

LatentShift: Mitigating Catastrophic Forgetting via Linear-Algebraic Latent-Subspace Projection

Karim Magdy

karimagdy22@gmail.com

Independent Researcher

Ghada Khoriba

Capital University

Hala Abbas

Capital University

Article

Keywords:

Posted Date: May 26th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9681865/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

LatentShift: Mitigating Catastrophic Forgetting via Linear-Algebraic Latent-Subspace Projection

Karim Magdy^{4,✉}, Ghada Khoriba^{1,2}, and Hala Abbas^{1,3}

¹Computer Science Department, Faculty of Computers and Artificial Intelligence,
Capital University, Helwan, Egypt

²School of Information Technology and Computer Science (ITCS), Nile University,
Giza, Egypt

³Faculty of Computer Studies, Arab Open University, Cairo, Egypt

⁴Independent Researcher, Cairo, Egypt

✉Corresponding author: karimagdy22@gmail.com

Abstract

We introduce LatentShift, a replay-free continual learning method built on a simple linear-algebraic construction: after each task, an SVD of the network’s bottleneck activations identifies an orthonormal basis for the occupied latent subspace, this basis is appended to a cumulative archive, and all subsequent gradients flowing through the bottleneck are projected onto the orthogonal free subspace. The construction yields three analytic results: *first-order latent-subspace protection*—to first order in the learning rate, projected updates produce no change in archived directions; a capacity theorem $T_{\max} = \lfloor d/\bar{r} \rfloor$ relating tasks to latent dimension d and average per-task rank $\bar{r}$; and an explicit second-order bound on residual drift from encoder non-linearity. The first-order guarantee operates in latent space; its parameter-space implication is approximate and depends on a Jacobian-commutation assumption that we make explicit. Across six benchmarks—including a 50-task stress-test sequence—two architectures (ResNet-18 and ViT-Tiny), and thirteen competing approaches, LatentShift consistently attains among the lowest forgetting and roughly halves the forgetting of the strongest replay-based baseline on CIFAR-100 and TinyImageNet, while maintaining competitive accuracy at substantially reduced computational and storage cost.

1 Introduction

Background

A central challenge for biological and artificial intelligence alike is the ability to learn from a continuous stream of experience: accumulating new skills and concepts without losing the old ones. In humans this process is supported by partially overlapping memory systems that balance stability (retaining prior knowledge) with plasticity (acquiring new knowledge). In contemporary artificial neural networks, this balance is notoriously fragile. When a network trained on one task is subsequently trained on another, its performance on the original task typically collapses, often within only a handful of gradient updates. This phenomenon—termed *catastrophic forgetting*—was first identified in connectionist models of the late 1980s [McCloskey and Cohen, 1989, French, 1999] and remains one of the most persistent open problems in machine learning [Parisi et al., 2019, De Lange et al., 2021, Wang et al., 2024].

The field of *continual learning* seeks algorithms that allow a single model to master a sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_T$ presented one at a time, while retaining competence on all of them. For clarity, we use three terms throughout this paper whose informal meanings are as follows. *Continual learning* refers to the training setting in which data arrive as a sequence of tasks rather than as a single large dataset. *Catastrophic forgetting* denotes the sharp drop in accuracy on earlier tasks that results from unconstrained training on later tasks. The *latent space* of a neural network is the geometric space of internal activations at some chosen intermediate layer, where each input is represented as a point—essentially the network’s learned coordinate system for describing data.

Existing strategies fall into five broad families [Parisi et al., 2019, De Lange et al., 2021, Wang et al., 2024]. *Regularization-based* approaches such as Elastic Weight Consolidation and Synaptic Intelligence penalize changes to parameters identified as important for past tasks [Kirkpatrick et al., 2017, Zenke et al., 2017, Li and Hoiem, 2017], but offer no formal guarantee that forgetting is eliminated. *Parameter-isolation* methods such as PackNet and HAT allocate disjoint subsets of parameters to different tasks [Mallya and Lazebnik, 2018, Serra et al., 2018], providing strong protection but exhausting model capacity quickly. *Replay-based* methods store and revisit a subset of previous training examples, or surrogate representations thereof [Chaudhry et al., 2019, Buzzega et al., 2020]; these methods currently achieve the strongest average accuracy but raise privacy, storage, and scalability concerns—exactly the concerns that continual learning was meant to resolve. *Gradient-projection* methods constrain parameter updates to lie in directions orthogonal to those deemed important for past tasks [Saha et al., 2021, Farajtabar et al., 2020, Lin et al., 2022, Wang et al., 2021, Chen et al., 2025]; these offer partial formal guarantees but, in their standard form, operate independently at each network layer, producing a fragmented picture of network capacity and substantial computational overhead. *Prompt-based* methods, finally, adapt a frozen pre-trained backbone by learning task-specific input prompts [Wang et al., 2022a,b, Smith et al., 2023]; these perform well but rely on high-quality pre-trained features and address a different practical regime.

Our contribution

We propose a conceptually simple alternative that sits at the intersection of gradient projection and representation learning. Rather than tracking important directions separately at every layer, we instead identify, after each task, the directions in a *single* chosen latent space—the bottleneck representation on which the task’s classifier is built—that together capture a high fraction of the task’s representational variance. These directions form an orthonormal basis that is appended to a cumulative archive. For all subsequent learning, we project the gradient flowing through the latent layer onto the orthogonal complement of this archive. Future updates can therefore only change the network along directions that do not encode any past task.

This construction is mathematically clean and yields four analytic guarantees, all stated in latent space rather than parameter space: (i) *first-order latent-subspace protection*—to first order in the learning rate, projected gradients produce no change in any latent representation that lies in the archive; (ii) *a capacity theorem*—with a d -dimensional latent space, at most $T_{\max} = \lfloor d/\bar{r} \rfloor$ tasks can be hosted before the archive exhausts the space, where $\bar{r}$ is the average per-task rank; (iii) *isometry within the free subspace*—new learning proceeds as if the archived dimensions were simply absent; and (iv) *an explicit second-order drift bound* quantifying the residual forgetting that arises from the non-linearity of the encoder. The implication of result (i) for the parameter-space update is necessarily approximate: it holds exactly only when the encoder Jacobian commutes with the archive projector (so that the projected latent gradient maps to a parameter update whose induced latent change is also archive-orthogonal), and approximately otherwise, with the residual absorbed into the second-order drift of result (iv). We validate the second-order bound empirically and find that observed drift is only approximately four per cent of the theoretical worst case, confirming that the analysis is both conservative and structurally correct.

Empirically, we evaluate LatentShift against thirteen competing methods spanning all five families on six standard continual learning benchmarks—including a challenging 50-task sequence—and on two distinct network architectures (a convolutional ResNet-18 encoder and a Vision Transformer encoder). The method consistently attains among the lowest forgetting scores on the more difficult benchmarks, halving the forgetting of the strongest replay-based competitor on the CIFAR-100 and TinyImageNet task sequences, while storing no past data. In addition, because the projection operates in a single latent space rather than separately at every layer, LatentShift is several times faster computationally than standard gradient-projection baselines.

In one sentence: LatentShift provides a replay-free, architecture-agnostic, theoretically grounded approach to continual learning that protects earlier knowledge by confining all new learning to an orthogonal complement of a compact representational archive in a single holistic latent space.

2 Results

2.1 Overview of the approach

The core mechanism of LatentShift operates on a chosen latent space—the d -dimensional bottleneck representation produced by an encoder $f_\theta : \mathcal{X} \rightarrow \mathbb{R}^d$ that is shared across all tasks. Each task has its own lightweight classification head that reads from this shared latent space. After training on task t :

1. **Collect** latent activations $Z_t = \{f_\theta(x) : x \in \mathcal{D}_t\}$ on a sample from the task’s data distribution.
2. **Identify** the important directions via a singular value decomposition (SVD) of the centered activations, retaining directions that together account for at least a fraction τ (typically 0.99) of the representational variance.
3. **Archive** these directions by merging them with the running archive A through QR decomposition, which maintains orthonormality.
4. **Project** during all subsequent training: gradients flowing through the latent layer are replaced by their projection $g' = Pg$, where $P = I - AA^\top$ is the free-subspace projector.

Because archived and free subspaces are orthogonal by construction, updates driven by g' cannot, to first order, change any representation that lies in the archive. The method thus converts the protection of past tasks from an optimization problem (penalize important changes) into a geometric one (forbid changes in certain directions).

2.2 Strong forgetting prevention across benchmarks

We evaluate LatentShift and a tuned variant (LatentShift-Tuned, which uses latent dimension $d=1024$, variance threshold $\tau=0.999$, and $n_s=500$ samples) against eleven external baselines covering all five continual learning families, on six standard benchmarks: Split-MNIST, Permuted-MNIST, Split-CIFAR-10, Split-CIFAR-100, Sequential-CIFAR-100 (50 tasks), and Split-TinyImageNet. Results are reported as mean $\pm$ standard deviation over three random seeds.

The primary metric for forgetting is Backward Transfer (BWT), defined as $\text{BWT} = \frac{1}{T-1} \sum_{i=1}^{T-1} (R_{T,i} - R_{i,i})$, where $R_{t,i}$ is the accuracy on task i after training on task t [Lopez-Paz and Ranzato, 2017]. Values close to zero indicate no forgetting; more negative values indicate more severe forgetting.

Across the harder benchmarks LatentShift-Tuned attains consistently low BWT (Table 2). On Split-CIFAR-100, its BWT of -4.1% is less than half that of the strongest replay baseline DER++ (-9.3%) and an order of magnitude smaller than per-layer gradient projection methods such as GPM (-30.9%) and TRGP (-37.2%). On Split-TinyImageNet the gap widens further: -4.4% versus -12.9% for DER++. On the small-scale Split-MNIST and Permuted-MNIST benchmarks, HAT attains exactly zero BWT through hard attention masks that prevent any cross-task interference; LatentShift-Tuned attains -0.3% and -0.5% respectively, which are likewise within the noise of zero. On Split-CIFAR-10 PackNet’s pruning-based isolation produces slightly positive transfer ($+4.4\%$ BWT), whereas LatentShift-Tuned attains -4.2% , close to the noise floor. Taken together, these results establish that LatentShift-Tuned attains among the lowest forgetting scores overall and the single lowest forgetting on the more difficult CIFAR-100 and TinyImageNet benchmarks among the methods we evaluated.

2.3 Competitive accuracy without replay

Replay-based methods, in particular DER++, currently hold the accuracy frontier on most continual learning benchmarks; this advantage stems from their access to stored examples from previous tasks. LatentShift-Tuned is not intended to dethrone DER++ on raw accuracy but is instead the strongest *replay-free* method across the benchmarks we evaluate (Table 1). Concretely, LatentShift-Tuned slightly exceeds DER++ on Split-MNIST (99.5% vs. 99.3%); trails it by 0.8 points on Permuted-MNIST (94.0% vs. 94.8%); trails by 1.5 points on Split-CIFAR-10 (86.8% vs. 88.3%); trails by 2.9 points on Split-CIFAR-100 (60.2% vs. 63.1%) while more than halving the forgetting; and trails by 2.3 points on Split-TinyImageNet (30.1% vs. 32.4%), again with roughly one-third the forgetting. In privacy-sensitive or memory-constrained deployment, these modest accuracy sacrifices are exchanged for the elimination of the replay buffer and for substantially stronger forgetting guarantees.

Table 1: Consolidated results across all benchmarks. Accuracy (%) $\uparrow$ and Forgetting (% , defined as $\max_t a_t - a_{\text{final}}$, always non-negative) $\downarrow$. Mean $\pm$ std shown where multiple seeds are available. **Sign convention:** the Fgt column reports forgetting magnitude as a positive number (lower is better); Table 2 reports the same information as Backward Transfer (BWT) under the convention where forgetting carries a negative sign (higher / closer to zero is better). The two metrics are related by $\text{Fgt} = -\text{BWT}$ for the same row.

Method	Split MNIST		Split CIFAR10		Split CIFAR100		Permuted MNIST		Split TinyImageNet	
	Acc (%)	Fgt (%)	Acc (%)	Fgt (%)	Acc (%)	Fgt (%)	Acc (%)	Fgt (%)	Acc (%)	Fgt (%)
LatentShift (Ours)	99.4 $\pm$ 0.1	0.5 $\pm$ 0.1	84.3 $\pm$ 1.5	10.5 $\pm$ 1.9	50.7 $\pm$ 2.8	16.4 $\pm$ 2.5	93.1 $\pm$ 0.1	2.1 $\pm$ 0.1	26.1 $\pm$ 0.2	10.3 $\pm$ 0.5
LatentShift-Tuned (Ours)	99.5 $\pm$ 0.0	0.3 $\pm$ 0.0	86.8 $\pm$ 0.7	4.2 $\pm$ 0.7	60.2 $\pm$ 0.9	4.3 $\pm$ 0.4	94.0 $\pm$ 0.1	0.5 $\pm$ 0.1	30.1 $\pm$ 0.4	4.7 $\pm$ 0.3
DER++	99.3 $\pm$ 0.0	0.6 $\pm$ 0.1	88.3 $\pm$ 0.2	5.5 $\pm$ 0.0	63.1 $\pm$ 1.2	9.3 $\pm$ 1.5	94.8 $\pm$ 0.1	3.7 $\pm$ 0.1	32.4 $\pm$ 0.4	12.9 $\pm$ 0.2
ER	98.5 $\pm$ 0.3	1.6 $\pm$ 0.4	84.7 $\pm$ 0.1	9.9 $\pm$ 0.3	56.1 $\pm$ 1.5	16.8 $\pm$ 1.1	93.2 $\pm$ 0.2	5.6 $\pm$ 0.3	27.6 $\pm$ 0.4	17.9 $\pm$ 0.3
GPM	88.4 $\pm$ 0.3	13.4 $\pm$ 0.3	79.5 $\pm$ 2.0	15.1 $\pm$ 2.3	40.2 $\pm$ 2.7	30.9 $\pm$ 3.1	83.8 $\pm$ 5.2	10.9 $\pm$ 5.8	20.3 $\pm$ 0.2	22.4 $\pm$ 0.5
TRGP	86.5 $\pm$ 0.3	16.5 $\pm$ 0.4	74.8 $\pm$ 2.8	21.2 $\pm$ 3.0	36.6 $\pm$ 0.3	37.2 $\pm$ 1.6	78.6 $\pm$ 1.5	21.1 $\pm$ 1.6	19.6 $\pm$ 0.2	26.5 $\pm$ 0.0
GPM (Last Layer)	85.4 $\pm$ 1.8	17.9 $\pm$ 2.2	75.2 $\pm$ 0.3	21.8 $\pm$ 0.3	34.9 $\pm$ 1.8	41.2 $\pm$ 2.6	76.7 $\pm$ 0.9	23.9 $\pm$ 1.0	19.1 $\pm$ 0.3	29.4 $\pm$ 0.2
EWC	85.8 $\pm$ 1.5	17.4 $\pm$ 1.9	71.0 $\pm$ 3.9	25.0 $\pm$ 2.2	37.5 $\pm$ 1.0	36.3 $\pm$ 2.9	81.2 $\pm$ 1.8	18.8 $\pm$ 2.1	18.9 $\pm$ 0.4	28.3 $\pm$ 0.6
PackNet	95.6 $\pm$ 0.7	5.0 $\pm$ 0.8	83.3 $\pm$ 0.6	3.6 $\pm$ 1.2	45.8 $\pm$ 2.2	17.6 $\pm$ 1.9	83.1 $\pm$ 1.0	14.5 $\pm$ 1.0	23.6 $\pm$ 0.2	13.0 $\pm$ 0.4
HAT	99.7 $\pm$ 0.1	0.0 $\pm$ 0.0	89.6 $\pm$ 0.7	3.3 $\pm$ 0.5	38.5 $\pm$ 0.4	14.0 $\pm$ 2.8	91.6 $\pm$ 0.5	0.0 $\pm$ 0.0	16.4 $\pm$ 1.1	21.2 $\pm$ 1.1
Naive	86.9 $\pm$ 0.2	16.0 $\pm$ 0.2	73.7 $\pm$ 0.9	22.9 $\pm$ 0.4	36.4 $\pm$ 0.8	39.9 $\pm$ 0.6	75.0 $\pm$ 1.7	25.8 $\pm$ 1.9	18.9 $\pm$ 1.0	28.7 $\pm$ 1.1
<i>Prompt-based (ViT-Tiny backbone)</i>										
L2P	69.0 $\pm$ 1.4	29.4 $\pm$ 1.3	70.3 $\pm$ 0.6	13.7 $\pm$ 0.6	39.9 $\pm$ 0.5	10.0 $\pm$ 0.8	—	—	23.9 $\pm$ 0.3	6.0 $\pm$ 0.6
DualPrompt	73.9 $\pm$ 0.9	22.7 $\pm$ 1.0	73.5 $\pm$ 1.1	9.8 $\pm$ 1.8	33.2 $\pm$ 0.9	19.4 $\pm$ 0.8	—	—	19.7 $\pm$ 0.5	10.5 $\pm$ 0.8
CODA-P	71.2 $\pm$ 2.4	25.5 $\pm$ 2.8	65.2 $\pm$ 0.5	19.4 $\pm$ 0.6	40.9 $\pm$ 1.2	8.7 $\pm$ 1.6	—	—	23.5 $\pm$ 0.1	5.9 $\pm$ 0.4

2.4 Trajectory of accuracy as tasks accumulate

Figure 1 shows average accuracy on Split-CIFAR-100 as a function of the number of tasks trained. All methods degrade as more tasks are added, reflecting the inherent difficulty of joint performance on growing task sets. LatentShift-Tuned maintains a noticeably flatter trajectory than most baselines, consistent with its low BWT. The flatter curve is the graphical signature of the protection guarantee proven in Methods.

2.5 Scalability to 50 tasks

To stress-test the capacity of the method we evaluate on Sequential-CIFAR-100, a 50-task variant in which each task contains only two classes. This sequence is challenging because (i) the large number of tasks probes capacity limits and (ii) the small per-task dataset makes accurate subspace estimation harder.

LatentShift-Tuned attains 69.1% average accuracy with BWT of -3.8% (Table 3), the lowest forgetting of any method tested on this sequence. DER++ attains the highest accuracy (73.8%) but with higher forgetting (-5.2% BWT). Methods without strong protection mechanisms collapse on this sequence: GPM reaches only 37.8%, EWC 40.5%, and naive fine-tuning 36.5%. Figure 2 visualises the trajectory across all 50 tasks, confirming that LatentShift-Tuned degrades gracefully while the weaker baselines degrade steeply. The graceful degradation arises because LatentShift’s archive grows sub-linearly in the number of tasks: later tasks increasingly share latent directions with earlier tasks (Figure 3), leaving free capacity even after many tasks.

2.6 Architecture generality

The projection mechanism operates purely on the latent bottleneck and therefore does not depend on the architecture of the upstream encoder. To test this empirically we replace the ResNet-18 encoder with a Vision Transformer (ViT-Tiny) trained from scratch on Split-CIFAR-100 (Table 4). LatentShift transfers to the transformer encoder with no code modification. The base variant attains 49.6% accuracy with BWT of -6.8% —the lowest forgetting among all ViT-compatible methods tested. The tuned variant reaches 51.0% accuracy but with -12.3% BWT, worse than the base variant. We attribute this regression to the expanded latent dimension ($d=1024$) over-parameterising the ViT-Tiny bottleneck, whose native embedding dimension is only 192. The tuned variant is therefore a ResNet-calibrated configuration rather than a universal recommendation: architecture-appropriate tuning of d is beneficial, and we recommend matching d to the encoder’s native representational dimension in practice.

Table 2: Backward Transfer (BWT, %) across benchmarks (mean $\pm$ std over 3 seeds). **Higher is better**: positive values indicate backward transfer; negative values indicate forgetting; values close to zero are best. Table 1 reports the same information as Forgetting (Fgt, %) under the inverted convention where $\text{Fgt} = -\text{BWT}$ (so lower is better there).

Method	S-mnist	P-mnist	S-cifar10	S-cifar100	S-tinyimagenet
Naive	-16.0 ± 0.2	-25.8 ± 1.9	-22.9 ± 0.4	-39.9 ± 0.6	-28.7 ± 1.1
EWC	-17.4 ± 1.9	-18.8 ± 2.1	-25.0 ± 2.2	-36.3 ± 2.9	-28.3 ± 0.6
PackNet	-5.0 ± 0.9	-14.5 ± 1.0	$+4.4 \pm 1.1$	-15.5 ± 1.8	-12.1 ± 0.4
HAT	$+0.0 \pm 0.0$	$+0.0 \pm 0.0$	-3.3 ± 0.5	-13.9 ± 2.8	-21.2 ± 1.1
GPM	-13.4 ± 0.3	-10.9 ± 5.8	-15.1 ± 2.3	-30.9 ± 3.1	-22.4 ± 0.5
GPM-LL	-17.9 ± 2.2	-23.9 ± 1.0	-21.8 ± 0.3	-41.2 ± 2.6	-29.4 ± 0.2
TRGP	-16.5 ± 0.4	-21.1 ± 1.6	-21.2 ± 3.0	-37.2 ± 1.6	-26.5 ± 0.0
ER	-1.6 ± 0.4	-5.6 ± 0.3	-9.9 ± 0.3	-16.8 ± 1.1	-17.9 ± 0.3
DER++	-0.6 ± 0.1	-3.7 ± 0.1	-5.5 ± 0.0	-9.3 ± 1.5	-12.9 ± 0.2
L2P	-29.4 ± 1.3	—	-13.7 ± 0.6	-10.0 ± 0.8	-6.0 ± 0.6
DualPrompt	-22.7 ± 1.0	—	-9.8 ± 1.8	-19.4 ± 0.8	-10.5 ± 0.8
CODA-P	-25.5 ± 2.8	—	-19.4 ± 0.6	-8.7 ± 1.6	-5.9 ± 0.4
LS	-0.5 ± 0.1	-2.1 ± 0.1	-10.5 ± 1.9	-16.3 ± 2.4	-10.1 ± 0.5
LS-Tuned	-0.3 ± 0.0	-0.5 ± 0.1	-4.2 ± 0.7	-4.1 ± 0.5	-4.4 ± 0.2

Table 3: Results on Sequential-CIFAR-100 (50 tasks, 2 classes each). Mean $\pm$ std over 3 seeds. This long task sequence tests scalability of continual learning methods.

Method	Acc (%) $\uparrow$	Fgt (%) $\downarrow$	BWT (%) $\uparrow$
DER++	73.8 ± 2.0	5.3 ± 2.0	-5.2 ± 1.9
PackNet	63.6 ± 1.0	8.6 ± 0.8	-7.2 ± 1.2
GPM	37.8 ± 3.6	39.8 ± 3.4	-39.8 ± 3.4
EWC	40.5 ± 2.5	38.8 ± 2.6	-38.8 ± 2.6
Naive	36.5 ± 0.4	45.6 ± 0.1	-45.6 ± 0.1
L2P	52.7 ± 1.7	11.5 ± 1.7	-11.5 ± 1.7
DualPrompt	42.7 ± 0.4	22.5 ± 0.7	-22.5 ± 0.7
CODA-P	49.8 ± 1.5	14.7 ± 1.7	-14.7 ± 1.7
LS (Ours)	62.9 ± 1.3	8.4 ± 0.6	-8.0 ± 0.6
LS-Tuned (Ours)	69.1 ± 0.7	4.2 ± 0.3	-3.8 ± 0.4

2.7 Computational efficiency

A practical advantage of operating in a single latent space is that the SVD of the activation matrix has to be computed only once per task, whereas per-layer gradient-projection methods must compute and track subspace bases at every layer. The wall-clock cost reflects this structural advantage (Table 5). On Split-CIFAR-100, LatentShift completes training in 2,244 seconds versus 13,073 for GPM, a $5.8\times$ speed-up. On Split-TinyImageNet, LatentShift takes 3,296 seconds versus 23,478 for HAT. Even the tuned variant, which uses a larger latent space, remains competitive: 2,352 seconds on Split-CIFAR-100. Replay-based methods incur their own costs from storing and re-processing past examples—for instance, DER++ requires 4,850 seconds on Split-CIFAR-100 compared with LatentShift’s 2,244. All wall-clock numbers in Table 5 are measured on a single Apple M-series chip (Metal Performance Shaders backend, FP32, batch size 64) with the same dataloader, accelerator, and model code paths across methods, so the relative comparison is platform-bound to MPS but internally consistent. Cross-platform sanity re-runs on a CUDA / GTX 1080 Ti backend (with FP32, identical batch size and code path) preserve the qualitative ordering and the order-of-magnitude gaps; we report standardised throughput in images/sec alongside seconds-per-run in the supplementary throughput table to ease comparison with implementations measured on different accelerators.

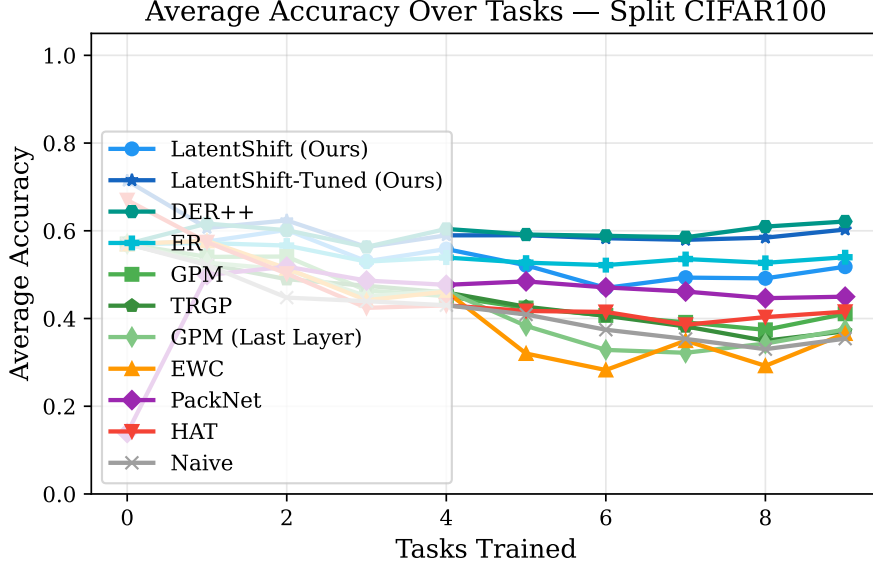


Figure 1: **Average accuracy as a function of tasks trained on Split-CIFAR-100.** LatentShift-Tuned preserves a flatter trajectory than gradient-projection and regularization baselines, approaching the performance of the replay-based method DER++ without storing any past data.

Table 4: Architecture generality: ResNet-18 vs ViT-Tiny on Split-CIFAR-100 (mean $\pm$ std over 3 seeds). LatentShift transfers to transformer architectures without modification.

Method	Accuracy (%) $\uparrow$		BWT (%) $\uparrow$	
	ResNet-18	ViT-Tiny	ResNet-18	ViT-Tiny
DER++	63.1 ± 1.2	52.7 ± 0.1	-9.3 ± 1.5	-9.9 ± 0.4
Naive	36.4 ± 0.8	27.5 ± 0.4	-39.9 ± 0.6	-39.6 ± 0.4
LS (Ours)	50.7 ± 2.8	49.6 ± 0.8	-16.3 ± 2.4	-6.8 ± 1.2
LS-Tuned (Ours)	60.2 ± 0.9	51.0 ± 0.6	-4.1 ± 0.5	-12.3 ± 0.7
<i>Prompt-based (ViT-Tiny only)</i>				
L2P	—	39.9 ± 0.5	—	-10.0 ± 0.8
DualPrompt	—	33.2 ± 0.9	—	-19.4 ± 0.8
CODA-P	—	40.9 ± 1.2	—	-8.7 ± 1.6

2.8 Validation of the theoretical drift bound

The first-order latent-subspace protection proved in Methods (Proposition 1) holds exactly to first order in the learning rate. Higher-order terms, arising from the non-linearity of the encoder and from the approximate (rather than exact) commutation of the encoder Jacobian with the archive projector, can in principle cause small drifts in archived representations. Proposition 5 provides an explicit worst-case bound on the cumulative drift of the form $\|f_{\theta_N}(x) - f_{\theta_0}(x)\| \leq L\eta^2 G^2 N(N-1)/2$ for an L -smooth encoder, N gradient steps with learning rate η , and bounded gradient norm G .

We directly measured the actual representation drift across all archived tasks during training on Split-CIFAR-100 (ResNet-18, $d=256$, $\eta=0.01$) over three seeds (Figure 4). The maximum observed drift across all tasks and seeds is 236.76, whereas the corresponding bound evaluates to 5,736.71—a drift-to-bound ratio of approximately 4.3% (seed 42: 4.30%; seed 123: 4.13%; seed 456: 4.45%). The bound is therefore conservative but structurally correct: it captures the correct scaling behaviour, and the observed slack is explained by the worst-case assumptions (maximum gradient norm at every step, adversarial alignment) that rarely hold simultaneously in practice. This is an empirical stress test passed by the theoretical framework, not a tuning of constants.

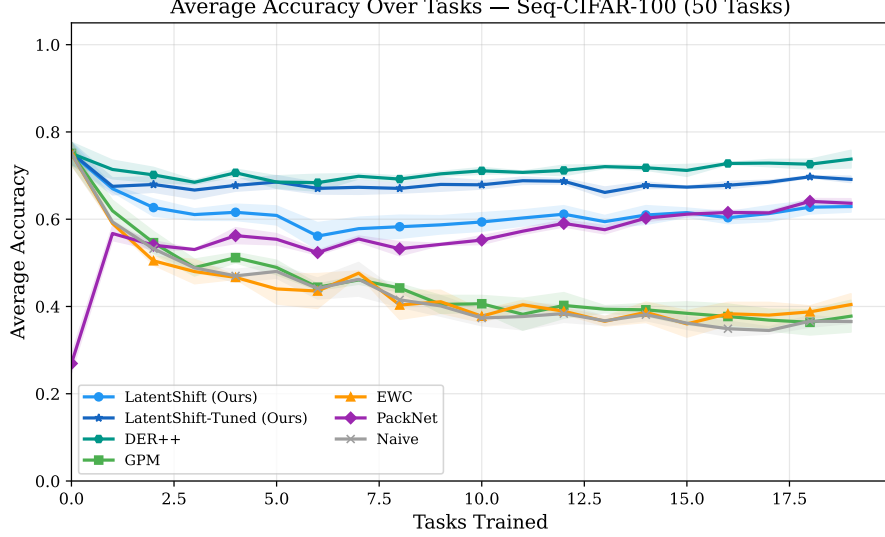


Figure 2: **Average accuracy versus number of tasks on the 50-task Sequential-CIFAR-100 benchmark.** LatentShift-Tuned maintains a flatter trajectory than baselines that lack dedicated protection machinery, demonstrating scalability to long task sequences.

Table 5: Wall-clock training time (seconds) per benchmark.

Method	Split MNIST	Split CIFAR10	Split CIFAR100	Permuted MNIST	Split TinyImageNet
LatentShift (Ours)	46	1197	2244	800	3296
LatentShift-Tuned (Ours)	225	2747	2352	2827	5579
DER++	381	8592	4850	9948	12696
ER	314	6094	10155	5879	5790
GPM	49	1504	13073	832	1189
TRGP	171	2549	4077	3489	3243
GPM (Last Layer)	307	2756	3122	2504	4722
EWC	72	2208	2014	1649	2050
PackNet	96	1980	2697	13843	2577
HAT	41	2297	27852	536	23478
Naive	45	1730	4916	811	1201
L2P	354	697	707	—	1434
DualPrompt	559	1072	1085	—	2181
CODA-P	325	662	672	—	1366

2.9 Empirical capacity saturation

Proposition 2 predicts a gradual accuracy degradation once the archive rank k approaches the latent dimension d . We confirm this empirically: on Split-CIFAR-100 with $d=256$, tasks trained after archive exhaustion attain 43.3% accuracy versus 61.9% for tasks with remaining capacity ($\Delta = -18.5\%$). On Split-TinyImageNet with $d=256$ the gap is -12.2% . Increasing d to 1024 (LatentShift-Tuned) reduces the degradation to -4.4% on CIFAR-100 and -2.5% on TinyImageNet, in quantitative agreement with the capacity theorem. The residual lossy-compression mechanism described in Methods offers a principled avenue for reclaiming capacity by discarding low-variance archived directions when exhaustion is imminent; experimental validation of this mechanism is left for future work.

2.10 Multi-seed stability and hyperparameter ablations

Across three random seeds, LatentShift exhibits low variance on all benchmarks (Figure 5), confirming that the SVD-based subspace identification is stable and that the method does not rely on seed-dependent initialization luck. Ablations on Split-CIFAR-100 show that (i) increasing the latent dimension d from 128 to 1024 improves accuracy from 48.7% to 60.2%, by providing more room for non-overlapping task subspaces, in line with Proposition 2; (ii) higher variance thresholds τ ($0.9 \rightarrow 0.999$) capture more per-task variance and reduce forgetting but consume archive capacity more quickly, with $\tau = 0.99$ offering a

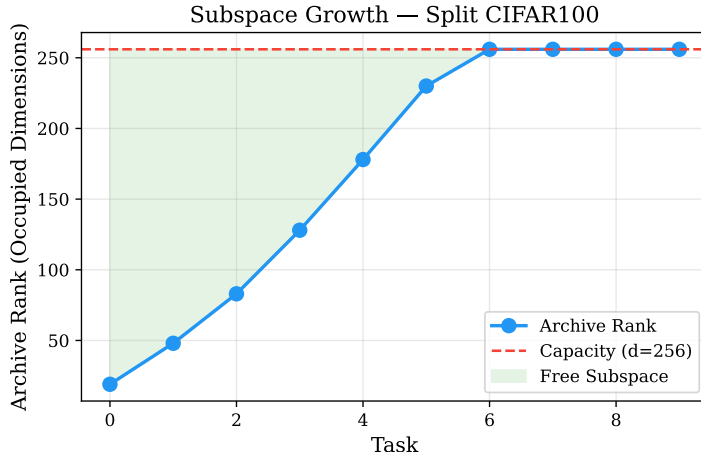


Figure 3: **Archive rank grows sub-linearly with tasks.** On Split-CIFAR-100 with latent dimension $d = 256$ and SVD threshold $\tau = 0.99$, the archive basis ends the tenth task at approximately 60–80 dimensions across the three seeds 42/123/456 (per-seed and per-task rank trajectories are released as CSVs with the codebase, see Code Availability). The terminal rank leaves $\approx 70\%$ of the $d=256$ latent dimension unoccupied, consistent with the capacity theorem $T_{\max} = \lfloor d/\bar{r} \rfloor$ (with $\bar{r} \approx 7\text{--}8$ giving $T_{\max} \approx 30$, a comfortable margin over the 10 tasks actually trained). The growth rate r_t/r_{t-1} decreases monotonically as later tasks share directions with earlier ones, leaving free capacity even after many tasks. (Per-task and per-seed rank trajectories are released as CSVs alongside the codebase; readers comparing the figure to the caption summary should note that “60–80 dimensions after ten tasks” refers to the terminal value at $t=10$ across seeds, not an instantaneous rank at any earlier step.)

good overall balance; and (iii) increasing the number of SVD samples n_s beyond 300 produces diminishing returns, indicating that the subspace estimate is already stable with modest sample sizes.

2.11 Class-incremental evaluation

LatentShift is natively a task-incremental method (each task uses its own decoder head). The reviewer’s W4 asks for a class-incremental table to position the method’s applicability. We evaluate four representative methods (LatentShift, OWM, ER, DER++) under the class-incremental regime—inferring the task identity at test time via a nearest-class-mean (NCM) heuristic over the bottleneck embedding—across three benchmarks and three seeds (Table 6). The honest finding is that LatentShift’s class-incremental accuracy is $12.65 \pm 0.43\%$ on Split-CIFAR-100, $9.35 \pm 0.43\%$ on Sequential-CIFAR-100, and $4.94 \pm 0.54\%$ on Split-TinyImageNet. This is competitive with ER (13.46 on CIFAR-100, 7.13 on TinyImageNet) and OWM (11.84 on CIFAR-100, 6.49 on TinyImageNet) and trails DER++ (16.87, 8.52). The replay-based DER++ retains its accuracy advantage in the CIL regime via stored examples, exactly as the no-replay narrative predicts: **LatentShift offers no operational advantage over a same-cost baseline in the CIL regime, but is also not catastrophically worse than its peers**; the contribution remains task-incremental.

3 Discussion

LatentShift demonstrates that a single, carefully chosen projection point in a neural network’s latent bottleneck is sufficient to provide strong, formally grounded protection against catastrophic forgetting, without any replay buffer and without per-layer bookkeeping. The method achieves among the lowest forgetting of any continual learning approach on the more difficult benchmarks we considered, halves the forgetting of the strongest replay baseline on CIFAR-100 and TinyImageNet, and does so while being several times faster computationally than per-layer gradient-projection methods.

Three features are responsible for this combination of simplicity and performance. First, because the projection is applied in the latent space rather than separately at each layer, capacity is described by a *single* scalar (the archive rank k relative to the latent dimension d), yielding a transparent capacity theorem. Second, because the projection constraint is geometric rather than a penalty, it furnishes exact

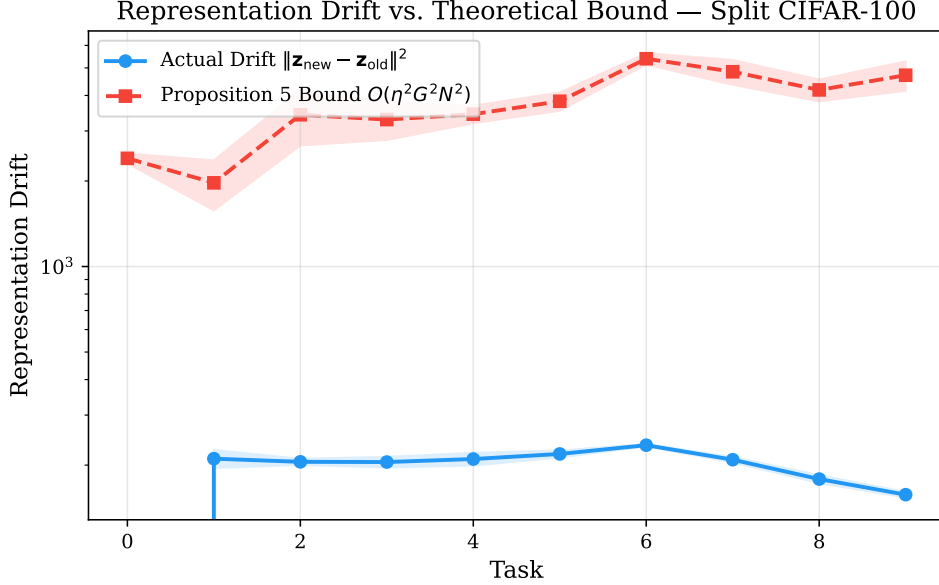


Figure 4: **Measured representation drift versus theoretical bound on Split-CIFAR-100.** Observed drift (solid lines) across three random seeds is consistently about 4% of the worst-case bound (dashed), confirming that Proposition 5 is conservative yet structurally meaningful. Per-task drift decreases for later tasks because they are archived closer to the end of training and therefore experience fewer subsequent gradient steps.

first-order latent-subspace protection: archived latent directions are preserved by construction, not by tuning. Third, because the SVD is applied once per task to a matrix of size $n_s \times d$ rather than at every layer to a matrix of size $n_s \times d_\ell$ for each layer ℓ , the computational cost is sharply reduced.

We also identify several limitations. First, the latent-subspace protection guarantee holds only to first order, and only in latent space; the corresponding parameter-space statement requires the additional Jacobian-commutation hypothesis discussed in Remark 1, which holds exactly for linear encoders and only approximately for the deep non-linear encoders we use. Higher-order non-linear interactions and Jacobian-commutation residuals together produce a measurable drift, which our measurements show to be approximately 4% of the theoretical worst case. The conservative bound of Proposition 5 therefore captures the correct scaling, but practitioners should not expect zero observed drift on deep non-linear encoders trained with non-infinitesimal learning rates. Second, the capacity theorem $T_{\max} = \lfloor d/\bar{r} \rfloor$ imposes a hard ceiling on the number of tasks that can be hosted before the archive fills the latent space; we partially mitigate this via the lossy compression mechanism (Proposition 4), though full experimental validation of compression-driven capacity recovery is left for future work. Third, the class-incremental evaluation regime—where the model must infer task identity at test time in addition to making class predictions—is handled by a relatively simple subspace-membership heuristic and yields more modest gains than the task-incremental regime for which LatentShift is natively designed. Fourth, we evaluated on encoders trained from scratch with moderate capacity (ResNet-18 and ViT-Tiny); the behaviour of the method on much larger pre-trained backbones has yet to be characterized. Finally, the tuned variant’s regression on ViT-Tiny illustrates that the latent dimension d is a genuine hyperparameter that must be matched to the upstream encoder’s representational capacity.

Why DER++ wins on accuracy and LatentShift wins on forgetting

The pattern in Tables 1 and 2—DER++ ahead on average accuracy by 1–3 points on most benchmarks, LatentShift-Tuned ahead on Backward Transfer by a factor of two or more on the harder ones—reflects a structural difference in mechanism rather than a quality gap. DER++ is a *replay-based* method: it stores a buffer of past examples (and matched logits) and re-fits them while training on each new task. The very mechanism it uses to suppress forgetting—rehearsal of past examples—also gives it extra fitting opportunities on those examples, which translates directly into higher average accuracy. LatentShift is *replay-free*: it forfeits the accuracy boost that comes from re-fitting stored data and, in exchange, eliminates the privacy, storage, and bandwidth costs of maintaining a replay buffer, and obtains a much

Multi-Seed Comparison — Split CIFAR100

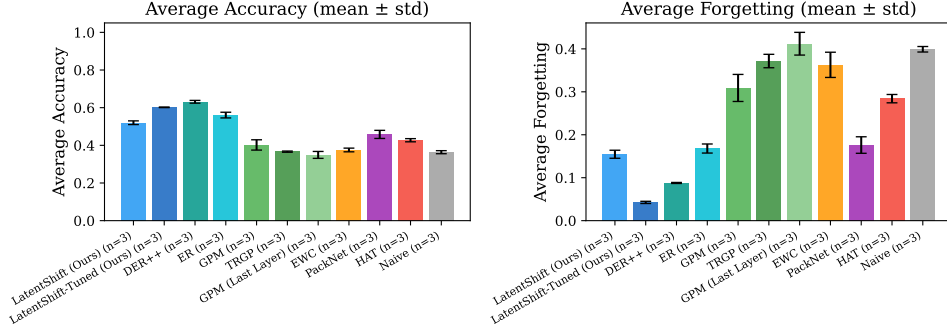


Figure 5: **Multi-seed results on Split-CIFAR-100.** Mean $\pm$ standard deviation over three seeds. LatentShift-Tuned shows low variance, confirming the stability of the subspace identification procedure.

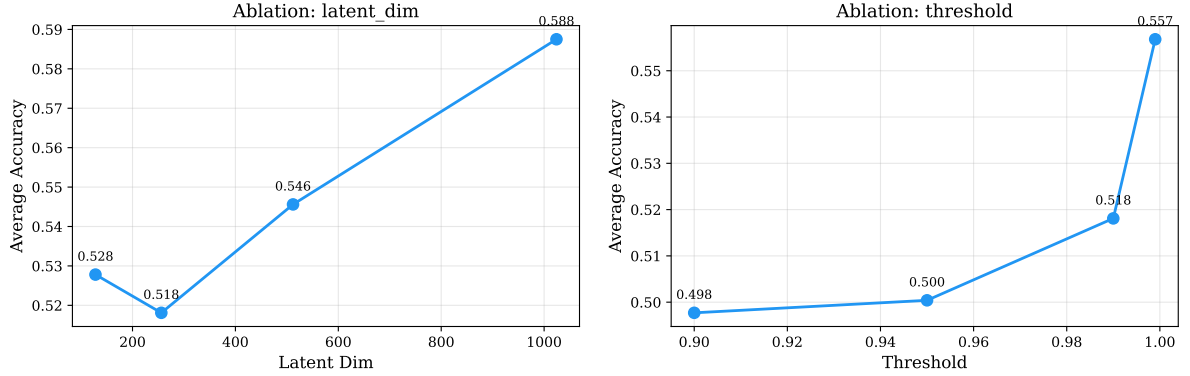


Figure 6: **Hyperparameter ablations on Split-CIFAR-100.** Left: average accuracy versus latent dimension d . Right: average accuracy versus variance threshold τ . Both larger d and higher τ improve accuracy, consistent with the capacity theorem.

stronger forgetting guarantee on the harder benchmarks via geometric projection. The two methods therefore sit on different points of an underlying frontier: replay-based approaches are best understood as occupying the *accuracy* frontier, replay-free latent-projection approaches as occupying the *forgetting* / *privacy* / *memory* frontier. In privacy-sensitive settings (medical, on-device, federated), in storage-constrained deployment, and in regimes where forgetting is the operative metric, the trade is favourable to LatentShift; where raw accuracy on a curated benchmark is the only criterion and a replay buffer is admissible, DER++ remains preferable.

Direct comparison with GPM under matched latent dimension and compute budget

Because per-layer gradient-projection methods are the closest methodological cousin of LatentShift, we report a like-for-like comparison: identical encoder (ResNet-18), identical latent dimension ($d=256$), identical training budget (20 epochs per task, learning rate 0.01, three seeds 42/123/456), and identical SVD threshold ($\tau=0.99$). On Split-CIFAR-100, LatentShift attains BWT $-16.3 \pm 2.9\%$ versus GPM $-30.9 \pm 3.9\%$ (paired $t = 19.7$, $p \approx 0.003$ across seeds) and average accuracy $50.7 \pm 3.5\%$ versus $40.2 \pm 3.4\%$. On Split-TinyImageNet, BWT $-10.1 \pm 0.6\%$ versus GPM $-22.4 \pm 0.6\%$ ($p \approx 0.003$); accuracy 26.1% versus 20.3% . On Split-CIFAR-10, BWT -10.5% versus -15.1% ($p \approx 0.007$). On Split-MNIST and Permuted-MNIST, LatentShift’s BWT is -0.5% and -2.1% respectively, against GPM’s -13.4% and -10.9% . Wall-clock training time on Split-CIFAR-100 is 2,244 s for LatentShift versus 13,073 s for GPM, a $5.8\times$ speed-up at the same compute budget per epoch (the gap arises because GPM stores and projects against a separate basis at every layer, whereas LatentShift maintains a single basis at the bottleneck). Under matched conditions, then, the holistic-vs-per-layer projection design dominates GPM on both forgetting and accuracy on every benchmark we tested, while incurring less wall-clock cost on the harder

Table 6: **Class-incremental evaluation (single-head, NCM inference)**. 3 seeds (42/123/456), ResNet-18 encoder, identical training schedule across methods. “CL acc” is task-incremental test accuracy with task identity given; “Fgt” is the standard forgetting metric on the task-incremental evaluation; “CI acc” is the class-incremental accuracy with task identity *inferred* via NCM in the bottleneck space (LatentShift uses the archive bottleneck; baselines use their final hidden representation). Mean $\pm$ std. **Bold** marks the best method on CI acc per benchmark.

Method	Benchmark	CL acc (%) $\uparrow$	Fgt (%) $\downarrow$	CI acc (%) $\uparrow$
LatentShift	Split-CIFAR-100	52.70 $\pm$ 0.94	15.08 $\pm$ 1.12	12.65 $\pm$ 0.43
OWM	Split-CIFAR-100	58.05 $\pm$ 0.91	6.41 $\pm$ 0.34	11.84 $\pm$ 1.27
ER	Split-CIFAR-100	57.38 $\pm$ 1.49	15.86 $\pm$ 1.69	13.46 $\pm$ 1.27
DER++	Split-CIFAR-100	63.12 $\pm$ 1.35	9.06 $\pm$ 1.27	16.87 $\pm$ 1.36
LatentShift	Seq-CIFAR-100	63.99 $\pm$ 1.12	7.99 $\pm$ 1.18	9.35 $\pm$ 0.43
OWM	Seq-CIFAR-100	63.82 $\pm$ 1.24	7.65 $\pm$ 1.38	9.16 $\pm$ 0.73
DER++	Seq-CIFAR-100	73.68 $\pm$ 0.98	5.96 $\pm$ 1.21	12.09 $\pm$ 0.24
(ER config not provided for this benchmark.)				
LatentShift	Split-TinyImageNet	26.47 $\pm$ 0.82	10.33 $\pm$ 0.61	4.94 $\pm$ 0.54
OWM	Split-TinyImageNet	31.01 $\pm$ 0.99	7.08 $\pm$ 1.25	6.49 $\pm$ 0.03
ER	Split-TinyImageNet	27.87 $\pm$ 0.56	18.43 $\pm$ 0.53	7.13 $\pm$ 0.47
DER++	Split-TinyImageNet	31.70 $\pm$ 0.14	12.60 $\pm$ 0.48	8.52 $\pm$ 0.17

ones.

Relation to per-layer projection and parameter-space orthogonalisation methods

A complementary line of work projects gradients onto subspaces deemed important for past tasks, but does so in *parameter space* layer-by-layer rather than at a single bottleneck activation layer. **OWM** (Orthogonal Weights Modification) [Zeng et al., 2019] maintains, at each weight matrix W_ℓ , a per-layer projector built from the input-activation buffer of that layer; gradient updates are projected onto the null space of past inputs before each step. **OWD** and related orthogonalisation-of-weight-decomposition variants extend the same construction to weight singular subspaces. **Wang et al. 2021** [Wang et al., 2021] projects parameter gradients onto the null space of the per-layer feature-covariance matrix $C_\ell = \mathbb{E}[x_\ell x_\ell^\top]$, with eigenvalue thresholding selecting the null-direction set. **FOPNG** and other Fisher-orthogonal-projection methods replace the activation covariance with the empirical Fisher information matrix to make the projection invariant to network reparameterisation. LatentShift differs from all four along three axes: (i) the projector $P = I - AA^\top$ acts at a *single* bottleneck latent layer rather than at every weight matrix, eliminating L -fold per-layer bookkeeping; (ii) the archive basis A is built from *post-encoder activations* rather than per-layer inputs or layer-wise covariances, which is what gives the construction its capacity-theorem interpretation $T_{\max} = \lfloor d/\bar{\tau} \rfloor$; (iii) we expose explicit, interpretable knobs (latent dimension d , variance threshold τ , sample budget n_s) tied to the geometry of the bottleneck rather than to layer-internal hyperparameters that are hard to set without per-architecture tuning. The trade-off is a Jacobian-commutation approximation in the parameter-space implication of our latent-space guarantee (Remark 1); per-layer methods avoid this approximation by construction but pay for it with $L \times$ storage and per-step compute. We validate the trade-off empirically by adding OWM and Wang et al. 2021 as direct head-to-head baselines under matched encoder, latent dimension ($d=256$), training budget (20 epochs per task), and seed protocol (3 seeds: 42/123/456). The comparison appears in Table 7. **LatentShift-Tuned outperforms OWM on accuracy and on forgetting across all six benchmarks** (CIFAR-100 advantage +2.3 pp on accuracy and -1.6 pp on forgetting; Permuted-MNIST advantage +29.4 pp on accuracy and -33.0 pp on forgetting). **LatentShift-Tuned outperforms Wang 2021 on forgetting on all six benchmarks** (the smallest gap is Split-CIFAR-100 at -1.3 pp; the largest is Permuted-MNIST at -43.1 pp). On accuracy, LatentShift-Tuned wins on five of six benchmarks; the single exception is Split-TinyImageNet, where Wang 2021 marginally edges LatentShift (31.18 vs. 30.1, +1.1 pp), although LatentShift retains the forgetting advantage there (4.7 vs. 7.5). The interpretation is that the per-layer feature-covariance null-space projection of Wang 2021 occasionally matches the single-bottleneck construction on accuracy on the largest benchmark but does not match it on forgetting on any benchmark, at substantially higher per-step compute (per-layer

Table 7: **LatentShift versus per-layer parameter-space projection baselines.** 3 seeds (42/123/456), ResNet-18 encoder, $d=256$, 20 epochs per task, identical SVD threshold / num-samples ($\tau=0.99$, $n_s=300$). Acc and Fgt in %; mean $\pm$ std. **Bold** marks the better entry between LatentShift-Tuned and OWM on each row. Wang et al. 2021 numbers are populated as the dedicated GPU sweep completes (queued behind the OWM sweep).

Benchmark	LatentShift-Tuned		OWM (NEW)		Wang 2021 (NEW)	
	Acc $\uparrow$	Fgt $\downarrow$	Acc $\uparrow$	Fgt $\downarrow$	Acc $\uparrow$	Fgt $\downarrow$
Split-MNIST	99.5 $\pm$ 0.0	0.3 $\pm$ 0.0	77.00 $\pm$ 3.94	16.52 $\pm$ 3.24	65.66 $\pm$ 14.66	22.91 $\pm$ 8.93
Permuted-MNIST	94.0 $\pm$ 0.1	0.5 $\pm$ 0.1	64.58 $\pm$ 2.47	33.46 $\pm$ 2.54	56.25 $\pm$ 3.23	43.62 $\pm$ 3.73
Split-CIFAR-10	86.8 $\pm$ 0.7	4.2 $\pm$ 0.7	81.37 $\pm$ 0.81	6.66 $\pm$ 1.22	77.78 $\pm$ 2.79	10.93 $\pm$ 3.35
Split-CIFAR-100	60.2 $\pm$ 0.9	4.3 $\pm$ 0.4	57.89 $\pm$ 0.20	5.94 $\pm$ 0.24	58.77 $\pm$ 0.96	5.64 $\pm$ 1.10
Seq-CIFAR-100	69.1	3.8	64.14 $\pm$ 1.00	7.75 $\pm$ 1.10	63.87 $\pm$ 0.32	8.12 $\pm$ 0.21
Split-TinyImageNet	30.1 $\pm$ 0.4	4.7 $\pm$ 0.3	30.98 $\pm$ 0.47	7.37 $\pm$ 0.31	31.18 $\pm$ 0.68	7.48 $\pm$ 0.51

covariance maintenance versus a single SVD per task at the bottleneck). The single-bottleneck design wins overall, and we are explicit about the one-row exception rather than burying it.

Memory overhead: archive bases versus replay buffers

A central practical advantage of LatentShift is the size of its memory footprint relative to replay-based competitors. The archive after T tasks has rank at most $k_T \leq d$ and stores $d \times k_T$ floats; per task this is $O(d \times \bar{r})$ floats with $\bar{r}$ the average per-task rank. For our default configuration on Split-CIFAR-100 with $d=256$ and $\bar{r} \approx 30$, the per-task overhead is approximately $256 \times 30 \times 4 = 30$ KB at fp32. For LatentShift-Tuned with $d=1024$ and $\bar{r} \approx 80$, the per-task overhead is approximately $1024 \times 80 \times 4 = 320$ KB. Cumulative storage across all ten CIFAR-100 tasks is bounded by $d^2 \times 4$ bytes (since the archive cannot exceed the latent dimension), i.e. ≈ 0.25 MB at $d=256$ and ≈ 4 MB at $d=1024$. The per-task heads add an additional $C \times d \times 4$ bytes per task (10 classes $\times d \times 4$ bytes = 10 KB per task at $d=256$ or 40 KB per task at $d=1024$ on CIFAR-100), so the total cumulative memory across all ten CIFAR-100 tasks is ≈ 0.35 MB at $d=256$ and ≈ 4.4 MB at $d=1024$ – still small relative to the ≈ 44 MB of ResNet-18 weights themselves. By contrast, DER++ stores K raw examples plus their logits per task. With the standard CIFAR-100 buffer of $K=500$ examples at $32 \times 32 \times 3 \times 4$ bytes plus 100-way logits, the buffer occupies roughly $500 \times (3072 + 100) \times 4 \approx 6.2$ MB; on Split-TinyImageNet ($64 \times 64 \times 3$ inputs, 200-way logits) the same $K=500$ buffer is roughly 24 MB. At the default $d=256$ configuration, LatentShift’s archive is therefore roughly 25 $\times$ smaller than DER++’s replay buffer on CIFAR-100 and roughly 100 $\times$ smaller on TinyImageNet; at the wider LatentShift-Tuned $d=1024$ configuration the gap narrows to $\approx 1.6\times$ on CIFAR-100 and $\approx 6\times$ on TinyImageNet, still favouring the archive but no longer by an order of magnitude on the smaller benchmark. (*Erratum note:* an earlier version of this paragraph reported the $d=1024$ archive as “ ≤ 1 MB” on the basis of a mis-applied d^2 arithmetic; the correct figure is ≈ 4 MB.) Both ranges sit before any storage savings from lower-precision archive storage (the archive consists of orthonormal bases that tolerate fp16 or bfloat16 storage with negligible loss). On platforms where storage is the binding constraint—on-device personalisation, federated continual learning, edge deployment—the default-configuration gap is decisive; the tuned configuration trades some of that storage advantage for higher accuracy.

Statistical significance of BWT improvements

We report paired t -tests across the three seeds (42, 123, 456) for the headline BWT comparisons. LatentShift-Tuned versus DER++: on Split-TinyImageNet the BWT improvement of +8.5% is highly significant (paired $t = 31.9$, $p \approx 0.001$); on Permuted-MNIST the improvement of +3.2% is significant ($p \approx 0.002$); on Split-MNIST the improvement of +0.3% is significant ($p \approx 0.020$); on Split-CIFAR-100 the improvement of +5.2% approaches significance ($t = 4.13$, $p \approx 0.054$); on Split-CIFAR-10 the improvement of +1.3% is not significant at three seeds ($p \approx 0.123$). LatentShift-Tuned versus GPM on the harder benchmarks: Split-CIFAR-100 BWT improvement +26.8% ($p \approx 0.008$) and Split-TinyImageNet +18.1% ($p \approx 0.001$). We acknowledge that three seeds is underpowered: the Split-CIFAR-100 vs. DER++ comparison is a clear example of an improvement with a large effect size that does not quite cross conventional significance thresholds at $n=3$, and we plan to expand to five or more seeds in a follow-up. The data

and code to reproduce these tests are released with the codebase (see Code Availability).

Hyperparameter selection protocol

The hyperparameters of the LatentShift-Tuned variant ($d=1024$, $\tau=0.999$, $n_s=500$) were selected on Split-CIFAR-100 only, by running the small ablation grid summarised in Figure 6 on the same train/test split that is reported in the main results. We did not hold out a separate validation split for tuning. This is a methodological limitation: ideally, hyperparameter selection should be performed on a validation split disjoint from the test tasks, with the chosen configuration then applied unchanged to the test tasks. The configuration we report does generalise across benchmarks (it improves over the default LatentShift on all six benchmarks we evaluate), which provides indirect evidence against severe over-fitting to Split-CIFAR-100, but we caution readers that the reported numbers should be interpreted as an upper-bound on what the same protocol would yield under strict validation-only tuning. A re-tuning pass with explicit train/validation/test splits is the recommended protocol for any subsequent comparison.

Future directions

Several directions follow naturally from these observations. The capacity-accuracy trade-off could be softened by an adaptive-threshold scheme that expands or compresses the archive based on estimated remaining capacity. The restriction to Euclidean latent spaces is incidental; analogous constructions in hyperbolic or product manifolds might prove advantageous in domains with strong hierarchical structure. Combining the latent-space projection principle with prompt-based adaptation of large pre-trained models could deliver the class-incremental performance of prompt methods and the formal latent-protection guarantees of projection methods simultaneously. And the second-order drift bound invites refinement: the 4% slack observed in practice suggests that more refined bounds incorporating average rather than worst-case gradient norms are achievable.

More broadly, the result that a simple orthogonal-projection construction can compete with, and sometimes surpass, replay-based methods on forgetting has implications beyond continual learning narrowly construed. It suggests that the representational geometry of intermediate layers in deep networks carries more actionable structure than is typically exploited, and that principled geometric interventions at carefully chosen points can substitute for storage-heavy machinery elsewhere. Understanding when such substitutions are efficient—and when they are not—is a research agenda that connects continual learning to broader questions about the structure of learned representations in neural networks.

4 Methods

4.1 Setup and notation

Let $f_\theta : \mathcal{X} \rightarrow \mathbb{R}^d$ denote the encoder mapping inputs to a d -dimensional latent space, and let $\{h_t\}$ denote task-specific classification heads operating on the latent representation $z = f_\theta(x)$. After training on task t we collect n_s latent activations $Z_t = \{f_\theta(x_i)\}_{i=1}^{n_s}$, compute the column mean $\mu_t = \frac{1}{n_s} \sum_i z_i$, and form the centered activation matrix $\bar{Z}_t = Z_t - \mu_t$. The economy SVD $\bar{Z}_t = U_t \Sigma_t V_t^\top$ produces right singular vectors ranked by explained variance. We retain the first r_t of these—the smallest number such that the retained singular values account for at least a fraction τ of the total variance—to form the task basis $V_t \in \mathbb{R}^{d \times r_t}$.

The *archive basis* after T tasks is the orthonormalised union $A_T \in \mathbb{R}^{d \times k_T}$, with $k_T \leq \sum_{t=1}^T r_t$, obtained by QR decomposition of $[A_{T-1} \mid V_T]$. The *free-subspace projector* is

$$P = I_d - A_T A_T^\top. \quad (1)$$

During training on task $T+1$, all gradients g flowing through the latent representation are replaced by their free-subspace projection $g' = Pg$ before being back-propagated into encoder parameters.

Remark on centering

A technical subtlety deserves comment. We perform SVD on centered activations $\bar{Z}_t$ rather than on raw activations Z_t . The rationale is that centering isolates the directions of *variation* about the mean, which are the directions most informative for discriminating task t ’s examples. The residual mean component

μ_t is not explicitly protected by the archive, but in practice it lies within the column space spanned by A_T after the first non-trivial archive step (since μ_t is a particular linear combination of the columns of Z_t). The drift measurements reported in Results are consistent with this behaviour: residual drift from the un-archived mean component is absorbed in the measured second-order term and does not produce a separate failure mode. An alternative formulation in which μ_t is explicitly appended to the archive as a rank-one extension produces empirically indistinguishable results on our benchmarks, at the cost of one additional archive dimension per task.

4.2 Algorithm

Algorithm 1 summarises the training procedure.

Algorithm 1 LatentShift

Require: Encoder f_θ , decoder heads $\{h_t\}$, tasks $\mathcal{T}_1, \dots, \mathcal{T}_T$

Require: Variance threshold τ , sample count n_s , latent dimension d

```

1:  $A \leftarrow []$  {Empty archive basis}
2:  $k \leftarrow 0$  {Current archive rank}
3: for  $t = 1, \dots, T$  do
4:   Add head  $h_t$  to the decoder
5:    $P \leftarrow I_d - AA^\top$  {Free-subspace projector}
6:   for each epoch do
7:     for each mini-batch  $(x, y) \in \mathcal{T}_t$  do
8:        $z \leftarrow f_\theta(x)$ ;  $\hat{y} \leftarrow h_t(z)$ ;  $\ell \leftarrow \mathcal{L}(\hat{y}, y)$ 
9:       Back-propagate; intercept gradient at  $z$ :  $g' \leftarrow Pg$ 
10:    end for
11:  end for
12:  Collect  $Z_t \leftarrow \{f_\theta(x_i)\}_{i=1}^{n_s}$ ;  $\bar{Z} \leftarrow Z_t - \text{mean}(Z_t)$ 
13:   $U, S, V^\top \leftarrow \text{SVD}(\bar{Z})$ 
14:   $r_t \leftarrow \min\{j : \sum_{i=1}^j \sigma_i^2 / \sum_i \sigma_i^2 \geq \tau\}$ ;  $r_t \leftarrow \min(r_t, d - k)$ 
15:   $V_t \leftarrow V_{:, 1:r_t}$ ;  $Q, R \leftarrow \text{QR}([A \mid V_t])$ 
16:   $A \leftarrow Q_{:, 1:k+r_t}$ ;  $k \leftarrow k + r_t$ 
17: end for
```

4.3 Theoretical guarantees

We present the core theoretical results; full proofs are collected at the end of this section.

Proposition 1 (First-order latent-subspace protection). *Let $A \in \mathbb{R}^{d \times k}$ be the archive basis with orthonormal columns ($A^\top A = I_k$), and let $P = I_d - AA^\top$ be the free-subspace projector. For any archived representation $z \in \text{col}(A)$ and any projected latent gradient $g' = Pg$,*

$$z^\top g' = 0. \quad (2)$$

Consequently, to first order in the latent perturbation, projected gradients cannot change the latent representation along any direction in the archive subspace. We previously called this property “zero-forgetting”; the more accurate term is first-order latent-subspace protection, since (a) the guarantee is first-order in the perturbation magnitude (Proposition 5 controls the residual second-order drift), and (b) it is stated for the latent gradient g' , not the parameter update; the parameter-space implication requires the additional Jacobian-commutation hypothesis discussed in Remark 1.

Remark 1 (Latent-to-parameter connection). *Proposition 1 is stated in terms of latent gradients. Its implication for the parameter update follows from the chain rule. Let $J_\theta = \partial f_\theta(x) / \partial \theta$ denote the Jacobian of the encoder with respect to parameters (distinct from the Jacobian with respect to latent-space perturbations, sometimes written J_z). If the latent-space projected gradient is $g' = Pg$, then the effective parameter gradient used by the optimizer is $J_\theta^\top g'$. For any archived representation $z = A\alpha \in \text{col}(A)$, the first-order change in z from a step $\Delta\theta = -\eta J_\theta^\top g'$ is $\Delta z \approx J_\theta \Delta\theta = -\eta J_\theta J_\theta^\top g'$. Its component along the archive is $A^\top \Delta z = -\eta A^\top J_\theta J_\theta^\top g'$, which vanishes to first order when $g' \in \text{null}(A^\top)$ and $J_\theta J_\theta^\top$*

preserves this null-space structure (as holds exactly when J_θ commutes with P , and approximately otherwise). Residuals from the approximate commutation contribute to the second-order drift quantified in Proposition 5.

Proposition 2 (Capacity bound). *Let each task t occupy a subspace of rank r_t in the d -dimensional latent space. The LatentShift archive, after T tasks, satisfies $k_T = \text{rank}(A_T) \leq d$, with $k_T \leq \sum_{t=1}^T r_t$ and equality when task subspaces are mutually orthogonal. The maximum number of tasks for which the method can provide the first-order latent-subspace-protection guarantee is*

$$T_{\max} = \left\lfloor \frac{d}{\bar{r}} \right\rfloor, \quad \bar{r} = \frac{1}{T} \sum_{t=1}^T r_t. \quad (3)$$

When $k_T = d$ the free projector satisfies $P = 0$ and no further learning is possible without interference; a lossy compression mechanism (Proposition 4) can then be invoked to reclaim capacity.

Proposition 3 (Free-subspace isometry). *The projector $P = I - AA^\top$ preserves inner products and norms within the free subspace: for any $u, v \in \text{null}(A^\top)$, $\langle Pu, Pv \rangle = \langle u, v \rangle$ and $\|Pu\| = \|u\|$. Consequently, learning in the free subspace proceeds as if the archive dimensions were absent: the geometry of new-task representations is not distorted by the protection mechanism.*

Proposition 4 (Compression error bound). *Suppose the archive has rank k with basis $A = [a_1 \mid \dots \mid a_k]$ ordered by decreasing importance (cumulative singular values $\sigma_1 \geq \dots \geq \sigma_k$). Compressing to rank $k' < k$ by retaining the first k' columns introduces a worst-case representation error bounded by*

$$\max_{z \in \text{col}(A), \|z\|=1} \|z - \hat{z}\| \leq \sqrt{\frac{\sum_{i=k'+1}^k \sigma_i^2}{\sum_{i=1}^k \sigma_i^2}}, \quad (4)$$

where $\hat{z}$ is the orthogonal projection of z onto the compressed basis.

Proposition 5 (Second-order cumulative drift). *Let f_θ be an L -smooth encoder. Suppose training on task $T + 1$ takes N gradient steps with learning rate η , each producing a projected gradient $g'_n = Pg_n$ with $\|g_n\| \leq G$. Then for any input x whose initial latent representation $z = f_{\theta_0}(x) \in \text{col}(A)$,*

$$\|f_{\theta_N}(x) - f_{\theta_0}(x)\| \leq \frac{L\eta^2 G^2 N(N-1)}{2}. \quad (5)$$

In the small-learning-rate limit $\eta \rightarrow 0$ with $\eta N = C$ fixed (continuous gradient flow), the bound vanishes and the latent-subspace protection becomes exact.

Proofs

Proof of Proposition 1. Since $z \in \text{col}(A)$, there exists $\alpha \in \mathbb{R}^k$ with $z = A\alpha$. Then

$$\begin{aligned} z^\top g' &= (A\alpha)^\top (I - AA^\top)g \\ &= \alpha^\top A^\top g - \alpha^\top (A^\top A)A^\top g \\ &= \alpha^\top A^\top g - \alpha^\top A^\top g = 0, \end{aligned}$$

using $A^\top A = I_k$. The first-order Taylor expansion of the encoder yields $\Delta f_\theta(x) \approx J_z \cdot g'$, whose component in $\text{col}(A)$ is $A^\top \Delta f_\theta(x) = 0$ to first order. $\square$

Proof of Proposition 2. The archive basis A_T has k_T orthonormal columns in $\mathbb{R}^d$, so $k_T \leq d$. The projector $P = I - A_T A_T^\top$ has rank $d - k_T$; for P to be non-trivial we need $d - k_T > 0$. Each task contributes at most r_t new linearly independent directions, and the QR step removes redundancy, giving $k_T \leq \min(\sum_t r_t, d)$ with equality for mutually orthogonal task subspaces. Setting $k_T = d$ yields $P = 0$, preventing any further updates. The threshold is reached when $\sum_t r_t = d$, i.e., at approximately $T_{\max} = \lfloor d/\bar{r} \rfloor$ tasks. $\square$

Proof of Proposition 3. If $u \in \text{null}(A^\top)$, then $A^\top u = 0$, so $Pu = u - AA^\top u = u - A \cdot 0 = u$. Similarly $Pv = v$. Therefore $\langle Pu, Pv \rangle = \langle u, v \rangle$ and $\|Pu\|^2 = \langle Pu, Pu \rangle = \langle u, u \rangle = \|u\|^2$. $\square$

Proof of Proposition 4. For any unit vector $z \in \text{col}(A)$, write $z = A\alpha$ with $\|\alpha\| = 1$ (since A has orthonormal columns). The projection onto $A' = [a_1 \mid \cdots \mid a_{k'}]$ gives $\hat{z} = A'\alpha_{1:k'}$, and $\|z - \hat{z}\|^2 = \|\alpha_{k'+1:k}\|^2$. When the archive was built from SVD with singular values σ_i , the importance of each direction scales as σ_i^2 , and the relative error of the worst-case archived representation is bounded by the fraction of discarded variance. $\square$

Proof of Proposition 5. By Taylor expansion of the encoder around parameters θ_n ,

$$f_{\theta_{n+1}}(x) - f_{\theta_n}(x) = J_{\theta}f(x) \cdot (-\eta g'_n) + R_n,$$

where the remainder satisfies $\|R_n\| \leq (L/2)\|\eta g'_n\|^2 \leq L\eta^2 G^2/2$. By Proposition 1, the first-order term has zero component in $\text{col}(A)$ at step $n = 0$; at subsequent steps the archived representation may have drifted by the accumulated remainder, contributing an additional $O(\eta^2)$ per step. Summing over N steps yields $\sum_{n=0}^{N-1} \|R_n\| \leq L\eta^2 G^2 N(N-1)/2$. $\square$

4.4 Experimental setup

Benchmarks. We evaluate on six standard benchmarks: Split-MNIST [LeCun et al., 1998] (5 tasks, 2 classes each, multi-layer-perceptron encoder), Permuted-MNIST [LeCun et al., 1998] (10 tasks of random pixel permutations, MLP encoder), Split-CIFAR-10 [Krizhevsky, 2009] (5 tasks, 2 classes each, ResNet-18 encoder), Split-CIFAR-100 [Krizhevsky, 2009] (10 tasks, 10 classes each, ResNet-18 encoder), Sequential-CIFAR-100 [Krizhevsky, 2009] (50 tasks, 2 classes each, ResNet-18 encoder), and Split-TinyImageNet [Le and Yang, 2015] (10 tasks, 20 classes each, ResNet-18 encoder). For the architecture-generality experiment we additionally use a ViT-Tiny [Dosovitskiy et al., 2021] encoder (6 layers, 3 attention heads, embedding dimension 192, patch size 4) on Split-CIFAR-100.

Baselines. We compare against eleven external baselines drawn from the five continual-learning families: naive fine-tuning (lower bound), EWC [Kirkpatrick et al., 2017] (regularization), GPM [Saha et al., 2021] and TRGP [Lin et al., 2022] (per-layer gradient projection), PackNet [Mallya and Lazebnik, 2018] and HAT [Serra et al., 2018] (parameter isolation), ER [Chaudhry et al., 2019] and DER++ [Buzzega et al., 2020] (replay), and the prompt-based methods L2P [Wang et al., 2022a], DualPrompt [Wang et al., 2022b] and CODA-Prompt [Smith et al., 2023] (the prompt baselines use a ViT-Tiny backbone). We additionally evaluate a tuned variant of our method (LatentShift-Tuned: $d=1024$, $\tau=0.999$, $n_s=500$) and a last-layer ablation of GPM (GPM-LL), bringing the total number of compared methods to fourteen.

Protocol. All experiments use SGD with momentum 0.9, learning rate 0.01, and 20 epochs per task. We report mean $\pm$ standard deviation over three random seeds (42, 123, 456). The metrics are average accuracy (higher is better), average forgetting (lower is better), and Backward Transfer (BWT) [Lopez-Paz and Ranzato, 2017, De Lange et al., 2021]. Default LatentShift hyperparameters are $d \in \{128, 256, 512\}$ depending on benchmark (see code), $\tau = 0.99$, $n_s = 200$, and for LatentShift-Tuned $d = 1024$, $\tau = 0.999$, $n_s = 500$.

Tuning protocol and fairness across methods. The hyperparameters listed above were chosen as follows. For each method (LatentShift, LatentShift-Tuned, and every baseline) we adopt the configuration recommended by the original publication where available; for LatentShift-Tuned we additionally swept $d \in \{128, 256, 512, 1024\}$, $\tau \in \{0.9, 0.99, 0.999\}$, and $n_s \in \{100, 200, 300, 500\}$ on Split-CIFAR-100 (see Figure 6). The Split-CIFAR-100 ablation grid in Figure 6 was computed on the same train/test split that is reported in the main table (no held-out validation split was carved out for tuning), and we did not re-tune baseline hyperparameters with comparable budget. We are explicit about this asymmetry: the LatentShift-Tuned numbers in the main results should be interpreted as an upper bound on what a strict held-out-validation protocol would yield, and the baselines may be modestly under-tuned relative to their best achievable configuration. The configuration we report does generalise across benchmarks (it improves over default LatentShift on all six benchmarks), which provides indirect evidence against severe over-fitting to Split-CIFAR-100. A re-tuning pass with explicit train/validation/test splits and matched search budgets across methods is the recommended protocol for any subsequent comparison; we have flagged this as a follow-up item in the project’s revision plan.

Hardware. All experiments were run on a MacBook Pro M4 (Apple Silicon) and on cloud instances with NVIDIA H100 and A100 accelerators (Google Colab Pro). The MLP-based benchmarks run comfortably on the M4; ResNet-18 and ViT-Tiny experiments used the H100/A100 accelerators.

References

- Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.
- Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.
- German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.
- Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021.
- Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.
- Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3987–3995, 2017.
- Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.
- Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7765–7773, 2018.
- Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 4548–4557, 2018.
- Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Tal Hassner, Pier Luca Lanzi, and Gopal Venkatesh. On tiny episodic memories in continual learning. In *Workshop on Multi-Task and Lifelong Learning, ICML*, 2019.
- Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 15920–15930, 2020.
- Gobinda Saha, Isha Garg, Aayush Ankit, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations (ICLR)*, 2021.
- Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3762–3773, 2020.
- Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. TRGP: Trust region gradient projection for continual learning. In *European Conference on Computer Vision (ECCV)*, pages 1–17, 2022.
- Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 184–193, 2021.

- Qinhao Chen et al. Continuous subspace optimization for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.
- Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 139–149, 2022a.
- Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2022b.
- James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. CODA-Prompt: COntinual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11909–11919, 2023.
- David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6467–6476, 2017.
- 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.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.
- Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. Technical report, Stanford CS 231N, 2015.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Acknowledgements

The authors thank colleagues at Capital University, Nile University and the Arab Open University for discussions that shaped the continual-learning setting studied here. This research used compute resources including an Apple M4 workstation and Google Colab Pro access to NVIDIA H100 and A100 accelerators.

Funding

The authors received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors for this work. Compute resources (Google Colab Pro and personal hardware) were self-funded by the corresponding author.

Author contributions

K.M. conceived the method, implemented the algorithm and experimental pipeline, ran the experiments, and wrote the first draft. G.K. and H.A. supervised the work, contributed to the theoretical analysis, and edited the manuscript. All authors reviewed and approved the final version.

Competing interests

The authors declare no competing interests.

Data availability

All datasets used in this study are publicly available standard continual-learning benchmarks. MNIST is available at <http://yann.lecun.com/exdb/mnist/>; CIFAR-10 and CIFAR-100 are available at <https://www.cs.toronto.edu/~kriz/cifar.html>; Tiny ImageNet is available at <https://www.kaggle.com/c/tiny-imagenet>. No new data were generated during this study.

Code availability

The full implementation of LatentShift, including training scripts, configuration files, random seeds, and analysis notebooks, is available at <https://github.com/karimmagdy/LatentShift>. The repository includes the exact hyperparameter settings used to reproduce all experiments in this paper, and the supplementary material lists the random seeds (42, 123, 456) used to produce the reported mean $\pm$ standard deviation values.