = \cos(\partial_{\boldsymbol{\theta}} R_\kappa, \boldsymbol{\theta}' - \boldsymbol{\theta})$ is the angle between the projected update and the current task gradient -which must be non negative. Thus, the MT loss is still reduced by the OGD update, although the optimization is significantly slowed down (by a factor of $\sqrt{\kappa} \|\partial_{\boldsymbol{\theta}} \Omega_{MT}\|^2 / \beta \|\partial_{\boldsymbol{\theta}} R_\kappa\|^2$). Lanzillotta et al. (2024) prove that OGD implement an optimal quadratic constraint (Theorem 5.1, Lanzillotta et al., 2024), effectively minimizing the MT loss.

Gradient Episodic Memory . Gradient Episodic memory (GEM) minimizes a constrained objective where the parameter update has to be at a negative angle with the gradient of the previous task losses, i.e.:

$$\langle \partial_{\boldsymbol{\theta}} R_i, \boldsymbol{\theta}' - \boldsymbol{\theta} \rangle \leq 0 \tag{44}$$

The connection to the MT objective is similar to what we have seen for OGD. Simply considering a first order Taylor expansion of the MT objective we approximate its change due to the parameter update by:

$$\Omega_{MT}(\boldsymbol{\theta}', \kappa) \approx \Omega_{MT}(\boldsymbol{\theta}, \kappa) + \eta \sum_{i=1}^{\kappa} \beta_i \| {}^1\!/\!_\kappa \; \partial_{\boldsymbol{\theta}} R_i \|^2 \tag{45}$$

where $\beta_i = \langle \partial_{\boldsymbol{\theta}} R_i, \boldsymbol{\theta}' - \boldsymbol{\theta} \rangle$. Thus applying the GEM condition we know that the update reduces the MT objective.

B.3 Dynamic architecture methods

Finally, we consider the set of dynamic architecture methods (e.g. Zhou et al., 2012; Rusu et al., 2016; Mallya & Lazebnik, 2018). Generally, these methods use new units or new parameters for each task, freezing the parameters where learning already happened. Effectively, one can formalize this considering a partition of the full set of parameters $S = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_p\}$ in subsets $S_1, \dots, S_K$ and enforcing the condition $(\boldsymbol{\theta}' - \boldsymbol{\theta})[S_i] = 0 \; \forall \, i \neq \kappa$ ($(\boldsymbol{\theta}' - \boldsymbol{\theta})$ is the vector of parameter update during task κ) and $\partial_{S_j} R_i(\boldsymbol{\theta}) = 0$ for all $j > i$ (Section 5, Lanzillotta et al., 2024).

To see the effect of this update strategy let's look at the angle of the update with the gradients of the MT objective:

$$
\begin{aligned}
\langle \, \boldsymbol{\theta}' - \boldsymbol{\theta}, \, \partial_{\boldsymbol{\theta}} \Omega_{MT}(\boldsymbol{\theta}, \kappa) \, \rangle &= \frac{1}{\kappa} \sum_{i=1}^{\kappa} \langle \boldsymbol{\theta}' - \boldsymbol{\theta}, \partial_{\boldsymbol{\theta}} R_i(\boldsymbol{\theta}) \rangle \\
&= \frac{1}{\kappa} \sum_{i=1}^{\kappa} \sum_{j=1}^{K} \langle (\boldsymbol{\theta}' - \boldsymbol{\theta})[S_j], \partial_{S_j} R_i(\boldsymbol{\theta}) \rangle \\
&= \frac{1}{\kappa} \sum_{i=1}^{\kappa} \langle (\boldsymbol{\theta}' - \boldsymbol{\theta})[S_\kappa], \partial_{S_\kappa} R_i(\boldsymbol{\theta}) \rangle \quad \text{(first condition)} \\
&= \frac{1}{\kappa} \langle (\boldsymbol{\theta}' - \boldsymbol{\theta})[S_\kappa], \partial_{S_\kappa} R_\kappa(\boldsymbol{\theta}) \rangle \quad \text{(second condition)}
\end{aligned}
\tag{46}
$$

The parameter update is typically a gradient-based update on the current loss (and satisfying the above conditions). Therefore we know that $\langle (\boldsymbol{\theta}' - \boldsymbol{\theta})[S_\kappa], \partial_{S_\kappa} R_\kappa(\boldsymbol{\theta}) \rangle < 0$ and thus in general $\langle \boldsymbol{\theta}' - \boldsymbol{\theta}, \, \partial_{\boldsymbol{\theta}} \Omega_{MT}(\boldsymbol{\theta}, \kappa) \rangle < 0$, which - by a first order Taylor expansion argument - results in a reduction in the MT objective.

Assuming that the optimization on each task is run to convergence, the final parameters at the end of each task are (local) minima of the task loss: $\boldsymbol{\theta}_\kappa^{end}[S_\kappa] = \arg\min_{\boldsymbol{\theta}[S_\kappa]} \{R_\kappa(\boldsymbol{\theta})\}$. Thus, after the entire sequence of tasks has been learned the model parameters $\boldsymbol{\theta}^{end}$ will satisfy the following conditions:

$$
\begin{cases}
\boldsymbol{\theta}^{end}[S_1] = \arg\min_{\boldsymbol{\theta}[S_1]} \{R_1(\boldsymbol{\theta})\} \\
\dots \\
\boldsymbol{\theta}^{end}[S_K] = \arg\min_{\boldsymbol{\theta}[S_K]} \{R_K(\boldsymbol{\theta})\}
\end{cases}
$$

Thus effectively this class of methods assign a different subnetwork to each task, optimizing the tasks in isolation. Partitioning the network capacity compromises the performance on the task -which could be higher if the whole network were to be used- but it avoids forgetting. Under a capacity constraint for each task, these methods minimize the MT loss, assuming the optimization converges to a minima for each task. To see why simply notice that $\min_{\boldsymbol{\theta} \in S} R_1(\boldsymbol{\theta}) + R_2(\boldsymbol{\theta}) \leq \min_{\boldsymbol{\theta} \in S} R_1(\boldsymbol{\theta}) + \min_{\boldsymbol{\theta} \in S} R_2(\boldsymbol{\theta})$.

C Empirical Setup

C.1 Benchmarks, Networks and General configuration

Table 5: Supervised Learning benchmark statistics

Benchmark	K	**Input Size**	**Classes**	N_κ
CLEAR	10	224x224	100	109M
MD10	5	224x224	30	$\in [1480, 27750]$
P/S CIFAR10	10	32x32	10	10K

Table 6: MULTIDATASET datasets statistics

Dataset	**Classes**	N_κ
StanfordCars	196	1523
FGVCAircraft	100	2467
DTD	47	1480
Food101	101	27750
OxfordPet	37	3680

C.1.1 CLEAR

The CLEAR dataset (Lin et al., 2021), a collection of images of 10 different classes spanning the years 2004-2014. We split the collection into 10 tasks, one for each year. The tasks are organised in their natural temporal ordering, i.e., by increasing year. All the input images are resized to 224x224 squares and normalised by subtracting the mean $\mu = [0.485, 0.456, 0.406]$ and dividing by $\mathbf{\Sigma} = [0.229, 0.224, 0.225]$.

C.1.2 MULTIDATASET (MD5)

The MULTIDATASET benchmark consists of a sequence of 5 different open source classification datasets, with no semantic overlap between them. In particular, the tasks consists in classification of automobile models (Krause et al., 2013), aircraft models (Maji et al., 2013), textures (Cimpoi et al., 2014), dishes (Bossard et al., 2014) and pets (Parkhi et al.). Each dataset has originally a different number of classes, samples and a different input size - see Table 6. To avoid introducing biases in the models, we standardize all tasks to have only 30 classes, and we use the same batch size and amount of update steps in each task, regardless of the original dataset size. All the input images are resized to 224x224 squares and normalised by subtracting the mean $\mu = [0.485, 0.456, 0.406]$ and dividing by $\mathbf{\Sigma} = [0.229, 0.224, 0.225]$. Additionally, the training dataset samples are augmented with random crops, random horizontal flips and random rotations of 15 degrees.

C.1.3 Permuted CIFAR 10

Permuted CIFAR 10 is a benchmark built from the CIFAR 10 dataset Krizhevsky & Hinton (2009), applying fixed random permutations to the images in the dataset. We use two different permutation sizes in all experiments, namely 16 and 32. The size of the permutation measures one length of the square box of pixels which will be permuted, centered at the center of the image (see Figure 8 and Figure 9 for examples). We refer to the respective benchmarks as 'CIFAR10 Permuted - 16' (PC-16) and 'CIFAR10 Permuted - 32' (PC-32). All the input images are normalised by subtracting the mean $\mu = [0.507, 0.486, 0.441]$ and dividing by $\boldsymbol{\sigma} = [0.267, 0.256, 0.276]$.

Figure 6: Samples from CLEAR benchmark. Each column corresponds to a different task.

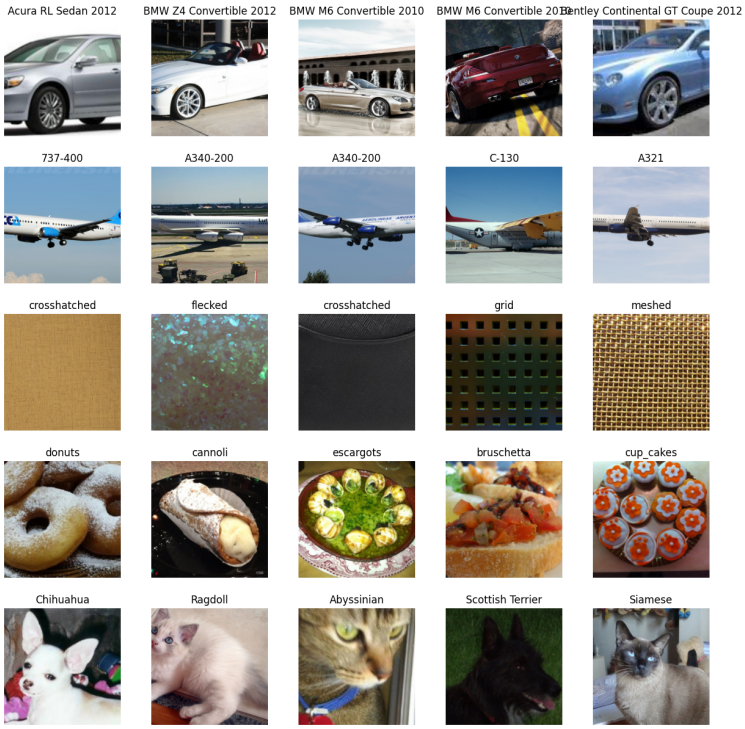

Figure 7: Samples from the MD5 benchmark. Each row corresponds to a different task.

C.1.4 Shuffled CIFAR 10

Shuffled CIFAR 10 is a second benchmark built from the CIFAR 10 dataset Krizhevsky & Hinton (2009), applying label resampling to a set of samples *fixed at the beginning of training*. We change the proportion of *shuffled labels* between 10%, 30% and 50% . In order to guarantee that this proportion of labels is *wrong* we do not allow to draw the true label when reshuffling. We refer to the respective benchmarks as 'CIFAR10 Shuffled - 10%' (SC-10), 'CIFAR10 Shuffled - 30%' (SC-30) and 'CIFAR10 Shuffled - 50%'

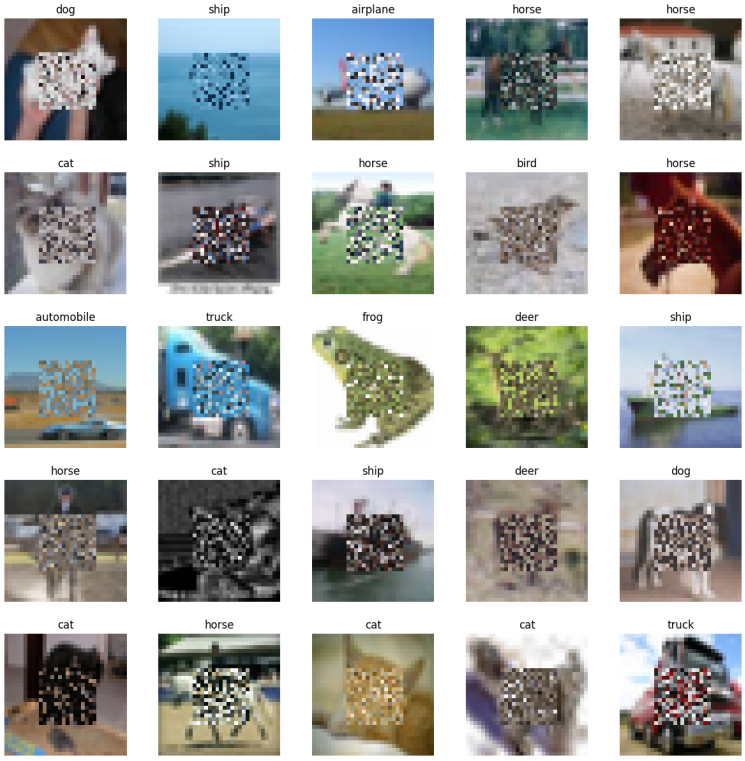

Figure 8: Samples from a Permuted CIFAR10 - 16 task.

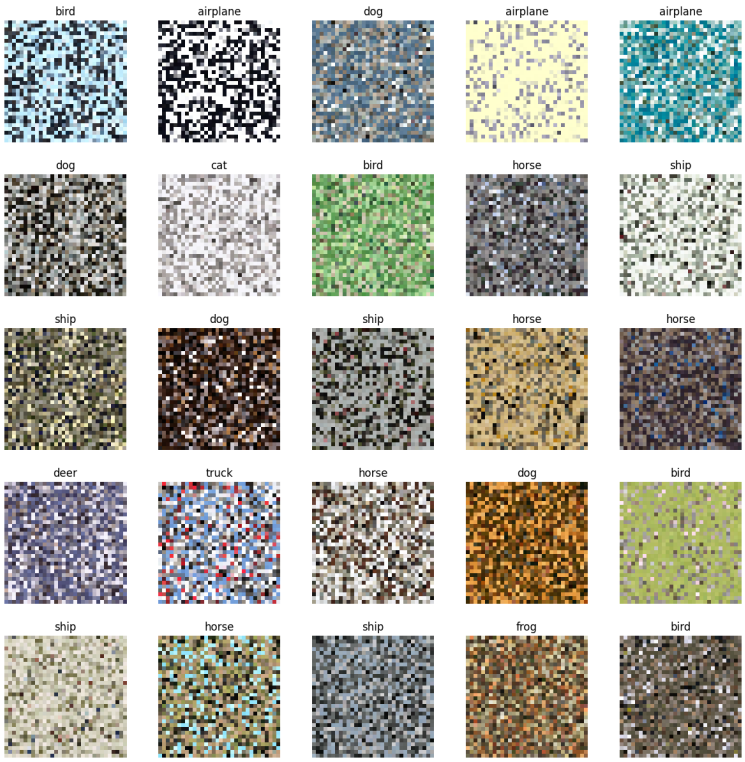

Figure 9: Samples from a Permuted CIFAR10 - 32 task.

(SC-50). All the input images are normalised by subtracting the mean $\mu = [0.507, 0.486, 0.441]$ and dividing by $\boldsymbol{\sigma} = [0.267, 0.256, 0.276]$. Importantly, when shuffling labels in the training we expect the test error to suffer.

C.1.5 Meta-World

Meta-World is a collection of 50 distinct robotic manipulation tasks simulated in the MuJoCo physics engine. Each task involves controlling a robotic arm to interact with objects in its environment, such as pushing, picking, placing, opening drawers, or pressing buttons. The tasks are designed to test a range of skills and are suitable for evaluating both single-task and multi-task learning agents.

Each observation includes the robot's joint positions, velocities, and positions of relevant objects in the environment. For the multi-task agent, the observation is augmented with a task identifier.

Actions are continuous control signals sent to the robot's joints. Actions are sampled from a normal distribution defined by the policy network outputs. Log probabilities and entropies are computed to facilitate the learning process. Generalized Advantage Estimation (GAE) (Schulman et al., 2015) is utilized to compute advantages and target values for training.

C.2 Training procedures & Networks

C.2.1 Supervised Learning Experiments

All supervised learning agents consists of a network, an optimizer and a scheduler. In all supervised learning experiments the network is a residual network, *RN18* with the final linear head size being the number of classes in each task (100 for CLEAR, 30 for MD5 and 10 for PC). The final head is shared among all the tasks. See Table 10 and Table 9 for the network and optimization hyperparameters. The ST and MT agents are trained for the same number of steps h with the same batch size per step.

Single-Task Agent. Given a sequence of K tasks the ST agent is trained to minimize a given loss function on the current task training data. The optimizer chosen is stochastic gradient descent. The ST agent network and optimizers are reset at the beginning of every task.

Multi-Task Agent. Given a sequence of K tasks the MT agent is trained to minimize a given loss function on the union of all the observed tasks training data, including the current task data. The optimizer chosen is stochastic gradient descent.

Replay. In order to ensure comparability with the ST agent, the Experience Replay and Selective Replay agents are trained with the same batch size, which is equally partitioned between the current task data and the buffer data. The buffer is randomly filled at the end of each task with the data from the task. For all the agents we use a replay buffer of 500, meaning that we store 500 samples of each task in the buffer. While the ER agent is trained in a similar fashion as the MT agent, to minimize the loss on the the observed tasks, the SR agent resets the buffer when the instability increases.

C.2.2 Reinforcement Learning Experiments

In our reinforcement learning experiments, we aim to compare the performance of single-task and multi-task agents using Proximal Policy Optimization (PPO) Schulman et al. (2017) on the Meta-World benchmark Yu et al. (2020). PPO is a widely used policy-gradient method known for its stability and reliability in training deep reinforcement learning agents.

Single-Task Agent. For each task, we train a separate PPO agent with its own policy and value networks. The policy network is a multi-layer perceptron (MLP) consisting of two hidden layers with 128 units each and ReLU activation functions. The output layer produces the mean and standard deviation for a Gaussian action distribution. The value network shares the same architecture but outputs a scalar value estimate.

Multi-Task Agent. We train a single PPO agent across all selected tasks. The agent uses a shared policy network with the same architecture as the single-task agents. Multi-task agent uses a replay buffer to sample and update the PPO. The replay buffer at each task has the data from the current task and previous ones.

The RL agents were exposed to 10 tasks from ML10 benchmark, the tasks are as following: *Reach*, *Push*, *Pick & Place*, *Door Open*, *Drawer Close*, *Button Press*, *Peg Insert Side*, *Window Open*, *Sweep*, and *Basketball*. The order is preserved while running experiments for various task durations.

C.2.3 Hyperparameters

C.2.4 Supervised Learning Benchmarks

The key hyperparameters which are tuned separately for each agent and benchmark are the learning rate, the batch size and the weight decay. The optimizer and scheduler are fixed across all supervised learning experiments. We employ SGD with a cosine annealing of the learning rate every h time step, which means the learning rate is annealed over the course of each task and increased again at the beginning of the next task in order to allow the network to minimize the changing objective.

Table 7: Fixed HyperParameters for Supervised Learning Experiments. Note that the batch size has been tuned but the optimal batch size is the same for all agents and benchmarks

HP	**Value**
Momentum	0.9
Scheduler	Cosine Annealing
Batch Size	256
Optimizer	SGD

Table 8: Tuned HyperParameters for Supervised Learning Experiments

Dataset	**Agent**	**Network**	**LR**	**Weight Decay**
CLEAR	ST	RN18	0.1	3×10^{-4}
CLEAR	MT	RN18	0.1	1×10^{-4}
C10 mixed	MT	RN18	0.075	7×10^{-4}
C10 mixed	ST	RN18	0.055	8×10^{-4}
PC-16	MT	RN18	0.075	7×10^{-4}
PC-16	ST	RN18	0.055	8×10^{-4}
PC-32	MT	RN18	0.074	1×10^{-3}
PC-32	ST	RN18	0.06	1×10^{-3}
MD5	MT	RN18	0.08	6×10^{-4}
MD5	ST	RN18	0.089	9×10^{-4}

C.2.5 ML10

We use the same set of hyperparameters for both agents where applicable to ensure a fair comparison. Key hyperparameters include a learning rate, a discount factor, and a clip ratio, enthropy coefficient, and lambda for GAE for training the PPO. Both agents are trained using the Adam optimizer. For the multi-task agent, gradients are calculated for each task and aggregated before the update step to ensure balanced learning across tasks. We train two type of agents, single-task and multi-task. The single task agent only receives

observation from the current task, multi-task agent receives data from the current task as well as previous ones. We train these two type of agents for different task duration. For the RL experiments, we picked 50 and 500 episodes. All the results shown in tables Each episode is 500 time steps.

We use a batch size of 256 for updating the policy in ST agent and 512 for MT agent. In the multi-task setting, the batch is composed of an equal number of timesteps from each environment to prevent task imbalance.

Table 9: Optimal Configurations for ML10 for single-task and multi task agents

Parameter	**ST**	**MT**
Batch Size	256	512
Entropy Coefficient	0.02	0.02
Learning Rate (Value)	1×10^{-3}	1×10^{-3}
Learning Rate (Policy)	1×10^{-4}	1×10^{-5}
Lambda for GAE	0.95	0.8

Table 10: Software, hardware, and libraries used in the experiments

	Python	**MuJoCo**	**Meta-World**	**Gymnasium**	**GPU**	**RAM**
Version	3.8	2.3.2	2.0.0	$\geq$ 1.0.0	*NVIDIA RTX 2080 Ti*	128 GB

C.3 Additional Empirical results

The table presents the task average reward for single-task and multi-task agents across various environments in the ML10 benchmark. Single-task agents generally perform better in most tasks, as seen in environments like *SawyerReachEnvV2* and *SawyerDrawerCloseEnvV2*. However, there are cases, such as *SawyerPushEnvV2*, where the multi-task agent outperforms the single-task agent.

Table 11: Task average reward over 500 episodes for single-task and multi-task agents in ML10.

Task	reward$_{ST}$ (↑)	reward$_{MT}$ (↑)
SawyerReachEnvV2	$1.9130_{\pm 1.8771}$	$1.8040_{\pm 1.6159}$
SawyerPushEnvV2	$0.0349_{\pm 0.0456}$	$0.0760_{\pm 0.3693}$
SawyerPickPlaceEnvV2	$0.0079_{\pm 0.0055}$	$0.0112_{\pm 0.0140}$
SawyerDoorEnvV2	$0.5736_{\pm 0.2662}$	$0.5052_{\pm 0.2829}$
SawyerDrawerCloseEnvV2	$2.8774_{\pm 3.8648}$	$2.2409_{\pm 3.4909}$
SawyerButtonPressTopdownEnvV2	$0.4940_{\pm 0.4084}$	$0.3761_{\pm 0.2232}$
SawyerPegInsertionSideEnvV2	$0.0105_{\pm 0.0066}$	$0.0124_{\pm 0.0088}$
SawyerWindowOpenEnvV2	$0.4924_{\pm 0.4618}$	$0.4294_{\pm 0.3024}$
SawyerBasketballEnvV2	$0.0114_{\pm 0.0082}$	$0.0128_{\pm 0.0080}$

D Limitations

Our work is a small step towards understanding and formalizing the existing assumptions in continual learning. The theoretical framework is limited to the convex case with linear models. Nevertheless we argue that theory is useful as long as it is predictive of behavior, even if it does not describe the actual setup.

Additionally, the proposed formalism is not descriptive enough to address complex shifts in the data distribution, as it relies on the assumption that there are contiguous time intervals (called tasks) where the data distribution is locally i.i.d..

Another limitation of the work is the choice of the MT agent, which is an abstract and unattainable rendition of continual learning algorithms. For example, experience replay may be biased to the current task, or simply fail to represent the past data distributions due to the limited buffer. In order to evaluate the exact degree of optimality of any specific algorithm the multitask objective should be modified in accordance with the algorithm.

Finally, the selective replay algorithm only provides a proof-of-concept idea of how the structure of the non-stationarity can be exploited by continual learning algorithms. In practice, one would need to estimate the sequence instability in order to run it. We believe that the online estimate of a sequence instability for the design of adaptive objectives is a promising avenue of future work.