BRAIN-INSPIRED EFFICIENT CODING IMPROVES CONTINUAL LEARNING IN SMALL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Humans and animals continually learn new tasks and stimuli, but artificial neural networks often struggle to do so without disrupting prior knowledge. A successful continual learning (CL) approach must preserve what has already been learned while using its representational capacity efficiently enough to support future learning. Inspired by efficient coding in information theory and sensory neuroscience, which proposes that neural systems balance response diversity against metabolic cost, we introduce the Efficient Coding Regularizer (ECR), which penalizes response energy while encouraging diverse, high-effective-rank representations. We then study ECR across CL problems of increasing complexity. We first consider domain-incremental learning, where labels remain fixed as the input distribution changes. On permuted MNIST, ECR improves both acquisition of new tasks and retention of previous ones, reduces the rate of task-learning collapse by up to 19.3 percentage points, requires approximately 26% fewer epochs per task, and maintains up to $4.5\times$ the representation diversity (effective rank) of standard training. On CIFAR-100, ECR continues to preserve representational diversity in the corrupted setting but yields smaller and less consistent accuracy gains, and in class-incremental learning, it alone does not prevent forgetting. We therefore develop a nullspace-based approach that protects previous classifiers from destructive updates, providing a foundation on which ECR can act. In this context, applying ECR to the task heads constrains their response scale and reduces global forgetting while keeping accuracy comparable. Together, these results show that efficient-coding principles can preserve representational capacity and regulate network responses, with the clearest benefits in smaller networks and simpler tasks. In larger networks and more complex tasks, efficient coding works best when paired with explicit constraints. Beyond machine learning, these findings inspire neuroscience by suggesting that efficient coding operates most effectively within smaller neural circuits, such as cortical columns, a possibility that remains to be tested.

1 INTRODUCTION

Humans and animals continually adapt to new tasks and stimuli while retaining what they have learned. Continual learning (CL) aims to give artificial neural networks a similar ability by learning from a sequence of tasks while preserving knowledge acquired from previous ones. Learning a new task requires enough plasticity for the model to change, but those same changes can disrupt previously learned information and lead to forgetting (Kirkpatrick et al., 2017). Conversely, restricting the model too strongly to prevent forgetting can reduce its ability to learn new information (Saha & Roy, 2023). A successful CL approach must therefore preserve prior knowledge while maintaining enough representational capacity for future learning. This trade-off parallels the principle of efficient coding from information theory and neuroscience, which asks how a system with limited representational capacity can preserve useful information while reducing redundancy and unnecessary activity. Motivated by this connection, we investigate how efficient coding affects CL and the representations that support it. In the following sections, we first review recent progress in CL, then trace efficient coding from its origins to its use in machine learning, and finally bring the two lines of work together. We show that efficient coding promotes more diverse and informative representations while regulating network responses, with its clearest benefits in small networks and simpler tasks.

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.

Recent work on continual learning. A range of approaches have been developed to address the challenges posed by CL. Replay-based methods preserve previous knowledge by revisiting stored data while learning new tasks. iCaRL stores exemplars from previous classes and uses them to maintain the learned representation and classifier as new classes are introduced (Rebuffi et al., 2017). Other approaches instead limit parameter or representation updates to reduce interference with previously learned tasks (Kirkpatrick et al., 2017; Li & Hoiem, 2016). Projection-based approaches restrict updates more explicitly by controlling the directions in which the model can change (Farajtabar et al., 2020; Saha et al., 2021; Saha & Roy, 2023). In exemplar-free class-incremental learning, where previous task data are not stored, Elastic Feature Consolidation limits feature changes along directions that are important for previous tasks (Magistri et al., 2024), while Self-Sustaining Representation Expansion maintains previously learned features and expands the representation for new classes (Zhu et al., 2022). Auxiliary unlabeled data have also been used to provide broader visual context without storing examples from previous tasks (Sieradzki et al., 2026). Representation geometry has also been structured to reduce interference between classes, for example through fixed equiangular or orthogonal class prototypes (Yang et al., 2023; Seo et al., 2024; Kim & Kim, 2024). This has also been extended by progressively adapting the class-prototype geometry as new classes are introduced (Wang et al., 2026). However, constraining how a model changes can protect prior knowledge without directly addressing whether the representation retains enough capacity to learn new tasks. Critically, loss of plasticity has been associated with an increasing number of inactive units and a decline in the effective rank of learned representations (Dohare et al., 2024). These observations suggest that it may also be important to regulate the statistics of the representation itself so that the network's capacity remains available as tasks accumulate.

Efficient coding. Efficient coding is a normative principle proposing that sensory systems use their limited response capacity to preserve useful information while reducing redundancy and unnecessary neural activity. Early work by Attneave (1954) argued that visual inputs contain substantial redundancy, which motivated the hypothesis that sensory systems transform their inputs to reduce this redundancy while preserving useful information that guides behavior (Barlow, 1961). Subsequent studies showed that sensory responses are shaped by the statistics of the environment, helping neurons use their limited response range more efficiently (Laughlin, 1981; Atick & Redlich, 1990; Dan et al., 1996; Simoncelli & Olshausen, 2001). Because neural activity is metabolically expensive (Laughlin et al., 1998), efficient coding theories have formalized a trade-off between information transmission and the cost of neural responses (Balasubramanian et al., 2001; Simoncelli & Olshausen, 2001; Schwartz & Simoncelli, 2001; Karklin & Simoncelli, 2011; Doi et al., 2012; Pitkow & Meister, 2012; Zhaoping, 2012; Sharpee, 2017; Hsu et al., 2021). More generally, efficient coding can take different forms depending on the information to be represented and the constraints imposed on the neural system (Chalk et al., 2018), including formulations based on balanced network dynamics (Denève & Machens, 2016). Recent theoretical work further generalized this perspective to multimodal and context-sensitive neural circuits, showing that predictive coding, in which neurons signal the deviation of their input from an internal prediction, emerges from an efficient-coding objective rather than requiring a separate computational principle (Tavoni, 2026). Here, we extend this context-sensitive view of efficient coding to task-driven learning.

In machine learning, several approaches have sought to prevent representational collapse and maintain diversity in learned features, particularly in self-supervised learning. Barlow Twins learns representations by enforcing invariance across augmented views of the same input while reducing correlations between representation dimensions (Zbontar et al., 2021). Similarly, VICReg enforces invariance across augmented views, while explicitly reducing covariance (Bardes et al., 2022). Information-theoretic approaches have also proposed objectives that promote diverse and discriminative representations through log-determinant coding-rate criteria (Yu et al., 2020). In these approaches, however, the representation-shaping criterion is itself part of the objective used to learn the representation, typically from relationships between augmented views or samples. Our approach is conceptually different because the representation is learned to solve an independent supervised task, while efficient coding acts as an additional regularizer on neural activity. This provides a direct way to regulate the representational capacity that becomes increasingly important as tasks accumulate.

Our contribution. Inspired by efficient coding in information theory and sensory neuroscience, we introduce the Efficient Coding Regularizer (ECR) for artificial neural networks, which penalizes response energy while encouraging diverse, high-effective-rank representations. We then study ECR

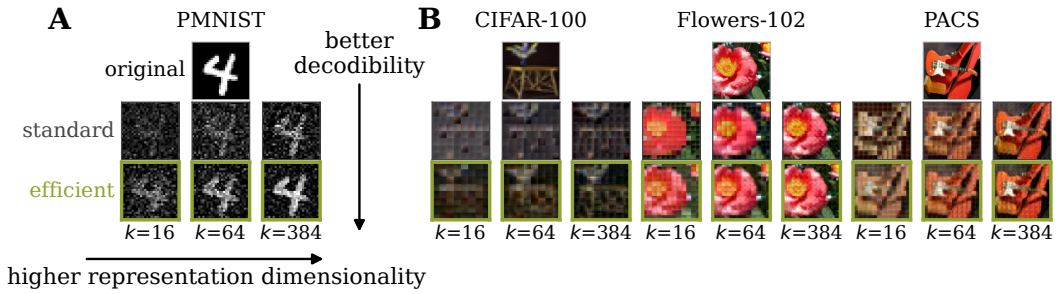

Figure 1: **Efficient coding improves decodability from learned representations.** Images are reconstructed from frozen representations after the sixth transformer block using a linear ridge decoder and the first k principal components of the patch features. A) Analysis is shown for the PMNIST dataset. Moving from $k = 16$ to $k = 384$ increases the number of retained principal components and reduces compression. The rows compare standard and efficient training. B) The same analysis is shown for CIFAR-100, Flowers-102, and PACS. Efficient coding preserves more decodable image information than standard training, especially when fewer principal components are retained.

across CL problems of increasing complexity to examine how efficient-coding principles can preserve representational capacity and regulate network responses, with the clearest benefits in small networks and simpler tasks. Fig. 1 provides a direct view of why maintaining this capacity matters, showing that representations learned with efficient coding retain more information about their inputs and allow the original images to be recovered more faithfully across datasets. Whether such regulation is sufficient depends on the CL setting.

Our approach. We study two CL settings that differ in how tasks change and how predictions are made (van de Ven et al., 2022). In domain-incremental learning, the input distribution changes across tasks while the label space remains the same, so preserving a shared representation may be enough to support both old and new tasks. In class-incremental learning (CIL), new classes are introduced over time and predictions are made over all classes observed so far, so previously learned classifiers must also be protected from interference. We study the CIL setting in the exemplar-free regime, which lets us examine forgetting and the ability to learn new tasks without the additional support of replay, and test how far efficient coding alone can go before additional constraints become necessary. As the problem becomes more complex, we find that efficient coding works best when paired with explicit constraints. We therefore develop a nullspace approach for CIL inspired by projection-based subspace editing (Hatami et al., 2026) to protect previously learned classifiers, and apply ECR to the task heads to regulate their response scale. Since newly trained classifiers do not observe examples from previous tasks, they do not learn when to remain inactive for those inputs. We therefore use unlabeled external images to provide negative evidence for the current classifier and suppress its responses to inputs outside its task. We use Elastic Feature Consolidation (Magistri et al., 2024) as our main exemplar-free baseline.

2 Efficient Coding Regularization (ECR)

Following efficient-coding formulations balancing the spread of neural responses against their energetic cost (Balasubramanian et al., 2001; Tavoni, 2026), we define the Efficient Coding Regularizer (ECR) as the critical combination of two complementary objectives. The first discourages large activations, and the second prevents the representation from collapsing onto a small number of directions. This combined approach is important because reducing response energy alone can encourage the representation to shrink, while encouraging diversity alone does not constrain response magnitude. Let $R \in \mathbb{R}^{N \times D}$ denote the neural responses used by the regularizer, where N is the number of observations and D is the representation dimensionality. For fully connected layers, each row corresponds to the activation of one sample. For convolutional layers, the batch and spatial dimensions are flattened and the channels are treated as the response dimensions. We center the responses as $R_c = R - \frac{1}{N}\mathbf{1}\mathbf{1}^\top R$ and compute their covariance as $C_R = \frac{1}{N} R_c^\top R_c$. We then define

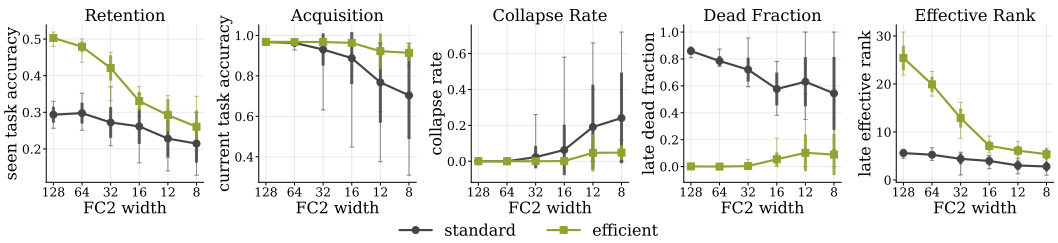

Figure 2: **Effect of representational capacity on continual learning in PMNIST.** Results for a two-hidden-layer network with the first fully connected layer (FC1) width fixed at 256 and FC2 varying. Retention is the mean accuracy on previous tasks, whereas acquisition is the mean accuracy on each task after it is learned. Collapse rate is the fraction of tasks whose acquisition accuracy falls below 50%. Dead fraction and effective rank characterize the FC2 representation and are averaged over the final 10 task checkpoints. Curves show the mean across 20 seeds; thick inner error bars indicate ± 1 standard deviation and thin outer whiskers show the observed seed range.

the response energy and diversity as

$$E(R) = \frac{1}{ND} \|R\|_F^2 , \qquad V(R) = \frac{1}{D} \log\det\left(C_R + \epsilon I\right), \tag{1}$$

where $\epsilon > 0$ is included for numerical stability. We combine these two terms as

$$\mathcal{L}_{\text{ECR}} = \gamma E(R) - \lambda V(R), \tag{2}$$

where γ controls the response-energy penalty and λ controls the response-diversity term. For the classification tasks, we use the cross-entropy task loss $\mathcal{L}_{\text{task}} = \text{CE}(z(x), y)$, where $z(x)$ denotes the model logits and y the corresponding class label. The total objective is $\mathcal{L} = \mathcal{L}_{\text{task}} + \eta \mathcal{L}_{\text{ECR}}$. We adapt η during training by balancing the gradient magnitude of ECR relative to that of the task objective. Let θ_s denote the parameters that receive gradients from both losses, and define $g_{\text{task}} = \|\nabla_{\theta_s} \mathcal{L}_{\text{task}}\|_2$ and $g_{\text{ECR}} = \|\nabla_{\theta_s} \mathcal{L}_{\text{ECR}}\|_2$. We set the target ECR weight as

$$\eta^* = \rho \frac{g_{\text{task}}}{\max(g_{\text{ECR}}, \epsilon_g)}, \tag{3}$$

where ρ determines the desired ratio between the ECR and task-loss gradient magnitudes and ϵ_g provides numerical stability. Sensitivity analyses for all three hyperparameters γ, λ, and ρ are provided in Appendix C. The value of η is updated toward η^* during training, allowing the contribution of ECR to adjust as the relative gradient magnitudes of the two objectives change. ECR therefore encourages the network to maintain diverse responses without requiring high response energy. Throughout the figures and tables, "efficient" denotes the corresponding network trained with ECR. Armed with this approach, we next study how ECR contributes to continual-learning settings of increasing complexity.

2.1 Efficient Coding in Domain-Incremental Learning

We first consider domain-incremental learning (DIL), where the class labels remain fixed while the input distribution changes across tasks. We evaluate ECR on two DIL settings: Permuted MNIST (PMNIST), where tasks differ by fixed pixel permutations, and CIFAR-100-Corrupted (CIFAR-100-C), where tasks correspond to different image corruptions. These settings allow us to examine ECR as the data and model complexity increase.

2.1.1 Permuted MNIST (PMNIST)

We use 50 sequential PMNIST tasks and a fully connected network with two hidden layers. Full dataset, training, and evaluation details are provided in Appendix B. We compare standard training with ECR using the same architectures and task sequences. Fig. 2 summarizes how performance and representation statistics change as the second layer (FC2) becomes narrower, while the first fully connected layer (FC1) is fixed at 256 units. As FC2 becomes narrower, standard training shows lower acquisition and retention, together with an increase in collapse rate. In contrast, ECR

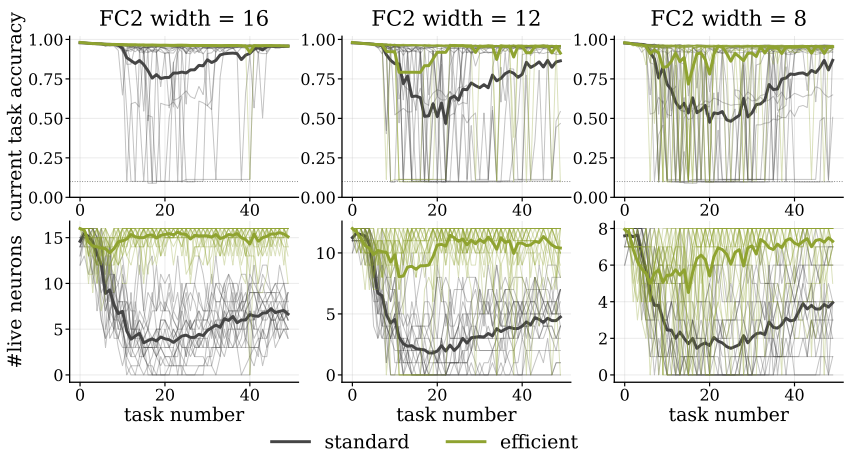

Figure 3: **Task acquisition and representation survival under bottlenecks on PMNIST.** The first fully connected (FC1) layer width is fixed at 256, and columns correspond to FC2 widths. The top row shows accuracy on the current task after learning it, and the bottom row shows the number of live FC2 units at the same point. Thin curves denote individual seeds and thick curves denote their mean. The dotted horizontal line in the accuracy panels indicates chance performance (0.1).

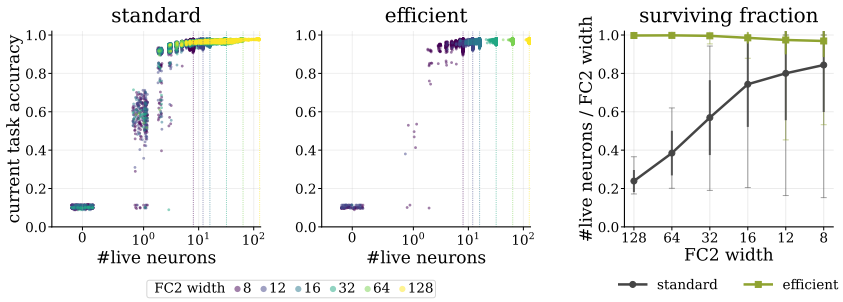

Figure 4: **Relationship between surviving representational capacity and task acquisition in PMNIST.** The first two panels show current-task accuracy as a function of the number of live neurons for standard and efficient training, respectively. Each point corresponds to a task checkpoint, colors identify the FC2 width, and the matching vertical dotted line marks that width's maximum possible number of live units. The right panel summarizes the surviving fraction, defined as the number of live neurons divided by FC2 width, for each FC2 width. Lines and markers show the mean; thick inner bars indicate ± 1 standard deviation, and thin outer whiskers show the observed range across 20 seed summaries.

maintains higher acquisition and retention, suggesting that improved retention does not come at the cost of learning new tasks, and improved acquisition does not come at the cost of retaining previous ones. These results are also associated with ECR reducing the collapse rate by up to 19.3 percentage points (pp) and maintaining a higher effective rank, reaching up to $4.5\times$ that of standard training. ECR also reduces the dead fraction by up to 86 pp, indicating that a larger portion of the available representation remains active as capacity becomes constrained.

To evaluate how learning changes over the task sequence, Fig. 3 shows these differences across tasks for FC1 fixed at 256 and varying FC2 widths. Under standard training, the number of live FC2 units decreases sharply as more tasks are learned, and current-task accuracy can drop to 10%, the chance accuracy. In some cases, the network remains in this collapsed state for several tasks. ECR can also show drops in accuracy, but it usually recovers by the next task. At the same time, ECR keeps most FC2 units active throughout training, often close to the full layer width, whereas standard training often struggles to keep 50% of them active. This suggests that keeping more of the representation active helps the network recover from failures and continue learning new tasks.

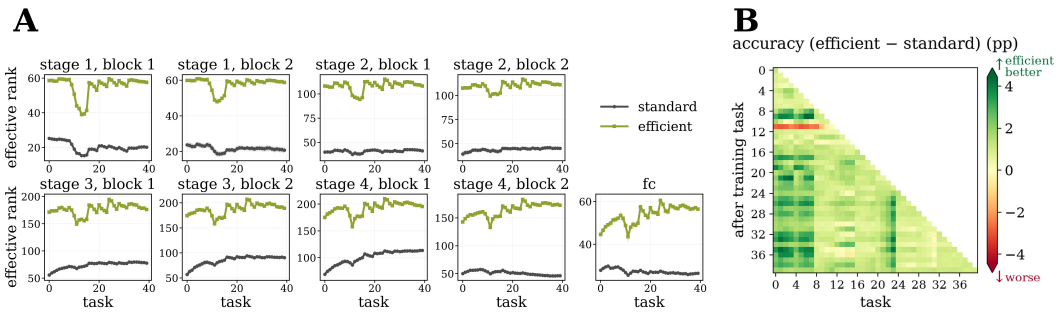

Figure 5: **Representation diversity and performance during CIFAR-100-C continual learning.** A) Effective rank across ResNet-18 layers as tasks are learned, averaged over all observed tasks at each checkpoint. Curves show the mean across seeds and shading denotes ± 1 standard deviation. B) Test-accuracy difference between efficient coding and standard training across training and evaluation tasks. Each cell shows the seed-averaged paired difference, with training task on the y-axis and evaluation task on the x-axis. Positive values indicate higher accuracy with efficient coding.

To examine the relationship between representational capacity and learning performance, Fig. 4 plots current-task accuracy against the number of live FC2 units. Across both methods, fewer active units are associated with lower current-task accuracy, indicating loss of representation is linked to poorer task acquisition. The surviving-fraction summary in the right panel shows that ECR keeps the fraction of active FC2 units close to 100% across widths, suggesting its association with better maintained performance. Finally, to examine the effect of ECR on convergence speed, Fig. 6 shows the number of epochs required for each task. Standard training generally requires more epochs as the CL sequence progresses, whereas ECR reaches the convergence criterion in fewer epochs and remains more stable across later tasks. Across the architectures tested, ECR requires approximately 26% fewer epochs per task than standard training.

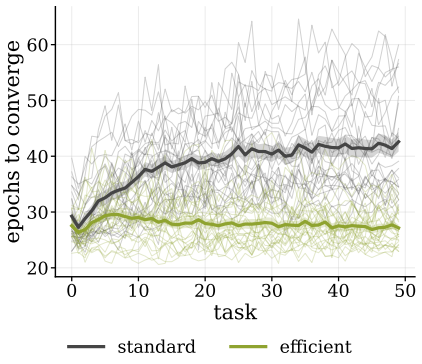

Figure 6: **Training length across the PMNIST continual-learning sequence.** The number of epochs run for each task is shown as a function of task number for standard and efficient training. Thin curves represent the seed-averaged trajectory of individual architecture runs, whereas thick curves show the mean across runs. Shaded regions indicate the standard error across trajectories. The corresponding trajectories for each individual architecture are shown in Appendix C, Fig. 11.

Where to see the impact of representations? The PMNIST results above show that ECR keeps more units active and maintains a higher effective rank. To see whether this preserved representation also retains more information, we freeze the networks after training on the full task sequence and return to held-out images from the first task. We reconstruct each image from its learned representation using a linear ridge decoder, while varying the number of principal components k available to the decoder. Figure 1 shows that ECR preserves more decodable image information than standard training at the same k, with the clearest difference when the representation is more compressed. This shows that the dimensions kept active by ECR continue to carry more decodable information about the first task than standard training, even after the full task sequence.

2.1.2 CIFAR-100-Corrupted (CIFAR-100-C)

We next consider CIFAR-100-C, following the common-corruption benchmark introduced by Hendrycks & Dietterich (2019) and adapting it to a sequential domain-incremental setting. We use 40 tasks and a ResNet-18, with full experimental details provided in Appendix B. Fig. 5A shows how representation diversity changes across the ResNet-18 layers as more corrupted domains

Table 1: **Performance comparison of Nullspace and OOD-based configurations on CIFAR-100 and Tiny-ImageNet class-incremental learning.** Current task and global accuracy are measured after the final increment, while average incremental global accuracy summarizes performance throughout the learning sequence. Forgetting is reported under task and global evaluation. OOD ratio denotes the ratio of unlabeled external images to training data. CIFAR-10 images are used for CIFAR-100 and ImageNet images for Tiny-ImageNet, with labels discarded. Results are reported as mean $\pm$ standard deviation across 3 seeds. For CIFAR-100, the corresponding results across 20 seeds are reported in Appendix D.

	Configuration	Current Global Acc. (%)	Avg. Incremental Global Acc. (%)	Global Forgetting (pp)	Current Task Acc. (%)	Task Forgetting (pp)
CIFAR-100	Standard	4.7 ± 0.2	17.8 ± 0.5	85.8 ± 1.4	27.9 ± 1.4	65.3 ± 1.3
	Efficient	4.4 ± 0.1	16.3 ± 0.1	89.5 ± 0.9	25.0 ± 0.6	67.8 ± 0.5
	Nullspace	29.6 ± 2.6	47.3 ± 3.2	24.0 ± 1.0	87.0 ± 1.8	0.0 ± 0.0
	Nullspace + Efficient Head	28.0 ± 3.8	46.7 ± 3.3	21.8 ± 3.1	87.1 ± 2.2	0.0 ± 0.0
	EFC (Magistri et al., 2024)	34.6 ± 1.9	51.7 ± 2.1	24.2 ± 1.3	85.4 ± 1.2	1.5 ± 0.2
	Nullspace + 0.05 OOD	39.3 ± 2.3	54.4 ± 4.0	14.6 ± 0.5	87.0 ± 2.1	0.0 ± 0.0
Tiny-ImageNet	Standard	4.9 ± 0.2	16.9 ± 0.2	76.9 ± 0.3	23.0 ± 1.7	60.8 ± 1.6
	Efficient	4.3 ± 0.1	15.1 ± 0.1	81.3 ± 0.4	16.1 ± 0.8	68.9 ± 1.0
	Nullspace	24.1 ± 0.6	38.9 ± 0.4	28.3 ± 1.4	78.6 ± 0.4	0.0 ± 0.0
	Nullspace + Efficient Head	23.7 ± 0.9	40.9 ± 0.9	18.0 ± 2.0	80.0 ± 0.6	0.0 ± 0.0
	EFC (Magistri et al., 2024)	29.1 ± 0.1	44.5 ± 0.4	29.5 ± 0.4	78.8 ± 0.4	1.5 ± 0.2
	Nullspace + 0.05 OOD	37.6 ± 0.1	51.2 ± 0.2	11.6 ± 0.5	79.0 ± 0.1	0.0 ± 0.0

are learned, and Fig. 5B shows the corresponding accuracy difference between ECR and standard training across training and evaluation tasks. The reduction in this difference after every four tasks coincides with the introduction of a new corruption type, at which point both networks must adapt to a newly shifted domain. The largest differences arise from Gaussian and shot noise, which are considered among the most challenging corruption types (Lim et al., 2026). Around task 11 in Fig. 5A, the effective rank drops across all layers. A similar change appears in Fig. 5B, where the row for task 11 becomes red, indicating that ECR performs worse than standard training at that point. Although this alignment suggests a connection between effective rank and performance, it also indicates that higher effective rank alone is not sufficient to ensure better performance, as ECR still maintains at least twice the effective rank of standard training. Therefore, while regularizers such as ECR help preserve representational diversity and regulate network responses, they are not sufficient on their own to preserve performance in continual learning. In the next section, we consider the more complex class-incremental setting and examine the additional mechanisms needed to support ECR.

2.2 Efficient Coding in Class-Incremental Learning

We next consider class-incremental learning (CIL), where each task introduces a new set of classes and the output space grows throughout the learning sequence. During training, only data from the current task are available, while at test time the task identity is unknown and predictions are made over all classes observed so far. We study this setting on CIFAR-100 and Tiny ImageNet. Compared with DIL, CIL introduces an additional challenge because newly learned classes must coexist with previously learned classes within a shared prediction space. We therefore examine how ECR behaves in this setting and introduce additional mechanisms when representation regularization alone is not sufficient to preserve previous classifiers and support global prediction. We begin with CIFAR-100 using the same exemplar-free setting as Elastic Feature Consolidation (Magistri et al., 2024), and then evaluate the same approach on Tiny ImageNet. Within this setting, we first evaluate standard training and ECR alone. As shown in Table 1, neither provides satisfactory performance in this more challenging setting. After the final increment, standard training reaches only 4.7% global accuracy and 27.9% current task accuracy, while ECR reaches 4.4% and 25.0%, respectively. Both also exhibit substantial forgetting. These results indicate that applying ECR alone is not sufficient to protect previously learned classifiers as new classes are introduced. We therefore first introduce a mechanism with an explicit preservation guarantee, and then evaluate the additional effect of ECR under this constraint.

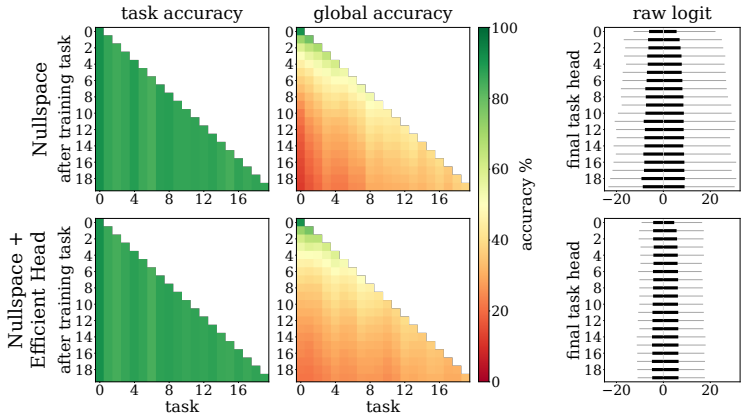

Figure 7: **Classifier preservation and cross-head response scale in CIFAR-100 class-incremental learning.** The first two columns show current-task and global accuracy, respectively. The last column shows the final raw-logit distribution of each task head, with gray lines spanning the minimum to maximum and black lines marking the 5th-95th percentiles. Head-level efficient coding constrains head-response growth while preserving task accuracy and smoothing global performance.

2.2.1 Preserving Previous Classifiers with Nullspace Projection

A challenge in CIL is to learn representations for new classes without changing the outputs of classifiers learned for previous tasks. Inspired by the projection-based subspace editing of Hatami et al. (2026), we restrict each new representational change to the nullspace of all previously learned classification heads. Let $h(x)$ denote the shared representation before the final residual stage, and let B_t denote the final residual block used to learn task t. We define the feature vector for task t as $g_t(x) = \mathrm{pool}(B_t(h(x))) \in \mathbb{R}^d$, where $\mathrm{pool}(\cdot)$ denotes global average pooling over the spatial dimensions and $d = 512$ for ResNet-18. For each task t, B_t begins from the representation learned for the previous task, and we define the representational change as $\Delta_t(x) = g_t(x) - g_{t-1}(x)$. This change is zero at the beginning of the task and evolves as the new task is learned. Let $W_j \in \mathbb{R}^{C_j \times d}$ denote the classification head for task j, where C_j is the number of classes in that task. Before learning task t, we vertically concatenate all previous classification heads and denote the matrix by $W_{<t}$. We then define the orthogonal projector onto the nullspace of the previous classifiers as

$$P_t = I - W_{<t}^\top \left(W_{<t} W_{<t}^\top\right)^\dagger W_{<t}, \tag{4}$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse. The projector removes the component of a representation change that lies in directions used by the previous classifiers. The representation is then updated using only the projected change,

$$f_t(x) = f_{t-1}(x) + P_t \Delta_t(x), \qquad f_0(x) = g_0(x). \tag{5}$$

By construction, $W_{<t} P_t = 0$, so for every previously learned head W_j with $j < t$, we have $W_j f_t(x) = W_j(f_{t-1}(x) + P_t \Delta_t(x)) = W_j f_{t-1}(x)$. The logits of every previous classifier therefore remain unchanged as a new task is learned. Since this relation holds for any input x, preserving the previous classifiers does not require storing or replaying samples from earlier tasks. In our setting, when learning task t, the shared representation and previously learned classifiers remain fixed, while the new task representation and classifier are learned under the nullspace constraint.

2.2.2 ECR at the Task Head

The nullspace constraint preserves previous classifiers, but the responses of newly learned heads can still vary in scale. We therefore apply ECR directly to the current task logits $z_t(x) = W_t f_t(x) \in \mathbb{R}^{C_t}$. For a batch of size B, we use the same energy and diversity terms as in Eq. 2, but compute the energy from the raw logits:

$$E_t = \frac{1}{BC_t} \sum_{b=1}^{B} \|z_t(x_b)\|_2^2, \qquad V_t = \frac{1}{C_t} \log\det(\Sigma_t + \epsilon I), \qquad \mathcal{L}_{\text{head}} = \gamma E_t - \lambda V_t, \tag{6}$$

where Σ_t is the covariance of the logits across the batch. As in the representation-level ECR, the energy term is not centered, so it also penalizes large common offsets in the logits. The objective is applied only to the current task head, and previous heads remain fixed. Table 1 summarizes the results for Nullspace and Nullspace + Efficient Head. Nullspace improves accuracy and reduces forgetting relative to standard training. Adding ECR at the task head further reduces global forgetting by 2.2 pp, from 24.0 to 21.8, while keeping global and current-task accuracies comparable. The same pattern holds on Tiny ImageNet, where head-level ECR reduces global forgetting by 10.3 pp while maintaining comparable accuracy. Fig. 7 shows how this difference develops across tasks. In the task-accuracy heatmaps, the color within each task column remains unchanged as later tasks are learned, reflecting the absence of task-level forgetting. Head-level ECR produces a smoother pattern of global accuracy across previously learned tasks while keeping the task-head logit scales more controlled. As a separate comparison, we evaluate whether directly suppressing responses to task-external inputs using unlabeled external images can improve global prediction.

2.2.3 Suppressing Out-of-Distribution Responses

The issue across the networks we have tried so far is that, after the full training sequence, each classifier unit has learned when to activate for its own class and when not to activate for other classes encountered within its task or in later tasks. However, because examples from previous tasks are unavailable when a new classifier is trained, it never learns when it should remain inactive for previously seen classes. This is important in class-incremental inference because even when a task is perfectly preserved, its classifier can still activate strongly for an input from another task and dominate the global prediction. We therefore use external images to help the current classifier learn when not to respond. These images should be sufficiently similar to the training domain to produce meaningful network responses, while remaining outside the distribution of the task data. For an external input x^{ood}, we use only the current task head and compute $z_t(x^{\mathrm{ood}}) = W_t f_t(x^{\mathrm{ood}}) \in \mathbb{R}^{C_t}$. For a batch of B external inputs, we penalize the current-head response energy as

$$\mathcal{L}_{\mathrm{ood}} = \frac{1}{BC_t} \sum_{b=1}^{B} \left\| z_t(x_b^{\mathrm{ood}}) \right\|_2^2, \tag{7}$$

which penalizes head responses to external inputs, weighted by $\beta_{\mathrm{ood}} = 10^{-2}$. Therefore, with only the current task head optimized and all previous heads kept fixed, the complete training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{task}} + \beta_{\mathrm{ood}} \mathcal{L}_{\mathrm{ood}}. \tag{8}$$

As reported in Table 1, on CIFAR-100, adding OOD suppression with a $0.05{:}1$ OOD-to-training-data ratio improves the latest global accuracy from 29.6% to 39.3% and reduces global forgetting by 9.4 pp, while task forgetting remains zero. The head responses were also analyzed in Appendix Sec. D, showing that OOD training reduces incorrect head responses to inputs from other tasks.

3 Conclusion

In this work, we examined how efficient-coding principles can support continual learning by preserving representational capacity while regulating neural responses. Across domain-incremental settings, ECR maintains more active and higher-rank representations, with the clearest benefits under constrained capacity, where it supports both acquisition and retention, reduces collapse rate, and enables faster convergence. As the learning setting becomes more complex, however, ECR alone is not sufficient to maintain performance. In class-incremental learning, we therefore combine efficient coding with complementary constraints, without storing or replaying previous-task examples. Under these constraints, ECR further regulates classifier responses and reduces global forgetting while maintaining task performance. Overall, the benefits of efficient coding are most apparent in smaller networks and simpler settings, while more complex problems benefit from complementary constraints. These findings also suggest that efficient coding may be particularly effective within smaller neural circuits, such as cortical columns, where representational capacity is more constrained, although this possibility remains to be tested experimentally.

AI USE STATEMENT

In this work, we used generative AI tools to improve the grammar, fluency, and readability of text written by the authors. We also used generative AI tools to help prepare a compact and anonymized version of our existing code for the supplementary material. This use was limited to simplifying the presentation of the code and removing references that could reveal author identity. We did not use generative AI tools for research tasks requiring disclosure under the ICLR policy, including designing the methodology or experiments, implementing the methods used to produce the reported results, processing data, formulating mathematical claims, or interpreting results. All AI-assisted text and supplementary code were carefully reviewed by the authors and checked for correctness and anonymity. We take responsibility for the final content of the paper and all supplementary materials.

ETHICS STATEMENT

This work uses existing public benchmark datasets and does not involve human subjects, collection of private or sensitive data, or release of a new dataset. The authors take responsibility for the integrity of the reported experiments and results.

REPRODUCIBILITY STATEMENT

The main method and training objectives are described in Section 2. Appendix B provides the experimental setup and implementation details, including task construction, datasets, architectures, training procedures, hyperparameters, and evaluation protocols. Appendices C and D provide additional analyses and sensitivity results. We also provide an anonymized compact version of the code in the supplementary material to support understanding and reproduction of the implementation. The complete codebase will be made publicly available after the review process and de-anonymization.

REFERENCES

Joseph J. Atick and A. Norman Redlich. Towards a theory of early visual processing. *Neural Computation*, 2(3):308–320, 1990.

Fred Attneave. Some informational aspects of visual perception. *Psychological Review*, 61(3): 183–193, 1954.

Vijay Balasubramanian, Don Kimber, and Michael J. Berry. Metabolically efficient information processing. *Neural Computation*, 13(4):799–815, 2001. doi: 10.1162/089976601300014358.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022.

Horace B. Barlow. Possible principles underlying the transformations of sensory messages. In Walter A. Rosenblith (ed.), *Sensory Communication*, pp. 217–234. MIT Press, 1961.

Matthew Chalk, Olivier Marre, and Gašper Tkačik. Toward a unified theory of efficient, predictive, and sparse coding. *Proceedings of the National Academy of Sciences*, 115(1):186–191, 2018.

Yang Dan, Joseph J. Atick, and R. Clay Reid. Efficient coding of natural scenes in the lateral geniculate nucleus: Experimental test of a computational theory. *Journal of Neuroscience*, 16 (10):3351–3362, 1996. doi: 10.1523/JNEUROSCI.16-10-03351.1996.

Sophie Denève and Christian K. Machens. Efficient codes and balanced networks. *Nature Neuroscience*, 19(3):375–382, 2016.

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*, 632:768–774, 2024. doi: 10.1038/s41586-024-07711-7.

Eizaburo Doi, Jeffrey L. Gauthier, Greg D. Field, Jonathon Shlens, Alexander Sher, Martin Greschner, Timothy A. Machado, Lauren H. Jepson, Keith Mathieson, Deborah E. Gunning, Alan M. Litke, E. J. Paninski, Liam Chichilnisky, and Eero P. Simoncelli. Efficient coding of spatial information in the primate retina. *The Journal of Neuroscience*, 32(46):16256–16264, 2012.

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.

Arman Hatami, Romina Aalishah, and Ilya E. Monosov. Class unlearning via depth-aware removal of forget-specific directions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 8516–8525, 2026.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Wei-Mien M. Hsu, David B. Kastner, Stephen A. Baccus, and Tatyana O. Sharpee. How inhibitory neurons increase information transmission under threshold modulation. *Cell Reports*, 35(8), 2021.

Yan Karklin and Eero Simoncelli. Efficient coding of natural images with a population of noisy linear-nonlinear neurons. *Advances in Neural Information Processing Systems*, 24, 2011.

Hoyong Kim and Kangil Kim. Fixed non-negative orthogonal classifier: Inducing zero-mean neural collapse with feature dimension separation. In *International Conference on Learning Representations*, 2024.

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.

Simon Laughlin. A simple coding procedure enhances a neuron's information capacity. *Zeitschrift für Naturforschung C*, 36:910–912, 1981.

Simon B. Laughlin, Rob R. de Ruyter van Steveninck, and John C. Anderson. The metabolic cost of neural information. *Nature Neuroscience*, 1:36–41, 1998. doi: 10.1038/236.

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

Taejun Lim, Joong-Won Hwang, and Kibok Lee. When and where to reset matters for long-term test-time adaptation. In *International Conference on Learning Representations*, 2026.

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 *The Twelfth International Conference on Learning Representations*, 2024.

Xaq Pitkow and Markus Meister. Decorrelation and efficient coding by retinal ganglion cells. *Nature Neuroscience*, 15(4):628–635, 2012.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2001–2010, 2017. doi: 10.1109/CVPR.2017.587.

Gobinda Saha and Kaushik Roy. Continual learning with scaled gradient projection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9677–9685, 2023. doi: 10.1609/aaai.v37i8.26157.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

Odelia Schwartz and Eero P. Simoncelli. Natural signal statistics and sensory gain control. *Nature Neuroscience*, 4(8):819–825, 2001.

Minhyuk Seo, Hyunseo Koh, Wonje Jeung, Minjae Lee, San Kim, Hankook Lee, Sungjun Cho, Sungik Choi, Hyunwoo Kim, and Jonghyun Choi. Learning equi-angular representations for online continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23933–23942, 2024.

Tatyana O. Sharpee. Optimizing neural information capacity through discretization. *Neuron*, 94(5): 954–960, 2017.

Igor Sieradzki, Łukasz Struski, Igor Tomasz Podolak, and Romuald Aleksander Janik. Class incremental learning and auxiliary unlabelled data: The importance of neutral examples. *Machine Learning*, 115(4), 2026.

Eero P. Simoncelli and Bruno A. Olshausen. Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24:1193–1216, 2001.

Gaia Tavoni. A unified theory of context-conditioned efficient and predictive coding. *bioRxiv*, 2026.

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.

Zheng Wang, Wanhao Yu, Li Yang, and Sen Lin. Rethinking continual learning with progressive neural collapse. In *International Conference on Learning Representations*, 2026.

Yibo Yang, Haobo Yuan, Xiangtai Li, Jianlong Wu, Lefei Zhang, Zhouchen Lin, Philip Torr, Dacheng Tao, and Bernard Ghanem. Neural collapse terminus: A unified solution for class incremental learning and its variants. *arXiv preprint arXiv:2308.01746*, 2023.

Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12310–12320. PMLR, 2021.

Li Zhaoping. Understanding vision: theory, models, and data. *University College London, UK*, 2012.

Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9296–9305, 2022.

A Appendix

A.1 Limitations and Future Work

Our results show that efficient coding is most effective in smaller networks and simpler continual-learning settings. In larger networks and class-incremental learning, preserving representation diversity alone is not sufficient to prevent forgetting, and additional constraints are needed. Our experiments are also limited to a small set of image-classification benchmarks. In addition, the parameter ρ is fixed throughout training in ECR. Future work could make ρ adaptive, allowing the strength of efficient coding relative to the task objective to change across tasks or training stages. It would also be useful to test efficient coding on larger architectures, more diverse datasets, and other continual-learning settings, and to better understand when higher representation diversity leads to improved task performance.

B Experimental Setup and Implementation Details

We use the same task sequence, model initialization, and training procedure when comparing standard training with ECR within each experimental setting. Table 2 summarizes the main dataset and training settings.

Table 2: **Summary of the experimental settings.** The number of classes per task refers to newly introduced classes in the class-incremental settings. PMNIST and CIFAR-100-C retain a shared label space across tasks.

Dataset	Setting	Tasks	Classes	Model	Batch size	Training
PMNIST	Domain-IL	50	10 shared	Two-layer MLP	600	Validation convergence
CIFAR-100-C	Domain-IL	40	100 shared	ResNet-18	128	Sequential domains
CIFAR-100	Class-IL	20	5/task	ResNet-18	64	100 epochs/task
Tiny ImageNet	Class-IL	20	10/task	ResNet-18	64	100 epochs/task

Permuted MNIST. For PMNIST, the original MNIST training set is divided into fixed training and validation subsets, and the original MNIST test set is used for evaluation. The same split is used for every task. Each task applies a different fixed permutation to the 784 input pixels while keeping the digit labels unchanged. We use 50 sequential tasks. The model is a fully connected network with ReLU activations and two hidden layers. We use a batch size of 600 and Adam optimization. Training is stopped using validation performance so that the number of epochs required to learn each task can also be measured. Standard training and ECR use the same stopping procedure. For ECR, we use $\epsilon = 10^{-4}$ for covariance regularization and $\epsilon_g = 10^{-12}$ for numerical stability in the adaptive gradient-balancing rule. We use $\gamma = 10^{-3}$, $\lambda = 5 \times 10^{-4}$, and $\rho = 0.1$.

CIFAR-100-Corrupted. For CIFAR-100-C, the class labels remain fixed while the image distribution changes across tasks. We reproduce the corruption procedure of the CIFAR-100-C benchmark introduced by Hendrycks & Dietterich (2019) and adapt it to a sequential domain-incremental setting. We use 10 corruption types, namely Gaussian noise, shot noise, defocus blur, motion blur, zoom blur, fog, brightness, contrast, elastic transform, and pixelate, each at four severity levels. This gives 40 domains in total, with each corruption and severity combination defining one task. Figure 8 shows an example of the resulting task sequence for a single CIFAR-100 image. All 100 classes remain available in every task. We use a ResNet-18 and compare standard training with ECR using the same task ordering and optimization procedure. Representation statistics are measured throughout the residual network after each task, and the reported results are averaged across 5 seeds.

CIFAR-100 class-incremental learning. For CIFAR-100, the 100 classes are divided into 20 tasks containing five classes each. Only examples from the current task are available during training. Current task accuracy is computed by restricting prediction to the five classes of the corresponding task, while global accuracy is computed over all classes observed up to that point. We use a ResNet-18 with a 512-dimensional pooled representation and one bias-free linear classification head per task. Task 0 is trained for 100 epochs using Adam with learning rate 10^{-3}, weight decay 2×10^{-4},

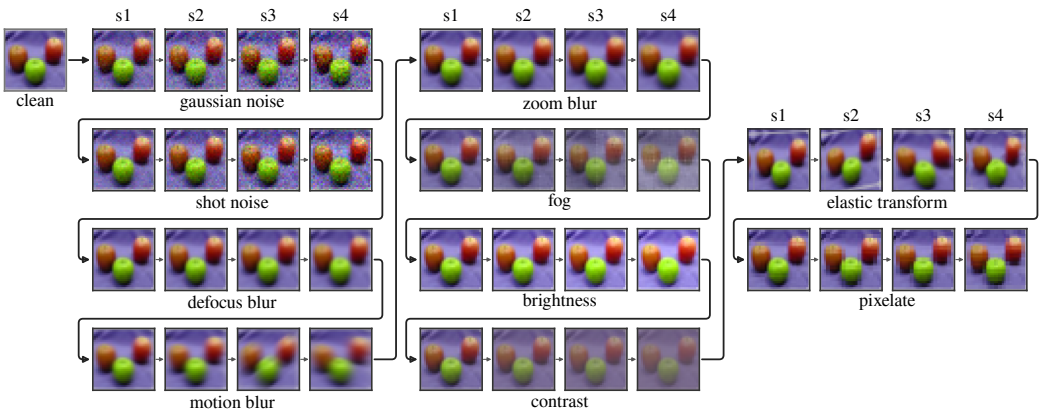

Figure 8: **Sequential CIFAR-100-C corruption stream.** Example of the 40-task domain-incremental sequence constructed from a single CIFAR-100 image. The stream follows the CIFAR-100-C corruption procedure (Hendrycks & Dietterich, 2019) and contains 10 corruption types, each presented at four increasing severity levels (s1-s4). The class label remains unchanged across all tasks, while the image distribution changes with the corruption type and severity.

and learning-rate reductions by a factor of 0.1 at epochs 45 and 90. For subsequent tasks, the shared network through layer3, all previous task heads, and all previous layer4 branches are fixed. Only the current layer4 branch and current classification head are optimized for 100 epochs using Adam with learning rate 10^{-4} and weight decay 2×10^{-4}.

For the head-level ECR experiments, we use $\gamma = 10^{-3}$, $\lambda = 5 \times 10^{-4}$, and $\epsilon = 10^{-4}$. When OOD suppression is used, the OOD energy weight is 10^{-2}. CIFAR-10 training images are used as external OOD inputs, their labels are ignored, and no CIFAR-100 examples from previous or future tasks are used.

Tiny ImageNet class-incremental learning. Tiny ImageNet contains 200 classes, which we divide into 20 tasks containing 10 classes each. We use the same custom ResNet-18 architecture as the original EFC Tiny ImageNet implementation. For 64×64 inputs, the network uses a 3×3 stride-one stem without an initial max-pooling layer, followed by four residual stages with channel dimensions 64, 128, 256, and 512. The final layer4 activation has spatial size 8×8 and is reduced to a 512-dimensional representation by average pooling. The class-incremental training procedure follows the CIFAR-100 setup, with 100 epochs for the first and subsequent tasks. For experiments using external OOD inputs, we use ImageNet-1k training classes after excluding all 200 Tiny ImageNet synsets. This leaves 800 external ImageNet synsets with no class overlap with Tiny ImageNet. Their labels are not used during training.

B.1 Effective Rank of Learned Representations

We use effective rank to quantify how broadly a representation uses its available dimensions. Let $R \in \mathbb{R}^{N \times D}$ denote a set of neural responses and let R_c denote the centered responses. We compute the regularized covariance

$$\widetilde{C}_R = \frac{1}{N} R_c^\top R_c + \epsilon I. \tag{9}$$

Let $\lambda_1, \ldots, \lambda_D$ denote the eigenvalues of $\widetilde{C}_R$, and define their normalized spectrum as

$$p_i = \frac{\lambda_i}{\sum_{j=1}^{D} \lambda_j}. \tag{10}$$

The effective rank is then

$$r_{\text{eff}}(R) = \exp\left(-\sum_{i=1}^{D} p_i \log p_i \right). \tag{11}$$

Effective rank therefore measures how many representational directions meaningfully contribute to the variability of the responses. It is low when the variance is concentrated along only a few directions and higher when the variance is distributed across more directions.

For convolutional layers, the batch and spatial dimensions are flattened, and the channels are treated as the representation dimensions. In the PMNIST analysis, effective rank is not included when a layer has no live units. Since the representation has no variation in this case, the ϵI term added for numerical stability would otherwise give a misleadingly high effective rank. Dead statistics are reported separately.

B.2 Dead Fraction

We use the dead fraction to measure how much of the available representation becomes inactive during learning, and we measure it at the end of each task. Using the centered response matrix $R_c \in \mathbb{R}^{N \times D}$ defined above, the response variance of unit j is

$$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^{N} R_{c,ij}^2. \tag{12}$$

We consider a unit dead when its response variance is below a small threshold τ, $\sigma_j^2 < \tau$, and define the dead fraction as

$$f_{\text{dead}} = \frac{1}{D} \sum_{j=1}^{D} \mathbb{I}\left[\sigma_j^2 < \tau\right], \tag{13}$$

where $\mathbb{I}[\cdot]$ is the indicator function. We use $\tau = 10^{-6}$ throughout the PMNIST analysis. A larger dead fraction therefore indicates that a larger portion of the available representation shows little or no variation across the evaluated inputs. The number of live units is defined equivalently as the number of dimensions with $\sigma_j^2 \geq \tau$. For PMNIST, these statistics are computed after each task for each seen task using its full permuted test set.

B.3 Transformer Experiments

The analyses in Appendices E to I, and Figure 1, use a vision transformer rather than the networks described above. This subsection gives the setup for those runs. Table 3 summarizes the settings.

Table 3: **Summary of the transformer experiments.** All four settings use the same architecture, optimizer, and stopping rule, so only the data and the patch size change. Classes per task refers to newly introduced classes in the class-incremental settings, and PMNIST and PACS keep a shared label space across tasks.

Dataset	Setting	Tasks	Classes	Image size	Patch size	Batch size
PMNIST	Domain-IL	20	10 shared	28×28	4	90
CIFAR-100	Class-IL	20	5/task	32×32	4	90
Flowers-102	Class-IL	20	5/task	96×96	8	32
PACS	Domain-IL	4	7 shared	96×96	8	32

Architecture. We use a vision transformer with six encoder blocks, six attention heads, a token width of 384, and an MLP ratio of four. An image is divided into square patches, each patch is embedded linearly, a learned class token is prepended, and learned position embeddings are added. Every block updates all tokens, so after each block the image is represented by one state per patch together with the class token. We refer to the per patch states as patch tokens and to the state of the class token as the classifier token, since the linear classification head reads the class token after a final layer normalization. With a patch size of 4 a 28×28 image gives 49 patch tokens and a 32×32 image gives 64, and with a patch size of 8 a 96×96 image gives 144. Both training methods start from the same initialization within a run.

Permuted MNIST. The transformer runs use the same permuted MNIST construction as above, with 20 tasks instead of 50 so that the sequence matches the other transformer settings. Each task applies a different fixed permutation to the input pixels while the digit labels stay the same, so the label space is shared and only the input distribution changes.

CIFAR-100 class-incremental learning. CIFAR-100 contains 100 classes of natural images at 32×32 resolution. We divide the classes into 20 tasks of five classes, with the class order fixed by the run seed, and only the examples of the current task are available during training. Prediction is restricted to the classes seen so far, so accuracy reflects what the network has been shown rather than the full label space. Compared with PMNIST this setting adds colour, natural image statistics, and a growing label space, and it is the middle level of difficulty in our transformer experiments.

Flowers-102. Flowers-102 contains 8,189 photographs of 102 flower species. The classes are visually similar and the images are larger and more detailed than CIFAR-100, which makes it the hardest of our class-incremental settings. We pool the official training, validation, and test images and split them again per class, holding out a fixed fifth of each class for testing and a tenth of the remainder for validation, so that every class contributes to all three splits. The class order is fixed by the run seed, and the first 100 classes in that order are divided into 20 tasks of five classes in the same way as CIFAR-100. Two classes therefore remain outside the sequence and are never trained on or evaluated, and which two depends on the seed.

PACS. PACS contains 9,991 images of seven object categories drawn from four visual domains, namely photo, art painting, cartoon, and sketch. We use it as a domain-incremental sequence of four tasks, presented in that order, in which all seven classes remain available in every task and only the visual style of the inputs changes. The split is made inside each domain and class, again holding out a fixed fifth for testing and a tenth of the remainder for validation.

In every setting a task is trained only on its own data, with no replay and no revisiting of earlier tasks, and a checkpoint is stored after each task so that the representation can be analysed later.

Preprocessing. PMNIST and CIFAR-100 images are used at their native resolution and are normalized with the statistics of their own training set. Flowers-102 and PACS images are resized so that the shorter side is 96 pixels, center cropped to 96×96, and normalized with the usual ImageNet statistics. During training we apply random cropping with reflection padding and horizontal flipping to the three natural image datasets. No augmentation is used for PMNIST, and no augmentation is used anywhere during evaluation or when features are extracted for the analyses.

Optimization. All transformer runs use AdamW with a learning rate of 5×10^{-4}, weight decay of 0.05, a cosine schedule with ten warmup epochs, label smoothing of 0.1, and gradient clipping at a norm of one. The number of epochs per task is decided by validation accuracy rather than a fixed budget. Each task is trained for at least 20 and at most 200 epochs, training stops after ten epochs without improvement, and the best validation checkpoint is restored before the next task begins. A checkpoint is saved after every task, which is what the analyses read. Standard training and ECR use the same optimizer, schedule, and stopping rule.

Where ECR is applied. ECR acts on the patch token states after every block, which are the activations the objective treats as samples of a response distribution. The pooled representation is excluded, because with fewer samples per batch than feature dimensions its covariance is rank deficient. We use $\gamma = 10^{-3}$ for the response energy term and $\lambda = 5 \times 10^{-4}$ for the log determinant term, with $\epsilon = 10^{-4}$ for covariance regularization. The ECR loss is ramped in over the first ten epochs, and the adaptive gradient balancing rule keeps the norm of the ECR gradient near three percent of the cross entropy gradient.

Seeds. We use five seeds for PMNIST and CIFAR-100 and three seeds for Flowers-102 and PACS, unless a figure states otherwise. Within a run the seed fixes the model initialization, the class order or permutation sequence, and the data splits, and both methods share it, so every comparison is between two networks that saw the same task sequence from the same starting point.

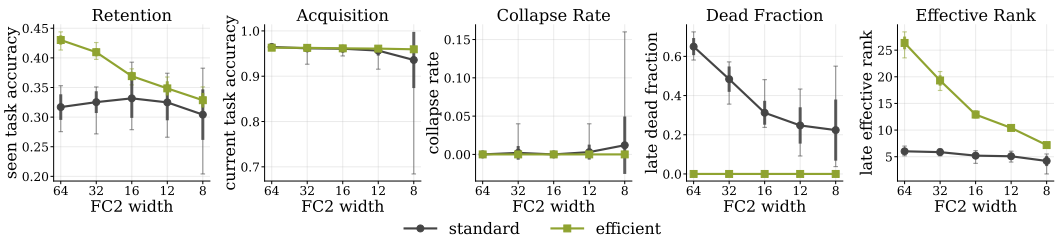

Figure 9: **Effect of representational capacity on continual learning in PMNIST.** Results for a two-hidden-layer network with the first fully connected layer (FC1) width fixed at 128 and the second layer (FC2) varying. Retention is the mean accuracy on previous tasks, whereas acquisition is the mean accuracy on each task after it is learned. Collapse rate is the fraction of tasks whose acquisition accuracy falls below 50%. Dead fraction and effective rank characterize the FC2 representation and are averaged over the final 10 task checkpoints. Curves show the mean across 20 seeds; thick inner error bars indicate ± 1 standard deviation and thin outer whiskers show the observed seed range.

Features used in the analyses. All representation analyses use task 1 data passed through the frozen network saved after the final task. We report results for blocks 1, 3, and 6, which cover the early, middle, and final stages of the encoder, and we use the patch tokens unless the classifier token is named. The decoding analyses fit a linear readout on one set of images and report results on held out images that the readout never saw, and the number of principal components given to that readout is written as k throughout.

C Additional PMNIST Results

We repeat the PMNIST analysis with FC1 fixed at 128 units. Figure 9 shows that ECR continues to maintain higher retention and effective rank while keeping nearly all FC2 units active across widths. Acquisition remains high for both methods over most widths, although standard training becomes less stable as the FC2 width narrows. These results show that the differences observed with FC1 fixed at 256 are also present when the capacity of the preceding layer is reduced.

Figure 10 shows how these differences develop across the task sequence. ECR keeps the number of live FC2 units close to the available layer width throughout training and maintains stable current-task accuracy. Under standard training, the number of live units decreases substantially as tasks accumulate, particularly for the narrower FC2 layers, and some runs show corresponding drops in current-task accuracy. The partial recovery of live units in later tasks is also accompanied by recovery in acquisition, further supporting the relationship between active representational capacity and continued learning.

To examine whether the reduction in training time is consistent across architectures, Fig. 11 shows the number of epochs required for each task separately for each network. Across most architectures, standard training requires progressively more epochs as the task sequence continues, whereas ECR generally converges in fewer epochs and remains more stable across later tasks. This shows that the reduction in training time reported in the main text is not driven by a single architecture.

Sensitivity to ρ. We also examine the sensitivity of ECR to the target gradient ratio ρ. Figure 12 shows that increasing ρ strengthens the effect of ECR on the representation, with higher retention and effective rank and a lower dead fraction. Acquisition remains similar for smaller values of ρ and begins to decrease as the regularization becomes stronger. We therefore use $\rho = 0.1$ in our experiments as an intermediate setting that preserves acquisition while regulating the learned representation.

Sensitivity to γ and λ. We also examine the sensitivity of ECR to the weights of its energy and diversity terms. When varying γ, we keep $\lambda = 5 \times 10^{-4}$ and $\rho = 0.1$ fixed. Figure 13 shows that increasing γ reduces effective rank and increases the dead fraction, while acquisition decreases slightly and retention generally increases. When varying λ, we keep $\gamma = 10^{-3}$ and $\rho = 0.1$ fixed.

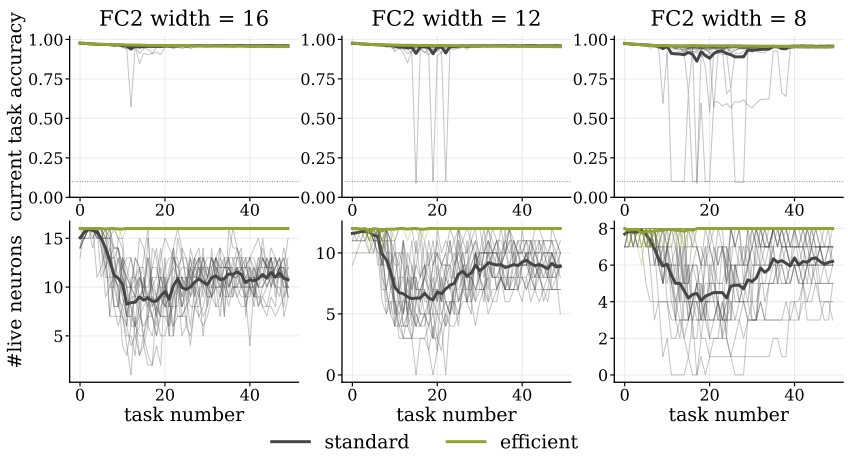

Figure 10: **Task acquisition and representation survival under bottlenecks on PMNIST.** The first fully connected (FC1) layer width is fixed at 128, and columns correspond to FC2 widths. The top row shows accuracy on the current task after learning it, and the bottom row shows the number of live FC2 units at the same point. Thin curves denote individual seeds and thick curves denote their mean. The dotted horizontal line in the accuracy panels indicates chance performance (0.1).

Table 4: **Performance comparison of Nullspace and OOD-based configurations on CIFAR-100 class-incremental learning.** Current task and global accuracy report mean performance over the seen tasks after the final increment, whereas average incremental global accuracy summarizes global performance throughout the learning sequence. Forgetting is computed separately under current task and global evaluation. "0.05 OOD" denotes the use of unlabeled CIFAR-10 images at a sampling ratio of 0.05:1 during training, with CIFAR-10 labels discarded. Results are reported as mean ± standard deviation across 20 seeds.

Configuration	Current Global Acc. (%)	Avg. Incremental Global Acc. (%)	Global Forgetting (pp)	Current Task Acc. (%)	Task Forgetting (pp)
Standard	4.7 ± 0.3	17.3 ± 0.4	85.3 ± 1.3	27.3 ± 2.3	65.6 ± 2.3
Efficient	4.5 ± 0.2	16.1 ± 0.3	89.4 ± 0.8	25.7 ± 1.7	67.1 ± 1.9
Nullspace	28.0 ± 2.6	44.1 ± 2.5	24.8 ± 2.1	86.6 ± 1.5	0.0 ± 0.0
Nullspace + Efficient Head	27.3 ± 1.9	43.8 ± 2.1	21.3 ± 1.6	86.6 ± 1.5	0.0 ± 0.0
EFC (Magistri et al., 2024)	32.9 ± 1.8	48.4 ± 2.1	24.8 ± 1.3	85.0 ± 1.2	1.5 ± 0.3
Nullspace + 0.05 OOD	38.2 ± 1.7	51.7 ± 2.5	15.7 ± 1.3	86.7 ± 1.3	0.0 ± 0.0

Figure 14 shows that increasing λ generally increases effective rank and reduces the dead fraction, although the largest value begins to reverse this trend. Acquisition and retention change relatively little across the tested range. We use $\gamma = 10^{-3}$ and $\lambda = 5 \times 10^{-4}$ in the main experiments.

D Additional CIFAR-100 Results

We first repeat the CIFAR-100 comparison using 20 seeds to evaluate the stability of the results reported in the main text. Table 4 shows the same overall pattern, with Nullspace eliminating task forgetting and the addition of OOD suppression substantially improving global accuracy while maintaining zero task forgetting. The EFC results are obtained from our replication using its publicly available implementation (Magistri et al., 2024).

We next examine the sensitivity of Nullspace + OOD to the amount of external data used during training. Table 5 shows that performance on CIFAR-100 remains relatively stable across OOD-to-training-data ratios from 0.05:1 to 1:1. Current global accuracy, average incremental global accuracy, and current task accuracy change only modestly across this range, while task forgetting remains zero. We observe a similar pattern on Tiny-ImageNet in Table 6, where performance also changes relatively little across the tested OOD ratios.

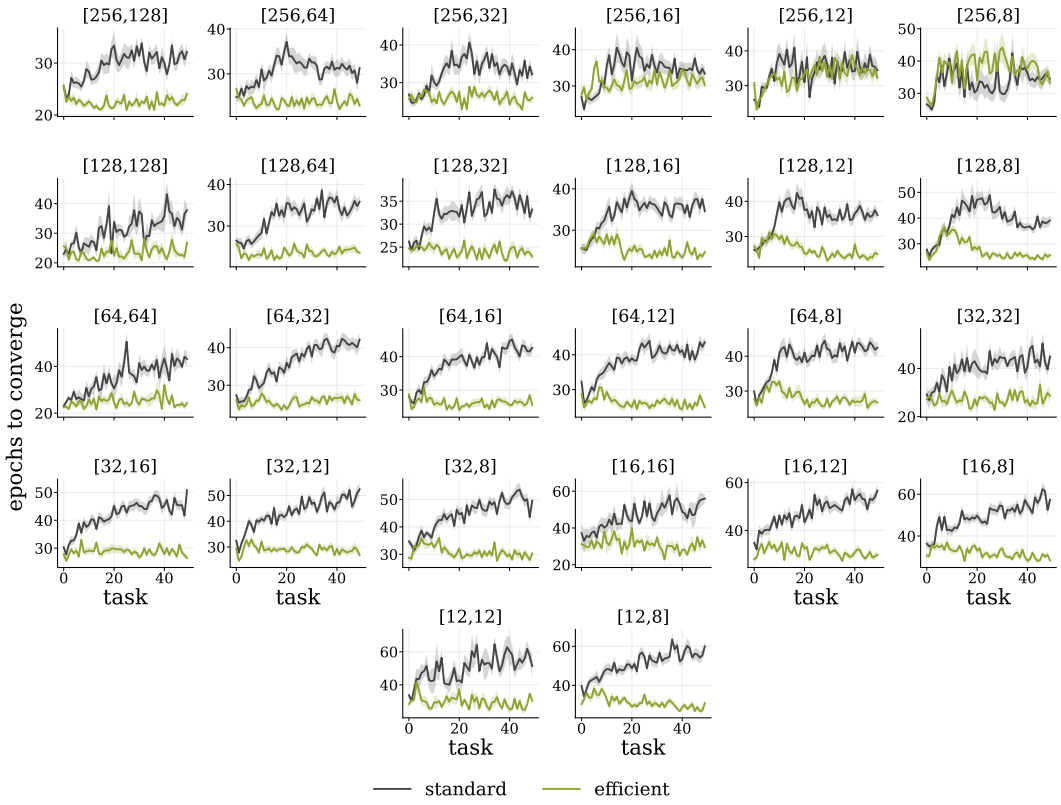

Figure 11: **Training length across PMNIST architectures.** Each panel shows the number of epochs required to converge on each task for one two-hidden-layer architecture, labeled as [FC1, FC2]. Curves show the mean across seeds and shaded regions indicate the standard error. ECR generally requires fewer epochs than standard training as the task sequence progresses. This figure provides the individual architecture results summarized in Fig. 6, where the trajectories are averaged across architectures.

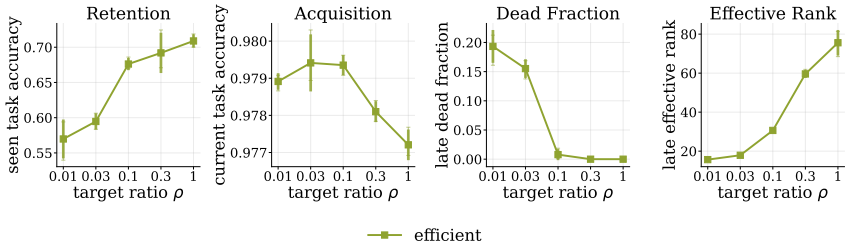

Figure 12: **Sensitivity of ECR to the target gradient ratio ρ on PMNIST.** Results are shown for five sequential PMNIST tasks using a two-hidden-layer network with FC1 width 256 and FC2 width 128. Acquisition, retention, dead fraction and effective rank are shown as ρ varies. Curves show the mean across seeds and shaded regions indicate ± 1 standard deviation. We use $\rho = 0.1$ in the main experiments.

To examine how OOD suppression changes the responses of individual task heads, we analyze the final CIFAR-100 model after all 20 tasks have been learned. For each image, the response of a head is defined as its largest logit. Since OOD training can change the overall scale of the logits, we assign each head a separate post hoc threshold chosen so that 95% of images from its own task exceed that threshold. These thresholds are used only for this analysis and do not affect training or prediction. We then measure how many incorrect heads respond to each CIFAR-100 image and how many respond more strongly than the correct head. We additionally evaluate held-out CIFAR-10 test

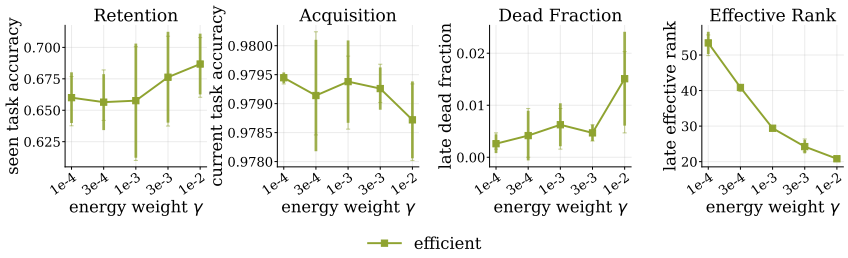

Figure 13: **Sensitivity of ECR to the response-energy weight γ on PMNIST.** Results are shown for five sequential PMNIST tasks using a two-hidden-layer network with FC1 width 256 and FC2 width 128. We vary γ while keeping $\lambda = 5 \times 10^{-4}$ and $\rho = 0.1$ fixed. Retention, acquisition, dead fraction, and effective rank are shown across the tested values of γ. We use $\gamma = 10^{-3}$ in the main experiments.

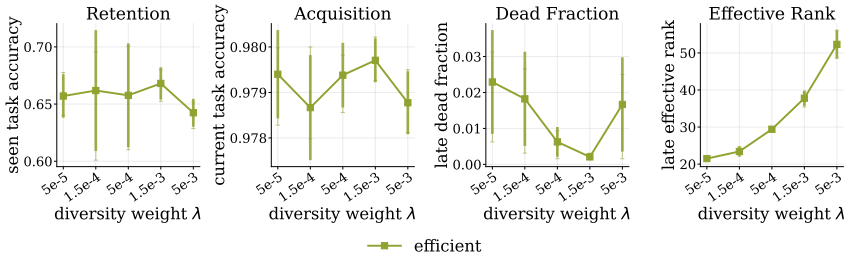

Figure 14: **Sensitivity of ECR to the response-diversity weight λ on PMNIST.** Results are shown for five sequential PMNIST tasks using a two-hidden-layer network with FC1 width 256 and FC2 width 128. We vary λ while keeping $\gamma = 10^{-3}$ and $\rho = 0.1$ fixed. Retention, acquisition, dead fraction, and effective rank are shown across the tested values of λ. We use $\lambda = 5 \times 10^{-4}$ in the main experiments.

images, which are not used during training, and measure how many of the 20 heads respond to each image. This analysis uses 20 paired seeds per configuration, with the same seeds compared with and without OOD training.

E Representation Geometry on PMNIST

We inspect the geometry of the PMNIST representation directly. For each run we take the images of task 1, pass them through the frozen network after all tasks have been learned, and embed the final representation in two dimensions using t-SNE. Points are coloured by digit, and we report the mean fraction of the 15 nearest neighbours that share the same digit. A higher fraction means that images of the same digit remain close together after the rest of the sequence.

Figure 16 shows the pair with the largest difference. With standard training the digits are mixed, whereas with ECR they remain separated into visible groups. Figure 17 recolours the same embedding by the neighbour fraction, which makes the preserved groups easier to see. Figures 18 and 19 repeat both views across five model seeds, so the effect is not specific to one run.

Table 5: **Comparison of OOD sampling ratios for Nullspace + OOD on CIFAR-100 class-incremental learning.** Current task and global accuracy report mean performance over the seen tasks after the final increment, whereas average incremental global accuracy summarizes global performance throughout the learning sequence. Forgetting is computed separately under current task and global evaluation. OOD ratios denote the sampling ratio of unlabeled CIFAR-10 images to training data, with CIFAR-10 labels discarded. Results are reported as mean ± standard deviation across 20 seeds.

Configuration	Current Global Acc. (%)	Avg. Incremental Global Acc. (%)	Global Forgetting (pp)	Current Task Acc. (%)	Task Forgetting (pp)
Nullspace + 0.05 OOD	38.2 ± 1.7	51.7 ± 2.5	15.7 ± 1.3	86.7 ± 1.3	0.0 ± 0.0
Nullspace + 0.1 OOD	38.6 ± 1.7	52.0 ± 2.5	17.1 ± 1.3	86.7 ± 1.3	0.0 ± 0.0
Nullspace + 0.25 OOD	38.9 ± 1.7	52.2 ± 2.6	17.7 ± 0.9	86.9 ± 1.2	0.0 ± 0.0
Nullspace + 0.5 OOD	38.9 ± 1.2	52.0 ± 2.4	17.9 ± 1.4	86.9 ± 1.0	0.0 ± 0.0
Nullspace + 1.0 OOD	38.5 ± 1.8	51.6 ± 2.6	17.7 ± 1.4	86.6 ± 1.2	0.0 ± 0.0

Table 6: **Comparison of OOD sampling ratios for Nullspace + OOD on Tiny-ImageNet class-incremental learning.** Current task and global accuracy report mean performance over the seen tasks after the final increment, whereas average incremental global accuracy summarizes global performance throughout the learning sequence. Forgetting is computed separately under current task and global evaluation. OOD ratios denote the sampling ratio of unlabeled ImageNet images to training data, with ImageNet labels discarded. Results are reported as mean ± standard deviation across 3 seeds.

Configuration	Current Global Acc. (%)	Avg. Incremental Global Acc. (%)	Global Forgetting (pp)	Current Task Acc. (%)	Task Forgetting (pp)
Nullspace + 0.05 OOD	37.6 ± 0.1	51.2 ± 0.2	11.6 ± 0.5	79.0 ± 0.1	0.0 ± 0.0
Nullspace + 0.1 OOD	38.8 ± 0.4	52.3 ± 1.1	13.3 ± 0.6	79.8 ± 0.4	0.0 ± 0.0
Nullspace + 0.25 OOD	38.6 ± 0.8	52.9 ± 1.3	14.0 ± 0.7	80.2 ± 0.2	0.0 ± 0.0

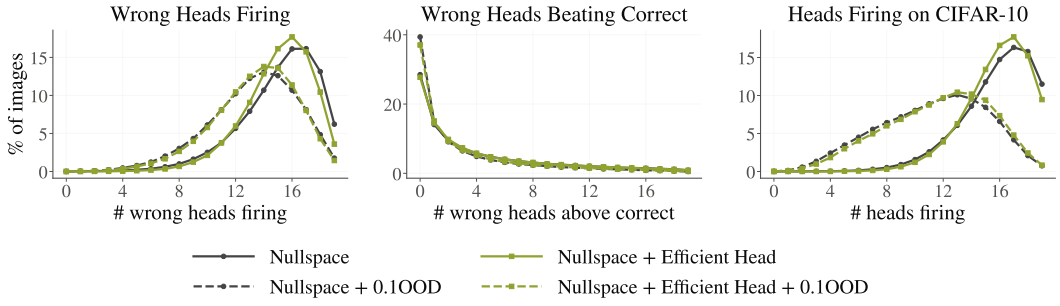

Figure 15: **OOD training reduces responses to inputs outside a task head's own distribution.** CIFAR-100 is divided into 20 tasks, each with its own classification head. For each image, a head's response is defined as its largest logit. Each head is assigned a separate post hoc threshold chosen so that 95% of its own task's test images exceed the threshold, which accounts for differences in response scale across heads and training conditions. The left panel shows, for each CIFAR-100 test image, how many of the 19 incorrect heads exceed their thresholds. The middle panel shows how many incorrect heads produce a larger response than the correct head, where a count of zero means that the correct head produces the strongest response. The right panel shows, for each held-out CIFAR-10 image, how many of the 20 heads exceed their thresholds. Curves report the percentage of images at each count averaged over 20 seeds. Gray curves denote Nullspace and olive curves denote Nullspace + Efficient Head. Solid curves are trained without OOD data and dashed curves use an OOD-to-training-data batch ratio of 0.1:1. A shift toward lower counts indicates stronger suppression of responses outside a head's own task.

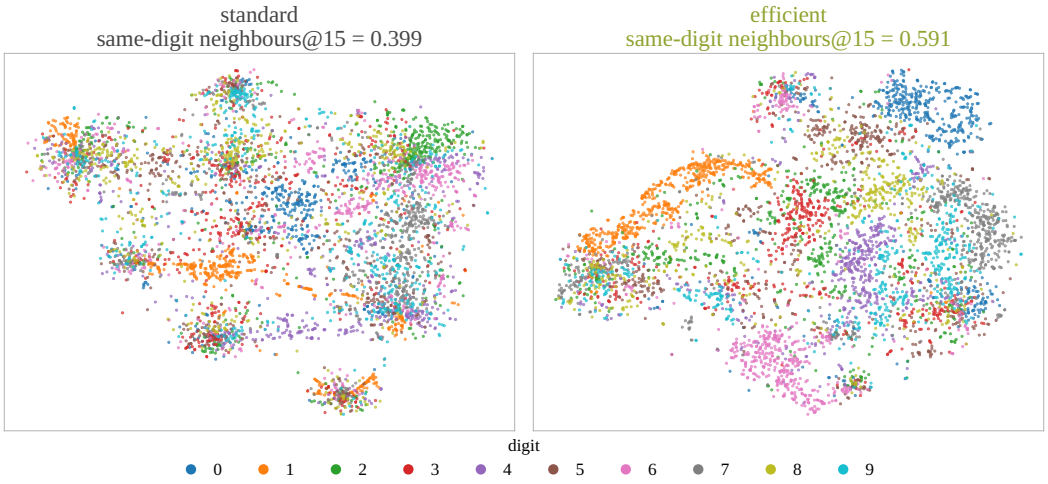

Figure 16: **Task-1 representation after the full PMNIST sequence.** Two-dimensional t-SNE embedding of the final representation for standard training and ECR, coloured by digit. The value above each panel is the mean fraction of the 15 nearest neighbours with the same digit.

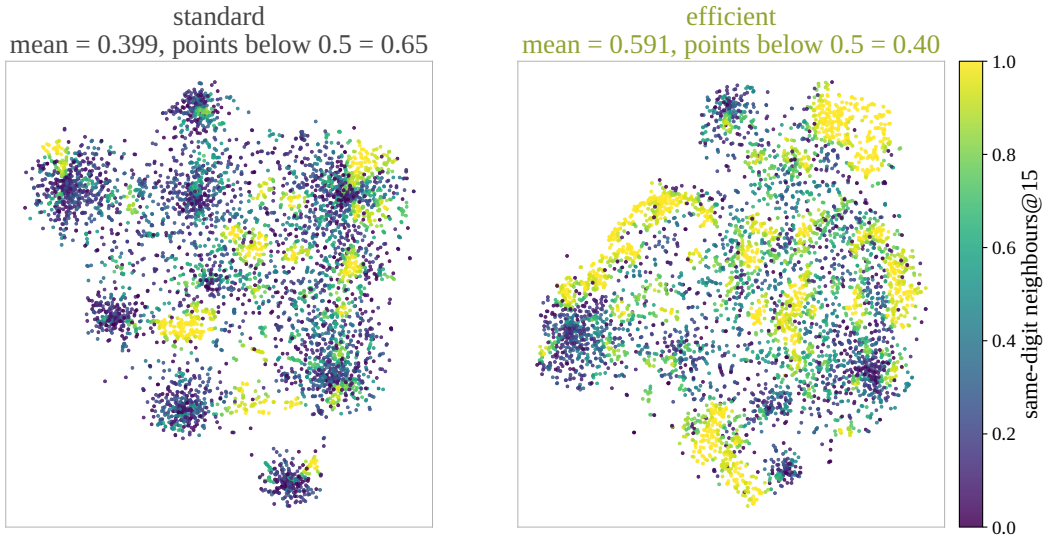

Figure 17: **The same embedding coloured by neighbourhood purity.** Each point is coloured by the fraction of its 15 nearest neighbours that share its digit. Bright points keep their own digit as neighbours; dark points are surrounded by other digits.

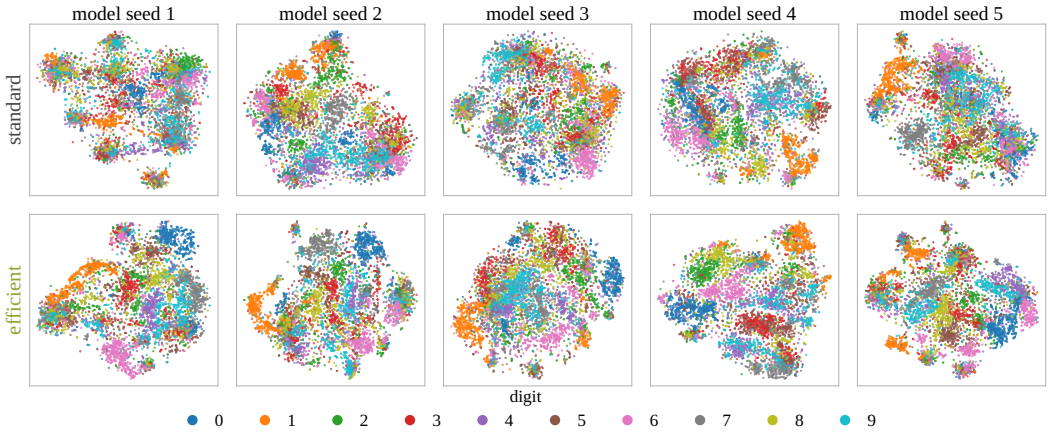

Figure 18: **Task-1 representation across model seeds.** The embedding of Figure 16 for five model seeds, with the task permutation held fixed.

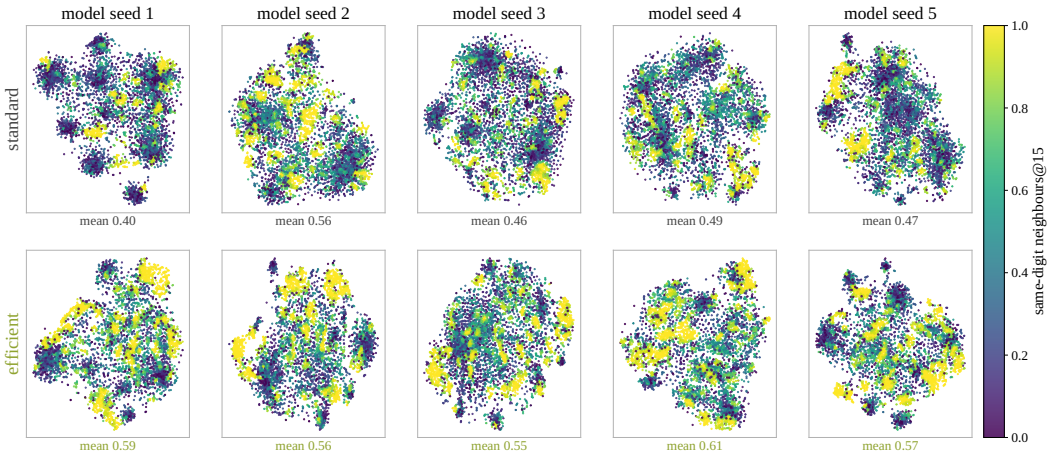

Figure 19: **Neighbourhood purity across model seeds.** The same five pairs recoloured as in Figure 17. The number under each panel is its mean.

F How Tasks Share the Representation

Effective rank says how many directions a representation uses, but not where each task sits inside it. To see that, we take the network at the end of the sequence, pool the features of all tasks, and order the directions of that pooled space from the most used to the least. We then measure, for every task separately, the share of its energy that falls on each direction.

Figure 20 reports the cumulative share against the number of directions counted. With standard training a task is concentrated, and a handful of directions already account for most of it. With ECR the same task is spread, and many more directions are needed to reach the dashed line at 90%. Figure 21 shows the same measurement as a heat map, with one row per task. Tasks that are spread over more directions may interfere less with one another, providing a possible mechanism for the reduced forgetting.

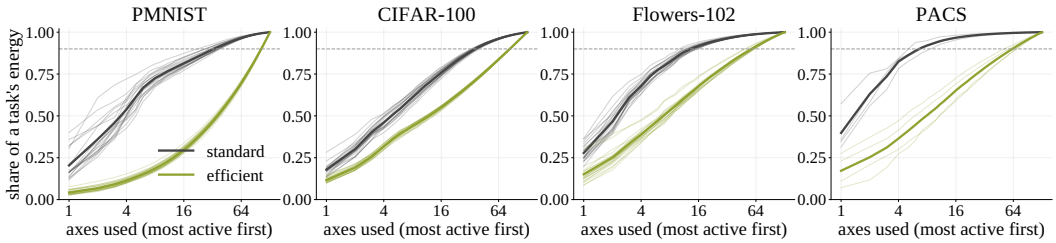

Figure 20: **How much of the representation one task uses.** Cumulative share of a taskś energy against the number of directions counted, most active first. Thin lines are individual tasks and thick lines their mean; the dashed line marks 90%.

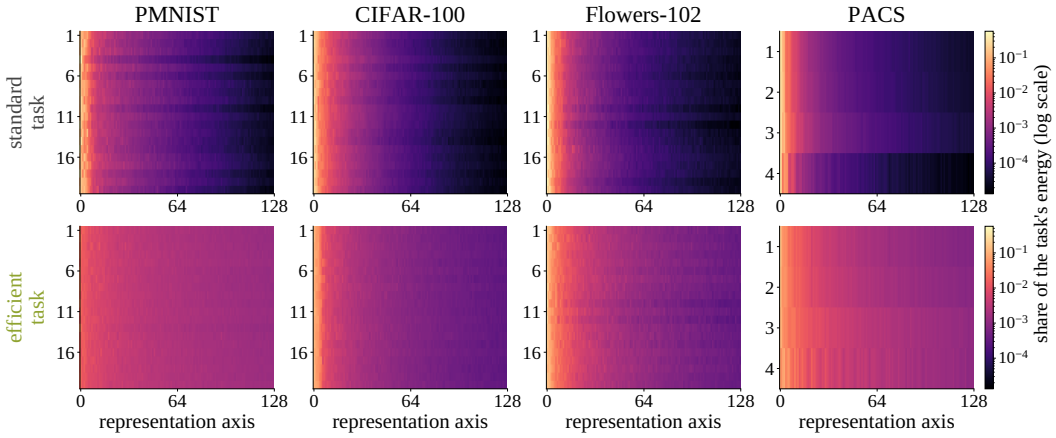

Figure 21: **Where each task sits in the representation.** Share of every taskś energy on every direction, on a logarithmic colour scale, for the final checkpoint at block 6. Rows are tasks and columns are directions, ordered from most to least used.

G PCA COMPRESSION LOSS

Effective rank summarises the representation with a single number. To see the whole spectrum, we apply PCA to the task-1 features of each frozen network and record the fraction of variance that is lost when only the leading k components are kept. A curve that stays high means the variance is spread over many directions; a curve that falls quickly means a few directions carry almost everything. Both methods reach zero at $k = 384$, because the features have 384 dimensions.

Figure 22 reports the curves for the patch tokens, which the ECR objective acts on, and Figure 23 for the classifier token, which it affects only indirectly. ECR loses more variance at every k and in every block, and the gap is larger for the patch tokens. The experiment that varies the task permutation instead of the model seed gives the same ordering.

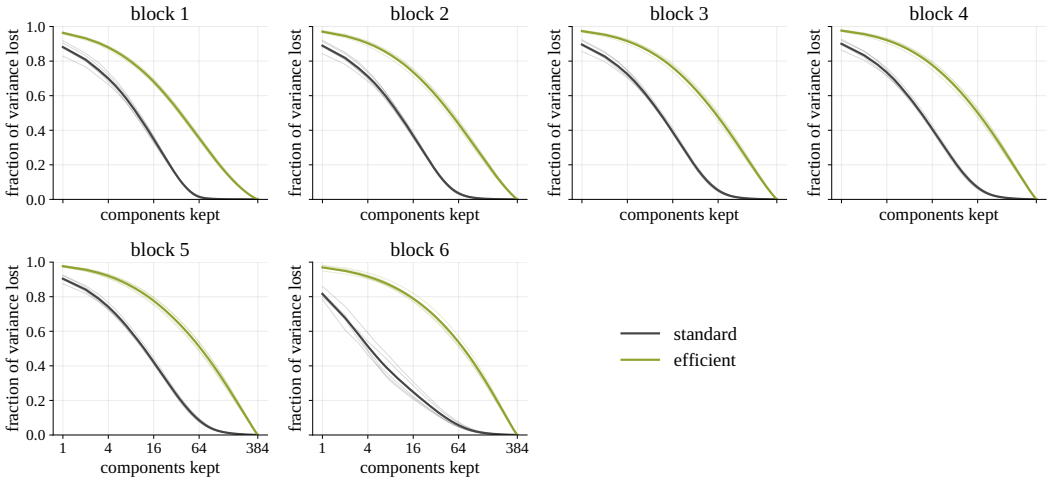

Figure 22: **PCA compression loss for patch tokens.** Fraction of task-1 variance lost when the leading k principal components are kept, for five model seeds. Thin lines are individual runs and thick lines their mean.

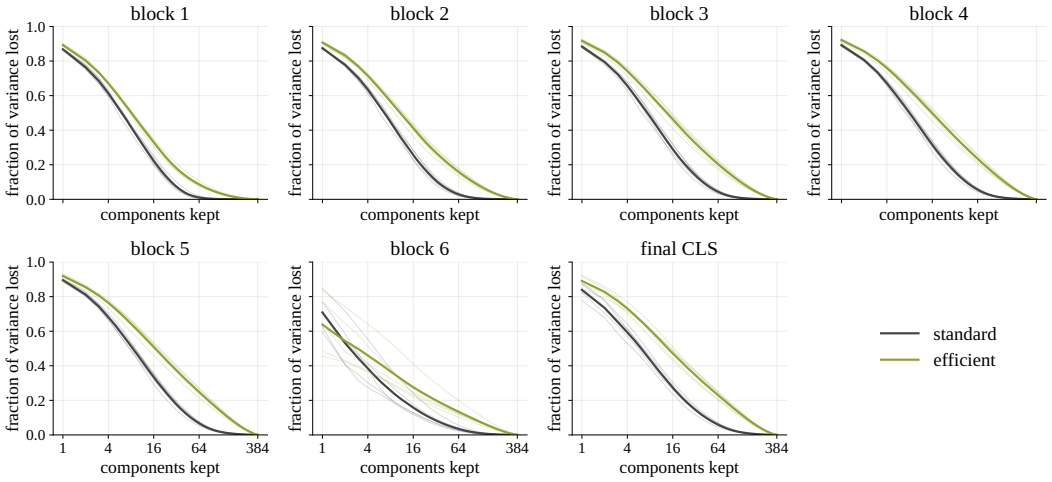

Figure 23: **PCA compression loss for the classifier token.** As in Figure 22, for the token that is read by the classifier.

H RELEARNING THE FORGOTTEN TASK

If a representation keeps more of an old task, that task should be easier to learn again. We take each frozen network at the end of the sequence, fine-tune it on task-1 data, and count the optimizer steps needed to reach a given accuracy. Prediction is restricted to the task-1 classes, so the measure does not depend on the drifted classifier.

The two methods do not always reach the same accuracy when they first learn task 1, and a run with a higher original accuracy would be held to a harder bar. We therefore ask both runs of a pair to reach the same absolute accuracy, defined as a fraction of the weaker run's own accuracy right after it first learned task 1. Figure 24 reports the result. ECR needs fewer steps at every level and on every dataset, and the difference grows as the target rises.

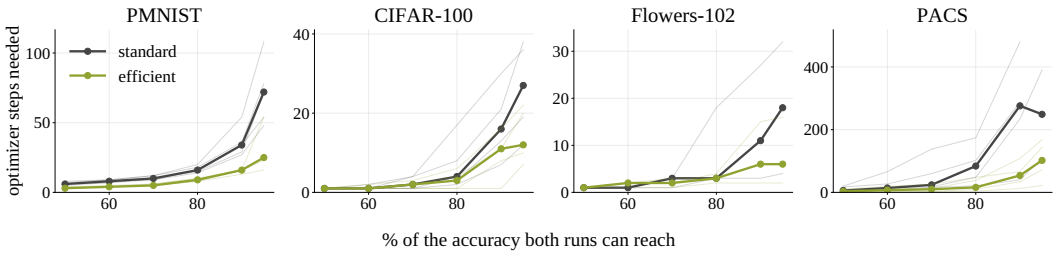

Figure 24: **Steps needed to relearn task 1.** Optimizer steps to reach a target both runs of a pair can reach, expressed as a fraction of the weaker runś original task-1 accuracy. Thin lines are individual runs and thick lines their median.

I PIXEL DECODABILITY

Figure 1 shows how much of an image can be recovered from the representation of a task the network learned first. Here we give the full version of that analysis. Each network is trained on its own task sequence and then frozen, and we return to held-out images of task 1. Every image is split into patches, and each patch becomes a feature vector. We fit a ridge decoder that maps a patch's feature vector back to that patch's own pixels, using only the leading k principal components of the features. A patch is therefore reconstructed from its own vector alone, without information from its neighbours. The decoder is fitted on one set of images and evaluated on held-out images, and k controls how strongly the representation is compressed.

The three sections below vary the budget given to the decoder. The figures show randomly chosen examples, and the accompanying tables report PSNR and SSIM over every held-out image and three seeds, so the examples can be checked against the averages.

I.1 Decoded Examples

Here the decoder is given a fixed number of components. Figures 25 to 30 show six images drawn at random from each held-out set. Unlike Figure 1, whose examples are selected to illustrate the effect, these show the typical case. For PMNIST we show all three blocks; for the other datasets we show block 1 and report the remaining blocks in Table 7.

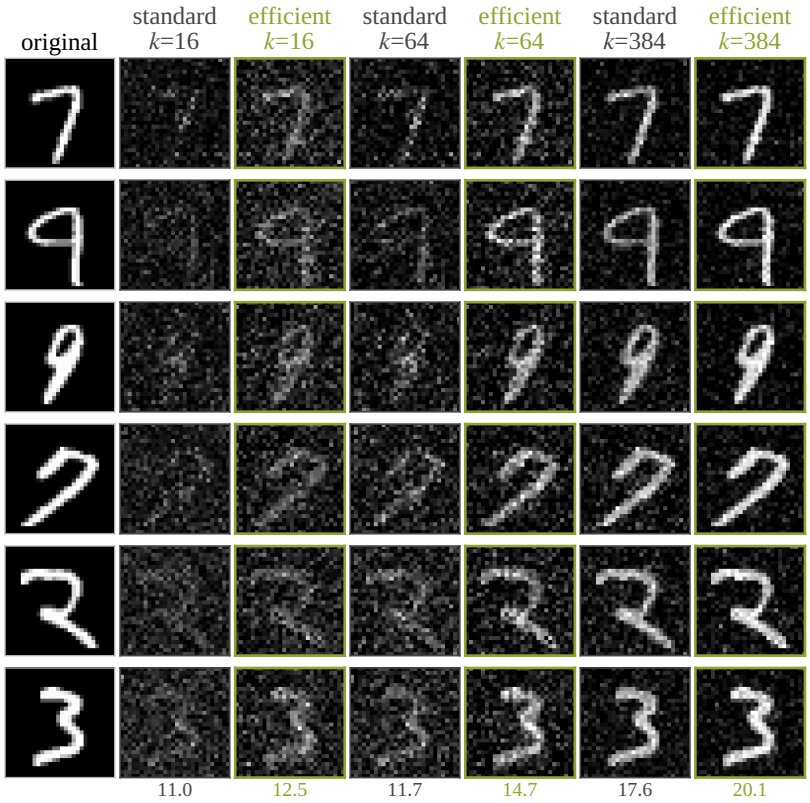

Figure 25: **Decoded PMNIST images from block 1.** Six randomly chosen held-out task-1 images, reconstructed from block-1 patch tokens after the full task sequence. The numbers under the last row are PSNR in dB.

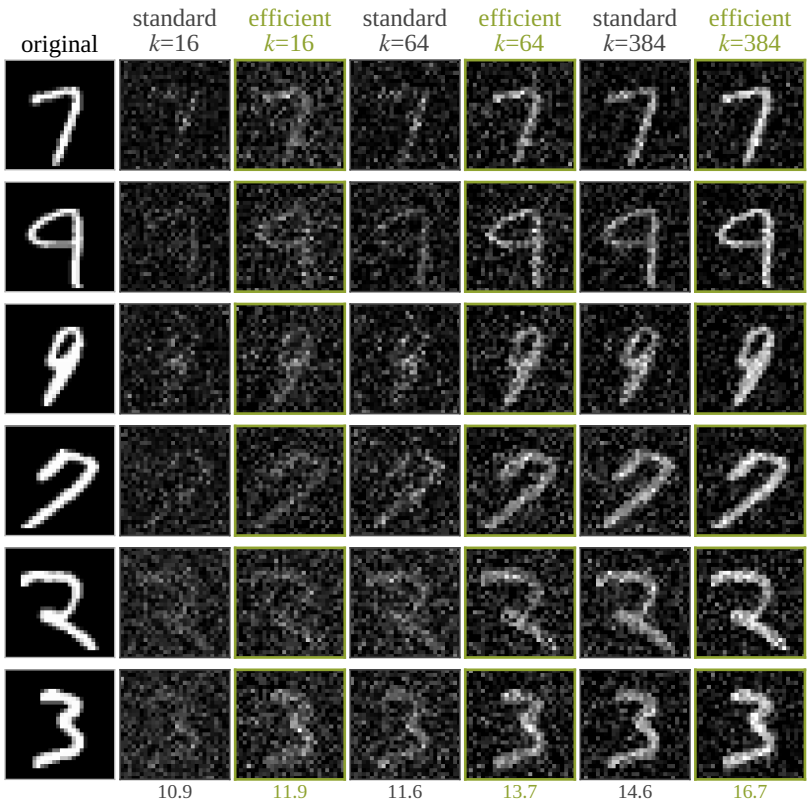

Figure 26: **Decoded PMNIST images from block 3.** As in Figure 25, for PMNIST and block 3.

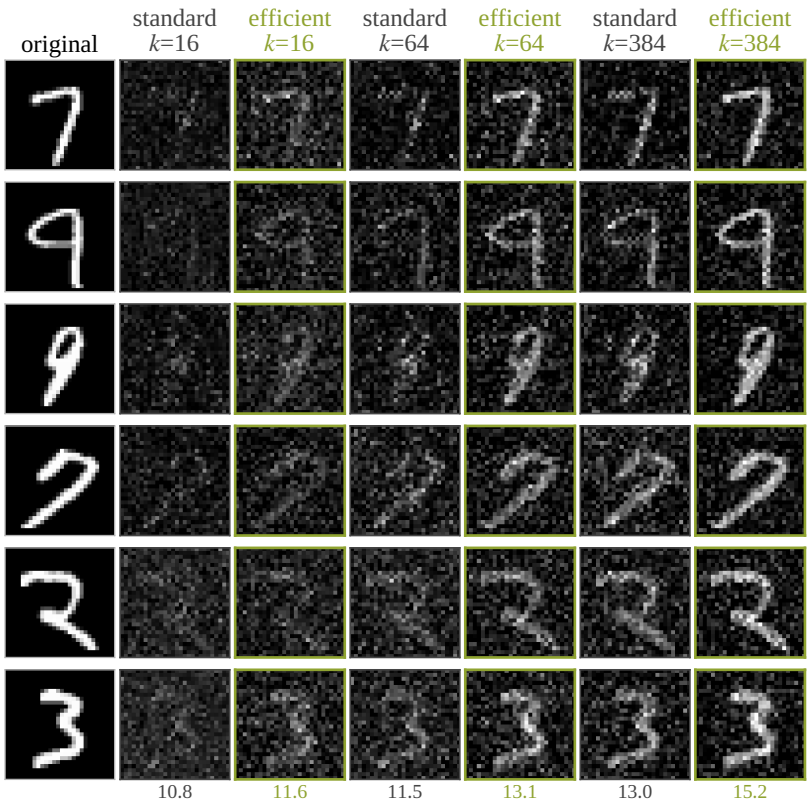

Figure 27: **Decoded PMNIST images from block 6.** As in Figure 25, for PMNIST and block 6.

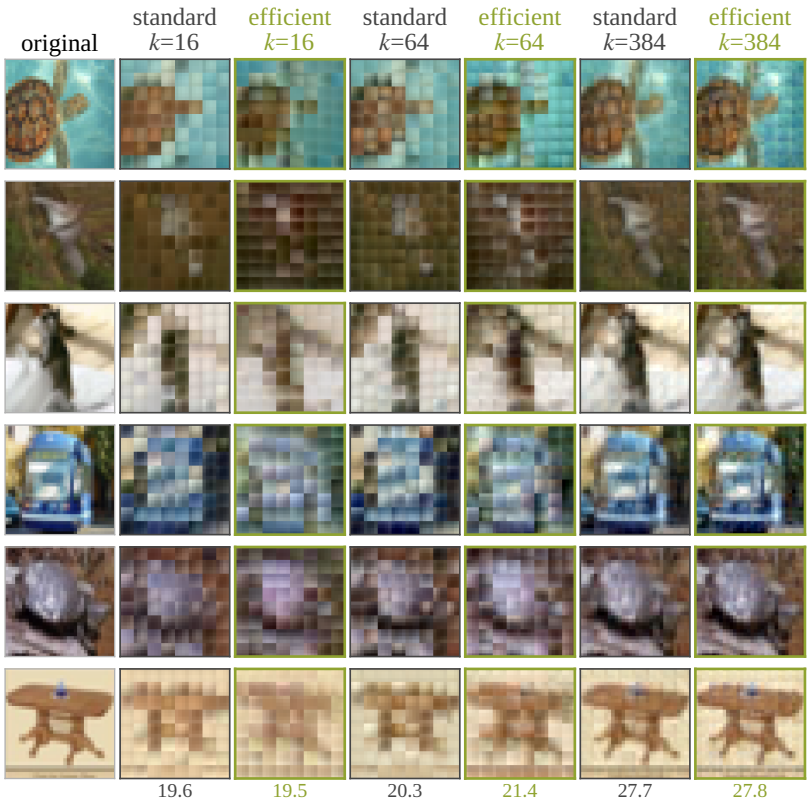

Figure 28: **Decoded CIFAR-100 images from block 1.** As in Figure 25, for CIFAR-100 and block 1.

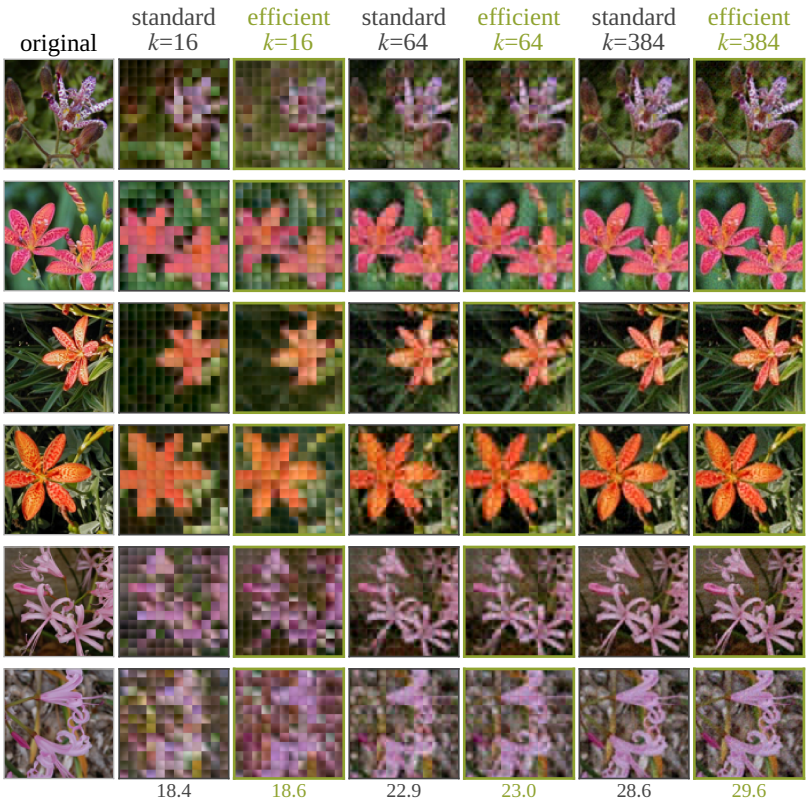

Figure 29: **Decoded Flowers-102 images from block 1.** As in Figure 25, for Flowers-102 and block 1.

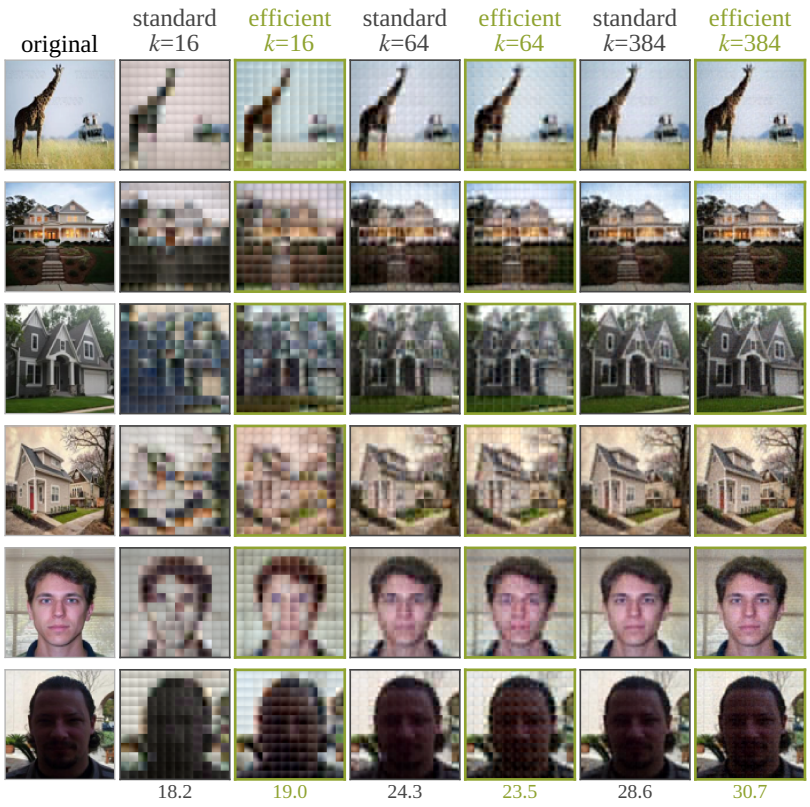

Figure 30: **Decoded PACS images from block 1.** As in Figure 25, for PACS and block 1.

Table 7: **Decoding quality at a fixed budget.** PSNR and SSIM of the decoded held-out task-1 images, where the decoder is restricted to the leading k principal components of the patch features. Both are higher-is-better. Averaged over every held-out image and reported as mean $\pm$ standard deviation across 3 seeds.

	Configuration	PSNR (dB) $k = 16$	PSNR (dB) $k = 64$	PSNR (dB) $k = 384$	SSIM $k = 16$	SSIM $k = 64$	SSIM $k = 384$
PMNIST	Standard, block 1	11.1 ± 0.0	11.7 ± 0.0	17.6 ± 0.5	0.16 ± 0.00	0.25 ± 0.00	0.68 ± 0.01
	Efficient, block 1	12.1 ± 0.1	14.3 ± 0.4	20.3 ± 0.6	0.30 ± 0.02	0.51 ± 0.03	0.75 ± 0.01
	Standard, block 3	11.1 ± 0.0	11.6 ± 0.0	14.2 ± 0.1	0.15 ± 0.01	0.23 ± 0.00	0.51 ± 0.01
	Efficient, block 3	11.6 ± 0.1	13.2 ± 0.4	16.5 ± 0.7	0.23 ± 0.02	0.42 ± 0.04	0.64 ± 0.03
	Standard, block 6	10.9 ± 0.1	11.5 ± 0.0	12.7 ± 0.0	0.13 ± 0.01	0.22 ± 0.00	0.38 ± 0.01
	Efficient, block 6	11.5 ± 0.1	12.9 ± 0.2	15.2 ± 0.5	0.21 ± 0.01	0.39 ± 0.03	0.58 ± 0.03
CIFAR-100	Standard, block 1	17.6 ± 0.7	18.6 ± 0.4	26.3 ± 0.5	0.46 ± 0.07	0.54 ± 0.04	0.86 ± 0.02
	Efficient, block 1	17.2 ± 0.2	19.5 ± 0.6	27.8 ± 0.5	0.44 ± 0.01	0.59 ± 0.02	0.90 ± 0.01
	Standard, block 3	17.1 ± 0.8	18.2 ± 0.5	21.3 ± 0.3	0.43 ± 0.07	0.52 ± 0.05	0.69 ± 0.01
	Efficient, block 3	16.2 ± 0.2	18.0 ± 0.4	21.2 ± 0.5	0.38 ± 0.01	0.51 ± 0.02	0.71 ± 0.01
	Standard, block 6	16.7 ± 0.7	18.0 ± 0.5	19.6 ± 0.1	0.41 ± 0.06	0.50 ± 0.05	0.61 ± 0.02
	Efficient, block 6	15.7 ± 0.3	17.4 ± 0.5	18.9 ± 0.4	0.35 ± 0.03	0.48 ± 0.03	0.60 ± 0.02
Flowers-102	Standard, block 1	18.3 ± 0.6	23.2 ± 0.5	28.4 ± 0.6	0.37 ± 0.00	0.67 ± 0.00	0.87 ± 0.01
	Efficient, block 1	19.5 ± 0.6	23.5 ± 0.5	30.3 ± 0.6	0.44 ± 0.03	0.69 ± 0.01	0.90 ± 0.01
	Standard, block 3	17.3 ± 0.7	20.3 ± 0.5	25.0 ± 0.4	0.33 ± 0.01	0.49 ± 0.01	0.74 ± 0.00
	Efficient, block 3	18.2 ± 0.5	20.7 ± 0.3	24.6 ± 0.6	0.38 ± 0.02	0.52 ± 0.01	0.71 ± 0.01
	Standard, block 6	16.7 ± 0.4	19.5 ± 0.5	23.0 ± 0.6	0.31 ± 0.02	0.44 ± 0.01	0.64 ± 0.01
	Efficient, block 6	17.2 ± 0.5	19.9 ± 0.5	22.5 ± 0.6	0.34 ± 0.01	0.48 ± 0.01	0.61 ± 0.01
PACS	Standard, block 1	17.6 ± 0.1	23.4 ± 0.1	28.6 ± 1.0	0.40 ± 0.02	0.70 ± 0.00	0.88 ± 0.02
	Efficient, block 1	19.2 ± 0.2	23.3 ± 0.3	30.0 ± 0.8	0.46 ± 0.02	0.68 ± 0.02	0.90 ± 0.01
	Standard, block 3	17.0 ± 0.1	19.9 ± 0.2	25.4 ± 0.1	0.38 ± 0.01	0.51 ± 0.01	0.78 ± 0.00
	Efficient, block 3	18.2 ± 0.2	20.7 ± 0.2	25.0 ± 0.4	0.43 ± 0.02	0.55 ± 0.02	0.74 ± 0.02
	Standard, block 6	16.9 ± 0.1	19.2 ± 0.3	23.9 ± 0.2	0.38 ± 0.01	0.47 ± 0.01	0.71 ± 0.01
	Efficient, block 6	17.7 ± 0.2	20.0 ± 0.2	22.9 ± 0.4	0.41 ± 0.03	0.52 ± 0.01	0.66 ± 0.02

I.2 Matched Retained Variance

Giving both methods the same number of components favours the representation that concentrates its variance, because a few components already cover most of it. A fairer budget is information rather than component count. Here each method is given as many components as it needs to retain the same share of its own variance, 50%, 90% and 99%. The number of components used is printed above each column and differs between the methods by construction. Figures 31 to 36 show examples and Table 8 reports the averages, where ECR is ahead on every dataset and in every block.

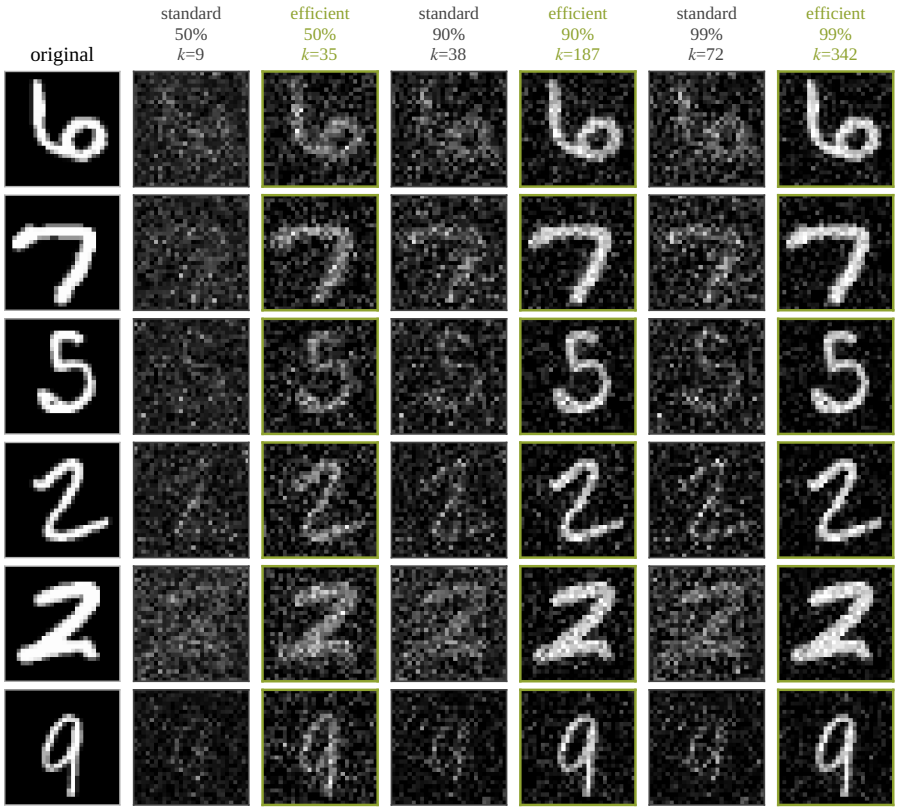

Figure 31: **Matched retained variance, PMNIST block 1.** Six randomly chosen held-out task-1 images decoded when each method keeps 50%, 90% and 99% of its own variance. The value of k above each column is the number of components that method needed.

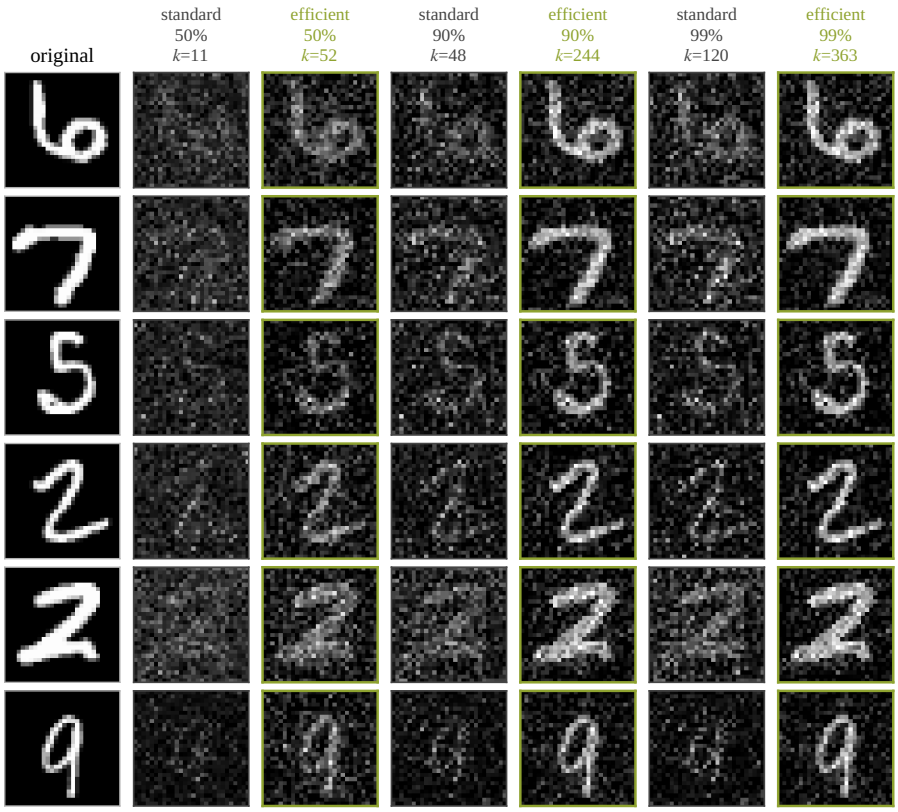

Figure 32: **Matched retained variance, PMNIST block 3.** As in Figure 31, for PMNIST and block 3.

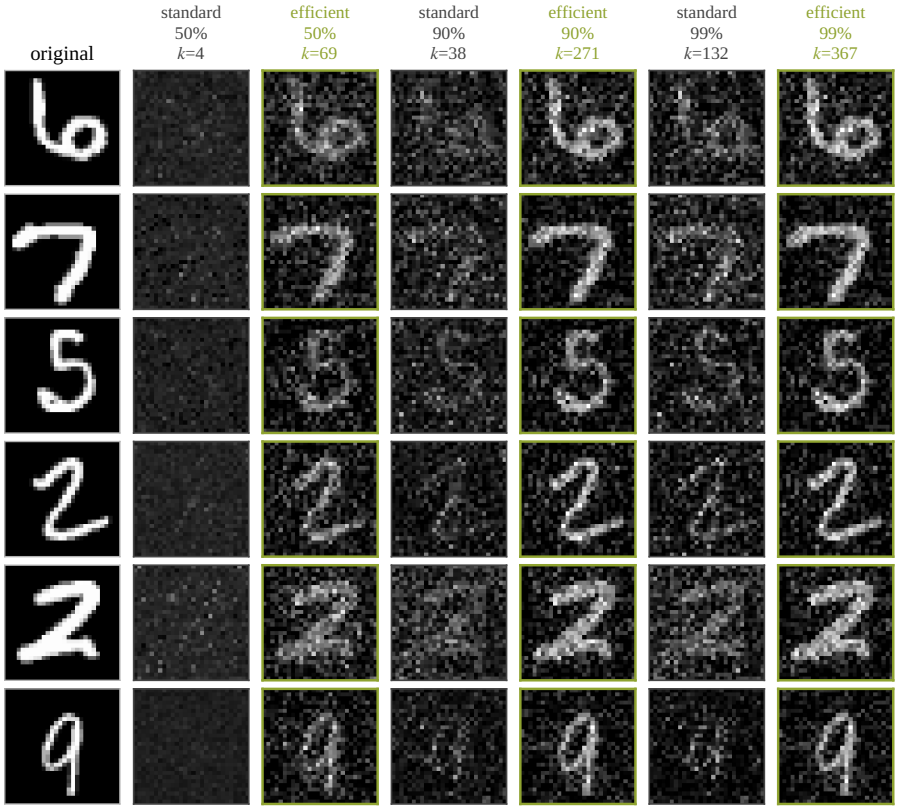

Figure 33: **Matched retained variance, PMNIST block 6.** As in Figure 31, for PMNIST and block 6.

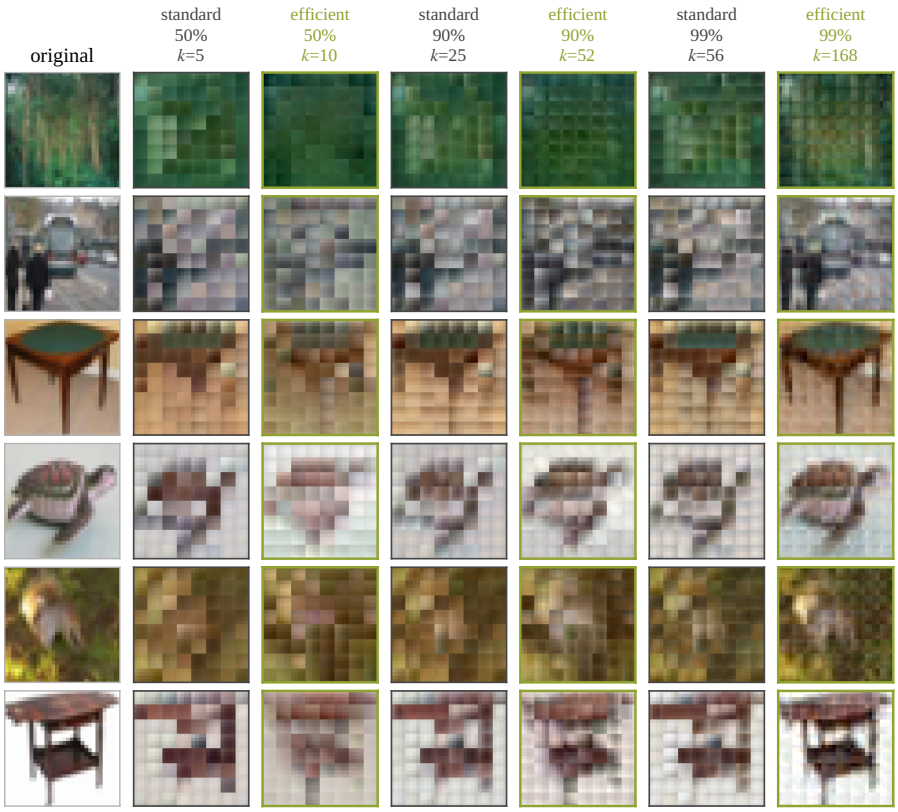

Figure 34: **Matched retained variance, CIFAR-100 block 1.** As in Figure 31, for CIFAR-100 and block 1.

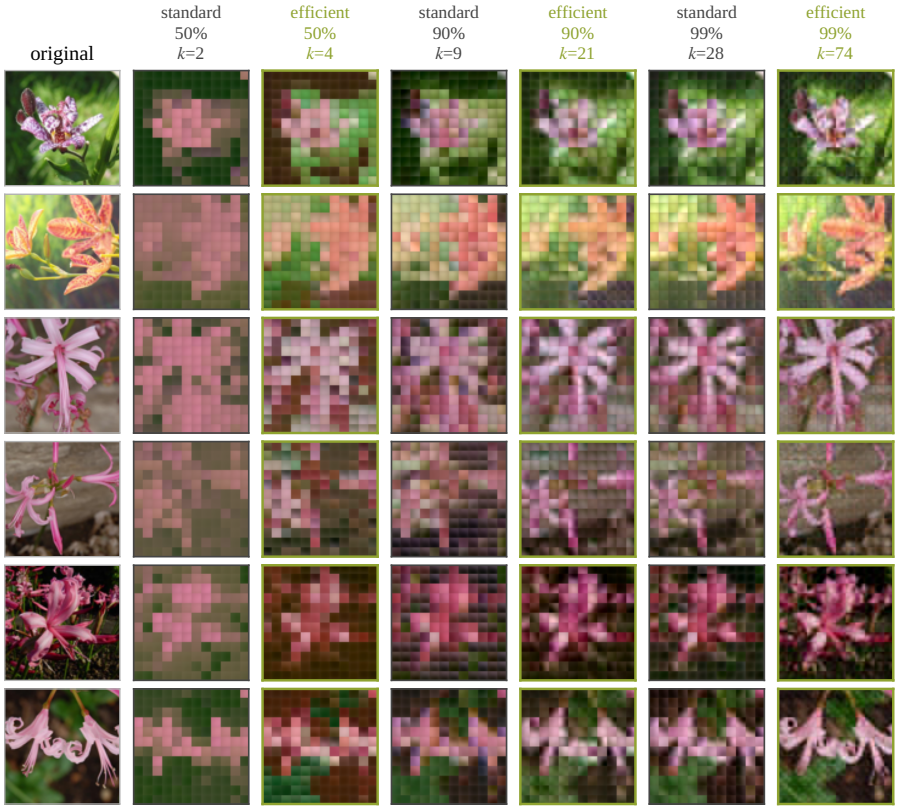

Figure 35: **Matched retained variance, Flowers-102 block 1.** As in Figure 31, for Flowers-102 and block 1.

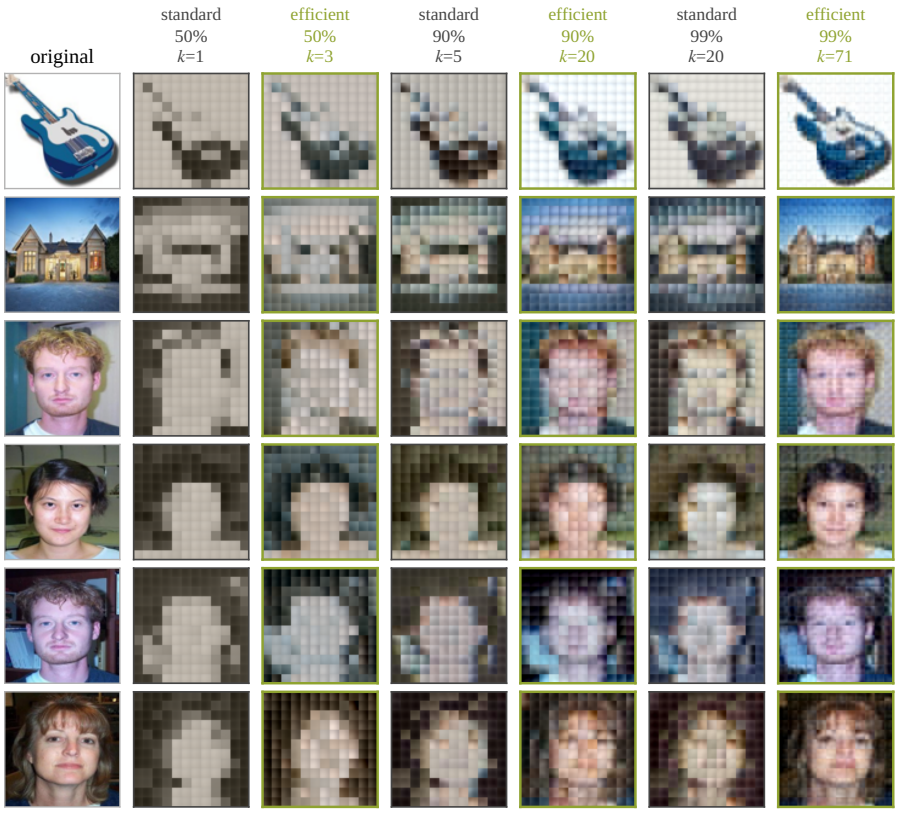

Figure 36: **Matched retained variance, PACS block 1.** As in Figure 31, for PACS and block 1.

Table 8: **Decoding quality at matched retained variance.** Each method is given as many principal components as it needs to retain the same share of its own variance, so the budget is information rather than component count. Averaged over every held-out image and reported as mean ± standard deviation across 3 seeds.

	Configuration	PSNR (dB) 50% of variance	PSNR (dB) 90% of variance	PSNR (dB) 99% of variance	SSIM 50% of variance	SSIM 90% of variance	SSIM 99% of variance
PMNIST	Standard, block 1	11.0 ± 0.0	11.4 ± 0.0	11.7 ± 0.0	0.14 ± 0.00	0.21 ± 0.00	0.25 ± 0.01
	Efficient, block 1	13.2 ± 0.3	18.0 ± 0.6	20.0 ± 0.6	0.43 ± 0.03	0.69 ± 0.02	0.74 ± 0.01
	Standard, block 3	11.0 ± 0.0	11.5 ± 0.0	11.8 ± 0.0	0.14 ± 0.01	0.22 ± 0.00	0.27 ± 0.00
	Efficient, block 3	13.1 ± 0.3	15.7 ± 0.7	16.4 ± 0.7	0.41 ± 0.03	0.60 ± 0.04	0.64 ± 0.03
	Standard, block 6	10.5 ± 0.1	11.3 ± 0.1	11.7 ± 0.1	0.04 ± 0.03	0.19 ± 0.01	0.26 ± 0.01
	Efficient, block 6	13.1 ± 0.3	14.8 ± 0.5	15.2 ± 0.5	0.42 ± 0.03	0.55 ± 0.03	0.58 ± 0.03
CIFAR-100	Standard, block 1	16.4 ± 1.1	17.9 ± 0.5	18.5 ± 0.5	0.38 ± 0.10	0.49 ± 0.05	0.53 ± 0.04
	Efficient, block 1	16.6 ± 0.1	18.9 ± 0.2	24.6 ± 0.3	0.39 ± 0.04	0.56 ± 0.02	0.82 ± 0.01
	Standard, block 3	15.8 ± 1.5	17.7 ± 0.6	18.7 ± 0.4	0.35 ± 0.10	0.48 ± 0.05	0.54 ± 0.04
	Efficient, block 3	16.4 ± 0.2	19.0 ± 0.3	20.9 ± 0.4	0.40 ± 0.03	0.58 ± 0.03	0.69 ± 0.01
	Standard, block 6	14.3 ± 2.0	17.6 ± 0.6	18.6 ± 0.4	0.27 ± 0.11	0.47 ± 0.05	0.54 ± 0.03
	Efficient, block 6	15.7 ± 0.3	18.4 ± 0.4	18.9 ± 0.4	0.35 ± 0.06	0.55 ± 0.02	0.60 ± 0.02
Flowers-102	Standard, block 1	13.8 ± 0.8	17.0 ± 0.9	20.3 ± 0.5	0.24 ± 0.00	0.32 ± 0.02	0.49 ± 0.01
	Efficient, block 1	16.9 ± 0.3	20.0 ± 0.2	23.7 ± 0.0	0.32 ± 0.02	0.48 ± 0.03	0.70 ± 0.03
	Standard, block 3	14.0 ± 0.8	16.7 ± 0.4	20.3 ± 0.5	0.23 ± 0.00	0.31 ± 0.01	0.49 ± 0.01
	Efficient, block 3	16.6 ± 0.4	20.4 ± 0.0	23.2 ± 0.2	0.31 ± 0.01	0.50 ± 0.03	0.65 ± 0.01
	Standard, block 6	13.8 ± 1.4	16.1 ± 0.6	19.7 ± 0.7	0.24 ± 0.01	0.29 ± 0.00	0.45 ± 0.03
	Efficient, block 6	14.6 ± 0.9	19.9 ± 0.1	22.0 ± 0.4	0.25 ± 0.01	0.49 ± 0.03	0.59 ± 0.00
PACS	Standard, block 1	15.4 ± 0.2	16.5 ± 0.5	19.3 ± 1.2	0.30 ± 0.00	0.36 ± 0.03	0.48 ± 0.06
	Efficient, block 1	15.6 ± 0.7	18.6 ± 1.5	23.2 ± 1.5	0.32 ± 0.02	0.44 ± 0.05	0.67 ± 0.07
	Standard, block 3	15.1 ± 0.4	16.2 ± 0.2	19.3 ± 0.5	0.30 ± 0.01	0.35 ± 0.02	0.48 ± 0.03
	Efficient, block 3	15.8 ± 0.8	19.3 ± 1.3	22.8 ± 0.9	0.33 ± 0.02	0.48 ± 0.05	0.65 ± 0.04
	Standard, block 6	14.6 ± 0.8	15.8 ± 0.4	18.9 ± 0.6	0.28 ± 0.01	0.32 ± 0.02	0.46 ± 0.03
	Efficient, block 6	15.4 ± 0.6	19.2 ± 1.2	22.0 ± 0.3	0.31 ± 0.02	0.48 ± 0.05	0.62 ± 0.01

I.3 Noise in the Features

A representation that spreads a task over many directions should also lose less when its features are disturbed. We repeat the decoding with a fixed budget of 64 components and add Gaussian noise to the features before decoding, with σ giving the size of the noise relative to the features themselves. Figures 37 to 42 show examples and Table 9 reports the averages. Standard training degrades quickly, and at $\sigma = 1$ its reconstructions are close to noise, while ECR still recovers the shape of the object. This is a stress test of the representation and not a condition seen during training.

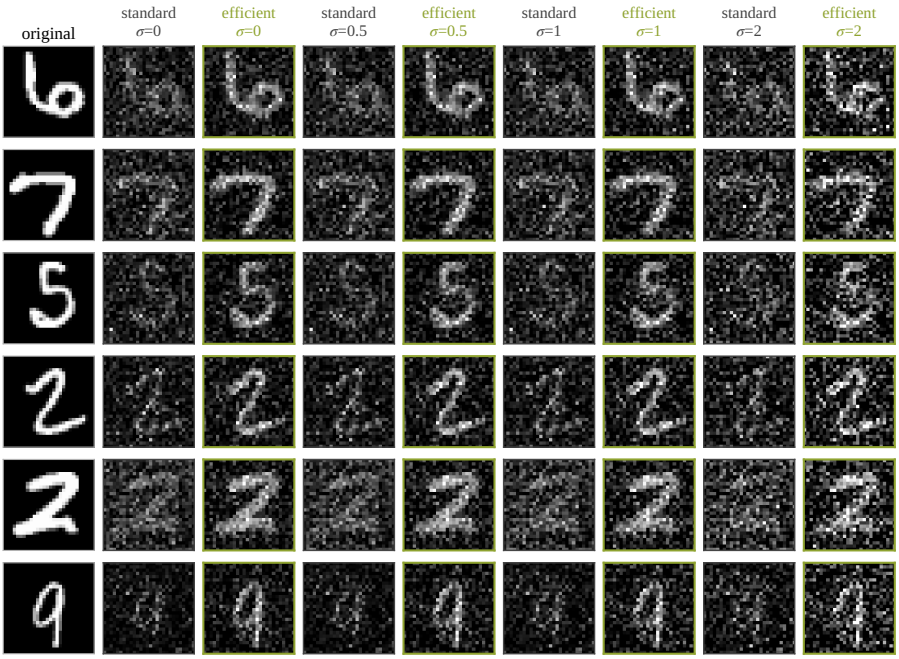

Figure 37: **Noise in the features, PMNIST block 1.** Six randomly chosen held-out task-1 images decoded from 64 components after Gaussian noise of size σ is added to the features.

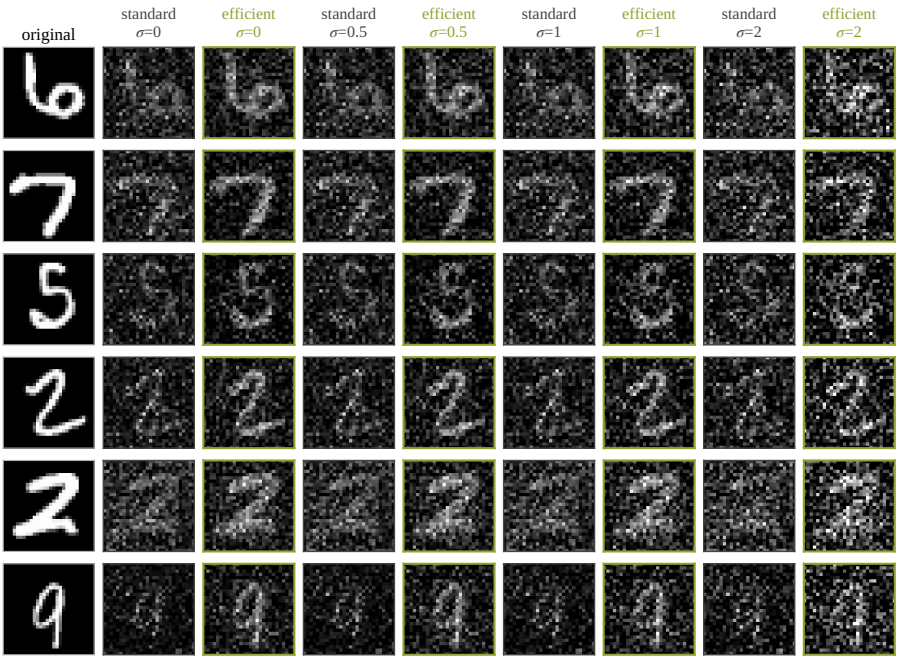

Figure 38: **Noise in the features, PMNIST block 3.** As in Figure 37, for PMNIST and block 3.

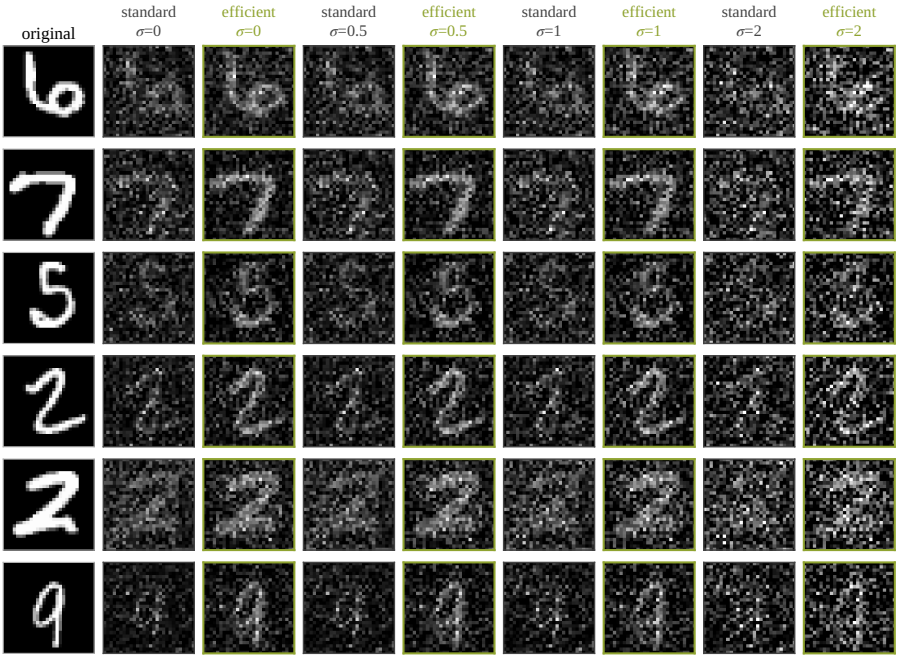

Figure 39: **Noise in the features, PMNIST block 6.** As in Figure 37, for PMNIST and block 6.



Figure 40: **Noise in the features, CIFAR-100 block 1.** As in Figure 37, for CIFAR-100 and block 1.



Figure 41: **Noise in the features, Flowers-102 block 1.** As in Figure 37, for Flowers-102 and block 1.

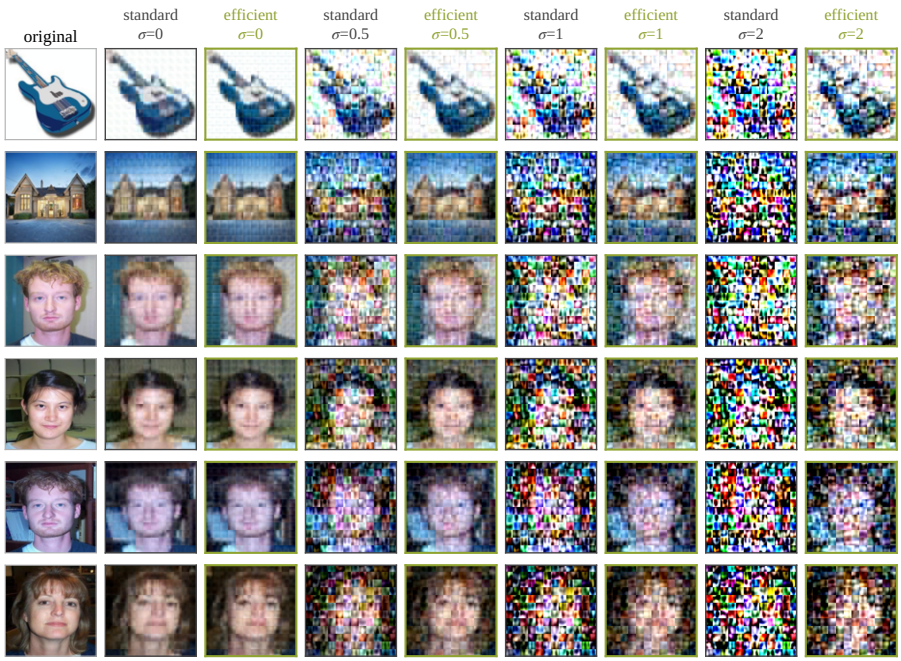

Figure 42: **Noise in the features, PACS block 1.** As in Figure 37, for PACS and block 1.

Table 9: **Decoding quality with noise in the features.** A fixed budget of 64 principal components, with Gaussian noise of size σ added to the features before decoding, relative to the scale of the features themselves. Averaged over every held-out image and reported as mean $\pm$ standard deviation across 3 seeds.

	Configuration	PSNR (dB) $\sigma = 0$	PSNR (dB) $\sigma = 0.5$	PSNR (dB) $\sigma = 1$	PSNR (dB) $\sigma = 2$	SSIM $\sigma = 0$	SSIM $\sigma = 0.5$	SSIM $\sigma = 1$	SSIM $\sigma = 2$
PMNIST	Standard, block 1	11.7 ± 0.0	11.6 ± 0.0	11.4 ± 0.0	10.8 ± 0.0	0.25 ± 0.00	0.24 ± 0.00	0.23 ± 0.00	0.21 ± 0.00
	Efficient, block 1	14.3 ± 0.4	14.0 ± 0.4	13.3 ± 0.3	11.5 ± 0.1	0.51 ± 0.03	0.50 ± 0.03	0.47 ± 0.03	0.39 ± 0.02
	Standard, block 3	11.6 ± 0.0	11.5 ± 0.0	11.4 ± 0.0	10.9 ± 0.0	0.23 ± 0.00	0.23 ± 0.00	0.22 ± 0.00	0.20 ± 0.00
	Efficient, block 3	13.2 ± 0.4	13.0 ± 0.3	12.5 ± 0.3	10.9 ± 0.1	0.42 ± 0.04	0.41 ± 0.04	0.39 ± 0.03	0.32 ± 0.02
	Standard, block 6	11.5 ± 0.0	11.4 ± 0.0	11.1 ± 0.1	10.3 ± 0.2	0.22 ± 0.00	0.21 ± 0.00	0.20 ± 0.00	0.17 ± 0.00
	Efficient, block 6	12.9 ± 0.2	12.7 ± 0.2	12.2 ± 0.2	10.6 ± 0.1	0.39 ± 0.03	0.38 ± 0.03	0.36 ± 0.03	0.29 ± 0.02
CIFAR-100	Standard, block 1	18.6 ± 0.4	18.0 ± 0.5	16.7 ± 0.7	13.9 ± 0.9	0.54 ± 0.04	0.49 ± 0.05	0.41 ± 0.06	0.28 ± 0.06
	Efficient, block 1	19.5 ± 0.6	18.8 ± 0.4	17.4 ± 0.0	14.6 ± 0.4	0.59 ± 0.02	0.54 ± 0.02	0.46 ± 0.03	0.31 ± 0.04
	Standard, block 3	18.2 ± 0.5	17.8 ± 0.5	16.7 ± 0.7	14.3 ± 1.0	0.52 ± 0.05	0.48 ± 0.05	0.41 ± 0.06	0.29 ± 0.06
	Efficient, block 3	18.0 ± 0.4	17.6 ± 0.3	16.5 ± 0.2	14.0 ± 0.1	0.51 ± 0.02	0.47 ± 0.02	0.40 ± 0.02	0.28 ± 0.02
	Standard, block 6	18.0 ± 0.5	17.5 ± 0.6	16.3 ± 0.7	13.7 ± 1.0	0.50 ± 0.05	0.46 ± 0.05	0.39 ± 0.06	0.26 ± 0.06
	Efficient, block 6	17.4 ± 0.5	16.8 ± 0.4	15.6 ± 0.2	12.9 ± 0.1	0.48 ± 0.03	0.43 ± 0.03	0.35 ± 0.03	0.23 ± 0.02
Flowers-102	Standard, block 1	23.2 ± 0.5	16.9 ± 0.4	12.5 ± 0.3	8.8 ± 0.2	0.67 ± 0.00	0.31 ± 0.01	0.16 ± 0.01	0.07 ± 0.00
	Efficient, block 1	23.5 ± 0.5	21.1 ± 0.2	17.8 ± 0.5	13.5 ± 0.6	0.69 ± 0.01	0.52 ± 0.03	0.35 ± 0.04	0.19 ± 0.03
	Standard, block 3	20.3 ± 0.5	18.1 ± 0.3	15.1 ± 0.3	11.1 ± 0.3	0.49 ± 0.01	0.35 ± 0.01	0.23 ± 0.02	0.12 ± 0.01
	Efficient, block 3	20.7 ± 0.3	20.0 ± 0.3	18.4 ± 0.2	15.3 ± 0.1	0.52 ± 0.01	0.47 ± 0.02	0.38 ± 0.03	0.25 ± 0.02
	Standard, block 6	19.5 ± 0.5	17.2 ± 0.6	14.2 ± 0.8	10.4 ± 0.8	0.44 ± 0.01	0.31 ± 0.02	0.19 ± 0.03	0.10 ± 0.02
	Efficient, block 6	19.9 ± 0.5	19.1 ± 0.4	17.4 ± 0.3	14.2 ± 0.4	0.48 ± 0.01	0.43 ± 0.02	0.34 ± 0.02	0.21 ± 0.02
PACS	Standard, block 1	23.4 ± 0.1	15.9 ± 1.3	11.5 ± 1.1	8.2 ± 0.7	0.70 ± 0.00	0.31 ± 0.06	0.16 ± 0.04	0.07 ± 0.02
	Efficient, block 1	23.3 ± 0.3	20.1 ± 1.6	16.5 ± 2.2	12.2 ± 2.1	0.68 ± 0.02	0.49 ± 0.08	0.33 ± 0.09	0.17 ± 0.06
	Standard, block 3	19.9 ± 0.2	16.6 ± 0.4	13.0 ± 0.5	9.3 ± 0.4	0.51 ± 0.01	0.33 ± 0.03	0.19 ± 0.02	0.09 ± 0.01
	Efficient, block 3	20.7 ± 0.2	19.4 ± 0.6	17.2 ± 1.5	13.6 ± 1.9	0.55 ± 0.02	0.47 ± 0.03	0.36 ± 0.06	0.22 ± 0.06
	Standard, block 6	19.2 ± 0.3	16.2 ± 0.3	12.8 ± 0.4	9.2 ± 0.3	0.47 ± 0.01	0.31 ± 0.02	0.19 ± 0.02	0.09 ± 0.01
	Efficient, block 6	20.0 ± 0.2	18.8 ± 0.4	16.8 ± 1.0	13.3 ± 1.3	0.52 ± 0.01	0.45 ± 0.02	0.35 ± 0.04	0.21 ± 0.04