RETHINKING THE STABILITY–PLASTICITY DILEMMA IN CONTINUAL LEARNING THROUGH FUNCTIONAL EQUIVALENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning requires models to acquire new knowledge without disrupting previously learned capabilities, creating a fundamental tension between stability and plasticity. Stability, however, is often imposed without distinguishing which internal changes are actually detrimental to retention. This paper reconceptualizes continual learning through *functional equivalence*: internal states that realize the same function are treated as equivalent, thereby leaving functionally invariant degrees of freedom available for adaptation. In multiclass softmax classification, this equivalence admits an exact decomposition of logits into relative class scores that determine class probabilities within a task and common score shifts that leave them unchanged. Building on this structure, we instantiate the principle in continual learning through **FEAST** (**F**unctional **E**quivalence–guided **A**llocation of **ST**ability and Plasticity). FEAST yields strong Class-IL and Task-IL performance compared with methods that do not retain an additional full teacher network. We further provide a mechanistic account of how these gains arise: stability is concentrated on function-defining coordinates, while functionally invariant freedom provides an alternative path for adaptation. This division of labor improves acquisition while reducing function-defining output change and forgetting. Together, these results recast the stability–plasticity dilemma in terms of the functional consequences of change rather than change itself.

1 INTRODUCTION

Continual learning requires a model to retain previously acquired knowledge while remaining able to learn from new experience, creating a fundamental tension between stability and plasticity (McCloskey & Cohen, 1989; Mermillod et al., 2013; Dohare et al., 2024). Existing approaches address this tension through mechanisms such as restricting parameter updates, replaying past experience, or preserving previous outputs (Kirkpatrick et al., 2017; Chaudhry et al., 2019; Li & Hoiem, 2016; Buzzega et al., 2020). Although these approaches differ in mechanism, each must implicitly or explicitly determine what should be protected as learning proceeds. The central question is therefore not only how much stability to impose, but what, exactly, must remain stable for previously learned capabilities to be retained.

Biological learning systems offer a useful clue to this question. Neural representations can undergo substantial reorganization across timescales while task-relevant information and behavior remain comparatively stable, a phenomenon often referred to as representational drift (Mankin et al., 2012; Driscoll et al., 2017; Rule et al., 2020; Ziv et al., 2013). Related work on output-null population activity shows that substantial neural change can occur along dimensions that do not affect a downstream readout (Kaufman et al., 2014). Together, these observations illustrate that substantial internal change can coexist with preserved task-relevant function.

This distinction extends naturally beyond biological systems. More generally, when multiple internal states realize the same function, preserving that function need not require preserving a particular state. This motivates a function-relative notion of stability in continual learning, which we formalize through *functional equivalence*. We distinguish function-defining coordinates, whose changes alter the function being preserved, from functionally invariant coordinates, whose changes leave that

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.

function unchanged. Stability can therefore be imposed selectively on function-defining coordinates, while functionally invariant coordinates remain available for adaptation. This separation provides a principled basis for allocating stability and plasticity.

We instantiate this principle in class-incremental learning (Class-IL) (van de Ven et al., 2022; Masana et al., 2023), where new classes are introduced sequentially and all learned classes compete jointly under a single shared classifier (hereafter, full-way learning). In multiclass softmax classification, functional equivalence takes an exact form: relative class scores determine class probabilities within a task, whereas a common shift of all class scores leaves those probabilities unchanged while still affecting competition across tasks. Building on this structure, we introduce **FEAST** (**F**unctional **E**quivalence–guided **A**llocation of **ST**ability and Plasticity), a concrete continual-learning realization of this principle. FEAST preserves a reference map from features to previously learned class contrasts, encourages the evolving encoder to remain compatible with that map, and leaves the classifier's common-score coordinates available for adaptation. The remaining cross-task competition is reconciled through one scalar offset per task block. Thus, the same decomposition determines what must be preserved, where adaptation can occur, and what remains to be resolved across tasks.

Under matched conditions, FEAST substantially outperforms the compared methods without an additional full teacher network in both Class-IL and task-incremental learning (Task-IL) across CIFAR-100 and TinyImageNet, while requiring substantially less retained memory than teacher-based approaches. Controlled interventions further show that functionally invariant classifier freedom improves acquisition while reducing function-defining output change and forgetting. These results support the view that successful continual learning depends not simply on suppressing change, but on allocating it according to its functional consequence.

Our contributions are as follows:

- We formulate functional equivalence as a principle for allocating stability and plasticity in continual learning, distinguishing changes that alter a preserved function from those that leave it unchanged.
- We derive its exact structure in multiclass softmax classification and instantiate it through FEAST, which preserves function-defining class contrasts while retaining functionally invariant classifier freedom for adaptation under full-way Class-IL. FEAST yields strong Class-IL and Task-IL performance without retaining an additional full teacher network.
- We provide a mechanistic account of how stability and plasticity are allocated across functional coordinates in FEAST, showing how this division of labor gives rise to strong acquisition and retention.

2 Related Work

Selective protection and functionally silent change. Several prior works share the broader motivation behind our functional-equivalence view by imposing stability selectively rather than constraining all model changes uniformly (Aljundi et al., 2018; Farajtabar et al., 2020; Saha et al., 2021). Closest to our motivation, Anthes et al. (2025) distinguish readout-visible from readout-null representational change and shows that the latter can support plasticity without directly impairing an old task. Our work shares this broader view, but takes functional equivalence itself as the allocation principle and derives a continual-learning rule from it.

Related continual-learning designs. Several components of FEAST have methodological parallels in existing Class-IL approaches. DER preserves previous behavior by matching stored logits (Buzzega et al., 2020), making it a particularly relevant output-preservation baseline, but does not distinguish logit changes by their functional consequence. SS-IL addresses old–new interference by separating softmax normalization across task groups (Ahn et al., 2021), whereas FEAST retains full-way learning and structures how historical classifier coordinates respond to that competition. BiC learns a low-dimensional classifier correction for old–new bias (Wu et al., 2019), whereas FEAST's block-common reconciliation follows from the functional degree of freedom left unresolved during learning. Although these mechanisms have parallels in prior work, FEAST derives preservation, adaptation, and reconciliation jointly from the same principle of functional equivalence. Appendix A provides a more detailed comparison with these methods and additional related approaches.

3 FUNCTIONAL EQUIVALENCE AND FUNCTION-RELATIVE STABILITY

3.1 FUNCTIONAL EQUIVALENCE AND FUNCTIONAL COORDINATES

Let $f : \mathbb{R}^n \to \mathcal{Y}$ be a fixed map from an internal state $\boldsymbol{s} \in \mathbb{R}^n$ to a functional output. The choice of f specifies what is to be preserved. For example, preserving an entire probability distribution imposes a stronger requirement than preserving only its most likely class. We call two states *functionally equivalent* with respect to f if

$$\boldsymbol{s} \sim_f \boldsymbol{s}' \quad \Longleftrightarrow \quad f(\boldsymbol{s}) = f(\boldsymbol{s}'). \tag{1}$$

The *functional equivalence class* of $\boldsymbol{s}$ is the set $[\boldsymbol{s}]_f := \{\boldsymbol{s}' : f(\boldsymbol{s}') = f(\boldsymbol{s})\}$. Thus, preserving the functional output does not require preserving a particular internal state; it requires the state to remain within the same functional equivalence class.

As a simple case, suppose every equivalence class is a translate of the same linear subspace $\mathcal{N} \subseteq \mathbb{R}^n$, so that $[\boldsymbol{s}]_f = \boldsymbol{s}+\mathcal{N}$. In other words, adding a vector in $\mathcal{N}$ leaves the output unchanged, and all states with the same output differ in this way. Then we can define the Euclidean orthogonal projectors Q and P onto $\mathcal{N}$ and its orthogonal complement $\mathcal{N}^\perp$, respectively:

$$Q := \mathrm{Proj}_{\mathcal{N}}, \qquad P := I_n - Q, \tag{2}$$

which gives

$$\boldsymbol{s} = P\boldsymbol{s} + Q\boldsymbol{s}, \qquad f(\boldsymbol{s}') = f(\boldsymbol{s}) \quad \Longleftrightarrow \quad P\boldsymbol{s}' = P\boldsymbol{s}. \tag{3}$$

We refer to $P\boldsymbol{s}$ as the *function-defining coordinates*, since they uniquely identify the functional equivalence class, and to $Q\boldsymbol{s}$ as the *functionally invariant coordinates*, since changing this component while holding $P\boldsymbol{s}$ fixed leaves f unchanged. Here, "coordinates" denotes the projected components in $\mathbb{R}^n$, not a choice of independent basis coordinates. Their numbers of independent degrees of freedom are $n - \dim \mathcal{N}$ and $\dim \mathcal{N}$, respectively.

This decomposition is exact when the equivalence classes are translates of a common linear subspace. More generally, functional equivalence can arise from scaling, rotation, and other symmetries, as illustrated in Table 1.

Table 1: Examples of exact functional equivalence. Different internal states can produce the same functional output, even when their equivalence classes are not affine.

Function	Equivalent states	Geometry
Linear readout $W\boldsymbol{h}$	$\boldsymbol{h}' = \boldsymbol{h} + \boldsymbol{n},\ \boldsymbol{n} \in \ker W$	affine
Softmax probabilities	$\boldsymbol{z}' = \boldsymbol{z} + a\mathbf{1},\ a \in \mathbb{R}$	affine
Correlation $\mathrm{Corr}(\Sigma)$	$\Sigma' = D\Sigma D,\ D \succ 0$ diagonal	scaling orbit
Factor covariance $\Lambda\Lambda^\top$	$\Lambda' = \Lambda R,\ R^\top R = I$	orthogonal orbit

We focus here on the affine case because it admits a state-independent decomposition into function-defining and functionally invariant coordinates. Multiclass softmax classification provides a particularly simple exact instance, derived next.

3.2 EXACT FUNCTIONAL COORDINATES OF SOFTMAX CLASSIFICATION

For a K-class classifier, take the functional output to be the entire softmax probability vector, $f(\boldsymbol{z}) = \mathrm{softmax}(\boldsymbol{z})$, where $\boldsymbol{z} \in \mathbb{R}^K$ is the logit vector, viewed as an internal state. Then, $\boldsymbol{z} \sim_f \boldsymbol{z} + c\mathbf{1}$ for any $c \in \mathbb{R}$. Here,

$$f(\boldsymbol{z}') = f(\boldsymbol{z}) \quad \Longleftrightarrow \quad \boldsymbol{z}' - \boldsymbol{z} \in \mathrm{span}\{\mathbf{1}\}, \tag{4}$$

where $\mathbf{1} \in \mathbb{R}^K$ is the all-ones vector.

With $\mathcal{N} = \mathrm{span}\{\mathbf{1}\}$, Eqs. 2 and 3 give

$$Q = \frac{1}{K}\mathbf{1}\mathbf{1}^\top, \qquad P = I_K - Q, \qquad \boldsymbol{z} = \widetilde{\boldsymbol{z}} + a\mathbf{1}, \tag{5}$$

where $\widetilde{\boldsymbol{z}} := P\boldsymbol{z}$ is the centered logit vector and $a := \frac{1}{K}\mathbf{1}^\top\boldsymbol{z}$ is the mean logit. Here, Q extracts the component shared across classes, whereas P removes this shared component, retaining the relative

scores (Fig. 1a). Since two logit vectors have the same softmax output if and only if their centered logits agree, the centered logit vector $\widetilde{\boldsymbol{z}}$ provides the function-defining coordinates, with $K-1$ independent degrees of freedom, while the scalar a parameterizes the single functionally invariant degree of freedom (see Appendix B.1).

3.3 Function-Defining Coordinates Determine Pairwise Logit Margins

The function-defining coordinates have a direct decision-level interpretation. For distinct classes c and c', the pairwise logit margin is $m_{cc'}(\boldsymbol{z}) := z_c - z_{c'} = (P\boldsymbol{z})_c - (P\boldsymbol{z})_{c'}$. The common component cancels from every pairwise comparison, so $P\boldsymbol{z}$ fully determines all pairwise logit margins.

For two logit vectors, let $\boldsymbol{\delta} := P(\boldsymbol{z}' - \boldsymbol{z})$ denote their function-defining change. The corresponding margin change is $\Delta m_{cc'} = \delta_c - \delta_{c'}$, and the changes across all unordered class pairs satisfy

$$\sum_{c<c'} (\Delta m_{cc'})^2 = K\|\boldsymbol{\delta}\|_2^2. \tag{6}$$

Thus, squared change in the function-defining coordinates measures aggregate change in pairwise logit margins exactly, up to a factor of K. This provides a direct link between changes in centered logits and changes in pairwise logit margins, without requiring the entire logit vector to remain fixed. Appendix B.2 derives the identity and a sufficient condition for preserving the predicted class.

3.4 Functionally Invariant Coordinates Provide Flexibility for Adaptation

Functional invariance is relative to the specified function. To see this, partition the classes into disjoint groups $\{\mathcal{C}_g\}_{g=1}^G$, and write each group's logits as $\boldsymbol{z}_g = \widetilde{\boldsymbol{z}}_g + a_g\mathbf{1}$, with $\widetilde{\boldsymbol{z}}_g$ centered within the group. The common score a_g does not affect the class probabilities within group g, as it does not change relative scores inside the group. The dependence on input x is omitted for notational simplicity.

The common scores, however, affect the probability assigned to each group. Group g's total unnormalized mass $\sum_{c\in\mathcal{C}_g} \exp z_{g,c}$ can be written as $\exp(e_g + a_g)$, where e_g is the group's centered log-evidence, defined as $e_g := \log \sum_{c\in\mathcal{C}_g} \exp \widetilde{z}_{g,c}$. Then $p(y \in \mathcal{C}_g) \propto \exp(e_g + a_g)$ across groups.

Thus, the common-score vector $\boldsymbol{a} := (a_g)_{g=1}^G$ is functionally invariant for within-group discrimination but not for the joint softmax over classes. These common scores therefore provide natural degrees of freedom for adaptation, since their relative values can change competition among groups without altering class probabilities within any group.

4 Learning with Separated Functional Coordinates

Section 3 shows that, in multiclass softmax classification, function-defining coordinates determine relative class scores, whereas functionally invariant coordinates leave class probabilities within a group unchanged while remaining consequential for competition among groups. We now instantiate this distinction in a Class-IL setting with a single shared classifier, where we define the *within-task function* as the softmax distribution over each task's classes. This setting creates two simultaneous requirements: preserving previously learned within-task functions while adapting competition across tasks as new classes are introduced.

FEAST follows a logical pipeline: (1) learn with a single shared classifier, (2) keep a *reference contrast map* that maps features to the previously learned class contrasts, (3) allow the encoder to change while encouraging the outputs of this map to remain aligned with their reference targets (*functional encoder alignment*; FEA), (4) allow classifier parameters for previous tasks to change while preserving the corresponding reference contrast maps (*functional classifier alignment*; FCA), and finally (5) reconcile the remaining cross-task competition.

4.1 Learning with a Single Shared Classifier

We consider T tasks in the Class-IL setting, where task k introduces a disjoint class set $\mathcal{C}_k$ with $|\mathcal{C}_k| = K_k$. Let $\mathcal{C}_{\le t} := \bigcup_{k=1}^t \mathcal{C}_k$ denote the classes observed up to task t, and $\mathcal{C} := \bigcup_{k=1}^T \mathcal{C}_k =$

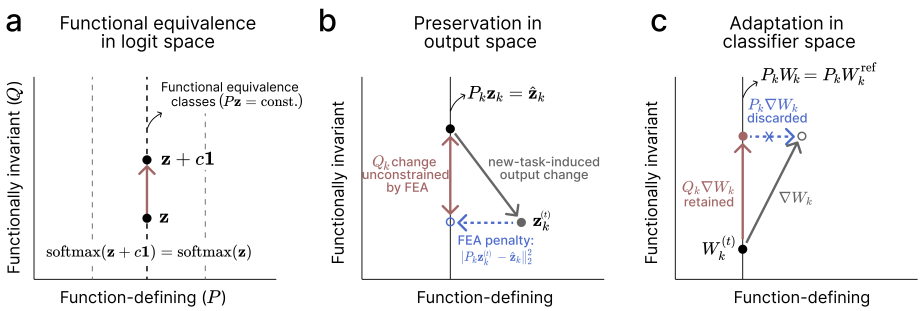


Figure 1: **Functional equivalence separates preservation from adaptation.** **(a)** Softmax-equivalent logits can differ along Q as long as $P\boldsymbol{z}$ is fixed. **(b)** Functional encoder alignment (FEA) penalizes deviations of the function-defining output $P_k\boldsymbol{z}_k^{(t)}$ from its reference target $\widehat{\boldsymbol{z}}_k$, while leaving changes along Q_k unconstrained. **(c)** Functional classifier alignment (FCA) discards the P_k component and retains the Q_k component in previous-task classifier updates, keeping P_kW_k fixed.

$\{1, \ldots, C\}$ the full class set. All classes share an encoder $\boldsymbol{h}_\theta(x) \in \mathbb{R}^d$, which maps the input $\boldsymbol{x}$ onto features $\boldsymbol{h}$, and a single shared classifier $\boldsymbol{z}(x) = W\boldsymbol{h}_\theta(x) + \boldsymbol{b}$ that maps the features to logits, where $W \in \mathbb{R}^{C\times d}$ and $\boldsymbol{b} \in \mathbb{R}^C$.

All C classifier outputs are allocated from the beginning. Current-task examples are trained with full-way cross-entropy $\mathcal{L}_{\text{CE}}^{\text{full}}$ over all outputs, including outputs corresponding to classes that are not yet observed. Thus, training includes competition with both old and as-yet-unobserved classes. At evaluation, Class-IL compares only classes in $\mathcal{C}_{\leq t}$, whereas Task-IL restricts prediction to the classes of the supplied task. The full-way training objective above alone does not preserve learned within-task decisions, motivating the following steps.

4.2 Keeping a Reference Contrast Map for Within-Task Functions

We separate two components of retaining previously learned within-task functions. First, we keep the classifier's map from features to class contrasts fixed. At the same time, we encourage the encoder to produce features that preserve those contrasts under the fixed map. This fixed map, which we call the *reference contrast map*, serves as the interface between the encoder and classifier.

To define the reference contrast map for task k, let W_k and $\boldsymbol{b}_k$ denote the rows and entries indexed by $\mathcal{C}_k$, respectively, defining the classifier block $\boldsymbol{z}_k(x) = W_k\boldsymbol{h}_\theta(x) + \boldsymbol{b}_k$. This block is a subset of the shared classifier, not a separate task-specific head. We use the corresponding blockwise projectors $Q_k = \mathbf{1}_{K_k}\mathbf{1}_{K_k}^\top/K_k$ and $P_k = I_{K_k} - Q_k$.

Let W_k^{ref} and $\boldsymbol{b}_k^{\text{ref}}$ denote the weight and bias of the classifier block at the end of training on task k, respectively. We hold their function-defining components fixed during subsequent learning:

$$P_kW_k = P_kW_k^{\text{ref}}, \qquad P_k\boldsymbol{b}_k = P_k\boldsymbol{b}_k^{\text{ref}}. \tag{7}$$

Formally, the reference contrast map is the centered affine map $(P_kW_k^{\text{ref}}, P_k\boldsymbol{b}_k^{\text{ref}})$. It specifies how features are converted into the relative scores that determine the within-task functions. This constraint fixes the classifier's class contrasts, not its predictions under a changing encoder. A feature change $\Delta\boldsymbol{h}$ produces a centered-output change $P_kW_k^{\text{ref}}\Delta\boldsymbol{h}$; changes in the kernel of $P_kW_k^{\text{ref}}$ are invisible to the reference contrast map. On the encoder side, the objective is therefore to preserve what the reference contrast map reads from changing features, rather than the features themselves. This reference contrast map guides which aspects of the encoder and classifier are maintained and left for plasticity, as described below.

4.3 Functional Encoder Alignment with the Reference Contrast Map

We encourage compatibility of the encoder with the reference contrast map through FEA (Fig. 1b). FEA penalizes changes in within-block class contrasts while leaving the common score uncon-

strained by incorporating a loss term $\mathcal{L}_{\mathrm{FEA}}^{(t)}$ for training on task t:

$$\mathcal{L}_{\mathrm{FEA}}^{(t)} = \mathbb{E}_{(x,k)\sim\mathcal{M}_{t-1}}\left[\frac{1}{K_k}\left\|\widetilde{\boldsymbol{z}}_k(x) - \widehat{\boldsymbol{z}}_k(x)\right\|_2^2\right], \qquad \mathcal{L}_t = \mathcal{L}_{\mathrm{CE}}^{\mathrm{full}} + \lambda\mathcal{L}_{\mathrm{FEA}}^{(t)} \tag{8}$$

Here, x denotes a memory exemplar from block k, $\mathcal{M}_{t-1}$ denotes the set of exemplars up to task $t-1$, and $\widehat{\boldsymbol{z}}_k(x)$ denotes its stored centered scores, refreshed at the preceding task boundary and held fixed during the current task. $\widetilde{\boldsymbol{z}}_k(x) := P_k\boldsymbol{z}_k(x)$ denotes the current centered scores. λ is constant across tasks, and $\mathcal{L}_{\mathrm{FEA}}^{(1)} := 0$. Input-view and minibatch dependence are suppressed for readability. The precise estimator and implementation conventions appear in Appendices E.1 and E.2.

Because the function-defining classifier component is fixed by Eq. 7, changes in $\widetilde{\boldsymbol{z}}_k(x)$ arise through the evolving encoder. FEA therefore penalizes encoder changes that alter the function-defining output of the reference contrast map, while leaving changes invisible to that map unpenalized.

4.4 Functional Classifier Alignment While Retaining Invariant Flexibility

Fixing the function-defining component of a classifier block for a previous task does not require freezing the entire block. Any classifier block satisfying Eq. 7 can be written as

$$W_k = P_kW_k^{\mathrm{ref}} + Q_kV_k, \qquad \boldsymbol{b}_k = P_k\boldsymbol{b}_k^{\mathrm{ref}} + Q_k\boldsymbol{c}_k, \tag{9}$$

for arbitrary V_k and $\boldsymbol{c}_k$. Thus, an old classifier block can continue to change along functionally invariant coordinates while its function-defining component remains fixed.

During full-way learning, we retain this freedom by projecting the gradients of each previous classifier block onto the functionally invariant subspace (Fig. 1c):

$$\nabla_{W_k}\mathcal{L}_t \leftarrow Q_k\nabla_{W_k}\mathcal{L}_t, \qquad \nabla_{\boldsymbol{b}_k}\mathcal{L}_t \leftarrow Q_k\nabla_{\boldsymbol{b}_k}\mathcal{L}_t. \tag{10}$$

The classifier-side gradient of FEA lies in the P_k subspace and is therefore removed by this projection. We call this projection FCA (Fig. 1c). After each optimizer step, we reproject the block onto the admissible family in Eq. 9, restoring its reference P_k component exactly. Previous classifier blocks therefore remain involved in full-way learning, but only through changes that preserve their learned class contrasts. Optimizer-state and weight-decay details are given in Appendix D.

4.5 Reconciling Cross-Block Competition

The preceding steps protect class contrasts for old tasks while allowing common scores to adapt. This common-score freedom corresponds to the remaining degree of freedom identified in Section 3.4. Because the preserved within-task functions do not determine how the common scores of different blocks relate to each other, we adjudicate them after completing training on each task t.

To do so, for each learned block $k \le t$, we replace the learned, potentially input-dependent common score with one input-independent scalar offset β_k:

$$\boldsymbol{z}_k^{\mathrm{eval}}(x) = \widetilde{\boldsymbol{z}}_k(x) + \beta_k\mathbf{1}, \tag{11}$$

where $\widetilde{\boldsymbol{z}}_k(x) = P_k\boldsymbol{z}_k(x)$. We fit $\boldsymbol{\beta}$ on the class-balanced memory $\mathcal{M}_t$ while holding the encoder and function-defining classifier fixed, and retain the reconciled classifier as the initialization for the next task. Because each β_k is block-common, this reconciliation changes only cross-block competition while leaving all within-block pairwise margins and conditional class probabilities unchanged. See Appendix C for the derivation and supporting controls.

5 Experiments

5.1 Experimental settings

Benchmark and training protocol. Using Mammoth (Buzzega et al., 2020; Boschini et al., 2022), we evaluate CIFAR-100 (Krizhevsky, 2009) and TinyImageNet (Le & Yang, 2015), split into ten tasks of 10 and 20 classes, respectively. All methods use a ResNet-18 (He et al., 2016) trained from scratch with exemplar budget $M = \{200, 500\}$. We report final task-averaged Class-IL accuracy as

Table 2: Final average accuracy (%) with $M = 500$, reported as mean $\pm$ one sample standard deviation (1σ) over five seeds. **Bold** marks the best sequential result within each teacher-network storage group. Joint is an offline reference.

Additional full teacher network	Method	Full-way training	CIFAR-100		TinyImageNet	
			Task-IL	Class-IL	Task-IL	Class-IL
–	Joint	✓	92.42±0.17	73.14±0.29	81.94±0.21	59.50±0.08
No	SGD	✓	36.66±2.72	9.36±0.06	18.58±1.57	8.13±0.05
	LwF†	✓	26.88±0.54	9.80±0.09	14.89±0.53	8.04±0.71
	ER	✓	74.50±0.57	21.43±0.72	48.80±0.63	10.26±0.28
	DER	✓	74.38±1.20	34.83±1.17	52.80±0.82	17.26±0.53
	DER++	✓	74.80±0.52	36.63±1.20	51.55±0.42	18.95±1.40
	ER-ACE	×	71.03±0.37	34.30±0.61	39.18±0.80	13.54±0.59
	LODE(DER++)	△	77.67±0.34	41.90±0.86	58.76±0.42	17.61±0.73
	FEAST (ours)	✓	**83.77±0.49**	**44.69±1.24**	**66.72±0.48**	**25.93±0.77**
Yes	BiC	✓	**85.66±0.33**	36.63±1.66	68.95±4.92	14.82±1.03
	WA	✓	80.93±0.53	36.24±0.60	63.85±4.32	19.27±2.15
	LUCIR	✓	85.55±0.34	37.63±0.74	68.43±0.34	**34.41±0.17**
	SS-IL	×	85.58±0.10	**44.50±0.29**	**73.37±0.60**	33.73±0.47

Full-way training: ✓ uses joint CE normalization across old and new classes, × masks or separates their competition, and △ denotes LODE's reweighted loss decoupling. † Mammoth's LwF caches teacher outputs rather than retaining a separate full teacher network. Joint, SGD, and LwF do not use exemplar memory.

the primary metric, together with Task-IL, forgetting, and backward transfer. Sequential methods use SGD with batch size 32 for 50/100 epochs per task on CIFAR-100/TinyImageNet. We use Mammoth implementations under a shared dataset-specific schedule while preserving each method's native objective and classifier design (see Appendices D and E).

Comparison models. Joint and sequential SGD provide offline upper and sequential lower references. We then compare FEAST with representative methods spanning three aspects most relevant to our setting: canonical continual-learning baselines (ER, DER/DER++, and LwF) (Chaudhry et al., 2019; Buzzega et al., 2020; Li & Hoiem, 2016); methods that directly modify old–new class competition (ER-ACE, LODE(DER++), and SS-IL) (Caccia et al., 2022; Liang & Li, 2023; Ahn et al., 2021); and methods that modify or rebalance the classifier (BiC, WA, and LUCIR) (Wu et al., 2019; Zhao et al., 2020; Hou et al., 2019). For a fairer comparison of memory use, Table 2 groups methods by whether they retain an additional full teacher network. (See Appendix A.)

5.2 Class-IL and Task-IL Performance

Table 2 reports final Task-IL and Class-IL accuracy at $M = 500$. Among methods without an additional full teacher network, FEAST yields strong performance across both datasets. Methods such as LUCIR and SS-IL achieve comparable or higher performance on some metrics, but with substantially greater retained tensor memory (see Fig. F.3 and Fig. F.4). FEAST remains strong at $M = 200$ (see Appendix F.1). Additional trajectories and retention metrics are reported in Appendix F.

5.3 Complementary Allocation of Stability and Adaptation

We next provide a mechanistic account of how FEAST's gains arise. The analyses in Sections 5.3 and 5.4 use CIFAR-100 with $M = 500$. To isolate the roles of FEA and FCA, we first use a matched 2×2 intervention while keeping the function-defining component of each previous classifier fixed across all conditions.

FEA improves functional stability. FEA shifts representational change away from directions visible to the reference contrast map rather than uniformly suppressing change. Across both FCA conditions, it reduces the classifier-visible fraction of representational-change energy by 27.76 percentage points (95% CI $[27.30, 28.21]$; Fig. 2a).

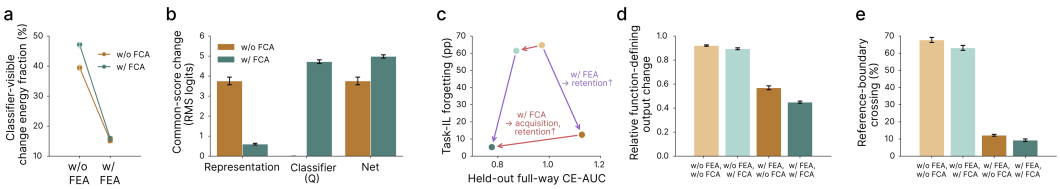


Figure 2: **Complementary allocation of stability and adaptation.** Matched FEA×FCA intervention over five seeds. **(a)** FEA reduces classifier-visible representational change. **(b)** With FEA enabled, FCA reallocates common-score adaptation from the encoder to the classifier. **(c)** Acquisition–retention plane across the four conditions. **(d–e)** Function-defining output change and reference-boundary crossing across conditions. Error bars denote 95% confidence intervals. Metric definitions are provided in Appendix H.1–H.3.

FCA reallocates adaptation. The resulting common-score adaptation can be carried either through representation changes induced by the encoder or through functionally invariant classifier coordinates. With FEA, FCA reduces the encoder-side common-score contribution by 3.16 RMS logits (95% CI $[2.99, 3.32]$) while increasing the classifier-side contribution by 4.72 RMS logits (95% CI $[4.63, 4.81]$; Fig. 2b). The net common-score change also increases by 1.23 RMS logits (95% CI $[1.07, 1.39]$), showing that this reflects a reallocation rather than a reduction in common-score adaptation.

This reallocation benefits both acquisition and retention. With FEA, FCA reduces held-out full-way CE-AUC by 0.35 (95% CI $[0.32, 0.38]$; Fig. 2c), relative function-defining output change by 0.12 (95% CI $[0.11, 0.13]$; Fig. 2d), and Task-IL forgetting by 7.22 percentage points (95% CI $[5.90, 8.53]$; Fig. 2c). The FEA–FCA interaction is significant for both acquisition and forgetting, supporting their complementary roles. Together, FEA determines where stability is imposed, whereas FCA provides an alternative path for adaptation. Their combination separates the target of preservation from the path of adaptation, framing the stability–plasticity dilemma as a problem of allocating change according to its functional consequence.

From function-defining preservation to decision retention. We next examine whether this allocation translates into preservation of previously learned decisions. Across conditions, lower classifier-visible representational change is accompanied by smaller function-defining output change (Fig. 2d). As established in Section 3.3, function-defining output changes directly perturb pairwise decision margins. Consistent with this link, smaller output change is accompanied by less reference-margin erosion; with FEA, FCA reduces margin erosion by 1.38 logits (95% CI $[1.10, 1.66]$). These differences are reflected at the decision level in fewer crossings of reference decision boundaries (Fig. 2e), accompanied by lower Task-IL forgetting (Fig. 2c).

5.4 Comparison with Logit Replay

We next compare our preservation mechanism with DER (Buzzega et al., 2020), a canonical logit-replay method that constrains previous outputs while allowing the encoder and classifier to co-adapt. Accordingly, all functional comparisons use each method's current encoder and classifier. Figure 3a visualizes the evolution of the within-task function in centered-logit space, using a separate PCA basis for each method, fit to Task-1 centered logits at acquisition and fixed thereafter. The trajectories reveal qualitatively different evolution of the learned class geometry, but do not identify how representation and classifier changes contribute to it. We therefore decompose function-defining output change into representation- and classifier-side contributions. DER exhibits negative alignment between the two contributions (cosine similarity $= -0.27$, 95% CI $[-0.29, -0.24]$), indicating partial compensation, whereas our function-defining classifier component remains fixed by construction (Fig. 3b).

Despite this co-adaptation, our relative function-defining output change is lower than DER by 0.17 on average (95% CI $[0.16, 0.18]$; Fig. 3d), with the gap widening over subsequent tasks. Among examples correctly classified at acquisition, final within-task decision failure is lower by 12.22 percent-

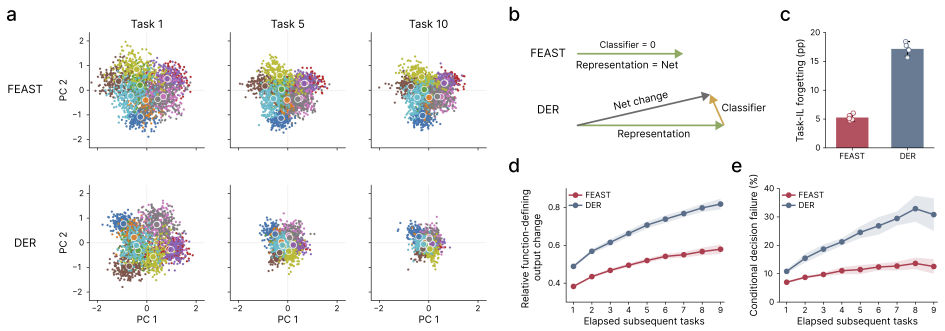


Figure 3: **Comparison with logit replay. (a)** Task-1 centered-logit geometry for FEAST and DER, using method-specific acquisition-time PCA bases fixed across checkpoints. **(b)** Representation- and classifier-side contributions to function-defining output change; FEAST fixes the function-defining classifier, whereas DER permits classifier–representation co-adaptation. **(c)** Task-IL forgetting. **(d–e)** Relative function-defining output change and within-task decision failure over subsequent tasks. Error bars and shaded regions show 95% confidence intervals. Metric definitions are provided in Appendix H.4.

age points (95% CI $[10.31, 14.13]$; Fig. 3e). Task-IL forgetting is likewise lower by 11.91 percentage points (95% CI $[10.51, 13.31]$; Fig. 3c). Thus, although DER partially compensates representation-side change through classifier–representation co-adaptation, our method exhibits smaller function-defining output change and fewer within-block decision failures as learning proceeds.

6 Discussion

We formulate the stability–plasticity dilemma in terms of functional equivalence, providing a function-relative criterion for what must remain stable and what can remain available for adaptation. Functionally invariant coordinates are therefore not necessarily redundant: changes that are immaterial to one function can remain useful for another. In FEAST, common-score changes leave the within-task function unchanged while remaining consequential for full-way competition. This separates the target of preservation from the path of adaptation: stability is concentrated on function-defining coordinates, while part of the required adaptation can be carried through degrees of freedom that the preserved function does not determine. Our experiments support this account, showing that adaptation can be redirected toward functionally invariant classifier coordinates rather than uniformly suppressed. Thus, plasticity compatible with previously learned functions can support retention rather than merely compete with it.

The exact functional equivalence exploited by FEAST should be distinguished from exact retention of the learned input–output function. FEAST preserves the reference contrast map exactly, but compatibility of an evolving encoder with that map is encouraged only through a soft loss on finite exemplars, so preservation is not guaranteed on all previous inputs. The current implementation also assumes known task blocks and restricts cross-block reconciliation to one input-independent offset per block. Extending the approach beyond these assumptions, particularly to settings where functional equivalence is approximate or must itself be learned, remains an open direction.

More broadly, when multiple states realize the same function, preservation need not require matching a particular internal or output state. Objectives that do so may constrain change more tightly than functional preservation requires. Functional equivalence instead suggests defining stability modulo the invariances of the quantity being preserved, so that only function-changing deviations are constrained, while the remaining freedom can be retained for learning. Our work provides one exact instance of this broader design principle.

Reproducibility statement

We provide executable supplementary code with final configurations for all benchmark methods, pinned source revisions and dependencies, dataset integrity checks, and automated tests. The code supports training, checkpoint re-evaluation, and numerical mechanism analyses, while recording configurations, seeds, and source hashes. Experimental protocols and final settings are specified in Appendix D, complete FEAST procedures in Appendix E and Algorithm 1, and analysis metrics and aggregation in Appendix H.

AI use statement

In this work, we used generative AI tools for refining mathematical formulations, code implementation, result analysis, literature search, and manuscript polishing. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

References

Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. SS-IL: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 844–853, 2021.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. doi: 10.1007/978-3-030-01219-9_9.

Daniel Anthes, Sushrut Thorat, Peter König, and Tim C. Kietzmann. Keep moving: Identifying task-relevant subspaces to maximise plasticity for newly learned tasks. In *Proceedings of The 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 815–831. PMLR, 2025.

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

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15920–15930. Curran Associates, Inc., 2020.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=N8MaByOzUfb.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *ICML Workshop on Multi-Task and Lifelong Reinforcement Learning*, 2019.

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. doi: 10.1038/s41586-024-07711-7.

Laura N. Driscoll, Noah L. Pettit, Matthias Minderer, Selmaan N. Chettih, and Christopher D. Harvey. Dynamic reorganization of neuronal activity patterns in parietal cortex. *Cell*, 170(5):986–999.e16, 2017. doi: 10.1016/j.cell.2017.07.021.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773. PMLR, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019. doi: 10.1109/CVPR.2019.00092.

Matthew T. Kaufman, Mark M. Churchland, Stephen I. Ryu, and Krishna V. Shenoy. Cortical activity in the null space: permitting preparation without movement. *Nature Neuroscience*, 17: 440–448, 2014. doi: 10.1038/nn.3643.

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, and Bing Liu. A theoretical study on solving continual learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5065–5079, 2022. doi: 10.52202/068431-0366.

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. doi: 10.1073/pnas.1611835114.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. Cs231n project report, Stanford University, 2015. URL https://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pp. 614–629, 2016. doi: 10.1007/978-3-319-46493-0_37.

Yan-Shuo Liang and Wu-Jun Li. Loss decoupling for task-agnostic continual learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11151–11167, 2023. doi: 10.52202/075280-0492.

Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

Emily A. Mankin, Fraser T. Sparks, Begum Slayyeh, Robert J. Sutherland, Stefan Leutgeb, and Jill K. Leutgeb. Neuronal code for extended time in the hippocampus. *Proceedings of the National Academy of Sciences*, 109(47):19462–19467, 2012. doi: 10.1073/pnas.1214107109.

Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: Survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2023. doi: 10.1109/TPAMI.2022.3213473.

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. doi: 10.1016/S0079-7421(08)60536-8.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4:504, 2013. doi: 10.3389/fpsyg.2013.00504.

Martin Mundt, Steven Lang, Quentin Delfosse, and Kristian Kersting. CLEVA-Compass: A Continual Learning evaluation assessment compass to promote research transparency and comparability. In *International Conference on Learning Representations*, 2022.

Michael E. Rule, Adrianna R. Loback, Dhruva V. Raman, Laura N. Driscoll, Christopher D. Harvey, and Timothy O'Leary. Stable task information from an unstable neural population. *eLife*, 9: e51121, 2020. doi: 10.7554/eLife.51121.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=ZK4Cvv5Eci.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. doi: 10.1038/s42256-022-00568-3.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019. doi: 10.1109/CVPR.2019.00046.

Zheng Zhai, Jiali Zhang, Haiyu Wang, Mingxin Wu, Keshun Yang, Xiaoyan Qiao, and Qiang Sun. Rethinking softmax in incremental learning. *Neural Networks*, 193:108017, 2026. doi: 10.1016/j.neunet.2025.108017.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13208–13217, 2020. doi: 10.1109/CVPR42600.2020.01322.

Yaniv Ziv, Laurie D. Burns, Eric D. Cocker, Elizabeth O. Hamel, Kunal K. Ghosh, Lacey J. Kitch, Abbas El Gamal, and Mark J. Schnitzer. Long-term dynamics of CA1 hippocampal place codes. *Nature Neuroscience*, 16(3):264–266, 2013. doi: 10.1038/nn.3329.

APPENDIX

A EXTENDED RELATED WORK

Output preservation and logit matching. Knowledge distillation preserves a teacher's predictive behavior by matching softened output distributions (Hinton et al., 2015), and related output-preservation objectives are widely used in continual learning (Li & Hoiem, 2016; Buzzega et al., 2020). The centered form of such an objective is not entirely new. Hinton et al. (2015) showed that, under a zero-mean logit convention, high-temperature knowledge distillation becomes equivalent, up to scaling, to squared teacher–student logit matching. The centered target-matching objective used in FEA is closely related to this result, but centering serves a different purpose here: it follows from the softmax equivalence being preserved, rather than being introduced as an alternative distillation objective. More recently, Zhai et al. (2026) analyze softmax shift invariance in incremental learning as a source of non-identifiability and task-wise imbalance, and mitigate it using imbalance-invariant distillation or shift-sensitive objectives. FEAST takes the complementary view: the block-common shift is retained as functionally invariant freedom for plasticity during learning, while only the residual cross-block ambiguity is subsequently reconciled.

DER is particularly relevant as the logit-replay baseline underlying our comparison (Buzzega et al., 2020). It stores and matches raw logits, and therefore penalizes common logit shifts together with changes in relative class scores. Such shifts do not affect the corresponding softmax distribution; within a task block, they likewise leave all class contrasts unchanged. FEAST instead preserves only the relative-score component for the specified within-task function, leaving the block-common component outside that preservation constraint. The empirical consequence of this distinction is examined directly by the centered-versus-uncentered control in Appendix G.

Selective stability and functionally silent change. Selective stabilization has also been explored in parameter, gradient, and representation spaces. OGD and GPM restrict updates along subspaces associated with previously learned tasks (Farajtabar et al., 2020; Saha et al., 2021). More closely related to our motivation, Anthes et al. (2025) decompose representational change into components visible and invisible to a task readout, showing that restricting readout-null change can unnecessarily reduce plasticity. Their analysis is formulated around task-specific readouts in a Task-IL setting, where the previously learned readout provides the reference relative to which null and visible representational change are defined. FEAST instead considers a shared Class-IL classifier and asks how an exact invariance of its preserved conditional function can be used directly to organize continual learning. The two perspectives therefore share the motivation of selective stability, while differing in both the setting and the object from which the protected freedom is defined.

Teacher-network storage and comparison scope. A central difficulty in Class-IL arises from competition between previously learned and newly introduced classes, which can produce substantial old–new interference (Kim et al., 2022). A common strategy for stabilizing previous predictions is to retain the preceding full network and use it as a teacher for distillation. Several methods in our comparison, including BiC, WA, LUCIR, and SS-IL, follow this general design. Such a teacher provides persistent information about the previous model beyond the exemplar buffer, but also incurs the memory cost of retaining an additional network. Because memory consumption is an important dimension of continual-learning evaluation and comparability (Mundt et al., 2022), equal exemplar budgets alone do not constitute equal-memory conditions. We therefore distinguish methods according to whether they retain an additional full teacher network in Table 2, and make comparisons within these storage groups explicit.

Separated objectives and old–new competition. Several Class-IL methods intervene directly on competition between old and new classes. SS-IL attributes part of the prediction bias to joint normalization over old and new classes, and combines separated softmax normalization with task-wise knowledge distillation (Ahn et al., 2021). ER-ACE restricts the logits participating in the current-example classification objective (Caccia et al., 2022), while LODE separates within-new-class discrimination from old–new discrimination and weights the two objectives differently (Liang & Li, 2023). SS-IL is especially close in exploiting a block-wise softmax structure, but it reduces cross-task interaction by separating the competition itself. FEAST instead retains full-way learning

and constrains the response of previously learned classifier blocks while that competition remains present.

Classifier correction and cross-block reconciliation. Classifier-side correction provides another route to addressing old–new imbalance. WA rescales classifier weights (Zhao et al., 2020), and LUCIR uses a cosine-normalized classifier together with distillation and margin-based separation (Hou et al., 2019). BiC is most directly related to the calibration stage of FEAST: it learns a two-parameter affine transformation $q_c = \alpha o_c + \beta$ for newly introduced classes from balanced validation data (Wu et al., 2019). In FEAST, reconciliation is restricted to one additive offset per task block after removing the learned block-common component. It therefore adjusts cross-block score alignment without changing within-block logit differences. This reconciliation is the remaining alignment problem after the within-block function has been specified, rather than a general affine correction of the learned classifier.

Relation among these approaches. These works provide precedents for several aspects that also appear in FEAST: output preservation, selective stabilization, modification of old–new competition, and classifier calibration. The corresponding components of FEAST, however, are not introduced independently to reproduce these mechanisms. They follow from the same functional-equivalence construction: the specified function determines the output coordinates to preserve, the degrees of freedom that can remain available for adaptation, and the residual cross-block freedom that must subsequently be reconciled.

B Mathematical Details

B.1 Notation and exact softmax equivalence

For task block k with class set $\mathcal{C}_k$ and $K_k = |\mathcal{C}_k|$, define

$$P_k = I_{K_k} - \frac{1}{K_k}\mathbf{1}\mathbf{1}^\top, \qquad Q_k = \frac{1}{K_k}\mathbf{1}\mathbf{1}^\top. \tag{12}$$

Both matrices are symmetric idempotent projectors, $P_k Q_k = 0$, and $P_k + Q_k = I_{K_k}$. For logits $\mathbf{z}_k$, let

$$\widetilde{\mathbf{z}}_k = P_k \mathbf{z}_k, \qquad a_k = \frac{1}{K_k}\mathbf{1}^\top \mathbf{z}_k, \qquad \mathbf{z}_k = \widetilde{\mathbf{z}}_k + a_k \mathbf{1}. \tag{13}$$

Thus P_k retains all within-block class contrasts and Q_k retains the block-common coordinate.

Softmax probabilities are invariant exactly to additive shifts along the all-ones direction. Specifically, for $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^{K_k}$,

$$\mathrm{softmax}(\mathbf{z}) = \mathrm{softmax}(\mathbf{z}') \quad \Longleftrightarrow \quad P_k \mathbf{z} = P_k \mathbf{z}'. \tag{14}$$

To see this, if $P_k\mathbf{z} = P_k\mathbf{z}'$, then $\mathbf{z}' - \mathbf{z} = \alpha\mathbf{1}$ for some scalar α, and softmax is invariant to this shift. Conversely, equality of the softmax vectors implies $z_c - z_{c'} = z'_c - z'_{c'}$ for every pair (c, c'), so $\mathbf{z}' - \mathbf{z}$ is constant across coordinates and lies in $\mathrm{span}\{\mathbf{1}\}$.

For a linear classifier $\mathbf{z}_k = W_k\mathbf{h} + \mathbf{b}_k$, write $\widetilde{W}_k = P_k W_k$ and $\widetilde{\mathbf{b}}_k = P_k \mathbf{b}_k$. The conditional classifier is determined by $(\widetilde{W}_k, \widetilde{\mathbf{b}}_k)$, whereas $(Q_k W_k, Q_k \mathbf{b}_k)$ may change without altering the within-block conditional distribution at a fixed representation.

B.2 Softmax margins and a retention bound

For distinct classes c, c' in a block, define

$$m_{cc'}(\mathbf{z}) := z_c - z_{c'} = (P_k\mathbf{z})_c - (P_k\mathbf{z})_{c'}. \tag{15}$$

For any two logit states, write

$$\boldsymbol{\delta} := P_k(\mathbf{z}' - \mathbf{z}). \tag{16}$$

If $\mathbf{e}_c$ denotes the cth standard basis vector, then

$$\Delta m_{cc'} = (\mathbf{e}_c - \mathbf{e}_{c'})^\top \boldsymbol{\delta}, \qquad |\Delta m_{cc'}| \le \sqrt{2}\|\boldsymbol{\delta}\|_2. \tag{17}$$

The aggregate identity used in the main text follows from

$$\sum_{c<c'} (\delta_c - \delta_{c'})^2 = \frac{1}{2}\sum_{c,c'} (\delta_c - \delta_{c'})^2 \tag{18}$$

$$= K_k \sum_c \delta_c^2 - \left(\sum_c \delta_c\right)^2 \tag{19}$$

$$= K_k \|\boldsymbol{\delta}\|_2^2, \tag{20}$$

where the last equality uses $\mathbf{1}^\top\boldsymbol{\delta} = 0$. Consequently, the per-anchor class-mean FEA loss $\|\boldsymbol{\delta}\|_2^2/K_k$ equals

$$K_k^{-2} \sum_{c<c'} (\Delta m_{cc'})^2. \tag{21}$$

This is an exact relation between score changes, not a claim that logit margin is a feature-space distance.

For the softmax case, let the reference prediction be y and define its minimum reference logit margin as

$$m^{\mathrm{ref}}_{\min}(x) = \min_{c \ne y} \left(z^{\mathrm{ref}}_y(x) - z^{\mathrm{ref}}_c(x)\right). \tag{22}$$

Let

$$\boldsymbol{\delta}(x) = P_k\left(\mathbf{z}^{\mathrm{new}}(x) - \mathbf{z}^{\mathrm{ref}}(x)\right). \tag{23}$$

For every competitor $c \neq y$, Eq. 17 gives

$$m_{yc}^{\text{new}}(x) \geq m_{yc}^{\text{ref}}(x) - \sqrt{2}\|\boldsymbol{\delta}(x)\|_2. \tag{24}$$

Hence a sufficient condition for preserving the reference prediction is

$$m_{\min}^{\text{ref}}(x) > \sqrt{2}\|\boldsymbol{\delta}(x)\|_2. \tag{25}$$

Under this condition, every reference-positive pairwise margin remains strictly positive, so the predicted class cannot change.

C Cross-Task Reconciliation

C.1 Preservation of conditional evidence

For block k, define the centered log-evidence

$$e_k(x) := \log \sum_{c \in \mathcal{C}_k} \exp \widetilde{z}_{k,c}(x). \tag{26}$$

For $c \in \mathcal{C}_k$, the deployed logits are

$$z^{\text{eval}}_{k,c}(x) = \widetilde{z}_{k,c}(x) + \beta_k. \tag{27}$$

Therefore, for any $c, c' \in \mathcal{C}_k$,

$$z^{\text{eval}}_{k,c} - z^{\text{eval}}_{k,c'} = \widetilde{z}_{k,c} - \widetilde{z}_{k,c'}, \tag{28}$$

and

$$p(y = c \mid x, k; \boldsymbol{\beta}) = p(y = c \mid x, k). \tag{29}$$

Calibration therefore preserves all within-block conditional evidence. The input-dependent centered block evidence $e_k(x) = \log \sum_{c \in \mathcal{C}_k} \exp \widetilde{z}_{k,c}(x)$ is also unchanged; calibration adds only an input-independent offset β_k to the block log-evidence. The resulting block probability is

$$\pi_k(x; \boldsymbol{\beta}) = \frac{\exp(e_k(x) + \beta_k)}{\sum_{j=1}^{t} \exp(e_j(x) + \beta_j)}. \tag{30}$$

C.2 Class/block likelihood factorization

For $y = c \in \mathcal{C}_k$,

$$p(y = c \mid x; \boldsymbol{\beta}) = p(y = c \mid x, k)\, p(k \mid x; \boldsymbol{\beta}), \tag{31}$$

so

$$\begin{aligned} -\log p(y = c \mid x; \boldsymbol{\beta}) = &-\log p(y = c \mid x, k) \\ &-\log p(k \mid x; \boldsymbol{\beta}). \end{aligned} \tag{32}$$

Averaging over the data used to fit $\boldsymbol{\beta}$ separates a term independent of $\boldsymbol{\beta}$, and therefore

$$\arg\min_{\boldsymbol{\beta}} \mathcal{L}_{\text{class}}(\boldsymbol{\beta}) = \arg\min_{\boldsymbol{\beta}} \mathcal{L}_{\text{block}}(\boldsymbol{\beta}) \tag{33}$$

within the scalar-offset family.

C.3 Folded-head equivalence

The deployed head is obtained by

$$W^{\text{eval}}_k = P_k W_k, \qquad \boldsymbol{b}^{\text{eval}}_k = P_k \boldsymbol{b}_k + \beta_k \mathbf{1}. \tag{34}$$

This removes the input-dependent common weights as well as replacing the common bias. Modifying only the bias of an otherwise raw head would not implement Eq. 11. No separate inference module is required.

C.4 Reconciliation data and fitting procedure

Offsets are fit on the class-balanced memory $\mathcal{M}_t$ after its boundary rebuild, with the encoder and centered classifier fixed. The objective uses log-sum-exp block evidence and is minimized using L-BFGS (Liu & Nocedal, 1989), with no ridge regularization and unit offset scale. After unconstrained fitting, the offset mean is subtracted to impose $\mathbf{1}^\top \boldsymbol{\beta} = 0$; shift invariance makes this equivalent to fixing that gauge during optimization. No separate held-out fitting split is used, and test data are not used. Fitting on memory supports the empirical calibration objective, not a guarantee of held-out calibration accuracy.

Offsets start at zero. L-BFGS uses learning rate 1, at most 100 iterations, and strong-Wolfe line search. Features use unaugmented, normalized images in evaluation mode (Appendix E.2); the released configuration records the remaining solver defaults.

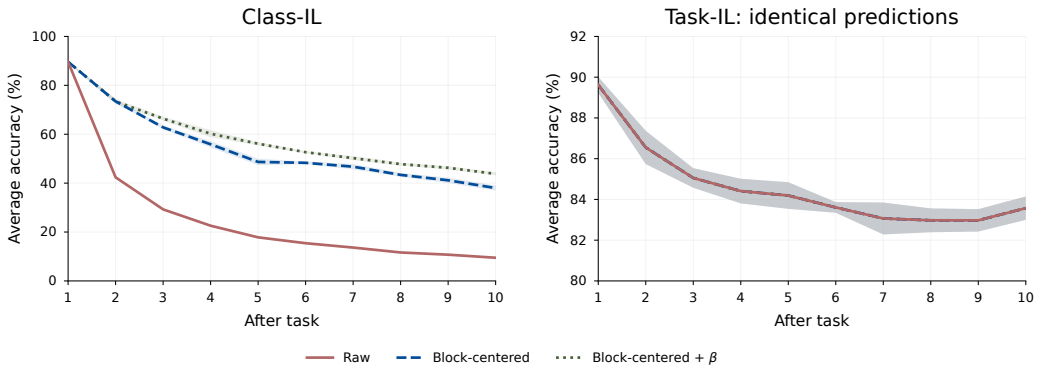


Figure C.1: Fixed-checkpoint inference-state decomposition on CIFAR-100 ($M = 500$). Three readouts are evaluated at every boundary of the same ablation runs, without retraining. Lines and bands show the mean and ± 1 sample standard deviation over five seeds. Blockwise logit centering alone recovers much of the Class-IL improvement; fitted offsets further improve cross-block competition. Task-IL predictions are identical for all three readouts, so the right-panel curves overlap.

C.5 Fixed-checkpoint inference-state decomposition

We evaluate three readouts of each fixed checkpoint: raw logits, block-centered logits $P_k \boldsymbol{z}_k(x)$, and block-centered logits plus fitted β_k. The encoder and class contrasts are shared across readouts; only block-common scores change. We use the five-seed CIFAR-100 $M = 500$ ablation control described in Appendix E.3.

Reconciliation without changing within-block decisions. At the final boundary, raw Class-IL accuracy is $9.48 \pm 0.08\%$. Simply subtracting each input's class mean within each block, with no fitted parameters, raises it to $37.99 \pm 1.09\%$; fitting β then raises it to $43.78 \pm 0.61\%$. The paired gains are 28.51 ± 1.15 and 5.79 ± 0.87 percentage points, respectively. Thus, blockwise logit centering already accounts for most of the endpoint improvement, although the additional gain from fitting is appreciable. All three readouts give $83.57 \pm 0.58\%$ Task-IL accuracy, with identical predictions across every seed and boundary.

These results support viewing boundary reconciliation as operating on the block-common freedom used for plasticity during learning. Blockwise logit centering already removes much of the resulting cross-block misalignment; fitted scalar offsets refine the remaining competition without changing any within-block logit contrast or conditional probability. The analysis isolates readout effects at fixed checkpoints and does not measure the effect of omitting boundary reconciliation throughout training.

C.6 Transferring boundary reconciliation to DER

Control design. We next ask whether FEAST's gains can be reproduced by transferring its boundary calibration procedure to a standard replay method. On CIFAR-100 ($M = 500$), we compare DER with two persistent boundary corrections: blockwise centering alone and blockwise centering followed by fitted scalar offsets β_k. Both variants otherwise retain DER's original raw-logit replay objective ($\alpha = 0.3$), reservoir sampling, and training configuration (learning rate 0.03), with no hyperparameter retuning or modification of the stored logit targets. At each task boundary, the seen classifier blocks are centered and, for the offset variant, β_k is fitted using the same objective and solver as in FEAST on DER's own reservoir; the resulting head is then retained for subsequent learning.

This control deliberately preserves DER's raw-logit objective rather than reformulating DER in FEAST's functional coordinates. The distinction is important: FEAST constrains only centered logits, so a block-common shift introduced by calibration leaves its functional-consistency objective

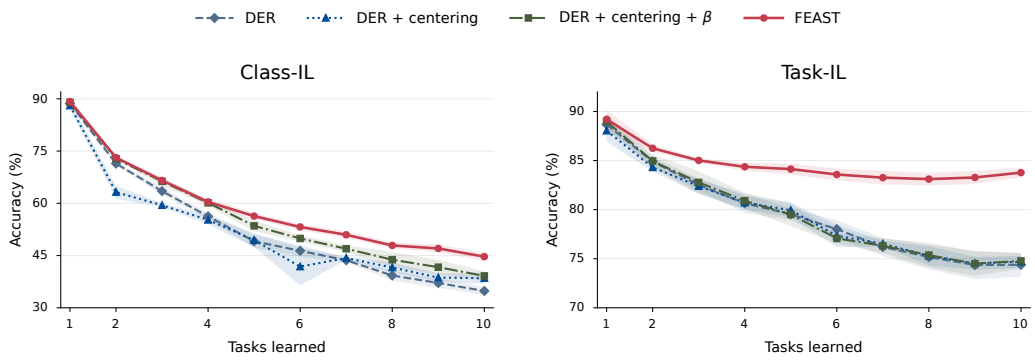


Figure C.2: Transferring persistent boundary calibration to DER on CIFAR-100 ($M = 500$). Class-IL (left) and Task-IL (right) accuracy after each task for four separately trained conditions. Lines and shaded bands denote the mean and ± 1 sample standard deviation over five seeds.

unchanged. In DER, by contrast, raw-logit replay explicitly preserves the common component that calibration modifies. The experiment therefore tests whether boundary calibration acts as a generic plug-in improvement, or whether its effectiveness depends on the functional decomposition with which it is paired. FEAST is evaluated using its main-benchmark configuration; in particular, its class-balanced calibration memory differs from DER's reservoir.

Boundary calibration transfers only partially to DER. Calibration improves DER's global classification, but does not recover FEAST's performance. Final Class-IL accuracy increases from $34.83 \pm 1.17\%$ for standard DER to $38.48 \pm 1.63\%$ with persistent centering and $39.16 \pm 1.35\%$ with persistent centering plus fitted offsets, whereas FEAST reaches $44.69 \pm 1.24\%$ (Figure C.2). The corresponding Task-IL accuracy changes only modestly, from $74.38 \pm 1.20\%$ for DER to $74.78 \pm 0.79\%$ with fitted offsets, while FEAST reaches $83.77 \pm 0.49\%$. The remaining gap is therefore not confined to cross-block score alignment: FEAST also retains substantially stronger within-task discrimination, particularly on previously learned tasks.

These results indicate that boundary calibration can also benefit DER, but its contribution alone does not account for FEAST's advantage. Importantly, the same correction interacts differently with the two preservation objectives. In DER, changes along the block-common direction remain visible to raw-logit matching, so persisting a calibrated head introduces deviations from the logits that replay explicitly seeks to preserve. The limited transfer is therefore consistent with a conflict between calibration and DER's original preservation objective. In FEAST, by contrast, the functional decomposition removes this conflict: calibration acts only along coordinates excluded from centered functional consistency, allowing cross-block competition to be adjusted without altering the quantity being preserved. This highlights the role of functional equivalence not only in determining what should remain stable, but also in identifying degrees of freedom that can be safely repurposed for adaptation.

Table D.1: Benchmark construction and data splits. Selection reserves 10% per training class using a fixed split seed of 0. Final runs use the full training split.

	CIFAR-100	TinyImageNet
Tasks × classes per task	10×10	10×20
Image size	32×32	64×64
Training / final evaluation examples	$50{,}000/10{,}000$	$100{,}000/10{,}000$
Final evaluation split	official test	official validation
Selection train / holdout per class	$450/50$	$450/50$
Final training examples per class	500	500

Table D.2: Main-stage optimization. Baseline auxiliary stages follow their pinned implementations. FEAST-specific updates are defined in Appendix E.

Setting	CIFAR-100	TinyImageNet
Optimizer†	SGD	SGD
Epochs per sequential task	50	100
Batch size / explicit replay draw	$32/32$	$32/32$
Learning-rate schedule	$\times 0.2$ after epochs 35, 45	constant
Default momentum / weight decay	$0/0$	$0/0$
Nesterov / dropped last batch	off / no	off / no

D Experimental Protocol and Reproducibility

D.1 Benchmark construction

We use Mammoth's `seq-cifar100` and `seq-tinyimg` benchmarks (Buzzega et al., 2020). Table D.1 summarizes the benchmark construction. Tasks contain contiguous class blocks in the processed dataset's identity label order, shared across methods and seeds. Task boundaries are available during training. TinyImageNet's labeled official validation set serves as its final test set; it is separate from the training-derived holdout used for hyperparameter selection.

D.2 Shared training protocol

All methods use Mammoth's ResNet-18 backbone, trained from scratch, with native classifier heads and objectives. We retain the framework's training transforms (random crop, horizontal flip, and dataset normalization), deterministic evaluation transforms, and method-specific BatchNorm and replay procedures. Evaluation uses the network in evaluation mode. Table D.2 records the shared schedule and explicit optimizer settings († refers to the exceptions below). For TinyImageNet, we use 100 epochs per task, as in the original DER benchmark (Buzzega et al., 2020).

Optimizer exceptions (†). Method-specific settings follow Mammoth for native baselines and the reference implementations for ports. LwF uses weight decay 5×10^{-4}; LUCIR uses momentum 0.9 and zero weight decay. WA and SS-IL also use momentum 0.9, with weight decay selected in the hyperparameter tables for WA and fixed at 10^{-4} for SS-IL. Learning rates and optimizers are reset at task boundaries following Mammoth.

Joint reference. Joint uses Mammoth's SGD learner with joint training enabled. It pools all training tasks and optimizes a single all-class objective for 50 epochs on CIFAR-100 or 100 epochs on TinyImageNet, using the corresponding schedule above. It is an offline upper reference with access to all training data; its epochs are not repeated once per task.

D.3 Comparison implementations and memory

Implementation sources. Joint, SGD fine-tuning, ER, DER, DER++, ER-ACE, LwF, BiC, and LUCIR use Mammoth commit `e75a491c69fd729edeb01431afb753d9157d9a81`.

LODE(DER++) ports the authors' released `models/lode_derpp.py` (commit `10c72cd`). SS-IL ports the loss and prefix-memory rule from the authors' official implementation (`8c32381`). WA follows PyCIL's WA implementation (`f3509b8`), with Mammoth's normalized herding. The source lock records full commits and local overlay hashes. Unmodified settings, including auxiliary fitting and correction stages, follow these pinned implementations.

Additional full teacher network. Table 2 distinguishes whether training retains an additional full teacher network. BiC, WA, LUCIR, and SS-IL do so; Mammoth's LwF instead caches teacher outputs on current-task inputs. DER-based methods retain historical logits, and FEAST retains refreshed own-block targets. The grouping therefore distinguishes network storage, not the use of teacher-derived information in general.

Exemplar memory and storage conventions. M is the total exemplar capacity across tasks. ER, DER/DER++, ER-ACE, and LODE(DER++) use native reservoir sampling; BiC and LUCIR retain their native buffer rules. Class-balanced selection is used by several methods: WA uses normalized herding, SS-IL uses the authors' prefix rule, and FEAST uses random sampling within class quotas. Appendix E.2 details FEAST's quota allocation and target refresh. SGD and LwF store no past exemplars; Joint retains access to the full training set and is outside the replay budget. We record actual occupancy as well as capacity, since native selection can leave slots unused. Native replay procedures and auxiliary stages are retained. Retained-memory comparisons use a common FP32 image-storage convention and exclude optimizer state for all methods (Appendix F.4).

D.4 Hyperparameter selection and final configurations

Configuration reuse and adaptation. We reuse Mammoth's registered best hyperparameters when they apply to the dataset and memory setting. When adaptation is needed, we search a small grid around a public or previously validated configuration. Methods absent from Mammoth follow the same policy, using a reference or development anchor and at most nine candidates per method–dataset search. This retains established settings while bounding the tuning budget for adapted methods. If multiple search parameters produce more than nine combinations, the policy samples nine complete configurations uniformly without replacement using a fixed random seed. Each declared grid contains at most nine combinations; this cap applies to a search and excludes earlier development.

Search ranges and anchors. Learning rates use multipliers $\{1/3, 1, 3\}$ and principal coefficients use $\{1/2, 1, 2\}$, probing optimization scale and method strength around the anchor. CIFAR-100 WA uses weight decays $\{1, 2, 5\} \times 10^{-4}$, including its public reference values. Most anchors come from public configurations; CIFAR-100 LODE(DER++) and SS-IL use earlier validation selections, and TinyImageNet Joint, WA, and SS-IL transfer the selected CIFAR-100 settings. FEAST uses the learning-rate anchor $\eta = 0.03$, matching the scale of the DER-family reference configurations, and searches $\eta \in \{0.01, 0.03, 0.09\}$ and $\lambda \in \{0.25, 0.5, 1\}$. The coefficient anchor $\lambda = 0.5$ is a development choice; clipping is fixed at 5. Tables D.3 and D.4 disclose candidate sets and the configurations used for the reported runs.

Selection protocol. Candidates use training seed 0 and the fixed training holdout, completing all ten tasks (or Joint's single pooled stage). We select by final task-averaged Class-IL validation accuracy, breaking ties by proximity to the anchor and then candidate ID. The configuration is fixed before retraining on the full training data with independent evaluation seeds. Test results do not rank candidates within these searches.

Here α weights the method's distillation or logit loss, β weights DER++'s replay-label loss, d denotes weight decay, and w_{KD} is SS-IL's distillation weight. LODE(DER++) inherits DER++'s fixed (α, β) for the same dataset and memory budget. Other method-specific parameters retain their pinned implementation settings. To test the selected $M = 500$ configurations with less memory, $M = 200$ reuses their hyperparameters unchanged and performs no additional search. Only the exemplar capacity is reduced; FEAST uses $(\eta, \lambda) = (0.09, 1)$ on both datasets.

Table D.3: CIFAR-100 hyperparameters ($M = 500$ for memory-based methods). Braces list candidates; **bold** marks the setting used in reported runs. Scalars are fixed settings; a dash denotes no additional tuned parameter.

Method	Learning rate η	Principal parameter(s)
Joint	$\{0.01, 0.03, \mathbf{0.09}\}$	–
SGD	0.03	–
ER	0.1	–
DER	0.03	$\alpha = 0.3$
DER++	0.03	$\alpha = 0.3,\ \beta = 0.5$
ER-ACE	0.03	–
LODE(DER++)	$\{0.01, \mathbf{0.03}, 0.09\}$	$\rho \in \{\mathbf{0.4}, 0.8, 1.6\}$
FEAST	$\{0.01, 0.03, \mathbf{0.09}\}$	$\lambda \in \{0.25, 0.5, \mathbf{1}\}$
LwF	0.03	$\alpha = 0.5$
BiC	0.03	$T = 2$
WA	$\{1/30, \mathbf{0.1}, 0.3\}$	$d \in \{\mathbf{1}, 2, 5\} \times 10^{-4}$
LUCIR	0.03	$\lambda_{\text{base}} = 5$
SS-IL	$\{1/900, \mathbf{1/300}, 0.01\}$	$w_{\text{KD}} \in \{\mathbf{0.5}, 1, 2\}$

Table D.4: TinyImageNet hyperparameters ($M = 500$ for memory-based methods). Candidates, reported-run settings, and fixed settings follow the notation in Table D.3.

Method	Learning rate η	Principal parameter(s)
Joint	$\{\mathbf{0.03}, 0.09, 0.27\}$	–
SGD	0.03	–
ER	0.03	–
DER	0.03	$\alpha = 0.1$
DER++	0.03	$\alpha = 0.2,\ \beta = 0.5$
ER-ACE	$\{0.01, 0.03, \mathbf{0.09}\}$	–
LODE(DER++)	$\{0.01, \mathbf{0.03}, 0.09\}$	$\rho \in \{\mathbf{0.05}, 0.1, 0.2\}$
FEAST	$\{0.01, 0.03, \mathbf{0.09}\}$	$\lambda \in \{0.25, 0.5, \mathbf{1}\}$
LwF	0.01	$\alpha = 1$
BiC	$\{0.01, 0.03, \mathbf{0.09}\}$	$T \in \{1, 2, \mathbf{4}\}$
WA	$\{\mathbf{1/30}, 0.1, 0.3\}$	$d \in \{\mathbf{0.5}, 1, 2\} \times 10^{-4}$
LUCIR	$\{\mathbf{0.01}, 0.03, 0.09\}$	$\lambda_{\text{base}} \in \{\mathbf{2.5}, 5, 10\}$
SS-IL	$\{1/900, \mathbf{1/300}, 0.01\}$	$w_{\text{KD}} \in \{\mathbf{0.25}, 0.5, 1\}$

D.5 Evaluation and metrics

Evaluation protocol. We evaluate after each sequential task, including the method's native boundary processing. Class-IL scores all observed classes, masking future classes; Task-IL masks to the true task's class block. Both use the same checkpoint. Joint is evaluated after its pooled training stage and is reported as a separate offline reference.

Accuracy and retention. Let $a_{t,k}$ denote percentage accuracy on task k after task t, under either evaluation protocol. We compute

$$A_t = \frac{1}{t} \sum_{k=1}^{t} a_{t,k}, \tag{35}$$

$$F_T = \frac{1}{T-1} \sum_{k=1}^{T-1} \left(\max_{u=k,\ldots,T-1} a_{u,k} - a_{T,k} \right), \tag{36}$$

$$\text{BWT}_T = \frac{1}{T-1} \sum_{k=1}^{T-1} \left(a_{T,k} - a_{k,k} \right). \tag{37}$$

A_T is final average accuracy; F_T and BWT are forgetting and backward transfer in percentage points. The forgetting maximum excludes the final checkpoint, and negative forgetting is permitted. Tasks receive equal weight, which also equals example weighting for these balanced evaluation sets. Forgetting and BWT are not defined for Joint.

D.6 Uncertainty, computation, and artifacts

Seeds and uncertainty. The $M = 200$ and $M = 500$ results use training seeds $1, 2, 3, 4, 5$, separate from selection seed 0, with fixed class order. Tables report mean $\pm$ sample standard deviation across runs. Paired differences are computed within matching seed IDs before aggregation. Tasks and individual test examples are not treated as independent runs.

Computation. Training uses FP32 with TF32 disabled. The registered run manifests record resolved software versions.

E Complete Method and Algorithmic Details

E.1 Functional encoder and classifier alignment

FEAST allocates all C classifier outputs at initialization and uses full-way CE on current examples. For task $t > 1$, let $\bar{\boldsymbol{z}}_i^{t-1}$ be exemplar i's stored own-block logits and $\widehat{\boldsymbol{z}}_i^{t-1} = P_{k_i}\bar{\boldsymbol{z}}_i^{t-1}$ their centered form. The functional encoder alignment (FEA) estimator is

$$\mathcal{L}_{\mathrm{FEA}}^{(t)} = \frac{1}{|\mathcal{B}_{\mathcal{M}}|}\sum_{i\in\mathcal{B}_{\mathcal{M}}}\frac{1}{K_{k_i}}\left\|P_{k_i}\boldsymbol{z}_{k_i}^{\mathrm{tr}}(A_i(x_i);\mathcal{B}_{\mathcal{M}}) - \widehat{\boldsymbol{z}}_i^{t-1}\right\|_2^2. \tag{38}$$

Here A_i is a sampled augmentation; the batch argument records the BatchNorm dependence. The estimator averages over anchors and own-block classes. We optimize $\mathcal{L}_{\mathrm{CE}}^{\mathrm{full}} + \lambda\mathcal{L}_{\mathrm{FEA}}$ with constant λ, setting $\mathcal{L}_{\mathrm{FEA}} = 0$ on task 1. There is no additional replay-label CE.

Functional classifier alignment (FCA). After backpropagation, FCA projects old classifier gradients through Q_k. Since $Q_kP_k = 0$, this removes the classifier-side FEA gradient, while allowing common coordinates to adapt through CE. Current and future rows receive ordinary gradients. We clip the global gradient norm at 5, apply SGD with zero momentum and weight decay, and restore each old block's pre-step centered weights and bias, as in Eq. 9. This preserves the reference contrast map up to FP32 numerical residuals.

E.2 Memory, input views, and target refresh

Balanced memory. At each boundary, FEAST assigns $\lfloor M/C_t \rfloor$ slots per seen class, distributing the remaining slots in sorted class order. Old examples are subsampled from memory; new examples are sampled without replacement from the completed task. Discarded old data are not reread. Memory contains images, labels, block/source indices, and FP32 own-block logits. Each training step draws 32 anchors uniformly without replacement within the draw.

Input views and targets. Current and anchor batches use Mammoth's crop/flip and normalization conventions. Separate training-mode forwards update shared BatchNorm statistics. Calibration, target refresh, and evaluation use normalized, unaugmented inputs in evaluation mode. After rebuilding memory and calibrating the head, all own-block targets are refreshed and then held fixed throughout the next task. FEA centers these targets when computing the loss. FEA-off controls retain anchor forwards to match BatchNorm exposure.

E.3 Boundary reconciliation

With the encoder and centered head fixed, we fit one scalar per seen block on memory (Appendix C.4) and fold it into $W_k \leftarrow P_kW_k$ and $\boldsymbol{b}_k \leftarrow P_k\boldsymbol{b}_k + \beta_k\mathbf{1}$. Inference then uses a single classifier without an additional calibration module or full teacher network.

Continuation used in ablations. Retaining the calibrated head and restoring the pre-calibration head are functionally equivalent with respect to the within-block function at the task boundary, and yield comparable accuracy in a matched CIFAR-100 comparison. For the ablations, we use the latter convention: targets are refreshed before calibration, and the pre-calibration head is restored before subsequent learning. This prevents boundary calibration from propagating into later optimization, allowing the FEA/FCA interventions to isolate the effects of within-task preservation and adaptation more cleanly.

E.4 Algorithm summary

Algorithm 1 describes the benchmark implementation. The notation below makes the state and updates explicit.

Algorithm 1 FEAST: functional encoder and classifier alignment

Require: Tasks $\{\mathcal{D}_t\}_{t=1}^{T}$, capacity M, coefficient λ

1: Initialize $\Phi = (\theta, W, \boldsymbol{b})$ and $\mathcal{M}_0 \leftarrow \varnothing$
2: **for** $t = 1, \ldots, T$ **do**
3: Reset the task optimizer and learning-rate schedule
4: **for** each training epoch **do**
5: **for** each minibatch $\mathcal{B} \subset \mathcal{D}_t$ **do**
6: Forward augmented current inputs; compute $\mathcal{L}_{\mathrm{CE}}^{\mathrm{full}}$
7: $\mathcal{L}_{\mathrm{FEA}} \leftarrow 0$
8: **if** $t > 1$ **then**
9: Sample $\mathcal{B}_{\mathcal{M}}$ from $\mathcal{M}_{t-1}$; forward augmented anchors separately
10: Compute $\mathcal{L}_{\mathrm{FEA}}$ using Eq. 38
11: **end if**
12: $G \leftarrow \nabla_{\Phi}(\mathcal{L}_{\mathrm{CE}}^{\mathrm{full}} + \lambda \mathcal{L}_{\mathrm{FEA}})$
13: For $k < t$, save $R_k \leftarrow P_k W_k$, $\mathbf{r}_k \leftarrow P_k \boldsymbol{b}_k$
14: For $k < t$, project $G_{W_k} \leftarrow Q_k G_{W_k}$, $G_{\boldsymbol{b}_k} \leftarrow Q_k G_{\boldsymbol{b}_k}$
15: Clip $\|G\|_2$ at 5 and take one SGD step
16: For $k < t$, restore $W_k \leftarrow R_k + Q_k W_k$, $\boldsymbol{b}_k \leftarrow \mathbf{r}_k + Q_k \boldsymbol{b}_k$
17: **end for**
18: Advance the learning-rate schedule
19: **end for**
20: Rebuild balanced $\mathcal{M}_t$ from retained and current-task examples
21: Fix Φ in evaluation mode; compute $e_k(x)$ on $\mathcal{M}_t$
22: Fit block offsets by CE and subtract their mean ($\beta_1 = 0$ if $t = 1$)
23: For $k \le t$, fold $W_k \leftarrow P_k W_k$, $\boldsymbol{b}_k \leftarrow P_k \boldsymbol{b}_k + \beta_k \mathbf{1}$
24: Refresh every exemplar's own-block target $\bar{\boldsymbol{z}}_i$ with the folded head
25: Evaluate Class-IL and Task-IL; retain the head for subsequent learning
26: **end for**

Symbol	Meaning
$\mathcal{D}_t,\ T,\ C$	Task-t training data, number of tasks, and total allocated classes.
$\Phi = (\theta, W, \boldsymbol{b})$	Encoder parameters and the shared linear classifier.
$k,\ K_k,\ \boldsymbol{z}_k$	Class block, its number of classes, and its logits.
$P_k,\ Q_k$	Projectors onto class contrasts and the block-common coordinate; $P_k + Q_k = I$.
$\mathcal{M}_t,\ M$	Memory after task t and its total exemplar capacity.
$\mathcal{B},\ \mathcal{B}_{\mathcal{M}}$	Current-task minibatch and memory-anchor minibatch.
$G,\ R_k,\ \mathbf{r}_k$	Joint parameter gradient and temporary pre-step centered weight/bias references.
$e_k(x),\ \beta_k$	Centered block log-evidence and its fitted scalar offset.
$\lambda,\ \bar{\boldsymbol{z}}_i$	FEA weight and the stored own-block target of exemplar i.

Learning within a task (lines 1–19). Lines 1–7 initialize training and compute current-example CE. Lines 8–11 add FEA on memory anchors; line 12 differentiates the joint objective. Lines 13–16 implement FCA, clipping, and centered-component restoration in that order. Lines 17–19 close the minibatch loop and advance the epoch schedule.

Task boundary (lines 20–26). Line 20 rebuilds memory before calibration. Lines 21–23 fit and fold the block-common offsets without changing centered scores. Line 24 refreshes targets from the head that starts the next task. Lines 25–26 evaluate and continue the task sequence.

Table F.1: Final average accuracy (%) at $M = 200$: mean ± 1 sample standard deviation over five seeds. Bold marks the best sequential mean within each group. Joint is an offline reference.

Additional full teacher network	Method	Full-way training	CIFAR-100		TinyImageNet	
			Task-IL	Class-IL	Task-IL	Class-IL
–	Joint	✓	92.42±0.17	73.14±0.29	81.94±0.21	59.50±0.08
No	SGD	✓	36.66±2.72	9.36±0.06	18.58±1.57	8.13±0.05
	LwF†	✓	26.88±0.54	9.80±0.09	14.89±0.53	8.04±0.71
	ER	✓	66.21±0.83	14.92±0.49	38.61±0.47	8.76±0.13
	DER	✓	66.27±1.08	23.24±1.01	41.68±0.39	12.14±1.24
	DER++	✓	66.06±0.77	25.91±1.56	40.26±1.46	13.65±0.86
	ER-ACE	×	60.87±0.54	24.92±0.83	32.24±0.75	10.44±0.24
	LODE(DER++)	△	69.55±1.17	29.91±1.50	45.90±1.45	8.75±1.84
	FEAST	✓	**79.88±0.31**	**36.99±0.54**	**59.76±0.68**	**16.58±1.30**
Yes	BiC	✓	83.63±0.69	27.54±1.22	66.37±3.78	11.97±0.70
	WA	✓	79.49±1.61	29.96±1.15	65.09±1.55	19.11±2.23
	LUCIR	✓	83.01±0.62	28.41±0.41	68.07±0.54	27.67±0.48
	SS-IL	×	**85.47±0.26**	**40.92±0.73**	**73.37±0.30**	**30.09±0.33**

Full-way training: ✓ uses joint CE normalization across old and new classes; × masks or separates their competition; △ denotes reweighted loss decoupling in LODE. † LwF caches teacher outputs without retaining an additional full teacher network. Joint, SGD, and LwF do not use exemplar memory; their reference results are reused from Table 2.

F Additional Benchmark Results

We use the methods in Table 2 and report means ± 1 sample standard deviation over five seeds. Memory-free references are shared across budgets; Joint is excluded from sequential retention and memory comparisons.

F.1 Performance with a smaller exemplar budget

To assess robustness to a smaller exemplar budget, we reduce $M = 500$ to $M = 200$ while retaining the hyperparameters selected at $M = 500$, without additional search. Table F.1 shows that FEAST maintains the highest mean Class-IL and Task-IL accuracy among methods without an additional full teacher network on both datasets. This persistence of the within-group advantage supports an inductive bias that handles old–new competition with limited replay. Methods retaining a full teacher remain strong competitors, particularly on TinyImageNet.

F.2 Learning trajectories

Figures F.1 and F.2 show all methods in one panel per dataset and protocol. Class-IL and Task-IL use the same checkpoints, separating cross-task competition from within-task discrimination. Each point averages over all tasks observed so far.

F.3 Forgetting and backward transfer

Tables F.2 and F.3 report forgetting and BWT using Appendix D.5. FEAST's accuracy advantage is accompanied by stronger Task-IL retention than the other methods without an additional full teacher network. SS-IL achieves still stronger within-task retention. Forgetting and negative BWT need not coincide when a task improves after acquisition.

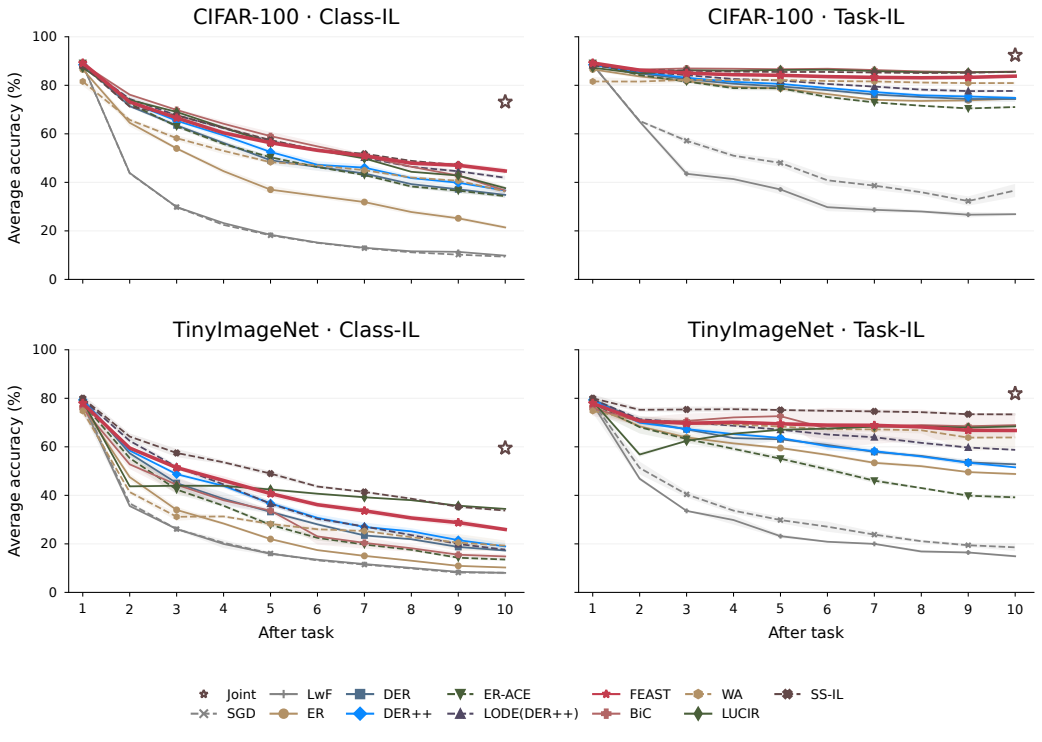


Figure F.1: Learning trajectories at $M = 500$. Rows correspond to CIFAR-100 and TinyImageNet; columns show Class-IL and Task-IL. Every sequential method from Table 2 appears in each corresponding panel. Lines and bands show five-seed means and $\pm 1\sigma$; the open star at task 10 is Joint's offline endpoint.

Table F.2: Forgetting (F, lower is better) and backward transfer (BWT, higher is better), in percentage points, at $M = 500$. Values are mean $\pm 1\sigma$ across five seeds.

	CIFAR-100				TinyImageNet			
Method	Class F	Class BWT	Task F	Task BWT	Class F	Class BWT	Task F	Task BWT
SGD	89.73±0.15	−89.73±0.15	59.40±3.12	−59.40±3.12	78.19±0.37	−78.19±0.37	66.58±1.66	−66.58±1.66
LwF	89.15±0.19	−89.15±0.19	70.18±0.71	−70.18±0.71	77.15±0.93	−77.15±0.93	69.65±0.96	−69.65±0.96
ER	74.58±0.48	−74.58±0.48	16.18±0.22	−16.18±0.22	74.51±0.74	−74.51±0.74	31.84±1.00	−31.84±1.00
DER	56.54±1.17	−56.54±1.17	17.17±1.03	−17.17±1.03	65.92±1.91	−65.92±1.91	28.71±1.00	−28.66±1.03
DER++	53.09±2.10	−53.09±2.10	16.38±0.46	−16.38±0.46	60.01±3.01	−60.01±3.01	28.69±0.79	−28.69±0.79
ER-ACE	53.39±0.62	−53.39±0.62	21.18±0.59	−21.18±0.59	68.22±0.89	−68.22±0.89	41.64±0.91	−41.64±0.91
LODE(DER++)	38.71±1.26	−38.71±1.26	12.99±0.37	−12.99±0.37	53.86±5.97	−53.86±5.97	22.10±1.00	−22.10±1.00
FEAST	28.59±1.53	−28.59±1.53	5.01±0.27	−4.98±0.27	44.97±0.94	−44.97±0.94	10.20±0.43	−10.20±0.43
BiC	51.18±2.14	−51.18±2.14	3.50±0.42	−3.29±0.44	60.02±4.35	−60.02±4.35	4.53±3.35	−4.18±3.42
WA	48.65±0.42	−48.65±0.42	4.68±0.33	−4.56±0.39	56.88±1.23	−56.88±1.23	12.35±3.85	−12.32±3.82
LUCIR	50.42±1.17	−50.42±1.17	2.24±0.30	−1.92±0.44	21.03±1.30	−20.78±1.51	6.72±0.51	−6.58±0.52
SS-IL	25.19±0.51	−22.06±0.61	0.63±0.15	−0.31±0.22	30.91±0.62	−30.71±0.60	2.39±0.13	−2.24±0.15

F.4 Retained memory and accuracy

Figures F.3–F.6 compare final accuracy and retained state at both budgets. We count the current network, any additional full teacher network, and retained memory/algorithm state, excluding optimizer state. All footprints use FP32 image accounting to correct implementation-specific FP32/uint8 storage differences.

FEAST delivers the strongest accuracy among the compared methods without an additional full teacher network while using a footprint comparable to logit replay and substantially below methods retaining a full teacher. The same pattern holds at $M = 200$: strong performance does not require storing another full network.

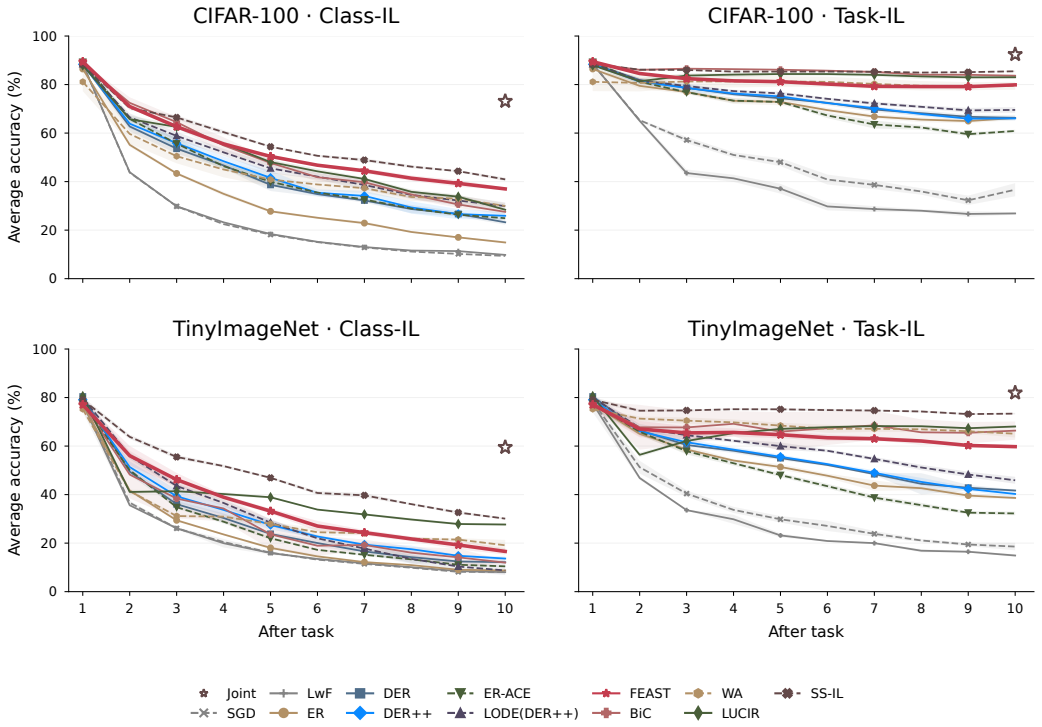


Figure F.2: Learning trajectories at $M = 200$, using the same arrangement and uncertainty convention as Figure F.1. Means and bands summarize five seeds; Joint appears only at its offline endpoint.

Table F.3: Forgetting (F, lower is better) and backward transfer (BWT, higher is better), in percentage points, at $M = 200$. Values are mean $\pm 1\sigma$ across five seeds.

	CIFAR-100				TinyImageNet			
Method	Class F	Class BWT	Task F	Task BWT	Class F	Class BWT	Task F	Task BWT
SGD	89.73±0.15	−89.73±0.15	59.40±3.12	−59.40±3.12	78.19±0.37	−78.19±0.37	66.58±1.66	−66.58±1.66
LwF	89.15±0.19	−89.15±0.19	70.18±0.71	−70.18±0.71	77.15±0.93	−77.15±0.93	69.65±0.96	−69.65±0.96
ER	81.74±0.44	−81.74±0.44	24.95±1.24	−24.95±1.24	76.43±0.57	−76.43±0.57	43.31±0.73	−43.31±0.73
DER	70.34±2.18	−70.34±2.18	25.70±0.74	−25.70±0.74	72.62±2.90	−72.62±2.90	41.00±1.11	−40.97±1.15
DER++	66.13±2.55	−66.13±2.55	25.67±0.70	−25.67±0.70	66.15±1.92	−66.15±1.92	40.63±0.73	−40.63±0.73
ER-ACE	62.87±1.20	−62.87±1.20	32.50±0.78	−32.50±0.78	72.89±0.48	−72.89±0.48	49.71±1.03	−49.71±1.03
LODE(DER++)	55.38±3.08	−55.38±3.08	21.96±1.44	−21.96±1.44	50.84±7.37	−50.84±7.37	35.20±0.93	−35.20±0.93
FEAST	40.47±0.77	−40.47±0.77	9.88±0.19	−9.88±0.19	60.45±2.32	−60.45±2.32	18.68±0.67	−18.68±0.67
BiC	63.47±1.69	−63.47±1.69	5.49±0.87	−5.36±0.88	63.29±2.20	−63.29±2.20	6.16±3.36	−5.81±3.36
WA	56.45±1.02	−56.45±1.02	5.93±0.37	−5.90±0.40	56.84±2.35	−56.84±2.35	10.70±1.23	−10.69±1.23
LUCIR	62.47±0.72	−62.47±0.72	4.80±0.58	−4.69±0.59	41.98±1.00	−41.98±1.00	6.79±0.48	−6.60±0.56
SS-IL	29.05±0.46	−27.41±0.36	0.92±0.08	−0.64±0.17	34.38±0.27	−33.94±0.33	2.05±0.13	−1.84±0.13

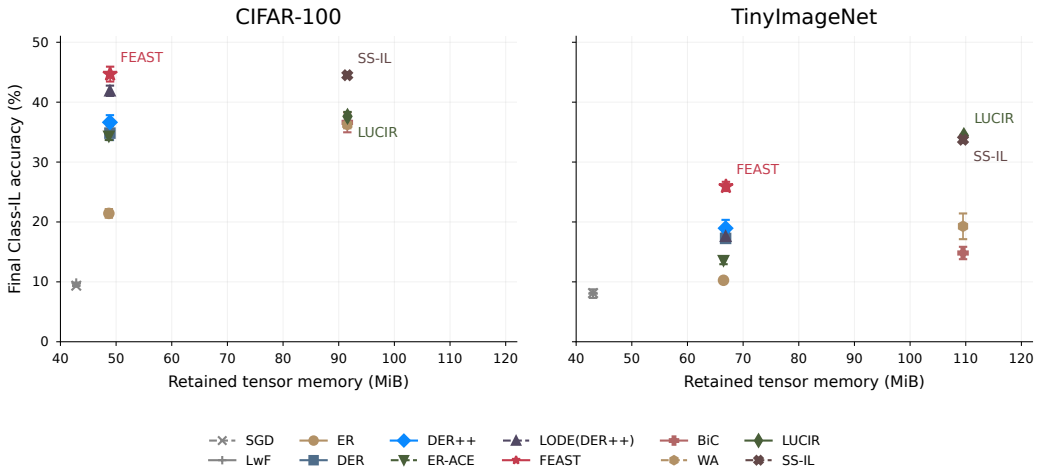


Figure F.3: Retained tensor memory versus final Class-IL accuracy at $M = 500$, with one panel per dataset. All exemplar images are accounted as FP32; optimizer state is excluded. Points show five-seed means with $\pm 1\sigma$ accuracy bars. Joint is excluded from the memory comparison.

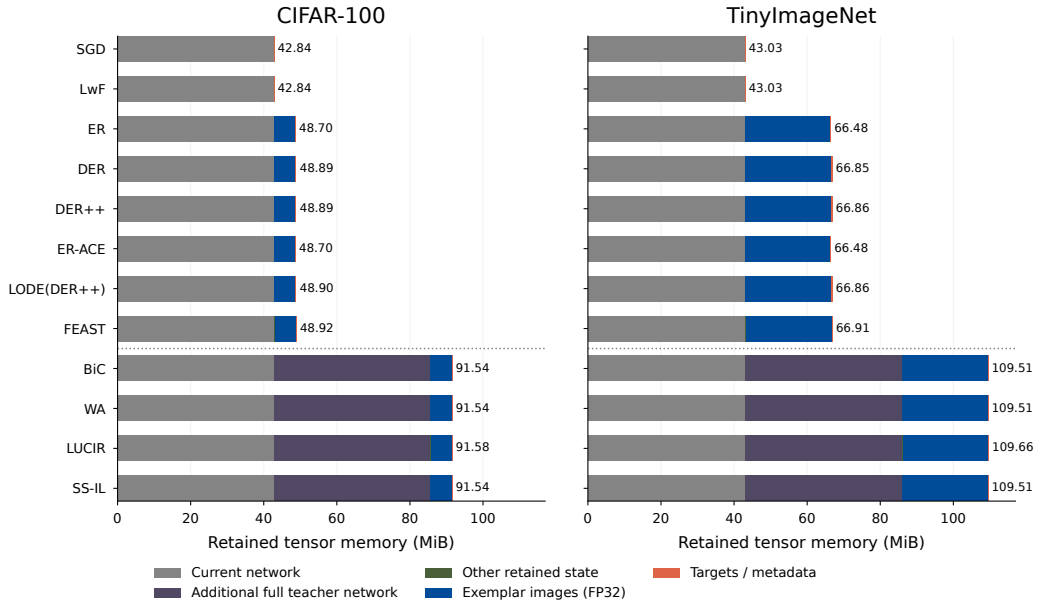


Figure F.4: Retained-memory components at $M = 500$ under the same FP32-image, optimizer-excluded accounting. The dotted separator divides methods without and with an additional full teacher network. Labels give total MiB; small components are included even when visually indistinguishable.

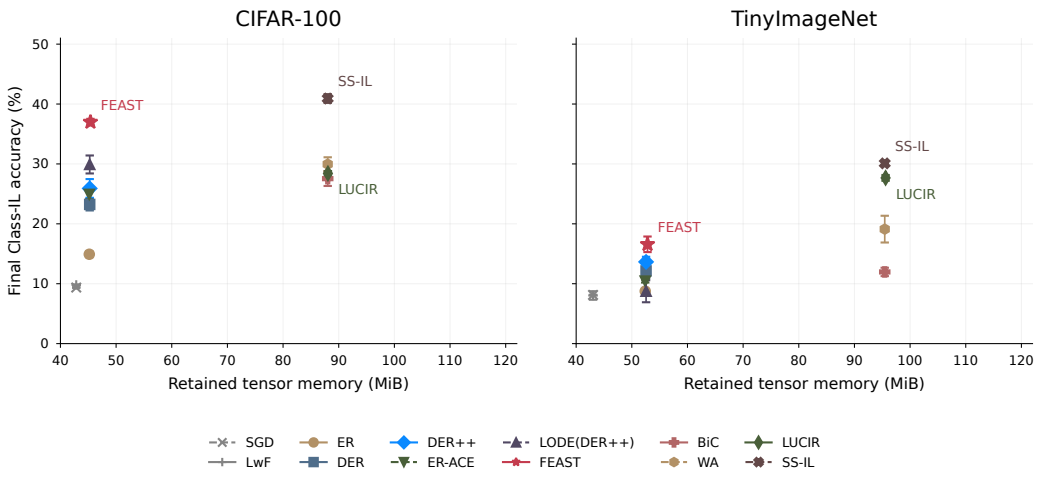


Figure F.5: Retained memory versus Class-IL accuracy at $M = 200$, using the same accounting as Figure F.3. Accuracy bars show five-seed sample standard deviations; FP32 image accounting and optimizer exclusion match the $M = 500$ figures.

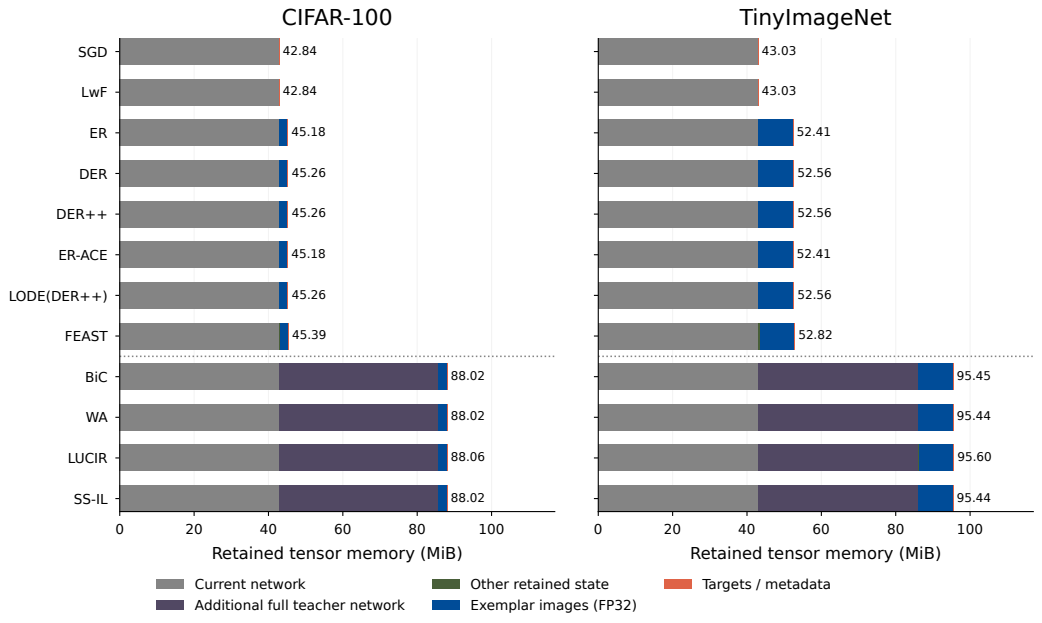


Figure F.6: Retained-memory components at $M = 200$, with FP32 exemplar images and no optimizer state. The accounting convention matches the other memory figures.

Table G.1: Complete mean results for the FEA×FCA intervention over five seeds. Visible denotes the classifier-visible fraction of cumulative representational-change energy. Representation, classifier-Q, and net denote RMS common-score changes in logits. CE-AUC is the held-out full-way cross-entropy area, and forgetting is Task-IL forgetting in percentage points.

FEA	FCA	Visible (%)	Rep.	Classifier-Q	Net	CE-AUC	Forgetting
Off	Off	39.43	4.133	0.000	4.133	0.971	64.67
Off	On	47.15	1.296	4.312	5.045	0.872	61.39
On	Off	15.21	3.757	0.000	3.757	1.127	12.48
On	On	15.86	0.602	4.721	4.984	0.776	5.26

Table G.2: Function-defining preservation in the FEA× FCA intervention. Values are mean ± sample standard deviation over five seeds. Relative function-defining output change is normalized by the acquisition-time function-defining output scale. Reference-margin erosion is measured in raw logits using the fixed acquisition-time reference classifier.

FEA	FCA	Relative function-defining output change	Reference-margin erosion
Off	Off	0.9212 ± 0.0045	9.451 ± 0.237
Off	On	0.8944 ± 0.0062	9.102 ± 0.186
On	Off	0.5693 ± 0.0130	3.348 ± 0.227
On	On	0.4495 ± 0.0070	1.969 ± 0.138

G Component Ablations and Specificity Controls

G.1 Functional Encoder Alignment × Functional Classifier Alignment

We report the complete results underlying the FEA×FCA intervention in Sec. 5.3. The four conditions independently enable or disable functional encoder alignment (FEA) and functional classifier alignment (FCA), while keeping the function-defining component of each previous classifier fixed across all conditions. Metric definitions and aggregation conventions are provided in Appendix H.

Across both FCA conditions, enabling FEA reduces the classifier-visible fraction of representational change by 27.76 percentage points (95% CI $[27.30, 28.21]$). With FEA enabled, FCA reduces the representation-side common-score contribution by 3.155 RMS logits (95% CI $[2.993, 3.317]$), while increasing the classifier-side contribution by 4.721 RMS logits (95% CI $[4.630, 4.812]$) and the net common-score change by 1.227 RMS logits (95% CI $[1.066, 1.388]$).

With FEA enabled, FCA also reduces held-out full-way CE-AUC by 0.351 (95% CI $[0.321, 0.381]$), Task-IL forgetting by 7.22 percentage points (95% CI $[5.90, 8.53]$), relative function-defining output change by 0.120 (95% CI $[0.107, 0.132]$), reference-margin erosion by 1.379 logits (95% CI $[1.096, 1.661]$), and reference-boundary crossing by 2.88 percentage points (95% CI $[2.21, 3.55]$).

The FEA×FCA interaction is significant for both acquisition (CE-AUC; $p = 1.16 \times 10^{-4}$) and Task-IL forgetting ($p = 0.00331$). These tests use the five seed-level difference-of-differences and a two-sided one-sample t-test with four degrees of freedom.

G.2 Effect of Removing Centering from FEA

To isolate the role of centering in FEA, we replace its centered objective with an uncentered full-logit matching objective while leaving the remainder of FEAST unchanged. Both conditions keep FCA enabled and use identical training settings and paired seeds. The only difference is whether the alignment loss preserves only the function-defining coordinates or additionally constrains the block-common coordinate.

Let $\bar{z}_k(x)$ denote the stored raw own-block target and $\widehat{z}_k(x) := P_k \bar{z}_k(x)$ its centered form. Centered FEA penalizes

$$\|P_k \boldsymbol{z}_k(x) - \widehat{z}_k(x)\|_2^2 ,$$

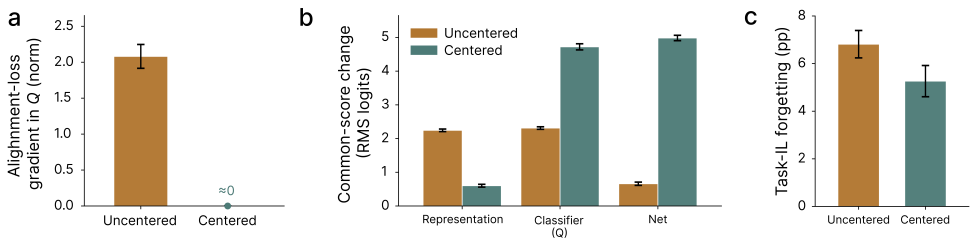


Figure G.1: **Effect of removing centering from FEA.** Matched comparison over five paired seeds with FCA enabled. **(a)** The uncentered variant introduces a substantial classifier-gradient component along the common coordinate Q, whereas this component is numerically negligible under centered FEA. **(b)** Removing centering shifts common-score adaptation away from classifier-Q coordinates and toward the representation, while substantially reducing the resulting net common-score change. Representation, classifier, and net quantities are RMS magnitudes and are not additive shares. **(c)** Removing centering increases Task-IL forgetting. Error bars denote 95% confidence intervals across seeds.

whereas the uncentered variant penalizes

$$\|\boldsymbol{z}_k(x) - \bar{\boldsymbol{z}}_k(x)\|_2^2 = \big\|P_k\big(\boldsymbol{z}_k(x) - \bar{\boldsymbol{z}}_k(x)\big)\big\|_2^2 + \big\|Q_k\big(\boldsymbol{z}_k(x) - \bar{\boldsymbol{z}}_k(x)\big)\big\|_2^2 .$$

Thus, removing centering additionally constrains the common coordinate that is invariant to the preserved within-task function. This distinction is also visible in the optimization dynamics: the norm of the Q_k component of the classifier gradient induced by the alignment loss is numerically negligible under centered FEA (1.9×10^{-7}), but increases to 6.58 under the uncentered variant (Fig. G.1a).

Removing centering substantially changes how common-score adaptation is carried (Fig. G.1b). Using the same decomposition as in Sec. 5.3, the representation-side contribution increases from 0.602 ± 0.033 to 2.245 ± 0.030 RMS logits, whereas the classifier-side Q contribution decreases from 4.721 ± 0.073 to 2.312 ± 0.030. The resulting net common-score change decreases from 4.984 ± 0.064 to 0.660 ± 0.039 RMS logits. These RMS quantities are not additive shares, so their magnitudes should not be interpreted as a partition of the net change. Nevertheless, the comparison shows a clear redistribution: when the alignment loss additionally constrains the common coordinate, less common-score adaptation is carried through the available classifier-Q coordinates, while substantially more is carried through the representation and the realized net adjustment is much smaller.

This restriction is accompanied by worse acquisition and retention. Removing centering increases held-out full-way CE-AUC from 0.776 ± 0.015 to 1.183 ± 0.011, corresponding to a paired increase of 0.407 (95% CI $[0.380, 0.433]$). Task-IL forgetting likewise increases from 5.26 ± 0.53 to 6.82 ± 0.46 percentage points, a paired increase of 1.55 points (95% CI $[0.73, 2.38]$; Fig. G.1c). Final Class-IL and Task-IL accuracy also decrease by 2.35 and 2.07 percentage points, respectively. The direction of these differences is consistent across all five paired seeds.

Together, this ablation clarifies why FEA is centered. Centering is not introduced as a generally superior form of logit matching; rather, it prevents the alignment loss from constraining a coordinate that does not determine the preserved within-task function. When this separation is removed, common-score adaptation shifts away from the functionally invariant classifier coordinates, the effective common-score adjustment is reduced, and both acquisition and retention deteriorate.

H Analysis Metrics and Aggregation

This appendix defines the nonstandard analysis metrics used in Secs. 5.3–5.4. Standard continual-learning metrics, including Class-IL and Task-IL accuracy, forgetting, and backward transfer, follow the definitions in Appendix D.5.

We follow the notation of the main text, using a superscript (t) only to distinguish checkpoints. Reference quantities W_k^{ref} and b_k^{ref} denote their acquisition-time values, and seed indices are suppressed for readability.

H.1 Analysis Population and Aggregation

The mechanism analyses in Secs. 5.3–5.4 use a fixed diagnostic population of 50 official-test examples per class. For task block k, let

$$D_k = \{(x_i, y_i)\}_{i=1}^{N_k}$$

denote this population, with $N_k = 500$ for the CIFAR-100 analyses. Task k is acquired immediately after training task k, and $t > k$ denotes a later task boundary. We write

$$a = t - k$$

for the elapsed number of subsequently learned tasks.

For a block–checkpoint scalar $m_{s,k,t}$ measured in seed s, we use three aggregation operators:

$$H_s(m) = \frac{1}{T-1} \sum_{t=2}^{T} \frac{1}{t-1} \sum_{k=1}^{t-1} m_{s,k,t}, \tag{39}$$

$$F_s(m) = \frac{1}{T-1} \sum_{k=1}^{T-1} m_{s,k,T}, \tag{40}$$

$$A_s(m; a) = \frac{1}{T-a} \sum_{k=1}^{T-a} m_{s,k,k+a}, \qquad a = 1, \ldots, T-1. \tag{41}$$

Thus, H_s first averages old blocks within each checkpoint and then averages checkpoints; it is not a uniform average over all block–checkpoint pairs. F_s averages old tasks at the final checkpoint, whereas A_s averages all task blocks observed at a given task age. Age zero is excluded.

For $X \in \mathbb{R}^{N_k \times K_k}$, we define

$$\mathrm{rms}(X) = \sqrt{\frac{1}{N_k K_k} \sum_{i=1}^{N_k} \sum_{c=1}^{K_k} X_{ic}^2}.$$

For scalar-valued quantities, RMS is defined analogously over examples.

All mechanism analyses use the five training seeds as independent replicates. Reported 95% confidence intervals are

$$\bar{v} \pm t_{0.975,4} \frac{s_v}{\sqrt{5}},$$

where s_v is the sample standard deviation across seed-level quantities. Paired comparisons are formed within matching seeds before computing the confidence interval.

H.2 Stability and Adaptation Allocation

Classifier-visible representational change. For old task k, let

$$\widetilde{W}_k^{\mathrm{ref}} = P_k W_k^{\mathrm{ref}}$$

denote the centered acquisition-time reference classifier. We obtain an orthonormal basis for the row space of $\widetilde{W}_k^{\mathrm{ref}}$ from the right singular vectors whose singular values exceed $10^{-6}\sigma_{\max}$, and let Π_k denote the corresponding orthogonal projector in representation space.

The cumulative representation change from acquisition to checkpoint t is

$$\Delta h_{k\to t}(x) = h^{(t)}(x) - h^{(k)}(x).$$

The classifier-visible change fraction is

$$f_{s,k,t} = \frac{\sum_{x\in D_k} \|\Pi_k \Delta h_{k\to t}(x)\|_2^2}{\max\left(\sum_{x\in D_k} \|\Delta h_{k\to t}(x)\|_2^2, 10^{-12}\right)}.$$

The ratio is formed separately within each block before applying Eq. 39. Figure 2a reports $100H_s(f)$.

Common-score adaptation. The block-common component of the classifier can be parameterized by

$$Q_k W_k^{(u)} = \mathbf{1}(v_k^{(u)})^\top, \qquad Q_k b_k^{(u)} = \alpha_k^{(u)}\mathbf{1},$$

where

$$v_k^{(u)} = \frac{1}{K_k}(W_k^{(u)})^\top \mathbf{1}, \qquad \alpha_k^{(u)} = \frac{1}{K_k}\mathbf{1}^\top b_k^{(u)}.$$

The resulting scalar common score is

$$a_k^{(u)}(x) = (v_k^{(u)})^\top h^{(u)}(x) + \alpha_k^{(u)}.$$

Unlike the acquisition-referenced analyses below, Fig. 2b decomposes common-score change across *adjacent* task boundaries. For an old block $k < t$, define

$$R_{k,t}^a(x) = (v_k^{(t-1)})^\top \left(h^{(t)}(x) - h^{(t-1)}(x)\right), \tag{42}$$

$$C_{k,t}^a(x) = \left(v_k^{(t)} - v_k^{(t-1)}\right)^\top h^{(t)}(x) + \alpha_k^{(t)} - \alpha_k^{(t-1)}. \tag{43}$$

These terms satisfy the exact identity

$$R_{k,t}^a(x) + C_{k,t}^a(x) = a_k^{(t)}(x) - a_k^{(t-1)}(x).$$

The cross term

$$(v_k^{(t)} - v_k^{(t-1)})^\top (h^{(t)} - h^{(t-1)})$$

is therefore included in $C_{k,t}^a$.

For each block–transition pair, we report

$$\mathrm{RMS}_R = \sqrt{\frac{1}{N_k}\sum_{x\in D_k}\left(R_{k,t}^a(x)\right)^2},$$

$$\mathrm{RMS}_C = \sqrt{\frac{1}{N_k}\sum_{x\in D_k}\left(C_{k,t}^a(x)\right)^2},$$

and

$$\mathrm{RMS}_{\mathrm{net}} = \sqrt{\frac{1}{N_k}\sum_{x\in D_k}\left(R_{k,t}^a(x) + C_{k,t}^a(x)\right)^2}.$$

These quantities are RMS magnitudes and are not additive shares.

Held-out full-way CE-AUC. During training on task t, let $\ell_t(u_j)$ denote the cross-entropy on the fixed held-out current-task panel at normalized training progress $u_j \in [0,1]$. The loss is computed over the full classifier output space, including allocated future-class outputs:

$$\ell_t(u_j) = \frac{1}{N_t}\sum_{(x_i,y_i)\in D_t}\left[\log\sum_{c=1}^{C}\exp z_c^{(u_j)}(x_i) - z_{y_i}^{(u_j)}(x_i)\right].$$

The task-level area is computed by trapezoidal integration,

$$U_{s,t} = \sum_j \frac{u_{j+1} - u_j}{2} \left[\ell_t(u_j) + \ell_t(u_{j+1})\right],$$

and the reported acquisition metric is

$$U_s = \frac{1}{T-1} \sum_{t=2}^{T} U_{s,t}.$$

We report the raw CE area without normalization by $\log C$.

Relative function-defining output change. For task k, define the acquisition centered-logit scale

$$S_{s,k} = \mathrm{rms}\left(\left\{P_k z_k^{\mathrm{ref}}(x) : x \in D_k\right\}\right),$$

where

$$z_k^{\mathrm{ref}}(x) = W_k^{\mathrm{ref}} h^{(k)}(x) + b_k^{\mathrm{ref}}.$$

For this metric, we evaluate the current representation through the fixed acquisition-time classifier. In the intervention, the function-defining classifier component remains fixed, while the classifier-side common component can change when FCA is enabled. The relative function-defining output change shown in Fig. 2d is

$$d_{s,k,t}^{\mathrm{ref}} = \frac{\mathrm{rms}\left(\left\{P_k W_k^{\mathrm{ref}}\left(h^{(t)}(x) - h^{(k)}(x)\right) : x \in D_k\right\}\right)}{\max(S_{s,k}, 10^{-12})}.$$

For the 2×2 intervention, let $y_{fq,s}$ denote a seed-level outcome for FEA state $f \in \{0,1\}$ and FCA state $q \in \{0,1\}$. The FEA×FCA interaction reported in Sec. 5.3 is the seedwise difference of differences

$$\Delta_{\mathrm{FEA}\times\mathrm{FCA},s} = (y_{11,s} - y_{10,s}) - (y_{01,s} - y_{00,s}),$$

tested against zero across the five seeds.

H.3 Decision-Retention Metrics

For each example $x_i \in D_k$, we fix at acquisition the strongest non-true class,

$$c_i^* = \arg\max_{c \in \mathcal{C}_k \setminus \{y_i\}} z_{k,c}^{\mathrm{ref}}(x_i).$$

Let $(w_{k,c}^{\mathrm{ref}})^\top$ denote row c of W_k^{ref}. Using the acquisition classifier throughout, the corresponding reference margin at checkpoint u is

$$m_i^{(u)} = \left(w_{k,y_i}^{\mathrm{ref}} - w_{k,c_i^*}^{\mathrm{ref}}\right)^\top h^{(u)}(x_i) + b_{k,y_i}^{\mathrm{ref}} - b_{k,c_i^*}^{\mathrm{ref}}.$$

Reference-margin erosion is

$$E_{s,k,t} = \frac{1}{N_k} \sum_{x_i \in D_k} \left[m_i^{(k)} - m_i^{(t)}\right].$$

The competitor is fixed at acquisition, and all diagnostic examples are included regardless of whether they were correctly classified at acquisition. The metric is a signed raw-logit margin change; no clipping or geometric normalization is applied.

Reference-boundary crossing. Let

$$I_{s,k} = \left\{i : \arg\max_{c \in \mathcal{C}_k} z_{k,c}^{\mathrm{ref}}(x_i) = y_i\right\}$$

denote the acquisition-correct cohort. At checkpoint t, we insert the current representation into the fixed acquisition classifier:

$$\hat{y}_i^{\mathrm{ref},(t)} = \arg\max_{c \in \mathcal{C}_k} \left[W_k^{\mathrm{ref}} h^{(t)}(x_i) + b_k^{\mathrm{ref}}\right]_c.$$

The reference-boundary crossing rate is

$$B_{s,k,t} = \frac{100}{|I_{s,k}|} \sum_{i \in I_{s,k}} \mathbf{1}\left[\hat{y}_i^{\mathrm{ref},(t)} \neq y_i\right].$$

Unlike reference-margin erosion, this metric evaluates the full within-task argmax at each checkpoint rather than following one fixed competitor.

H.4 Comparison with Logit Replay

The comparison in Sec. 5.4 requires distinguishing the acquisition and current classifiers explicitly. For a task block k and later checkpoint t, we use the local shorthand

$$W_0 := W_k^{(k)}, \qquad b_0 := b_k^{(k)}, \qquad W_1 := W_k^{(t)}, \qquad b_1 := b_k^{(t)},$$

and, for a given example x,

$$h_0 := h^{(k)}(x), \qquad h_1 := h^{(t)}(x).$$

Function-defining representation–classifier decomposition. For each example,

$$R(x) = P_k W_0 (h_1 - h_0), \tag{44}$$

$$C(x) = P_k \left[(W_1 - W_0)h_1 + (b_1 - b_0)\right]. \tag{45}$$

These terms satisfy

$$R(x) + C(x) = P_k \left[W_1 h_1 + b_1 - (W_0 h_0 + b_0)\right] = P_k \left[z_k^{(t)}(x) - z_k^{(k)}(x)\right].$$

Bias changes and the cross term $P_k \Delta W \, \Delta h$ are included in C. For FEAST, the function-defining classifier component is fixed, so $C = 0$ up to numerical precision; DER permits both terms to vary.

Representation–classifier alignment. For each block–checkpoint pair, let $R, C \in \mathbb{R}^{N_k \times K_k}$ collect the corresponding samplewise contributions. The alignment reported in Sec. 5.4 is

$$\cos(R, C) = \frac{\sum_{i,c} R_{ic} C_{ic}}{\sqrt{\sum_{i,c} R_{ic}^2} \sqrt{\sum_{i,c} C_{ic}^2}}.$$

Thus, the complete example-by-class tensors are flattened before a single cosine is computed for each block–checkpoint pair. This is not an average of samplewise cosine similarities.

Relative function-defining output change with a current classifier. The DER comparison allows both the representation and the classifier to change. Using the acquisition scale $S_{s,k}$ defined in Appendix H.2, the relative change shown in Fig. 3d is

$$d_{s,k,t}^{\mathrm{live}} = \frac{\mathrm{rms}\left(\left\{P_k \left[z_k^{(t)}(x) - z_k^{(k)}(x)\right] : x \in D_k\right\}\right)}{S_{s,k}}.$$

This differs from $d_{s,k,t}^{\mathrm{ref}}$ in Appendix H.2, which holds the acquisition-time classifier fixed.

Within-task decision failure. For each method and seed, define the acquisition-correct cohort

$$I_{s,k} = \left\{i : \arg\max_{c \in \mathcal{C}_k} z_{k,c}^{(k)}(x_i) = y_i\right\}.$$

At checkpoint t, the current within-task prediction is

$$\hat{y}_i^{\mathrm{live},(t)} = \arg\max_{c \in \mathcal{C}_k} \left[P_k \left(W_k^{(t)} h^{(t)}(x_i) + b_k^{(t)}\right)\right]_c.$$

The within-task decision-failure rate is

$$L_{s,k,t} = \frac{100}{|I_{s,k}|} \sum_{i \in I_{s,k}} \mathbf{1}\left[\hat{y}_i^{\mathrm{live},(t)} \neq y_i\right].$$

Unlike reference-boundary crossing, this metric uses both the current representation and the current classifier and therefore allows classifier co-adaptation. Figure 3e reports the task-age trajectory $A_s(L; a)$, whereas the final within-task decision-failure result in Sec. 5.4 uses $F_s(L)$.

Aggregate vector decomposition in Fig. 3b. Figure 3b is constructed from normalized squared energies rather than from direct averages of the magnitudes of R and C. For each block–checkpoint pair, define

$$e_R = \frac{\mathrm{mean}_{i,c}\, R_{ic}^2}{S_{s,k}^2}, \qquad e_C = \frac{\mathrm{mean}_{i,c}\, C_{ic}^2}{S_{s,k}^2},$$
$$e_I = \frac{2\,\mathrm{mean}_{i,c}\, R_{ic}C_{ic}}{S_{s,k}^2}, \qquad e_N = \frac{\mathrm{mean}_{i,c}(R+C)_{ic}^2}{S_{s,k}^2},$$

so that

$$e_N = e_R + e_C + e_I.$$

After applying the historical aggregation in Eq. 39 and averaging the resulting energies across seeds, denote them by $\bar{e}_R, \bar{e}_C, \bar{e}_I, \bar{e}_N$. The displayed vector lengths are

$$r = \sqrt{\bar{e}_R}, \qquad c = \sqrt{\bar{e}_C}, \qquad n = \sqrt{\bar{e}_N},$$

with angle

$$\cos\theta = \frac{\bar{e}_I}{2rc}.$$

This two-dimensional construction preserves the aggregate energy decomposition but does not represent two coordinates of the original feature or logit space. In particular, the angle displayed in Fig. 3b is not numerically identical to the mean blockwise $\cos(R, C)$ reported in Sec. 5.4.