This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

Schedule-Robust Continual Learning

Ruohan Wang[1], Marco Ciccone[2], Massimiliano Pontil[3,4], and Carlo Ciliberto[3]

Abstract—Continual learning (CL) tackles a fundamental challenge in machine learning, aiming to continuously learn novel data from non-stationary data streams while mitigating forgetting of previously learned data. Although existing CL algorithms have introduced various practical techniques for combating forgetting, little attention has been devoted to studying how data schedules – which dictate how the sample distribution of a data stream evolves over time – affect the CL problem. Empirically, most CL methods are susceptible to schedule changes: they exhibit markedly lower accuracy when dealing with more "difficult" schedules over the same underlying training data. In practical scenarios, data schedules are often unknown and a key challenge is thus to design CL methods that are robust to diverse schedules to ensure model reliability. In this work, we introduce the novel concept of *schedule robustness* for CL and propose Schedule-Robust Continual Learning (SCROLL), a strong baseline satisfying this desirable property. SCROLL trains a linear classifier on a suitably pre-trained representation, followed by model adaptation using replay data only. We connect SCROLL to a meta-learning formulation of CL with provable guarantees on schedule robustness. Empirically, the proposed method significantly outperforms existing CL methods and we provide extensive ablations to highlight its properties.

Index Terms—Continual Learning, Lifelong Learning, Meta-Learning, Representation Learning.

1 Introduction

The ability to continually absorb new knowledge while retaining and updating the existing one is a hallmark of natural intelligence. Realizing such ability in machines is precisely the goal of continual learning (CL). Ideally, CL algorithms should learn from a never-ending and non-stationary stream of data without catastrophic forgetting [1, 2]. Conceptually, a data stream comprises an *underlying dataset* and a *schedule*, wherein the schedule determines how the dataset is presented to CL algorithms. Existing works have investigated both *task-based* [3] and *task-free* [4] schedules. Task-based schedules split a dataset into sequential tasks composed of disjoint classes, while task-free schedules require CL algorithms to learn *online* from small batches, featuring dynamic data distribution over time [5, 6]. Fig. 1a depicts different schedules over the same classification dataset.

Although schedules play a central role in CL, understanding how different data schedules affect CL performance is understudied. Only few works [7, 8] observed that model performance can fluctuate heavily with different schedules, even when the underlying training data remains unchanged. As an illustrative example, Fig. 1b reports the performance of L2P [9], a recent CL method, under three distinct schedules applied to the CIFAR100 dataset. These schedules simply divide CIFAR100 into varying numbers of classes and present them sequentially to the learner. However, they result in substantially different model accuracy: schedules comprising larger numbers of tasks lead to more forgetting since the learner must retain knowledge on more tasks while learning a new one. This pattern is consistently observed across most CL algorithms.

In this work, we demonstrate that the performance of most CL methods is biased towards schedules featuring fewer tasks, akin to the example from Fig. 1b. This presents a significant challenge for deploying CL methods in real-world applications, where CL methods may have to perform on novel data streams with unknown schedules. Since CL methods are typically evaluated on only a handful of "standard" schedules, their performance on diverse schedules is unknown, posing risks to model safety and reliability.

The above issue calls for an ideal notion of *schedule robustness*: CL performance should be minimally affected by different schedules applied to the same underlying data. However, an exponential number of possible schedules exists for any dataset, making it impossible to evaluate (or optimize) CL algorithms against all of them. Therefore, CL methods must be designed with schedule robustness as a primary objective, ideally with theoretical guarantees.

In this work, we formalize the theoretical notion of schedule robustness for CL and a corresponding metric. We then present **SC**hedule-**Ro**bust continua**L** **L**earning (SCROLL), a strong baseline designed to satisfy this desirable property. SCROLL trains a linear classifier on top of a pre-trained representation – with a random features projection layer to account for possible non-linear relations in the data – followed by representation update using only experience replay (ER) data [10]. By design, SCROLL is online [11] and capable of tackling arbitrary schedules.

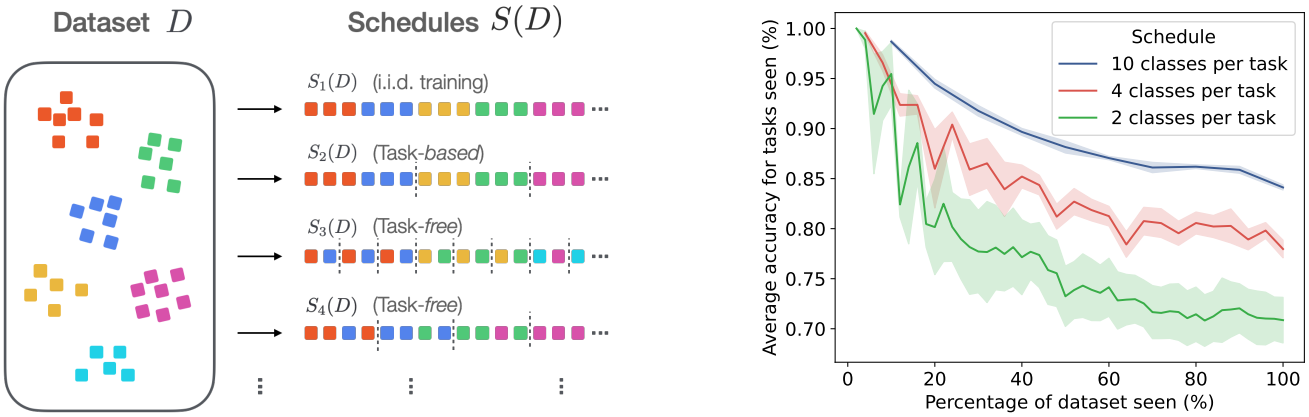
We prove that SCROLL is schedule-robust for all task-based schedules comprising disjoint tasks. Our result applies to a wide range of scenarios, including **tasks of different sizes and arbitrary class ordering**. In addition, we show that our method is principally motivated by a meta-learning formulation of CL, where we connect using pre-trained networks for CL [e.g. 9, 12, 13] as meta-learned priors. The theoretical analysis also justifies leveraging linear models for continuously assimilating novel data.

In the experiments, we introduce a diverse range of schedules for evaluating CL models. By quantifying the

[1]*Institute for Infocomm Research (I²R), Agency for Science, Technology and Research (A*STAR) Singapore*
[2]*Politecnico di Torino, DAUIN, Torino, Italy*
[3]*Centre for Artificial Intelligence, Computer Science Department, University College London, London, United Kingdom*
[4]*CSML, Istituto Italiano di Tecnologia, Genova, Italy*

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868



(a) **Example schedules over a dataset** (b) **CL accuracy on CIFAR100 for 3 schedules**

Fig. 1: **(a)** Different schedules for a classification dataset. Colored squares are samples from different classes. Vertical lines are task or batch boundaries. **(b)** L2P's CL performance on CIFAR100 under 3 schedules, which splits the dataset into a varying number of tasks.

schedule robustness of various CL models, we demonstrate that most existing CL methods are not schedule-robust with considerable risks to model reliability. In contrast, SCROLL performs consistently across all evaluated schedules, and significantly outperforms existing methods in test accuracy (over 10% in some settings). Lastly, we perform extensive ablation studies to highlight SCROLL's properties. We summarize our main contributions below:

- We introduce the notion of *schedule robustness* for CL and a new metric to measure it, highlighting its importance for practical applications of CL.
- We propose SCROLL, a simple yet effective baseline designed to be schedule-robust. Technically, we contribute a principled random projection technique for mitigating the distributional shift between pre-trained representations and novel classes encountered during CL. In addition, we introduce an ER strategy that retains schedule robustness and mitigates forgetting, specifically designed for leveraging pre-trained networks.
- We prove that SCROLL is schedule-robust (more precisely schedule-invariant) for task-based schedules. Additionally, we show how pre-training relates to learning a CL prior via meta-learning.
- We evaluate existing CL methods in the context of schedule robustness and their performance under diverse schedules in the experiments, showing that the majority of them are susceptible to changing dataset schedule. We demonstrate that SCROLL performs consistently across all evaluated schedules and significantly outperforms existing approaches.

2 Background

We formalize CL as the problem of learning from non-stationary data sequences. A supervised learning dataset $D = (x_i, y_i)_{i=1}^N$ is streamed to a continual learner as a sequence of batches. The resulting sequence $S(D)$ is characterized by a *schedule* S, which governs how D is streamed (see Fig. 1a for an illustration).

Formally, a schedule $S = \beta \circ \sigma$ comprises two functions: a permutation $\sigma(D)$ that applies a specific ordering to D, and a map $\beta(D) = (B_t)_{t=1}^T$ that divides the dataset into a sequence of batches in the form $B_t = (x_i, y_i)_{i=k_t}^{k_{t+1}}$. The indices k_t represent the batch boundaries (illustrated as vertical dashed lines in Fig. 1a). Intuitively, σ determines the order in which $(x, y) \in D$ are observed, while β dictates the number of samples observed at a given time. We use batch B_t to denote individual tasks in *task-based* settings, and mini-batches in *task-free* settings respectively. For instance, $S_2(D)$ in Fig. 1a depicts the task-based schedule that divides D into tasks each containing *all* samples from two classes, while $S_3(D)$ and $S_4(D)$ show two possible schedules where each batch contains a few samples from two classes. Here, $S_1(D)$ represents the trivial schedule where the entire dataset is observed as a single batch (no dashed separation lines), corresponding to the standard i.i.d learning setting. This is the most favorable schedule and we use it as a performance upper bound in our experiments.

2.1 Continual Learning

At time t, a CL algorithm updates the model $f_t : \mathcal{X} \to \mathcal{Y}$ to incorporate the new data batch B_t. To mitigate forgetting, many CL methods also assume access to a *replay buffer* M_t, which contains previously observed samples to be reused during training. Denoting $\text{Alg}(\cdot)$ as the incremental algorithm to process each batch B_t, the update step can be formalized as

$$(f_t,\ M_t) = \text{Alg}(B_t, f_{t-1}, M_{t-1}), \tag{1}$$

where the current model f_{t-1} is updated to f_t, and the replay buffer M_{t-1} is also revised to potentially include new data from B_t. At test time, $\text{Alg}(\cdot)$ is evaluated on a distribution π_D over samples (x, y) that share the same class labels y with those in D. In particular, we measure the performance of the final model f_T obtained recursively via (1) starting from an initial model f_0. This is measured by the generalization error

$$\mathcal{L}(S(D), f_0, \text{Alg}) = \mathbb{E}_{(x,y)\sim\pi_D} \ell(f_T(x), y). \tag{2}$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

Model Update. Most CL methods learn f_t by optimizing the following:

$$\min_f \underbrace{\sum_{(x,y)\in B_t} \ell(f(x),y)}_{\text{current batch loss}} + \underbrace{\sum_{(x,y)\in M_{t-1}} \ell(f(x),y)}_{\text{replay loss}} + \underbrace{R(f,f_{t-1})}_{\text{regularization loss}}$$

where ℓ is a loss function, and R a regularizer. The formulation above recovers replay-based methods such as iCarl [14] and DER [15] for specific choices of ℓ and R. When $M_t{=}\varnothing$ for all t, we recover regularization-based methods [e.g., 2, 16].

Experience Replay. Replay-based methods define a *buffering strategy* for Alg$(\cdot)$ in (1), to store samples in M_t for future reuse. Common strategies include random sampling, exemplar selection [14], or more sophisticated methods like gradient-based sample selection [17]. Replay-based methods are potentially limited by memory constraints, whereby the memory buffer M_t is unable to hold at least one sample per observed class. Common solutions to the memory constraint include generative memories [18] or sample compression [19, 20]. Existing methods typically mix current and replay data for model updates [e.g., 5, 17, 21], a strategy that might lack schedule robustness (see discussion in Sec. 3.2). In this work, we introduce a novel ER strategy designed to guarantee schedule robustness.

Model Initialization. The f_0 in (2) captures the prior knowledge of the model before learning from the sequence $S(D)$. Many methods assume random initialization without prior knowledge [e.g., 2, 14, 22]. While random initialization grants generality with respect to the application domain, it can be overly restrictive and sub-optimal in practice. For instance, in Computer Vision (CV) or Natural Language Processing (NLP), robust pre-trained models are readily available to provide valuable model priors to CL algorithms. To this end, one strategy for initializing f_0 involves a pre-trained model related to the target CL domain (e.g., ImageNet pre-training for CV tasks), which typically results in a significant boost in CL performance [9, 12, 13, 23, 24]. However, [25] observed that the benefits of pre-training diminish for longer CL sequences, since model updates continually push f_t away from the initial f_0.

3 Methodology

In this section, we formalize schedule robustness and motivate its significance in CL. We then present our method SCROLL, analyze its theoretical properties in terms of schedule robustness and lastly establish a relevant connection to meta-learning.

3.1 Schedule Robustness

CL benchmarks typically consider only schedules featuring few tasks containing many classes each (e.g. 10 tasks for 100 classes in split-CIFAR100 in [12, 15]). Most existing methods have an implicit bias toward such schedules. This poses a significant issue in practice, as it is often infeasible to control (or know) the data schedule in advance.

Model performance tends to be lower than expected when the schedule diverges from those used in standard CL benchmarks. As an example, Fig. 1b illustrates this problem with L2P [9], a recent CL method. We report its performance on CIFAR100 [26] over three schedules $S_i = \beta_i \circ \sigma$, which have the same sample ordering σ but varying splits β_i. S_1 is the standard schedule streaming 10 classes per task [e.g., 9, 15, 21]. In contrast, S_2 and S_3 split the dataset into more tasks. L2P exhibits a strong bias towards schedules with fewer tasks, with over a 10% drop in performance for alternative ones. Similar performance degradation applies to most existing CL methods (see Sec. B.1 for analogous plots of different CL methods).

The results show that designing CL algorithms tailored to specific schedules – albeit not necessarily intentional – can be detrimental to model performance and reliability in practice. Therefore, we argue that CL algorithms should be *schedule-robust* and maintain *consistent* performance across different schedules. We formalize the property below,

Definition 1 (Schedule Robustness)**.** *Given a dataset D, let ρ be a distribution of possible schedules S over D. A CL algorithm $Alg(\cdot)$ is ϵ-**schedule-robust** with respect to schedules from ρ if*

$$\mathbb{V}_{S\sim\rho}\left[\mathcal{L}(S(D), f_0, Alg)\right] \leq \epsilon, \tag{3}$$

*for any dataset D, with $\mathbb{V}$ denoting the variance with respect to schedules S sampled from ρ. Moreover, an algorithm is **schedule-invariant** if $\epsilon = 0$.*

Def. 1 quantifies the robustness of a CL algorithm in terms of the smallest $\epsilon \geq 0$ that upper bounds the variance $\mathbb{V}$ of its performance across the distribution of schedules ρ. Smaller ϵ corresponds to more robust algorithms. In particular, schedule-invariance guarantees the algorithm performs identically across all schedules from ρ. While appealing, precisely quantifying ϵ is often intractable since the number of schedules can be exponential. In our experiments, we approximate ϵ by the empirical standard deviation of model accuracies under a finite number of schedules.

Apart from the metric proposed above, [8] also introduced *Order-normalized Performance Disparity* (OPD), a different metric for schedule robustness in task-based settings only. OPD measures how individual tasks' performance fluctuates when varying task ordering in a data stream. For both OPD and our proposed metric, lower numbers imply more robust performance across different schedules.

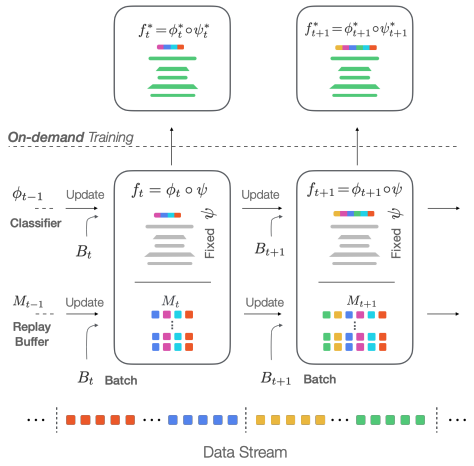
Lastly, we stress that schedule robustness is not the sole objective of a CL algorithm but must be evaluated jointly with model performance since schedule robustness could be trivially satisfied by an "algorithm" that never updates its initial model, clearly not a suitable learning strategy. In other words, while schedule robustness characterizes the reliability of a CL model, its performance remains a key goal to maximise. An ideal CL method should achieve high average performance along with high schedule robustness.

3.2 SCROLL: Schedule-Robust Continual Learning

In this section, we present a method for schedule-robust continual learning that is conceptually simple yet effective. We consider a model $f = \phi \circ \psi$, with $\psi : \mathcal{X} \to \mathbb{R}^m$ a feature map and $\phi : \mathbb{R}^m \to \mathcal{Y}$ a linear one-vs-rest classifier, that

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868



Algorithm 1 SCROLL (incremental)

Init: Buffer $M_0 = \varnothing$, data statistics $c_y^0 = 0, A_0 = 0$
Input: Embedding ψ, current statistics c_y^{t-1}, A_{t-1}, current buffer M_{t-1}, next batch B_t,

$c_y^t, A_t = \text{UpdateStat}(c_y^{t-1}, A_{t-1})$ (Eq.(5))
$\phi_t = \text{RidgeRegression}(c_y^t, A_t)$ (Eq.(5))
$f_t = \phi_t \circ \psi$
$M_t = \text{SelectExemplar}(M_{t-1}, B_t, \psi)$ (Sec. 3.2)
If t mod $UpdateFreq$ == 0 or On-demand:
  $f_t^* = \text{UPDATEREPR}(f_t, M_t)$ (Sec. 3.2)

Return c_y^t, A_t, M_t, f_t and f_t^* (optional)

Fig. 2: SCROLL update: each batch B_t updates the replay buffer M_t and classifier ϕ_t. **If** a representation update is required (*on-demand*), SCROLL further trains f_t over M_t, producing f_t^*. **Left.** Figure depicting the update process. **Right.** Corresponding algorithm.

is, $\phi(z) = \arg\max_y \ w_y^\top z$ for $z \in \mathbb{R}^m$, parametrized by a matrix $W = [w_1, \dots, w_K] \in \mathbb{R}^{m \times K}$, where K is the number of classes that have been learned so far. Given this model, SCROLL comprises four phases: 1) pre-training an embedding model ψ_0; 2) adding a random features projection layer ψ_{RF} to ψ_0 yielding the embedding $\psi = \psi_{\text{RF}} \circ \psi_0$; 3) updating the linear classifier ϕ_t online while keeping ψ fixed, and 4) updating both ψ_0 and ϕ_t *on-demand* via experience replay. We detail each phase below and discuss in Sec. 4 how their combination achieves schedule robustness.

1. Pre-training. Pre-training aims to find an embedding model $\psi_0 : \mathcal{X} \to \mathbb{R}^d$ that provides a suitable "backbone" as CL prior. To this end, we train a multi-class (MC) classifier $f_{\text{MC}} = \phi \circ \psi_{\text{MC}}$ and use the embedding model $\psi_0 = \psi_{\text{MC}}$ to initialize SCROLL. For instance, our CIFAR experiments in Sec. 5.1 use a ψ_0 pre-trained on Meta-Dataset [27].

We note that pre-training has recently gained increasing popularity in CL [6, 9, 12, 13, 28, 29] since it significantly improves test performance over random initialization. We stress that pre-training neither "bypasses" the CL problem nor makes it trivial: firstly, as demonstrated by [25] and further supported by our experiments in Sec. 5, any pre-trained CL model needs to adapt to new data to achieve competitive performance. Secondly, the knowledge contained within a pre-trained model is also susceptible to forgetting when it is updated during CL.

2. Random Features Projection. The representation ψ_0 is typically optimized for an *auxiliary* multi-class classification problem (see Sec. 4.2 for more details). While suitable for the originally learned classes with a linear classifier layer, it remains unclear whether novel classes encountered during CL will also exhibit a linear structure with respect to ψ_0 to enable efficient learning.

To tackle this issue, we propose augmenting ψ_0 with a fixed non-linear layer $\psi_{\text{RF}} : \mathbb{R}^d \to \mathbb{R}^m$. Following the random features literature [30], we implement *random Fourier features* as $\psi_{\text{RF}}(x) = \cos(Fx)$, where $F \in \mathbb{R}^{m \times d}$ has randomly sampled entries and cosine applied element-wise. Theories guarantee that as $m \to +\infty$, the inner product $\psi_{\text{RF}}(x)^\top \psi_{\text{RF}}(x') = k(x, x')$ approximates a reproducing kernel, with kernel type dependent on the sampling distribution of F [31]. This suggests using a relatively large m in practice (in our experiments, $m = 10K$). The final embedding model becomes $\psi = \psi_{\text{RF}} \circ \psi_0$. Our approach relates to and generalizes [12], which proposed random ReLU layers for CL problems.

3. Online Classifier Learning. Given the pre-trained ψ, this phase learns a classifier ϕ_t for all samples $D_t = B_1 \cup \dots \cup B_t$ observed so far. We chose Ridge Regression (RR) [32] to learn $W_t = \text{RR}(D_t, \psi)$, where W_t parameterize ϕ_t. Denoting $\Delta(y)$ the one-hot encoding for label y and λ a regularization constant, we have

$$W_t = \arg\min_W \sum_{(x,y) \in D_t} \left\| W^\top \psi(x) - \Delta(y) \right\|^2 + \lambda \|W\|^2 \quad (4)$$

We observe that the closed-form solution to (4) – which in principle needs access to the entire dataset D_t – can also be obtained online, by sequentially learning from the batches B_t via recursive least squares [32]: let A be the covariance matrix for data observed so far and c_y the sum of embeddings $\psi(x)$ for class y. Then, for any new (x', y'),

$$w_y = (A^{\text{new}} + \lambda I)^{-1} c_y^{\text{new}} \qquad \forall y \in \mathcal{Y}, \text{ s.t.}$$

$$A^{\text{new}} = A^{\text{old}} + \psi(x')\, \psi(x')^\top \quad (5)$$

$$c_y^{new} = \begin{cases} c_y^{\text{old}} + \psi(x') & \text{if } y = y' \\ c_y^{\text{old}} & \text{otherwise.} \end{cases}$$

Eq. (5) shows that it is sufficient to maintain A and c_y up-to-date to learn W_t at each step. Crucially, RR is vital for making SCROLL schedule-robust, by being invariant to data ordering (see Sec. 4).

4. Representation Update via ER. The goal of this phase is to update the model $f_t = \phi_t \circ \psi$ into $f_t^* = \phi_t^* \circ \psi_t^*$, with the underlying representation ψ updated to assimilate the novel data. A key question concerns how to update ψ effectively without forgetting.

In this context, Experience Replay (ER) methods have been empirically observed to yield significant improvement for CL [e.g., 14]. ER keeps a memory buffer of samples from

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

previous tasks/batches, that can be reused during training to mitigate forgetting. However, previous ER approaches typically 1) combine exemplars from memory with current task/batch samples to jointly train the model $f = \phi \circ \psi$ end-to-end and, 2) employ the resulting updated representation to select replay samples. This procedure is clearly schedule-dependent. In this section, we introduce a novel ER strategy specifically designed to ensure schedule robustness by departing from the standard ER cycle outlined above.

As CL data is observed, we use Exemplar Selection [14] for filling the replay buffer M_t. This strategy greedily chooses replay samples independently for each class, based on how well the mean embedding of the replay samples approximates the empirical class mean embedding. we leverage the pre-trained (and fixed) ψ for exemplar selection, which allows the process to be independent from the data schedule.

Model Updates. Given a memory buffer M_t, we then update the CL model f_t by minimizing the cross-entropy loss $\mathcal{L}(f, M_t) = \sum_{(x,y)\in M_t} \ell_{ce}(f(x), y)$ over the memory buffer M_t, warm-starting the fine-tuning from f_t to obtain f_t^*. Concretely, we leverage parameter-efficient fine-tuning [33] (PEFT) to update f_t while adapting the pre-trained model. This strategy is more resistant to overfitting given the relatively small sizes of M_t compared to all samples observed during CL. We use task-specific adapters (TSA) [34] for ResNet [35] pre-trained models, and low-rank adaptation [36] (LoRa) for vision transformers [37]. This adaptation tackles the distribution shift between the initial fixed ψ and the CL sequence observed so far.

On-demand Updates. We highlight here that SCROLL performs ER only periodically or *on-demand* to improve computational efficiency (see Alg. 1). At the extreme, we could perform ER only once, after observing the entire CL sequence. More realistically, we may perform ER at the end of each task for task-based settings, or after every K samples in a task-free setting (as illustrated in Fig. 2). At any time t, the current online model f_t and replay buffer M_t capture all the information needed for on-demand representation update.

Our proposed approach differs from standard ER in three key aspects. Firstly, standard ER mixes replay samples with the current task/batch for training. In contrast, SCROLL uses the current task/batch to update the classifier ϕ_t, followed by fine-tuning of ψ with only replay samples. Secondly, the proposed ER always starts from f_t with the pre-trained embedding model ψ, which is crucial for our theoretical analysis on schedule robustness in Sec. 4. Thirdly, our fine-tuning uses the closed-form RR solution ϕ_t as initialization for the fine-tuning process. In Sec. 6.2 we show that such initialization significantly improves fine-tuning performance.

5. SCROLL. Fig. 2 illustrates the overall method with Alg. 1 as the incremental algorithm for handling an incoming batch B_t: we first update data statistics c_y^{t-1} and A_{t-1} to incorporate B_t. The replay buffer M_{t-1} is also updated, using the embedding model ψ for exemplar selection. Lastly, we obtain f_t using the updated states c_y^t and A_t. When required, f_t is fine-tuned on M_t yielding f_t^*. We conclude by noting that both f_t and f_t^* are valid CL models for the observed data D_t. The key difference is that f_t^* has a representation better tailored to the observed data.

Computational Efficiency. SCROLL offers superior computational efficiency compared to most previous methods. The least-squares step uses Cholesky low-rank updates, requiring only $O(d^2|B_t|)$ (with a max of $O(d^3)$) operations and $O(d^2 + d|M_t|)$ memory (see [38]), with the time complexity independent of the number of tasks T. ER remains efficient through PEFT and our exemplar selection strategy that needs only per-class averages. Additionally, the on-demand nature of updates allows for adaptive computation that meets specific requirements.

4 Theoretical Properties of SCROLL

In this section, we discuss SCROLL's key properties and how they contribute to schedule robustness. We also connect SCROLL to meta-learning, offering a principled interpretation of pre-training in CL. Proof details for theoretical results are deferred to Sec. A.

4.1 Schedule Robustness

We first show that the model f_t at each time step during Phase 2 of SCROLL is schedule-invariant.

Proposition 1. *For any embedding ψ, the f_t learned via ridge regression (RR) according to (5) on D_t is deterministic and schedule-invariant.*

The proposition shows the benefits of using RR given a fixed representation: f_t trades off model expressiveness for schedule invariance. Specifically, our baseline leverages the invariance of RR to construct a deterministic f_t for all possible schedules. Prop. 1 also implies that the f_t learned under any schedule is equivalent to the model trained over the entire dataset. This, in turn, implies that RR mitigates forgetting since the model learned incrementally is identical to the one learned over the entire dataset. We recall that f_t is a valid CL model for all observed data D_t, and thus conclude that f_t **is schedule-invariant for all possible schedules over any given dataset.**

We remark that Prop. 1 holds also for other learning strategies, such as nearest centroid classifier [39] (NCC) and linear discriminant analysis [40] (LDA). Several concurrent works also leverage such classifiers for replay-free CL, including RanPAC [12] that uses RR, and FeCam [29] that uses LDA. For SCROLL we adopted RR given its superior empirical performance.

Proposition 2. *Let D_t be of the form $D_t = \{(X_k, y_k)\}_{k=1}^K$, with (X_k, y_k) containing all samples of class y_k for $k = 1, \ldots, K$. The f_t^* learned via SCROLL is invariant to schedules S such that, for any label y_k, there exists one batch B_t containing (X_k, y_k).*

Prop. 2 applies to replay-based f_t^*, for any task-based schedule where all tasks comprise disjoint classes, including **tasks of different sizes and arbitrary ordering of classes or tasks**. Proving Prop. 2 relies on the observation that

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

exemplar selection returns a deterministic M_t. Training the deterministic f_t, from Prop. 1, on M_t yields deterministic f_t^*, which again leads to schedule-invariance.

For all other schedules where the hypotheses of Prop. 2 do not hold (such as task-free schedules $S_3(D)$ and $S_4(D)$ in Fig. 1a), we note that exemplar selection still attains similar class distribution across different schedules, especially for larger buffer sizes (see Sec. 6.3 for details). Training f_t on similar M_t still leads to f_t^* of comparable performance.

From the above analysis, we conclude that SCROLL is schedule-robust with small performance variation. We will empirically measure its robustness using the empirical standard deviation of its error, as proposed in Sec. 3.1 and compare it with existing CL methods.

4.2 Pre-training as Meta CL

Given the importance of pre-training in SCROLL, we provide further theoretical motivation for its usage in CL. Specifically, we cast learning ψ_0 as meta-learning the CL prior $f_0 = \phi_0 \circ \psi_0$, with ϕ_0 parameterized by the zero matrix $W_0 = 0$. Specifically, we set $\text{Alg}(\cdot) = \text{RR}$ and minimize the CL generalization error in Eq. (2) over a distribution of CL sequences

$$\min_{\psi_0} \ \mathbb{E}_{(D,S)\sim\rho} \mathcal{L}(S(D), \psi_0, \text{RR}), \tag{6}$$

Intuitively, (6) is a meta-learning formulation of CL that aims to learn a suitable model initialization by training on many sequences $S(D)$. Similar to [41], (6) casts CL as a meta-learning problem. However, directly optimizing (6) or the objective proposed in [41] is prohibitively expensive for long sequences $S(D)$, since the cost of gradient descent scales linearly with the size of D. Fortunately for SCROLL, we can simplify (6) by exploiting the schedule invariance of RR from Prop. 1, which reduces (6) to

$$\min_{\psi_0} \ \mathbb{E}_D \mathbb{E}_{(x,y)\sim\pi_D} \ell(f_T(x), y) \quad \text{s.t.} \quad f_T = \text{RR}(D, \psi_0), \tag{7}$$

where f_T is the RR classifier trained on any sequence $S(D)$ over D (since it is independent on the sequence itself). Eq. (7) recovers the well-established meta-learning setting (i.e., without schedule dependence) and corresponds to meta-learning method proposed in [42].

For standard meta-learning, recent works strongly support learning ψ_0 by pre-training on standard multi-class classification [43, 44], instead of optimizing (7) directly. In particular, [44, 45] proved that pre-training is an upper bound to (7) and thus suitable for learning ψ_0. Crucially, pre-training is computationally more efficient than meta-learning and often yields more robust embedding. Consequently, initializing ψ_0 with pre-training can be principally justified as providing a meta-learning prior for CL, when RR is used as the CL algorithm. These are precisely the design choices adopted in SCROLL. Incidentally, recent works [e.g. 9, 12, 13, 28, 46] have also embraced pre-trained models and seen significant improvements in performance.

5 Experiments

In this section, we empirically study the relevance of schedule robustness in CL and evaluate the performance of SCROLL. We demonstrate that most existing CL methods lack schedule robustness, with a bias towards schedules with more classes per batch. In contrast, we show that SCROLL performs consistently over all evaluated schedules, in line with our theoretical analysis. SCROLL also outperforms most existing methods on average accuracy by a large margin. Lastly, we highlight in the ablation studies the contribution of various model components, including our ER routine and the usage of random features.

Setup. We consider class-incremental learning [14] models that observe novel classes incrementally and return a single-head model to discriminate all classes observed so far. For fairness, all methods are initialized with *the same pre-trained model*. The size $|M|$ denotes the total capacity of the replay buffer across all classes. We consider both task-based and task-free settings. We report CL performance both in terms of final *average accuracy* and *forgetting* according to [47]. Additional experiment details are provided in Sec. C. Code is available at https://github.com/RuohanW/scroll_cl.

5.1 Class-split Schedules on CIFAR100 Dataset

CIFAR100 is used as a standard benchmark for CL [e.g., 9, 14, 15], which splits the dataset into sequential tasks of disjoint classes. We denote the standard schedule as "10-split", which divides the dataset into 10 tasks of 10 disjoint classes each. Tab. 1 reports the average accuracy and forgetting metric for different CL methods, with the initial ψ_0 a ResNet pre-trained on Meta-Dataset [27]. We obtained the representation $\psi = \psi_{\text{RF}} \circ \psi_0$ by sampling $10K$ random features according to the process described in Sec. 3.2.

We compare SCROLL with *online* methods in the *task-free* setting and *offline* ones in the *task-based* setting. We observe that SCROLL outperforms both online and offline methods by a large margin (over 8% in most cases).

Tab. 1 also reports the performance of different CL methods on the 50-split schedule, which divides CIFAR100 into 50 tasks of 2 classes each. In clear contrast with the 10-split schedule, many existing methods suffer significant performance degradation. For instance, BIC's test accuracy drops from 57% to 36%, while iCaRL drops from 68.3% to 55.5%. In contrast, SCROLL performs consistently under both schedules, in line with our theoretical analysis. This also means that SCROLL demonstrates further performance advantage over existing methods under more challenging schedules. We note that RanPAC [12], SLDA [23] and FeCam [29] are also unaffected by the data schedules since their design guarantees these methods some level of schedule robustness (see discussion in Sec. 4.1). However, these methods are significantly outperformed by SCROLL.

One crucial reason for SCROLL's performance is representation update via ER, since f_T^* consistently improves upon f_T on both average accuracy and forgetting, suggesting its vital role in tackling the distribution shift of novel data from the pre-trained ψ_0. In contrast, methods such as RanPac, SLDA, and FeCam keep the pre-trained representation fixed, which limits final performance. We further remark that the minor performance differences in Tab. 1 for f_T^* between task-based vs task-free setting is caused by the difference in replay data. In the former setting, exemplar

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

		CIFAR100 10-splits				CIFAR100 50-splits			
		$\|M\| = 200$		$\|M\| = 2000$		$\|M\| = 200$		$\|M\| = 2000$	
	CL Algorithm	Accuracy (↑)	Forgetting (↓)	Accuracy (↑)	Forgetting (↓)	Accuracy (↑)	Forgetting (↓)	Accuracy (↑)	Forgetting (↓)
TASK-BASED	ER [21]	23.8 ±0.7	78.9 ±0.9	52.6 ±0.4	45.7 ±0.4	15.1 ±2.1	85.2 ±1.5	50.3 ±0.4	48.0 ±0.6
	iCaRL [14]	57.0 ±0.8	25.8 ±1.1	**68.3** ±0.3	17.2 ±0.2	42.5 ±0.8	27.1 ±0.4	55.5 ±0.7	19.6 ±0.6
	BIC [48]	35.2 ±3.3	64.4 ±4.0	57.1 ±0.8	39.4 ±0.9	16.1 ±4.1	85.5 ±1.5	36.8 ±2.7	65.7 ±2.8
	DER [15]	27.4 ±0.8	74.6 ±0.9	55.8 ±1.5	39.3 ±1.6	15.7 ±1.7	80.6 ±2.2	48.2 ±1.4	46.1 ±2.2
	DER++ [15]	28.7 ±1.3	72.7 ±1.5	59.5 ±1.0	35.4 ±1.1	18.2 ±1.4	76.7 ±1.4	53.1 ±0.9	35.4 ±0.9
	FeCam [29]	55.0 ±0	12.2 ±0.	55.0 ±0	12.2 ±0.	55.0 ±0	12.2 ±0.	55.0 ±0	12.2 ±0.
	RanPAC [12]	60.6 ±0.3	11.6 ±0.2	60.6 ±0.3	11.6 ±0.2	60.6 ±0.3	11.6 ±0.2	60.6 ±0.3	11.6 ±0.2
	SCROLL (f_T)	63.7 ±0.1	11.2 ±0.2	63.7 ±0.1	11.2 ±0.2	63.7 ±0.1	11.2 ±0.2	63.7 ±0.1	11.2 ±0.2
	SCROLL (f_T^*)	**65.3** ±0.1	**10.4** ±0.2	**68.4** ±0.2	**7.7** ±0.2	**65.3** ±0.1	**10.4** ±0.2	**68.4** ±0.2	**7.7** ±0.2
TASK-FREE	GDumb [22]	21.9 ±1.	24.3 ±1.5	59.3 ±0.4	16.7 ±0.4	22.8 ±1.9	27.9 ±1.6	58.9 ±0.4	18.2 ±0.2
	ER-ACE [5]	41.5 ±0.6	31.2 ±2.7	56.0 ±0.4	11.8 ±1.2	26.8 ±0.9	23.2 ±1.4	47.2 ±0.5	13.8 ±0.8
	ER-AML [5]	33.2 ±1.4	26.5 ±2.4	51.2 ±1.5	11.1 ±0.6	13.2 ±1.1	32.4 ±4.0	41.1 ±1.1	16.0 ±1.0
	SSIL [49]	46.0 ±3.3	20.8 ±1.0	59.4 ±1.7	10.6 ±1.6	32.9 ±0.6	28.1 ±0.5	52.2 ±0.9	16.4 ±0.6
	MIR [50]	27.5 ±0.6	69.4 ±0.5	56.1 ±1.3	33.7 ±1.7	7.9 ±0.7	81.4 ±0.6	41.1 ±0.5	37.3 ±3.8
	SLDA [23]	59.3 ±0.2	14.3 ±0.1	59.3 ±0.2	14.3 ±0.1	59.3 ±0.2	14.3 ±0.1	59.3 ±0.2	14.3 ±0.1
	SCROLL (f_T)	63.7 ±0	11.2 ±0	63.7 ±0	11.2 ±0	63.7 ±0	11.2 ±0	63.7 ±0	11.2 ±0
	SCROLL (f_T^*)	**65.0** ±0.3	**10.6** ±0.4	**68.0** ±0.3	**7.9** ±0.3	**65.0** ±0.3	**10.6** ±0.4	**68.0** ±0.3	**7.9** ±0.3
JOINT TRAINING (I.I.D.)		79.3 ±0.4							

TABLE 1: Class-incremental accuracy and forgetting on sequential CIFAR-100. Joint training (i.i.d.) accuracy is obtained by training on all classes with supervised learning. Best methods per setting are in bold.

	$\|M\| = 200$	$\|M\| = 500$	$\|M\| = 2000$
BIC	20.9 ±0.3	28.2 ±0.3	39.2 ±0.3
DER++	11.0 ±0.3	23.5 ±0.3	41.3 ±0.3
GDumb	22.9 ±1.3	37.3 ±1.5	59.2 ±0.5
iCaRL	43.5 ±0.8	52.8 ±1.4	56.4 ±0.7
SLDA	59.3 ±0.1	59.3 ±0.1	59.3 ±0.1
RanPAC	60.6 ±0.	60.6 ±0.	60.6 ±0.
SCROLL (f_T^*)	**65.3** ±0.1	**66.2** ±0.2	**68.4** ±0.2

TABLE 2: Class-incremental Accuracy on CIFAR100 with varied task sizes in a single schedule.

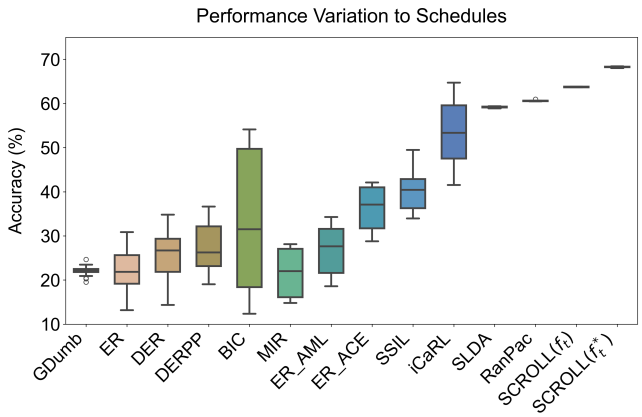


Fig. 3: Performance variation of CL methods on CIFAR100 under 5, 10, 25, and 50-split schedules with memory buffer $|M| = 200$. Taller boxes correspond to unreliable performance depending on the schedule, with a wide range of possible accuracy scores.

selection has access to full tasks for choosing representative replay samples. In the task-free setting, it has to choose them locally from mini-batches, leading to a minor drop in performance.

Varying Task Sizes in Schedules. So far we have only evaluated schedules where each task is of the same size. We thus further consider varying task sizes within a single schedule. Specifically, we let the class count for tasks to follow the pattern of 5, 2, 2, 1, 5, 2, 2, 1... resulting in a total of 40 tasks. Tab. 2 reports CL accuracy for the best performing models from Tab. 1.

We observe that SCROLL achieves the identical performance under this setting, consistent with Prop. 2 and the results against other schedules from Tab. 1. While SLDA and RanPAC's performance are unaffected by the very different schedules, SCROLL still outperforms all existing methods by a substantial margin.

5.2 Fair Comparison of CL Methods

The experimental evidence from Sec. 5.1 exemplifies the main motivation of this work, namely that model performance under a single schedule is insufficient to characterize CL methods. For instance, iCaRL is one of the best-performing models for 10-split CIFAR100 ($M = 2000$), achieving 68.3%. However, for both 50-split and dynamic schedules, iCaRL's test accuracy drops significantly and underperforms other methods. To visualize the performance variations across different schedules, Fig. 3 shows the spread of test accuracies for different CL methods (with buffer size $|M| = 200$, for other sizes, see Sec. B.3) in box plot format. Qualitatively, the taller the box for a method, the more its performance will be unreliable in practice.

The results reinforce our earlier observation that **model performance under a single schedule is insufficient for comparing different CL methods**, in particular when the data schedule encountered by a CL algorithm in practice is unknown or even dynamic. To fairly compare different CL methods under various possible schedules, we thus propose a new performance metric called **Schedule-Normalized Average Accuracy**, which computes a CL method's expected performance over a range of schedules (i.e., an empirical estimation of Def. 1). From our testing (see Sec. B for more details), we find that varying the number of class splits in a schedule effectively captures different performance levels of CL methods, with most existing methods implicitly biased towards lower-split schedules (i.e., more classes per batch), while suffering significant performance degradation under higher split schedules.

Tab. 3 reports the schedule-normalized average accuracy for CL methods. The result shows that SCROLL outperforms the existing methods on this metric, highlighting

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

CL Algorithm	$\|M\| = 200$	$\|M\| = 500$	$\|M\| = 2000$
iCaRL	52.7 ± 8.3	60.0 ± 6.1	65.2 ± 6.1
BIC	31.6 ± 14.5	41.4 ± 14.2	52.1 ± 12.2
DER++	26.7 ± 6.3	41.3 ± 5.5	58.4 ± 4.0
GDumb	22.0 ± 1.6	38.6 ± 1.3	59.6 ± 0.4
ER-ACE	36.9 ± 7.6	44.9 ± 6.7	54.4 ± 4.4
ER-AML	27.6 ± 9.0	37.4 ± 8.2	48.8 ± 4.6
SLDA	59.3 ± 0.2	59.3 ± 0.2	59.3 ± 0.2
RanPAC	60.6 ± 0.3	60.6 ± 0.3	60.6 ± 0.3
SCROLL (f_T^*)	**65.1** ± 0.2	**66.0** ± 0.3	**68.1** ± 0.3

TABLE 3: Schedule-Normalized Average Accuracy of different CL methods on CIFAR100.

Method	Schedule Robustness Metrics					
	$\|M\| = 200$		$\|M\| = 500$		$\|M\| = 2000$	
	PV	OPD	PV	OPD	PV	OPD
iCaRL	8.3	18.8	6.1	16.5	6.1	9.7
BIC	14.5	63.7	14.2	49.8	12.2	33.9
DER++	6.3	48.7	5.5	41.8	4.0	25.7
GDumb	1.6	17.7	1.3	12.9	0.4	6.7
ER-ACE	7.6	23.5	6.7	13.8	4.4	12.2
ER-AML	9.0	32.3	8.2	15.1	4.6	18.4
SLDA	0.2	0.6	0.2	0.6	0.2	0.6
RanPAC	0.3	0.2	0.3	0.2	0.3	0.2
SCROLL (f_T)	0.2	0.2	0.2	0.2	0.2	0.2
SCROLL (f_T^*)	0.3	0.5	0.1	0.5	0.4	0.5

TABLE 4: Schedule-Normalized Performance Variance (PV) and Average Order-Normalized Performance Disparity (OPD) on CIFAR100 for different buffer sizes $|M|$. For both metrics, lower (↓) is better.

its (expected) performance advantage if the data schedule is unknown. Lastly, we remark that, if there is any prior knowledge about the potential data schedules, we may sample them to obtain a more accurate performance measure.

5.3 Quantifying Schedule Robustness

We consider two metrics to quantify the schedule robustness of different CL methods. We report **Schedule-Normalized Performance Variance** (PV), the empirical variance of CL methods as an approximation of Def. 1. On the other hand, we report average Order-Normalized Performance Disparity (OPD) from [8], which computes how sensitive each task's performance is with respect to task orderings. For both metrics, lower is better and implies higher schedule robustness. Tab. 4 reports the two metrics for different CL methods. We observe that in all settings, and for all buffer sizes considered, SCROLL is schedule-robust for both metrics, consistent with the theoretical analysis.

For evaluated methods, all suffer from high PV, OPD, or both, with the exception of SLDA [23] and RanPAC [12]. SLDA achieves CL via streaming linear discriminant analysis given a fixed pre-trained model. As noted in Sec. 4.1, Prop. 1 holds for standard LDA. SLDA, as an approximation, is expected to also perform consistently in practice but lacks the strict theoretical guarantees of SCROLL. Moreover, SLDA does not perform experience replay to correct for distribution shift, leading to worse performance compared to SCROLL (e.g., see Tabs. 1 and 6).

RanPac (without representation update), corresponds to performing SCROLL(f_T) using a random ReLU layer (see Sec. 3.2) rather than the random Fourier feature (RFF) layer we adopted. This explains the similar behavior between RanPac and our non-fine-tuned f_T in the table. However, in contrast to random ReLUs, RFFs are a theoretically sound approach to approximating a *universal* reproducing kernel [51], which could explain the performance improvement of SCROLL(f_T) over RanPac in our experiments.

5.4 Bootstrapping Schedule on *mini*IMAGENET Dataset

We consider the CL setting introduced in [12, 52] where a dataset is split into one large initial *"bootstrap"* task (or batch) containing samples from several classes, followed by sequential tasks with only a few classes each. Each method is first trained on the initial task using standard i.i.d training and then performs CL on the following tasks. In this setting, learning on the bootstrap task may be interpreted as pre-training, with two key differences: first, such a task contains much fewer samples compared to typical pre-training

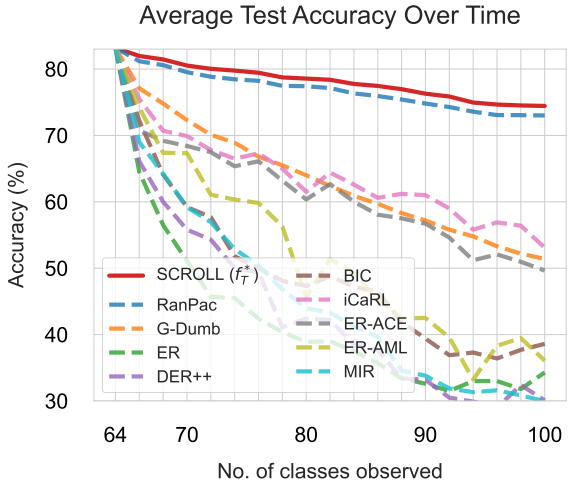


Fig. 4: CL Accuracy over time on *mini*IMAGENET, from the "bootstrap" task followed by a sequence of binary classification tasks (see the setup described in Sec. 5.4).

datasets. Second, models are tested also on samples from the bootstrap task, posing a further challenge in terms of forgetting. We consider this setup on the *mini*IMAGENET dataset, with the initial task containing 64 classes and the remaining 36 classes evenly split into 18 tasks.

To better explore the *stability-plasticity* tradeoff in CL, Tab. 5 reports the average performance of each method on all 100 classes from *mini*IMAGENET (Acc_{avg}), as well as the accuracy on the first 64 classes from the bootstrap task (Acc_{boot}), separate from the average accuracy on the remaining 36 classes from subsequent CL tasks ($\text{Acc}_{\text{novel}}$). We observe that SCROLL outperforms all previous methods by a significant margin and retains high average accuracy on both sets, while most previous methods fail to preserve the knowledge from the initial task and are unable to adapt sufficiently to the new tasks. RanPAC retains test accuracy comparable to SCROLL on the bootstrap tasks and performs slightly worse on the subsequent data, as reflected by $\text{Acc}_{\text{novel}}$.

As a separate relevant observation, Tab. 5 shows also that SCROLL does not require pre-training on massive datasets to learn ψ_0 and works well with limited data.

Model Performance over Time. CL methods are designed

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

*mini*IMAGENET			
CL Algorithm	Acc_{boot}	$\text{Acc}_{\text{novel}}$	Acc_{avg}
iCaRL	66.4 ±0.2	49.2 ±0.1	60.2 ±0.1
BIC	53.5 ±3.3	34.6 ±3.5	46.7 ±3.3
ER	29.3 ±1.7	62.4 ±0.6	41.2 ±1.3
DER++	34.1 ±1.5	46.3 ±0.7	38.5 ±1.0
GDumb	75.6 ±3.2	15.8 ±3.3	54.1 ±3.7
ER-ACE	60.6 ±1.2	30.2 ±1.6	49.6 ±1.0
ER-AML	42.3 ±1.6	25.0 ±3.2	36.1 ±1.5
SSIL	56.2 ±2.7	0.1 ±0.1	36.0 ±1.7
MIR	37.2 ±1.5	17.0 ±0.5	30.0 ±1.0
RanPAC	77.9 ±0.3	64.1 ±0.5	73.0 ±0.4
SCROLL (f_T^*)	**78.2** ±0.1	**68.4** ±0.3	**74.7** ±0.2

TABLE 5: Class-incremental classification accuracy on sequential *mini*IMAGENET, with $|M| = 200$.

to learn continuously. To evaluate this setting, Fig. 4 reports the performance of different CL methods after every task, for all classes observed so far. We note that SCROLL outperforms all previous methods, with the gap increasing with each observed task. Specifically, SCROLL's performance degrades slowly as new tasks are added, suggesting that it can perform reliably over long data sequences.

5.5 Long, Dynamic Schedule with Meta-Dataset

Lastly, we introduce a long and dynamic schedule on Meta-Dataset [27], which combines popular image classification datasets such as ILSVRC-2012 (ImageNet) [53], Aircraft [54], VGG Flower (Flower) [55], and Traffic Signs [56]. **In this setting, each CL model is pre-trained on the ImageNet dataset** and performs class-incremental learning on Aircraft, Flower, and Traffic Signs sequentially. All final models are single-head and are evaluated on a test set combining all test samples from the three datasets.

The proposed data stream presents several realistic challenges for assessing the efficacy of CL methods. Firstly, the input distribution of the data stream is evolving (from aircraft to flowers and to traffic signs), as the three datasets are learned sequentially. Secondly, the data stream has a large number of classes (245 in total) and requires fine-grained classification (e.g., different variants of the same aircraft model), which poses additional challenges in terms of forgetting and knowledge reuse. Lastly, we vary the task sizes and the order of tasks within the schedule to mimic the dynamic nature of CL in real-world scenarios (see Sec. D for more details on this data stream).

Tab. 6 reports schedule-normalized classification accuracy, PV (reported as standard deviation with accuracy), forgetting, as well as OPD for different CL methods on sequential Meta-Dataset as described above. In this challenging setting, SCROLL again outperforms existing methods by a large margin, while achieving schedule robustness as indicated by its low PV and OPD. Lastly, we highlight that with a buffer size $M = 5000$, SCROLL's performance is less than 3% lower than i.i.d training, which acts as the performance upper bound for CL. In contrast, the second-best performing methods for Tab. 6 are RanPAC and SLDA, both of which lag more than 10% from i.i.d training.

5.6 CL with Vision Transformers

All our experiments so far used pre-trained ResNet [35] as base models. A more recent alternative in the CL literature is to consider vision transformers (ViT) [37], which can encode richer prior knowledge and allow for a wide range of potential strategies for model updates, including incremental prompt tuning [28, 57] and LoRa [13]. In particular, SCROLL employs LoRa during ER-based updates when using ViT as base model (see Sec. 3.2.4).

Tab. 7 compares SCROLL with state-of-the-art CL approaches based on ViT (see Sec. C for additional details on the ViT-based SCROLL). We observe that, with the exception of FeCam and SCROLL, the performance of all other methods deteriorates in 50-split schedules. In particular, representation update in RanPac, while improving the 10-split performance, is detrimental to higher-split schedules. In contrast, SCROLL retains consistent performance across the different schedules and outperforms all baselines.

6 Ablation Studies

In this section, we perform ablation experiments to better understand the properties of SCROLL. We focus on our proposed random feature layer and ER procedure.

6.1 Ablations on Random Projection Layer

We study the effect of random projection layer on CL performance. To this end, we compare RanPAC [12], our proposed random Fourier features, and the original pre-trained features in the replay-free setting. We use RR to learn the optimal classifiers for each of the features and compare their test accuracy in Tab. 8.

The results validate our hypothesis from Sec. 3.2, that the pre-trained representation, learned from auxiliary classes, may not be linearly separable for the novel classes encountered during CL. Consequently, non-linear projection of the original features into higher-dimensional spaces (e.g., $m = 10000$), could induce more separability across different classes, as evidenced by both RanPac and our proposed Fourier features. Between the two, the results further suggest that random Fourier features are more robust and provide better test performance across all evaluated datasets, as already discussed in Sec. 5.3.

6.2 Experience Replay with Pre-trained Models

Classifier initialization. When finetuning SCROLL to obtain $f_T^* = \phi_T^* \circ \psi_T^*$ from $f_T = \phi_T \circ \psi$, the role of the RR solution ϕ_T may be interpreted as warm-starting the model's last layer before the representation updates to ψ_T. In principle, one could initialize ϕ differently, for example adopting a random initialization as typically done in standard fine-tuning. Tab. 9 compares RR vs random initialization.

Random initialization corresponds to ignoring the data stream and learning from replay data only. It performs worse in Tab. 9, suggesting that learning from replay data alone is insufficient, even with strong pre-training. Further, the results suggest that an appropriately "warm-starting" f_T is crucial for final performance via fine-tuning, and SCROLL offers one such strategy with RR.

Buffering Strategies. We ablate different buffering strategies used in ER. We consider exemplar selection [14] and

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

	$\|M\| = 1250$			$\|M\| = 2500$			$\|M\| = 5000$		
CL Algorithm	Accuracy (↑) ± PV (↓)	Forgetting (↓)	OPD (↓)	Accuracy ± PV	Forgetting	OPD	Accuracy ± PV	Forgetting	OPD
DER++	59.2 ±6.1	38.4 ±6.4	28.9	60.1 ±6.7	36.9 ±6.9	31.3	70.2 ±2.2	26.8 ±2.1	19.3
iCaRL	48.0 ±5.9	42.5 ±4.8	20.6	46.3 ±3.6	43.8 ±3.7	16.5	50.9 ±2.7	38.3 ±2.4	14.6
ER-ACE	34.3 ±1.5	13.8 ±1.5	19.5	34.0 ±3.3	15.1 ±3.7	25.8	33.1 ±2.7	15.1 ±3.2	23.9
ER-AML	25.5 ±1.2	9.1 ±1.6	25.0	23.8 ±1.7	10.9 ±1.7	24.9	24.9 ±2.4	9.8 ±2.0	24.1
SSIL	42.4 ±0.9	8.2 ±0.9	16.6	42.3 ±0.9	9.8 ±2.0	20.1	45.4 ±0.9	6.9 ±0.7	18.7
MIR	49.5 ±1.2	16.9 ±1.3	22.1	49.5 ±2.8	16.9 ±2.8	25.4	47.9 ±1.7	18.1 ±1.1	27.1
SLDA	72.1 ±0.1	8.9 ±0.5	**0.9**	72.1 ±0.1	8.9 ±0.5	**0.9**	72.1 ±0.1	8.9 ±0.5	**0.9**
RanPac	76.6 ±0.5	7.0 ±0.5	1.8	76.6 ±0.5	7.0 ±0.5	1.8	76.6 ±0.5	7.0 ±0.5	1.8
SCROLL (f_t^*)	**78.9** ±0.3	**5.7** ±0.1	1.9	**83.4** ±0.3	**4.6** ±0.2	2.1	**87.6** ±0.2	**5.1** ±0.6	2.4
I.I.D. TRAINING				90.8 ±0.7					

TABLE 6: Schedule-normalized class-incremental accuracy, forgetting and OPD on sequential Meta-Dataset. Joint training (i.i.d.) accuracy is obtained by training on all classes with supervised learning.

	CIFAR100				ImageNet-R				Core50
Schedule	10-split		50-split		10-split		50-split		8-split
CL Algorithm	Acc (↑) ± PV (↓)	Forgetting (↓)	Acc ± PV	Forgetting	Acc ± PV	Forgetting	Acc ± PV	Forgetting	Acc ± PV
L2P	82.3±0.7	9.8 ±1.3	65.6±2.1	12.2 ±1.2	73.6±0.4	5.9 ±0.3	65.8±0.7	8.7 ±0.5	78.3 ±0.7
FeCam	85.7±0	5.0 ±0	85.7±0	5.0 ±0	66.3±0	5.6 ±0.	66.3±0	5.6 ±0.	89.1 ±0.7
DualPrompt [57]	84.6 ±0.3	5.1 ±0.7	67.8 ±0.4	9.7 ±0.7	72.5 ±0.4	3.9 ±0.4	66.8 ±0.8	5.2 ±0.4	-
Coda-P [28]	85.9 ±0.6	4.9 ±0.1	67.3 ±0.7	8.8 ±0.6	74.8 ±0.4	4.3 ±0.3	63.6 ±0.1	5.4 ±0.2	89.1 ±0.7
InfLoRa	86.5 ±0.7	5.2 ±0.4	59.8 ±1.2	15.2 ±0.7	74.8 ±0.3	6.6 ±0.5	56.6 ±0.8	11.1 ±0.6	-
RanPAC	88.7 ±0.4	5.1 ±0.4	88.7 ±0.4	5.1 ±0.4	70.9 ±0.6	5.6 ±0.3	70.9 ±0.6	5.6 ±0.3	94.1±0.2
RanPAC (with Phase 1)	**91.4** ±0.8	3.8 ±0.3	88.2 ±0.4	5.6 ±0.4	78.5 ±0.4	4.3 ±0.3	75.4 ±0.7	6.6 ±0.6	**96.4**±0.2
SCROLL (f_t)	89.2 ±0.2	4.6 ±0.2	89.2 ±0.2	4.6 ±0.2	71.2 ±0.3	5.4 ±0.2	71.2 ±0.3	5.4 ±0.2	94.4 ±0.2
SCROLL (f_t^*)	**91.9** ±0.2	3.5 ±0.2	**91.9** ±0.2	3.5 ±0.2	**78.9** ±0.3	3.2 ±0.3	**78.9** ±0.3	3.2 ±0.3	**96.2** ±0.1

TABLE 7: Comparison of Vit-based CL Approaches. SCROLL outperforms all baselines and is schedule-robust. RanPac (with Phase 1) performs representation fine-tuning on the first task followed by RR using the updated representation.

	CIFAR100	**ImageNet-R**	**Meta-Dataset**
Original Pre-trained	57.1	65.1	71.6
RanPAC (ReLU)	60.6	70.7	76.6
Ours (Fourier)	**63.7**	**71.2**	**77.8**

TABLE 8: Comparing the quality of different feature representations. SCROLL with Random Fourier features outperforms both the original pre-trained representation and random ReLU features (RanPAC [12]).

	CIFAR100 (ResNet)		**ImageNet-R (VIT)**	
ϕ Init	$f_T \rightarrow f_T^*$		$f_T \rightarrow f_T^*$	
Random	1.2 ±0.3	66.1 ±0.2	0.26 ±0.3	77.0 ±0.5
RR	**63.7** ±0	**68.4** ±0.1	**71.2** ±0.3	**78.9** ±0.3

TABLE 9: Classification accuracy with alternative classifier initialization in SCROLL, $|M| = 2000$.

	CIFAR100 (ResNet)		**ImageNet-R (Vit)**	
Buffering \ $\|M\|$	500	2000	600	2000
Ours	**66.2** ±0.1	**68.4** ±0.2	**77.0** ±0.2	**78.9** ±0.3
Random	65.8 ±0.3	67.5 ±0.3	76.8 ±0.3	77.5 ±0.2
Nearest	64.2 ±0.1	64.9 ±0.2	65.8 ±0.3	74.9 ±0.4
Outlier	64.0 ±0.1	64.8 ±0.3	75.7 ±0.3	77.6 ±0.4

TABLE 10: Classification accuracy with different buffering strategies.

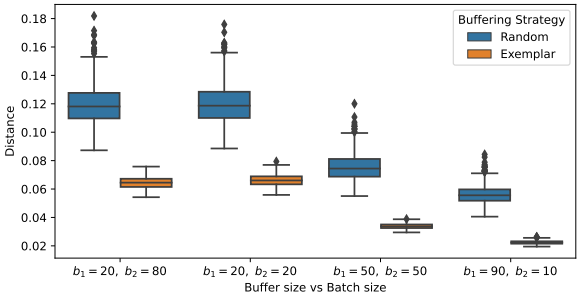


Fig. 5: Effect of buffer (b_1) and batch (b_2) sizes on the population mean approximation for samples in M_T. Our exemplar selection (Orange) vs. Random Sampling [14] (Blue). Lower distance implies better population mean approximation. Lower variance indicates a more consistent M_T.

random sampling, which are commonly used in existing works. We also introduce two variants to exemplar selection, including selecting samples closest to the mean representation of each class (Nearest selection) or selecting samples furthest to the mean representation of each class (Outlier selection).

We report the effects of different buffering strategies in Tab. 10. The exemplar strategy clearly performs the best, followed by random reservoir sampling. Both nearest and outlier selections perform drastically worse. From the results, we hypothesize that an ideal buffering strategy should store representative samples of each class (e.g., still tracking the mean representation of each class as in exemplar selection) while maintaining sufficient sample diversity (e.g., avoiding nearest selection).

The results have several implications for CL algorithms. While random reservoir sampling performs well *on average*, it can perform poorly in the *worst cases*, such as when the random sampling coincides with either nearest or outlier selection. Therefore, it appears preferable to rank the samples observed during CL and maintain representative yet diverse samples, compared to random sampling. Further, we remark that random sampling contributes towards worse OPD metrics. This is evident from Tab. 4, where GDumb [22], an existing CL baseline that only learns from experience replay with random sampling, shows high OPD despite achieving low PV.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

6.3 Schedule Robustness of proposed ER

Proposition 2 proves our exemplar selection strategy yields a deterministic replay buffer when each class is observed at once. Here we investigate its behavior under more general schedules where class data is spread across a sequence, comparing with random reservoir sampling [14] as a baseline.

We analyze each class independently since both strategies select samples per class. For clarity, buffer size b_1 and batch size b_2 refer to per-class capacities. We measure performance as $\|\hat{\mu}_y - \mu_y\|$ with $\hat{\mu}_y$ the last buffer's (M_T) class mean and μ_y the population mean for each class $y \in \mathcal{Y}$.

Fig. 5 compares performance over 4 different (b_1, b_2) averaged across 100 random data shuffles on CIFAR dataset to simulate schedules variability. Our strategy (Exemplar) consistently approximates the population mean better than reservoir sampling (Random), across all settings, corroborating the results from Tab. 10. Our method also exhibits significantly lower variance, demonstrating that SCROLL produces similar high-quality buffer distributions regardless of the schedule, making it inherently schedule-robust.

7 Conclusion

In this work, we introduced *schedule robustness* in CL, requiring algorithms to perform consistently across multiple data schedules—a crucial property for model reliability. Our goal was motivated by an initial evaluation revealing that most existing CL methods lack this property, resulting in degraded performance in certain regimes. Across diverse datasets and schedules, we found no previous method consistently outperforming the others, making it difficult to select an appropriate method for varying scenarios.

To achieve schedule robustness we proposed SCROLL, which recasts CL into a meta-CL formulation with subsequent model fine-tuning. Key innovations include a random feature layer to address distribution shifts between pretraining and CL data (inspired by [12]), and a novel experience replay strategy maintaining a schedule-robust memory buffer. Theoretical analysis and extensive empirical evaluation demonstrate that SCROLL is schedule-robust by design, outperforming existing CL algorithms significantly across all datasets in both average accuracy and robustness. While the *on-demand* paradigm could potentially impose computational costs when frequent updates are needed for small-sized tasks, in most realistic applications, updates are only required after substantial data accumulation, effectively mitigating this concern.

Our findings suggest a promising general strategy for CL: *learning online a coarse-yet-robust model for each class followed by on-demand adaptation*. This approach grants both flexibility and consistency to CL algorithms. The online step prevents inter-class interference by learning each class independently, while the adaptation step corrects for distribution shifts in model representation. Using cross-entropy loss for adaptation optimizes for the most likely class, improving overall performance. SCROLL's robust performance offers strong support for adopting this strategy.

References

[1] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

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

[3] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021.

[4] R. Aljundi, K. Kelchtermans, and T. Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019.

[5] L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau, and E. Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022.

[6] M. Shanahan, C. Kaplanis, and J. Mitrović. Encoders and ensembles for task-free continual learning. *arXiv preprint arXiv:2105.13327*, 2021.

[7] S. Farquhar and Y. Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

[8] J. Yoon, S. Kim, E. Yang, and S. J. Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *International Conference on Learning Representations*, 2020.

[9] Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022.

[10] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486, 2019*, 2019.

[11] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

[12] M. D. McDonnell, D. Gong, A. Parvaneh, E. Abbasnejad, and A. Van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36: 12022–12053, 2023.

[13] Y.-S. Liang and W.-J. Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647, 2024.

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

[15] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in Neural Information Processing Systems*, 33:15920–15930, 2020.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

[16] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

[17] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] F. Ye and A. G. Bors. Online task-free continual generative and discriminative learning via dynamic cluster memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26202–26212, 2024.

[19] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan. Remind your neural network to prevent catastrophic forgetting. In *European conference on computer vision*, pages 466–483. Springer, 2020.

[20] L. Wang, X. Zhang, K. Yang, L. Yu, C. Li, L. HONG, S. Zhang, Z. Li, Y. Zhong, and J. Zhu. Memory replay with data compression for continual learning. In *International Conference on Learning Representations*, 2022.

[21] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

[22] A. Prabhu, P. H. Torr, and P. K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, pages 524–540, 2020.

[23] T. L. Hayes and C. Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 220–221, 2020.

[24] X. Wei, G. Li, and R. Marculescu. Online-lora: Task-free online continual learning via low rank adaptation. *arXiv preprint arXiv:2411.05663*, 2024.

[25] M. Boschini, L. Bonicelli, A. Porrello, G. Bellitto, M. Pennisi, S. Palazzo, C. Spampinato, and S. Calderara. Transfer without forgetting. In *European Conference on Computer Vision*, 2022.

[26] A. Krizhevsky, V. Nair, and G. Hinton. Cifar 100 dataset. https://www.cs.toronto.edu/~kriz/cifar.html, 2009.

[27] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, and H. Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2019.

[28] J. S. Smith, L. Karlinsky, V. Gutta, P. Cascante-Bonilla, D. Kim, A. Arbelle, R. Panda, R. Feris, and Z. Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11909–11919, 2023.

[29] D. Goswami, Y. Liu, B. Twardowski, and J. Van De Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems*, 36:6582–6595, 2023.

[30] F. Liu, X. Huang, Y. Chen, and J. A. Suykens. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7128–7148, 2021.

[31] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20, 2007.

[32] T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation*. Prentice Hall, 2000.

[33] Z. Fu, H. Yang, A. M.-C. So, W. Lam, L. Bing, and N. Collier. On the effectiveness of parameter-efficient fine-tuning. In *The AAAI Conference on Artificial Intelligence*, volume 37, pages 12799–12807, 2023.

[34] W. Li, X. Liu, and H. Bilen. Cross-domain few-shot learning with task-specific adapters. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022*, 2022.

[35] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[36] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[38] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta. Incremental robot learning of new objects with fixed update time. In *2017 IEEE International Conference on Robotics and Automation*, pages 3207–3214, 2017.

[39] R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pages 412–419. PMLR, 2007.

[40] P. Xanthopoulos, P. M. Pardalos, T. B. Trafalis, P. Xanthopoulos, P. M. Pardalos, and T. B. Trafalis. Linear discriminant analysis. *Robust data mining*, pages 27–33, 2013.

[41] K. Javed and M. White. Meta-learning representations for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[42] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. *International Conference on Learning Representations*, 2019.

[43] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, pages 266–282. Springer, 2020.

[44] R. Wang, M. Pontil, and C. Ciliberto. The role of global labels in few-shot classification and how to infer them. *Advances in Neural Information Processing Systems*, 34, 2021.

[45] R. Wang, J. I. T. Falk, M. Pontil, and C. Ciliberto. Robust meta-representation learning via global label inference

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3614868

and classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[46] M. D. McDonnell, D. Gong, E. Abbasnejad, and A. v. d. Hengel. Premonition: Using generative models to preempt future data changes in continual learning. *arXiv preprint arXiv:2403.07356*, 2024.

[47] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[48] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.

[49] H. Ahn, J. Kwak, S. Lim, H. Bang, H. Kim, and T. Moon. Ss-il: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 844–853, 2021.

[50] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars. Online continual learning with maximally interfered retrieval. *Advances in Neural Information Processing Systems*, 2019.

[51] A. Christmann and I. Steinwart. *Support vector machines*. Springer, 2008.

[52] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019.

[53] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[54] S. Maji, E. Rahtu, J. Kannala, M. B. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013.

[55] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008.

[56] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *The 2013 International Joint Conference on Neural Networks*, pages 1–8, 2013.

[57] Z. Wang, Z. Zhang, S. Ebrahimi, R. Sun, H. Zhang, C.-Y. Lee, X. Ren, G. Su, V. Perot, J. Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer, 2022.

Ruohan Wang is a senior research scientist at Institute for Infocomm Research, A*STAR Singapore. Previously, he was a postdoctoral researcher at UCL's Intelligent Systems Group, supervised by Prof. Massimiliano Pontil. He completed his Ph.D. in machine learning at Imperial College, funded by Singapore National Science Scholarship. He has a broad interests in topics of machine learning, including representation learning, meta-learning, and imitation learning. His research goal is to design robust ML systems that could efficiently leverage past experiences and existing knowledge for future learning.

Marco Ciccone is a Distinguished Postdoctoral Fellow at Vector Institute in Toronto. From 2021 to 2024, he was an ELLIS Postdoctoral Researcher at the Polytechnic University of Turin and University College London. He served as Competition Track co-chair at NeurIPS in 2021, 2022 and 2023. He received a Ph.D. in Computer Science and Engineering cum laude at Polytechnic of Milan, working on iterative and conditional models for visual representation learning. His research focuses on building modular machine learning systems that can efficiently learn to solve new problems by reusing, composing, and improving previously learned skills. This includes research on: Federated Learning, to enable collaboration and decentralization by sharing knowledge and resources while preserving privacy; Continual Learning, to enable continuous knowledge integration without forgetting; Modular Learning, for designing more reusable, maintainable, and efficient models.

Massimiliano Pontil is Senior Researcher at the Italian Institute of Technology (IIT), where he leads the Computational Statistics and Machine Learning group, and co-director of the ELLIS Unit Genoa, a joint effort of IIT and the University of Genoa. He is also part-time professor at University College London and member of the UCL Centre for Artificial Intelligence. He obtained his PhD in Physics from the University of Genoa in 1999. He has made significant contributions to machine learning, particularly in the areas of kernel methods, multitask and transfer learning, sparsity regularization and statistical learning theory. He has published about 150 papers at international journals and conferences, is regularly on the programme committee of the main machine learning conferences, and has been on the editorial board of the Machine Learning Journal, Statistics and Computing, and JMLR. He has held visiting positions at a number of universities and research institutes, including the Massachusetts Institute of Technology, the Isaac Newton Institute for Mathematical Sciences in Cambridge, the City University of Hong Kong, the University Carlos III de Madrid, ENSAE Institute Polytechnique Paris, and Ecole Polytechnique.

Carlo Ciliberto is Associate Professor with the Centre for Artificial Intelligence at University College London, He is member of the ELLIS society and of the ELLIS Unit based at UCL. He obtained his bachelor and master degrees in Mathematics at the Università Roma Tre (Magna Cum Laude) and a PhD in machine learning applied to robotics and computer vision at the Istituto Italiano di Tecnologia. He has been Postdoctoral Researcher at the Massachusetts Institute of Technology with the Center for Brain Minds and Machines and became Lecturer (Assistant Professor) at Imperial College London before joining UCL, where he now carries out his main research activity. Carlo's research interests focus on foundational aspects of machine learning within the framework of statistical learning theory. He is particularly interested in the role of "structure" (being it in the form of prior knowledge or structural constraints) in reducing the sample complexity of learning algorithms with the goal of making them more sustainable both computationally and financially. He investigated these questions within the settings of structured prediction, multi-task and meta-learning, with applications to computer vision, robotics and recommendation systems.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/