Beyond Exemplar Selection: Value-Aware Memory Allocation in Replay-Based Continual Learning

Yuhang Li
Guangdong University of Technology
liyuhang34@mails.gdut.edu.cn

Guoxu Zhou*
Guangdong University of Technology
gx.zhou@gdut.edu.cn

Zhenhao Huang
Guangdong University of Technology
zhenhao.huang@gdut.edu.cn

Yuning Qiu
RIKEN AIP
yuning.qiu@riken.jp

Qibin Zhao
RIKEN AIP
qibin.zhao@riken.jp

Abstract

Replay is a standard strategy for mitigating catastrophic forgetting in continual learning, where a bounded memory buffer stores exemplars from previously seen classes. Existing replay methods have largely focused on instance-level decisions, such as which samples to store, replay, or replace, while the class-wise allocation of memory slots is often treated as a uniform heuristic. Here, we study class-wise memory allocation as a complementary design axis and propose VAMA, a value-aware memory allocation framework. VAMA keeps cross-task budgets balanced, estimates task-local class values from Shapley-inspired sample utility scores, and converts them into class-wise quotas through a regularized allocation rule that conservatively deviates from class-balanced replay. This design changes only the class-wise budgeting rule and leaves within-class exemplar selection unchanged, making VAMA easy to integrate into replay pipelines. Experiments on CIFAR-100 and ImageNet-100 show that VAMA consistently improves ER, iCaRL, and FOSTER across different memory budgets and task protocols. Further analyses show that the improvements are not due to non-uniform allocation alone, and that allocation-relevant class-value rankings stabilize early in training, enabling efficient value-aware allocation. These results suggest that class-wise memory allocation is a consequential design choice in replay-based continual learning.

1 Introduction

Continual Learning (CL) [1, 2, 3, 4, 5] aims to enable models to acquire new knowledge over time without catastrophically forgetting [6, 7, 8] previously acquired knowledge. A dominant family of approaches for alleviating forgetting is replay [9, 10, 11, 12, 13, 14], in which a bounded buffer stores a small subset of old exemplars and interleaves them with new training data. Most replay methods primarily focus on instance-level decisions, including well-known strategies such as herding [9, 15], coreset construction [16, 17], and sample selection based on gradient [17, 18] or utility [19, 20] criteria. Although these methods differ in how they identify representative, diverse, or informative samples, they largely optimize the same question: *which* examples should be stored, replayed, or replaced in the buffer. In contrast, another design choice, subtle but consequential, is often left as a default: *how* a memory budget should be allocated across classes.

A common practice is uniform or class-balanced allocation [9], which assigns memory symmetrically across classes under a fixed budget. In incremental settings, however, classes are rarely homogeneous

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

in their learning dynamics [21, 22]. Differences in label noise [23], task similarity [24, 25], and vulnerability to forgetting [21] can induce substantial variation in the marginal benefit of retaining additional exemplars. Consequently, uniform allocation may be inefficient: it can over-allocate memory to classes that are already stable while under-allocating classes for which additional replay would be more beneficial. This suggests that, under a fixed budget, replay performance depends not only on *which* samples are stored, but also on *how many* exemplar slots are assigned to each class.

To isolate this effect, we conduct a controlled toy study on Split MNIST [1, 19] with three incremental tasks, each containing two classes. We fix the total replay budget to 60 and keep the replay protocol and sample selection rule unchanged, varying only the class-wise memory allocation. To reduce confounding effects from prolonged optimization, we train each task for a single epoch, so that replay exposure is more tightly coupled to the available budget. As shown in Figure 1, different feasible allocations produce substantial differences in final performance under the same total budget, and the best allocation differs markedly from the uniform baseline. These results provide controlled evidence that, under a fixed replay budget, class-wise memory allocation can materially affect final performance and therefore deserves explicit treatment rather than a default heuristic.

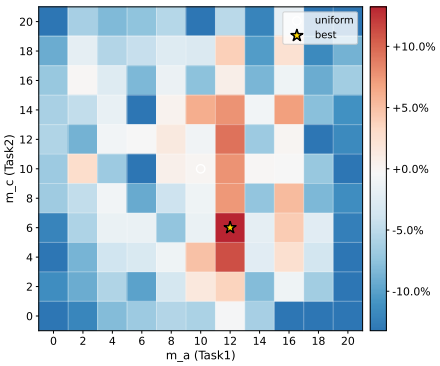


Figure 1: Controlled toy study on Split MNIST with three incremental tasks under ten seeds. The heatmap shows the change in final accuracy relative to uniform allocation for different class-wise memory quotas. Here m_a and m_c denote the memory slots assigned to one digit in Task 1 and Task 2, respectively. The best allocation outperforms uniform allocation by 13.6%.

Despite its practical relevance, class-wise memory allocation has received relatively limited attention in replay-based CL. To the best of our knowledge, the closest prior work is RMM [26], which shows that memory allocation under a fixed replay budget can be optimized dynamically rather than prescribed by a static heuristic. However, its formulation relies on an additional offline reinforcement learning stage on pseudo-tasks, which incurs substantial extra training cost and is not naturally compatible with the standard continual learning protocol [26]. Moreover, its group-level adaptivity remains relatively coarse rather than fully dynamic. This leaves open the question of how class-wise memory allocation can be performed effectively and efficiently.

To address this gap, we study class-wise memory allocation as a distinct design axis in replay-based continual learning and instantiate this perspective with a **V**alue-**A**ware **M**emory **A**llocation (**VAMA**) framework. At a high level, VAMA treats class-wise memory allocation as a budgeting problem under a fixed memory budget. The framework maintains a balanced split across tasks to preserve a stable cross-task memory structure, and then dynamically allocates memory within each task according to class-level utility scores. To acquire this utility signal, we adopt a Shapley-inspired data valuation perspective to estimate the marginal contribution of individual examples [27, 28, 29], and then aggregate these sample-level values into class-level scores for budgeting. The resulting scores are converted into class-specific memory quotas, yielding a simple and protocol-compatible alternative to static allocation rules.

Our contributions are threefold. First, we identify class-wise memory allocation as a distinct and consequential design axis in replay-based continual learning, and provide controlled evidence that allocation alone can substantially affect performance under a fixed replay budget. Second, we instantiate this perspective with VAMA, a simple value-aware allocation framework that performs class-wise budgeting based on task-local class utility signals while remaining compatible with standard replay pipelines. Third, we show that this plug-in design consistently improves replay-based continual learning across representative baselines, memory budgets, and task protocols.

2 Related Work

Continual Learning (CL), also called incremental learning or lifelong learning, methods are commonly categorized into regularization-based, architecture-based, bias-correction/rebalancing, and

replay-based approaches [1, 2, 3, 5]. Regularization-based methods reduce forgetting by constraining parameter updates [30, 31, 32], while architecture-based methods alleviate interference through parameter isolation [33, 34], expansion [35, 36], or modularization [37, 38]. In parallel, an important line of CL research addresses the old-new imbalance problem through classifier rebalancing or bias correction [39, 40, 41, 42]. Our work is most closely related to the replay-based family, and in particular to exemplar-based replay under a bounded memory budget.

In **exemplar-based replay**, a limited set of past samples is stored and rehearsed together with incoming data [9, 19]. Prior work has explored a broad design space, including exemplar construction, replacement, rehearsal scheduling, and retrieval [1, 3, 5, 13]. A large body of work focuses on instance-level decisions, such as which samples should be stored, retained, or replayed, using heuristics based on herding [9, 15], coreset [16, 17], gradients [18, 17], interference [19], or sample utility [20]. These methods improve the quality of memory contents under a fixed budget. Our focus is complementary: rather than introducing a new within-class exemplar selection rule, we study class-wise memory allocation as a separate design axis and ask how many memory slots should be assigned to each class under a fixed budget.

The closest line of work is **adaptive memory allocation**. Recent methods such as RMM [26] move beyond static allocation by learning a memory-management policy that adapts allocation decisions across incremental phases and classes. Our work differs in problem formulation and integration: instead of introducing a separate offline policy-learning stage on pseudo CL tasks, we study class-wise memory allocation directly within the target replay-based CL training pipeline. While replay can also be implemented in latent or generative form [43, 44], we focus on explicit exemplar memory, where class-wise allocation under a fixed storage budget is a first-class design choice.

3 Method

3.1 Problem Formulation

Continual Learning [5, 3] aims to train a model on a sequence of tasks $\{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^T\}$, while preserving performance on previously learned classes. The training set of task t contains N_t labeled examples, $\mathcal{D}^t = \{(x_n^t, y_n^t)\}_{n=1}^{N_t}$, where $x_n^t \in \mathcal{X}_t$ denotes the input and $y_n^t \in \mathcal{Y}_t$ denotes its label. Here we consider the standard Class-Incremental Learning (CIL) setting, where tasks arrive sequentially and each task introduces a disjoint set of new classes, i.e., $\mathcal{Y}_t \cap \mathcal{Y}_{t'} = \varnothing$ for $t \neq t'$. At test time, the model is evaluated over the union of all classes observed so far, without access to task identity.

Ideally, the learner seeks parameters θ that perform well on all tasks observed throughout the sequence, which can be abstractly written as:

$$\min_{\theta} \sum_{t=1}^{T} \mathcal{L}_t, \quad \text{where} \quad \mathcal{L}_t \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}^t} \left[\ell(f_\theta(x), y)\right]. \tag{1}$$

However, under the standard CIL protocol, the full training data from previous tasks are no longer accessible once training proceeds to a new task; directly optimizing this objective will result in catastrophic forgetting.

Replay-based methods [9, 13, 14] mitigate this problem by maintaining a memory buffer $\mathcal{M}$ that stores a subset of old exemplars under a fixed memory budget $|\mathcal{M}| \leq M$. The replay objective at task t can be written as:

$$\mathcal{L}_t^{\text{replay}} = \mathbb{E}_{(x,y)\sim\mathcal{D}^t} \left[\ell(f_\theta(x), y)\right] + \alpha \mathbb{E}_{(x,y)\sim\mathcal{M}} \left[\ell(f_\theta(x), y)\right]. \tag{2}$$

Most replay-based methods focus on which exemplars should be retained or replayed in the buffer. In contrast, our focus is on a complementary question: under a fixed replay budget, how should memory be allocated across classes?

3.2 Cross-Task Balanced Budgeting

We decouple replay memory management into two levels: *cross-task budgeting* and *within-task class-wise allocation*. The former determines how the global replay budget is distributed across observed tasks, while the latter determines how the budget assigned to each task is further allocated

across its classes. In this work, we keep cross-task budgeting balanced by design and focus the adaptive component on within-task class-wise allocation.

Formally, let M denote the total replay budget. After learning task t, each observed task is assigned the same target budget

$$K_t = \left\lfloor \frac{M}{t} \right\rfloor. \tag{3}$$

Let $\mathcal{M}_j^{(t)}$ denote the memory retained for task j after finishing task t. Then, for all $j \le t$,

$$|\mathcal{M}_j^{(t)}| \le K_t, \qquad \sum_{j=1}^{t} |\mathcal{M}_j^{(t)}| \le tK_t \le M. \tag{4}$$

Accordingly, after task t is learned, the new task receives budget K_t, while each old task is compressed from K_{t-1} to K_t. For old tasks, we preserve their previously estimated intra-task allocation proportions and rescale them to the new task-level budget, followed by the same rounding rule described in Sec. 3.4. This avoids repeatedly re-estimating old task values while maintaining a balanced cross-task memory structure.

We adopt balanced cross-task budgeting for two reasons. First, a fully dynamic cross-task budgeting policy introduces an additional decision layer beyond the target replay pipeline, which is not naturally aligned with the standard CIL protocol. Second, and more importantly, our value signal is estimated from a task-local validation objective, as described in Sec. 3.3. Its magnitude therefore reflects relative class utility within the current task and is not calibrated for direct comparison across different tasks. We therefore use the value signal only for within-task allocation and keep cross-task budgeting balanced by design.

Under this balanced cross-task budget, the remaining problem is to distribute the task-level budget K_t among the C_t classes of the current task. The desired allocation should assign relatively more exemplar slots to classes with larger estimated replay utility, while remaining conservative under noisy task-local estimates. Formally, we seek class-wise quotas $\{m_c^t\}_{c=1}^{C_t}$:

$$\sum_{c=1}^{C_t} m_c^t = K_t, \qquad m_c^t \in \mathbb{Z}_{\ge 0}, \tag{5}$$

where m_c^t denotes the number of exemplar slots assigned to class c in task t. The next two subsections describe how we estimate task-local class-level replay utility and convert it into stable quotas.

3.3 Intra-Task Class-Value Estimation

Given the current task t, our goal is to construct a *task-local class-wise replay value signal*. We first estimate a value score for each current-task training sample, and then aggregate these sample-level scores into class-level values. Since memory allocation should reflect expected replay benefit, a natural starting point is *Data Shapley* [45, 27, 28, 29], which quantifies the value of a training sample by its marginal contribution to a utility function. In standard form, the Shapley value of a sample z is defined as the weighted average of its marginal utility $U(S \cup \{z\}) - U(S)$ over subsets S of the training data:

$$\phi_z(U) = \frac{1}{N} \sum_{k=1}^{N} \binom{N-1}{k-1}^{-1} \sum_{\substack{S \subseteq D \setminus \{z\} \\ |S| = k-1}} \big[U(S \cup \{z\}) - U(S)\big]. \tag{6}$$

In our setting, D corresponds to the current-task training set $\mathcal{D}_{\mathrm{tr}}^t$, and z denotes a training example $z_i = (x_i, y_i)$. This marginal-contribution view provides a principled motivation for defining sample value as contribution to model utility.

However, directly computing retraining-based Data Shapley is prohibitively expensive in replay-based CIL, where valuation would need to be repeated along the incremental sequence. Inspired by [29], we therefore adopt a practical first-order in-run approximation, which estimates sample contribution within a single training run rather than through repeated retraining. We do not aim to compute exact

Shapley values; instead, we use this approximation as an efficient sample-level utility proxy for memory allocation.

For each task t, we split the current-task training data $\mathcal{D}^t$ into a training subset $\mathcal{D}^t_{\mathrm{tr}}$ and a small hold-out validation subset $\mathcal{V}_t$. The hold-out set is constructed only from the current task training data and is not used for model optimization. Let

$$\mathcal{L}_{\mathrm{val}}(\theta; \mathcal{V}_t) = \frac{1}{|\mathcal{V}_t|} \sum_{(x,y)\in\mathcal{V}_t} \ell(f_\theta(x), y) \tag{7}$$

denote the average validation loss on $\mathcal{V}_t$. Following the in-run view, we measure the contribution of a training example by its first-order effect on reducing this task-local validation loss along the training trajectory. Let $B_j \subset \mathcal{D}^t_{\mathrm{tr}}$ denote the current-task training mini-batch used for value accumulation at iteration j, and let $B^{\mathrm{val}}_j \subset \mathcal{V}_t$ denote a validation mini-batch. Under a first-order approximation, the validation-loss reduction induced by sample z_i is proportional to the alignment between its training gradient and the task-local validation gradient. The sample-level value score is defined as

$$s_i = \sum_{j:\, z_i \in B_j} \eta_j \left\langle g^{\mathrm{val}}_j, g_j(z_i) \right\rangle, \qquad g^{\mathrm{val}}_j = \frac{1}{|B^{\mathrm{val}}_j|} \sum_{z\in B^{\mathrm{val}}_j} \nabla_\theta \ell(f_{\theta_j}(x), y), \tag{8}$$

where $g_j(z_i) = \nabla_\theta \ell(f_{\theta_j}(x_i), y_i)$ and η_j is the learning rate at iteration j. A positive s_i indicates that the gradient of z_i is aligned with the validation gradient, suggesting a potential reduction in the task-local validation loss. More details on Data Shapley and the in-run approximation are provided in Appendix D.

These sample-level scores are not used directly for replay retrieval, but only serve as an intermediate signal for class-level memory allocation. We aggregate them within each class using the positive mean:

$$V^t_c = \frac{1}{n^t_c} \sum_{i:\, y_i = c} \max(s_i, 0), \tag{9}$$

where n^t_c is the number of training samples in class c of task t. This measures the average beneficial task-local contribution of a class, while avoiding direct penalization from noisy negative estimates. Using the mean rather than the sum also makes the class value less sensitive to class-size variations. Empirically, positive-mean aggregation performs competitively and exhibits more stable value rankings in our ablation; see Appendix Figure 4.

3.4 Regularized Value-to-Quota Allocation

Given the task-local class values $\{V^t_c\}_{c=1}^{C_t}$, we convert them into feasible class-wise memory quotas under the fixed task-level budget K_t. Since these values are used as relative, task-local allocation preferences rather than calibrated estimates of future retention gain, the mapping should be value-aware but conservative: it should assign more memory to classes with larger estimated utility while avoiding overly concentrated quotas.

Let $\widetilde{V}^t_c \in [0, 1]$ denote the min-max normalized value of class c within task t, and let $u^t_c = 1/C_t$ denote the class-balanced allocation prior. Let $\Delta_{C_t} = \{p \in \mathbb{R}^{C_t}_+ : \sum_{c=1}^{C_t} p_c = 1\}$ denote the probability simplex over the C_t classes of task t. We first construct a smooth value-aware proposal by solving a KL-regularized linear utility objective:

$$\pi^t = \arg\max_{p\in\Delta_{C_t}} \sum_{c=1}^{C_t} p_c \widetilde{V}^t_c - \tau \mathrm{KL}(p \| u^t). \tag{10}$$

Since u^t is uniform, this objective is equivalent up to a constant to $\sum_c p_c \widetilde{V}^t_c + \tau \mathcal{H}(p)$, where $\mathcal{H}(p) = -\sum_c p_c \log p_c$. The KL term penalizes overly concentrated deviations from class-balanced replay, so Eq. 10 serves as a noise-robust value-to-quota mapping rather than a claim of globally optimal replay allocation. The closed-form solution is:

$$\pi^t_c = \frac{\exp(\widetilde{V}^t_c/\tau)}{\sum_{c'=1}^{C_t} \exp(\widetilde{V}^t_{c'}/\tau)}, \tag{11}$$

where τ controls the sharpness of the value-aware proposal. A larger τ yields a smoother proposal closer to the class-balanced prior, while a smaller τ places more emphasis on relative value differences.

Although π^t is smoothed by KL regularization, its overall deviation from uniform allocation is still determined by the value scale and τ. We therefore introduce an explicit shrinkage step toward the class-balanced prior:

$$p_c^t = (1-\lambda)\frac{1}{C_t} + \lambda \pi_c^t, \tag{12}$$

where $\lambda \in [0,1]$ controls the overall strength of value awareness. Here, τ controls how sharply the value-aware proposal distinguishes among classes, while λ controls how far the final allocation is allowed to deviate from uniform replay. Thus, VAMA can be viewed as a conservative deviation from class-balanced allocation: $\lambda = 0$ recovers standard uniform class-wise replay, while $\lambda = 1$ uses the fully value-aware proposal.

Finally, we convert p_c^t into integer quotas. Each class first receives a minimum quota $m_{\min} = 1$ to avoid class starvation. The remaining budget $R_t = K_t - C_t m_{\min}$ is distributed according to p_c^t: we set $\widehat{m}_c^t = R_t p_c^t$, assign $\lfloor \widehat{m}_c^t \rfloor$ slots, and allocate the leftover slots by the largest-remainder rule. Thus the final quota is

$$m_c^t = m_{\min} + \lfloor \hat{m}_c^t \rfloor + r_c^t, \tag{13}$$

where $r_c^t \in \{0,1\}$ indicates whether class c receives one of the remaining slots after rounding. This guarantees that the constraint in Eq. 5 is satisfied.

Note that VAMA only changes the class-wise memory budgeting rule. Once m_c^t is determined, exemplar selection within each class follows the original replay method, making VAMA a modular allocation layer for replay-based CIL.

4 Experiments

4.1 Experimental Settings

Datasets. We conducted experiments on two widely used datasets. CIFAR-100 [46] is a low-resolution and widely used dataset for image classification tasks. It comprises 60,000 32×32 RGB images categorized into 100 classes, each containing 500 training and 100 testing images. ImageNet-100 [47] is a subset of the larger ImageNet-1000 containing 100 randomly selected categories. Each category has approximately 1,300 training samples and 50 test samples.

Evaluation Metrics. We use two metrics for evaluation. *Last Accuracy* (Last) denotes the top-1 classification accuracy over all classes observed up to the current task, and is used to measure performance at each incremental stage. Since the final-stage accuracy alone does not reflect the full learning trajectory, we also report *Average Accuracy* (Avg), defined as $\bar{A}_t = \frac{1}{t}\sum_{i=1}^{t} A_{i,i}$, where $A_{i,i} \in [0,1]$ is the Last Accuracy after learning task i. We report the final value $\bar{A}_T$ in all experiments. All results are based on top-1 accuracy.

Protocols. We evaluate the proposed method on CIFAR-100 and ImageNet-100 under standard class-incremental learning settings. For both datasets, we consider 5-task, 10-task, and 20-task protocols, corresponding to different levels of incremental granularity. To examine performance under different memory capacities, we conduct experiments with budgets of 500, 1000, and 2000 exemplars. To validate the generality of the proposed allocation strategy, we integrate it into three representative replay-based baselines, namely Experience Replay (ER) [11], iCaRL [9], and FOSTER [12], and report results under all combinations of task splits and memory budgets.

Implementation Details. Our implementation is based on the PyCIL [48] framework. For both CIFAR-100 and ImageNet-100, we adopt ResNet-18 [49] as the backbone network. Following standard data valuation practice and the in-run data valuation setup [29], we reserve part of the current-task training data as a task-local hold-out set for estimating class values. This subset is used only to define the valuation objective and is not used for model optimization or test-time evaluation. Note that baselines are trained on the full training set. Unless otherwise specified, VAMA uses two hyperparameters, τ and λ. On CIFAR-100, we set $\tau = 2.25$ and $\lambda = 0.8$; on ImageNet-100, we set $\tau = 3.0$ and $\lambda = 0.9$. More implementation details are provided in Appendix A.

Table 1: Results on CIFAR-100 under different memory budgets and task protocols.

Buffer	Method	5-Task		10-Task		20-Task	
		Avg	Last	Avg	Last	Avg	Last
500	ER	51.79	31.61	46.01	24.53	42.56	20.14
	w/ VAMA	**53.41**	**33.91**	**48.14**	**27.03**	**46.17**	**24.59**
	iCaRL	53.62	31.86	44.45	22.45	42.11	18.04
	w/ VAMA	**54.94**	**33.74**	**49.27**	**28.74**	**45.25**	**23.63**
	FOSTER	64.95	44.48	62.20	43.34	57.99	40.60
	w/ VAMA	**67.33**	**48.32**	**64.17**	**48.23**	**59.57**	**41.65**
1000	ER	57.41	39.44	53.10	34.38	51.32	29.25
	w/ VAMA	**59.09**	**41.42**	**56.62**	**38.40**	**55.11**	**36.11**
	iCaRL	59.52	40.43	52.03	31.82	51.86	28.83
	w/ VAMA	**61.56**	**43.08**	**56.35**	**36.36**	**54.86**	**34.27**
	FOSTER	69.26	57.21	66.70	51.35	62.32	47.04
	w/ VAMA	**70.41**	**60.25**	**68.01**	**53.67**	**64.92**	**49.18**
2000	ER	63.08	47.27	60.03	43.08	60.19	40.89
	w/ VAMA	**64.97**	**49.32**	**63.59**	**47.22**	**62.45**	**44.20**
	iCaRL	64.74	49.11	60.10	43.30	60.94	41.81
	w/ VAMA	**66.62**	**51.79**	**63.24**	**46.90**	**62.29**	**42.18**
	FOSTER	70.86	60.69	68.39	56.58	63.79	50.12
	w/ VAMA	**73.64**	**64.57**	**70.28**	**58.52**	**68.34**	**54.77**

Table 2: Results on ImageNet-100 under different memory budgets and task protocols.

Buffer	Method	5-Task		10-Task		20-Task	
		Avg	Last	Avg	Last	Avg	Last
500	ER	48.31	27.76	44.66	22.28	37.25	17.58
	w/ VAMA	**50.00**	**29.48**	**45.91**	**25.90**	**40.96**	**20.38**
	iCaRL	56.49	33.92	45.65	22.94	40.30	17.66
	w/ VAMA	**59.63**	**38.42**	**50.55**	**26.50**	**44.98**	**22.86**
	FOSTER	63.34	34.30	62.52	40.20	59.53	41.14
	w/ VAMA	**65.64**	**38.40**	**64.01**	**42.38**	**59.72**	**42.32**
1000	ER	57.18	37.18	52.42	33.24	47.71	26.26
	w/ VAMA	**58.12**	**39.08**	**54.71**	**36.04**	**50.10**	**28.88**
	iCaRL	62.28	42.86	55.42	34.22	49.68	27.72
	w/ VAMA	**65.54**	**46.16**	**59.08**	**38.66**	**53.87**	**32.88**
	FOSTER	72.44	56.98	66.51	54.52	61.93	47.86
	w/ VAMA	**73.64**	**59.96**	**69.71**	**58.68**	**62.98**	**48.90**
2000	ER	63.90	47.92	59.21	41.01	55.39	35.98
	w/ VAMA	**64.86**	**49.28**	**62.43**	**45.64**	**58.22**	**38.96**
	iCaRL	68.32	51.82	62.79	43.56	59.95	39.10
	w/ VAMA	**70.81**	**55.38**	**65.27**	**46.56**	**63.83**	**42.88**
	FOSTER	75.27	68.88	69.22	60.78	65.31	52.42
	w/ VAMA	**76.63**	**69.40**	**72.15**	**62.14**	**67.53**	**55.46**

4.2 Main Results

Table 1 and Table 2 report the main results on CIFAR-100 and ImageNet-100, respectively. We integrate VAMA into three representative replay-based methods, ER, iCaRL, and FOSTER, and evaluate them under multiple memory budgets and task protocols. Across both datasets, VAMA consistently improves the corresponding base methods, demonstrating that value-aware class-wise budgeting is complementary to different exemplar selection and replay pipelines. The improvements are observed under all memory budgets, from the more constrained 500 setting to the larger 2000 setting, indicating that the benefit of adaptive allocation is not limited to a particular memory regime.

Notably, the gains are often more pronounced in Last accuracy than in Avg accuracy. This suggests that VAMA is effective at improving the final retained performance after the full incremental sequence, which is consistent with its goal of allocating limited replay capacity to classes with higher estimated utility. These results suggest that VAMA provides a conservative yet effective deviation from uniform class-wise replay: under the same replay budget and the same within-class exemplar selection rule, value-aware class-wise budgeting can improve final retained performance. We further provide a multi-seed stability analysis in Appendix B.3.

4.3 Ablation and Analysis

Disentangling selection and allocation. Table 3 disentangles exemplar selection and class-wise memory allocation. VAMA consistently improves over Equal allocation under Herding, Value-based, and Random selection, showing that its benefit is not tied to a particular exemplar selection method. Rand-Q samples feasible non-uniform class quotas under the same memory budget. It does not consistently improve over Equal allocation, whereas VAMA improves across all selection strategies. This suggests that the gain comes from the proposed value-aware allocation signal rather than non-uniformity alone.

Table 3: Disentangling exemplar selection and memory allocation on CIFAR-100 with ER, $M = 2k$, and 10 tasks.

Alloc.	Herding		Value		Random	
	Avg	Last	Avg	Last	Avg	Last
Equal	60.0	43.1	57.8	36.1	59.3	41.1
$\mathrm{RMM}^\star$	61.5	44.5	57.6	35.8	60.6	42.9
Rand-Q	60.6	43.4	57.0	35.1	58.9	40.8
Loss-Q	60.6	42.9	57.4	35.3	59.0	39.0
VAMA	**63.5**	**47.2**	**58.3**	**36.7**	**61.5**	**44.4**

We further compare with two diagnostic allocation baselines. $\mathrm{RMM}^\star$ denotes the level-2 RMM [26] allocation rule adapted to the same ER pipeline, and Loss-Q replaces VAMA's class-value signal with task-local class-wise training loss under the same quota mapping. VAMA outperforms both in all cases, indicating that its advantage is not explained by the adapted RMM rule or by simply assigning more memory to high-loss classes. The smaller gain under Value-based exemplar selection suggests partial overlap between value-aware allocation and value-based selection, as both exploit utility-related information at different granularities. Notably, Random selection with VAMA even surpasses Herding with Equal allocation, highlighting class-wise allocation as a complementary design axis beyond instance-level selection.

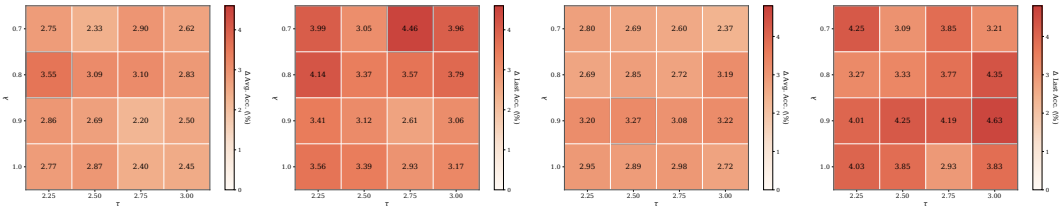
One-step intervention study. We further test whether the class-value signal is aligned with short-term replay benefit through a controlled one-step intervention on the $t = 1 \to 2$ transition. After learning task $t = 1$, we keep the total memory budget and exemplar selection rule fixed, and only change the direction of class-wise memory transfer. *Favor High* transfers memory slots from low-value classes to high-value classes, while *Favor Low* applies the reverse transfer. We then train task $t + 1$ and evaluate retention on task t.

Table 4: One-step retention for the $t = 1 \to 2$ transition.

Strategy	Ret. (%) ↑	Δ ↑
Favor Low	$57.21_{\pm 1.32}$	-0.51
Uniform	$57.72_{\pm 0.63}$	0.00
Favor High	$58.78_{\pm 1.21}$	$+1.06$

As shown in Table 4, *Favor High* improves task-t performance retention after learning task $t + 1$ by 1.06 on average relative to uniform allocation, whereas *Favor Low* decreases retention by 0.51. Moreover, *Favor High* outperforms *Favor Low* consistently across all five random seeds, with an average paired improvement of 1.57. Although small in scale, this directional pattern provides supportive evidence that the estimated class values are aligned with short-term replay benefit. A more fine-grained matched intervention is provided in Appendix B.1.

Effect of allocation regularization. We further analyze the two parameters that control the regularized allocation policy. The temperature τ determines the sharpness of the KL-regularized value-aware proposal, while λ controls the shrinkage strength toward the class-balanced prior. As shown in Figure 2, VAMA yields positive gains over the baseline across all tested combinations on both CIFAR-100 and ImageNet-100. Notably, the positive gains at $\lambda = 1$, where the explicit shrinkage step is removed, indicate that the value-aware proposal itself provides useful allocation information, while the stable performance across λ values suggests that shrinkage mainly controls the conservativeness of this deviation. The absence of sharp degradation indicates that the proposed allocation rule does



(a) CIFAR-100 (b) ImageNet-100

Figure 2: Effect of allocation regularization on CIFAR-100 and ImageNet-100 under the ER baseline with $M = 2000$ in the 10-task setting.

not depend on a narrow hyperparameter choice. Instead, a broad range of smooth and conservative deviations from uniform allocation leads to consistent improvements.

Table 5: Sensitivity to the early stopping epoch over five random seeds.

Epoch	5	10	20	40	60	Full
Avg	$63.08_{\pm 0.27}$	$63.09_{\pm 0.29}$	$63.31_{\pm 0.25}$	$63.27_{\pm 0.27}$	$63.24_{\pm 0.20}$	$63.31_{\pm 0.15}$
Last	$46.36_{\pm 0.71}$	$46.34_{\pm 0.50}$	$46.46_{\pm 0.65}$	$46.63_{\pm 0.38}$	$46.59_{\pm 0.24}$	$46.43_{\pm 0.11}$

Early convergence of class value. We analyze whether the class-value signal used by VAMA needs to be estimated from late-stage training. Figure 3 shows the Spearman correlation between the class-value ranking during training and the final ranking, averaged over multiple random seeds. The correlation is already high after the fifth epoch, reaching approximately $\rho = 0.838$, and then continues to increase gradually before stabilizing. This suggests that the relative ordering of class values is established very early in training.

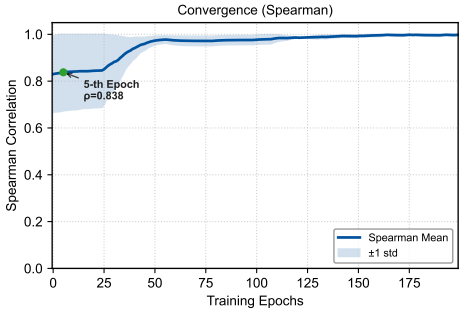


Figure 3: Convergence of class-value rank. Spearman correlation is measured against the final ranking, with shading denoting ± 1 standard deviation across seeds.

We further evaluate the effect of extracting class values at different training stages. As shown in Table 5, final performance remains stable across a wide range of extraction points, with only minor variations in both Avg and Last accuracy. Even values extracted at an early stage yield competitive performance, with only slightly larger variance in Last accuracy. These results show that VAMA does not require fully converged value estimates for effective allocation. Together with the use of a first-order in-run estimator, this early stabilization suggests that useful allocation signals can be obtained efficiently.

Combined with recent rehearsal-augmented methods. We further plug VAMA into MRFA [50] and STAR [51], which improve how stored exemplars are used during rehearsal rather than how many slots each class receives. As shown in Table 6, VAMA remains complementary under the 10-task protocol with $M = 2000$: it raises STAR Last accuracy by 2.30 on CIFAR-100 and 3.40 on ImageNet-100, and MRFA Avg/Last on CIFAR-100 from $65.66/47.94$ to $67.02/49.15$. The larger Last gains match VAMA's role in reallocating scarce retention capacity, indicating that buffer composition and rehearsal usage are not redundant.

Table 6: Integration with recent rehearsal-augmented methods under the 10-task setting.

Method	CIFAR-100		ImageNet-100	
	Avg	Last	Avg	Last
STAR	62.73	46.33	60.81	42.30
w/ VAMA	63.74	48.63	62.56	45.70
MRFA	65.66	47.94	65.31	49.52
w/ VAMA	67.02	49.15	66.21	51.14

Class-imbalanced setting. We additionally evaluate long-tailed CIFAR-100 (imbalance factor 20) under the 10-task protocol with ER, ResNet-18, herding, and $M = 500$. As shown in Table 7, VAMA

Table 7: Long-tailed CIFAR-100 (IF $= 20$, 10-task, $M = 500$).

Method	Avg	Last	Forget↓	Head	Medium	Tail
ER	34.30	20.34	49.01	18.59	24.94	17.55
w/ VAMA	**34.75**	**22.46**	**46.88**	**20.41**	**27.79**	**19.24**

improves Last accuracy by 2.12 points and reduces average forgetting by 2.13. Ranking classes by training frequency into Head/Medium/Tail groups ($34/33/33$) shows gains in *all* three groups, so the allocation does not improve the head by sacrificing the tail.

Holdout efficiency and fairness. On CIFAR-100 (10 tasks, $M = 2000$), we compare VAMA with full-data baselines using all 500 training images per class and matched baselines using 468 images per class for optimization, matching VAMA's training-data budget. As shown in Tables 8 and 9, VAMA improves Last accuracy over the matched baselines by 3.34% and 4.46% for ER and iCaRL, respectively. These results indicate that, in this setting, the gains cannot be explained solely by reducing the optimization data.

Table 8: Hold-out control for ER on CIFAR-100 (10 tasks, $M = 2000$).

Method	Samples per class	Avg ↑	Last ↑
Full	500	60.03	43.08
Matched	468	60.42	43.88
+ VAMA	468	**63.59**	**47.22**

Table 9: Hold-out control for iCaRL on CIFAR-100 (10 tasks, $M = 2000$).

Method	Samples per class	Avg ↑	Last ↑
Full	500	60.10	43.30
Matched	468	59.37	42.44
+ VAMA	468	**63.24**	**46.90**

5 Conclusion and Limitations

In this work, we studied class-wise memory allocation as an underexplored design axis in replay-based class-incremental learning. We proposed VAMA, a value-aware memory allocation framework that keeps the replay budget balanced across tasks while adaptively distributing memory within each task according to estimated class-level replay values. Experiments on CIFAR-100 and ImageNet-100 show that VAMA consistently improves multiple replay-based methods, including ER, iCaRL, and FOSTER, across different memory budgets and task protocols. Further analyses suggest that the gains are not merely due to exemplar selection, and that the allocation behavior is stable and robust to reasonable design choices.

Despite these results, VAMA is limited by its task-local value estimates and additional value-estimation overhead. Accordingly, we keep cross-task budgeting balanced and rely on early value extraction to reduce cost, which makes VAMA simple and compatible with most replay pipelines. Future work may jointly optimize class-wise allocation, exemplar selection, and replay scheduling.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62606102; in part by the Guangdong Natural Science Foundation under Grant 2024A1515010114; in part by JSPS Bilateral Program under Grant JPJSBP120257420; in part by JSPS KAKENHI under Grant JP24K20849 and JP23K28109.

References

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

[2] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

[3] 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, 2024.

[4] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[5] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

[6] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[7] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

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

[9] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[10] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[11] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

[12] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *European conference on computer vision*, pages 398–414. Springer, 2022.

[13] Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5497–5512, 2022.

[14] Yuhang Li, Guoxu Zhou, Zhenhao Huang, Xinqi Chen, Yuning Qiu, and Qibin Zhao. Tensor decomposition based memory-efficient incremental learning. In *Forty-second International Conference on Machine Learning*, 2025.

[15] Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th annual international conference on machine learning*, pages 1121–1128, 2009.

[16] Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in neural information processing systems*, 33:14879–14890, 2020.

[17] Rishabh Tiwari, Krishnateja Killamsetty, Rishabh Iyer, and Pradeep Shenoy. Gcr: Gradient coreset based replay buffer selection for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 99–108, 2022.

[18] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

[19] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32, 2019.

[20] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9630–9638, 2021.

[21] Jiahua Dong, Wenqi Liang, Yang Cong, and Gan Sun. Heterogeneous forgetting compensation for class-incremental learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11742–11751, 2023.

[22] Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost Van De Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems*, 36:6582–6595, 2023.

[23] Chris Dongjoo Kim, Jinseo Jeong, Sangwoo Moon, and Gunhee Kim. Continual learning on noisy data streams via self-purified replay. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 537–547, 2021.

[24] Sen Lin, Peizhong Ju, Yingbin Liang, and Ness Shroff. Theory on forgetting and generalization of continual learning. In *International Conference on Machine Learning*, pages 21078–21100. PMLR, 2023.

[25] Naoki Hiratani. Disentangling and mitigating the impact of task similarity for continual learning. *Advances in Neural Information Processing Systems*, 37:3243–3274, 2024.

[26] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *Advances in neural information processing systems*, 34:3478–3490, 2021.

[27] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.

[28] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd international conference on artificial intelligence and statistics*, pages 1167–1176. PMLR, 2019.

[29] Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. *arXiv preprint arXiv:2406.11011*, 2024.

[30] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[31] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. Pmlr, 2017.

[32] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[33] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018.

[34] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[35] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3014–3023, 2021.

[36] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. *arXiv preprint arXiv:2205.13218*, 2022.

[37] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375, 2017.

[38] Liyuan Wang, Xingxing Zhang, Qian Li, Mingtian Zhang, Hang Su, Jun Zhu, and Yi Zhong. Incorporating neuro-inspired adaptability for continual learning in artificial intelligence. *Nature Machine Intelligence*, 5(12):1356–1368, 2023.

[39] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018.

[40] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 831–839, 2019.

[41] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 374–382, 2019.

[42] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9046–9056, 2022.

[43] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

[44] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209. IEEE, 2020.

[45] Lloyd S Shapley et al. A value for n-person games. 1953.

[46] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[47] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[48] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: a python toolbox for class-incremental learning, 2023.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[50] Bowen Zheng, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Multi-layer rehearsal feature augmentation for class-incremental learning. In *Forty-first international conference on machine learning*, 2024.

[51] Masih Eskandar, Tooba Imtiaz, Davin Hill, Zifeng Wang, and Jennifer Dy. Star: Stability-inducing weight perturbation for continual learning. In *International Conference on Learning Representations*, volume 2025, pages 49102–49117, 2025.

[52] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[53] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 9275–9285. IEEE, 2022.

[54] Zhuang Qi, Ying-Peng Tang, Lei Meng, Han Yu, Xiaoxiao Li, and Xiangxu Meng. Class-wise balancing data replay for federated class-incremental learning. In *Advances in Neural Information Processing Systems*, volume 38, pages 145309–145334, 2025.

[55] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. *arXiv preprint arXiv:2106.01085*, 2021.

[56] James Seale Smith, Lazar Valkov, Shaunak Halbe, Vyshnavi Gutta, Rogerio Feris, Zsolt Kira, and Leonid Karlinsky. Adaptive memory replay for continual learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3605–3615. IEEE, 2024.

A Implementation Details

Our implementation is based on PyTorch [52] and the class-incremental learning toolbox PyCIL [48]. We implement VAMA as a plug-in class-wise memory allocation module on top of ER, iCaRL, and FOSTER, while keeping the replay pipeline, optimization setting, and within-class exemplar selection strategy the same as in the corresponding PyCIL baseline unless otherwise specified. For both CIFAR-100 and ImageNet-100, we use ResNet-18 [49] as the backbone.

Following the main setting in the paper, for each task we reserve 5-8% of the current-task training data as a task-local hold-out set for value estimation, and use the remaining data for standard replay-based training. Note that baselines are trained on the full training set. Class values are computed using

Table 10: Matched class-targeted intervention on task-t classes ($t = 2 \rightarrow 3$). We report mean and standard deviation over 12 paired runs. The total memory budget is fixed, and only the direction of class-wise memory transfer is changed. *High Ret.* and *Low Ret.* denote the mean retention over the selected top-2 and bottom-2 task-t classes, respectively; *Task-t Ret.* denotes retention averaged over all classes in task t.

Strategy	High Ret. (%)	Low Ret. (%)	Task-t Ret. (%)
Favor Low	$47.75_{\pm 2.77}$	$\mathbf{89.67_{\pm 3.61}}$	$73.11_{\pm 0.91}$
Uniform	$51.88_{\pm 3.52}$	$89.17_{\pm 3.79}$	$73.55_{\pm 1.11}$
Favor High	$\mathbf{53.63_{\pm 4.42}}$	$88.29_{\pm 4.21}$	$73.95_{\pm 0.87}$

the first-order in-run approximation described in Sec. 3.3 and are aggregated within each class by positive-mean. Unless otherwise specified, we set $\tau = 2.25$ and $\lambda = 0.8$ on CIFAR-100, and $\tau = 3.0$ and $\lambda = 0.9$ on ImageNet-100, and the main results in Table 1 and Table 2 use value extraction at last epoch.

For baselines with additional auxiliary objectives, such as iCaRL and FOSTER, the validation gradient used for class-value estimation is computed only from the cross-entropy classification loss on the task-local hold-out set, while the full original training objective is kept unchanged for model optimization. This gives a unified valuation objective across different replay baselines and avoids entangling class-value estimation with method-specific auxiliary losses.

For FOSTER [12], we extract validation gradients only in the stages that directly optimize the main network with labeled current-task data, namely the initial stage and the feature-boosting stage, and do not use the later feature-compression stage for valuation. This is because the compression stage mainly serves teacher–student feature compression rather than the task-local supervised classification objective used by VAMA for value estimation.

Unless otherwise specified, all other optimization hyperparameters follow the PyCIL implementations of the corresponding baselines. Our experiments were conducted on NVIDIA RTX A6000 GPUs with 48GB memory. Since the reported study covers several baselines, datasets, memory budgets, and task protocols, exact runtimes differ substantially across runs; we therefore provide representative results. For example, ER + CIFAR-100 runs typically require approximately 1-2 GPU-hours depending on the task protocol, while ER + ImageNet-100 runs require approximately 5-8 GPU-hours.

B More Experimental Results

B.1 Matched Class-Targeted Intervention

As a complementary experiment to the one-step intervention in the main text, we further examine whether the estimated class values provide a meaningful direction for memory reallocation at the class-group level. We conduct the intervention on the $t = 2 \rightarrow 3$ transition, where task t has a task-level budget of 1000 under uniform allocation. After finishing task $t = 2$, we estimate class values within task t and select the top-2 and bottom-2 classes as high-value and low-value classes, respectively. We keep the task-t budget, exemplar selection rule, training recipe, class order, and random seed fixed within each paired run, and only change the direction of class-wise memory transfer among task-t classes. Specifically, *Favor High* reallocates slots from low-value classes to high-value classes, while *Favor Low* applies the reverse transfer; *Uniform* serves as the matched control.

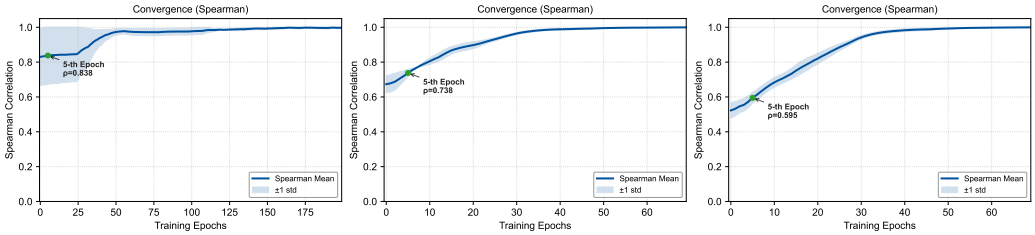
High- and low-value groups have different baseline retention levels under uniform allocation, so their absolute retention values are not directly comparable. We therefore focus on within-group changes across matched intervention arms. For the high-value group, *Favor High* improves retention over *Favor Low* by $+5.88$ points on average, and this targeted gain is positive in all 12 paired runs. For the low-value group, the reverse intervention is weaker: *Favor Low* improves retention over *Favor High* by $+1.38$ points on average, with positive gains in 8 of 12 paired runs. At the task level, *Favor High* also improves retention over *Favor Low* by $+0.84$ points on average. These results suggest that, in this matched setting, the high-value classes identified by VAMA tend to benefit more from additional replay slots than the low-value classes. This diagnostic experiment does not claim that class value

Table 11: Ablation of class-level aggregation rules on CIFAR-100 under ER with $M = 2000$ in the 10-task setting. All variants use the same sample-level value estimator and allocation rule, with $\tau = 1$ and $\lambda = 1$. Only the aggregation from sample-level values to class-level values is changed.

Metric	Signed sum	Positive sum	Signed mean	Positive mean
Avg	62.45	62.34	62.44	62.34
Last	46.34	45.10	45.14	45.58

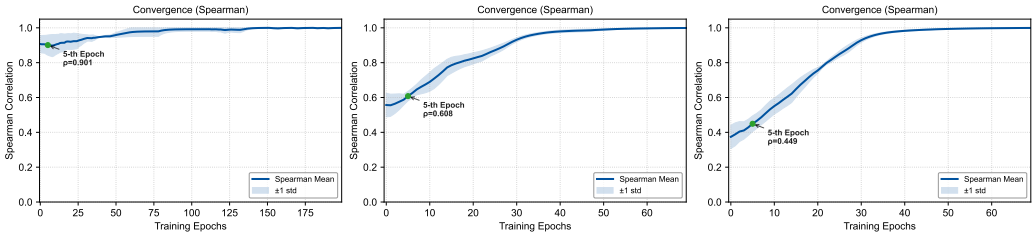
alone determines future replay utility; rather, it provides controlled evidence that the estimated signal contains allocation-relevant information for class-wise memory transfer.

B.2 Ablation of Aggregation Rules



Task 1 Task 5 Task 9

(a) Positive-mean aggregation.



Task 1 Task 5 Task 9

(b) Signed-mean aggregation.

Figure 4: Convergence of class-value rankings under different class-level aggregation rules. Spearman correlation is measured against the final class-value ranking for Task 1, Task 5, and Task 9. The top row uses positive-mean aggregation and the bottom row uses signed-mean aggregation. Shading denotes ±1 standard deviation across seeds.

We further study how the class-level aggregation rule affects VAMA. Recall that the proposed method first computes sample-level value scores $\{s_i\}$ and then aggregates them into a class-wise replay value signal $\{V_c^t\}$. In principle, this aggregation can be defined in different ways depending on whether signed values are preserved or only their positive part is retained, and whether class-wise values are computed by averaging or summation. To examine the sensitivity of VAMA to this design choice, we compare four variants:

$$\text{Signed Sum:} \quad V_c^t = \sum_{i:\, y_i = c} s_i, \qquad \text{Positive Sum:} \quad V_c^t = \sum_{i:\, y_i = c} \max(s_i, 0),$$

$$\text{Signed Mean:} \quad V_c^t = \frac{1}{n_c^t} \sum_{i:\, y_i = c} s_i, \qquad \text{Positive Mean:} \quad V_c^t = \frac{1}{n_c^t} \sum_{i:\, y_i = c} \max(s_i, 0).$$

Table 11 reports the results on CIFAR-100 under the ER baseline with a memory budget of 2000 in the 10-task setting, using $\tau = \lambda = 1$ for controlled comparison. Overall, all four aggregation rules yield similar performance, with differences confined to a small range. This suggests that VAMA is not particularly sensitive to the exact form of class-level aggregation, and that its gains are not driven by a delicate aggregation choice. Considering that the training data volume for each class may

vary across different datasets, we first employ mean aggregation. In addition, as shown in Figure 4, positive-mean aggregation tends to reach stability earlier in the training process. Therefore, we choose the positive-mean aggregation method as our preferred option.

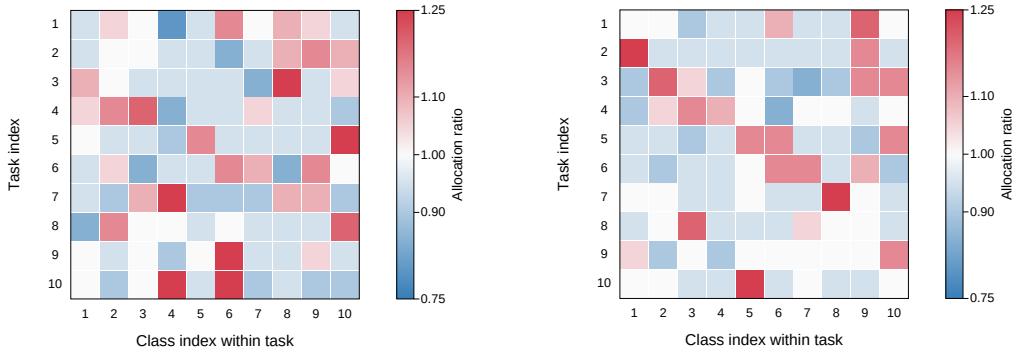
B.3 Stability across Seeds

Table 12: Stability analysis on CIFAR-100 under the 10-task setting with a memory budget of 2000. Results are reported as mean ± standard deviation over five random seeds.

Base method	Baseline Avg	+VAMA Avg	Baseline Last	+VAMA Last
ER	60.15 ±0.23	63.40 ±0.15	43.46 ±0.40	46.68 ±0.36
iCaRL	60.47 ±0.36	63.32 ±0.17	43.05 ±0.21	46.96 ±0.22
FOSTER	68.57 ±0.18	70.35 ±0.12	56.61 ±0.05	58.63 ±0.15

Following the default PyCIL protocol, the main results are reported under a fixed class order and random seed. To further examine stability, we run an additional multi-seed analysis on CIFAR-100 under the 10-task setting with $M = 2000$. As shown in Table 12, VAMA consistently improves ER, iCaRL, and FOSTER in both Avg and Last accuracy across seeds, indicating the robustness of the proposed method.

B.4 Allocation Visualization



(a) CIFAR-100 (b) ImageNet-100

Figure 5: Final class-wise allocation ratios produced by VAMA on CIFAR-100 and ImageNet-100 under the ER baseline with a memory budget of 2000 in the 10-task setting. Each cell reports the ratio between the allocated quota and the uniform quota within the corresponding task. Values above one indicate over-allocation relative to uniform replay, while values below one indicate under-allocation.

Figure 5 visualizes the final allocation ratios produced by VAMA on CIFAR-100 and ImageNet-100. The allocation patterns are clearly non-uniform, indicating that VAMA adapts the memory budget according to the estimated class-level replay value rather than applying a fixed uniform rule. Meanwhile, the allocation remains smooth and non-collapsed: the ratios range from 0.80 to 1.25 on CIFAR-100 and from 0.85 to 1.25 on ImageNet-100. This suggests that VAMA introduces controlled deviations from uniform allocation while avoiding overly concentrated memory assignment.

B.5 Additional Dataset and Transformer-Based Learner

We further evaluate VAMA on Food-101 and with the Transformer-based continual learner DyTox [53]. For Food-101, we use ER with ResNet-18 and herding under a 10-task protocol. We reuse the ImageNet-100 configuration without dataset-specific tuning. For DyTox, we use CIFAR-100 with 10 tasks and train the model from scratch without external pretraining. Similarly, we reuse the CIFAR-100 configuration without method-specific tuning. Both experiments use a fixed total memory budget of $M = 2000$. VAMA changes only the class-wise memory quotas while preserving the host learner's training procedure and within-class exemplar selection rule.

Table 13: Additional evaluations on Food-101 and with DyTox. Both settings use 10 tasks and a fixed memory budget of $M = 2000$.

Settings	Baseline Avg	+VAMA Avg	Baseline Last	+VAMA Last
ER, Food-101	65.31	66.51	46.71	48.68
DyTox, CIFAR-100	73.56	73.94	60.62	61.66

As shown in Table 13, VAMA consistently improves their performance. These results provide additional evidence that class-wise memory allocation is complementary to replay across datasets and architectures.

C Additional Discussion of Related Work

Several recent works incorporate class awareness or sample utility into replay, but under different settings and decision variables from VAMA. FedCBDR [54] coordinates class-wise balancing replay to correct class imbalance across clients in federated CIL, whereas VAMA assigns non-uniform quotas within each task in centralized, storage-bounded CIL, even when the incoming stream is class-balanced. OCS [55] selects *which* samples to retain via online coreset selection, while VAMA decides *how many* slots each class receives and leaves exemplar identity to the host selector; the two are thus complementary. AMR [56] assumes full access to past data and allocates replay computation, whereas VAMA assumes a strictly bounded exemplar buffer and allocates storage. Given these differences, we position VAMA as a selector-agnostic, task-local class-wise quota allocation layer for bounded exemplar replay, rather than as the first class-aware or adaptive replay-management method.

D Details about Shapley

D.1 Data Shapley

We briefly review the Data Shapley [45, 27] perspective that motivates our class-value estimator. Given a training set $D = \{z_i\}_{i=1}^N$ and a utility function $U(\cdot)$ defined on subsets of D, the Shapley value of a sample $z \in D$ is

$$\phi_z(U) = \frac{1}{N} \sum_{k=1}^{N} \binom{N-1}{k-1}^{-1} \sum_{\substack{S \subseteq D \setminus \{z\} \\ |S|=k-1}} \big[U(S \cup \{z\}) - U(S)\big]. \tag{14}$$

That is, $\phi_z(U)$ averages the marginal utility of z over all possible subset contexts. In standard Data Shapley, the utility is typically defined as the validation performance of a model trained on subset S, e.g.,

$$U(S) = \mathrm{Perf}(\mathcal{A}(S)), \tag{15}$$

where $\mathcal{A}$ is a learning algorithm and $\mathrm{Perf}(\cdot)$ measures performance on a hold-out validation set. The appeal of the Shapley value comes from the fact that it is the unique value notion satisfying the standard fairness axioms of null player, symmetry, efficiency, and linearity [45]. In our context, linearity is particularly useful because it allows contribution over an entire training run to be decomposed into a sum of per-step contributions.

However, directly applying retraining-based Data Shapley is not suitable for our setting. Exact computation requires evaluating many subsets, which is computationally prohibitive. Moreover, retraining-based Data Shapley measures contribution with respect to a learning algorithm averaged over many retrainings, whereas VAMA needs a task-local utility signal tied to the actual training trajectory used for memory allocation.

D.2 In-Run Data Shapley and First-Order Form

To avoid repeated retraining, In-Run Data Shapley [29] measures contribution within a single training run. Consider an iterative optimization process with parameters $\{w_0, w_1, \ldots, w_T\}$. At iteration t,

the model is updated using mini-batch B_t as

$$w_{t+1} = w_t - \eta_t \sum_{z \in B_t} \nabla \ell(w_t; z), \tag{16}$$

where η_t is the learning rate. We omit mini-batch normalization constants for notational simplicity, since VAMA only uses relative values.

For a validation example $z^{(\mathrm{val})}$, we define the local utility at step t as the reduction in validation loss induced by an update using subset $S \subseteq B_t$:

$$U^{(t)}(S; z^{(\mathrm{val})}) = \ell(w_t, z^{(\mathrm{val})}) - \ell(\tilde{w}_{t+1}(S), z^{(\mathrm{val})}), \tag{17}$$

where

$$\tilde{w}_{t+1}(S) = w_t - \eta_t \sum_{z \in S} \nabla \ell(w_t; z). \tag{18}$$

By the linearity of the Shapley value, the contribution over the full training run can be decomposed into local contributions along the optimization trajectory:

$$\phi_z(U) = \sum_{t=0}^{T-1} \phi_z\Big(U^{(t)}\Big). \tag{19}$$

To make this practical, In-Run Data Shapley approximates the one-step utility using a Taylor expansion. Under the first-order approximation,

$$U^{(t)}(S; z^{(\mathrm{val})}) \approx -\nabla \ell(w_t, z^{(\mathrm{val})})^\top \big(\tilde{w}_{t+1}(S) - w_t\big). \tag{20}$$

Substituting Eq. 18 into Eq. 20 gives

$$U^{(t)}(S; z^{(\mathrm{val})}) \approx \eta_t \sum_{z \in S} \nabla \ell(w_t, z^{(\mathrm{val})})^\top \nabla \ell(w_t; z). \tag{21}$$

Because the approximated utility is additive over samples in S, its Shapley value has a closed form:

$$\phi_z\Big(U^{(t)}\Big) \approx \eta_t \, \nabla \ell(w_t, z^{(\mathrm{val})})^\top \nabla \ell(w_t; z). \tag{22}$$

Averaging over the task-local validation subset V and summing over the iterations in which z is used yields the first-order in-run contribution:

$$\phi_z(U) \approx \sum_{t: z \in B_t} \eta_t \, \nabla L_{\mathrm{val}}(w_t; V)^\top \nabla \ell(w_t; z). \tag{23}$$

Therefore, in the first-order in-run view, a sample is valuable if its training gradient is aligned with the validation gradient, since such an update is expected to reduce the validation loss. This first-order form avoids repeated retraining and provides an efficient task-local approximation for in-run data valuation.

E Broader Impacts

This work proposes a memory allocation module for replay-based continual learning. Its potential positive impact is improved memory efficiency, which may reduce the need for storing large replay buffers or repeatedly retraining models. However, replay-based methods retain examples from past data, which can raise privacy concerns when applied to sensitive or user-generated data. Moreover, because VAMA allocates memory non-uniformly across classes, it may interact with class imbalance or class-dependent errors and should be accompanied by per-class auditing in deployment. Our experiments are limited to standard image classification benchmarks, and the method should not be interpreted as validated for high-stakes applications without additional privacy, fairness, and robustness evaluation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We ensure that the abstract and introduction clearly state the main contributions, which are consistently supported by the empirical evaluations presented throughout the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the proposed approach are explicitly discussed in the last section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: Our paper does not include theoretical results. Therefore, this item is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Detailed descriptions of datasets, training protocols, and evaluation metrics are provided in Sec. 4.1 and appendix A, enabling faithful reproduction of the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code is not released with this paper. All datasets used are publicly available benchmarks, and Sec. 4.1 and Appendix A provide the implementation details needed to reproduce the main results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: This paper provides a detailed description of the dataset, data splitting, training process, hyperparameters, and evaluation methods in both the main text and the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Following [48, 3], we report the average accuracy and last accuracy in the main results. Additional statistical information is provided in the appendix

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix A specifies the types of GPU used in the experiments. Because the reported study covers several baselines, datasets, memory budgets, and task protocols, exact runtimes differ substantially across runs; we therefore provide representative results in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses existing public image classification benchmarks and does not involve human subjects, crowdsourcing, private data, or high-risk model releases. To the best of our knowledge, the work conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: We have discussed broader impact in Appendix E.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not introduce models or datasets with a high risk of misuse, so safeguards are not applicable.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have properly credited the assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release new datasets, models, or other research assets. The experiments are conducted on existing public benchmarks, including CIFAR-100 and ImageNet-100, and build on existing continual learning implementations.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects. All experiments are conducted on existing image classification benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, crowdsourcing, user studies, or collection of data from human participants. Therefore, IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as an important, original, or non-standard component of the core method, experiments, data generation, or analysis. The proposed method does not rely on LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.