CONTINUAL LEARNING WITH GROWING SELF-ORGANIZING MAPS AND SYNTHETIC REPLAY

Anonymous authors
Paper under double-blind review

ABSTRACT

This work presents a generative continual learning framework based on growing self-organizing maps (GSOMs) that are augmented with learned distributional statistics as well as encoder–decoder models for class-incremental learning. The proposed approach enables exemplar-free replay using distributional statistical memory, which eliminates the need to store raw data. Each GSOM unit maintains its own mean, variance, and covariance estimates, which are subsequently used to generate synthetic samples for replay; in encoder–decoder configurations, these samples are then decoded back into the input space (via ancestral sampling) for subsequent training. Our method is fully unsupervised, as it does not rely on explicit task boundaries or class labels during training. Results across multiple benchmarks show that the proposed approach achieves performance competitive even with supervised state-of-the-art memory-based methods while consistently outperforming memory-free approaches. In several settings, our framework matches or exceeds existing baselines, particularly in challenging single-class incremental scenarios. We also provide baseline results for single-class incremental TinyImageNet and MiniImageNet, offering a useful reference for future work. This work highlights the effectiveness of an unsupervised, adaptive, topology-driven neural form of statistical replay as a scalable, flexible approach to continual learning.

1 INTRODUCTION

Computational intelligent systems operating in real-world environments are often exposed to continuous streams of data, where the underlying data distribution may change over time. Continual learning (CL), or lifelong learning (Thrun, 1998), seeks to incorporate new information from such streams while retaining previously acquired knowledge. A key challenge in continual learning is catastrophic forgetting French (1999), where artificial neural networks (ANNs) tend to overwrite previously acquired knowledge when trained on new data. This problem is closely tied to the stability–plasticity dilemma Abraham & Robins (2005), which governs the trade-off between learning new information and preserving prior knowledge (encoded as internal representations). This challenge is particularly pronounced in class-incremental learning (CIL), where predictions must discriminate among all previously observed classes without task identity at inference time (van de Ven et al., 2022).

Standard CL benchmarks assume clearly defined task boundaries, with new tasks introducing several classes simultaneously, as in Split-MNIST or Split-CIFAR (Yang et al., 2025; Zhou et al., 2024a; Wickramasinghe et al., 2024). However, such assumptions rarely hold in real-world data streams, where task boundaries are often unknown or ill-defined. As a result, recent work has shifted to task-free CL (TFCL), where data is presented as a continuous stream without explicit task boundaries or identities (Aljundi et al., 2019b; Gunasekara et al., 2023; Ororbia, 2021). Early work by Aljundi et al. (2019b) proposed one of the first TFCL formulations, extending importance-weight regularization to online settings. Memory-based task-free methods, e.g., GMED (Jin et al., 2021) and ODDL (Ye & Bors, 2022b), demonstrated that replay can be strengthened not only by selecting past samples but also by actively "editing" stored examples to make them more useful for future replay. Other more recent memory-based schemes include neural Dirichlet mixture processes (Lee et al., 2020) and continuous organization and maintenance of samples/prototypes per class (Lange & Tuytelaars, 2021). Although effective, memory-based methods underscore a broader reliance on stored samples and replay heuristics, raising concerns regarding memory growth, privacy, and long-term scalability. More recently, memory-free task-free approaches have been explored that avoid explicit replay buffers

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.

and instead rely on implicit knowledge retention and adaptive optimization strategies (Michel et al., 2025). Nevertheless, such methods often struggle to maintain stable representations over long data streams due to the absence of explicit mechanisms to preserve past information.

Most CL research has focused on deep ANNs (Zhou et al., 2024b); however, self-organizing maps (SOMs) (Kohonen, 1990), which come from a broader class of competitive neural models (Ororbia, 2021), offer a topology-preserving, unsupervised alternative with inherently local updates that can reduce interference between past and new knowledge. Prior SOM-based CL approaches, such as SOMLP (Bashivan et al., 2019), DendSOM (Pinitas et al., 2021), and c-SOM (Vaidya et al., 2021), demonstrate this potential yet are limited by fixed capacity or lack adaptive, scalable replay mechanisms. To address these limitations for task-free settings, we propose a *growing self-organizing map (GSOM) framework* for task-agnostic continual learning in the context of a class-sequential data stream. Unlike conventional approaches, GSOM provides an adaptive, topology-preserving structure that expands incrementally as new patterns arrive (Alahakoon et al., 2000), organizing its latent space into localized regions, which reduces interference (or neural cross-talk (French, 1999; Ororbia, 2021)). Notably, our model "grows" new portions of its topology via a criterion we develop based on statistical "surprisal", which is used to determine if incoming samples are poorly explained, and further provides visual interpretability of a model's quality during and after training (see Appendix L). The GSOM's combination of localized representation and probabilistic growth enables it to stably and continually adapt to data complexity without relying on task boundaries or explicitly-stored patterns. Furthermore, we integrate our GSOM with encoder-decoder structures (specifically convolutional VAEs (Kingma & Welling, 2019) and pretrained visual encoders such as CLIP (Radford et al., 2021)) in order to handle complex data patterns, e.g., natural images.

The main contributions of this work are *i)* a GSOM-centric CL framework that combines generative modeling with topology-preserving clustering to enable interpretable, unsupervised learning from data streams; *ii)* a surprisal-driven growth mechanism for the GSOM that dynamically expands the internal map when current representations become insufficient to explain incoming samples (meaning the topology expands as needed, without task boundary triggers); and, *iii)* a generative replay scheme that stores summary statistics (mean, variance, and covariance) per GSOM unit, enabling rehearsal without buffers or data storage. We evaluate three variants of our model framework on MNIST, CIFAR-10/100, TinyImageNet, and MiniImageNet, under task-boundary-free protocols, and show improved or competitive performance with prior CL methodology.

2 METHODOLOGY

Self-organizing maps (SOMs) project data onto a 2-D grid of prototypes while preserving topology (Kohonen, 1990), with each sample input updating its best-matching unit (BMU) prototype and that unit's neighbors, with the update reducing as grid distance increases.

2.1 GROWING SELF-ORGANIZING MAPS

Conventional SOMs require the topological grid size to be specified in advance, whereas growing SOMs (GSOMs) (Fritzke, 1994; Alahakoon et al., 2000) dynamically expand their structure when the existing representation becomes insufficient (Figure 2; see Appendix). Standard GSOM growth is driven by accumulated quantization error: when a unit exceeds a threshold determined by the spread factor, new units are introduced around it. This allows representational capacity to be allocated adaptively rather than through a predetermined grid.

We extend this mechanism with *surprisal-driven growth*, which measures how poorly the current representation explains an incoming sample. A unit becomes a *growth node* when its accumulated quantization error or surprisal exceeds its corresponding threshold. Growth occurs locally in its 4-neighborhood, at $(x \pm 1, y)$ and $(x, y \pm 1)$, with new units inserted at unoccupied locations and initialized using the growth-triggering observation and neighboring units. This enables the map to allocate capacity to newly encountered or poorly represented regions while leaving established regions largely unchanged.

Building on the above mechanism, we develop a task-agnostic class-incremental CL framework in which GSOM neurons act as generative memory units storing per-unit distributional summaries (mean, variance, and covariance) of previously observed data. The full training procedure is formalised

in Algorithm 1 (see Appendix). At each batch t, the model processes $\mathcal{T}_t = \mathcal{B}_t \cup \mathcal{R}$, where $\mathcal{B}_t$ is the incoming batch and $\mathcal{R}$ contains replayed samples from the current GSOM units. If a global encoder is used, it is updated on $\mathcal{T}_t$ and the GSOM receives latent codes $\mathcal{Z}_t$; otherwise the GSOM operates directly on $\mathcal{T}_t$. After each batch, K synthetic samples are drawn from each active BMU's bias-corrected statistics and decoded to form $\mathcal{R}$ for the next step. The GSOM is then trained with batch t for I_{total} iterations, with growth only being allowed for the first I_{grow} iterations.

Our framework naturally extends to different data modalities; for high-dimensional visual datasets (e.g., CIFAR-10/100, TinyImageNet and MiniImageNet), our GSOM is applied over a learned latent space obtained from models such as variational autoencoders or pre-trained encoders. Importantly, the training process remains fully unsupervised: class labels are never used for representation learning or the GSOM updates and are only employed post-hoc for evaluation (via BMU-based labeling).

2.2 Tracking Neural Unit Distributional Statistics

Each GSOM neural unit maintains running estimates of the distribution of samples that are mapped to it. These statistics serve two purposes in the proposed framework: (i) enable generative replay, and (ii) drive our probabilistic surprisal criterion used to trigger our GSOM's topology growth. Given a GSOM unit at position (i, j) and a momentum factor α, an input sample x mapped to the best matching unit (BMU) updates the unit's core properties as follows: *(i)*, a running **mean vector** (or exponential moving average): $\mu_{ij} \leftarrow (1-\alpha)\mu_{ij} + \alpha x$; *(ii)*, a running **variance vector** σ^2_{ij}, which captures per-dimension variability, calculated by: $\sigma^2_{ij} \leftarrow (1-\alpha)\sigma^2_{ij} + \alpha(\mu_{ij} - x)^2$; and, *(iii)*, a running **covariance matrix** Σ_{ij}, which models inter-feature relationships, calculated via: $\Sigma_{ij} \leftarrow (1-\alpha)\Sigma_{ij} + \alpha(x - \mu_{ij})(x - \mu_{ij})^\top$. Here, x denotes either a raw input pattern (when the GSOM operates directly on data) or a latent representation produced by an encoder model at step t. These statistics characterize the local distribution of pattern vectors assigned to each GSOM unit. For a given BMU, neighboring units are updated according to the standard GSOM neighborhood adaptation rule to preserve topological smoothness. However, distributional statistics (mean, variance, and covariance) are updated only for the BMU associated with the input sample to maintain a localized and statistically consistent estimate of the underlying data distribution. We also further apply bias-correction similar to the Adam optimizer (Kingma & Ba, 2017) to these statistics to alleviate bias during initialization (fully described in Appendix B).

2.3 Gaussian Surprisal for BMU-Based GSOM.

Given our model's per-unit Gaussian statistics, each GSOM neuron b at position (i, j) in the map models the distribution of the patterns assigned to it as a multivariate Gaussian $z \sim \mathcal{N}(\mu_b, \Sigma_b)$, where $z \in \mathbb{R}^d$ is the data (or latent representation of it), $\mu_b \in \mathbb{R}^d$ is the running mean of unit b, and $\Sigma_b \in \mathbb{R}^{d\times d}$ is the corresponding covariance matrix. The resulting likelihood of observing z under the BMU distribution is:

$$p(z \mid \mu_b, \Sigma_b) = \frac{1}{(2\pi)^{d/2}|\Sigma_b|^{1/2}} \exp\left(-\frac{1}{2}(z-\mu_b)^T\Sigma_b^{-1}(z-\mu_b)\right). \tag{1}$$

Consequently, we define the surprisal of sample z under BMU b as the negative log-likelihood $S_b(z) = -\log p(z \mid \mu_b, \Sigma_b)$. Substituting the Gaussian density, taking the negative logarithm, and applying standard log identities yields the following:

$$S_b(z) = \frac{1}{2}(z-\mu_b)^T\Sigma_b^{-1}(z-\mu_b) + \frac{1}{2}\log|\Sigma_b| + \frac{d}{2}\log(2\pi). \tag{2}$$

The full derivation is provided in the Appendix A. For topology growth decisions, additive constants in the negative log-likelihood can be omitted because they do not affect inequality-based threshold comparisons. The term $\frac{d}{2}\log(2\pi)$ is constant, while the log-determinant term $\frac{1}{2}\log|\Sigma_b|$ reflects the global volume of the Gaussian region rather than the local deviation of z from the BMU center. Since the goal of our GSOM's growth is to detect when a sample lies far from the existing representation of a unit, our criterion can be based solely on the quadratic term. This leads to our GSOM's normalized Mahalanobis surprisal measure:

$$S_b(z) = \frac{1}{d}(z-\mu_b)^T\Sigma_b^{-1}(z-\mu_b), \tag{3}$$

which provides a dimension-independent measure of statistical deviation. When the surprisal of an incoming sample exceeds a threshold (defined later), the GSOM expands locally around the BMU to accommodate the new region of the data distribution. Unlike quantization error, surprisal accounts for the spread and shape of each unit's assigned data.

Growth decision rule: Building on the surprisal measure $S_b(z)$ defined before, growth decisions are governed by both distributional as well as structural criteria. Our model's surprisal threshold adapts online to the recent history $\mathcal{H} = \{S(\cdot)\}_{t-1000:t}$:

$$T_S = \begin{cases} \text{Percentile}_{92}(\mathcal{H}) & \text{if } |\mathcal{H}| > 50, \\ 1.7 & \text{otherwise,} \end{cases} \tag{4}$$

where the fallback value (the second case of the piece-wise function) provides a conservative cold-start threshold before sufficient history has been accumulated. Three constraints stabilize our GSOM's expansion: *(i)* a minimum hit count $h_b \geq h_{\min}$ that ensures reliable per-unit statistics before growth is permitted, *(ii)* a cool-down $\Delta t_b > t_{\text{cool}}$ which prevents repeated growth from the same unit in consecutive iterations, and *(iii)* a linearly decaying growth probability (given the ith iteration of the batch), $p_{\text{grow}}(i) = \max\left(0.2,\ 1 - 0.5\,\frac{i}{I_{\text{grow}}}\right)$, which reduces the GSOM's expansion pressure as training progresses through the growth phase of I_{grow} batches. The full growth condition at unit b for sample z at iteration i can then be defined as:

$$\begin{aligned} \mathcal{G}_b(z) = \mathbf{1}\big[\ & S_b(z) > T_S \wedge E_b > T_E \wedge h_b \geq h_{\min} \\ & \wedge\ \Delta t_b > t_{\text{cool}} \wedge u < p_{\text{grow}}(i) \wedge \exists (x', y') \in \mathcal{N}_4(b) : (x', y') \notin \mathcal{A}\ \big] \end{aligned} \tag{5}$$

where $u \sim \mathcal{U}(0, 1)$. Here h_b denotes the accumulated hit count of unit b, Δt_b the number of iterations since b last triggered growth, and T_E is the quantization error threshold derived from the spread factor $T_E = -d \log(\text{SF})$ (Alahakoon et al., 2000). E_b denotes the accumulated quantization error of unit b, updated whenever b is selected as the BMU, and $\mathcal{N}_4(b)$ denotes the 4-neighbourhood of unit b (i.e., adjacent grid locations) while $\mathcal{A}$ is the set of currently active units. Values for $h_{\min}$, t_{cool}, and $G_{\max}$ are provided in Table 7 in the Appendix.

When no free position exists in $\mathcal{N}_4(b)$, i.e., an interior unit demands growth, growth cannot be applied directly. In this case, the accumulated error is redistributed to neighbouring units following the standard GSOM update rule (Alahakoon et al., 2000). This allows growth pressure to propagate toward nearby regions where expansion is feasible. A detailed formulation of the error E_b update and propagation process is provided in Appendix C.

Growth expansion control: To control model expansion, GSOM growth is restricted to the early stages of training each batch, where the model is trained for a total of I iterations but growth is only permitted during the first I_{grow} iterations. During each growth batch, the number of newly added units is further limited to, at most, G_{max} per iteration. After the growth phase, the GSOM structure is fixed and only weight updates are performed for remaining iterations. This strategy prevents uncontrolled expansion while allowing the model to adapt to the data distribution stably and gradually.

2.4 Task-Free Continual Learning with GSOM and Synthetic Replay

Unlike task-based CL, our framework receives no task identifiers, class labels, or boundary signals during training. Samples arrive as a continuous stream of mini-batches, and all components are updated solely from the incoming data and GSOM-generated replay, without distinguishing between the real and synthetic samples. Thus, our learning and replay mechanism are task-free and label-free. When a new mini-batch of samples arrives, synthetic samples are generated from the GSOM by sampling from the stored distributions of its neural units. These replayed samples are then combined with the mini-batch to mitigate forgetting. Synthetic samples are generated as:

$$\tilde{x} \sim \mathcal{N}(\hat{\mu}_{ij}, \hat{\sigma}^2_{ij}) \quad \text{(MNIST)}$$

$$\tilde{z} \sim \mathcal{N}(\hat{\mu}_{ij}, \hat{\Sigma}_{ij}) \quad \text{(CIFAR-10 / CIFAR-100 / TinyImageNet/ MiniImageNet)}$$

For comparatively low-dimensional inputs such as MNIST, per-dimension mean–variance statistics provide an efficient approximation for replay directly in input space, as it can capture the data distribution reasonably well. However, this approach does not scale to higher-dimensional natural image datasets. In pixel space, mean–variance sampling produces low-quality samples and degrades

downstream performance. Moreover, modeling full covariance in the original space is computationally prohibitive; for instance, a single BMU for a CIFAR10 image would require storing a 3072×3072 covariance matrix for RGB images, which is infeasible in practice. To address these limitations, we perform replay in a lower-dimensional latent space. High-dimensional inputs are first mapped to lower-dimensional embeddings via an encoder, which allows the GSOM to operate on compact representations. This significantly reduces both storage and computational requirements while enabling full covariance modeling. For this covariance-based sampling, eigenvalue regularization is applied to ensure numerical stability and to maintain a valid positive semi-definite covariance matrix (see Algorithm 3 in Appendix E).

We study four configurations of our framework: *(i)* a GSOM operating directly on input space, *(ii)* a GSOM combined with a VAE where replay occurs in latent space and samples are decoded (i.e., ancestral sampled) into input space, *(iii)* a GSOM operating on pretrained embedding spaces such as those obtained from ResNet-18 and CLIP, where a trained decoder reconstructs images from our sampled latents (see Appendix K), and *(iv)* a modular form of replay where each GSOM unit is paired with its own local encoder-decoder (see Appendix D). Our methodology is usefully agnostic to the choice of encoder-decoder architecture and for the later configurations, the encoder produces latent codes (of dimensionality $d = 128$ or $d = 512$) making full covariance estimation for each BMU tractable.

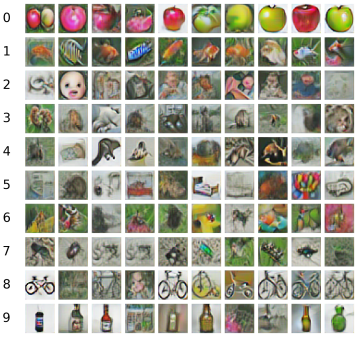


Figure 1: CIFAR-100 samples for the first 10 classes generated by sampling from a GSOM integrated with ResNet-18 embeddings. Each row corresponds to one class (0–9).

In all cases, the GSOM serves as a structured memory that stores only statistical summaries (mean, variance, and covariance) for each neural unit. This enables exemplar-free replay and the storage scales with the number of GSOM units as opposed to the size of the dataset. The GSOM's topology-preserving structure further supports interpretability; Figure 1 shows synthetic CIFAR-100 samples decoded from BMU statistics and Appendix H provides further visualisations on other datasets. By decoding unit representations or sampled latent vectors, it is also possible to inspect the evolution of learned features over time (see Appendix L). Well-separated and coherent clusters on the GSOM grid indicate stable learning, whereas fragmentation may reflect forgetting or underfitting. This makes the framework not only generative but also visually interpretable.

3 Results

Experiments were conducted under a streaming setting where data from different classes was presented sequentially as well as under a split protocol in which the dataset was divided into disjoint class groups. To account for variability due to data ordering, results are averaged over 20 independent runs with different random class sequences. Performance is reported as the mean accuracy along with the standard deviation across runs. Hyperparameters were selected based on extensive ablation studies (see Appendix E).

Label Usage and Evaluation Protocol. All GSOM and encoder-decoder models were trained in an unsupervised manner without using class labels, task labels, or task identifiers – data was organized into class-wise or split-wise sequences only for evaluation purposes against existing work. After training, each unit is assigned the majority class of its mapped training samples; test samples inherit their BMU's label. Therefore, class-wise representations, task-wise accuracy matrices, forgetting metrics, and confusion matrix analyses (see Appendices I, J and L) are post-hoc evaluation tools and not used to guide model training or replay.

3.1 Standard Split Multi-Class Incremental Learning

Split Benchmarks without pretraining: Table 1 presents results on Split-MNIST, Split CIFAR-10, and Split CIFAR-100 without the use of pretrained representations. We compare our VAE-integrated GSOM (GSOM+VAE) framework with standard CL baselines, including regularization methods (EWC, SI, LwF), replay-based methods (GEM, iCaRL, GSS), and probabilistic approaches, e.g.,

CN-DPM as well DendSOM, a SOM-based method. As expected, regularization-based methods perform poorly across all datasets, with accuracies close to fine-tuning, indicating severe catastrophic forgetting. The global VAE-GSOM achieves the best Split CIFAR-10 accuracy (50.11%), on par with GSS (49.22%). On Split-MNIST, it comes within 1.7 points of CN-DPM without labels, task boundaries, or stored exemplars. It trails iCaRL on Split CIFAR-100 (13.81% vs. 15.76%); however, iCaRL stores real exemplars and uses labels to compute class means, and both advantages grow in value as the number of classes increases. Our method also outperforms DendSOM, the closest SOM-based continual learner, by 3.3 points (50.11% vs. 46.79%), showing that surprisal-driven growth with generative replay scales better than fixed dendritic SOMs to natural images.

As to the contributions of the VAE, on Split-MNIST it adds only 0.9 points over GSOM alone (92.11% vs. 91.22%), as its per-dimension mean–variance statistics are sufficient to describe each unit's data for replay (Sec. 2.4). The VAE removes this bottleneck for CIFAR-10 showing top performance (50.11%), however it does not scale further. Accuracy falls to about 6% on TinyImageNet and MiniImageNet (Table 4). An encoder trained from scratch on a non-stationary stream with few samples per class produces entangled latents, motivating the use of pretrained feature representations.

Table 1: Classification accuracy (%) for class-incremental learning on Split-MNIST, Split CIFAR-10, and Split CIFAR-100 without pre-trained models. Our results are reported as mean ± standard deviation over 20 runs, whereas compared results are over 5 runs. **Sup.** indicates whether class labels are used during training, and **Task-free** indicates whether explicit task-boundary information is unavailable during training. Accuracy should therefore be interpreted alongside these qualifiers.

Method	Sup.	Task-free	Split-MNIST	Split-CIFAR-10	Split-CIFAR-100
iid-offline	Yes	–	95.82 ± 0.33	80.54 ± 0.63	48.09 ± 0.90
Fine-Tune	Yes	Yes	19.68 ± 0.02	19.19 ± 0.06	8.32 ± 0.23
Task-aware baselines					
EWC (Kirkpatrick et al., 2017)	Yes	No	19.92 ± 0.35	16.18 ± 1.37	4.41 ± 0.37
SI (Zenke et al., 2017)	Yes	No	19.76 ± 0.01	17.27 ± 0.87	5.87 ± 0.21
LwF (Li & Hoiem, 2018)	Yes	No	20.54 ± 0.64	18.53 ± 0.12	6.93 ± 0.32
GEM Lopez-Paz & Ranzato (2022)	Yes	No	48.57 ± 5.26	25.54 ± 0.19	6.18 ± 0.20
iCaRL (Rebuffi et al., 2017)	Yes	No	72.55 ± 0.45	35.88 ± 1.43	$\mathbf{15.76 \pm 0.15}$
DendSOM (Pinitas et al., 2021)	**No**	No	$\mathbf{95.28 \pm 0.02}$	46.79 ± 0.10	–
Supervised task-free baselines					
CN-DPM (Lee et al., 2020)	Yes	**Yes**	93.81 ± 0.07	46.98 ± 0.62	–
GSS (Aljundi et al., 2019c)	Yes	**Yes**	54.14 ± 4.68	49.22 ± 1.71	11.33 ± 0.40
Ours: unsupervised and task-free					
GSOM only	**No**	**Yes**	91.22 ± 1.23	–	–
Global VAE-GSOM	**No**	**Yes**	92.11 ± 0.98	$\mathbf{50.11 \pm 0.89}$	13.81 ± 1.05
VAE-per-BMU-GSOM	**No**	**Yes**	91.01 ± 0.12	47.12 ± 1.23	12.49 ± 2.01

Split Benchmarks with pretrained models: Table 2 compares GSOM against replay methods (GEM, iCaRL, reservoir, MIR, GSS, A-GEM, ER$_a$, ER+GMED) and expansion or prototype methods (CN-DPM , CURL, CoPE, Dynamic OCM, DCM), all using a ResNet-18 backbone. Replay methods weaken as classes accumulate because a fixed buffer holds fewer samples per class. Expansion and prototype methods, which add capacity as new data arrives, hold up better; with GSOM following the same principle but only growing when surprisal indicates the current map fails to explain the data.

Our GSOM-based methods consistently achieve superior or competitive performance across all benchmarks, where a significant aspect of our results is the learning setting in which they are achieved. All compared baselines – except CURL and DCM variants – are trained with full class supervision, receiving ground-truth labels at every update step throughout the data stream, whereas our method is unsupervised. With a ResNet-18 encoder, GSOM achieves the best accuracy on all four benchmarks: 97.21% on Split-MNIST, 58.10% on Split CIFAR-10, 35.67% on Split CIFAR-100, and 33.60% on Split MiniImageNet. On Split CIFAR-10, Welch's t-tests against the reported statistics of all baselines confirm that the improvement is significant ($p \leq 2\times10^{-4}$ in every case), and it remains so after Bonferroni correction (Appendix G).

The margin over the strongest supervised baseline grows with task difficulty: $+1.5$ points on MNIST and $+6.8$ on CIFAR-10 over Dynamic OCM, and $+4.9$ on CIFAR-100 over ER$_a$. Against DCM, a label-free baseline evaluated on all three datasets, the margin widens from $+2.1$ to $+8.1$ to $+12.7$

Table 2: Performance comparison on Split-MNIST, Split-CIFAR10, Split-CIFAR100, and Split MiniImageNet. Our results are reported as mean ± standard deviation over 20 independent runs; baseline results are as reported in the original works (5–20 runs). **Sup.** indicates whether class labels are used during training, and **Task-free** indicates whether explicit task-boundary information is unavailable during training. Global denotes a frozen shared encoder, Global FT denotes a globally fine-tuned shared encoder, and BMU Specific denotes BMU-wise encoder adaptation.

Method	Sup.	Task Free	MNIST	CIFAR10	CIFAR100	MiniImageNet
Supervised task-aware baselines						
GEM (Lopez-Paz & Ranzato, 2022)	Yes	No	93.25 ± 0.36	24.13 ± 2.46	11.12 ± 2.48	–
iCARL (Rebuffi et al., 2017)	Yes	No	83.95 ± 0.21	37.32 ± 2.66	10.80 ± 0.37	–
AGEM (Hu et al., 2020)	Yes	No	29.02 ± 5.30	18.49 ± 0.60	2.40 ± 0.20	2.09 ± 0.30
Supervised Task-free baselines						
CN-DPM (Lee et al., 2020)	Yes	Yes	93.23 ± 0.09	45.21 ± 0.18	20.10 ± 0.12	27.07 ± 2.30
Reservoir (Vitter, 1985)	Yes	Yes	92.16 ± 0.75	42.48 ± 3.04	19.57 ± 1.79	–
MIR (Aljundi et al., 2019a)	Yes	Yes	93.20 ± 0.36	42.80 ± 2.22	20.00 ± 0.57	25.21 ± 2.20
CoPE (Lange & Tuytelaars, 2021)	Yes	Yes	93.94 ± 0.20	48.92 ± 1.32	21.62 ± 0.69	–
CoPE-CE (Lange & Tuytelaars, 2021)	Yes	Yes	91.77 ± 0.87	39.73 ± 2.26	18.33 ± 1.52	–
ER + GMED (Jin et al., 2021)	Yes	Yes	82.67 ± 1.90	34.84 ± 2.20	27.27 ± 1.80	–
ER_a (Rolnick et al., 2019)	Yes	Yes	80.14 ± 3.20	46.29 ± 2.70	30.77 ± 2.20	–
Dynamic OCM (Ye & Bors, 2022a)	Yes	Yes	95.67 ± 0.22	51.27 ± 1.47	29.87 ± 0.69	28.03 ± 2.10
DCM-SE (Ye & Bors, 2024)	Yes	Yes	95.12 ± 0.32	50.02 ± 1.12	22.03 ± 0.79	–
DCM-JS (Ye & Bors, 2024)	Yes	Yes	94.76 ± 0.29	49.89 ± 1.23	22.98 ± 0.74	–
GSS (Aljundi et al., 2019c)	Yes	Yes	92.47 ± 0.92	38.45 ± 1.41	13.10 ± 0.94	–
Unsupervised Task-free baselines						
CURL Rao et al. (2019)	No	Yes	92.59 ± 0.66	–	–	–
Ours (GSOM + ResNet-18): Task-free and Unsupervised						
Global	**No**	Yes	96.12 ± 0.30	58.00 ± 1.31	$\mathbf{35.67 \pm 1.96}$	32.67 ± 1.23
Global FT	**No**	Yes	$\mathbf{97.21 \pm 0.45}$	$\mathbf{58.10 \pm 1.42}$	35.33 ± 1.28	$\mathbf{33.60 \pm 1.45}$
BMU Specific	**No**	Yes	94.34 ± 1.10	57.11 ± 1.56	34.89 ± 1.72	32.50 ± 1.06
Ours (GSOM + CLIP): Task-free and Unsupervised						
Global	**No**	Yes	91.11 ± 1.42	75.22 ± 1.18	62.31 ± 1.67	$\mathbf{52.78 \pm 1.23}$
Global FT	**No**	Yes	$\mathbf{92.23 \pm 1.36}$	78.84 ± 1.52	$\mathbf{65.01 \pm 1.41}$	52.33 ± 1.45
BMU Specific	**No**	Yes	90.01 ± 1.58	$\mathbf{81.92 \pm 1.27}$	64.22 ± 1.49	51.56 ± 1.72

points. We attribute this trend to how GSOM handles novelty. Surprisal-driven growth gives a new class new units rather than overwriting existing ones, and replay samples each unit's own Gaussian, so old classes are rehearsed in proportion to the map regions they occupy. Both mechanisms matter more as the number of classes, and hence the potential for interference, increases. Unit purity analysis supports this: on CIFAR-10, GSOM's units are substantially purer than those of a fixed-size SOM, and purity stabilizes rather than degrading as classes accumulate (Appendix F.6). Replacing ResNet-18 with CLIP raises accuracy to 81.92% on Split CIFAR-10, 65.01% on Split CIFAR-100, and 52.78% on Split MiniImageNet. On Split-TinyImageNet (Table 4), accuracy rises from 32–33% with ResNet-18 to 58–61% with CLIP.

3.2 Single Class Incremental Learning

Table 3 compares our GSOM-based methodology with several well-established CL approaches under the single-class incremental (SCI) setting, where each class is introduced sequentially (10, 100, and 200 tasks for MNIST, CIFAR-10, CIFAR-100, and TinyImageNet, respectively). While these baselines were originally designed for task-based class-incremental learning, we adapt them to the SCI protocol by treating each class arrival as an incremental update step. Importantly, these methods still rely on boundary-driven updates and are therefore not inherently task-free. In contrast, our GSOM-based approach operates in a fully task-free manner, where both representation learning and replay are driven continuously by the evolving data distribution without requiring explicit task or class boundaries. This distinction is especially important in the SCI setting, where the absence of task boundaries leads to more severe distribution shifts and increased forgetting.

As shown in Table 3, regularization methods largely collapse under SCIL (e.g., EWC reaches 1.03% on CIFAR-100). Without pretraining, VAE-GSOM leads on CIFAR-10 and CIFAR-100, exceeding DisCOIL, the strongest rehearsal-based method, by 6.6 points on CIFAR-10 without storing exemplars

Table 3: Final classification accuracy for single-class incremental learning. *All baseline methods in this table are task-aware and supervised, while the GSOM framework is task-free and unsupervised.*

Method	**w/o pretraining**			**w/ pretraining**	
	MNIST	**CIFAR10**	**CIFAR100**	**CIFAR10**	**CIFAR100**
EWC (Kirkpatrick et al., 2017)	9.91	10.01	1.03	10.21	2.93
LwF (Li & Hoiem, 2018)	19.96	10.05	2.13	19.39	6.25
IMM (Lee et al., 2018)	29.16	10.25	1.21	51.22	12.58
PGMA (Hu et al., 2019)	71.36	20.08	1.86	56.22	12.37
RPSNet (Rajasegaran et al., 2020)	40.29	16.31	1.96	55.54	4.13
OWM (Zeng et al., 2018)	94.46	19.63	3.67	**83.03**	63.26
DisCOIL (Sun et al., 2022)	**96.69**	44.54	–	–	–
Ours (GSOM-based)					
GSOM only	92.12 ± 0.84	–	–	–	–
VAE (FT global)	91.11 ± 1.26	$\mathbf{51.12 \pm 0.57}$	11.22 ± 1.43	–	–
VAE (FT BMU specific)	90.55 ± 0.91	47.72 ± 1.68	$\mathbf{11.91 \pm 0.36}$	–	–
ResNet-18 (global)	–	–	–	58.21 ± 1.12	36.11 ± 0.49
ResNet-18 (FT global)	–	–	–	59.01 ± 1.75	35.91 ± 0.28
ResNet-18 (FT BMU specific)	–	–	–	58.65 ± 0.63	35.93 ± 1.34
CLIP (global)	–	–	–	76.11 ± 0.72	$\mathbf{63.87 \pm 1.19}$
CLIP (FT global)	–	–	–	79.21 ± 1.51	63.83 ± 0.41
CLIP (BMU specific)	–	–	–	79.03 ± 0.95	63.12 ± 1.07

or using labels. On MNIST, DisCOIL and OWM remain ahead, since digit classes are easy to separate and supervised decision boundaries preserve them almost perfectly. OWM also leads on CIFAR-10 (83.03% vs. 79.21%) but is tied with our GSOM with CLIP backbone on CIFAR-100. OWM protects old classes by restricting each update to directions orthogonal to earlier inputs, which works well for a short 10-class sequence on strong frozen features, but these directions become crowded as the number of classes grows. GSOM instead adds units for new classes, so its capacity grows with the stream. Finally, on CIFAR-10 and CIFAR-100, GSOM's SCIL accuracy closely matches its split accuracy, differing by at most about 3 points and usually by less than 1. This is expected of a task-free learner: GSOM's updates use no task or batch-composition information, so the two protocols differ only in sample order. The task-aware baselines, in contrast, degrade markedly under SCIL; for example, LwF falls from 18.53% to 10.05% on CIFAR-10.

Benchmark: Single Class Incremental Learning (SCIL) TinyImageNet (200 **sequential classes) and MiniImageNet (**100 **sequential classes):** To our knowledge, prior work does not report results under this protocol, as most studies adopt multi-class task splits. SCIL presents a significantly more challenging setting due to the absence of intra-task diversity, as the sequential batch of data consists of only a single class at a time. As shown in Table 4, while VAE-based variants trained from scratch perform poorly (6%), pretrained features substantially improve results, with GSOM + CLIP achieving 42.61% on TinyImageNet and 34.57% on MiniImageNet. These performance improvements are achieved without explicit task boundaries, suggesting that GSOM is better suited for realistic task-free continual learning scenarios when such information is unavailable.

3.3 Analysis and Ablations

Table 5 tests the two mechanisms central to GSOM: when the map grows and how its units are replayed. Additional results on map size, bias correction, unit purity, and replay quality are in Appendix F.

Growth. Using surprisal instead of quantization error improves accuracy by 3.76 points on CIFAR-10 and 2.05 on CIFAR-100; using both improves it further. Surprisal accounts for the spread and shape of samples assigned to a unit, so distance from its centre alone does not determine whether a new unit is needed. Consistent with this explanation, unit purity declines less as later CIFAR-10 classes arrive (Appendix F.6).

Replay. Removing replay sharply reduces accuracy, showing that growth alone does not preserve earlier classes. Yet increasing K from 1 to 2 also reduces accuracy. At the final CIFAR-10 map size, $K = 1$ already produces about 7.5 replay samples per incoming sample; more draws from the same units may give new data too little influence. Increasing the map from 956 to 1621 units, by comparison, changes accuracy only from 58.00% to 58.18% (Appendix F). The default map size is therefore close to the observed performance plateau.

Table 4: GSOM-based method performance on Split-TinyImageNet and single-class incremental learning (SCIL) TinyImageNet and MiniImageNet.

Method	Split-TinyImageNet	TinyImageNet (SCIL)	MiniImageNet (SCIL)
VAE (Global FT)	6.22 ± 0.67	6.12 ± 0.58	6.28 ± 1.12
VAE (BMU Specific)	6.33 ± 0.88	6.01 ± 0.73	5.94 ± 1.36
GSOM + ResNet-18 (global)	32.12 ± 0.78	30.67 ± 1.23	24.21 ± 0.92
GSOM + ResNet-18 (global FT)	32.78 ± 1.56	31.23 ± 0.15	25.04 ± 0.71
GSOM + ResNet-18 (BMU Specific)	32.59 ± 1.89	31.61 ± 1.51	24.63 ± 0.84
GSOM + CLIP (global)	57.98 ± 1.11	40.45 ± 1.02	33.18 ± 0.58
GSOM + CLIP (global FT)	58.25 ± 1.02	**42.61 ± 0.91**	34.02 ± 1.47
GSOM + CLIP (BMU Specific)	**60.89** ± 0.72	40.91 ± 1.14	**34.57 ± 1.52**

Table 5: Component ablations on Split CIFAR (final accuracy, %). Growth experiments use a global VAE; replay experiments use a frozen ResNet-18. Entries are mean ± standard deviation.

Component	Setting	CIFAR-10	CIFAR-100
Growth criterion	Error only	44.87±1.03	10.92±0.88
	Surprisal only	48.63±0.95	12.97±0.96
	Error + surprisal	50.11±0.89	13.81±1.05
Replay samples/unit	$K = 0$	25.11±1.01	8.89±0.78
	$K = 1$	58.00±1.31	35.67±1.96
	$K = 2$	52.11±1.92	22.68±2.45

Representation. The encoder remains consequential even with the same growth and replay rules: replacing a frozen ResNet-18 with frozen CLIP raises Split CIFAR-100 accuracy from 35.67% to 62.31% (Table 2). A more separable feature space can make both unit assignment and replay easier.

Capacity and protocol sensitivity. Increasing the CIFAR-10 map from 956 to 1621 units improves accuracy by only 0.18 points, suggesting that the default growth budget is near the observed performance plateau (Appendix F). The harder case is learning visually similar classes one at a time. With a frozen CLIP encoder, accuracy changes little between split and single-class CIFAR streams, but falls from 57.98% to 40.45% on TinyImageNet and from 52.78% to 33.18% on MiniImageNet. Confusion among related classes suggests that their features may overlap within existing units.

4 Conclusion and Future Work

In this work, we developed a GSOM-based framework for task-free continual learning (CL) that combines surprisal-driven structural growth with generative replay. The GSOM model dynamically adapts its topology to evolving data distributions while its units maintain distributional statistics used to generate synthetic samples, eliminating the need to store raw data. By integrating encoder–decoder architectures, our framework scales from low- to high-dimensional settings. Experimental results on MNIST, CIFAR-10, CIFAR-100, TinyImageNet, and MiniImageNet demonstrate that the proposed approach mitigates catastrophic forgetting and remains competitive with existing CL schemes.

Despite the strengths shown by our proposed method, several limitations remain. The per-BMU encoder–decoder variant, while conceptually preferable for localized specialization, suffers from limited data availability per unit, leading to unstable training and reduced generative quality compared to global models. Additionally, our reliance on Gaussian-based distributional assumptions may not fully capture complex, multimodal data distributions, which can affect the fidelity of synthetic replay. In some cases, overlapping distributions across units may also reduce discriminative performance.

Future work will focus on improving both scalability and representation quality. Parallelizing GSOM updates is a natural direction for improving computational efficiency, particularly in large-scale settings. We will explore parameter-efficient per-BMU fine-tuning, where most encoder–decoder parameters are shared and only lightweight components (e.g., final layers or adapter modules) are BMU-specific. Extending the framework to a purely online learning regime, where each sample is processed in a single pass, is another key step toward real-world deployment. Additionally, more expressive generative models and non-Gaussian or mixture-based sampling strategies could improve our GSOM replay. In sum, this study demonstrates that combining topology-preserving structures with probabilistic growth and generative replay provides a promising direction for scalable, task-agnostic continual learning.

REFERENCES

Wickliffe Abraham and Anthony Robins. Memory retention - the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, 28:73–8, 03 2005. doi: 10.1016/j.tins.2004.12.003.

D. Alahakoon, S.K. Halgamuge, and B. Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, 11(3):601–614, 2000. doi: 10.1109/72.846732.

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *CoRR*, abs/1908.04742, 2019a. URL http://arxiv.org/abs/1908.04742.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019b.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning, 2019c. URL https://arxiv.org/abs/1903.08671.

Pouya Bashivan, Martin Schrimpf, Robert Ajemian, Irina Rish, Matthew Riemer, and Yuhai Tu. Continual learning with self-organizing maps, 2019. URL https://arxiv.org/abs/1904.09330.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Bernd Fritzke. A growing neural gas network learns topologies. *Advances in neural information processing systems*, 7, 1994.

Nuwan Gunasekara, Bernhard Pfahringer, Heitor Murilo Gomes, and Albert Bifet. Survey on online streaming continual learning. In *IJCAI*, pp. 6628–6637, 2023.

Guannan Hu, Wu Zhang, Hu Ding, and Wenhao Zhu. Gradient episodic memory with a soft constraint for continual learning. *CoRR*, abs/2011.07801, 2020. URL https://arxiv.org/abs/2011.07801.

Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming catastrophic forgetting via model adaptation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGvcoA5YX.

Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning, 2021. URL https://arxiv.org/abs/2006.15294.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. ISSN 1935-8245. doi: 10.1561/2200000056. URL http://dx.doi.org/10.1561/2200000056.

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. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. doi: 10.1109/5.58325. URL https://doi.org/10.1109/5.58325.

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams, 2021. URL https://arxiv.org/abs/2009.00919.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching, 2018. URL https://arxiv.org/abs/1703.08475.

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning, 2020. URL https://arxiv.org/abs/2001.00689.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2022. URL https://arxiv.org/abs/1706.08840.

Nicolas Michel, Maorong Wang, Jiangpeng He, and Toshihiko Yamasaki. From offline to online memory-free and task-free continual learning via fine-grained hypergradients, 2025. URL https://arxiv.org/abs/2502.18762.

Alexander G Ororbia. Continual competitive memory: A neural system for online task-free lifelong learning. *arXiv preprint arXiv:2106.13300*, 2021.

Kosmas Pinitas, Spyridon Chavlis, and Panayiota Poirazi. Dendritic self-organizing maps for continual learning, 2021. URL https://arxiv.org/abs/2110.13611.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Khan, Ling Shao, and Ming-Hsuan Yang. An adaptive random path selection approach for incremental learning. 01 2020.

Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. *Advances in neural information processing systems*, 32, 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning, 2017. URL https://arxiv.org/abs/1611.07725.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning, 2019. URL https://arxiv.org/abs/1811.11682.

Wenju Sun, Qingyong Li, Jing Zhang, Danyu Wang, Wen Wang, and Yangli-ao Geng. Exemplar-free class incremental learning via discriminative and comparable one-class classifiers. *CoRR*, abs/2201.01488, 2022. URL https://arxiv.org/abs/2201.01488.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pp. 181–209. Springer, 1998.

Hitesh Vaidya, Travis Desell, and Alexander Ororbia. Reducing catastrophic forgetting in self organizing maps with internally-induced generative replay. *CoRR*, abs/2112.04728, 2021. URL https://arxiv.org/abs/2112.04728.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00568-3. URL https://doi.org/10.1038/s42256-022-00568-3.

Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985. ISSN 0098-3500. doi: 10.1145/3147.3165. URL https://doi.org/10.1145/3147.3165.

Barry Payne Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

Buddhi Wickramasinghe, Gobinda Saha, and Kaushik Roy. Continual learning: A review of techniques, challenges, and future directions. *IEEE Transactions on Artificial Intelligence*, 5(6): 2526–2546, 2024. doi: 10.1109/TAI.2023.3339091.

Yutao Yang, Jie Zhou, Xuanwen Ding, Tianyu Huai, Shunyu Liu, Qin Chen, Yuan Xie, and Liang He. Recent advances of foundation language models-based continual learning: A survey. *ACM Comput. Surv.*, 57(5), January 2025. ISSN 0360-0300. doi: 10.1145/3705725. URL https://doi.org/10.1145/3705725.

Fei Ye and Adrian G. Bors. Lifelong infinite mixture model based on knowledge-driven dirichlet process. *CoRR*, abs/2108.12278, 2021. URL https://arxiv.org/abs/2108.12278.

Fei Ye and Adrian G. Bors. Continual variational autoencoder learning via online cooperative memorization, 2022a. URL https://arxiv.org/abs/2207.10131.

Fei Ye and Adrian G Bors. Task-free continual learning via online discrepancy distance learning. *Advances in Neural Information Processing Systems*, 35:23675–23688, 2022b.

Fei Ye and Adrian G. Bors. Online task-free continual generative and discriminative learning via dynamic cluster memory. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26202–26212, 2024. doi: 10.1109/CVPR52733.2024.02476.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continuous learning of context-dependent processing in neural networks. *CoRR*, abs/1810.01256, 2018. URL http://arxiv.org/abs/1810.01256.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence, 2017. URL https://arxiv.org/abs/1703.04200.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey, 2024a. URL https://arxiv.org/abs/2401.16386.

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9851–9873, 2024b. doi: 10.1109/TPAMI.2024.3429383.

Appendix

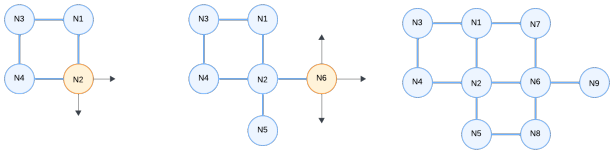


Figure 2: The proposed GSOM growth mechanism. When a unit (highlighted in yellow) exceeds the growth threshold, new units are added in its immediate 4-neighborhood, i.e., at $(x, y \pm 1)$ and $(x \pm 1, y)$ if those positions are not already occupied.

A Negative Log-Likelihood of a Multivariate Gaussian

The surprisal (negative log-likelihood) of a sample z under a Gaussian distribution $\mathcal{N}(\mu_b, \Sigma_b)$ is obtained by taking the negative logarithm of the probability density function and applying standard logarithmic identities.

$$I_b(z) = -\log\left[\frac{1}{(2\pi)^{d/2}|\Sigma_b|^{1/2}}\exp\left(-\frac{1}{2}(z-\mu_b)^T\Sigma_b^{-1}(z-\mu_b)\right)\right] \tag{6}$$

$$= -\log\left(\frac{1}{(2\pi)^{d/2}|\Sigma_b|^{1/2}}\right) - \log\left(\exp\left(-\frac{1}{2}(z-\mu_b)^T\Sigma_b^{-1}(z-\mu_b)\right)\right) \tag{7}$$

$$= \log\left((2\pi)^{d/2}|\Sigma_b|^{1/2}\right) + \frac{1}{2}(z-\mu_b)^T\Sigma_b^{-1}(z-\mu_b) \tag{8}$$

$$= \frac{1}{2}(z-\mu_b)^T\Sigma_b^{-1}(z-\mu_b) + \frac{1}{2}\log|\Sigma_b| + \frac{d}{2}\log(2\pi). \tag{9}$$

Algorithm 1 Unified Algorithm for batch-wise Continual learning with the GSOM and optional encoder/decoder models.

Input: Stream of batches $\{\mathcal{B}_t\}_{t=1}^{T}$; flags `GLOBAL`; replay samples per BMU K
Output: GSOM, (optional) global encoder/decoder

Initialize GSOM; initialize replay set $\mathcal{R} \leftarrow \emptyset$
if `GLOBAL` **then** initialize global encoder/decoder
for $t = 1$ to T **do**
  $\mathcal{T}_t \leftarrow \mathcal{B}_t \cup \mathcal{R}$
  // Encode features for GSOM update
  if `GLOBAL` **then**
    Train or update global encoder/decoder on $\mathcal{T}_t$
    $\mathcal{Z}_t \leftarrow \mathrm{Enc}_{\mathrm{global}}(\mathcal{T}_t)$
    Update GSOM with $\mathcal{Z}_t$ (including neighborhood and growth updates)
  else
    Update GSOM with $\mathcal{T}_t$ (including neighborhood and growth updates)
  end if
  $\mathcal{R}_t \leftarrow \emptyset$ **// Build replay for subsequent batches**
  for each active BMU (i, j) **do**
    Obtain BMU statistics (μ_{ij}, Σ_{ij})
    $(\hat{\mu}_{ij}, \hat{\sigma}^2_{ij}, \hat{\Sigma}_{ij}) \leftarrow$
    BiasCorrect$(\mu_{ij}, \sigma^2_{ij}, \Sigma_{ij}, \beta, t_{ij}, \lambda)$
    for $k = 1$ to K **do**
      $\tilde{z} \sim \mathcal{N}(\hat{\mu}_{ij}, \hat{\Sigma}_{ij})$
      $\tilde{x} \leftarrow \begin{cases} \mathrm{Dec}_g(\tilde{z}), & \texttt{GLOBAL} \\ \tilde{z}, & \text{GSOM-only} \end{cases}$
      Append $\tilde{x}$ to $\mathcal{R}_t$
    end for
  end for
  Set $\mathcal{R} \leftarrow \mathcal{R}_t$
end for
return GSOM, (global encoder/decoder if used)

B Running Distributional Statistic Bias Correction

A practical issue with exponential moving averages, as computed by the GSOM units, is their bias towards the initial value encountered during early training. For instance, when updating the running mean via $\mu_{ij,t} = \alpha x_t + (1-\alpha)\mu_{ij,t-1}$, with initial condition $\mu_{ij,0} = 0$, the estimate $\mu_{ij,t}$ tends to underestimate the true mean because it implicitly incorporates a zero-initialization. This issue is particularly pronounced in continuous learning settings, where some units may initially receive only a small number of samples. To mitigate this effect, we apply a bias correction similar to that used in the well-established Adam optimizer (Kingma & Ba, 2017). The corrected estimates for each

unit are then as follows: $\hat{\mu}_{ij,t} = \frac{\mu_{ij,t}}{1-\beta_\mu^t}, \quad \hat{\sigma}^2_{ij,t} = \frac{\sigma^2_{ij,t}}{1-\beta_\sigma^t}, \quad \hat{\Sigma}_{ij,t} = \frac{\Sigma_{ij,t}}{1-\beta_\Sigma^t}$ where t denotes the number of BMU updates received by the unit to date and $\beta_\mu, \beta_\sigma, \beta_\Sigma \in [0, 1)$ are exponential decay rates for the corresponding statistics. This correction ensures that the estimated parameters remain unbiased during early training, providing stable distribution estimates and well-conditioned covariances for both generative replay and surprisal evaluation. Unlike unbiased online estimators, such as Welford's algorithm Welford (1962), exponential updates are used because the distribution of samples mapped to a GSOM unit may evolve as the topology grows and change as a result of the data stream.

C GSOM Error Accumulation and Redistribution

For each unit b, the accumulated quantization error is updated whenever the unit is selected as the Best Matching Unit (BMU):

$$E_b \leftarrow E_b + \|z - w_b\|_2, \tag{10}$$

where z is the input sample and w_b is the weight vector of unit b. This formulation follows the standard GSOM definition, where the error accumulates as the Euclidean distance between the input and the BMU weight.

When growth is triggered at a boundary unit, the accumulated error at that unit is reset:

$$E_b \leftarrow 0. \tag{11}$$

When an interior unit exceeds the growth threshold but no free neighbouring position exists in its four-neighbourhood $\mathcal{N}_4(b)$, direct expansion is not possible. In this case, the accumulated error is redistributed to its neighbours following the standard GSOM rule (Alahakoon et al., 2000):

$$E_b^{\text{old}} = E_b, \tag{12}$$

$$E_b \leftarrow \frac{1}{2}E_b^{\text{old}}, \quad E_n \leftarrow E_n + \frac{1}{8}E_b^{\text{old}} \quad \forall\, n \in \mathcal{N}_4(b), \tag{13}$$

where $\mathcal{N}_4(b)$ denotes the set of the four adjacent grid neighbours of unit b.

This redistribution conserves the total accumulated error: the BMU retains half of its error, while the remaining half is distributed equally among its four neighbours. This mechanism allows growth pressure to propagate from interior regions toward nearby boundary units where expansion is feasible.

D Encoder-Decoder per BMU setup

While a global encoder–decoder uses single model memory only and benefits from training on the full dataset, it must generalize across diverse and often underrepresented regions of the latent space. In contrast, the per-BMU approach trains localized models using samples assigned to each unit, enabling region-specific specialization at the cost of additional memory (Algorithm 2). After mapping inputs to their corresponding BMUs, the associated samples are used to train local models, whose decoders focus on reconstructing data from specific regions of the latent space. During replay, latent samples drawn from GSOM statistics are decoded using the corresponding local model rather than a shared decoder. The motivation for this design is to align the generative capacity with the SOM topology, allowing each decoder to specialize in a local region of the latent space and improving the quality of reconstruction. However, this approach introduces additional memory overhead due to maintaining a separate model per active unit, and may become less scalable as the number of units increases. The results presented in the main paper for this variant are intended as an initial exploration of its potential with further discussion on scalability and potential improvements provided in the Discussion and Future Work section.

E Experimental Settings

All experiments were conducted on a desktop workstation equipped with an Intel Core i7-14700F CPU, 32 GB RAM, and an NVIDIA GeForce RTX 4060 Ti GPU with 16GB VRAM, running Windows 11.

Algorithm 2 Unified Algorithm for Batch-Wise Continual Learning with GSOM with Encoder/Decoder Models per GSOM unit

Input: Stream of batches $\{\mathcal{B}_t\}_{t=1}^{T}$; replay samples per BMU K
Output: GSOM, global encoder/decoder, per-BMU encoder/decoder models $\mathcal{V}$

Initialize GSOM; initialize replay set $\mathcal{R} \leftarrow \emptyset$
Initialize global encoder/decoder $(\mathrm{Enc}_g, \mathrm{Dec}_g)$
Initialize per-BMU model dictionary $\mathcal{V} \leftarrow \emptyset$
for $t = 1$ to T **do**
  $\mathcal{T}_t \leftarrow \mathcal{B}_t \cup \mathcal{R}$
  Train or update global encoder/decoder on $\mathcal{T}_t$
  $\mathcal{Z}_t \leftarrow \mathrm{Enc}_{\mathrm{global}}(\mathcal{T}_t)$
  Update GSOM with $\mathcal{Z}_t$ (including neighborhood and growth updates)
  // Train per-BMU local models on assigned subsets
  Assign each $x \in \mathcal{T}_t$ to BMU (i, j) using its current representation
    ($\mathcal{Z}_t$ if global encoder is used, else raw $\mathcal{T}_t$)
  for each BMU (i, j) with assigned set $\mathcal{S}_{ij} \subseteq \mathcal{T}_t$ **do**
    Train or update local model $\mathcal{M}_{ij}$ on $\mathcal{S}_{ij}$
    Store/refresh $\mathcal{V}[(i, j)] \leftarrow \mathcal{M}_{ij}$
    Re-encode $\mathcal{S}_{ij}$ with the encoder of $\mathcal{M}_{ij}$ to refine local GSOM updates
  end for
  // Build replay for subsequent batches
  $\mathcal{R}_t \leftarrow \emptyset$
  for each BMU (i, j) such that $(i, j) \in \mathcal{V}$ **do**
    Obtain BMU statistics $(\mu_{ij}, \sigma^2_{ij}, \Sigma_{ij})$
    $(\hat{\mu}_{ij}, \hat{\sigma}^2_{ij}, \hat{\Sigma}_{ij}) \leftarrow$
      BIASCORRECT$(\mu_{ij}, \sigma^2_{ij}, \Sigma_{ij}, \beta, t_{ij}, \lambda)$
    for $k = 1$ to K **do**
      $\tilde{z} \sim \mathcal{N}(\hat{\mu}_{ij}, \hat{\Sigma}_{ij})$
      $\tilde{x} \leftarrow \mathrm{Dec}_{ij}(\tilde{z})$
      Append $\tilde{x}$ to $\mathcal{R}_t$
    end for
  end for
  Set $\mathcal{R} \leftarrow \mathcal{R}_t$
end for
return GSOM, global $(\mathrm{Enc}_g, \mathrm{Dec}_g)$, per-BMU models $\mathcal{V}$

Table 7 summarizes the hyperparameter configurations used across all GSOM-based experiments. Unless otherwise specified, GSOM units were initialized using random samples drawn from the input data or latent representations of the first batch of data observed during training, consistent with the class-incremental learning setup. Alternative initialization strategies, including uniform sampling over $[0, 1]$ and initialization using the global mean of the data, were also evaluated but resulted in inferior performance. For each GSOM unit, the associated statistics were initialized as follows: the mean vector was initialized to zero, the variance vector to ones, and the covariance matrix to zero. We additionally explored diagonal covariance initialization and initialization using latent statistics from a pretrained VAE; however, these alternatives did not provide consistent improvements. During training, exponential moving averages were maintained for the mean, variance, and covariance of each unit. The Best Matching Units (BMUs) were identified using Euclidean distance between the input (or latent) vector and GSOM unit weights. While other distance metrics, including cosine similarity, Manhattan (L1) distance, and Mahalanobis distance, were explored, Euclidean distance as a measure for the BMUs consistently provided the most stable and accurate results across datasets. The following hyperparameter settings were explored for the bias correction:

Table 6: Bias-correction hyperparameters with best-performing values and additional settings tested.

Parameter	Symbol	Best value	Other values tested
Mean	β_μ	0.99	0.98, 0.95
Variance / Cov.	$\beta_\sigma, \beta_\Sigma$	0.95	0.91, 0.99, 0.90

Unlike fixed SOMs, GSOM dynamically expands its topology during training. To control model growth, expansion was constrained using two hyperparameters: the number of growth iterations (I_{grow}) and the maximum number of units added per iteration (G_{max}). Growth was only permitted during the initial I_{grow} iterations of each batch, after which the map structure was fixed for the remaining training iterations. This design prevents uncontrolled expansion while allowing the model to adapt to new data.

We observed that increasing growth flexibility (i.e., larger I_{grow} and G_{max}) improves representational capacity and accuracy but leads to higher computational and memory costs. Conversely, more restrictive growth produces compact maps with faster training at the expense of reduced performance. For all reported experiments, we adopt conservative growth settings (e.g., $I_{\text{grow}} = 9$, $G_{\text{max}} = 4$) to balance accuracy and efficiency. Growth decisions were driven by a Mahalanobis-based surprisal criterion computed using the running statistics of each unit. The growth threshold was determined adaptively to ensure stable, data-driven expansion. The threshold was set to the 92nd percentile of

the most recent values after sufficient observations had been collected, allowing growth sensitivity to adjust to the evolving data distribution.

Algorithm 3 Synthetic Sample Generation using Mean and Covariance (bias-corrected, Appendix B)

Require: Trained SOM, BMU coordinates (i, j), number of samples n, regularization constant ϵ
$\mu \leftarrow$ GSOM.running_mean$[i][j]$
$\Sigma \leftarrow$ GSOM.running_cov$[i][j]$
$\Sigma \leftarrow \Sigma + \epsilon I$ // Add diagonal regularization
Compute eigen-decomposition: $Q\Lambda Q^\top = \Sigma$
$\Lambda \leftarrow \max(\Lambda, \epsilon)$ // Clamp eigenvalues
$\Sigma_{\text{pd}} \leftarrow Q \cdot \text{diag}(\Lambda) \cdot Q^\top$
Sample $\tilde{z} \sim \mathcal{N}(\mu, \Sigma_{\text{pd}})$
return n samples $\{\tilde{z}_1, ..., \tilde{z}_n\}$

For generative configurations, synthetic samples were generated using the estimated mean and covariance of each GSOM unit, following the procedure described in Algorithm 3. VAE hyperparameters, including latent dimensionality, learning rate, and batch size, were selected based on reconstruction quality and training stability. In pretrained encoder-based setups (e.g., CLIP ViT-B/32 and ResNet-18), embeddings were kept frozen by default to preserve learned representations, with optional fine-tuning of the final layers. These configurations provided stable performance across datasets while maintaining reasonable computational cost.

F Ablation Studies

F.1 GSOM versus Fixed-Capacity SOM

The results in Table 8 compare fixed SOM and GSOM under bias-corrected settings across datasets and configurations. While the fixed SOM generally achieves higher accuracy—particularly in the CLIP-based setting—this is expected due to differences in the learning paradigm. The fixed SOM operates in a task-aware manner, with replay performed at the end of each class, thereby effectively leveraging known task boundaries. In contrast, GSOM is fully task-free and performs replay continuously at the batch level without access to class information, making the learning problem inherently more challenging due to the absence of explicit distribution shift signals.

In addition to accuracy, the table highlights differences in representational capacity. The fixed SOM uses a constant 40×40 topology (1600 units) across all datasets, whereas GSOM dynamically adapts its size based on data complexity. As observed, GSOM produces significantly more compact maps for simpler datasets (e.g., 357–376 units for MNIST), while scaling to larger sizes for more complex datasets (e.g., 956–1392 units for VAE-based settings and up to 1508 units for CLIP-based settings).

Overall, these results highlight a trade-off between performance and flexibility. While fixed SOM benefits from task-aware replay and a fixed high-capacity topology, GSOM offers a fully task-agnostic alternative with adaptive capacity, achieving competitive performance while efficiently allocating representational resources based on the data.

F.2 Growth Sensitivity

The growth in GSOM is controlled by two key hyperparameters: the number of growth iterations (I_{grow}) and the maximum number of units added per iteration (G_{max}). These parameters directly influence the trade-off between representational capacity and performance. Increasing I_{grow} and G_{max} allows the map to expand more aggressively, leading to larger topologies and improved accuracy due to enhanced representation of incoming data. However, excessive growth leads to disproportionately large maps with diminishing returns in performance, increasing memory and computational cost without significant accuracy gains. Conversely, restricting growth results in more compact maps with reduced computational cost but lower accuracy. We use $I_{\text{grow}} = 9$ and $G_{\text{max}} = 4$ as the baseline configuration for all primary comparisons. As shown in Table 9, GSOM demonstrates consistent behavior across datasets, where increased growth flexibility improves performance at the expense of model size. The selected baseline configuration balances this trade-off, providing stable accuracy with controlled expansion.

Table 7: Unified hyperparameter configuration across all experiments. Shared GSOM parameters are consistent unless otherwise specified.

Component	Parameter	Value
Datasets		
	MNIST	28×28 (grayscale)
	CIFAR-10 / CIFAR-100	$32 \times 32 \times 3$
	TinyImageNet / MiniImageNet	$64 \times 64 \times 3$ / $84 \times 84 \times 3$
Shared GSOM Parameters		
	Initial Map Size	2×2 (MNIST), 5×5 (CIFAR10) 7×7 (CIFAR100/Mini/TinyImagenet)
	Max Nodes	2000
	SOM Iterations per Batch/Class	30 (MNIST), 100 (Others)
	Learning Rate	0.4–0.5
	Sigma (Neighborhood Width)	0.9–0.96
	Neighborhood Function	Gaussian
	Distance Metric	Euclidean
	Batch Size	32 (MNIST), 64–128 (others)
Growth Parameters		
	Spread Factor	0.7–0.8
	Growth Iterations (I_{grow})	5 (MNIST), 9 (ResNet/VAE), 10 (CLIP)
	Max Growth per Iterations ($G_{\max}$)	2 (MNIST), 4 (ResNet/VAE), 14 (CLIP)
	Min BMU Hits	5–7
	Growth Cooldown	40 iterations
	Mahalanobis Threshold	92–93 percentile (adaptive)
	Growth Probability	Linearly decayed ($1.0 \rightarrow 0.5$)
	Error Decay	0.85–0.88
Replay and Statistics		
	Sampling Method	Gaussian (per-BMU mean/covariance)
	Covariance Regularization	$\lambda = 1 \times 10^{-4}$
	Replay Samples per BMU	$K = 1$
VAE Configuration		
	Latent Dimension	32, 64, 128
	Iterations	200
	Batch Size	128, 256
	Learning Rate	$1e{-}4$ to $1e{-}5$
	Optimizer	Adam
	Architecture	ResNet-style VAE
	Loss	Reconstruction + KL
ResNet-18 Configuration		
	Encoder	ResNet-18 (ImageNet pretrained)
	Embedding Dimension	512
	Input Resolution	224×224
	Fine-tuning	Frozen / last block FT
	Decoder	Conv / transposed conv
	Training	End-to-end with replay
CLIP Configuration		
	Encoder	CLIP ViT-B/32 (OpenAI)
	Embedding Dimension	512
	Input Resolution	224×224 (upsampled)
	Fine-tuning	Frozen / last block FT
	Decoder	Transposed conv (attention-based)
	Feature Handling	Pooled embedding

F.3 Growth Criterion

Table 10 compares three variants of the growth trigger: accumulated quantization error alone ($E_b > T_E$, equivalent to standard GSOM (Alahakoon et al., 2000)), Mahalanobis surprisal alone ($S_b(z) > T_S$), and the proposed combined criterion. The results indicate that the original GSOM growth rule based solely on accumulated quantization error yields the lowest performance across all datasets. Replacing the error criterion with the proposed surprisal measure improves performance, suggesting that distribution-aware growth better captures emerging data structure. The combined criterion achieves the strongest overall results, indicating that the quantization error term acts as a stabilizing gate while the surprisal term enables more adaptive topology expansion.

Table 8: Comparison of fixed SOM and GSOM (with bias) across datasets. Results are reported as accuracy (%), with the number of best-matching units (BMUs) after final training shown in parentheses. The fixed SOM uses a 40×40 grid (1600 units) for all datasets.

Method	Model	Split-MNIST	Split-CIFAR-10	Split-CIFAR-100	Split-TinyImageNet
SOM-only					
Fixed	SOM-only	92.11 ± 1.18 (1600)	–	–	–
GSOM	GSOM-only	91.22 ± 1.23 (357)	–	–	–
VAE (Global FT)					
Fixed	SOM + VAE	91.23 ± 1.21 (1600)	53.01 ± 1.18 (1600)	14.63 ± 0.33 (1600)	12.74 ± 0.84 (1600)
GSOM	GSOM + VAE	92.11 ± 0.98 (376)	50.11 ± 0.89 (956)	13.81 ± 1.05 (1167)	6.22 ± 0.67 (1241)
CLIP (Global FT)					
Fixed	SOM + CLIP	–	81.24 (1600)	67.19 (1600)	60.13 (1600)
GSOM	GSOM + CLIP	92.23 ± 1.36 (428)	78.84 ± 1.52 (1034)	65.01 ± 1.41 (1262)	58.25 ± 1.02 (1408)

Table 9: Growth sensitivity analysis of GSOM with global ResNet-18. Increasing growth parameters improves accuracy but increases map size. Results are reported as accuracy (%), with the number of best-matching units (BMUs) after final training shown in parentheses.

I_{grow}	$G_{\max}$	Split-MNIST	Split-CIFAR-10	Split-CIFAR-100	Split-TinyImageNet
6	3	95.01 (432)	57.41 (921)	35.01 (1099)	31.74 (1251)
7	3	95.26 (468)	57.72 (934)	35.32 (1120)	32.05 (1278)
9	4	96.12 (498)	58.00 (956)	35.67 (1161)	32.12 (1312)
10	5	96.42 (548)	58.11 (1184)	35.74 (1426)	32.28 (1502)
20	14	96.45 (712)	58.18 (1621)	35.81 (1895)	32.36 (2143)

F.4 Studying Bias versus No Bias Correction

We evaluate the impact of bias correction on the running statistics used within GSOM. Without bias correction, exponential moving averages are biased toward their initial values, particularly during early training when units receive limited updates, which leads to underestimation of the true mean and poorly conditioned variance and covariance estimates, which negatively affect both generative replay and surprisal-based growth decisions.

As observed in our experiments (Table 11), the bias-corrected variant consistently outperforms the non-corrected version across all datasets. Although the improvements are modest, they are consistent, indicating that bias correction provides more reliable statistical estimates and improves overall training stability.

F.5 Replay Volume and Samples per Unit

We examine the effect of K, the number of synthetic samples generated per active unit at each training step. At approximately 956 units on Split CIFAR-10 with a batch size of 128 and $K = 1$ already yields a replay-to-real ratio of $7.5 : 1$. Replay volume therefore depends on both K and map size, and increases as the map expands. Setting $K = 0$ removes synthetic replay while retaining the growth mechanism, per-unit statistics, and BMU assignment.

Table 12 shows that replay contributes substantially to performance: removing it reduces final accuracy by 32.89 percentage points on Split CIFAR-10 and 26.78 points on Split CIFAR-100. However, increasing replay beyond $K = 1$ consistently reduces mean accuracy on both datasets, although every tested replay configuration outperforms $K = 0$. These results favor $K = 1$ among the evaluated settings and show that additional samples per unit do not necessarily improve retention and adaptation.

The growth ablation provides a complementary comparison (Table 9). Increasing the CIFAR-10 map from 921 to 1621 units raises mean accuracy from 57.41% to 58.18%, despite increasing replay volume as there are more units. In contrast, increasing K from 1 to 2 reduces accuracy by 5.89 points. This contrast suggests that replay volume alone does not explain performance: additional units can

Table 10: Ablation study of GSOM growth criteria across continual learning benchmarks on Split datasets.

Method	MNIST	CIFAR-10	CIFAR-100
Global VAE-GSOM (Error only)	89.42 ± 1.21	44.87 ± 1.03	10.92 ± 0.88
Global VAE-GSOM (Surprisal only)	91.26 ± 1.04	48.63 ± 0.95	12.97 ± 0.96
Global VAE-GSOM (Error + Surprisal)	$\mathbf{92.11 \pm 0.98}$	$\mathbf{50.11 \pm 0.89}$	$\mathbf{13.81 \pm 1.05}$

Table 11: Effect of bias correction across GSOM + ResNet-18 configurations. Results are reported as accuracy (%).

Setting	Model	Split-MNIST	Split-CIFAR10	Split-CIFAR100	Split-MiniImageNet
No Bias					
	Global	95.84 ± 0.41	57.21 ± 1.44	34.72 ± 2.03	31.58 ± 1.37
	Global FT	96.58 ± 0.52	57.34 ± 1.51	34.91 ± 1.39	32.71 ± 1.53
	BMU Specific	93.61 ± 1.28	56.32 ± 1.67	34.02 ± 1.88	31.74 ± 1.21
Bias					
	Global	96.12 ± 0.30	58.00 ± 1.31	$\mathbf{35.67 \pm 1.96}$	32.67 ± 1.23
	Global FT	$\mathbf{97.21 \pm 0.45}$	$\mathbf{58.10 \pm 1.42}$	35.33 ± 1.28	$\mathbf{33.60 \pm 1.45}$
	BMU Specific	94.34 ± 1.10	57.11 ± 1.56	34.89 ± 1.72	32.50 ± 1.06

improve coverage of the feature space, whereas a larger K samples the same set of unit distributions more densely.

Since $K = 0$ provides no replay and $K = 2$ already degrades accuracy, the best replay volume may lie between integer values of K. A future direction is to let each unit generate a sample with probability q (e.g., $q = 0.75$), which gives finer control over the size of the replay data.

F.6 Purity of the units in the GSOM

Classification accuracy (final as well as per-class task accuracy) measures the end result of BMU labeling, but it does not show how cleanly the map separates classes internally. We therefore measure *unit purity*, the fraction of a unit's assigned training samples that belong to its most common class. Let $n_{b,c}$ be the number of training samples of class c whose BMU is unit b, and let $n_b = \sum_c n_{b,c}$. The purity of unit b is

$$\mathrm{pur}(b) = \frac{\max_c n_{b,c}}{n_b}. \tag{14}$$

where $\max_c n_{b,c}$ is the count of the unit's majority class, which is also the label the unit receives under post-hoc BMU labeling. Purity therefore equals the fraction of the unit's own training samples that its label classifies correctly. For example, a unit receiving 40 cat, 8 dog, and 2 deer images has purity $40/50 = 0.8$.

We report the mean purity over all active units (units with $n_b > 0$) and the fraction of units with purity above 0.9. As with accuracy, labels are used only for this post-hoc analysis and never during training. All measurements use the final map and the full CIFAR-10 training set ($n = 50{,}000$) under the class-incremental sequence. We compare GSOM against a fixed-size 40×40 SOM (the same baseline as in Table 8) with both a VAE and a CLIP encoder.

Table 12: Effect of replay samples per unit, K, on final accuracy (%, mean $\pm$ standard deviation) for GSOM + ResNet-18 with a frozen encoder. We report 20 runs for the primary configuration ($K = 1$) and 3 runs for each other setting.

K	Split CIFAR-10	Split CIFAR-100
0	25.11 ± 1.01	8.89 ± 0.78
1	$\mathbf{58.00 \pm 1.31}$	$\mathbf{35.67 \pm 1.96}$
2	52.11 ± 1.92	22.68 ± 2.45
4	45.23 ± 2.18	15.11 ± 2.01

Table 13: Final-state unit purity on CIFAR-10 for fixed-size and growing maps. Active BMUs are units that are the BMU of at least one training sample. The fixed 40x40 SOM has 1600 units in total with 1225 active units.

Metric	Fixed SOM + VAE	GSOM + VAE	Fixed SOM + CLIP	GSOM + CLIP
Active BMUs	1225	**956**	1325	**1034**
Mean class purity	0.580	**0.821**	0.780	**0.900**
BMUs $> 90\%$ pure (%)	8.7	**15.1**	12.7	**25.1**

GSOM uses fewer active units than the fixed SOM (956 vs. 1225 with a VAE; 1034 vs. 1225 with CLIP). Its mean purity is nonetheless higher (0.821 vs. 0.580; 0.900 vs. 0.780), and it has roughly twice the share of near-pure units (15.1% vs. 8.7%; 25.1% vs. 12.7%). The fixed SOM, meanwhile, leaves 17% of its 1600 units unused. Growth places units where the data require them, whereas a fixed grid spreads capacity uniformly and forces units at class borders to mix classes. CLIP as a backbone yields purer units than the VAE for both map types ($+0.20$ fixed SOM, $+0.08$ GSOM) which indicates that a better-separated embedding space lets each unit's Gaussian describe a single class, which is consistent with the CLIP backbone's lower forgetting (Appendix J).

Purity stabilizes as classes accumulate. We further study purity along the class sequence over the course of training (Table 14). The per-step losses shrink steadily. With a VAE, purity falls by 0.068 when the second class arrives, then by 0.048, 0.027, and 0.014, and by only 0.022 in total over the final five classes. Our GSOM with the CLIP backbone follows the same shape at a higher level, falling by 0.032 at the second class and by 0.020 over the final five. If new classes were absorbed into existing units, purity would keep falling at a roughly constant rate. Instead, it stalls, which indicates that later classes are allocated to new units rather than overwriting old ones, as the surprisal-driven growth criterion (Sec. 2.3) intends.

Table 14: Mean GSOM unit purity after each class of the CIFAR-10 class-incremental sequence. Purity is computed over the training samples of all classes seen so far. The value after the first class is trivially 1 since all units are trained only on single class and we validate on that class only.

Class step	0	1	2	3	4	5	6	7	8	9
GSOM + VAE	1.000	0.932	0.884	0.857	0.843	0.834	0.829	0.825	0.823	0.821
GSOM + CLIP	1.000	0.968	0.946	0.931	0.920	0.913	0.908	0.904	0.902	0.900

F.7 Growth hyperparameter sensitivity

GSOM's growth mechanism has four hyperparameters: the cold-start surprisal threshold τ_0, the history window W, the growth percentile p, and the covariance regularizer ε. We vary each one over a small grid and hold the others at their defaults (τ_0=1.7, W=1000, p=92, ε=10^{-4}), using Global VAE-GSOM on Split-CIFAR-10 (Table 15). Across every value tested, accuracy stays within 2.3 points of the default, and most values are within one standard deviation of it. Because non-default settings use only 3 seeds, we do not read differences smaller than about one standard deviation as meaningful. We therefore treat the table as evidence of robustness, not as a precise ranking of values.

G Statistical Significance of Accuracy Comparisons

To test whether GSOM's accuracy gains exceed run-to-run variability, we compare our 20-run GSOM + ResNet-18 (Global FT) result on Split CIFAR-10 (58.10 ± 1.42) against every baseline in Table 2 that reports a Split CIFAR-10 score. We also include two additional task-free or SOM-based baselines with comparable reported accuracy: DCM (Ye & Bors, 2024) and DendSOM (Pinitas et al., 2021) in its class-incremental setting.

Raw per-run results are unavailable for baselines quoted from prior work; only the mean, standard deviation, and number of runs are reported. We therefore use Welch's unpaired t-test, which does not

Table 15: Hyperparameter sensitivity of Global VAE-GSOM on Split-CIFAR-10 (final accuracy, %). One hyperparameter is varied at a time, with the others at their defaults (bold). Defaults are our main results (mean ± std over 20 runs); all other values are mean ± std over 3 seeds.

τ_0	1.0	1.2	1.4	**1.7**	2.0
	48.72 ± 0.74	49.31 ± 0.62	49.85 ± 0.71	$\mathbf{50.11 \pm 0.89}$	49.64 ± 0.68
W	100	500	**1000**	2000	
	48.91 ± 0.83	49.56 ± 0.57	$\mathbf{50.11 \pm 0.89}$	49.73 ± 0.64	
p	50	75	**92**	95	99
	47.84 ± 0.91	49.21 ± 0.69	$\mathbf{50.11 \pm 0.89}$	49.87 ± 0.73	49.42 ± 0.81
ε	10^{-3}	$\mathbf{10^{-4}}$	10^{-5}		
	49.18 ± 0.66	$\mathbf{50.11 \pm 0.89}$	49.67 ± 0.72		

assume equal variances:

$$t = \frac{\bar{x}_{\text{GSOM}} - \bar{x}_{\text{B}}}{\sqrt{s^2_{\text{GSOM}}/n_{\text{GSOM}} + s^2_{\text{B}}/n_{\text{B}}}}, \tag{15}$$

with degrees of freedom given by the Welch–Satterthwaite approximation.

Table 16: Welch's t-tests on Split CIFAR-10: GSOM + ResNet-18 (Global FT, 58.10 ± 1.42, $n = 20$) against reported baseline statistics, except DendSOM, which doesn't use any pretrained encoder so we compare it with our VAE+GSOM results. Δ is the accuracy difference in percentage points. Baselines are sorted by mean accuracy.

Baseline	n	**Mean**	Δ	t	df	p-value
Fine-tune	5	18.55	+39.55	112.34	23.0	5.4×10^{-33}
A-GEM	10	18.49	+39.61	107.09	27.6	1.1×10^{-37}
GEM	5	24.13	+33.97	29.67	4.7	1.6×10^{-6}
ER+GMED	20	34.84	+23.26	39.73	32.5	4.1×10^{-29}
iCaRL	5	37.32	+20.78	16.88	4.6	2.6×10^{-5}
GSS	5	38.45	+19.65	27.83	6.2	9.5×10^{-8}
CoPE-CE	5	39.73	+18.37	17.34	4.8	1.6×10^{-5}
reservoir	5	42.48	+15.62	11.19	4.4	2.0×10^{-4}
MIR	5	42.80	+15.30	14.68	4.8	3.3×10^{-5}
CN-DPM	5	45.21	+12.89	39.35	21.1	3.1×10^{-21}
DendSOM	10	46.79	+3.32	16.48	19.9	4.4×10^{-13}
ER$_a$	20	46.29	+11.81	17.31	28.8	9.4×10^{-17}
CoPE	5	48.92	+9.18	13.70	6.5	4.7×10^{-6}
DCM-SE	5	50.02	+8.08	13.62	7.6	1.3×10^{-6}
Dynamic OCM	5	51.27	+6.83	9.36	6.0	8.3×10^{-5}

As we see from Table 16, our method significantly outperforms all baselines. The largest p-value is 2.0×10^{-4} (reservoir), and several are below 10^{-5}. DendSOM (Pinitas et al., 2021) uses no pretrained encoder, so we compare it against our non-pretrained VAE-GSOM (50.11 ± 0.89, $n = 20$) on Split CIFAR-10 instead: $\Delta = +3.32$, $t = 16.48$, $\text{df} = 19.9$, $p = 4.4\times10^{-13}$. All comparisons remain significant under a Bonferroni correction ($\alpha = 0.05/15 \approx 3.3\times10^{-3}$). These are cross-study comparisons against reported summary statistics. They indicate that the observed differences are unlikely under the reported variance. They do not substitute for a matched-seed comparison, because run counts, class orderings, and implementations differ across papers.

H Generative Capability of the Proposed Method

Our proposed model can also be used as a generative model, which is capable of synthesizing samples without storing original data. For MNIST, as we only use the SOM for training, image samples are generated using Gaussian sampling as described in the paper. After training, each SOM unit stores the specific statistics in the form of mean and variance, and they are used to generate sample images. Figure 3 illustrates some of the samples generated for each class using this method.

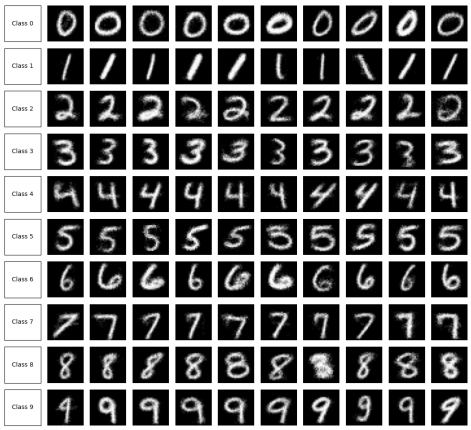


Figure 3: Synthetic samples generated using Gaussian sampling from the trained SOM. Each row corresponds to one class (0–9), and each row contains 10 synthetic samples generated from BMUs labeled with that class.



(a) CIFAR10 (b) CIFAR100

Figure 4: Synthetic CIFAR-10/100 samples generated by sampling latent vectors from a Self-Organizing Map (SOM) trained on VAE latent space. Each latent vector was sampled from the full-covariance Gaussian distribution of a Best Matching Unit (BMU) labeled with the target class. We then decoded the sampled latents into images using the VAE. Each row corresponds to one class (0–9).

For other datasets, we combine a VAE and the SOM trained on the VAE's latent space, where, after training, the SOM statistics store the mean, variance, and full covariance matrix of the latent samples. Synthetic latent samples are then generated using eigen-decomposition of the covariance, and then decoded using the VAE decoder to reconstruct the images. Figure 4a and 4b illustrate some of the samples generated for each class using this method.

H.1 Quantitative Evaluation of Replay Quality

The qualitative samples in Figure 3–4 show that decoded BMU statistics are visually coherent, but replay quality is better assessed quantitatively, under the conditions that actually occur during training.

The baselines in this comparison differ from those in Table 2 because the two evaluations address different questions and the prior literature reports one or the other. The methods below are generative continual learning approaches evaluated by sample quality and do not report class-incremental accuracy under our protocol, so they cannot be placed in the accuracy comparison; conversely, most

of the discriminative baselines in Table 2 report no generative metrics. CN-DPM (Lee et al., 2020) reports both and therefore appears in each table, providing a common reference point across the two evaluations.

We evaluate two regimes on Split CIFAR-10. In reconstruction, a real image is encoded and immediately decoded, which measures the encoder–decoder pair in isolation and is the protocol under which prior work reports. In ancestral sampling, a latent is drawn from a BMU's fitted Gaussian with no real image involved; this is precisely the path taken by replayed samples during training, and it additionally exercises the per-unit Gaussian assumption. Reporting both isolates how much fidelity is lost by replacing a real latent with a sampled one.

Table 17: Replay quality on Split CIFAR-10, measured as Inception Score (higher is better) and FID (lower is better). Reconstruction encodes and decodes a real image; ancestral sampling draws a latent from a BMU's Gaussian, as occurs during replay, and is reported additionally as a stricter condition that prior work does not evaluate.

Method	**Setting**	**IS** ↑	**FID** ↓
LIMix (Ye & Bors, 2021)	Reconstruction	3.02	156.46
VAE-ELBO-Random (Ye & Bors, 2022b)	Reconstruction	3.84	116.26
VAE-ELBO-OCM (Ye & Bors, 2022b)	Reconstruction	4.13	98.76
CN-DPM (Lee et al., 2020)	Reconstruction	4.12	95.23
Dynamic-ELBO-OCM (Ye & Bors, 2022b)	Reconstruction	–	**92.99**
Ours (GSOM)			
GSOM + ResNet-18	Reconstruction	**4.16**	94.00
GSOM + ResNet-18	Ancestral	4.02	100.00
GSOM + CLIP	Reconstruction	4.22	92.00
GSOM + CLIP	Ancestral	4.11	98.00

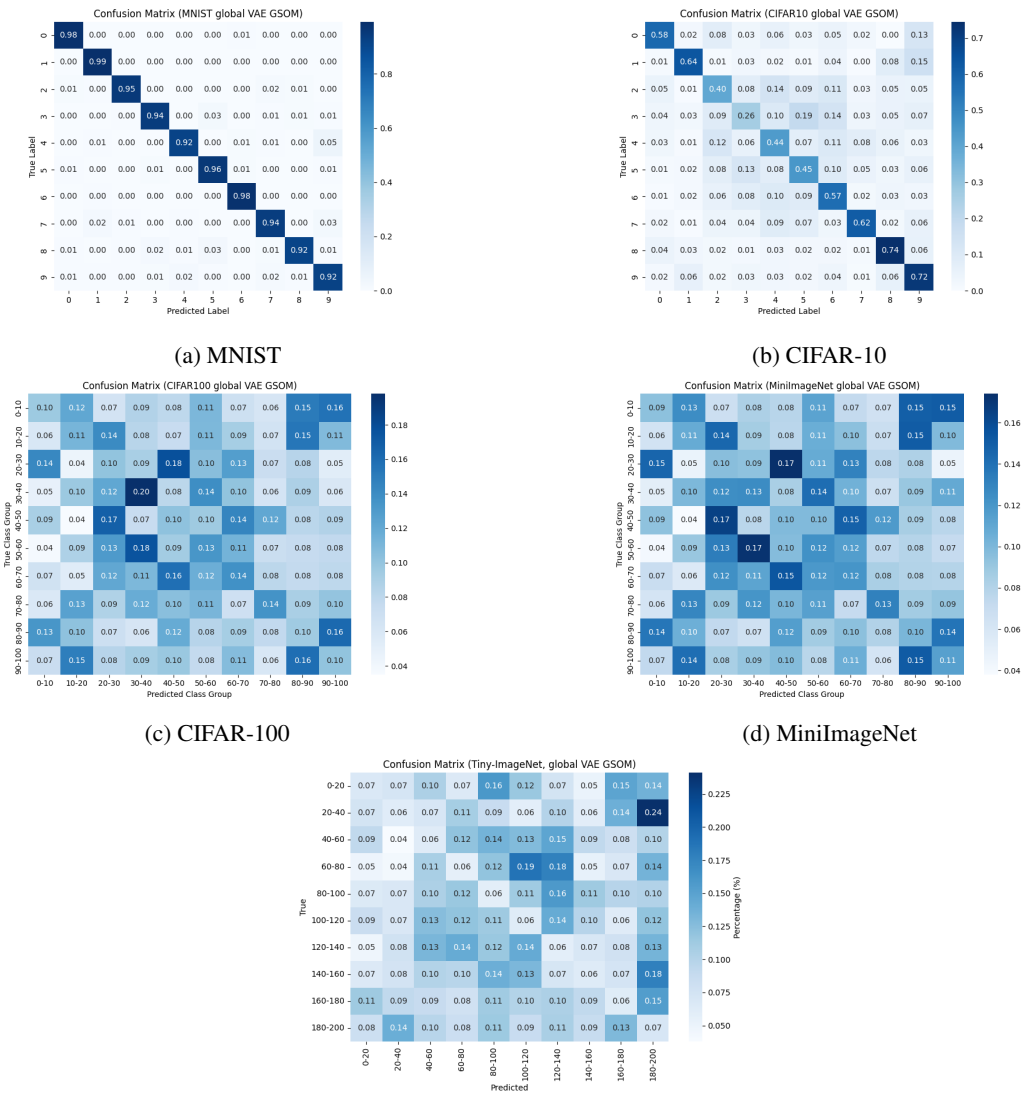
Our ResNet-18 configuration improves on all baselines except Dynamic-ELBO-OCM, which it is within roughly one FID point of (94.00 against 92.99), while attaining the highest Inception Score in the table (4.16). Our CLIP results are reported for completeness; a stronger encoder should be expected to improve sample quality independently of the replay mechanism, and we do not claim the improvement over ResNet-backbone baselines on that basis.

Moving from a real latent (reconstruction) to one drawn from a BMU's fitted Gaussian (ancestral sampling) costs only 6 FID points and 0.10–0.14 IS, and this holds for both encoder settings despite their latent spaces differing substantially in structure and quality. A poorly fitting per-unit Gaussian would degrade fidelity considerably relative to using a real latent; that the penalty is small, and that it replicates across two independent latent spaces, indicates the distributional assumption underlying our replay mechanism is not the limiting factor in sample quality.

I Class Level Analysis

To further analyze model behavior, we present row-normalized confusion matrices under the single-class incremental learning (SCIL) setting in Figure 5. We analyze the global VAE–GSOM behavior using row-normalized confusion matrices, which reveal how prediction mass is distributed for each true class as dataset complexity increases. For larger datasets (CIFAR-100, MiniImageNet, and TinyImageNet), classes are grouped into coarse categories to improve interpretability.

For MNIST, predictions are almost entirely concentrated along the diagonal, indicating strong class separability and effective retention with minimal interference. On CIFAR-10, the diagonal remains dominant but becomes broader, with noticeable confusion among visually similar classes, reflecting partial overlap in learned representations. This spread increases for CIFAR-100 and MiniImageNet, where predictions are more distributed across related groups. While correct classes still receive higher mass, the diagonals are less pronounced, and structured cross-group confusion becomes more evident. On TinyImageNet, the most challenging setting, the matrix is highly diffuse, with weak diagonal dominance and significant mixing across groups. Despite this, the predictions are not entirely random, and some residual structure is preserved.



(a) MNIST

(b) CIFAR-10

(c) CIFAR-100

(d) MiniImageNet

(e) TinyImageNet

Figure 5: Row-normalized confusion matrices for the global VAE–GSOM model across datasets. Each row shows the prediction distribution for a given true class.

Overall, these matrices show a gradual shift from concentrated, confident predictions to more distributed outputs as task difficulty increases, highlighting both the robustness and the limitations of replay in complex, fine-grained scenarios.

J Continual Learning Metrics

Tables 18–21 report the full per-task accuracy matrices and corresponding continual learning metrics for CIFAR-10 and CIFAR-100 using the GSOM with a globally trained VAE (from scratch). We evaluate forward transfer (FWT), backward transfer (BWT), and forgetting using the standard formulation ((Lopez-Paz & Ranzato, 2022)).

Across both datasets, the structure of the R matrices reveals a gradual degradation of performance along the rows, indicating the presence of catastrophic forgetting. This effect is relatively mild for CIFAR-10, where earlier tasks retain moderate accuracy, but it becomes significantly more pronounced for CIFAR-100 due to increased task complexity and class diversity. Since the model operates in

a single-head, fully unsupervised class-incremental setting, it has no prior capability on unseen classes, resulting in $R_{t-1,t} \approx 0$ before learning task t. Consequently, FWT is consistently negative and approximately equal in magnitude to the per-class baseline accuracy, confirming the absence of forward transfer. Backward transfer (BWT) is negative for early tasks, reflecting degradation in performance as new tasks are learned. For later tasks, BWT approaches zero and occasionally becomes slightly positive, indicating reduced interference and minor stabilization effects. Forgetting follows a similar trend: earlier tasks exhibit higher forgetting due to cumulative updates, while later tasks show reduced forgetting as fewer subsequent updates occur. The final task does not experience forgetting, as no further updates are applied after it is learned.

Table 18: Per-task class accuracy matrix R for CIFAR-10 across 10 sequential tasks for GSOM with a global VAE (latent dimension $32 \times 2 \times 2$).

t \ j	0	1	2	3	4	5	6	7	8	9
0	1	–	–	–	–	–	–	–	–	–
1	0.823	0.781	–	–	–	–	–	–	–	–
2	0.692	0.724	0.703	–	–	–	–	–	–	–
3	0.641	0.701	0.553	0.582	–	–	–	–	–	–
4	0.613	0.693	0.427	0.521	0.503	–	–	–	–	–
5	0.592	0.674	0.391	0.402	0.481	0.434	–	–	–	–
6	0.571	0.651	0.364	0.352	0.443	0.423	0.552	–	–	–
7	0.553	0.632	0.352	0.341	0.412	0.401	0.541	0.603	–	–
8	0.482	0.604	0.331	0.332	0.391	0.392	0.553	0.592	0.703	–
9	0.504	0.583	0.351	0.253	0.423	0.421	0.562	0.603	0.712	0.701

Table 19: Per-task Forward Transfer (FWT), Backward Transfer (BWT), and Forgetting for CIFAR-10 using GSOM with a VAE trained from scratch.

Task	**FWT**	**BWT**	**Forgetting**
0	–	-0.4960	0.4960
1	-0.6810	-0.1980	0.1980
2	-0.5130	-0.3520	0.3520
3	-0.6860	-0.3290	0.3290
4	-0.4900	-0.0800	0.0800
5	-0.5540	-0.0130	0.0130
6	-0.2680	0.0090	-0.0090
7	-0.4010	0.0070	-0.0070
8	-0.6150	0.0090	-0.0090
9	-0.8130	0.0000	–

Table 20: Task accuracy matrix R for CIFAR-100 across 10 sequential tasks for GSOM with a global VAE (latent dimension $32 \times 2 \times 2$).

t \ j	0	1	2	3	4	5	6	7	8	9
0	0.937	–	–	–	–	–	–	–	–	–
1	0.781	0.574	–	–	–	–	–	–	–	–
2	0.384	0.356	0.342	–	–	–	–	–	–	–
3	0.342	0.351	0.329	0.318	–	–	–	–	–	–
4	0.311	0.326	0.304	0.317	0.292	–	–	–	–	–
5	0.284	0.302	0.281	0.297	0.278	0.265	–	–	–	–
6	0.263	0.281	0.267	0.279	0.264	0.252	0.241	–	–	–
7	0.241	0.259	0.247	0.262	0.251	0.239	0.229	0.221	–	–
8	0.181	0.198	0.193	0.204	0.199	0.191	0.186	0.179	0.173	–
9	0.106	0.112	0.101	0.198	0.102	0.134	0.137	0.136	0.095	0.101

The matrices in Tables 18–21 report the from-scratch VAE configuration. Since the per-task structure of these metrics follows from the single-head, unsupervised protocol rather than from the choice of encoder, we do not reproduce full accuracy matrices for every backbone. Table 22 instead summarizes our two pretrained-encoder configurations, with the full matrix for the stronger of the two given in Table 23 for verification. The setting using CLIP as a backbone not only attains higher final accuracy but forgets 37% less than ResNet-18, indicating that a stronger embedding space improves *retention*

Table 21: Per-task Forward Transfer (FWT), Backward Transfer (BWT), and Forgetting for CIFAR-100 using GSOM with a global VAE.

Task	FWT	BWT	Forgetting
0	–	-0.8310	0.8310
1	-0.1020	-0.4620	0.4620
2	-0.0890	-0.2410	0.2410
3	-0.0820	-0.1200	0.1200
4	-0.0790	-0.1900	0.1900
5	-0.0750	-0.1310	0.1310
6	-0.0730	-0.1040	0.1040
7	-0.0710	-0.0850	0.0850
8	-0.0690	-0.0780	0.0780
9	-0.0670	0.0000	–

and not merely the instantaneous separability of classes. A better-conditioned latent space yields per-unit Gaussians that remain valid descriptions of their assigned data for longer, which makes replay more faithful as the stream progresses.

Table 22: Average accuracy, forgetting, and FWT on CIFAR-10 under the single-class incremental protocol.

Encoder	Final Acc	Mean Forgetting	Mean FWT
ResNet-18	58.21 ± 1.12	0.124	-0.418
CLIP	76.11 ± 0.72	**0.078**	$\mathbf{-0.235}$

Table 23: Per-task class accuracy matrix R for CIFAR-10 across 10 sequential classes for GSOM with a frozen CLIP encoder under the single-class incremental protocol.

t \ j	0	1	2	3	4	5	6	7	8	9
0	1.000	–	–	–	–	–	–	–	–	–
1	0.956	0.855	–	–	–	–	–	–	–	–
2	0.921	0.844	0.800	–	–	–	–	–	–	–
3	0.898	0.835	0.790	0.790	–	–	–	–	–	–
4	0.876	0.823	0.777	0.779	0.805	–	–	–	–	–
5	0.857	0.814	0.766	0.766	0.794	0.815	–	–	–	–
6	0.841	0.804	0.757	0.753	0.782	0.799	0.820	–	–	–
7	0.824	0.794	0.744	0.741	0.767	0.781	0.801	0.825	–	–
8	0.809	0.787	0.737	0.731	0.751	0.764	0.779	0.797	0.822	–
9	0.800	0.780	0.730	0.720	0.740	0.750	0.760	0.770	0.780	0.781

K Encoder Decoder architectures

For experiments involving pretrained encoders, we adopt a unified decoding strategy to reconstruct images from latent representations. Specifically, the ResNet-18 encoder (Table 25) produces a 512-dimensional feature vector, which is projected to a low-resolution spatial grid and progressively upsampled through a series of residual and transposed convolution blocks to recover the image. In parallel, CLIP-based configurations (Table 26) follow a similar decoding pipeline, but incorporate higher-capacity channels and attention modules to better capture the structure of semantic embeddings. By default, pretrained encoders are kept frozen to preserve their learned representations, with optional fine-tuning of the final layers (e.g., the last residual block in ResNet-18 or the final transformer block in CLIP) to improve adaptation in the continual learning setting. For models trained from scratch, we employ the residual VAE architecture (Table 24), which encodes images into compact latent representations and reconstructs them symmetrically using ResUp blocks. Across all configurations, decoded samples are used for generative replay, where latent representations are mapped onto GSOM grids of varying sizes depending on dataset complexity.

Table 24: Architecture of the VAE model used for CIFAR-10/100 (input size: $3 \times 32 \times 32$). ResDown and ResUp denote residual blocks with downsampling and upsampling, respectively. All activations are ELU except for the output layer, which uses Tanh.

Layer	Kernel / Type	Stride / Scale	Activation	Skip	Output Shape	Params
		Encoder				
Conv2d (Input)	3×3 / Conv	1	–	No	$32 \times 32 \times 32$	896
ResDown Block 1	3×3 / ×2	2 + 1	ELU	Yes	$64 \times 16 \times 16$	46.3K
ResDown Block 2	3×3 / ×2	2 + 1	ELU	Yes	$128 \times 8 \times 8$	184.6K
ResDown Block 3	3×3 / ×2	2 + 1	ELU	Yes	$256 \times 4 \times 4$	737.9K
ResDown Block 4	3×3 / ×2	2 + 1	ELU	Yes	$512 \times 2 \times 2$	2.36M
ResBlock	3×3 / ×2	1 + 1	ELU	Yes	$512 \times 2 \times 2$	2.36M
Conv2d (μ)	1×1 / Conv	1	–	No	$32 \times 2 \times 2$	16.4K
Conv2d ($\log\sigma^2$)	1×1 / Conv	1	–	No	$32 \times 2 \times 2$	16.4K
		Decoder				
Conv2d (Latent)	1×1 / Conv	1	ELU	No	$512 \times 2 \times 2$	16.9K
ResBlock	3×3 / ×2	1 + 1	ELU	Yes	$512 \times 2 \times 2$	2.36M
ResUp Block 1	3×3 / ×2	Upsample×2	ELU	Yes	$256 \times 4 \times 4$	627.4K
ResUp Block 2	3×3 / ×2	Upsample×2	ELU	Yes	$128 \times 8 \times 8$	295.2K
ResUp Block 3	3×3 / ×2	Upsample×2	ELU	Yes	$64 \times 16 \times 16$	184.5K
ResUp Block 4	3×3 / ×2	Upsample×2	ELU	Yes	$32 \times 32 \times 32$	82.9K
Conv2d (Output)	5×5 / Conv	1	Tanh	No	$3 \times 32 \times 32$	2.4K
Total Parameters						**9.3M**

Table 25: Decoder architecture used for ResNet-18 + GSOM experiments. The encoder is an ImageNet-pretrained ResNet-18 that outputs a 512-dimensional feature vector. The decoder maps this embedding back to the dataset-native RGB image resolution (32×32 for CIFAR-10/100, 64×64 for TinyImageNet, 84×84 for MiniImageNet). GN denotes GroupNorm.

Layer	Operation / Details	Activation	Norm	Output Shape
Linear projection	FC($512 \to 512 \cdot 4 \cdot 4$), reshape	SiLU	GN	$B \times 512 \times 4 \times 4$
ResBlock	Conv 3×3 + Conv 3×3	SiLU	GN	$B \times 512 \times 4 \times 4$
Upsample Block 1	ConvTranspose2d 4×4, stride 2	SiLU	GN	$B \times 256 \times 8 \times 8$
ResBlock	Conv 3×3 + Conv 3×3	SiLU	GN	$B \times 256 \times 8 \times 8$
Upsample Block 2	ConvTranspose2d 4×4, stride 2	SiLU	GN	$B \times 128 \times 16 \times 16$
ResBlock	Conv 3×3 + Conv 3×3	SiLU	GN	$B \times 128 \times 16 \times 16$
Upsample Block 3	ConvTranspose2d 4×4, stride 2	SiLU	GN	$B \times 64 \times 32 \times 32$
ResBlock	Conv 3×3 + Conv 3×3	SiLU	GN	$B \times 64 \times 32 \times 32$
Optional Upsample Block	ConvTranspose2d 4×4, stride 2; used only for TinyImageNet/MiniImageNet	SiLU	GN	$B \times 64 \times 64 \times 64$
Output Head	Conv 3×3 $\to$ 3 channels	Tanh	–	$B \times 3 \times H \times W$
Resize (MiniImageNet only)	Bilinear interpolation $64 \times 64 \to 84 \times 84$	–	–	$B \times 3 \times 84 \times 84$

L Visualization of Learned Representations

Figure 6–8 present the evolution of GSOM unit representations across tasks for MNIST, CIFAR10, and CIFAR100 datasets. For MNIST, the learned representations quickly organize into well-structured and visually coherent prototypes, with clear class-specific patterns emerging after each task and remaining relatively stable over time. In CIFAR-10, the decoded representations are more diverse but still exhibit recognizable semantic structure, indicating that the global VAE is able to capture meaningful features despite increased complexity. For CIFAR-100, the representations become significantly more fragmented and less visually distinct, reflecting the higher intra-class variability and task difficulty. Overall, these visualizations highlight the ability of GSOM to progressively organize latent space representations, while also illustrating the challenges in maintaining distinct and stable prototypes under more complex data distributions.

Table 26: Decoder Architecture used for CLIP+SOM experiments. The encoder is the CLIP ViT-B/32 visual tower (pretrained) and is *frozen* by default; when `finetune_last` is enabled, only the last transformer block of ViT-B/32 is unfrozen. The decoder upsamples a single CLIP embedding to the target RGB image in $[-1, 1]$. GN denotes GroupNorm(32).

Layer	Operation / Details	Blocks	Activation	GN	Output Shape
Linear (embed→4×4)	FC($D \to$base_ch$\cdot 4 \cdot 4$)	–	–	Yes	$B \times 1024 \times 4 \times 4$
ResBlock	Conv 3×3 (GN+SiLU)	1+1	SiLU	Yes	$B \times 1024 \times 4 \times 4$
Upsample	ConvTranspose2d 4×4	×2	–	Yes	$B \times 1024 \times 8 \times 8$
ResBlock (ch↓)	Conv 3×3 (GN+SiLU)	1+1	SiLU	Yes	$B \times 512 \times 8 \times 8$
Upsample	ConvTranspose2d 4×4	×2	–	Yes	$B \times 512 \times 16 \times 16$
ResBlock (+Attn)	Conv 3×3 + SelfAttn2d	1+1	SiLU	Yes	$B \times 512 \times 16 \times 16$
ResBlock (ch↓)	Conv 3×3 (GN+SiLU)	1+1	SiLU	Yes	$B \times 256 \times 16 \times 16$
Upsample	ConvTranspose2d 4×4	×2	–	Yes	$B \times 256 \times 32 \times 32$
ResBlock (+Attn)	Conv 3×3 + SelfAttn2d	1+1	SiLU	Yes	$B \times 256 \times 32 \times 32$
ResBlock (ch↓)	Conv 3×3 (GN+SiLU)	1+1	SiLU	Yes	$B \times 128 \times 32 \times 32$
Upsample	ConvTranspose2d 4×4	×2	–	Yes	$B \times 128 \times 64 \times 64$
ResBlock	Conv 3×3 (GN+SiLU)	1+1	SiLU	Yes	$B \times 128 \times$ target $\times$ target
Output head	GN → SiLU → 3×3 → Tanh	1	Tanh	Yes	$B \times 3 \times$ target $\times$ target
Resize (MiniImageNet only)	Bilinear interpolation $64 \times 64 \to 84 \times 84$	–	–	–	$B \times 3 \times 84 \times 84$

Notes: Encoder output dim D=512 for ViT-B/32. SelfAttention2d uses 4 heads with GN(32). By default, only the decoder is trainable; enabling `finetune_last` unfreezes the last ViT block. Decoder output target size is set to match dataset resolution (32×32 for CIFAR-10/100, 64×64 for TinyImageNet, 84×84 for MiniImageNet).

M Ethical Statement

This work uses only publicly available benchmark datasets (MNIST, CIFAR-10/100, TinyImageNet, and MiniImageNet) and publicly released pretrained encoders; it involves no human subjects or newly collected personal data. Because our method replays synthetic samples drawn from per-unit distributional statistics rather than storing raw exemplars, it may reduce some privacy risks associated with buffer-based replay. However, summary statistics combined with a trained decoder can still produce samples that resemble training data, and we have not formally evaluated memorization or privacy leakage. Continually adapting models may also inherit biases from evolving data streams or exhibit unreliable behavior under distribution shifts. In safety-critical or decision-making applications, such failures could lead to unintended consequences if models are deployed without appropriate oversight and evaluation. This work is intended for research purposes; responsible deployment would require further safeguards, including bias auditing and privacy evaluation.

N AI Disclosure

We have not used generative AI tools for research ideas, method design, code, experiments, data analysis, or writing the content of this work. Additionally, we used generative AI tools for proofreading (spelling and grammar). We have reviewed all AI-assisted work: all proofreading suggestions were manually checked by the authors before being accepted, and no text was generated by AI. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

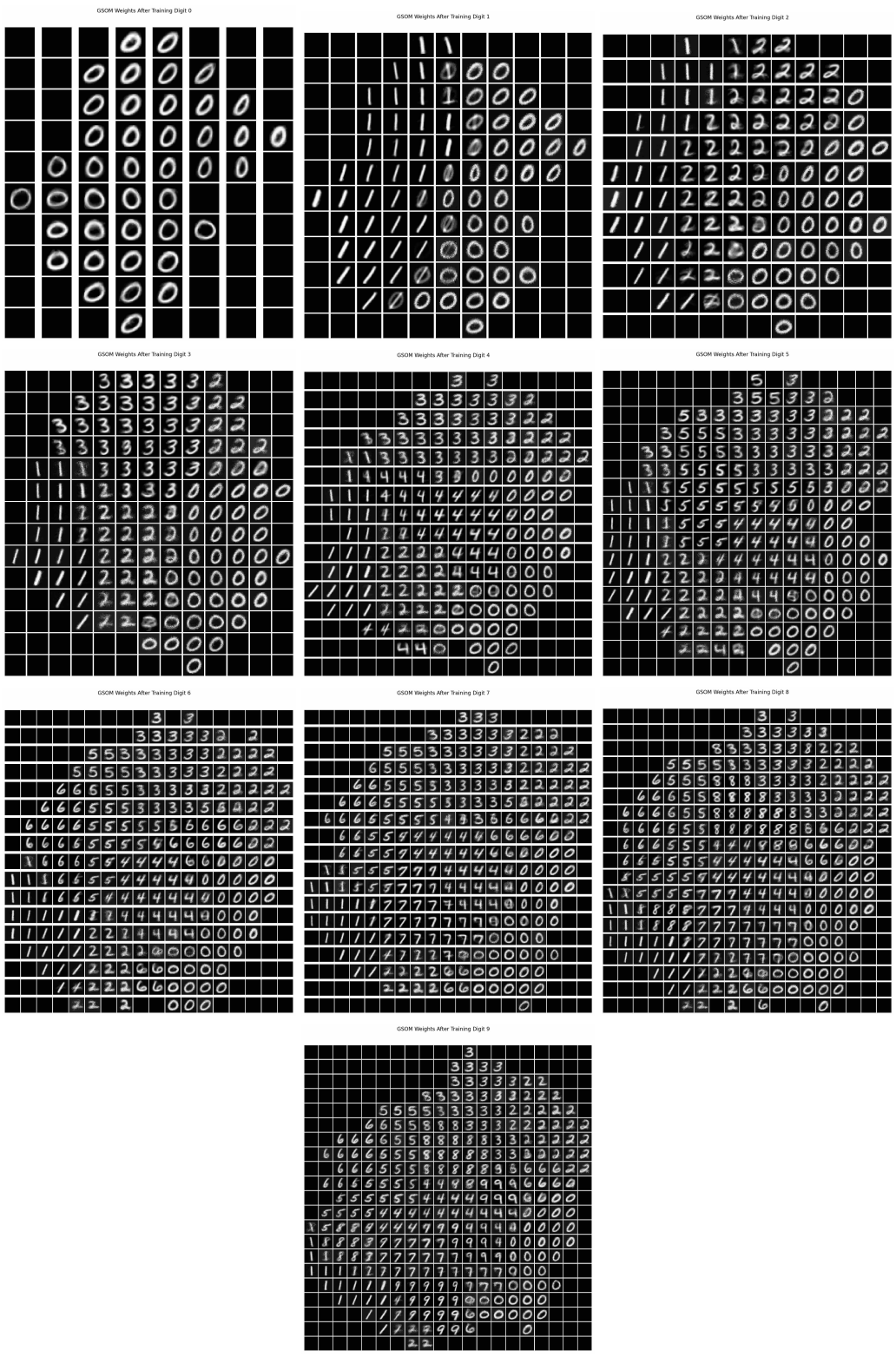


Figure 6: Visualization of the GSOM unit vectors after each task(every single task) for MNIST.

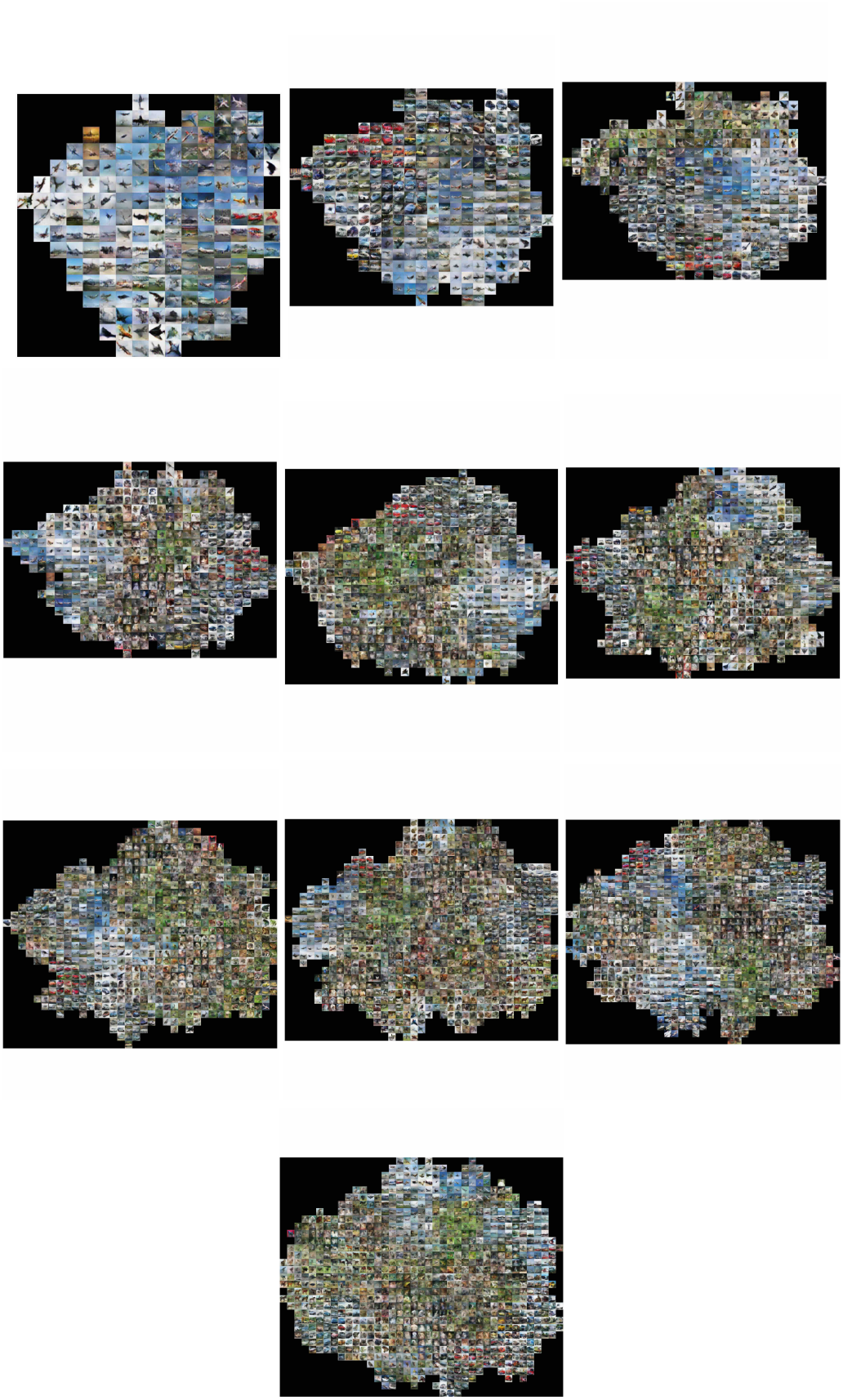


Figure 7: Visualization of decoded GSOM unit vectors after each task(every class) on CIFAR-10.

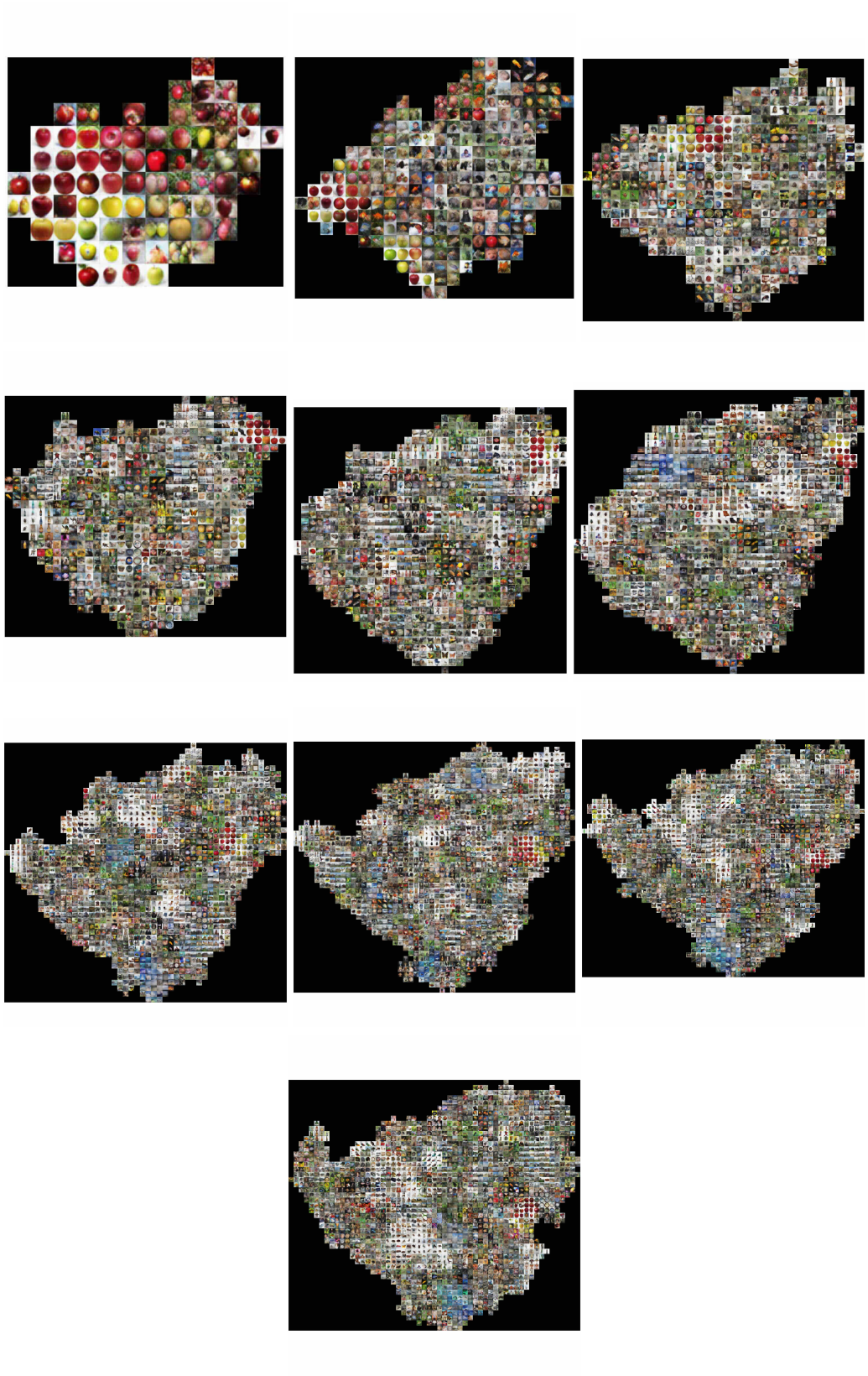


Figure 8: Visualization of decoded GSOM unit vectors after each task(every 10 classes) on CIFAR-100.