ADDRESS MIGRATION FOR REPLAY-FREE CONTINUAL LEARNING WITH SPARSE ROUTED MEMORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Updating a representation can invalidate the coordinates used to retrieve previously learned memories. We study this problem in a continual learner that stores sparse routing prototypes and local experts, but retains no past examples or feature exemplars. Our approach, *address migration*, fits a linear map from new activations to the previous memory coordinates using paired evaluations of current unlabeled inputs. A wake–sleep schedule separates supervised expert updates from unsupervised projection updates, and a distance-profile regularizer limits geometric distortion during sleep. We characterize exact recovery for linear drift, separate finite-sample error from irreducible approximation error, bound changes in top-k codes by the distribution of alignment error relative to coordinate margins, and show that cosine nearest-class-mean classification in code space is a special case of the architecture. On Split CIFAR-100 with a frontend pretrained on all classes without labels, migration with guided sleep achieves $27.3 \pm 1.1\%$ final accuracy, compared with $23.9 \pm 0.1\%$ for unguided migration and $28.1 \pm 1.0\%$ for the frozen configuration (three seeds), recovering about 80% of the accuracy cost of moving the projection and bringing a mobile representation within one point of the frozen configuration. Increasing routing capacity improves retention, whereas routing selectivity alone cannot compensate for weak per-task learning. The results establish coordinate alignment as a mechanism for keeping sparse routed memories addressable while the representation changes.

1 INTRODUCTION

Continual learning must preserve earlier predictions while incorporating new information. Representation updates make this difficult in a specific way: a stored prototype or local predictor is useful only if its coordinates remain compatible with the encoder that retrieves it, a requirement studied in retrieval as backward-compatible representation learning (Shen et al., 2020). This source of interference accompanies the more familiar overwriting of predictive parameters (McCloskey & Cohen, 1989; French, 1999; Kirkpatrick et al., 2017). Freezing the representation avoids coordinate drift but restricts adaptation, although a body adapted only in the first session and then frozen is a strong replay-free baseline (Panos et al., 2023). Retaining examples permits rehearsal or re-encoding, but changes the learner's storage and data-access requirements (Rolnick et al., 2019; Hayes et al., 2020).

We ask a narrower question: *can a learner update its representation while retaining the coordinates used by a sparse routed memory, without revisiting historical examples?* At an update boundary, the old and new encoders can evaluate the same current inputs. Their paired activations provide a regression problem: map the new representation back to the coordinates in which the memory was learned. We call this operation *address migration*. Related work already uses current-task drift to compensate old class prototypes without exemplars (Yu et al., 2020). Our focus is a shared map applied before sparse binarization and routing, together with an analysis of when this map preserves addresses and when its residual affects retrieval.

The learner combines a frozen feature extractor, a trainable linear projection, sparse top-k codes, and local classification experts. During *wake*, the projection is fixed and only the routed memory learns. During *sleep*, an unlabeled reconstruction objective updates the projection; migration then aligns the new activations to the previous memory coordinates. A distance-profile regularizer makes these

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.

updates more conservative while allowing rotations and uniform changes of scale. This separation lets us distinguish interference caused by supervised memory updates from interference introduced at representation-update boundaries.

The projection above the frozen frontend is the component that moves. The self-supervised frontend is pretrained on the full CIFAR-100 training split, including classes that arrive later in the labeled stream. The experiment therefore studies exemplar-free incremental classification *after all-class unlabeled pretraining*. Prototypes, experts, and a migration matrix persist across tasks.

Within this setting, the main result is a reduction in the cost of representation updates. At routing depth 11, guided migration improves final accuracy from 23.9% to 27.3%, compared with 28.1% for the frozen control, so the projection can move at almost no accuracy cost. The analysis identifies what governs the remaining cost: alignment error changes sparse codes only near the selection threshold, and a routing change is costly when experts serve narrow subsets of the stream. The evidence also distinguishes class separability from the usefulness of a representation for selective routing. More routing capacity improves retention on SSL features, but a highly selective router cannot compensate for weak within-task learning.

We make three contributions:

1. **A migration formulation for sparse routed memory.** We specify a persistent activation-alignment map, its update from current unlabeled inputs, and its interaction with a wake–sleep schedule (Section 4).
2. **Conditions for preserving addresses.** We characterize recoverable linear functionals, account for ridge bias and finite-sample error, bound top-k changes by the distribution of alignment error relative to each coordinate's margin, and show that cosine nearest-class-mean classification in code space is the special case of the routed memory with one leaf per class, class-mean prototypes, and no representation updates (Sections 3–4; Appendix A).
3. **A controlled empirical study of capacity and drift.** We show that guided migration recovers about 80% of the accuracy cost of projection updates at depth 11, that routing capacity improves retention on self-supervised features, and that routing selectivity cannot compensate for weak within-task learning (Section 5).

2 Related Work

Continual learning without example replay. Regularization methods retain parameter-importance information (Kirkpatrick et al., 2017; Zenke et al., 2017), while gradient-projection methods retain subspaces that constrain later updates (Zeng et al., 2019; Saha et al., 2021; Wang et al., 2021). Parameter isolation and expansion allocate additional capacity (Rusu et al., 2016), and gated expert networks route each input to a single task expert (Aljundi et al., 2017). Our local experts also use persistent model capacity; migration addresses coordinate compatibility rather than replacing these broader approaches to interference. Surveys review continual learning (Parisi et al., 2019) and methods built on pretrained models (Zhou et al., 2024a), and benchmark taxonomies distinguish the task-, domain-, and class-incremental settings (van de Ven et al., 2022; Wang et al., 2024; Zhou et al., 2024c). We use class-incremental evaluation without a task oracle, but allow task boundaries during training.

Drift compensation and prototype memory. Semantic Drift Compensation (SDC) estimates the displacement of old class prototypes from the drift observed on current-task data, without storing exemplars (Yu et al., 2020); later work perturbs current samples toward old prototypes to improve this estimate (Goswami et al., 2024) or learns a drift-compensation map for supervised and self-supervised backbones (Gomez-Villa et al., 2024). This family, and the learned drift map in particular, is the closest precedent for the data-access pattern studied here. PASS retains class prototypes and uses prototype augmentation and self-supervision to support incremental classification (Zhu et al., 2021), and elastic feature consolidation regularizes drift along feature directions important to earlier tasks (Magistri et al., 2024), a role comparable to that of guided sleep. Outside continual classification, compatible-embedding methods either constrain a new encoder to remain comparable with stored features (Shen et al., 2020) or learn a transformation between old and new embedding spaces (Ramanujan et al., 2022). Our formulation uses a shared linear alignment before top-k binarization, so the same aligned coordinates feed the router and local experts. The distinction is the

memory interface and its stability analysis, not the general idea of estimating historical drift from current inputs.

Stable representations and latent replay. Latent replay stores feature exemplars, reducing the cost of image rehearsal while requiring control of drift below the replay layer (Pellegrini et al., 2020; Hayes et al., 2020). Generated replay provides another way to revisit previous distributions (Shin et al., 2017; van de Ven et al., 2020). Prompt-based methods exploit pretrained backbones (Wang et al., 2022b;a); classifier-alignment approaches also highlight the importance of pretrained representations (Zhang et al., 2023). On frozen pretrained features, class-prototype classifiers are strong exemplar-free baselines (Zhou et al., 2025; Panos et al., 2023), and can be strengthened with class-specific covariances (Goswami et al., 2023) or a frozen random projection with decorrelated prototypes (McDonnell et al., 2023); adapter-based methods instead expand task-specific subspaces while keeping the backbone frozen (Zhou et al., 2024b). We retain no historical feature exemplars, but similarly depend on a strong frozen frontend. The evaluated mobility is confined to the projection above that frontend.

Wake–sleep schedules and sparse memory. Complementary-learning-systems theory motivates separating rapid acquisition from slower consolidation (McClelland et al., 1995). Several computational sleep methods use replay or internally generated activity (Tadros et al., 2022; González et al., 2020; Golden et al., 2022); SIESTA uses latent rehearsal during sleep (Harun et al., 2023). Here, sleep uses the current unlabeled data to update a projection and estimate alignment. Sparse distributed memory and sparse distributed codes motivate the address representation (Kanerva, 1988; Hawkins & Ahmad, 2016), and an MLP built from its components with top-k sparse activations learns continually without replay or task information (Bricken et al., 2023). Work on loss of plasticity motivates studying representation updates over time (Dohare et al., 2024).

3 Problem Formulation

Problem statement. A learner observes a sequence of tasks with disjoint class sets and may not retain examples or feature exemplars from completed tasks. Its memory consists of objects addressed in a code space—routing prototypes p_j and local experts f_j—together with a representation map that produces codes from inputs. When the representation changes, an address is *valid* for an input x if the code used to retrieve memory is unchanged, $\mathrm{TopK}(h_{\mathrm{new}}(x)) = \mathrm{TopK}(h_{\mathrm{old}}(x))$, so that the same prototype is selected and the same expert receives the same input. Address migration seeks a map G, estimated only from current unlabeled inputs, such that Gh_{new} keeps addresses valid for the inputs that past memories serve. This section characterizes when a linear G can do so and how its error translates into changes of the code.

Let $b(x) \in \mathbb{R}^n$ denote the activation used by an existing memory and $a(x) \in \mathbb{R}^n$ the activation after a representation update. Migration seeks G with $Ga(x) \approx b(x)$ on the inputs for which memory should be preserved. With $Q = \mathbb{E}[aa^\top]$ and $C = \mathbb{E}[ba^\top]$, the population ridge solution is

$$G_\gamma = C(Q + \gamma I)^{-1}, \qquad \gamma > 0. \tag{1}$$

The minimum-norm unregularized solution is $G_0 = CQ^\dagger$. In general, this is a distribution-dependent least-squares map, not the ordinary pseudoinverse of a forward drift operator.

Recoverability under exact linear drift. Suppose $a = Tb$ and $\mathbb{E}[bb^\top] \succ 0$. A linear memory score $w^\top b$ can be recovered from a by a linear map if and only if $w \in \mathrm{range}(T^\top)$, equivalently $w \perp \ker T$ (Proposition 2). Recovering all activation coordinates requires T to be injective. For sparse codes, score recovery alone is insufficient: preserving the full pre-binarization vector preserves top-k exactly, whereas approximate recovery requires a margin argument (Appendix E).

Non-invertible drift. Although both projections are linear in the frontend features, one projection need not be a function of the other. On a feature subspace V, an exact map from $P_{\mathrm{new}}z$ to $M_{t-1}P_{\mathrm{old}}z$ exists precisely when

$$\ker(P_{\mathrm{new}}|_V) \subseteq \ker(M_{t-1}P_{\mathrm{old}}|_V). \tag{2}$$

Thus, compression can discard information that migration cannot recover. The measured residual reflects this approximation error as well as regularization and estimation error; it cannot be identified with nonlinearity alone.

Current-stream fit versus historical preservation. The regression objective is evaluated under the current unlabeled distribution. A small residual there does not by itself bound error on old classes. Such a transfer requires a shared relation over historical support and sufficient coverage of the directions needed to identify it. Appendix A gives an explicit coverage counterexample, a finite-sample bound that retains ridge bias and population error, and a restricted lower bound for drift-blind readouts. Together, these results give the conditions under which addresses are preserved.

From alignment error to code changes. Address validity is a statement about top-k sets, whereas the migration objective controls a squared activation error. The following bound connects the two.

Proposition 1. *Let b be the activation used by the memory and $\widetilde{b} = b + \delta$ the aligned activation after an update, with top-k sets S and $\widetilde{S}$ under a fixed tie rule. Let $\tau = (b_{(k)} + b_{(k+1)})/2$ and $m_j = |b_j - \tau|$ be each coordinate's distance to the selection threshold. Then $\widetilde{S} = S$ whenever $\|\delta\|_\infty < (b_{(k)} - b_{(k+1)})/2$, and in general the fraction of changed code bits is at most $\frac{2}{n}\sum_{j=1}^{n} \mathbf{1}[|\delta_j| \geq m_j]$.*

The proof is in Appendix E. The bound makes code changes a function of how the error is distributed relative to each coordinate's margin, not of its overall size. A residual that is large in norm can leave most bits intact when its mass falls on coordinates far from the threshold, and a small residual concentrated near the threshold can change routing. In the three-phase SSL-1000 analysis the bound evaluates to 0.080–0.116 against an observed flip fraction of 0.018–0.026, and flipped bits have median margins of 0.034–0.056 compared with 0.66–0.74 for all bits (Appendix E). Changes therefore concentrate at the selection boundary, which is why the experiments report bit flips and routing changes alongside activation residuals.

4 METHOD

4.1 SPARSE CODES AND ROUTED EXPERTS

All activations are column vectors. A frozen frontend $\Phi : \mathbb{R}^{d_0} \to \mathbb{R}^d$ feeds a trainable projection $P_t \in \mathbb{R}^{n \times d}$. A persistent map $M_t \in \mathbb{R}^{n \times n}$ aligns its output to the coordinates used by the memory:

$$h_t(x) = M_t P_t \Phi(x), \qquad c_t(x) = \mathrm{TopK}(h_t(x), k) \in \{0,1\}^n. \tag{3}$$

Here, TopK selects exactly k coordinates, with a fixed tie-breaking rule, and $M_0 = I$; projection followed by top-k winner-take-all is a standard way to form sparse binary similarity codes (Dasgupta et al., 2017). The map remains active between sleep phases. Each leaf j has a prototype p_j, updated by an exponential moving average of assigned codes, $p_j \leftarrow (1-\alpha)p_j + \alpha c_t(x)$. Routing uses cosine similarity with a load constraint; the selected expert outputs logits for the shared class vocabulary:

$$j^*(x) = \mathrm{Route}(c_t(x); \{p_j\}), \qquad \hat{y}(x) = \operatorname*{arg\,max}_{y \in \{1,\ldots,C\}} [f_{j^*(x)}(c_t(x))]_y. \tag{4}$$

Exactly one local expert answers each query. All experts predict over the same classes, so this is a single prediction problem without a task oracle, but it does not use one shared parameterized classification head. Leaves are allocated within a maximum routing depth. Appendix B gives the architecture and implementation details, and Algorithm 1 gives the full cycle.

Relation to nearest-class-mean classification. The routed memory contains a familiar exemplar-free classifier as a special case, which clarifies what its additional components are for. If each class is assigned a single leaf whose prototype is the mean code of that class's training inputs, each expert predicts its own class for every input, routing selects the prototype of highest cosine similarity without a load constraint, and the projection is frozen, then Eq. (4) reduces to $\arg\max_c \cos(c(x), p_c)$: a cosine nearest-class-mean classifier in code space. The routed memory departs from this corner along two axes: *capacity*, through many leaves per class whose local experts learn within-leaf decision rules, and *mobility*, through projection updates made safe by migration. A comparison with nearest-class-mean classification on the same frozen features therefore measures what these two axes add beyond class means.

4.2 WAKE, SLEEP, AND MIGRATION

Wake. The frontend, projection, and alignment map are frozen. Only the selected experts and routing prototypes update on the current labeled task. Unassigned experts receive no gradient.

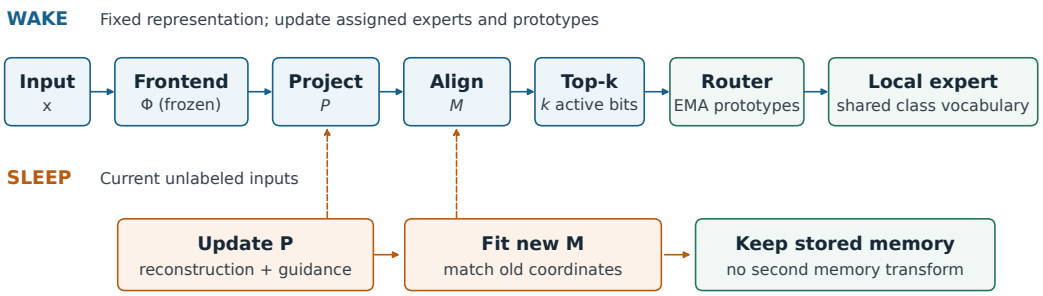

Figure 1: Wake–sleep cycle with persistent alignment. During wake, the projection and alignment are fixed while routed experts and prototypes learn. Sleep updates the projection on current unlabeled inputs; the fitted map then returns new activations to the existing memory coordinates before top-k. Memory objects are not transformed a second time.

Selective routing restricts which memories are modified, but does not prevent interference when tasks share leaves or prototypes move.

Sleep. At a task boundary, labels are withheld and the projection is updated on current-task inputs. A temporary linear decoder D minimizes

$$\mathcal{L}_{\text{rec}}(P, D) = \mathbb{E}_{x \sim \mathcal{U}_t} \|DP\Phi(x) - \Phi(x)\|_2^2. \tag{5}$$

The frontend and routed memory remain fixed. The decoder is discarded after sleep. This objective changes the compression of existing features; improving reconstruction need not improve classification. Using several passes over the current task is allowed by the protocol, while retaining its examples after the boundary is not.

Migration. After the sleep update, evaluate both projections on m current unlabeled inputs:

$$a_i = P_{\text{new}}\Phi(u_i), \qquad b_i = M_{t-1}P_{\text{old}}\Phi(u_i), \qquad A_{i,:} = a_i^\top, \quad B_{i,:} = b_i^\top. \tag{6}$$

With $A, B \in \mathbb{R}^{m \times n}$, solve

$$M_t = \arg\min_M \frac{1}{m}\|AM^\top - B\|_F^2 + \gamma\|M\|_F^2, \qquad M_t^\top = (A^\top A + m\gamma I)^{-1}A^\top B. \tag{7}$$

Use a linear solve in practice. Since B already contains M_{t-1}, Equation (7) *replaces* the map; it must not be composed with M_{t-1} again. At inference, Equation (3) aligns activations before binarization. Prototypes and experts remain in their existing coordinates. This convention avoids also transforming stored objects, which would apply the correction twice. A linear functional could instead be transformed by $w \leftarrow M_t^\top w$ in a purely linear memory, but that alternative does not generally commute with top-k or cosine routing.

Only the paired second moments $A^\top A$ and $A^\top B$ are needed; they can be accumulated over minibatches with the old and new encoders fixed. The dense alignment fit costs $O(mdn + mn^2 + n^3)$ after frontend computation and uses $O(n^2)$ sufficient-statistic storage. Applying M_t costs $O(n^2)$ per input, or the product M_tP_t can be cached between updates. The old projection and temporary statistics can be discarded once the map is fitted. These costs exclude expert storage, frontend evaluation, sleep optimization, and routing. The persistent state that migration adds is the $n \times n$ map, 65,536 parameters at $n = 256$, which is small beside the expert memory given in Section 5.1.

4.3 Geometry-guided sleep

Migration corrects activations after a projection update. Guided sleep additionally discourages changes to pairwise geometry during the update. For rows $H_{\text{old},i,:} = (P_{\text{old}}\Phi(u_i))^\top$ and $H_{\text{new},i,:} = (P\Phi(u_i))^\top$ on the same current minibatch, define

$$\mathcal{L}_{\text{sleep}} = \mathcal{L}_{\text{rec}} + \beta\left[1 - \rho\big(\text{pdist}(H_{\text{new}}), \text{pdist}(H_{\text{old}})\big)\right]. \tag{8}$$

Here, pdist collects Euclidean pairwise distances and ρ is Pearson correlation; the old activations are stop-gradient targets. The ridge penalty γ and guidance weight β are distinct hyperparameters. Correlation requires nonconstant distance vectors.

The penalty is invariant to translations, rotations, and positive uniform scaling of a point cloud. It is *not* invariant to every invertible linear map: anisotropic scaling and shear can be penalized even when migration can undo them. Conversely, a linear map without a bias cannot generally undo a translation. Guidance is therefore a geometric regularizer, not an exact test of migration recoverability. It adds no persistent historical examples, but does require the old projection and paired current activations during sleep. Computing all distances on a batch of size b costs $O(b^2 n)$. Appendix D discusses a residual-based alternative and its in-sample fitting pitfall.

5 Experiments

5.1 Protocol

Data access. Split CIFAR-100 (Krizhevsky, 2009) comprises 10 tasks of 10 disjoint classes with a per-seed random class order. Evaluation predicts among 100 classes without a task oracle. Task boundaries are available during training. No examples or feature exemplars from completed tasks are retained for subsequent training. However, the SSL frontend is trained beforehand on the *entire* CIFAR-100 training split without labels; it therefore has access to future classes' input distribution. This pretraining access is distinct from the later exemplar-free stream. A 20×5 split keeps the same 100 classes while changing task granularity and the number of sleep phases. Domain-incremental MNIST provides an exploratory editing comparison.

Frontends and capacity. The frozen frontends are a random CNN (1024 dimensions), a Coates-style k-means patch encoder (512 dimensions) (Coates & Ng, 2012), and a CIFAR ResNet-18 trained with a SimCLR-style objective for 200 or 1000 epochs (SSL-200/SSL-1000; 512 dimensions) (He et al., 2016; Chen et al., 2020). No external pretraining data are used. The projection has $n = 256$ dimensions with $k = 16$ active bits. Routing depth $D \in \{7, 9, 11\}$ permits up to 2^D experts. At depth 11, up to approximately 182M expert parameters (2048 experts) are stored, compared with approximately 0.58M parameters in the dense MLP baseline. Only one expert and the projection participate in a prediction (approximately 0.22M parameters, excluding routing, alignment, and the frontend).

Configurations and metrics. **A** freezes the projection; **B** updates it during sleep without migration; **C** adds migration; **C**$+g_1$ adds guidance with $\beta = 10$; **E** develops the projection before the task sequence. Reference methods are a dense MLP (**N**), MLP with EWC (**W**), and joint training (**J**). Mobile configurations incur additional sleep and migration costs. We report final average accuracy (avg), mean forgetting (F), and normalized task–leaf mutual information (TLI), with definitions in Appendix B. Task labels are used to measure TLI, not supplied to the predictor. A, C, and C$+g_1$ use three seeds at the reference configuration; baseline and exploratory seed counts are marked separately. Initialization and class order vary jointly.

5.2 Guided migration

Table 1 compares the configurations. A obtains 0.281 ± 0.010 final accuracy, or 51.7% of the joint-training reference (0.544). C decreases accuracy by 4.2 percentage points relative to A. Adding guidance increases C's accuracy by 3.4 points and leaves a 0.8-point gap to A: it recovers about 81% of the observed gap.

The dense MLP and EWC references have substantially lower final accuracy. The nearest-class-mean correspondence of Section 4 places cosine nearest-class-mean classification in code space inside the same architecture as its single-leaf, frozen limit.

5.3 Routing selectivity

At depth 7, replacing the frontend (random, k-means, SSL-200, SSL-1000) increases TLI from 0.081 to 0.098, 0.196, and 0.231, while final accuracy increases from 0.034 to 0.042, 0.130, and 0.135.

Table 1: Split CIFAR-100 (10×10). Main rows use SSL-1000 features; A/C use depth 11. Accuracy and forgetting are fractions. Three-seed results are mean $\pm$ s.d.; † denotes one seed. EWC with $\lambda_{\text{EWC}} = 10^6$ uses SSL-200 features and is listed separately.

Method	Update / memory policy	$F \downarrow$	avg $\uparrow$	TLI
N†	Dense MLP	0.740	0.081	—
W†	EWC, $\lambda_{\text{EWC}} = 10^3$	0.744	0.081	—
J†	Joint training	—	0.544	—
A	Frozen projection	0.225 ± 0.021	0.281 ± 0.010	0.671 ± 0.009
C	Sleep + migration	0.264 ± 0.018	0.239 ± 0.001	0.562 ± 0.004
C+g_1	Guided sleep + migration	0.243 ± 0.021	0.273 ± 0.011	0.618 ± 0.012
Different frontend (SSL-200)				
W†	EWC, $\lambda_{\text{EWC}} = 10^6$	0.676	0.076	—

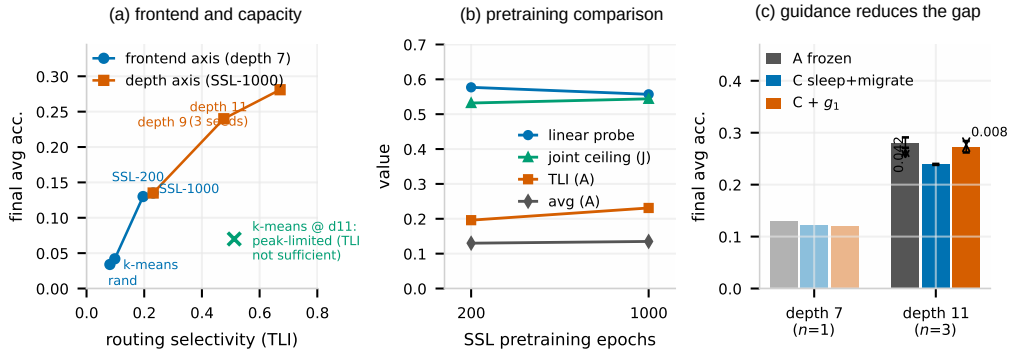

Figure 2: Frontend and capacity comparisons for A. *(a)* TLI and accuracy increase along both tested axes, but the k-means/depth-11 point shows that selectivity is not sufficient. *(b)* The two SSL pretraining durations produce similar downstream accuracy. *(c)* Guidance reduces the C–A gap at depth 11. Depth-7 guidance uses one seed; depth-11 bars show three-seed standard deviations.

Holding SSL-1000 fixed and increasing depth from 7 to 9 to 11 increases TLI from 0.231 to 0.476 to 0.671 and accuracy from 0.135 to 0.240 to 0.281 (Figure 2; Appendix I). Additional routing capacity therefore improves both selectivity and accuracy in this architecture.

TLI alone is not a predictor of accuracy across all frontends. K-means at depth 11 reaches TLI 0.512 but only 0.070 accuracy; its mean accuracy immediately after task learning is 0.232, compared with 0.493 for SSL-1000. Nor is linear-probe accuracy sufficient to rank downstream continual performance: extending SSL pretraining from 200 to 1000 epochs changes the probe from 0.577 to 0.557, while depth-7 continual accuracy changes from 0.130 to 0.135. Class separability and usefulness for selective routing are therefore distinct properties of a representation.

For interpretation, let acquisition accuracy be the mean accuracy immediately after each task is learned, and define retention as final accuracy divided by acquisition accuracy. Then $\text{avg} = \text{acquisition} \times \text{retention}$. SSL-1000 at depths 7 and 11 has acquisition/retention pairs $0.576/0.234$ and $0.493/0.570$; k-means at depth 11 has $0.232/0.302$. Capacity therefore changes acquisition as well as retention.

5.4 Migration residuals

At depth 7 with SSL-200, A and C have similar mean accuracy (0.130 ± 0.008 and 0.129 ± 0.005). At depth 11 with SSL-1000, the gap is larger (Table 1). Sensitivity to migration is thus larger in the deeper, higher-capacity configuration, where experts serve narrower subsets of the stream (Figure 2c).

The depth-11 phase diagnostics (seed 0; Figure 3) show residuals decreasing from approximately 0.21 to 0.08 across sleep phases, with a mean of 0.121, and code bit-flip rates of approximately 1–2%.

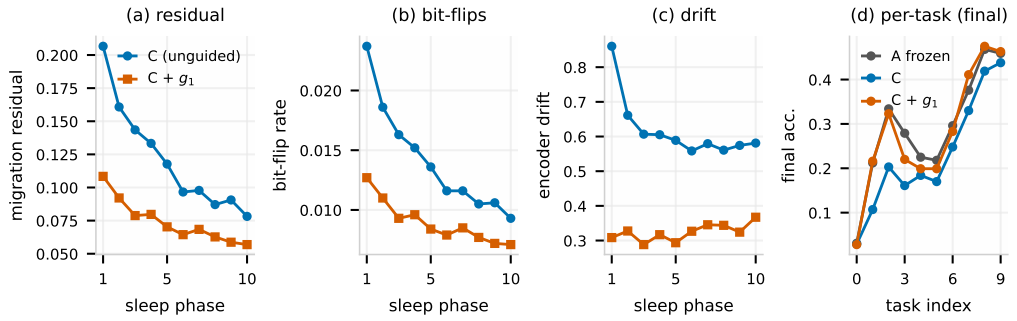

Figure 3: Depth-11 diagnostics, seed 0. *(a)–(c)* Migration residual, all-bit flip fraction, and projection drift with and without guidance. *(d)* Final per-task accuracy. Guidance lowers all three quantities and improves several intermediate tasks.

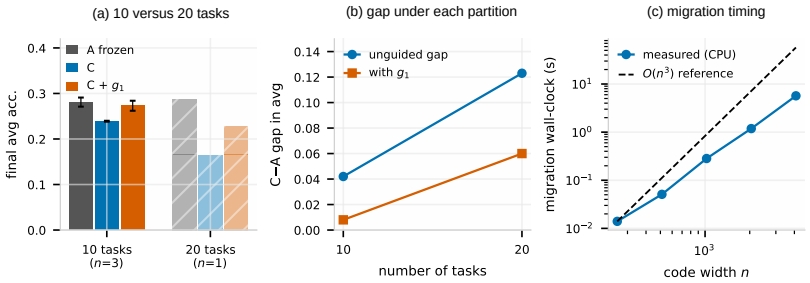

Figure 4: *(a)* Final accuracy for 10 versus 20 tasks (depth 11, SSL-1000; three seeds versus one). *(b)* The gap to A is larger for the finer task partition and roughly halves under guidance. *(c)* CPU migration timing versus code width with a cubic reference.

Guidance reduces the mean projection drift from 0.62 to 0.32, the migration residual by 39%, and bit flips by 37%. Guidance thus reduces drift, residual, and code changes together, consistent with fewer retrieval perturbations. The wake/sleep attribution assigns approximately 90% of task-0 loss to wake and 10% to sleep across seeds; guided sleep reduces the net sleep share to approximately 2%, so after guidance nearly all remaining loss on task 0 occurs during supervised wake phases.

More tasks at fixed class count. The single-seed 20×5 experiment gives A accuracy 0.287, compared with the three-seed 10×10 mean of 0.281, and TLI 0.651. The C–A gap grows to 0.123 and the guided gap is 0.060 (Figure 4). Guidance again roughly halves the gap, and the larger remaining gap in the longer stream points to accumulated migration error as the quantity to control.

5.5 Ablations and Memory Editing

The SSL migration operators have broad singular spectra (top-1 energy at most 2.4%), whereas the random-frontend operator concentrates 98–100% of its singular energy in its leading direction (Figure 6; Appendix I). The margin analysis in Appendix E explains how a moderate activation residual can coexist with a small fraction of changed bits. It also distinguishes bit flips from changes in active-code membership, routing, and predictions.

At depth 7/SSL-200, B (sleep without migration) has accuracy 0.097, below C's 0.129 mean: migration recovers the accuracy lost to the sleep update. The developmental variant E achieves 0.093, so a whitening-based pre-task update does not substitute for migration (Table 4).

Table 2: Leaf-removal interventions. Selectivity is target accuracy drop divided by mean non-target drop. Coverage is the fraction of target-task routing mass on the removed leaves, given qualitatively for MNIST.

Setting	Target	Coverage	Selectivity
Domain-MNIST (A)	Most concentrated domain	High	$311\times$
Domain-MNIST (A)	Median domain	Moderate	$3.1\times$
Domain-MNIST (C)	Median, post-migration	Moderate	$3.6\times$
Split CIFAR-100 (A/C)	Task 2	0.41–0.45	0.0–$0.6\times$

Finally, zeroing task-associated leaves probes whether retrieval concentration permits selective functional edits. The target/non-target drop ratios are $311\times$ for the most concentrated MNIST domain, $3.1\times$ for a median domain, and $3.6\times$ after migration. For the evaluated CIFAR-100 task, ratios are 0.0–$0.6\times$ with coverage 0.41–0.45 (Table 2). Selective removal is therefore possible when retrieval is concentrated, as in the MNIST domains.

6 Discussion and Limitations

Address migration lets a projection change without invalidating the coordinates of a sparse routed memory. With guided sleep, migration recovers about 80% of the accuracy cost of moving the projection in the 10-task setting, and the margin analysis explains why moderate activation residuals change few code bits. Because migration needs only the paired second moments of old and new activations on current inputs, it stores no examples and its per-phase cost does not grow with the number of completed tasks. The margin analysis also shows where alignment error matters: flipped bits have median margins of 0.034–0.056 against 0.66–0.74 for all bits, so preserving addresses amounts to controlling error on the coordinates near the selection threshold. The same framework identifies the next target: in longer streams, accumulated migration error is the quantity that determines whether a mobile representation can surpass a frozen one.

Data coverage and representation scope. Current-class alignment need not transfer to historical classes; adversarial drift estimation targets this gap by moving current samples toward old prototypes (Goswami et al., 2024). Our guarantees make the requirement explicit: a recoverable relation and coverage of the relevant directions, neither of which follows from a low current-batch residual. Updating the frontend itself, as in continual self-supervised learning (Fini et al., 2022), would introduce a broader class of drift than the linear projections studied here; learned drift compensation has been applied to such moving backbones (Gomez-Villa et al., 2024).

Limitations. The frontend is pretrained on all classes without labels and only the projection above it moves, which isolates coordinate drift and leaves frontend adaptation and genuinely new domains for future work. Exemplar-free methods that keep the representation fixed, such as nearest-class-mean classification (Zhou et al., 2025), FeCAM (Goswami et al., 2023), RanPAC (McDonnell et al., 2023), and FeTrIL (Petit et al., 2023), avoid coordinate drift by construction, and drift compensation (Yu et al., 2020; Goswami et al., 2024; Gomez-Villa et al., 2024) corrects stored prototypes; address migration addresses the complementary problem of keeping routed memories valid when the representation itself is updated.

Conclusion. A coordinate map estimated from current unlabeled inputs reduces incompatibility between an updated projection and existing sparse memories, and its success is governed by recoverable directions, distribution coverage, and the margins of the top-k code. Migration with geometric guidance is the most effective mobile configuration we tested.

Ethics Statement

The experiments use the public CIFAR-100 and MNIST benchmarks and involve no human participants. Avoiding the storage of past examples reduces one form of data retention, but model parameters and prototypes can still retain information about training data, and the method provides no

formal privacy guarantee. Selective accuracy loss after removing experts is a functional intervention, distinct from certified machine unlearning or compliance with a deletion request.

Reproducibility Statement

The main text states the assumptions of the analysis, specifies the architecture and the wake, sleep, and migration updates, and reports the data access of every configuration, including all-class unlabeled pretraining, and the number of seeds behind each result. The appendix gives complete proofs, pseudocode for the full cycle, and the architecture settings, implementation details, metric definitions, and evaluation protocol.

AI Use Statement

We used generative AI tools to assist with restructuring and editing the manuscript, searching for and verifying related literature, and preparing the LaTeX source. AI tools also assisted in revising and writing up the mathematical statements and proofs, including the top-k margin bound and the nearest-class-mean correspondence; the authors checked every derivation. We take responsibility for the final content of this paper, including text, claims, and proofs produced with the aid of generative AI.

References

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert Gate: Lifelong Learning with a Network of Experts. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

Trenton Bricken, Xander Davies, Deepak Singh, Dmitry Krotov, and Gabriel Kreiman. Sparse Distributed Memory is a Continual Learner. In *International Conference on Learning Representations*, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*, 2020.

Adam Coates and Andrew Y. Ng. Learning Feature Representations with K-Means. In *Neural Networks: Tricks of the Trade (2nd ed.)*, pp. 561–580. Springer, 2012.

Sanjoy Dasgupta, Charles F. Stevens, and Saket Navlakha. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796, 2017.

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.

Enrico Fini, Victor G. Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-Supervised Models are Continual Learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Robert M. French. Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Ryan Golden, Jean Erik Delanois, Pavel Sanda, and Maxim Bazhenov. Sleep prevents catastrophic forgetting in spiking neural networks by forming a joint synaptic weight representation. *PLoS Computational Biology*, 18(11):e1010628, 2022.

Alex Gomez-Villa, Dipam Goswami, Kai Wang, Andrew D. Bagdanov, Bartlomiej Twardowski, and Joost van de Weijer. Exemplar-Free Continual Representation Learning via Learnable Drift Compensation. In *European Conference on Computer Vision*, 2024.

Oscar C. González, Yury Sokolov, Giri P. Krishnan, Jean Erik Delanois, and Maxim Bazhenov. Can Sleep Protect Memories from Catastrophic Forgetting? *eLife*, 9:e51005, 2020.

Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost van de Weijer. FeCAM: Exploiting the Heterogeneity of Class Distributions in Exemplar-Free Continual Learning. In *Advances in Neural Information Processing Systems*, 2023.

Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Bartłomiej Twardowski, and Joost van de Weijer. Resurrecting Old Classes with New Data for Exemplar-Free Continual Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Md Yousuf Harun, Jhair Gallardo, Tyler L. Hayes, Ronald Kemker, and Christopher Kanan. SIESTA: Efficient Online Continual Learning with Sleep. *Transactions on Machine Learning Research*, 2023.

Jeff Hawkins and Subutai Ahmad. Why Neurons Have Thousands of Synapses, A Theory of Sequence Memory in Neocortex. *Frontiers in Neural Circuits*, 10:23, 2016.

Tyler L. Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. REMIND Your Neural Network to Prevent Catastrophic Forgetting. In *European Conference on Computer Vision*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

Pentti Kanerva. *Sparse Distributed Memory*. MIT Press, 1988.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

James Kirkpatrick, Razvan Pascanu, Neil C. 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.

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`.

Simone Magistri, Tomaso Trinci, Albin Soutif-Cormerais, Joost van de Weijer, and Andrew D. Bagdanov. Elastic Feature Consolidation for Cold Start Exemplar-Free Incremental Learning. In *International Conference on Learning Representations*, 2024.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why There Are Complementary Learning Systems in the Hippocampus and Neocortex: Insights from the Successes and Failures of Connectionist Models of Learning and Memory. *Psychological Review*, 102(3): 419–457, 1995.

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.

Mark D. McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. RanPAC: Random Projections and Pre-trained Models for Continual Learning. In *Advances in Neural Information Processing Systems*, 2023.

Aristeidis Panos, Yuriko Kobe, Daniel Olmeda Reino, Rahaf Aljundi, and Richard E. Turner. First Session Adaptation: A Strong Replay-Free Baseline for Class-Incremental Learning. In *IEEE/CVF International Conference on Computer Vision*, 2023.

German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent Replay for Real-Time Continual Learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023.

Vivek Ramanujan, Pavan Kumar Anasosalu Vasu, Ali Farhadi, Oncel Tuzel, and Hadi Pouransari. Forward Compatible Training for Large-Scale Embedding Retrieval Systems. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. In *Advances in Neural Information Processing Systems*, 2019.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv preprint arXiv:1606.04671*, 2016.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient Projection Memory for Continual Learning. In *International Conference on Learning Representations*, 2021.

Yantao Shen, Yuanjun Xiong, Wei Xia, and Stefano Soatto. Towards Backward-Compatible Representation Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems*, 2017.

Timothy Tadros, Giri P. Krishnan, Ramyaa Ramyaa, and Maxim Bazhenov. Sleep-like Unsupervised Replay Reduces Catastrophic Forgetting in Artificial Neural Networks. *Nature Communications*, 13:7742, 2022.

Gido M. van de Ven, Hava T. Siegelmann, and Andreas S. Tolias. Brain-inspired Replay for Continual Learning with Artificial Neural Networks. *Nature Communications*, 11:4069, 2020.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training Networks in Null Space of Feature Covariance for Continual Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer G. Dy, and Tomas Pfister. DualPrompt: Complementary Prompting for Rehearsal-Free Continual Learning. In *European Conference on Computer Vision*, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer G. Dy, and Tomas Pfister. Learning to Prompt for Continual Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022b.

Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic Drift Compensation for Class-Incremental Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

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

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, 2017.

Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. SLCA: Slow Learner with Classifier Alignment for Continual Learning on a Pre-trained Model. In *IEEE/CVF International Conference on Computer Vision*, 2023.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual Learning with Pre-Trained Models: A Survey. In *International Joint Conference on Artificial Intelligence*, 2024a.

Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable Subspace Ensemble for Pre-Trained Model-Based Class-Incremental Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024b.

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-Incremental Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9851–9873, 2024c.

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting Class-Incremental Learning with Pre-Trained Models: Generalizability and Adaptivity are All You Need. *International Journal of Computer Vision*, 133(3):1012–1032, 2025.

Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype Augmentation and Self-Supervision for Incremental Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

A Analysis of Address Migration

A.1 Population regression and recoverability

Let $a, b \in \mathbb{R}^n$ be paired new and old memory-coordinate activations on a distribution μ, with finite second moments. Write $Q = \mathbb{E}[aa^\top]$, $C = \mathbb{E}[ba^\top]$, and $\Sigma = \mathbb{E}[bb^\top]$. No centering or intercept is assumed. Minimizing $\mathbb{E}\|Ga - b\|_2^2 + \gamma\|G\|_F^2$ gives $G_\gamma = C(Q + \gamma I)^{-1}$. At $\gamma = 0$, $G_0 = CQ^\dagger$ is the minimum-Frobenius-norm solution. The residual second-moment matrix is

$$\mathbb{E}[(G_0 a - b)(G_0 a - b)^\top] = \Sigma - CQ^\dagger C^\top. \tag{9}$$

To see this, expand the square and use $C = CQ^\dagger Q$: any $v \in \ker Q$ satisfies $v^\top a = 0$ almost surely and hence $Cv = 0$. Equation (9) includes any error that remains after optimal linear prediction. Finite samples cannot remove it.

Proposition 2 (Recoverable linear scores)**.** *Suppose $a = Tb$ almost surely and $\Sigma \succ 0$, allowing T and $Q = T\Sigma T^\top$ to be singular. Let S be the span of a finite collection of score vectors w_j. There exists a linear map G satisfying $w_j^\top Ga = w_j^\top b$ almost surely for every j if and only if*

$$S \subseteq \operatorname{range}(T^\top), \quad \textit{equivalently } S \perp \ker T.$$

The population least-squares map $G_0 = \Sigma T^\top (T\Sigma T^\top)^\dagger$ recovers every such score exactly. It need not equal $T^\dagger$ when T is singular and $\Sigma \neq I$. If T is invertible, $G_0 = T^{-1}$.

Proof. For a fixed score, equality implies $\mathbb{E}[((T^\top G^\top w - w)^\top b)^2] = 0$. Positive definiteness of Σ gives $T^\top G^\top w = w$, which requires $w \in \operatorname{range}(T^\top)$. Conversely, if $w = T^\top v$, then $w^\top b = v^\top a$, so this score is exactly linearly predictable. The normal equations give $G_0 Q = C$. Thus the vector residual $r = G_0 a - b$ is orthogonal in L^2 to every linear function of a. In particular, $w^\top r$ is orthogonal to all such functions. But $w^\top r = (G_0^\top w - v)^\top a$ is itself a linear function of a, so its squared norm is zero. This applies simultaneously to all scores. The displayed formula follows by substitution into $CQ^\dagger$; for invertible T it reduces to T^{-1}. □

Scope of the proposition. The statement concerns linear scores before binarization. A prototype compared with a binary code by cosine similarity is not simply a linear functional of the pre-binarization activation. For the implemented pipeline, full activation recovery ensures code preservation; approximate recovery is treated in Appendix E. Preserving a selected score span alone need not preserve the routed classifier.

Noisy or nonlinear drift. If $a = Tb + \eta$ with $\mathbb{E}[\eta b^\top] = 0$, then

$$G_0 = \Sigma T^\top (T\Sigma T^\top + \Sigma_\eta)^\dagger, \qquad \Sigma_\eta = \mathbb{E}[\eta\eta^\top].$$

For any fitted $\widehat{G}$, the total error decomposes as

$$\widehat{G}a - b = (\widehat{G} - G_0)a + (G_0 a - b). \tag{10}$$

The second term is a population approximation residual. It does not shrink merely because the calibration sample grows, and it cannot be absorbed into a vanishing estimation term. Orthogonality of η is an additional modeling assumption, not a consequence of calling it a nonlinear residual. The general regression formulas above do not require that assumption.

A.2 Factorization and support coverage

For $z \in V$, define $A_V z = P_{\text{new}} z$ and $B_V z = M_{t-1} P_{\text{old}} z$. There exists G with $GA_V = B_V$ if and only if $\ker A_V \subseteq \ker B_V$. Necessity follows by applying both sides to a vector in $\ker A_V$. For sufficiency, define $G(A_V z) = B_V z$ on $\operatorname{range}(A_V)$. The kernel inclusion makes this well-defined; extend it linearly to the remaining output space. This proves Equation (2).

When this inclusion holds over the full frontend space, the known weights also provide the algebraic candidate $G = M_{t-1} P_{\text{old}} P_{\text{new}}^\dagger$. Without the inclusion, this candidate is a feature-space least-squares construction and need not agree with the data-weighted activation regression.

Current-task calibration can leave historical directions unidentified. For example, suppose calibration activations are supported on $\mathrm{span}\{e_1\}$ with $b = a$. Both $G_1 = I$ and $G_2 = \mathrm{diag}(1, -1)$ fit perfectly in two dimensions, yet give opposite outputs at the historical activation $a = e_2$. More current samples on the same line do not resolve the ambiguity. A shared exact linear relation transfers to a historical activation subspace when that subspace is contained in the span identified by calibration and its second moment is nondegenerate there. For approximate relations, transferring an error bound additionally requires control of distribution shift.

A.3 Finite-sample estimation with ridge bias

The following result separates estimation from approximation. It applies to fixed encoders and independent calibration samples; when sleep adapts an encoder to the same data, these assumptions need separate justification or fresh calibration data.

Proposition 3 (A perturbation bound for the fitted map)**.** *Assume $Q \succeq \kappa I$ with $\kappa > 0$. Define $\widehat{Q} = A^\top A/N$, $\widehat{C} = B^\top A/N$, and $\widehat{G} = \widehat{C}(\widehat{Q} + \gamma I)^{-1}$, with $\gamma \geq 0$. If*

$$\|\widehat{Q} - Q\|_2 \leq \epsilon_Q < \kappa, \qquad \|\widehat{C} - C\|_2 \leq \epsilon_C,$$

then, with $G_0 = CQ^{-1}$,

$$\|\widehat{G} - G_0\|_2 \leq \frac{\epsilon_C + \|G_0\|_2(\epsilon_Q + \gamma)}{\kappa + \gamma - \epsilon_Q} =: \varepsilon. \tag{11}$$

If the paired vectors (a, b) are independent, identically distributed, and uniformly sub-Gaussian in every unit direction with norm at most K, then with probability at least $1 - \delta$,

$$\max(\epsilon_Q, \epsilon_C) \leq cK^2 \left(\sqrt{\frac{2n + \log(2/\delta)}{N}} + \frac{2n + \log(2/\delta)}{N} \right) \tag{12}$$

for a universal constant c. This yields an estimation rate when the right-hand side is smaller than κ, but retains bias from $\gamma > 0$.

Proof. Using $C = G_0 Q$, subtract the population normal equations to obtain

$$\widehat{G} - G_0 = \left[(\widehat{C} - C) - G_0(\widehat{Q} - Q) - \gamma G_0 \right] (\widehat{Q} + \gamma I)^{-1}.$$

The inverse has operator norm at most $(\kappa + \gamma - \epsilon_Q)^{-1}$ by the eigenvalue perturbation inequality. Submultiplicativity gives Equation (11). Applying sub-Gaussian second-moment concentration to the joint $2n$-dimensional vector (a, b) bounds its diagonal and off-diagonal blocks and gives Equation (12) (Vershynin, 2018). This also makes explicit why controlling new activations alone is insufficient: the cross moment includes the old activations. □

For singular Q, the deterministic argument can be applied on an identified support subspace with a positive minimum eigenvalue. It does not provide information about directions outside that support. Ridge stabilization makes a numerical inverse possible but does not create identifiability.

Consequences for top-k preservation. Let $r_0(x) = G_0 a(x) - b(x)$ and let $\Delta(b)$ denote the difference between the kth and $(k+1)$th largest entries of b. On the estimation event of Proposition 3, for fixed thresholds $r, s \geq 0$ and an independent evaluation input,

$$\begin{aligned} \Pr(\mathrm{TopK}(\widehat{G}a, k) \neq \mathrm{TopK}(b, k)) \leq{}& \Pr(\|r_0\|_\infty > r) + \Pr(\|a\|_2 > s) \\ &+ \Pr(\Delta(b) \leq 2(r + \varepsilon s)). \end{aligned} \tag{13}$$

Indeed, outside these three events, Equation (10) gives $\|\widehat{G}a - b\|_\infty \leq r + \varepsilon s < \Delta(b)/2$. The unconditional probability over both calibration and evaluation adds at most δ. The bound thus isolates the three quantities that govern top-k preservation: the population residual, the activation scale, and the margin distribution. Increasing N reduces estimation error under the assumptions but does not remove the other terms.

A.4 A LIMITED RESULT FOR DRIFT-BLIND READOUTS

Let $f : \mathbb{R}^n \to \mathbb{R}^q$ be the old readout (e.g., logits), and μ_b the distribution of old activations. Define $\Delta_T = \|f \circ T - f\|_{L^2(\mu_b)}$. A drift-blind replacement $\widehat{f}$ is the same function whether the drift is the identity or T; in particular it has no access to drift-specific weights, correspondences, or metadata.

Proposition 4. *If $T_{\#}\mu_b \ll \mu_b$ and $C_T^2 = \|d(T_{\#}\mu_b)/d\mu_b\|_\infty < \infty$, then*

$$\max\{\|\widehat{f} - f\|_{L^2(\mu_b)}, \|\widehat{f} \circ T - f\|_{L^2(\mu_b)}\} \geq \frac{\Delta_T}{1 + C_T}.$$

Proof. The density-ratio bound implies $\|g \circ T\|_{L^2(\mu_b)} \leq C_T \|g\|_{L^2(\mu_b)}$. The triangle inequality gives

$$\Delta_T \leq \|(f - \widehat{f}) \circ T\| + \|\widehat{f} \circ T - f\| \leq C_T \|f - \widehat{f}\| + \|\widehat{f} \circ T - f\|.$$

Bounding both errors by their maximum proves the result. □

Corollary 1. *Within this two-drift model, a single drift-blind readout cannot be exactly invariant to both choices if $\Delta_T > 0$. Exact invariance requires either restricting the allowable drifts to invariances of f or allowing the replacement to depend on the drift.*

The corollary applies to the specified readout model; known encoder weights or architectural symmetries offer alternatives to paired data. Our learner is drift-dependent because it observes both projections and their paired activations.

B IMPLEMENTATION AND EVALUATION DETAILS

Architecture settings. The code width is $n = 256$ with $k = 16$ active bits. Routing depth is $D \in \{7, 9, 11\}$, prototype EMA rate is $\alpha = 0.1$, and the routing load cap is $(1 + 4.0)$ times mean load. Each local expert has four branches of 64 units, a branch-mean readout, a local 100-class output, and a local decoder, totaling approximately 88.7K parameters. At 2048 experts, this is approximately 181.7M expert parameters, or 727MB for FP32 weights alone (excluding optimizer state, frontend, projection, and routing state). The dense MLP reference has approximately 0.58M parameters. These counts follow from the architecture.

Training settings. Wake training uses three epochs per task and batch size 256, with the leaf learning rate set to three times the shared rate. Sleep uses three epochs and learning rate 10^{-3}. Migration uses $\gamma = 10^{-2}$ and approximately 5000 current unlabeled samples in the 10-task setup. For guidance, $\beta = 10$ is the reference setting; $\beta = 30$ changes final accuracy by less than 0.001.

Frontends. The SSL frontend is a CIFAR ResNet-18 with a 3×3 stem, no initial max-pool, and 512-dimensional global-average-pooled features. Pretraining uses the full CIFAR-100 training split, batch size 512, Adam (Kingma & Ba, 2015) with learning rate 10^{-3} and cosine decay, bf16 autocast, and an NT-Xent temperature of 0.5, for 200 or 1000 epochs. Two-view augmentation follows the SimCLR approach (Chen et al., 2020). The linear-probe accuracies are 0.577 and 0.557. The k-means frontend uses 6×6 patches, ZCA whitening, 256 centroids, bipolar ReLU, and 512-dimensional features (probe 0.347). The random CNN produces 1024 features from 3072 input values with two-level standardization (probe 0.300).

Accuracy and forgetting. For T tasks, let $R_{s,t}$ be test accuracy on task t after the complete training cycle for task s, evaluated over the shared 100-class vocabulary in CIFAR-100. Then

$$\text{avg} = \frac{1}{T}\sum_{t=1}^{T} R_{T,t}, \qquad F = \frac{1}{T-1}\sum_{t=1}^{T-1}\left(\max_{s\in\{t,\ldots,T\}} R_{s,t} - R_{T,t}\right).$$

Acquisition accuracy is $\frac{1}{T}\sum_t R_{t,t}$ and retention is avg divided by this quantity when it is nonzero. Acquisition is not necessarily the maximum accuracy ever achieved on a task; we avoid calling it "peak."

Routing selectivity. Using the final model and labeled evaluation inputs, let $\widehat{p}(t, j)$ be the empirical fraction with task label t routed to leaf j. Define

$$\mathrm{TLI} = \frac{\sum_{t,j} \widehat{p}(t, j) \log[\widehat{p}(t, j)/(\widehat{p}(t)\widehat{p}(j))]}{-\sum_t \widehat{p}(t) \log \widehat{p}(t)},$$

with zero-probability terms omitted. This lies in $[0, 1]$ when task entropy is positive. Task labels are used only for this diagnostic.

Alignment and code changes. For paired matrices in Equation (6), relative migration residual is $\|AM_t^\top - B\|_F / \|B\|_F$ for nonzero B. The bit-flip fraction is $\frac{1}{mn} \sum_i \| \mathrm{TopK}(M_t a_i, k) - \mathrm{TopK}(b_i, k)\|_1$. For $n = 256, k = 16$, at most $32/256 = 12.5\%$ of all bits can differ between two codes. A 2% all-bit flip rate corresponds to approximately 16% of the active set being replaced on average, since replacements equal half the number of changed bits. Neither measure is the router-switch rate.

Editing metrics. Coverage is the fraction of target-task routing mass assigned to the removed leaves before intervention. Let d_t be the target accuracy drop and d_o the mean drop over non-target tasks; selectivity is d_t/d_o, defined for $d_o > 0$ (Table 2). The top-q leaves selected for an edit are distinct from the top-k activation coordinates.

C Binary-Space Migration

An ablation that fits old binary codes from new binary codes gives a relative residual of approximately 0.75, with continual accuracy falling near chance. This motivates fitting the map before binarization. It does not imply that a linear correspondence between binary codes is impossible in general: a fixed permutation of coordinates is itself linear. Rather, top-k decisions can change differently across inputs, so a single map between binary vectors may not capture the observed relation. Similarly, linearity of both projections in frontend features does not by itself guarantee exact continuous-space recovery (Equation (2)).

D Residual-Based Guidance and Fitting Pitfalls

A second variant, g_2, fits W^* from new to old activations and penalizes $\|H_{\mathrm{new}} W^* - H_{\mathrm{old}}\|_F^2$ with W^* treated as stop-gradient. Fitting and evaluating on the same batch can underestimate error. In particular, an unregularized fit can interpolate arbitrary targets when $m \leq n$ and the input matrix has full row rank. With positive ridge regularization the residual is generally nonzero, even when $m = n$; dimension counting alone does not make the objective identically zero.

Accumulating statistics over more samples reduces this interpolation problem, but a clean diagnostic fits the map on separate calibration data and evaluates the residual on a held-out current batch, with encoder parameters fixed during map fitting. Otherwise, accumulated statistics may combine different encoders or include the evaluation batch. A controlled linear/nonlinear residual comparison gives 1.6×10^{-7} versus 385; these values are on a different scale from the relative residuals used elsewhere.

At depth 7, the g_1 and g_2 variants show no consistent downstream gain. Direct activation matching, $\|H_{\mathrm{new}} - H_{\mathrm{old}}\|_F^2$, is a further useful control. It penalizes rotations as well as other changes and approaches freezing only as its weight becomes sufficiently restrictive; a finite penalty does not mathematically force the projection to remain fixed.

E A Deterministic Bound for Top-k Changes

For an old activation vector b, let $\widetilde{b} = b + \delta$ be the aligned new vector. Let S and $\widetilde{S}$ be their top-k sets under a fixed tie rule. Write $\Delta = b_{(k)} - b_{(k+1)}$, where entries are sorted in descending order. If $\|\delta\|_\infty < \Delta/2$, every old selected entry remains above every old unselected entry, and $S = \widetilde{S}$. This is a sufficient, not necessary, condition.

For a finer bound, let $\tau = (b_{(k)} + b_{(k+1)})/2$ and $m_j = |b_j - \tau|$. Then

$$\|\mathbf{1}_{\widetilde{S}} - \mathbf{1}_S\|_1 \le 2\,\#\{j : |\delta_j| \ge m_j\}, \qquad \text{flip fraction} \le \frac{2}{n}\sum_{j=1}^{n} \mathbf{1}[|\delta_j| \ge m_j]. \tag{14}$$

To prove this, pair each displaced old selected coordinate with one newly selected coordinate. In every pair, at least one coordinate must reach or cross the old threshold; otherwise the old selected coordinate remains strictly larger. The pairs are disjoint, so the number of displaced coordinates is at most the number of threshold crossings. Each displacement changes two bits. Using $\ge$ includes boundary and tie cases. Both bounds use the *total* error $\delta = \widehat{G}a - b$, not only its estimation component.

In the three-phase SSL-1000 analysis, both deterministic inequalities hold without violation. The sample-level condition is usually uninformative: median $\|\delta\|_\infty$ is 0.24–0.38 while the median top-k gap is approximately 0.01. The bit-level upper bound is 0.080–0.116, compared with an observed all-bit flip fraction of 0.018–0.026, a factor of approximately 4.4–4.5. Median margins are 0.034–0.056 for flipped bits and 0.66–0.74 for all bits. These diagnostics show that code changes concentrate near the selection boundary.

F Negative and Exploratory Results

The developmental variant E underperforms freezing after pre-task projection training with truncated whitening; the retained rank is 151 of 256 directions. Preliminary learned routers gave low TLI. Unstandardized random features had cosine similarity at least 0.95, motivating the two-level standardization. Finally, the six low-depth guided variants (g_1/g_2 with weights $0.1, 1, 10$) had single-seed accuracies in 0.116–0.126, compared with 0.123 for the corresponding unguided run.

G Evaluation Targets

We set the following evaluation targets and report each against its literal threshold:

- **Migration fidelity:** old-task loss below 0.02 per phase, a task-level criterion distinct from the continuous residual.
- **C versus A:** $\text{avg}_C \ge \text{avg}_A$ and $F_C \le F_A + 0.02$. The accuracy condition is narrowly missed at depth 7 ($0.129 < 0.130$) and at guided depth 11 ($0.273 < 0.281$), and missed without guidance at depth 11 ($0.239 < 0.281$).
- **Routing:** TLI at least 0.25. The depth-7 SSL-200 mean is 0.196; the depth-11 SSL-1000 mean of 0.671 meets the threshold.
- **Forgetting versus dense references:** forgetting is 0.225 for A against 0.740 for the single-seed dense MLP reference.
- **Editing selectivity:** ratio above 3. The MNIST examples meet this threshold; the CIFAR-100 example does not.

H WAKE–SLEEP CYCLE PSEUDOCODE

Algorithm 1 One task cycle with persistent activation alignment

Require: Frozen Φ, projection P, map M, routed memory, current task data.

Wake: fix P, M; compute $c = \mathrm{TopK}(MP\Phi(x), k)$.
Update assigned experts and prototypes on current labeled data only.
Save temporary $P_{\mathrm{old}} \leftarrow P$ and $M_{\mathrm{old}} \leftarrow M$; freeze memory.
Sleep: update P, D on current inputs without labels using
    Equation (5), or Equation (8) for guided sleep.
Freeze P; initialize $S_{aa} = 0$ and $S_{ab} = 0$.
for current calibration minibatches U **do**
    Form rows $A_i = (P\Phi(u_i))^\top$ and $B_i = (M_{\mathrm{old}} P_{\mathrm{old}} \Phi(u_i))^\top$.
    Accumulate $S_{aa} \leftarrow S_{aa} + A^\top A$, $S_{ab} \leftarrow S_{ab} + A^\top B$.
end for
Solve $(S_{aa} + m\gamma I) M_{\mathrm{new}}^\top = S_{ab}$.
Replace $M \leftarrow M_{\mathrm{new}}$; keep prototype and expert coordinates fixed.
Discard decoder, old projection, and calibration statistics.
Evaluate accuracy, residuals, code changes, and routing diagnostics.

I EXTENDED FIGURES AND TABLES

I.1 ACCURACY MATRICES

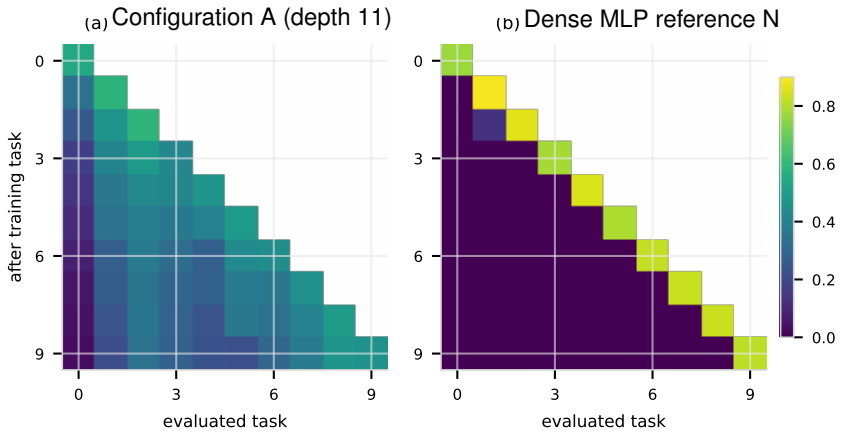

Figure 5: Accuracy matrices (SSL-1000, seed 0; row: completed task, column: evaluated task). *(a)* Frozen routed configuration A, depth 11. *(b)* Dense MLP reference N. The matrices show the retention difference between the two models.

I.2 GUIDED-SLEEP DIAGNOSTICS

Averaged over sleep phases (seed 0), g_1 reduces projection drift from 0.62 to 0.32, residual by 39%, and bit flips by 37%. Figure 3 shows the per-phase values with and without guidance.

I.3 TASK PARTITION AND MIGRATION TIMING

On CPU, migration takes 0.014–5.7 seconds for $n = 256$–4096, with residual at most 0.02 (Figure 4). The complete dense regression cost includes both $O(mn^2)$ moment accumulation and the $O(n^3)$ solve.

I.4 MIGRATION SPECTRA

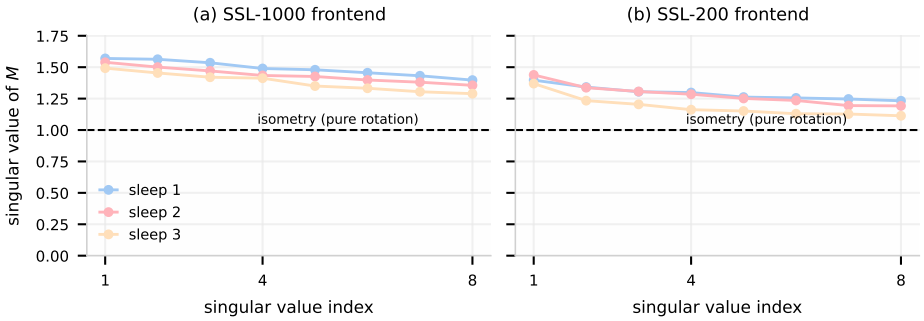

Figure 6: Leading eight singular values of the 256-dimensional migration operator over three sleep phases. *(a)* SSL-1000. *(b)* SSL-200. The dashed line marks unit singular value.

For SSL-1000, the leading singular values lie in $[1.29, 1.57]$, top-1 singular energy is at most 2.4%, and top-16 energy is 20–28%. Median singular value is 0.40–0.71 and the orthogonality deviation is 0.68–0.79. No active bit is shared by at least half the inputs; mean pairwise overlap is approximately one of 16 bits. With the random frontend, top-1 singular energy is 98–100%, and 14–16 active bits are shared by at least half the inputs. SSL codes and migration operators are thus broadly spread, whereas the random frontend concentrates both on a few shared directions.

I.5 FRONTEND AND CAPACITY VALUES

Table 3: Frozen configuration A. Probe is linear-probe accuracy on frontend features. Acquisition is mean task accuracy immediately after learning; retention is avg/acquisition. † denotes a three-seed mean; other results are single-seed. Probe accuracy is a frontend property and is listed once per frontend.

Frontend	Depth	Probe	TLI	avg	F	Acquisition / retention
Random	7	0.300	0.081	0.034	0.183	—
k-means	7	0.347	0.098	0.042	0.267	—
SSL-200	7	0.577	$0.196^\dagger$	$0.130^\dagger$	$0.470^\dagger$	—
SSL-1000	7	0.557	0.231	0.135	0.442	0.576 / 0.234
SSL-1000	9	—	0.476	0.240	0.294	—
SSL-1000	11	—	$0.671^\dagger$	$0.281^\dagger$	$0.225^\dagger$	0.493 / 0.570
k-means	11	—	0.512	0.070	0.162	0.232 / 0.302

I.6 ABLATIONS AND EDITING

Table 4: Depth-7/SSL-200 controls. †: mean $\pm$ s.d. over three seeds; otherwise seed 0. Guidance weights are $\beta = 10$ for g_1 and $\beta = 1$ for g_2.

Method	Configuration	$F \downarrow$	avg $\uparrow$	TLI
$A^\dagger$	Frozen	0.470 ± 0.014	0.130 ± 0.008	0.196 ± 0.012
$C^\dagger$	Sleep + migration	0.473 ± 0.006	0.129 ± 0.005	0.205 ± 0.009
B	Sleep, no migration	0.545	0.097	0.157
E	Developmental phase	0.409	0.093	0.108
C+g_1	Distance guidance	0.471	0.120	0.206
C+g_2	Residual guidance	0.470	0.121	0.210