WHY SUBSPACE PROTECTION NEEDS A FIXED RANK AND NEURON RECYCLING IN TASK-FREE CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks trained on long task-free streams suffer from two failure modes known as loss of plasticity and catastrophic forgetting. In practice both failures can co-occur and a method commonly used against one can worsen the other, yet few studies address them together. Here we design task-free streams for supervised and reinforcement learning in which tasks recur after long and variable gaps. On these streams we study how subspace protection against forgetting interacts with neuron recycling against loss of plasticity. We find that protection whose subspace continues to grow over the stream eventually saturates and reduces plasticity even when recycling keeps almost every neuron active. A fixed rank avoids saturation, but our analysis identifies a second cost. Protection aligned with the mean input direction of a layer can suppress one of the main pathways through which dormant neurons reactivate. Removing this direction from the protected subspace restores reactivation in our experiments as the analysis predicts. Recycling therefore remains necessary even at a fixed rank. We further show that protecting layers whose input directions are less shared across tasks lowers plasticity on both streams. These results suggest that continually learning neural networks may need a bound on the protected rank along with neuron recycling and a careful choice of the protected layers. They also indicate that studying forgetting and loss of plasticity together may be important for practical continual learning algorithms.

1 INTRODUCTION

A neural network trained on a long stream of changing tasks can fail in two ways. It can lose plasticity, the ability to fit new targets that a freshly initialized neural network fits readily (Dohare et al., 2024; Lyle et al., 2023; Abbas et al., 2023), and it can forget, overwriting earlier tasks as it learns later ones (Kirkpatrick et al., 2017). Much prior work treats the two separately. Methods against loss of plasticity (Sokar et al., 2023; Dohare et al., 2024; Kumar et al., 2024; Lewandowski et al., 2025; Chung et al., 2024) are mostly evaluated on streams that do not test retention of earlier tasks, methods against forgetting (Zeng et al., 2019; Saha et al., 2021; Wang et al., 2021) are mostly evaluated with task boundaries, and fewer methods address both (Elsayed & Mahmood, 2024). When tasks recur without boundaries, a learner faces both, and a remedy for one can worsen the other.

Subspace protection is an established approach to reducing forgetting. It projects weight updates away from the input directions that earlier tasks used (Zeng et al., 2019; Saha et al., 2021; Wang et al., 2021). We show that on a long task-free stream this protection can itself cause loss of plasticity through two mechanisms. The first is saturation, which prior work noted for projection methods with task boundaries (Zeng et al., 2019; Lin et al., 2022) and which also arises on task-free streams. When protected directions accumulate, the subspace left for new learning shrinks until none remains. The second, persistent dormancy, arises because the input of a layer after a rectified linear unit is dominated by its mean direction, which the protected subspace contains almost entirely, so projecting updates off that subspace can suppress one of the main pathways through which a dormant neuron reactivates. Section 4 states both as propositions, and Sections 6.1 and 6.2 test them.

Each mechanism can be addressed by one component. A fixed rank prevents saturation, since the free dimension of a layer then no longer shrinks with the stream, and an established neuron re-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

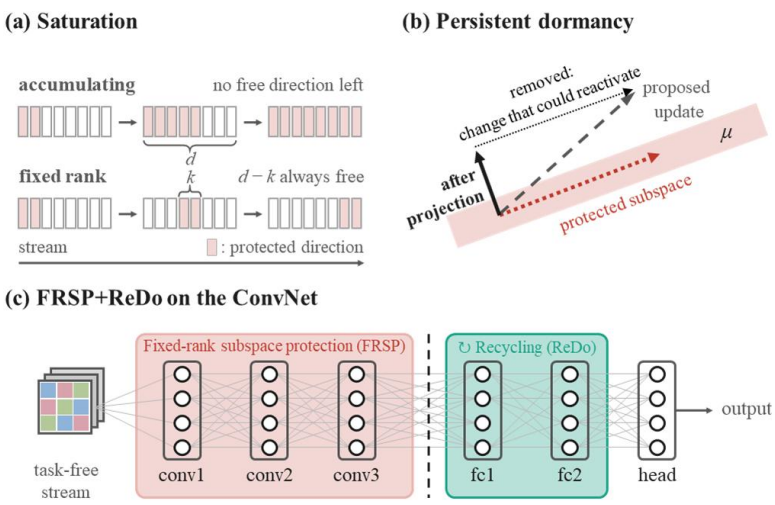


Figure 1: Loss of plasticity caused by subspace protection on a task-free stream, and fixed-rank protection with neuron recycling as the remedy. (a) Saturation caused by accumulating protected directions. (b) Persistent dormancy when the protected subspace contains the mean input direction μ. (c) FRSP+ReDo on the ConvNet of Dohare et al. (2024), with the protected layers and the layers in which recycling acts (Section 6.2).

cycling rule reinitializes the neurons that remain dormant, which in our experiments lie mainly in the unprotected layers that follow the protected ones. We combine fixed-rank subspace protection (FRSP) with the rule for recycling dormant neurons (ReDo) of Sokar et al. (2023) and call the learner FRSP+ReDo. It tracks the dominant input subspace of each protected layer at a fixed rank, projects every weight update off that subspace and reinitializes dormant neurons (Figure 1), without task boundaries, task identities or replay. We study it on two task-free streams with recurring tasks that the cyclic novelty (CN) protocol builds. CN-ImageNet is a new supervised benchmark from ImageNet-32 (Chrabaszcz et al., 2017), and CN-Procgen is a continual reinforcement learning stream from Procgen games (Cobbe et al., 2020).

Contributions.

- We show that subspace protection can itself cause loss of plasticity on task-free streams through two mechanisms, saturation and persistent dormancy. Each mechanism needs its own remedy, a fixed rank and neuron recycling, and a factorial comparison shows that neither replaces the other (Sections 4, 6.1 and 6.2).
- To test these mechanisms, we introduce the cyclic novelty protocol, which builds task-free streams with recurring tasks for supervised and reinforcement learning, CN-ImageNet and CN-Procgen, and measures plasticity against a learner reinitialized at every task change, and we provide the experiments and analyses on both streams (Sections 3 and 6).
- Based on these results, we propose three design principles for subspace protection on task-free streams. Keep the protected rank bounded, pair protection with neuron recycling, and apply it to the layers whose input directions are shared across tasks (Sections 6.3 and 7).

2 Related Work

Loss of plasticity. Loss of plasticity is the decline, as training on a changing stream continues, of the ability of a neural network to fit new targets, measured against a freshly initialized neural network. The phenomenon was noted early in reinforcement learning (Nikishin et al., 2022; Lyle et al., 2022), and related effects of non-stationarity on generalization were reported in reinforcement learning (Igl et al., 2021) and in warm-started supervised training (Ash & Adams, 2020). Dohare et al. (2024) showed on thousands of sequential tasks that accuracy decays to near or below that of a linear neural network, Lyle et al. (2023; 2024) traced the loss to curvature, pre-activation shift and

parameter-norm growth, Abbas et al. (2023) documented and mitigated it in continual reinforcement learning, and Nikishin et al. (2023) in deep reinforcement learning. Mitigation methods reinitialize neurons, as in ReDo (Sokar et al., 2023), continual backpropagation (CBP, Dohare et al., 2024) and self-normalized resets (Farias & Jozefiak, 2025), or regularize, as in shrink-and-perturb (Ash & Adams, 2020), L2-Init (Kumar et al., 2024), spectral regularization (Lewandowski et al., 2025), Parseval regularization (Chung et al., 2024) and weight clipping (Elsayed et al., 2024), and recent work uses gradient signals to set replacement rates (Zhou et al., 2026) or to select the neurons to reset (Yang et al., 2026). These methods do not aim to preserve earlier tasks. FRSP+ReDo uses the ReDo rule for recycling because its criterion is dormancy, the state that protection preserves (Section 4), and because, as in CBP, a reset zeroes the outgoing weights and so leaves the outputs nearly unchanged.

Catastrophic forgetting and subspace projection. Catastrophic forgetting is the loss of performance on earlier tasks as a neural network is trained on later ones (McCloskey & Cohen, 1989). It is addressed by regularization methods that anchor parameters at task ends, such as elastic weight consolidation (EWC, Kirkpatrick et al., 2017), synaptic intelligence (Zenke et al., 2017) and online EWC (Schwarz et al., 2018), by replay methods that store samples, such as experience replay (ER, Rolnick et al., 2019; Chaudhry et al., 2019), and by subspace-projection methods that confine updates to the complement of a protected subspace built from gradients of the outputs on earlier tasks (Farajtabar et al., 2020) or from accumulated feature statistics, such as orthogonal weights modification (OWM, Zeng et al., 2019), gradient projection memory (GPM, Saha et al., 2021) and Adam-NSCL, which trains in the null space of the feature covariance (Wang et al., 2021). Each accumulates protected directions as tasks arrive, so the number of protected directions grows with the stream. This line of work has noted that the growing protected subspace leaves too few free directions for later tasks (Zeng et al., 2019; Lin et al., 2022), and it relaxes the null-space constraint (Kong et al., 2022), reuses frozen weights of earlier tasks through learned scaling (Lin et al., 2022), decouples the feature space into shared and task-specific parts (Zhao et al., 2023), or scales the projection per direction by importance (Saha & Roy, 2023), still at task boundaries and with a protected dimension that grows with the tasks. FRSP+ReDo instead fixes the protected dimension on a task-free stream and recycles the neurons that the projection leaves dormant. Xiong et al. (2026) project only the reactivation updates of low-plasticity neurons onto the null space of a decayed feature second-moment matrix accumulated over earlier task chunks with known boundaries. By Proposition 2, such a projection also largely removes the change along the mean input direction through which these neurons reactivate, whereas FRSP+ReDo projects every update of the protected layers at a fixed small rank on task-free streams with recurrence. Utility-based perturbed gradient descent (UPGD, Elsayed & Mahmood, 2024) addresses both failures without boundaries by gating each update with a utility estimate, and the same work evaluates a streaming form of EWC (S-EWC), which we use as a baseline. Task-free continual learning, without boundaries or task identities, was named by Aljundi et al. (2019), recurring classes have been studied in class-incremental streams with boundaries between experiences (Hemati et al., 2023), and recurring tasks in reinforcement learning without task information (Rolnick et al., 2019). CN-ImageNet keeps the task-free setting while adding recurrence and a plasticity reference.

3 Task-free continual learning with recurring tasks

We study subspace protection in this setting. A learner receives one stream of mini-batches from a sequence of tasks without any boundary or task identity, and a seen task recurs after a long and variable gap. It should keep learning every task it meets, new or recurring (plasticity), and retain what it learned across the gaps (memory). Methods that use boundaries, task identity or stored samples are marked *privileged*.

CN-ImageNet. CN-ImageNet, the supervised instance of the cyclic novelty protocol, draws disjoint tasks of five ImageNet-32 classes (Deng et al., 2009; Chrabaszcz et al., 2017). A schedule is a sequence of segments, each a contiguous run of mini-batches from one task, that presents every task once as novel and revisits it after long and variable gaps, with segment lengths drawn at random (Figure 2). One head of five outputs with task-local labels is shared by all tasks. Hyperparameters are selected on a development stream of held-out classes, and results are reported on the main stream of the remaining classes (Appendix B).

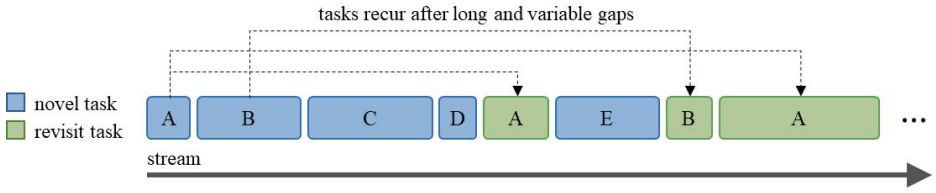


Figure 2: A schedule of the cyclic novelty protocol. Every task appears once as novel and recurs after long and variable gaps, in segments of random length within one stream.

CN-Procgen. CN-Procgen applies the same protocol to continual reinforcement learning with Procgen games. It draws twelve Procgen games (Cobbe et al., 2020) in easy mode, splits each game into three buckets of levels and presents the first appearances of games and of level buckets interleaved with revisits. Each parallel environment switches to the next task at its own episode end, so no boundary is visible in the data (Appendix B).

Metrics. Plasticity is measured against a restart oracle that reinitializes the neural network at every boundary and learns the segment alone. The plasticity score P is the mean over segments of the ratio of the online accuracy over the last quarter of a segment, less chance, to that of the oracle, less chance, split into P_{novel} and P_{revisit}, so $P < 1$ indicates loss of plasticity and $P > 1$ that the learner fits a segment better than a freshly initialized neural network. Memory is measured at revisits by the first-batch recall R_0 and by A_{320} and A_{3200}, the accuracy on the first 320 and 3200 samples of a revisit. R_0 is the accuracy above chance on the first mini-batch of a revisit, before any update, divided by that at the end of the previous visit and pooled over revisits. The state of the neural network is summarized by the dormant fraction, the share of hidden units whose mean activation relative to the layer mean is at most 0.025, and by the effective rank of the penultimate features (Appendix B defines these and further metrics).

4 Why subspace protection costs plasticity

We state why subspace protection can reduce plasticity on these streams and what a fixed rank guarantees. Consider a hidden layer with input $x \in \mathbb{R}^d$, second moment $\Sigma = \mathbb{E}[xx^\top]$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_d$, and a protected subspace with orthonormal basis $U \in \mathbb{R}^{d \times k}$. For every layer after the first, x is the output of the previous layer after the rectified linear unit (ReLU), and after the convolutional layers of the ConvNet also after max pooling. Protection replaces every weight update ΔW by $\Delta W P_\alpha$ with $P_\alpha = I - \alpha U U^\top$, and $\rho_k = 1 - \operatorname{tr}(U^\top \Sigma U)/\operatorname{tr}\Sigma$ is the fraction of the input energy outside the protected subspace. The proofs are in Appendix A.

Saturation. Accumulating projectors (Figure 1a) enlarge the protected subspace at every task end, or on a fixed period without boundaries, and never shrink it, with the following consequence.

Proposition 1 (Accumulating protection exhausts the learnable subspace)**.** *Let $S_m \subseteq \mathbb{R}^d$ be the protected subspace of a layer after m tasks, with $S_0 = \{0\}$, $S_{m-1} \subseteq S_m$, and every weight update projected as $\Delta W \leftarrow \Delta W(I - P_{S_m})$, where P_{S_m} is the orthogonal projector onto S_m. (i) If every task adds at least one direction outside the current protected subspace while one remains, then $\dim S_m \geq m$ for $m \leq d$, the free dimension $d - \dim S_m$ is at most $d - m$, and once $\dim S_m = d$, which happens after at most d such tasks, the projected update is zero and, for a fixed input, the layer output can change only through its bias. (ii) A protected subspace of fixed rank $k < d$ leaves $d - k$ free directions for every m.*

Saturation is structural, since the input energy on which a projected update can act does not grow with the number of tasks (Corollary 1 in Appendix A), whether or not boundaries are known.

Persistent dormancy. The second mechanism (Figure 1b) concerns every hidden layer whose input is post-ReLU and whose protected subspace is closely aligned with the mean input direction, which carries a large share of the energy of non-negative inputs.

Proposition 2 (Mean-aligned projection suppresses the weight pathway of neuron deactivation and reactivation)**.** *Let $a_j = w_j^\top x + b_j$ be the pre-activation of neuron j, $\mu = \mathbb{E}[x]$, and ϕ the angle*

between μ and $\mathrm{span}(U)$. The sign of $\mathbb{E}[a_j] = w_j^\top \mu + b_j$ changes only through (i) the μ-component of w_j, (ii) a shift of μ caused by updates of earlier layers, or (iii) the bias. Projection with $\alpha = 1$ bounds the change of (i) along one update by $\|\Delta w_j\| \, \|\mu\| \sin\phi$ to first order, which removes it when $\mu \in \mathrm{span}(U)$, and reduces the bound on the weight-mediated part of (ii) when earlier layers are projected too. What remains is the residual $\sin\phi$ part of (i), the bias, which is not projected, and the part of (ii) that the biases and nonlinearities of earlier layers pass through.

The proposition concerns the signed mean pre-activation, which covers every unit that ReDo calls dormant (Appendix A). It predicts that, under protection without recycling, dormant neurons reactivate more slowly and remain dormant longer, that with recycling few neurons of the protected layers become dormant, since the weight pathway of deactivation is suppressed as well, and that removing the mean direction from the protected subspace or weakening the projection restores reactivation.

Output change on old inputs. Protection bounds this change as follows.

Proposition 3 (Output change on old inputs)**.** *For any update ΔW, the expected squared change of the layer output over the input distribution satisfies $\mathbb{E}\|\Delta W P_\alpha x\|^2 = \mathrm{tr}(\Delta W P_\alpha \Sigma P_\alpha \Delta W^\top) \le \|\Delta W\|_2^2 \, [(1-\alpha)^2 \lambda_1 k + \mathrm{tr}\,\Sigma \cdot \rho_k]$, and for $\alpha = 1$ the bound is $\|\Delta W\|_2^2 \, \mathrm{tr}\,\Sigma \cdot \rho_k$. The bound is the same for every task whose inputs share Σ, and it concerns one layer and one update.*

The equality shows that the output change depends on where ΔW lies relative to Σ, so an isotropic shrinkage of the update to the Frobenius norm of the projected update should not reproduce the effect of the projection, and neither should a random basis of the same rank, which leaves a larger ρ_k. Since the bound holds uniformly across tasks that share Σ, protection is expected to be effective where the protected directions are shared across tasks, a premise measured in Section 6.3.

5 Fixed-rank subspace protection with neuron recycling

FRSP+ReDo addresses each mechanism of Section 4 with one component, and applies recycling to every hidden layer, since dormancy is not confined to the protected layers. Let $x_\ell \in \mathbb{R}^{d_\ell}$ be the input of layer ℓ and $\Sigma_\ell = \mathbb{E}[x_\ell x_\ell^\top]$.

Fixed-rank subspace protection. We maintain an orthonormal basis $U_\ell \in \mathbb{R}^{d_\ell \times k}$ that approximately spans the top-k eigenspace of an exponential moving average (EMA) of Σ_ℓ with a half-life of h updates. During a warm-up of t_0 updates the EMA is kept explicitly and eigendecomposed, and afterwards the basis is tracked by a streaming power iteration, $U_\ell \leftarrow \mathrm{QR}\big(\beta U_\ell + (1-\beta)\,\mathrm{colnorm}(\hat{\Sigma}_\ell U_\ell)\big)$ with $\beta = 2^{-1/h}$ and $\hat{\Sigma}_\ell U_\ell = X^\top (X U_\ell)/n$ for the current mini-batch $X \in \mathbb{R}^{n \times d_\ell}$, where colnorm scales every column to unit norm and QR re-orthonormalizes. On CN-ImageNet schedules 0 to 4, FRSP+ReDo instead eigendecomposes the EMA exactly, which is level with the tracking in P (Appendix C). After every optimizer step the realized update is projected,

$$\Delta W_\ell \leftarrow \Delta W_\ell - \alpha \, (\Delta W_\ell U_\ell) \, U_\ell^\top, \tag{1}$$

with $\alpha = 1$ unless stated. Biases are not projected, and convolutional weights are reshaped to (out $\times$ in$\cdot k_h \cdot k_w$) with x_ℓ the unfolded patch. A fixed rank moves the protected subspace instead of enlarging it, so $d_\ell - k$ free directions remain at every stream length (Proposition 1). The protected layers are always the leading layers of the neural network, from the input onwards. On the ConvNet the development stream (Section 3) selects the three convolutional layers at $k = 2$, and we use $k = 4$, which is level with $k = 2$ on the main stream (Appendix C). Section 6.3 relates the protected layers to the overlap between tasks.

Neuron recycling. Every T updates we apply the ReDo rule to every hidden layer, protected or not. A hidden neuron whose mean absolute activation on the current mini-batch, divided by the layer mean, is at most τ is reinitialized, with fresh input weights and bias and zeroed outgoing weights and Adam moments, which leaves the outputs nearly unchanged at that moment. A recycled neuron is then trained under the same projection as the other neurons of its layer.

Cost and hyperparameters. Neither the cost per update of FRSP+ReDo nor its extra state grows with the stream, unlike the state of GPM. The learning rate, weight decay, T and τ are those selected

Table 1: Plasticity P, first-batch recall R_0, effective rank and dormant fraction on CN-ImageNet with the ConvNet, grouped by the two mechanisms. Bold marks values within 0.005 of the best.

Method	P	R_0	rank / dorm.
Fixed-rank protection			
FRSP+ReDo	**1.410**	0.053	21.7 / 0.014
FRSP with self-normalized resets	**1.414**	0.050	21.9 / 0.000
FRSP only (no recycling)	0.994	0.082	7.2 / 0.520
Recycling without protection			
ReDo	1.383	0.043	17.9 / 0.084
Self-normalized resets	1.406	0.041	16.1 / 0.002
CBP	1.374	0.044	22.6 / 0.088
Accumulating protection			
OWM + ReDo	1.112	0.134	36.2 / 0.000
OWM	1.019	**0.145**	20.6 / 0.351
GPM	0.301	0.106	2.1 / 0.722
No protection, no recycling			
AdamW	1.114	0.066	8.9 / 0.584

for ReDo on CN-ImageNet, and the learning rate and rank are selected on the development stream of CN-Procgen (Appendix C).

6 Experiments

Section 6.1 tests the prediction of Section 4 on saturation, Section 6.2 those on persistent dormancy and the complementarity of the two components, and Section 6.3 the layers to protect.

Protocol. The architectures are the ConvNet of Figure 1, conv1 to conv3 followed by fc1, fc2 and the head, and a 3072-300-150 multilayer perceptron (MLP) with the hidden layers of Elsayed & Mahmood (2024). All learners except UPGD use AdamW (Loshchilov & Hutter, 2019) on CN-ImageNet and Adam on CN-Procgen (Appendix C). Projection methods that build or relax their subspace at task boundaries are represented by OWM and GPM, applied on a fixed period in place of boundaries and also with the true boundaries. Every method is selected on the development stream and then frozen, except where stated (Table 3, Appendix C). A schedule is one random draw of the task order and the segment lengths, and each learner is trained on schedules 0 to 4 with three initializations each (one on CN-Procgen, Appendix E). Aggregates are interquartile means over these 15 schedule-initialization pairs (Agarwal et al., 2021), and schedules 5 to 9, never used for selection, are added where stated to test the selections on unseen task orders. Two learners are compared pair by pair, which removes the variation that the task order causes in both, and we report on how many pairs the difference is positive. Differences within 0.005 are treated as ties (Appendix B). In P, FRSP+ReDo is level with self-normalized resets and above every other baseline against loss of plasticity on 15 of 15 pairs (Table 1 and Appendix D.5).

6.1 Saturation of accumulating protection

Proposition 1 predicts that a protected subspace whose dimension grows with the stream leaves fewer and fewer free directions for new learning. On CN-ImageNet (Figure 3), FRSP+ReDo and ReDo stay above the restart oracle, whereas the task-free GPM, whose basis only grows, collapses with most of its neurons dormant, and the task-free OWM declines steadily. The protected dimension shows the cause, since the basis of GPM covers most of the input dimension of every layer within 50 segments and the protected dimension of OWM grows through the stream (Figure 6 in Appendix D.1). Recycling does not remove the saturation. With ReDo, OWM keeps almost no dormant neurons and a higher effective rank than FRSP+ReDo and ReDo (Table 1), yet its online accuracy declines and falls below the restart oracle in the last quarter of the stream, while its protected subspace grows to cover the whole input space of the linear layers and the head. With recycling, saturation is therefore not visible in the dormant fraction or the effective rank. Given the true task boundaries and no recycling, OWM and GPM also remain far below FRSP+ReDo (Table 5 in Appendix D.1), and on a stream of 1200 segments the online accuracy of OWM keeps falling whereas that of FRSP+ReDo stays constant. What causes saturation is a protected dimension with-

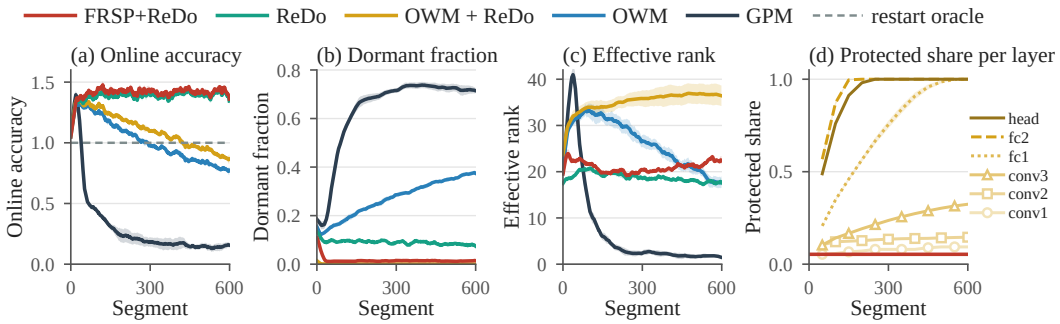


Figure 3: Hidden saturation on CN-ImageNet. (a) Online accuracy relative to the restart oracle, (b) dormant fraction, (c) effective rank and (d) share of the input directions protected by OWM with recycling, against the fixed rank of FRSP+ReDo.

out bound, not accumulation itself. When the tracked basis of FRSP+ReDo is accumulated up to a small cap per convolutional layer, P stays close to that of the tracked basis, although slightly below it on every pair, and with all layers protected, capped accumulation is above a tracked rank of the same size (Table 5 and Figure 7 in Appendix D).

6.2 Persistent dormancy and the complementarity of the two components

Proposition 2 predicts that protection preserves dormancy and that recycling is therefore needed. In a two-by-two factorial comparison of protection and recycling with the same hyperparameters (Table 2), protection without recycling leaves about half of the neurons dormant, nearly as many as AdamW, recycling without protection leaves under a tenth dormant, and the combination fewer still. The dormant fraction understates the effect of protection on dormancy. Without recycling, the dormant fraction of conv3 under protection is below that of AdamW early in the stream and above it later, and dormant periods in the protected layers, runs of consecutive segments in which a neuron is dormant, last longer than under AdamW (Figure 4). The recycling main effect accounts for most of the gain in P, the protection main effect is slightly negative, and their interaction is positive on 10 of 10 schedules, so protection raises P only together with recycling. FRSP+ReDo is above ReDo in P on 30 of 30 pairs and in first-batch recall on 29 of 30 (Table 11 in Appendix D.5). On first-batch recall the protection main effect is positive and the recycling main effect negative on 10 of 10 schedules, so recycling lowers recall with or without protection, and neither component replaces the other. Protection also raises P and R_0 of self-normalized resets, on 29 of 30 pairs each, whereas on the MLP it leaves almost no neuron dormant and recycling adds nothing (Table 8 in Appendix D.4), so the complementarity holds for a second recycling rule but only on the ConvNet.

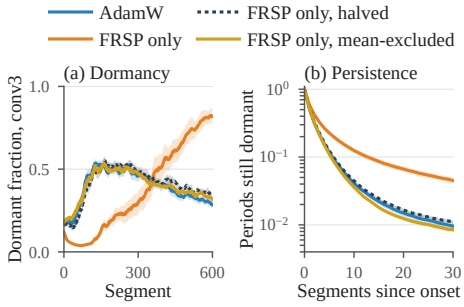


Figure 4: Persistent dormancy without recycling on CN-ImageNet. (a) Dormant fraction of conv3 and (b) share of the dormant periods of conv2 and conv3 still dormant after onset, under full, halved ($\alpha = 0.5$) and mean-excluded protection and under AdamW, without weight decay.

Where recycling acts. Under FRSP+ReDo the protected convolutional layers show almost no dormancy and few reinitializations, whereas ReDo reinitializes many neurons in conv3, and the dormancy that remains under FRSP+ReDo lies in the unprotected fc1, where recycling reinitializes the dormant neurons (Figure 5c,d). That layer is not protected, so Proposition 2 constrains there only the shift of its input mean and does not by itself account for the dormancy that recycling removes. Recycling only the linear layers matches recycling everywhere, whereas recycling only the protected layers leaves most of fc1 dormant (Appendix D.2).

Table 2: Two-by-two comparison of protection and recycling on CN-ImageNet, with main effects and interaction. Cells are means, so that the effects add up, all without weight decay, and parentheses count the schedules with a positive effect.

Method	Protection	Recycling	P	R_0	dorm.
AdamW	×	×	1.054	0.066	0.606
FRSP only	✓	×	1.002	**0.080**	0.537
ReDo	×	✓	1.385	0.041	0.084
FRSP+ReDo	✓	✓	**1.412**	0.050	**0.015**
Effect of recycling			+0.371 (10)	−0.028 (0)	−0.523 (0)
Effect of protection			−0.012 (0)	+0.011 (10)	−0.069 (0)
Interaction			+0.079 (10)	−0.005 (2)	0.000 (4)

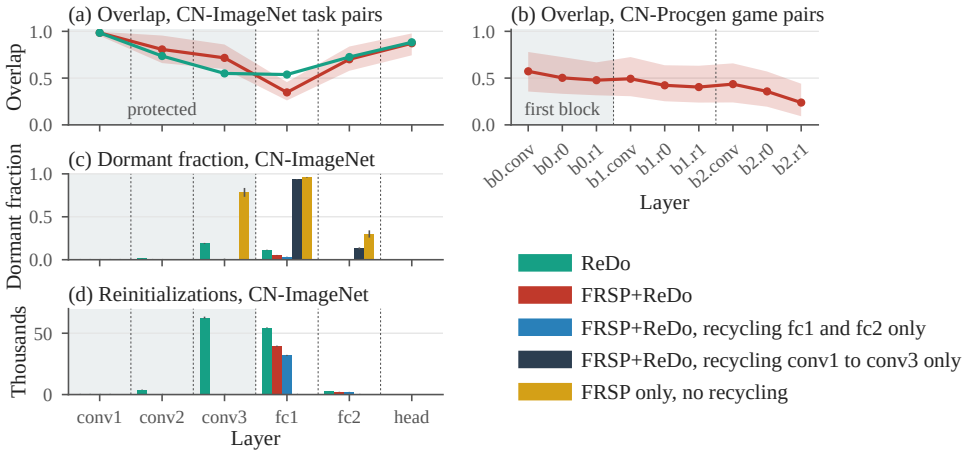


Figure 5: Where protection and recycling act. (a, b) Overlap between the top-four input subspaces of two tasks per layer on CN-ImageNet and CN-Procgen. (c) Dormant fraction and (d) reinitializations per layer on CN-ImageNet. Shading marks the protected layers, in (b) those of the first-block variant.

Dormancy passes through the mean direction. On the weights learned by protection without recycling, the protected subspace contains the mean input direction almost entirely in conv2 and conv3. Halving the projection brings the dormant periods close to those of AdamW (Figure 4b), so persistent dormancy requires the full projection. Removing the mean direction from the protected subspace does the same and raises P without recycling, which is consistent with dormancy passing through the mean direction. This control also weakens the projection, however, and raising the rank of the mean-excluded subspace eightfold does not restore its strength (Table 6 in Appendix D.2).

The protected directions matter. The two predictions of Proposition 3 hold (Table 7 in Appendix D.2). Against an energy-matched isotropic shrinkage, protection gains in P and R_0 on 15 of 15 pairs, and with recycling the isotropic control stays within 0.005 of ReDo, so the effect of protection comes from its directions. A fixed random orthonormal basis of the same rank is below the tracked basis in P on 15 of 15 pairs and in recall on 14 of 15, so the benefit comes from the choice of directions rather than from a low-rank constraint.

6.3 Protection helps where input directions are shared across tasks

Proposition 3 bounds the output change uniformly only across tasks that share their input statistics, so we ask in which layers protection is effective, on CN-ImageNet and on CN-Procgen. We measure the premise as the overlap $\|U_a^\top U_b\|_F^2/k$ between the top-four input subspaces of a layer under two tasks, on the weights of the trained neural network (Figure 5).

Protection ends where the overlap falls. On CN-ImageNet, the overlap on the weights learned by FRSP+ReDo is high in the convolutional layers and falls at fc1. On the weights learned by ReDo, which protects nothing, it falls one layer earlier, and at initialization it is uniformly high

Table 3: Plasticity on CN-Procgen over ten schedules, as normalized return overall, on the first appearances of games (New game) and of level buckets (New levels) and on revisits. Bold marks values within 0.005 of the best. Further baselines are in Appendix E (Table 12).

Method	P	New game	New levels	Revisit	rank / dorm.
PPO (matched learning rate)	0.795	0.644	0.796	0.846	63.7 / 0.236
ReDo	**0.912**	0.623	1.007	**0.980**	99.3 / 0.112
FRSP+ReDo (whole trunk)	0.855	0.558	0.917	0.936	101.4 / **0.075**
FRSP+ReDo (first block only)	0.906	**0.688**	**1.030**	0.939	**101.5** / 0.097

(Appendix D.3), so the profile is a property of the trained neural network and partly reflects the protection. The layers selected on the development stream end before this drop, whereas protecting only conv1 and conv2, the layers before the earlier drop on the weights of ReDo, gives a lower P than the selected layers, so the overlap describes the selection rather than predicting it. Protecting fewer leading layers lowers P on 15 of 15 pairs, and so does extending protection to fc2 alone or to both linear layers (Figure 7 in Appendix D.3), although fc2 and the head also have a high overlap.

Protecting less shared layers costs plasticity. On CN-Procgen, the learner is proximal policy optimization (PPO, Schulman et al., 2017) with the convolutional neural network of the importance weighted actor-learner architecture (IMPALA, Espeholt et al., 2018), whose trunk has three blocks of five convolutional layers, and FRSP+ReDo protects the trunk and leaves the policy and value heads unprotected (Appendix E). Here the overlap is highest in the first block. Protecting the whole trunk, the setting selected on the development stream, yields the lowest dormant fraction but is below ReDo in normalized return on 8 of 10 schedules, with the deficit largest on the first appearances of games and of level buckets (Table 3). The deficit is not caused by dormant neurons or a loss of effective rank, since this learner has the lowest dormant fraction and an effective rank second only to the first-block variant. Protecting only the first block removes most of the deficit. This setting was identified on schedules 0 to 4 of this stream, whereas the development stream selects the whole trunk. Its normalized return, overall and on new games, is higher than that of ReDo on 6 of 10 schedules each, so the two are not separated, and on the held-out schedules 5 to 9 it stays between ReDo and the whole trunk. Protecting layers whose input directions are less shared across tasks therefore reduces plasticity on both streams. The overlap does not predict the gain of FRSP+ReDo over ReDo at individual task changes on either benchmark (Appendix E).

7 Conclusion

Subspace protection is an established approach to reducing forgetting, but on long task-free streams with recurring tasks it can itself reduce plasticity, through two mechanisms. Protection whose dimension grows without bound tends to saturate, and on our streams recycling did not prevent it. Protection aligned with the mean input direction can suppress one of the main pathways through which dormant neurons reactivate. A fixed rank and an established recycling rule address these mechanisms, and on the ConvNet neither component replaced the other, with recycling acting mainly in the unprotected layers that follow the protected ones and protection helping mainly in the leading layers whose input directions are shared across tasks. These results suggest that subspace protection is best kept bounded, paired with neuron recycling and applied selectively, and that forgetting and loss of plasticity may be better studied together. Neither mechanism is specific to FRSP+ReDo. Saturation follows from accumulation alone, and the second cost from alignment with the mean input direction, a property of the tracked basis that projections built from uncentered second moments should share. The prescription itself costs little, since neither the cost per update nor the extra state of FRSP+ReDo grows with the stream.

The complementarity was established only on the ConvNet, since on the MLP protection alone left few dormant neurons and on CN-Procgen protecting the whole trunk lowered dormancy without raising the return, so which settings need both components remains open. The overlap was measured after training, and selecting the protected layers in advance is left to future work. FRSP+ReDo does not guarantee memory either, as its first-batch recall stays below that of AdamW. Normalized residual neural networks also remain untested on the supervised streams, where recycling may also need to reset the normalization parameters.

AI USE STATEMENT

In this work we used generative AI tools (Claude, Anthropic) for tasks with required disclosure, namely proposing and refining hypotheses, designing and giving feedback on experiments, implementing the methods and baselines, helping to formulate the propositions and assisting in the writing of their proofs, interpreting results, and translating the comments of the released code into English. We did not use generative AI tools to generate synthetic data sets, to clean or reformat data sets or to support qualitative data analysis, and the remaining tasks with required disclosure do not apply to this work. We also used generative AI tools for tasks with recommended disclosure, namely drafting and editing parts of the text, creating and editing analysis code and figures, and searching and summarizing the literature. All AI-assisted work was reviewed by the authors, who checked the proofs and tested AI-generated code before use, and every number in the paper is produced by scripts in the code of the supplementary material from the training logs and rechecked by a separate verification script. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies how subspace protection affects plasticity in continual learning, on the publicly available ImageNet-32 data set and the Procgen game suite. It involves no human subjects and no personal data, and we see no ethical concern beyond those common to machine learning research.

REPRODUCIBILITY STATEMENT

The proofs of the propositions are in Appendix A, and the benchmarks, the definition of every metric and the statistical protocol in Appendix B. Appendix C gives the selection protocol, the hyperparameters of FRSP+ReDo and ReDo and every constant of the baselines that is not taken unchanged from the original method, the supplementary material lists the searched and selected hyperparameters of every other method, and Appendices D and E give the complete tables behind every number of the main text. The supplementary material contains the code for the benchmarks, FRSP+ReDo and all baselines, the scripts that produce every table and every data figure from the training logs that this code writes, and the two scripts that check every number of the paper against them.

REFERENCES

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C. Machado. Loss of plasticity in continual deep reinforcement learning. In *CoLLAs*, 2023.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*, 2021.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *CVPR*, 2019.

Jordan T. Ash and Ryan P. Adams. On warm-starting neural network training. In *NeurIPS*, 2020.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of ImageNet as an alternative to the CIFAR datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Wesley Chung, Lynn Cherif, David Meger, and Doina Precup. Parseval regularization for continual reinforcement learning. In *NeurIPS*, 2024.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *ICML*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024.

Mohamed Elsayed and A. Rupam Mahmood. Addressing loss of plasticity and catastrophic forgetting in continual learning. In *ICLR*, 2024.

Mohamed Elsayed, Qingfeng Lan, Clare Lyle, and A. Rupam Mahmood. Weight clipping for deep continual and reinforcement learning. In *RLC*, 2024.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *ICML*, 2018.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *AISTATS*, 2020.

Vivek F. Farias and Adam D. Jozefiak. Self-normalized resets for plasticity in continual learning. In *ICLR*, 2025.

Hamed Hemati, Andrea Cossu, Antonio Carta, Julio Hurtado, Lorenzo Pellegrini, Davide Bacciu, Vincenzo Lomonaco, and Damian Borth. Class-incremental learning with repetition. In *CoLLAs*, 2023.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Böhmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In *ICLR*, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

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

Yajing Kong, Liu Liu, Zhen Wang, and Dacheng Tao. Balancing stability and plasticity through advanced null space in continual learning. In *ECCV*, 2022.

Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. Maintaining plasticity in continual learning via regenerative regularization. In *CoLLAs*, 2024.

Alex Lewandowski, Michał Bortkiewicz, Saurabh Kumar, András György, Dale Schuurmans, Mateusz Ostaszewski, and Marlos C. Machado. Learning continually by spectral regularization. In *ICLR*, 2025.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. TRGP: Trust region gradient projection for continual learning. In *ICLR*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *ICLR*, 2022.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *ICML*, 2023.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks. In *CoLLAs*, 2024.

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

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *ICML*, 2022.

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. In *NeurIPS*, 2023.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. In *NeurIPS*, 2019.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *EUSIPCO*, 2007.

Gobinda Saha and Kaushik Roy. Continual learning with scaled gradient projection. In *AAAI*, 2023.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *ICLR*, 2021.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & Compress: A scalable framework for continual learning. In *ICML*, 2018.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *ICML*, 2023.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *CVPR*, 2021.

Lin Xiong, Junnan Li, and Hailing Xiong. Targeting plasticity loss in continual learning via null-space orthogonal reorganization. *ICT Express*, 2026. In press.

Wentao Yang, Wenxin Liang, Yunlong Gao, Xinyue Liu, Linlin Zong, Guanglu Wang, and Xianchao Zhang. Gradient-aware balanced neuron resetting: Bridging plasticity and stability for sample-efficient deep reinforcement learning. In *ECML PKDD*, 2026.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

Zhen Zhao, Zhizhong Zhang, Xin Tan, Jun Liu, Yanyun Qu, Yuan Xie, and Lizhuang Ma. Rethinking gradient projection continual learning: Stability / plasticity feature space decoupling. In *CVPR*, 2023.

Weibang Zhou, Xiangsheng Huang, Xiao Yang, and Anfu Huang. Gradient-coupled continual backpropagation: Dynamic plasticity control for nonstationary data streams. *Evolving Systems*, 17(3): 108, 2026.

A PROOFS

This appendix proves the propositions of Section 4. Throughout, $x \in \mathbb{R}^d$ is the input of one hidden layer, $\Sigma = \mathbb{E}[xx^\top] \succeq 0$ is its second moment with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$, $U \in \mathbb{R}^{d\times k}$ has orthonormal columns, $P_\alpha = I - \alpha UU^\top$ and $\rho_k = 1 - \operatorname{tr}(U^\top \Sigma U)/\operatorname{tr}\Sigma$. For a subspace $S \subseteq \mathbb{R}^d$, P_S is the orthogonal projector onto S, $\|\cdot\|$ is the Euclidean norm of a vector and $\|\cdot\|_2$ the spectral norm of a matrix. Proposition 3 is proved before Proposition 2, whose proof uses it.

A.1 SATURATION

Proof of Proposition 1. (i) While $S_{m-1} \neq \mathbb{R}^d$, $S_{m-1} \subseteq S_m$ and S_m contains a direction outside S_{m-1}, so $\dim S_m \geq \dim S_{m-1} + 1$, and by induction from $S_0 = \{0\}$

$$\dim S_m \geq m \qquad \text{and} \qquad d - \dim S_m \leq d - m \qquad \text{for } m \leq d. \tag{A.1}$$

Since $\dim S_m \leq d$, the protected subspace equals $\mathbb{R}^d$ after at most d tasks. Then $P_{S_m} = I$, so for every update ΔW and every input x

$$\Delta W\,(I - P_{S_m}) = 0 \qquad \text{and} \qquad \Delta W\,(I - P_{S_m})\,x = 0, \tag{A.2}$$

and for a fixed input the output $Wx + b$ of the layer can change only through b. (ii) For a fixed rank, $I - UU^\top$ is the orthogonal projector onto the orthogonal complement of $\operatorname{span}(U)$, which has dimension $d - k$ for every m. □

Corollary 1 (Learnable input energy)**.** *Let $\Sigma \succeq 0$ be the second moment of the input of the layer and P_{S_m} the projector of Proposition 1. The learnable input energy*

$$E_m = \operatorname{tr}\big((I - P_{S_m})\,\Sigma\big), \tag{A.3}$$

the part of the input energy on which a projected update can still act, is non-increasing in m, and it decreases strictly at task m whenever a direction added to S_m carries energy under Σ. Under protection of a fixed rank k, the same quantity equals $\rho_k \operatorname{tr}\Sigma$ and is at least $\sum_{i>k}\lambda_i$, the sum of the $d-k$ smallest eigenvalues of Σ, at every task.

Proof. Since $S_{m-1} \subseteq S_m$, the difference $Q_m = P_{S_m} - P_{S_{m-1}}$ is the orthogonal projector onto $S_m \cap S_{m-1}^\perp$, the directions added at task m. With an orthonormal basis $\{q_j\}$ of this subspace,

$$E_m - E_{m-1} = -\operatorname{tr}(Q_m\Sigma) = -\sum_j q_j^\top \Sigma\, q_j \leq 0, \tag{A.4}$$

with equality if and only if every q_j lies in the null space of Σ, because $\Sigma \succeq 0$. For a fixed rank, $\operatorname{tr}((I-UU^\top)\Sigma) = \operatorname{tr}\Sigma - \operatorname{tr}(U^\top\Sigma U) = \rho_k \operatorname{tr}\Sigma$ by the definition of ρ_k, and $\operatorname{tr}(U^\top\Sigma U) \leq \sum_{i\leq k}\lambda_i$ for every U with k orthonormal columns by the Poincaré separation theorem, so

$$\rho_k \operatorname{tr}\Sigma \;\geq\; \operatorname{tr}\Sigma - \sum_{i\leq k}\lambda_i = \sum_{i>k}\lambda_i. \tag{A.5}$$

□

Remark (Accumulating baselines). GPM adds to S_m every direction needed to retain a threshold fraction of the input energy of task m, with a threshold raised at every basis update up to 0.999, and never removes one, so its subspaces are nested as Proposition 1 assumes and the energy left to later tasks cannot rise. OWM approaches the same limit through its recursive least-squares projector. In the task-free GPM and OWM, the fixed refresh period plays the part of the task (Appendix C).

A.2 OUTPUT CHANGE ON OLD INPUTS

Proof of Proposition 3. Fix ΔW and draw x from the input distribution of the layer. Since P_α is symmetric, $\|\Delta W P_\alpha x\|^2 = \operatorname{tr}(\Delta W P_\alpha\, xx^\top P_\alpha \Delta W^\top)$, and taking the expectation gives

$$\mathbb{E}\|\Delta W P_\alpha x\|^2 = \operatorname{tr}(\Delta W P_\alpha \Sigma P_\alpha \Delta W^\top) \leq \|\Delta W\|_2^2 \operatorname{tr}(P_\alpha \Sigma P_\alpha), \tag{A.6}$$

where the inequality uses $\operatorname{tr}(AMA^\top) = \operatorname{tr}(MA^\top A) \leq \|A^\top A\|_2 \operatorname{tr} M$ for $M = P_\alpha\Sigma P_\alpha \succeq 0$. Since $P_\alpha^2 = I - \alpha(2-\alpha)\,UU^\top$, the trace factor is, for every orthonormal U,

$$\operatorname{tr}(P_\alpha\Sigma P_\alpha) = \operatorname{tr}\Sigma - \alpha(2-\alpha)\operatorname{tr}(U^\top\Sigma U) = \operatorname{tr}\Sigma\,\big[(1-\alpha)^2(1-\rho_k) + \rho_k\big] \tag{A.7}$$

by the definition of ρ_k. Every column of U satisfies $u_i^\top \Sigma u_i \le \lambda_1$, so $(1-\rho_k)\operatorname{tr}\Sigma = \operatorname{tr}(U^\top \Sigma U) \le k\lambda_1$, and (A.6) and (A.7) give

$$\mathbb{E}\|\Delta W P_\alpha x\|^2 \le \|\Delta W\|_2^2 \left[(1-\alpha)^2 \lambda_1 k + \operatorname{tr}\Sigma \cdot \rho_k\right]. \tag{A.8}$$

For $\alpha = 1$ the first term vanishes and (A.7) gives $\operatorname{tr}(P_1 \Sigma P_1) = \operatorname{tr}\Sigma \cdot \rho_k$ exactly. The bound depends on the inputs only through Σ, which gives the last claim. □

Remark (Tracked and exact bases). The bound holds for the tracked basis of FRSP as it is, with ρ_k computed from that basis. The tracking affects only the size of ρ_k, which is smallest when U spans the top-k eigenspace of Σ, where $\rho_k \operatorname{tr}\Sigma = \sum_{i>k} \lambda_i$ (Corollary 1). Replacing the tracked basis by the exact top-k eigenspace of the same EMA changes P of FRSP+ReDo on CN-ImageNet by 0.001 (Appendix C).

A.3 Persistent dormancy

Proof of Proposition 2. Along one update, w_j and b_j change by Δw_j and Δb_j, and μ changes by $\Delta\mu$ through the updates of the earlier layers, so that exactly

$$\Delta\,\mathbb{E}[a_j] = \underbrace{\Delta w_j^\top \mu}_{\text{(i)}} + \underbrace{w_j^\top \Delta\mu}_{\text{(ii)}} + \underbrace{\Delta b_j}_{\text{(iii)}} + \Delta w_j^\top \Delta\mu, \tag{A.9}$$

where the last term is of second order and is dropped. The three remaining terms are the pathways (i) to (iii) of the proposition, and we bound each in turn.

Pathway (i). Decompose $\mu = UU^\top \mu + r$ with $r \perp \operatorname{span}(U)$, so that $\|r\| = \|\mu\| \sin\phi$. With $\alpha = 1$ the update that is applied is the projected $\Delta w_j^\top (I - UU^\top)$, which we keep calling Δw_j, so $\Delta w_j^\top U = 0$ and hence

$$|\Delta w_j^\top \mu| = |\Delta w_j^\top r| \le \|\Delta w_j\|\,\|r\| = \|\Delta w_j\|\,\|\mu\| \sin\phi, \tag{A.10}$$

against $\|\Delta w_j\|\,\|\mu\|$ without projection. The bound vanishes when $\mu \in \operatorname{span}(U)$.

Pathway (ii). Let the previous layer be linear with pre-activation $a' = W'x' + b'$, so that $x = \operatorname{ReLU}(a')$ and $\mu = \mathbb{E}[\operatorname{ReLU}(a')]$. Since the derivative of the ReLU is 0 or 1, the change of μ that the update $\Delta W'$ of the previous layer causes is, to first order, $\mathbb{E}[\mathbf{1}[a' > 0] \odot \Delta W' x']$, and by Jensen's inequality

$$\left\|\mathbb{E}[\mathbf{1}[a' > 0] \odot \Delta W' x']\right\| \le \mathbb{E}\|\Delta W' x'\| \le \left(\mathbb{E}\|\Delta W' x'\|^2\right)^{1/2}. \tag{A.11}$$

When the previous layer is projected with $\alpha = 1$, its update is $\Delta W' P_1'$ and Proposition 3 bounds the right side by $\|\Delta W'\|_2 \,(\operatorname{tr}\Sigma'\,\rho_k')^{1/2}$, against $\|\Delta W'\|_2\,(\operatorname{tr}\Sigma')^{1/2}$ without projection, where Σ' and ρ_k' are the second moment and the unprotected energy fraction of the previous layer, a reduction of the bound by the factor $(\rho_k')^{1/2}$. The changes of μ that the bias $\Delta b'$ of the previous layer causes, and those that the layers before it cause through $W'\Delta x'$, are not affected by this projection and pass through. In every case $|w_j^\top \Delta\mu| \le \|w_j\|\,\|\Delta\mu\|$. For a convolutional layer, x is a patch of the max-pooled output of the previous layer. Max pooling and unfolding are 0 or 1 selections as well, so the same argument bounds $\|\Delta\mu\|$ up to a constant that depends only on the architecture, and the reduction by $(\rho_k')^{1/2}$ is unchanged.

Pathway (iii). The bias Δb_j is not projected and remains.

Hence, to first order, a change of sign of $\mathbb{E}[a_j]$ can come from the bias, from the residual $\sin\phi$ part (A.10) of pathway (i), or from the part of $\Delta\mu$ that the biases and nonlinearities of the earlier layers pass through. The bounds do not depend on the direction of the change, so the weight pathway of deactivation is suppressed as well. □

Remark (Scope). A unit that ReDo calls dormant, one whose mean activation after the ReLU is at most τ times the layer mean, has a mean pre-activation no larger than that, since $\operatorname{ReLU}(a) \ge a$, so the proposition covers the weight pathway of every such unit. Reactivation through the variance of the pre-activation, with no change of its mean, lies outside its scope. The proposition concerns one update at a fixed input distribution. On the stream, μ also changes when the task changes, a change that no projection constrains and that the paired comparisons of Section 6.2 share across learners.

Remark (Measured alignment). In the second and third convolutional layers of the ConvNet the measured alignment $\cos^2\phi = \|U^\top \mu\|^2/\|\mu\|^2$ is 1.00 to two decimals (Appendix D.2), so $\sin\phi \le \sqrt{0.005} < 0.071$ and the bound (A.10) on pathway (i) is below $0.071\,\|\Delta w_j\|\,\|\mu\|$ per update.

B BENCHMARKS, METRICS AND STATISTICS

CN-ImageNet. The stream draws 150 tasks of five ImageNet-32 classes (class groups 0 to 149), with 600 training and 100 test images per class. A schedule has 600 segments, every task once as novel and 450 revisits drawn uniformly from the tasks already seen, with segment lengths log-uniform in $[12\text{k}, 48\text{k}]$ samples and mini-batches of 32 in order, which gives a median gap of 37 to 42 segments between visits of a task over schedules 0 to 9. The lengths are random so that the number of updates does not reveal a segment boundary, as it would with segments of equal length, and log-uniform so that short segments, where the speed of adaptation matters, and long ones, where the final accuracy matters, have equal weight on a multiplicative scale, following the common practice of sampling a positive quantity that spans several multiples on a logarithmic scale (Bergstra & Bengio, 2012). The ConvNet is that of Dohare et al. (2024), with 32, 64 and 128 channels in conv1 to conv3 and 128 units in fc1 and fc2. Online accuracy is measured on the stream, and held-out accuracy on all seen tasks every 25 segments. The development stream has 50 tasks of held-out classes (groups 150 to 199) and 300 segments with the same length distribution, with three schedules per configuration.

CN-Procgen. The stream draws twelve Procgen games in easy mode (bigfish, caveflyer, chaser, climber, coinrun, dodgeball, fruitbot, jumper, leaper, maze, miner, plunder), each split into three buckets of 200 levels, a pool of 36 tasks of which 24 appear in each schedule. A schedule has 60 segments, 24 first appearances (12 of a game and 12 of a further level bucket of a game already seen) and 36 revisits, about 68M environment steps in all. Each of 64 parallel environments switches at its own episode end within a window of 5,000 steps, so no boundary is visible. The development stream has four held-out games (bossfight, heist, ninja, starpilot) and twelve segments, and Appendix C describes how its schedules are used.

Normalized score. A is an accuracy on CN-ImageNet and a return on CN-Procgen, and A_{rand} is chance (0.2) or the return of a random policy. The normalized score of a segment is $s = (A - A_{\text{rand}})/(A^{\text{R}} - A_{\text{rand}})$, where A^{R} is the end-of-segment value of the restart oracle, which reinitializes the neural network at every boundary. A segment is excluded when the denominator is below 5% of its median or twice the standard error of A_{rand}.

Plasticity. P, P_{novel} and P_{revisit} are the means of the end-of-segment s, from A over the last quarter of the segment, over all, novel and revisited segments. ΔP is the mean end-of-segment s of the novel segments in the last 20% of the stream minus that in the first 20% (on CN-Procgen, the first appearances of a game in 30% windows), and forward transfer (FT) is the mean s over the first 500 samples of novel segments.

Memory. First-batch recall is $R_0 = \sum(A_{\text{first}} - A_{\text{rand}})/\sum(A_{\text{prev}} - A_{\text{rand}})$ pooled over revisits, where A_{first} is the accuracy on the first mini-batch of a revisit before any update (on CN-Procgen, episodes that start in the first 3% of the segment) and A_{prev} the end-of-segment accuracy of the previous visit of the task. A_{320}, A_{1024}, A_{3200} and A_{10000} are the accuracy on the first 320, 1024, 3200 and 10,000 samples of a revisit, averaged over revisits and then over schedules and initializations. ACC is the held-out accuracy on all seen tasks at the last evaluation, backward transfer (BWT) the final accuracy minus that of the first evaluation after the last visit, and forgetting (F) the maximum minus the final held-out accuracy, each averaged over tasks.

Dormancy and effective rank. The dormant fraction is the share of hidden units, channels for convolutional layers, whose mean absolute activation on the test images of the current task, divided by the layer mean, is at most 0.025, the threshold that Sokar et al. (2023) use for deep Q-networks at the default replay ratio, averaged over the last third of the stream, and the effective rank (Roy & Vetterli, 2007) is the exponential of the entropy of the normalized singular values of the penultimate features on the same images, averaged over the same window. For dormant periods, activations are

recorded at every segment on the same images, and a unit is dormant in a segment when its mean absolute activation is at most 0.01 times the layer mean, the value of τ at which ReDo recycles on CN-ImageNet. A dormant period is a maximal run of consecutive segments in which a unit is dormant, a period that lasts to the end of the stream is censored, and the reactivation rate is the share of dormant units that are active again in the next segment. The share of periods still dormant t segments after onset is the Kaplan–Meier estimate $S(t) = \prod_{s \le t}(1 - d_s/n_s)$, where n_s is the number of periods that last at least s segments and d_s the number that end after exactly s, computed for each trained neural network over the periods of conv2 and conv3, the protected layers with rectified inputs.

Statistics. Aggregates are interquartile means over schedule-initialization pairs (Agarwal et al., 2021), except the effective rank, the dormant fraction and the accuracies A_{320} to A_{10000}, which are means. Curves over the stream are moving averages over 25 segments, or 15 for the dormant fraction of a single layer, with one standard error over schedules shaded. Bands of the subspace overlap span the 5th to 95th percentiles over task pairs. The data order within a schedule depends on the schedule alone, so the initializations of one schedule see the same sample order and are not independent. Every interval we report therefore resamples schedules rather than pairs, which gives a wider interval than one that treats the pairs as independent. Paired counts give the number of shared pairs on which the difference is positive. Differences within 0.005 are treated as ties, since a single schedule moves these values by more than that, and bold marks the best value in a column and any value within 0.005 of it. Every table uses schedules 0 to 4 with three initializations each, 15 pairs per method, unless stated otherwise. On CN-Procgen every learner is trained once per schedule, and Appendix E gives its aggregation. Schedules 5 to 9 are held out from every selection. Tables 11 and 2 and the CN-ImageNet correlation of Appendix E add the held-out schedules with three initializations each, 30 pairs per method, and every paired margin of Table 11 that the main text quotes keeps its sign on 10 of 10 schedules, with a schedule-clustered interval excluding zero.

C Hyperparameters and selection protocol

Selection on CN-ImageNet. Every method was selected on the development stream (Appendix B) and then frozen (Table 4 and the supplementary material). A configuration is scored by P, the interquartile mean over three development schedules. The learning rate is selected on the first 20% of the development stream, because the whole-stream score prefers the smallest learning rate, at which loss of plasticity does not yet appear within 300 segments, and the method constants are selected on the whole development stream at that learning rate. Selecting the constants on the first 20% as well, the early-window rule, gives other constants for seven baselines, and the whole-stream constants are higher by 0.015 to 0.069 in development P (3 of 3 schedules each). AdamW is also reported at learning rate 10^{-4} and at weight decay 0 on the ConvNet. The restart oracle is selected on raw end-of-segment accuracy. For ReDo and CBP, the seven and ten development configurations whose bootstrap interval in the selection window overlapped that of the early-window selection were also evaluated on main-stream schedules 0 to 2. None of the ReDo configurations was above the selected one in P (the closest at -0.010), and the only CBP configuration above the early-window selection ($+0.021$, 3 of 3) is the development selection of the rule above and is the configuration used in every ConvNet table.

FRSP+ReDo on CN-ImageNet. On the development stream, the convolutional layers are above all layers at every rank ($+0.015$ to $+0.064$, 3 of 3 schedules each) and rank four is above rank eight ($+0.013$, 3 of 3). Rank two is above rank four by 0.006, on 2 of 3 schedules, so the rule selects rank two on the convolutional layers. We use rank four instead, an exception chosen on the main stream, where the two are level (P of 1.411 against 1.410 and R_0 of 0.050 against 0.053, Figure 7 and Table 10), so the headline values do not depend on this choice. The other constants of FRSP+ReDo are defaults of the implementation and were never selected on CN-ImageNet, $h = 32$k and $t_0 = 4$k updates, an eigendecomposition every 50 updates, 8 patch positions sampled per image for a convolutional layer, the rank cap $\lfloor d_\ell/4 \rfloor$ on convolutional layers and $\alpha = 1$. Each variant of FRSP+ReDo in Tables 5, 6, 7 and 10 differs from it only in what its row names. Both controls use the ordinary projection during the warm-up of t_0 updates. After it, the energy-matched isotropic control replaces the projected update by $\Delta W \sqrt{1 - \|\Delta W U\|_F^2 / \|\Delta W\|_F^2}$, which removes the same

energy without a preferred direction, and the fixed random basis is an orthonormal basis of the same rank that is drawn at the end of the warm-up and never updated.

Section 5 describes the streaming implementation of the protected basis. The exact implementation keeps the second-moment EMA of every protected layer for the whole stream and eigendecomposes it every 50 updates (a randomized singular value decomposition for the 3072-dimensional input of the MLP), with no warm-up. It is used by FRSP+ReDo on schedules 0 to 4, by FRSP+ReDo on all layers at $k = 2$, 4 and 8 and on conv1 to conv3 at $k = 2$, 8 and 16, by FRSP only on all layers at $k = 8$, by every protected learner on the MLP and by the development selection. Every other protected learner uses the streaming implementation, including FRSP+ReDo on the held-out schedules 5 to 9 and the learners whose neuron dynamics are recorded at every segment (Figures 4 and 5c,d, Appendix D.2), so the statistics over ten schedules combine the two. Paired by schedule, the streaming minus the exact implementation is -0.001 in P (positive on 1 of 5 schedules) and $+0.000$ in R_0 for FRSP+ReDo, -0.002 (2 of 5) for FRSP only on all layers at $k = 8$, both level. With one method alone on one graphics processing unit (GPU) over the first 200 segments of CN-ImageNet, the streaming implementation takes 4.11 ms per update against 2.28 for ReDo, 2.38 for AdamW, 3.59 for OWM and 3.50 for GPM, and its extra state, in float32 units divided by the parameter count and excluding optimizer moments, is 0.021, against 4.03 for OWM and 3.91 for GPM at segment 200, the only one that grows with the number of tasks. The exact one stores a second moment and a projector per protected layer, 4.73 times the parameter count for conv1 to conv3 and 8.05 times for all layers, a state that does not grow with the stream either.

Hyperparameter tables. Table 4 gives the hyperparameters of FRSP+ReDo and of ReDo. A set lists the values searched on the development stream of a setting, with the value used in bold, and a single value was not searched. Periods count updates, and PPO updates on CN-Procgen. The ConvNet and the MLP columns refer to CN-ImageNet. In the rows, lr is the learning rate and wd the weight decay, 0 unless a row states it and 10^{-2} on the MLP. The last row counts the development configurations. The searched grids and selected values of every other method are listed in the file configs/hyperparameter_grids.csv of the supplementary material, and configs/paper_runs.csv gives every configuration with its training command.

Table 4: Hyperparameters of FRSP+ReDo and of ReDo, from which FRSP+ReDo takes its learning rate, weight decay, period and threshold. ‡Exception chosen on the main stream. The anchoring penalty adds the quadratic term of S-EWC with weight λ to the loss, and on CN-Procgen the first block and the first two blocks were run at the constants selected for the whole trunk, so n counts the configurations of the whole trunk.

Hyperparameter	ConvNet	MLP	CN-Procgen
FRSP+ReDo			
protected layers	{**conv1–conv3**, all layers}	all layers	{first block, first two blocks, **whole trunk**}
rank k	{**2**, 4, 8, 16}, 4‡ used	{2, 4, **8**, 16}	{2, **4**, 8}
lr	that of ReDo	that of ReDo	{$\mathbf{2.5{\times}10^{-4}}$, $5{\times}10^{-4}$}
period T, threshold τ	those of ReDo	those of ReDo	those of ReDo
α	1	1	1
anchoring penalty λ	0	0	{**0**, 10, 100}
configurations n	8	4	18
ReDo			
lr	{$\mathbf{3{\times}10^{-4}}$, 10^{-3}}	{$\mathbf{10^{-4}}$, $3{\times}10^{-4}$, 10^{-3}}	{$\mathbf{2.5{\times}10^{-4}}$, $5{\times}10^{-4}$}
period T	{**250**, 1000, 4000}	250	{50, 100, **200**}
threshold τ	{**0.01**, 0.025, 0.1}	0.01	{0, **0.025**}
configurations n	18	3	12

Baseline implementations. Constants that the hyperparameter tables and the supplementary material do not list are those of the original methods, the contribution utility with decay 0.99 for CBP (Dohare et al., 2024), the first-order global variant with weight decay (UPGD-W) of Elsayed & Mahmood (2024), and the ReDo reinitialization of Section 5. S-EWC uses one decay β for the bias-corrected EMAs of the squared gradients and of the parameters instead of two, and shrink-and-perturb is applied after every update. OWM and GPM update their projection on a fixed period instead of at task boundaries, OWM from the batch-mean activations with its projector initialized to I, and GPM from the layer inputs of 256 samples with an energy threshold raised by 0.003 per basis update up to 0.999 and no direction removed. ER-1k appends 16 samples from a reservoir of 1,000 to every mini-batch.

The published description of self-normalized resets leaves two implementation choices open, and we state ours. A unit fires in an update when its activation over the mini-batch is not zero, a convolutional channel being averaged over the mini-batch and over positions, so time is counted in updates rather than in single examples. Counting single examples would reset active units continually, since at batch size 32 a rectified unit is exactly zero on many individual inputs while the layer as a whole is active. The mean inter-firing time of a unit is an EMA over the intervals that unit completes, with a half-life of 100 firing events, so a unit that has stopped firing keeps the estimate it had while it was still firing, which is the role of the trailing window in the original description. The reset itself is the ReDo rule, so the two recycling baselines differ only in when they reset a neuron.

Settings on the MLP. The MLP was selected by the same rule on its own development stream (Table 4 and the supplementary material), where Parseval and spectral regularization select the same constants on the first 20% and on the whole stream, and CBP keeps the replacement rate of the early-window selection on the ConvNet. The four ranks of FRSP+ReDo lie within 0.002 of each other in the selection window, with $k = 8$ highest, and within 0.007 on the whole development stream (Table 8 reports all four).

Settings on CN-Procgen. Every learner uses PPO (Schulman et al., 2017) with the IMPALA neural network (Espeholt et al., 2018), 64 environments, 256 steps per rollout, 3 epochs of 8 mini-batches, discount 0.999, generalized advantage estimation (Schulman et al., 2016) with $\lambda = 0.95$, clipping at 0.2 of the probability ratio and of the value update, advantages normalized per mini-batch, entropy coefficient 0.01, value coefficient 0.5, gradient norm clipped at 0.5 and Adam (Kingma & Ba, 2015) with $\epsilon = 10^{-5}$. The trunk of the IMPALA neural network has three blocks (b0 to b2 in Figure 5b), each a convolution followed by two residual units (r0, r1) of two convolutions. Rewards are divided by a running standard deviation of the discounted return, whose statistics are shared across tasks, and clipped to ± 10. Periodic reset and the restart oracle reinitialize the parameters, the optimizer state and these statistics. The development score is the normalized online return over the twelve development segments. Every configuration is scored on one development schedule and the three highest of each method also on a second one, every configuration of PPO, ReDo, FRSP+ReDo and the restart oracle on both, and the highest mean over two schedules is selected, except for the restart oracle, which is selected on raw end-of-segment return. FRSP+ReDo uses the streaming implementation with $h = 6{,}500$ and $t_0 = 1{,}000$ optimizer steps, without projection during the warm-up, 8 sampled positions per observation, rank $\min(k, \lfloor d_\ell/4 \rfloor)$ and unprotected policy and value heads. It takes the recycling constants of ReDo, which rank second of twelve ReDo configurations, 0.014 below period 200 with $\tau = 0$, within the spread between schedules. The first-block variant was identified on schedules 0 to 4 of the main stream. On the development stream, protecting the whole trunk scores 1.777, the first two blocks 1.611 and the first block 1.278, in this order on both schedules, so the development selection is the whole trunk, the FRSP+ReDo (whole trunk) row of Tables 3 and 12. There the whole trunk is also above ReDo (1.625), unlike on the main stream, so the twelve development segments do not show the deficit of the whole trunk against ReDo, and the first block is instead checked on the held-out schedules 5 to 9 (Appendix E).

D Additional results on CN-ImageNet

D.1 Saturation of accumulating protection

OWM keeps protecting more directions. OWM keeps the regularized projector $\Pi = \gamma(\gamma I + \sum xx^\top)^{-1}$ with $\gamma = 0.1$, so its protection is graded, and we count a direction as protected when its eigenvalue of $I - \Pi$ exceeds 0.5, that is when the corresponding eigenvalue of $\sum xx^\top$ exceeds γ (Figure 6). Its protected directions are therefore attenuated rather than annihilated, which is why its decline is gradual where the collapse of GPM is abrupt (Figure 3a). On the stream of 1200 segments the protected dimension of OWM grows further from 475.3 to 492.0 of 512 in fc1 and from 172.0 to 205.7 of 576 in conv3 between segments 600 and 1200, and its P keeps falling. Over windows of 150 segments of that stream, with three schedules and one initialization each, P of OWM falls from 1.365 to 0.661 and that of GPM from 0.769 to 0.108, whereas that of FRSP+ReDo stays between 1.39 and 1.45.

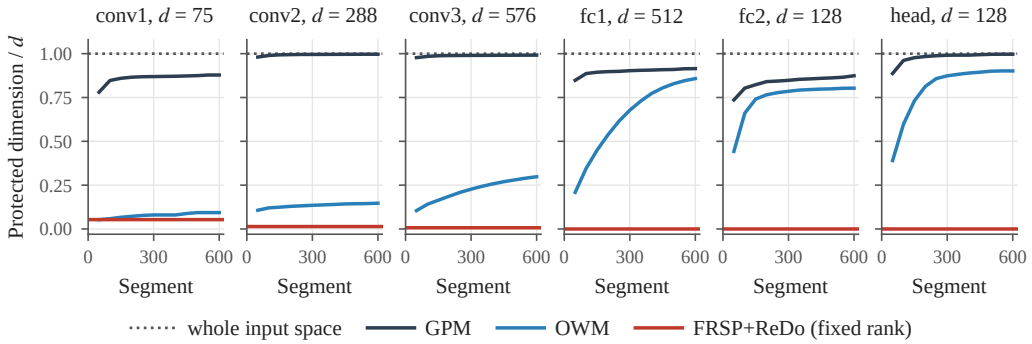


Figure 6: Share of the input dimension of each layer protected by GPM (columns of its basis) and OWM on CN-ImageNet, against the fixed rank of FRSP+ReDo and the whole input space.

Bounding the dimension is what matters. Accumulating protection without a bound is far below FRSP+ReDo on 15 of 15 pairs, with or without the true task boundaries (Table 5). Accumulating the tracked basis of FRSP+ReDo instead, up to a small cap per convolutional layer, stays close to the moving basis, although below it on every pair, and only a cap of 32 on all layers falls clearly below it. What saturates is thus an unbounded protected dimension rather than accumulation itself.

Table 5: Bounded and unbounded protection on CN-ImageNet, as the paired difference in P against FRSP+ReDo (first row) with the number of positive pairs. Capped accumulation adds the tracked rank-4 directions outside the accumulated basis every 1,000 updates until the cap is reached, so cap 4 freezes the first basis.

Setting	Difference in P vs FRSP+ReDo
FRSP+ReDo (tracked rank 4, conv1–conv3)	–
Accumulating protection without a bound	
OWM + ReDo	−0.302 (0/15)
OWM	−0.397 (0/15)
OWM with task boundaries (priv)	−0.335 (0/15)
GPM	−1.099 (0/15)
GPM with task boundaries (priv)	−1.242 (0/15)
Capped accumulation of the tracked rank-4 basis	
cap 4, conv1–conv3 (frozen basis)	−0.008 (0/15)
cap 8, conv1–conv3	−0.009 (0/15)
cap 16, conv1–conv3	−0.010 (0/15)
cap 32, all layers	−0.087 (0/15)

D.2 Persistent dormancy

The learners of this subsection are trained separately, record their activations at every segment and track their subspace by the streaming power iteration (Appendix C). Their values can differ slightly from those of Table 1, for FRSP only 1.008 against 0.994 in P.

Recycling acts in the unprotected layers. With the threshold of the dormant periods (Appendix B) recorded at every segment, and every count averaged over the trained neural networks, the protected convolutional layers under FRSP+ReDo have a dormant fraction per segment of 0.0% in all three layers, against 2.6% and 18.6% for ReDo in conv2 and conv3, and FRSP+ReDo reinitializes 42k times against 124k for ReDo. The remaining dormancy lies in the unprotected fc1 (5.1% per segment), where recycling reinitializes the dormant neurons (Figure 5d). With only the two unprotected linear layers recycled, P is 1.408, within 0.006 of FRSP+ReDo on every pair, with 34k reinitializations, all in those layers. With only the protected convolutional layers recycled, P is 1.074 with 374 reinitializations and fc1 is 86% dormant. Without recycling, protection makes the dormant periods of conv2 and conv3 longer than under AdamW without weight decay, the setting of Figure 4, with a reactivation rate in conv3 of 14.7% against 19.4% per segment and periods of 7.1 against 5.3 segments on average, lower and longer on 13 of 15 pairs each. Removing the mean direction from

the protected subspace restores the reactivation, with a rate in conv3 of 21.4% per segment and periods of 4.7 segments, at the level of AdamW and higher than under the mean-aligned protection on 14 of 15 pairs. Counted over all hidden layers, the periods are rarer under protection (9.4k against 12.3k) but not longer (11.9 against 12.0 segments), and the share of dormant unit-segments is 0.39 against 0.51.

The protected subspace contains the mean direction. On the weights learned by protection of conv1 to conv3 at $k = 4$ without recycling, on schedules 0 to 4 with one initialization each, the alignment $\|U^\top \mu\|^2/\|\mu\|^2$ of the protected subspace with the mean input direction is 0.93, 1.00 and 1.00 in the three convolutional layers. With the input second moment estimated on 256 images of each of 20 tasks, the mean-aligned basis holds 0.71 and 0.89 of the input energy in conv2 and conv3, of which the mean direction alone holds 0.54 and 0.87, whereas a basis of the same rank orthogonal to the mean direction holds 0.11 and 0.09. Table 6 gives the gradient energy that each subspace removes and P of these learners. Raising the rank of the mean-excluded subspace to 32 does not restore the gradient energy that the mean-aligned one removes, so the two controls cannot be matched in strength within this architecture.

Persistent dormancy requires the full projection. Halving the projection along the same directions ($\alpha = 0.5$) without recycling brings the dormancy of conv3 close to that of AdamW (Figure 4), with $P = 1.077$, $+0.021$ to $+0.127$ above the full projection on 15 of 15 pairs. With recycling it gives $P = 1.402$ against 1.410 (lower on 15 of 15 pairs).

The tracking half-life matters little. Table 7 gives the paired controls of Section 6.2. Since the top-k directions are dominated by the mean activation, the half-life of the tracking should matter little. It is an implementation default rather than a selected constant, and varying it from 8k to 512k updates on FRSP only moves P by at most 0.018, and from 32k to 512k on the recycling learners by at most 0.007 (Table 10), consistent with persistent rather than recency-weighted directions.

Table 6: Mean-aligned and mean-excluded protection of conv1 to conv3. Gradient energy is the fraction of the gradient energy of each convolutional layer that the projection removes, measured without recycling on five trained neural networks, and the counts are the pairs on which a row is above the mean-aligned learner.

Protected subspace	gradient energy removed (conv1 / conv2 / conv3)	P without recycling	pairs	P with recycling	pairs
mean-aligned, rank 4	0.94 / 0.89 / 0.93	1.008	–	1.410	–
mean-excluded, rank 4	0.65 / 0.09 / 0.09	1.067	15/15	1.382	0/15
mean-excluded, rank 8	0.70 / 0.11 / 0.12	1.057	15/15	1.373	0/15
mean-excluded, rank 16	0.69 / 0.16 / 0.15	1.033	12/15	1.348	0/15
mean-excluded, rank 32	0.67 / 0.18 / 0.21	1.003	7/15	1.318	0/15

Table 7: Paired controls on CN-ImageNet, differences in P and first-batch recall with the number of positive pairs (definitions in Appendix C). The controls track their subspace by the streaming power iteration and FRSP+ReDo by the exact eigendecomposition, so the half-life row supports only a change below 0.01.

Contrast	Difference in P	Difference in R_0
Protection − isotropic shrinkage, no recycling, all layers, k=8	+0.045 (15/15)	+0.013 (15/15)
Isotropic shrinkage with ReDo − ReDo, conv1–conv3, k=4	+0.005 (15/15)	+0.002 (10/15)
Tracked − fixed random basis, with ReDo, conv1–conv3, k=4	+0.030 (15/15)	+0.010 (14/15)
Half-life 512k − 32k updates, with ReDo, conv1–conv3, k=4	−0.004 (0/15)	−0.005 (1/15)

D.3 Protected layers and tracked rank

Figure 7 gives the protected layers and the tracked rank of FRSP+ReDo. The overlap between the top-four input subspaces of two tasks, over 190 task pairs with 256 images per task on the weights learned on schedules 0 to 4 with three initializations each, is 0.98, 0.81, 0.72, 0.35 and 0.70 for conv1, conv2, conv3, fc1 and fc2 on the weights learned by FRSP+ReDo, and 0.98, 0.74, 0.55, 0.54

and 0.73 on the weights learned by ReDo (Figure 5a), whereas on the untrained neural network of each of schedules 0 to 4 every layer after the first lies between 0.80 and 0.86.

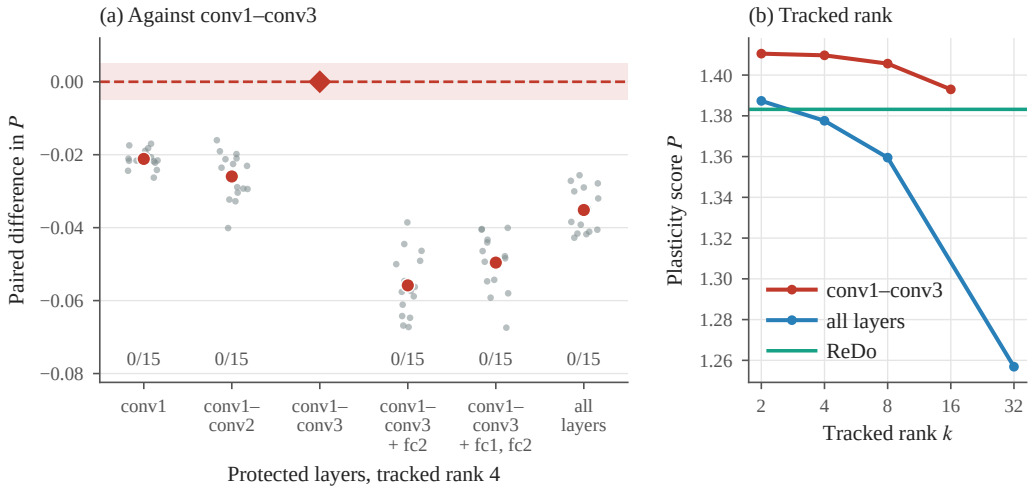


Figure 7: Protected layers and tracked rank of FRSP+ReDo on CN-ImageNet. (a) Paired difference in P against conv1 to conv3 at rank four, with pairs as dots, their mean as a large dot, the number of positive pairs below and the tie margin shaded. (b) P against the tracked rank on conv1 to conv3 and on all layers, against ReDo.

D.4 Other architectures and a stationary stream

The MLP needs no recycling. On the MLP (Table 8), FRSP+ReDo and protection alone are level in P. Under protection almost no neuron of the MLP becomes dormant (dormant fraction at most 0.002), so recycling has no effect on top of it, whereas without protection AdamW leaves over a quarter of its units dormant.

Table 8: All metrics on CN-ImageNet with the MLP.

Method	P	P_{novel}	P_{revisit}	ΔP	FT	R_0	ACC	BWT	$F\downarrow$	rank	dorm.
FRSP+ReDo all layers $k{=}8$	1.151	1.099	1.166	−0.021	0.062	0.162	0.224	−0.032	0.098	113.1	0.000
FRSP+ReDo all layers $k{=}4$	1.150	1.112	1.162	−0.029	0.078	0.129	0.221	−0.028	0.109	111.3	0.000
FRSP+ReDo all layers $k{=}2$	1.141	1.106	1.152	−0.037	0.096	0.112	0.219	−0.028	0.115	110.6	0.000
FRSP+ReDo all layers $k{=}16$	1.120	1.042	1.144	−0.048	0.053	0.205	0.230	−0.031	0.081	114.9	0.000
FRSP only, all layers $k{=}8$	1.152	1.101	1.168	−0.019	0.063	0.168	0.226	−0.031	0.095	108.7	0.002
Parseval	1.118	1.101	1.123	+0.004	0.154	0.056	0.217	−0.023	0.120	103.8	0.004
Spectral regularization	1.111	1.088	1.119	+0.029	0.146	0.063	0.217	−0.025	0.120	108.1	0.001
L2-Init	1.059	1.038	1.066	+0.013	0.143	0.050	0.217	−0.024	0.119	86.7	0.032
ReDo	0.937	0.895	0.951	−0.095	0.110	0.071	0.220	−0.023	0.125	95.1	0.002
CBP	0.930	0.884	0.945	−0.086	0.110	0.072	0.220	−0.024	0.126	93.4	0.002
S-EWC	1.048	0.975	1.072	−0.115	0.177	0.077	0.226	−0.020	0.113	55.4	0.130
AdamW	0.726	0.657	0.749	−0.505	0.080	0.101	0.224	−0.020	0.103	10.0	0.279
ER-1k (priv)	0.883	0.798	0.911	−0.441	0.106	0.140	0.224	−0.025	0.101	33.0	0.167
OWM	0.807	0.549	0.893	−0.554	0.047	0.463	0.244	−0.016	0.040	105.9	0.000
GPM	0.182	0.144	0.193	−0.454	0.017	0.413	0.203	−0.005	0.036	10.2	0.191

Protection does not impair stationary learning. In a stationary control that repeats one task for 600 segments with the segment lengths of CN-ImageNet, over 15 pairs, every continual learner reaches a raw end-of-segment online accuracy of 1.000 over the last 200 segments. FRSP+ReDo does not impair learning on this stream, with 0.990 over the first 50 segments against 0.986 to 0.987 for ReDo, AdamW and CBP, and a final held-out accuracy of 0.695 against 0.670 to 0.672.

Table 9: CN-ImageNet with a normalization before every activation of the ConvNet. The last column is the paired margin in P of FRSP+ReDo over the row within the block, and the second block also resets the normalization parameters of a recycled unit.

Method	P	rank / dorm.	FRSP+ReDo − row
Recycling that resets the weights of a recycled unit			
FRSP+ReDo	1.259	10.8 / 0.370	
ReDo	1.142	14.3 / 0.485	+0.117 (15 of 15)
AdamW	1.200	**30.3 / 0.292**	+0.063 (15 of 15)
Recycling that also resets the normalization parameters of a recycled unit			
FRSP+ReDo	**1.264**	8.4 / 0.325	
ReDo	1.195	11.4 / 0.350	+0.073 (15 of 15)

Under normalization, recycling must reset the normalization. The ConvNet of the main text has no normalization. A normalized variant places a GroupNorm with one group after each convolution and a LayerNorm after each linear layer, which keep no running statistics that could carry task statistics across boundaries, and FRSP+ReDo, ReDo and AdamW are trained on it with the hyperparameters of the main protocol, normalized by a restart oracle of the same architecture (Table 9). FRSP+ReDo remains above ReDo and AdamW in P, on 15 of 15 pairs each. Under normalization, however, FRSP+ReDo and ReDo end with far more dormant units than without it, most of them in the linear layers, and recycling no longer removes them, since the reset rule reinitializes the weights of a dormant unit but leaves the gain and bias of its normalization, with which it can become dormant again. Resetting these parameters together with the weights raises P of ReDo on 15 of 15 pairs and leaves that of FRSP+ReDo level (higher on 10 of 15 pairs), lowers the dormant fraction of both, and leaves FRSP+ReDo above ReDo on 15 of 15 pairs (second block of Table 9). Under normalization, recycling should therefore reset the normalization of a unit along with its weights.

D.5 All metrics and schedule-clustered statistics

Table 10 gives the metrics of Appendix B other than A_{1024} and A_{10000} for the ConvNet settings of the main text and their ablations, and Table 11 the paired margins and factorial effects over ten schedules, including A_{1024} and A_{10000}. In P, FRSP+ReDo is above ReDo, CBP, L2-Init, Parseval, spectral regularization, weight clipping, shrink-and-perturb, UPGD and S-EWC on 15 of 15 pairs.

Table 10: All metrics on CN-ImageNet for the ConvNet settings of the main text and their ablations.

Method	P	P_{novel}	P_{revisit}	ΔP	FT	R_0	A_{320}	A_{3200}	ACC	BWT	$F\downarrow$	rank	dorm.
Fixed-rank protection with recycling													
FRSP+ReDo (conv1–conv3, $k{=}4$)	1.410	1.406	1.414	+0.130	0.271	0.053	0.313	0.628	0.239	−0.019	0.154	21.7	0.014
FRSP with self-normalized resets	1.414	1.412	1.418	+0.121	0.286	0.050	0.316	0.637	0.239	−0.018	0.155	21.9	0.000
Protected layers and rank, with ReDo (Figure 7)													
FRSP+ReDo, conv1 $k{=}4$	1.390	1.387	1.394	+0.140	0.318	0.044	0.334	0.620	0.236	−0.019	0.157	16.6	0.078
FRSP+ReDo, conv1–conv2 $k{=}4$	1.387	1.378	1.391	+0.191	0.293	0.044	0.320	0.618	0.236	−0.021	0.157	14.1	0.074
FRSP+ReDo, conv1–conv3 and fc2 $k{=}4$	1.359	1.334	1.367	+0.134	0.221	0.060	0.298	0.598	0.241	−0.022	0.155	47.8	0.011
FRSP+ReDo, conv1–conv3, fc1 and fc2 $k{=}4$	1.364	1.343	1.371	+0.116	0.205	0.066	0.295	0.591	0.239	−0.027	0.154	79.0	0.001
FRSP+ReDo, conv1–conv3 $k{=}2$	1.411	1.410	1.414	+0.108	0.299	0.050	0.322	0.631	0.239	−0.020	0.155	24.4	0.014
FRSP+ReDo, conv1–conv3 $k{=}8$	1.406	1.396	1.412	+0.168	0.250	0.053	0.306	0.627	0.240	−0.019	0.153	18.7	0.015
FRSP+ReDo, conv1–conv3 $k{=}16$	1.393	1.372	1.402	+0.231	0.229	0.055	0.299	0.622	0.237	−0.019	0.152	16.6	0.019
FRSP+ReDo, all layers $k{=}2$	1.387	1.380	1.393	+0.103	0.238	0.062	0.303	0.601	0.238	−0.026	0.152	70.6	0.000
FRSP+ReDo, all layers $k{=}4$	1.378	1.362	1.384	+0.116	0.201	0.067	0.293	0.593	0.239	−0.025	0.152	31.4	0.002
FRSP+ReDo, all layers $k{=}8$	1.359	1.331	1.369	+0.159	0.159	0.068	0.283	0.568	0.236	−0.025	0.147	7.1	0.007
FRSP+ReDo, all layers $k{=}32$	1.257	1.199	1.277	+0.166	0.120	0.083	0.276	0.531	0.234	−0.027	0.150	17.3	0.005
Capped accumulation of the tracked rank-4 basis, with ReDo (Table 5)													
Capped accumulation, cap 4, conv1–conv3	1.402	1.399	1.406	+0.133	0.308	0.047	0.329	0.623	0.236	−0.021	0.156	15.5	0.041
Capped accumulation, cap 8, conv1–conv3	1.401	1.396	1.405	+0.141	0.294	0.049	0.322	0.622	0.237	−0.020	0.154	16.5	0.040
Capped accumulation, cap 16, conv1–conv3	1.402	1.393	1.407	+0.145	0.269	0.051	0.312	0.621	0.235	−0.022	0.156	18.4	0.028
Capped accumulation, cap 32, all layers	1.327	1.294	1.339	+0.124	0.163	0.069	0.282	0.561	0.239	−0.023	0.151	60.6	0.003
Protection without recycling													
FRSP only, conv1–conv3 $k{=}4$	0.994	0.959	1.006	−0.306	0.105	0.082	0.262	0.407	0.240	−0.018	0.132	7.2	0.520
FRSP only, all layers $k{=}8$	1.126	1.082	1.140	−0.105	0.097	0.089	0.268	0.450	0.239	−0.031	0.148	5.0	0.515
Controls and the half-life of the second-moment EMA (Table 7)													
Fixed random basis, conv1–conv3 $k{=}4$, + ReDo	1.381	1.378	1.384	+0.141	0.315	0.043	0.334	0.610	0.235	−0.020	0.157	17.7	0.081
Isotropic shrinkage, conv1–conv3 $k{=}4$, + ReDo	1.388	1.385	1.392	+0.147	0.312	0.044	0.331	0.617	0.236	−0.020	0.157	14.9	0.059
Isotropic shrinkage, all layers $k{=}8$	1.079	1.043	1.091	−0.102	0.139	0.076	0.272	0.498	0.248	−0.023	0.153	5.1	0.592
FRSP+ReDo, half-life 128k	1.404	1.398	1.408	+0.113	0.297	0.050	0.322	0.632	0.240	−0.018	0.155	26.4	0.022
FRSP+ReDo, half-life 512k	1.407	1.406	1.410	+0.117	0.311	0.047	0.329	0.626	0.238	−0.019	0.155	17.9	0.026
FRSP+ReDo, all layers $k{=}4$, half-life 128k	1.371	1.353	1.378	+0.113	0.226	0.063	0.301	0.597	0.240	−0.026	0.155	70.5	0.002
FRSP only, all layers $k{=}8$, half-life 8k	1.144	1.098	1.159	−0.087	0.098	0.093	0.271	0.453	0.241	−0.029	0.146	3.4	0.519
FRSP only, all layers $k{=}8$, half-life 128k	1.141	1.100	1.155	−0.114	0.115	0.086	0.272	0.464	0.243	−0.026	0.148	6.5	0.583
FRSP only, all layers $k{=}8$, half-life 512k	1.113	1.070	1.127	−0.144	0.124	0.084	0.271	0.473	0.243	−0.024	0.151	4.8	0.572
Accumulating protection													
OWM + ReDo	1.112	1.057	1.130	−0.340	0.077	0.134	0.291	0.362	0.250	−0.039	0.096	36.2	0.000
OWM	1.019	0.949	1.042	−0.389	0.076	0.145	0.292	0.356	0.249	−0.041	0.099	20.6	0.351
OWM with task boundaries (priv)	1.082	1.036	1.098	−0.360	0.087	0.117	0.279	0.374	0.248	−0.043	0.121	18.1	0.471
GPM	0.301	0.306	0.299	−0.602	0.032	0.106	0.230	0.277	0.205	−0.006	0.080	2.1	0.722
GPM with task boundaries (priv)	0.169	0.188	0.163	−0.537	0.015	0.109	0.225	0.252	0.201	−0.002	0.048	1.4	0.755
Recycling and regularization													
ReDo	1.383	1.381	1.387	+0.130	0.318	0.043	0.335	0.610	0.235	−0.019	0.157	17.9	0.084
Self-normalized resets	1.406	1.411	1.408	+0.102	0.327	0.041	0.340	0.620	0.234	−0.019	0.158	16.1	0.002
CBP	1.374	1.367	1.379	+0.153	0.324	0.044	0.340	0.620	0.238	−0.019	0.157	22.6	0.088
L2-Init	1.317	1.303	1.324	+0.069	0.267	0.048	0.315	0.587	0.237	−0.023	0.157	29.6	0.133
Parseval	1.335	1.321	1.341	+0.075	0.300	0.049	0.328	0.608	0.238	−0.023	0.159	51.0	0.088
Spectral regularization	1.333	1.316	1.340	+0.092	0.306	0.051	0.330	0.613	0.238	−0.021	0.159	46.5	0.065
UPGD	1.306	1.278	1.315	+0.084	0.317	0.051	0.357	0.592	0.238	−0.022	0.153	12.9	0.107
S-EWC	1.265	1.229	1.278	+0.088	0.285	0.064	0.344	0.599	0.242	−0.016	0.129	31.2	0.187
Shrink & Perturb	1.242	1.224	1.248	+0.039	0.255	0.046	0.308	0.564	0.232	−0.021	0.155	9.2	0.143
Weight clipping	1.277	1.259	1.284	+0.030	0.252	0.052	0.309	0.577	0.237	−0.020	0.153	8.1	0.189
Optimizer only and replay													
AdamW (lr 3×10^{-4}, wd 10^{-2})	1.114	1.085	1.123	−0.065	0.169	0.066	0.277	0.508	0.245	−0.019	0.151	8.9	0.584
AdamW (lr 10^{-4}, wd 10^{-2})	1.080	1.033	1.096	+0.016	0.150	0.082	0.278	0.513	0.252	−0.024	0.155	10.1	0.455
AdamW (lr 3×10^{-4}, wd 0)	1.050	1.027	1.058	−0.134	0.152	0.069	0.272	0.484	0.245	−0.019	0.151	6.1	0.604
ER-1k (priv)	1.110	1.042	1.132	−0.244	0.125	0.129	0.305	0.453	0.247	−0.025	0.129	24.3	0.437

Table 11: Schedule-clustered statistics on CN-ImageNet over ten schedules. Each row is a paired margin or a factorial effect over the common pairs, with the number of positive pairs, the number of schedules with a positive mean and a 95% interval. The factorial block is that of Table 2, where a main effect is the mean of the two simple effects and the interaction is the effect of recycling with protection minus that without.

Metric	Margin	Pairs	Schedules	Interval
FRSP+ReDo against ReDo, 30 pairs on 10 schedules				
P	+0.027	30 of 30	10 of 10	+0.025 to +0.029
P_{novel}	+0.024	30 of 30	10 of 10	+0.022 to +0.026
P_{revisit}	+0.028	30 of 30	10 of 10	+0.026 to +0.030
R_0	+0.009	29 of 30	10 of 10	+0.008 to +0.010
A_{320}	−0.020	0 of 30	0 of 10	−0.022 to −0.019
A_{1024}	+0.005	28 of 30	9 of 10	+0.003 to +0.007
A_{3200}	+0.018	30 of 30	10 of 10	+0.017 to +0.019
A_{10000}	+0.042	30 of 30	10 of 10	+0.040 to +0.043
Two-by-two factorial comparison of Table 2, 30 pairs on 10 schedules				
Recycling, P	+0.371	30 of 30	10 of 10	+0.362 to +0.379
Recycling, R_0	−0.028	0 of 30	0 of 10	−0.029 to −0.026
Recycling, dorm.	−0.523	0 of 30	0 of 10	−0.537 to −0.509
Protection, P	−0.012	5 of 30	0 of 10	−0.017 to −0.008
Protection, R_0	+0.011	30 of 30	10 of 10	+0.010 to +0.013
Protection, dorm.	−0.069	1 of 30	0 of 10	−0.083 to −0.055
Interaction, P	+0.079	30 of 30	10 of 10	+0.071 to +0.088
Interaction, R_0	−0.005	8 of 30	2 of 10	−0.008 to −0.002
Interaction, dorm.	0.000	16 of 30	4 of 10	−0.028 to +0.029
FRSP+ReDo against self-normalized resets, 30 pairs on 10 schedules				
P	+0.003	28 of 30	9 of 10	+0.002 to +0.003
R_0	+0.010	30 of 30	10 of 10	+0.009 to +0.012
FRSP with self-normalized resets against self-normalized resets, 30 pairs on 10 schedules				
P	+0.007	29 of 30	10 of 10	+0.006 to +0.008
R_0	+0.009	29 of 30	10 of 10	+0.007 to +0.010

E ADDITIONAL RESULTS ON CN-PROCGEN

Evaluation and normalization. Every method is evaluated on the stream of Appendix B with schedules 0 to 4, the four methods of Table 3 also on the held-out schedules 5 to 9, and each uses the hyperparameters of Appendix C, which are development selections except for the first-block variant, the recycling constants of ReDo and PPO at the matched learning rate. Each learner is trained once per schedule with Adam rather than AdamW, so Table 12 aggregates five runs per method and Table 3 ten. FRSP+ReDo protects the fifteen convolutional layers of the trunk at $k = 4$, leaves the policy and value heads unprotected and recycles with the period and threshold used by ReDo. Segments are normalized by a restart oracle as on CN-ImageNet, which reinitializes the neural network at every segment, so 1.0 is the oracle and 0.0 a random policy, P is a normalized return, and a segment is excluded when its normalized score is undefined, as in Appendix B. In Table 12, P pools all segments over five schedules and the next three columns split it by what a segment presents, the first appearance of a game, a level bucket of a game already seen, or a task seen before. On this stream R_0 uses episodes that start in the first 3% of a revisit and so includes some learning within it, which is why the restart oracle, which retains nothing, reaches 0.067.

Table 12: All methods on CN-Procgen over schedules 0 to 4, one initialization each. The restart oracle is not ranked.

Method	P	New game	New levels	Revisit	ΔP	R_0	rank / dorm.
Restart oracle	1.000	1.000	1.000	1.000	0.000	0.067	56.8 / 0.278
PPO	0.682	0.622	0.709	0.693	−0.881	−0.008	26.3 / 0.404
PPO, lr 2.5×10^{-4} (matched)	0.814	0.676	0.812	0.859	−0.466	**0.051**	64.5 / 0.236
ReDo	0.949	0.762	1.041	0.979	−0.423	−0.036	99.3 / 0.106
CBP	0.920	0.657	1.134	0.934	−0.382	−0.025	75.3 / 0.185
L2-Init	**0.958**	0.668	1.015	**1.034**	−0.233	0.018	96.0 / 0.118
Shrink & Perturb	0.924	0.739	0.973	0.969	−0.480	−0.042	68.0 / 0.211
S-EWC	0.644	0.587	0.625	0.668	−0.775	−0.039	20.4 / 0.414
UPGD	0.831	0.580	0.955	0.871	−0.530	−0.011	57.2 / 0.249
Periodic reset	0.712	**0.866**	0.650	0.683	**0.141**	−0.026	34.0 / 0.267
FRSP+ReDo (whole trunk)	0.891	0.592	0.997	0.953	−0.487	**0.055**	100.7 / **0.073**
FRSP+ReDo (first block only)	0.950	0.766	**1.141**	0.948	−0.384	0.015	**101.9** / 0.094
FRSP only	0.879	0.619	0.958	0.938	−0.516	**0.053**	68.1 / 0.194

PPO is compared at the matched learning rate. The development stream selected 5×10^{-4} for PPO, whereas FRSP+ReDo and ReDo selected 2.5×10^{-4}, and on the main stream PPO at 2.5×10^{-4} is higher in P on 5 of 5 schedules, with a lower dormant fraction. The margins of every method over PPO at its development learning rate therefore overstate their effect, and Table 3 reports PPO at the matched learning rate.

The deficit of the whole trunk comes from the projection. FRSP+ReDo protecting the whole trunk is below ReDo, CBP, L2-Init and shrink-and-perturb in P (Table 12), with the deficit on the first appearances of games and of level buckets. Removing recycling from FRSP+ReDo leaves P nearly unchanged (paired -0.010, positive on 3 of 5 schedules) and does not lower the new-game score, while the dormant fraction rises and the effective rank falls (Table 12). On this stream recycling lowers dormancy and raises the effective rank without raising P, and since protection alone is also far below ReDo on new games (0.619 against 0.762), the new-game deficit against ReDo comes from the projection itself.

Overlap is highest in the first block. Passing 512 observations of each game through the weights learned by FRSP+ReDo protecting the whole trunk on each of ten schedules, with the observations of each game drawn with a fixed seed, we took the top-four subspace of the input second moment of each layer per game and computed the overlap over all 66 game pairs (Figure 5b). The mean overlap over pairs is 0.48 to 0.57 in the three measured layers of the first block, 0.40 to 0.49 in the second and 0.24 to 0.44 in the third, with pairs as low as 0.03.

Overlap does not predict single transitions. For every first appearance of a game after the first segment (103 over ten schedules after excluding segments with an undefined normalized score), the Spearman correlation between the overlap of the preceding and the new game, measured on the weights learned by FRSP+ReDo on schedule 0, and the paired difference of the normalized end-of-segment return between FRSP+ReDo and ReDo is at most 0.20 in any block and 0.17 over the nine measured layers ($p \geq 0.04$, pooling the transitions without clustering by schedule). On CN-ImageNet, over the 540 segments of 30 schedule-initialization pairs whose task and preceding task both lie among 20 tasks, the correlation of the overlap with the paired gain of FRSP+ReDo over ReDo is 0.07 for the mean over the convolutional layers and 0.11 over all layers, and a bootstrap that resamples schedules gives -0.101 to 0.201 and -0.032 to 0.174. The overlap therefore separates the streams and the layers in which protection is effective but does not predict single transitions.

The first-block variant is not separated from ReDo. Table 3 reports ten schedules, schedules 0 to 4, on which the first-block setting was identified, together with the held-out schedules 5 to 9. The development stream does not support this setting, since with the other constants of FRSP+ReDo fixed it ranks the whole trunk above the first two blocks and the first block on both development schedules, and there the whole trunk is also above ReDo, so its twelve segments do not show the deficit of the whole trunk (Appendix C). The held-out schedules are harder for every learner. The whole trunk and PPO stay below ReDo and the first block, but the first block, level with ReDo on schedules 0 to 4, is below it on schedules 5 to 9. The normalized return over schedules 0 to 4 and over schedules 5 to 9 is 0.949 and 0.874 for ReDo, 0.950 and 0.859 for FRSP+ReDo on the first block, 0.891 and 0.817 for FRSP+ReDo on the whole trunk, and 0.814 and 0.775 for PPO at the matched learning rate. On new games the drop is largest for ReDo, from 0.762 to 0.486, whereas FRSP+ReDo on the first block falls from 0.766 to 0.612, so over ten schedules its mean margin over ReDo on new games is positive but not separated (higher on 6 of 10 schedules), whereas the two are level over schedules 0 to 4. The deficit of the whole trunk against ReDo is present on both halves. The other rows of Table 12 use schedules 0 to 4 only.