Less Forgetting, More OOD Generalization: Adaptive Augmented Reweighted Replay (AA-RR) for Continual Learning

Anonymous authors
Paper under double-blind review

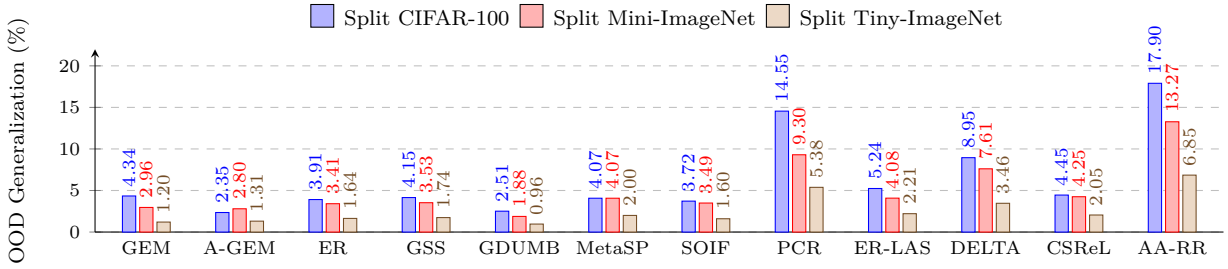

Figure 1: **Evaluating Out-of-Distribution (OOD) Generalization:** State-of-the-art rehearsal-based methods exhibit a significant performance drop on OOD samples across Split CIFAR-100, Split Mini-ImageNet, and Split Tiny-ImageNet, revealing limited generalization capability. The proposed AA-RR method effectively addresses this limitation, achieving consistent improvements across all benchmarks.

Abstract

Machine learning models often forget previously learned classes when trained sequentially. Rehearsal-based methods mitigate this by replaying stored samples, but their reliance on memorization leads to poor out-of-distribution (OOD) generalization—a problem that remains largely unstudied. This memorization is driven by unbalanced gradient updates, spurious correlations, and class-imbalanced replay buffers. To address these issues, we introduce Adaptive Augmented Reweighted Replay (AA-RR), a lightweight framework designed to improve generalization in rehearsal-based continual learning (CL). AA-RR applies adaptive, class-aware loss reweighting to correct gradient imbalance while accounting for data recency and limited buffer capacity. It further incorporates data-centric augmentation and a principled sample-selection strategy based on forgetting dynamics to retain representative, consistently learned examples. Experiments on standard CL benchmarks show that AA-RR markedly boosts generalization and surpasses state-of-the-art baselines, especially under covariate shift.[1]

1 Introduction

Continual Learning (CL) enables models to learn new tasks or classes sequentially without forgetting previously acquired knowledge. Approaches to CL include regularization-based methods, which constrain parameter updates to preserve prior knowledge Kirkpatrick et al. (2017); Chaudhry et al. (2018a); architecture-based methods, which expand or adapt network structures for new tasks Mallya & Lazebnik (2018); Hung et al. (2019); and rehearsal-based methods, which mitigate forgetting by replaying stored samples from earlier tasks during training Sun et al. (2022; 2023); Raghavan et al. (2024); Tong et al. (2025).

Among these approaches, rehearsal-based methods have attracted significant attention due to their simplicity and strong effectiveness. However, despite their strong performance, these methods remain underexplored in

[1]Code is available in the supplementary materials.

terms of robustness, especially regarding out-of-distribution (OOD) generalization. Some studies Verwimp et al. (2021); Zhang et al. (2022b) have raised concerns that such methods may rely heavily on memorizing replayed samples rather than truly learning and generalizing the underlying knowledge. Despite growing attention to generalization and overfitting in CL Bonicelli et al. (2022); Yu et al. (2022), the challenge of OOD generalization remains largely unexplored. Our investigations clearly demonstrate that current approaches fail to generalize to samples exhibiting covariate shifts (see Figure 1). Notably, by design, CL methods aim to retain knowledge from previously learned classes/tasks — in other words, to prevent forgetting. However, the critical point is that preventing forgetting should not come at the cost of memorization, which can severely degrade a model's ability to generalize, particularly to OOD data. In essence, while CL methods should preserve past knowledge, they must do so in a way that maintains the model's generalization capability. This overlooked aspect motivates our study, where we analyze the problem and propose an effective solution.

We identify three sources of *overfitting* in buffer-based CL. First, unbalanced gradient propagation due to sequential training, leading to dominance of recent tasks. Second, spurious correlations arise within individual training stages—patterns that fit the current domain but fail to generalize. Third, buffer distribution, depending on which samples are retained and whether class and task balance are maintained.

Buffer-based CL inherently suffers from class imbalance, as earlier task examples are increasingly underrepresented due to limited buffer capacity. This skews optimization toward recent classes, exacerbating forgetting. A common strategy to address this issue is to reweight the softmax cross-entropy loss to counteract unbalanced gradient contributions. Recent works Lin et al. (2023); Huang et al. (2023); Raghavan et al. (2024) adopt this approach by reweighting the loss based on class frequency, thereby mitigating the skewed gradient propagation.

However, existing reweighting strategies Lin et al. (2023); Huang et al. (2023) have critical limitations. They rely on batch-wise class frequency, providing no gradient signal for absent classes—a side effect of inheriting long-tail learning techniques designed for stationary settings. Moreover, they Lin et al. (2023); Huang et al. (2023); Raghavan et al. (2024) do not consider the temporal ordering (age or recency) of buffered samples, despite its importance for balancing stability and plasticity in CL.

Sample selection policies in rehearsal methods also play a central role in preventing forgetting. Most approaches use random sampling to update the memory buffer Lopez-Paz & Ranzato (2017); Chaudhry et al. (2018b; 2019); Prabhu et al. (2020); Lin et al. (2023); Huang et al. (2023); Raghavan et al. (2024). However, without principled selection, such strategies are prone to storing outliers or non-representative examples, which impairs generalization. Random sampling also tends to produce class-imbalanced buffers (Section 5.2, Table 2), leading to long-tailed distributions that distort decision boundaries Cui et al. (2019); Samuel & Chechik (2021); Shi et al. (2023).

To overcome these issues, more principled selection strategies have been proposed, such as those based on gradient informativeness Aljundi et al. (2019); Jin et al. (2021); Yoon et al. (2021); Tiwari et al. (2022), Shapley values Shim et al. (2021), and influence functions Sun et al. (2022; 2023), which aim to prioritize informative samples during buffer updates. Nevertheless, these methods still struggle with class and task imbalance (Section 5.2, Table 2) and often incur significant computational costs due to gradient tracking or auxiliary optimization. Even the more recent gradient-free coreset methods Tong et al. (2025) reduce some overhead but still require auxiliary model training for each task (Section 5.2, Table 3).

To overcome the above limitations, we propose **A**daptive **A**ugmented **R**eweighted **R**eplay (**AA-RR**)—a lightweight yet effective framework that enhances generalization in rehearsal-based CL.

AA-RR introduces a class-aware cross-entropy softmax reweighting mechanism that adaptively adjusts loss contributions to counter class imbalance caused by non-stationary data, recency effects, and constrained buffer size. This reduces gradient imbalance across classes and promotes stable learning over time.

In addition, we extend insights from data-centric generalization strategies in stationary settings to the CL paradigm, aiming to mitigate overfitting caused by spurious correlations. Prior work in stationary training has shown that data-centric techniques (e.g., data augmentation Yao et al. (2022); Noohdani et al. (2024)) and representation learning Zhang et al. (2022a); Lv et al. (2022); Eastwood et al. (2023); Venkataramani et al. (2024) can effectively reduce the influence of spurious correlations Ye et al. (2024).

To improve memory quality, we propose a simple yet effective sample selection strategy inspired by Toneva et al. (2018), who analyzed the learning dynamics of neural networks and introduced the concept of forgetting events. We populate the buffer with consistently learned (i.e., frequently correctly classified) samples from early training epochs while maintaining class- and task-balanced distributions. This selection is performed directly on the main continual learner and incurs negligible overhead.

The main contributions of this work are as follows:

- To the best of our knowledge, this is the first study to show that the performance of rehearsal-based CL methods degrades significantly under distributional shift, where the i.i.d. assumption no longer holds.
- We identify key sources of *overfitting* in buffer-based CL and propose AA-RR, a lightweight framework that combines adaptive class-aware reweighting, targeted data augmentation, and representative sample selection.
- We demonstrate the effectiveness and efficiency of AA-RR through comprehensive experiments on standard CL benchmarks, outperforming state-of-the-art baselines.

2 Related Work

Rehearsal-based CL Methods. By continually revisiting past experiences drawn from a memory buffer, rehearsal-based methods Lopez-Paz & Ranzato (2017); Chaudhry et al. (2018b; 2019); Aljundi et al. (2019); Prabhu et al. (2020); Sun et al. (2022; 2023); Lin et al. (2023); Huang et al. (2023); Raghavan et al. (2024); Tong et al. (2025) reinforce earlier knowledge and improve long-term retention across tasks. They achieve this by storing previously seen samples and periodically updating the model with a mixture of old and new data, effectively mitigating catastrophic forgetting.

Non-Policy-Driven Replay Methods. GEM Lopez-Paz & Ranzato (2017) mitigates forgetting by constraining gradients to avoid increasing loss on past tasks. A-GEM Chaudhry et al. (2018b) simplifies GEM by projecting gradients only when conflicting with a reference gradient from the buffer. Both methods maintain an equal number of samples for each task. GDUMB Prabhu et al. (2020) adopts a simple strategy by storing samples in the buffer and discarding the model during training. A fresh model is trained from scratch on the buffer at test time. Despite its simplicity, GDUMB often outperforms more complex methods, highlighting the importance of buffer composition over continual updates. ER Chaudhry et al. (2019) is a simple rehearsal approach that maintains the memory buffer via reservoir sampling and retrieves samples uniformly at random. Despite its minimal design, it continues to serve as a strong baseline.

Policy-Driven Replay Methods. GSS Aljundi et al. (2019) formulates buffer population as a constraint-reduction problem, selecting samples to minimize gradient interference by maximizing diversity in gradient space. MetaSP Sun et al. (2022) examines the example-level influence on the stability–plasticity trade-off in CL. It computes how each sample affects both remembering past tasks (stability) and learning new ones (plasticity) using influence functions, then fuses these influences via a Pareto-optimal formulation to guide replay buffer selection. SOIF Sun et al. (2023) extends MetaSP by modeling second-order influence—capturing how one sample affects future selections—and introduces a regularizer to limit cumulative bias in the buffer. CSReL Tong et al. (2025) formulates the memory buffer population as a coreset-selection problem via a bilevel formulation. It introduces the concept of reducible loss (ReL) to quantify how much performance would improve if a given sample were added to the buffer, thereby selecting samples that yield the greatest model gain. The CSReL-CL-Prv variant performs per-task selection while reducing task interference using memory, and is referred to as CSReL throughout the paper.

Reweighting-based Replay Methods. PCR Lin et al. (2023) introduces a proxy-based contrastive replay strategy, replacing anchor-to-sample pairs with anchor-to-proxy pairs in the contrastive loss to alleviate class-imbalance bias. It employs reservoir sampling to update the buffer.

DELTA Raghavan et al. (2024) addresses long-tailed online CL by decoupling representation learning and classification. It uses contrastive learning for robust features and then fine-tunes the classifier with an equalization loss to balance gradients for underrepresented classes. This method also uses reservoir sampling.

ER-LAS Huang et al. (2023) addresses class imbalance by adjusting logits based on class priors. It demonstrates that Bayes-optimal classification can be approached through a simple logit-adjustment term, with minimal computational cost. Buffer updates are performed using reservoir sampling.

Forgetting in Stationary Learning Settings. Toneva et al. (2018) investigates the phenomenon of sample-level forgetting events during standard single-task training: an example is considered forgotten when it goes from correctly classified to incorrectly classified during training. They find that some examples are consistently forgotten while others are never forgotten, and that this distinction is stable across architectures.

3 Problem Statement and Analysis

3.1 Preliminaries

We consider CL in an offline, buffer-based setting, where a model learns from a sequence of T tasks $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T\}$. Each task $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$ consists of labeled samples from a disjoint class set $\mathcal{C}_t$, i.e., $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for all $i \neq j$. We denote by $\mathcal{C}_{1:t} = \bigcup_{i=1}^{t} \mathcal{C}_i$ the cumulative set of classes observed up to task t, and $\mathcal{C}_{<t} = \mathcal{C}_{1:t-1}$ the set of classes seen before task t.

The model comprises a feature encoder $f_\theta : \mathbb{R}^d \to \mathbb{R}^m$ and a linear classifier with class weights $\{w_c \in \mathbb{R}^m\}_{c \in \mathcal{C}_{1:t}}$. For an input x, the encoder produces an embedding $z = f_\theta(x)$, and the classifier computes logits $\ell_c = w_c^\top z$. The corresponding probabilities are obtained via a softmax:

$$p_c = \frac{e^{(\ell_c)}}{\sum_{j \in \mathcal{C}_{1:t}} e^{(\ell_j)}}. \tag{1}$$

During training on task t, the model has access to the full dataset $\mathcal{D}_t$ and a memory buffer $\mathcal{B}$ that stores exemplars from past tasks. The training objective minimizes the expected cross-entropy loss over both sources:

$$\mathcal{L} = \mathbb{E}_{(x,y) \sim \mathcal{D}_t \cup \mathcal{B}} \big[-\log p_y \big]. \tag{2}$$

In this offline setup, multiple passes over $\mathcal{D}_t$ are allowed, but previous data $\mathcal{D}_{<t}$ are only accessible via $\mathcal{B}$.

3.2 Unbalanced Gradient Propagation

Cross-entropy training introduces a gradient imbalance between new and old tasks, primarily due to the disproportionate number of samples from $\mathcal{D}_t$ compared to the buffer $\mathcal{B}$. For a sample (x, y) with embedding z, the gradient with respect to the classifier weights is:

$$\frac{\partial \mathcal{L}}{\partial w_c} = (p_c - \mathbf{1}[c = y])\, z. \tag{3}$$

Where, $\mathbf{1}[.]$ is the indicator function. Meanwhile, the gradient with respect to the feature embedding is:

$$\frac{\partial \mathcal{L}}{\partial z} = \sum_{j \in \mathcal{C}_{1:t}} p_j w_j - w_y. \tag{4}$$

These updates pull w_y and z toward each other while pushing them away from other class weights, improving discrimination for frequent classes.

Let $n_t = |\mathcal{D}_t|$ and $n_b = |\mathcal{B}|$. The expected gradient on a weight w_c can be decomposed as:

$$\mathbb{E}_{\mathcal{D}_t \cup \mathcal{B}} \left[\frac{\partial \mathcal{L}}{\partial w_c} \right] = \frac{n_t}{n_t + n_b} \mathbb{E}_{\mathcal{D}_t} \left[\frac{\partial \mathcal{L}}{\partial w_c} \right] + \frac{n_b}{n_t + n_b} \mathbb{E}_{\mathcal{B}} \left[\frac{\partial \mathcal{L}}{\partial w_c} \right]. \tag{5}$$

Since $n_t \gg n_b$, gradients from $\mathcal{D}_t$ dominate. Consequently, weights of new classes receive frequent, large updates of $w_c^{(t+1)} \approx w_c^{(t)} - \eta \sum_{(x,y=c)\in\mathcal{D}_t}(p_c - 1)z$, whereas old-class weights are updated sparsely through replay, i.e. $w_c^{(t+1)} \approx w_c^{(t)} - \eta \sum_{(x,y=c)\in\mathcal{B}}(p_c - 1)z$.

Gradient Polarity Across Tasks. The imbalance also extends to non-target classes. For any sample (x, y), the gradient with respect to the logit of class $c \neq y$ is $\partial\mathcal{L}/\partial\ell_c = p_c$, implying that each non-target weight receives a repulsive update $\partial\mathcal{L}/\partial w_c = p_c z$.

When $x \sim \mathcal{D}_t$ (new data), all old classes $c \in \mathcal{C}_{<t}$ appear only as non-targets, accumulating strong negative gradients. Conversely, when $x \sim \mathcal{B}$ (replayed data), new classes $c \in \mathcal{C}_t$ act as non-targets, but their logits are low and the induced gradients are weak, $\left\|\frac{\partial\mathcal{L}}{\partial w_{c_{\text{old}}}}\right\|_{x\sim\mathcal{D}_t} \gg \left\|\frac{\partial\mathcal{L}}{\partial w_{c_{\text{new}}}}\right\|_{x\sim\mathcal{B}}$. Hence, old-class weights are continually pushed away from evolving feature distributions, amplifying forgetting and representation drift, $\langle f_\theta(x_{\text{old}}), w_{c_{\text{old}}}\rangle \downarrow, \quad \langle f_\theta(x_{\text{old}}), w_{c_{\text{new}}}\rangle \uparrow$.

With a fixed buffer size $|\mathcal{B}|$, the number of exemplars per old class decreases as tasks accumulate, $|\mathcal{B}_c| \approx \frac{|\mathcal{B}|}{|\mathcal{C}_{1:t}|}, \quad c \in \mathcal{C}_{<t}$, This progressively weakens the gradient signal from past tasks, amplifying both classifier bias and feature drift.

3.3 Logit Reweighting via Class-Prior Adjustment

A common strategy to mitigate this unbalanced gradient propagation is to reweight the softmax cross-entropy loss based on class frequency. Recent methods such as PCR Lin et al. (2023), DELTA Raghavan et al. (2024), and ER-LAS Huang et al. (2023) adopt this approach by estimating class priors using recently observed classes and scaling each class's loss contribution proportional to its observed frequency.

Proxy-based Contrastive Replay (PCR). PCR Lin et al. (2023) addresses gradient imbalance by replacing anchor-to-sample relationships in the contrastive loss with anchor-to-proxy relationships. Specifically, for a training sample (x, y), the loss is defined as:

$$\mathcal{L}_{\text{PCR}} = -\frac{1}{|P(x)|}\sum_{p\in P(x)} \log \frac{e^{\left(\cos(\tilde{w}_p^\top \tilde{z})/\gamma\right)}}{\sum_{c\in C_B} e^{(\cos(\tilde{w}_c^\top \tilde{z})/\gamma)}}, \tag{6}$$

where $\cos(\cdot)$ denotes cosine similarity, $\tilde{z}$ and $\tilde{w}_c$ are ℓ_2-normalized representations and class proxies, respectively, τ is the temperature parameter, and C_B denotes the set of classes present in the current batch (allowing repetitions). Assuming n_c denotes the frequency of class c in the current batch, the loss simplifies to:

$$\mathcal{L}_{\text{PCR}} = -\log \frac{e^{\left(\cos(\tilde{w}_y^\top \tilde{z})/\tau\right)}}{\sum_{c\in\mathcal{C}_{1:t}} e^{(\cos(\tilde{w}_c^\top \tilde{z})/\tau)} \cdot n_c}. \tag{7}$$

DELTA. DELTA Raghavan et al. (2024) addresses long-tailed *online* CL by decoupling representation learning and classifier optimization. It adopts a two-stage training scheme: the encoder is first trained using a supervised contrastive loss to learn class-discriminative features, followed by a classifier fine-tuning stage using an *Equalization Loss*:

$$\mathcal{L}_{\text{EQ}} = -\log \frac{e^{(\ell_y + \log\pi(y))}}{\sum_{c\in\mathcal{C}_{1:t}} e^{(\ell_c + \log\pi(c))}}, \tag{8}$$

where ℓ_c denotes the logit for class c, and $\pi(c)$ is the estimated prior probability of class c, computed from the data observed so far up to task t in the training stream.

Logit-Adjusted Softmax (LAS). LAS Huang et al. (2023) performs logit adjustment using estimated class priors $\pi(c)$ computed over a sliding window of recent training batches, along with a temperature parameter τ:

$$\mathcal{L}_{\text{LAS}} = -\log \frac{e^{(\ell_y + \tau\log\pi(y))}}{\sum_{c\in\mathcal{C}_{1:t}} e^{(\ell_c + \tau\log\pi(c))}}. \tag{9}$$

Rather than maintaining cumulative statistics over the full data stream (as done in DELTA), LAS estimates $\pi(c)$ from a sliding window that tracks class frequencies over the most recent k batches.

3.4 Limitations and Motivation

Class frequency-based reweighting methods—such as DELTA, LAS, and PCR—aim to mitigate gradient imbalance by adjusting logits or loss contributions based on estimated class priors or class count. LAS and PCR use batch-local or sliding-window statistics, making them highly sensitive to short-term sampling fluctuations. When a class is missing from recent batches, its estimated frequency drops to zero, eliminating its softmax contribution and halting further learning. DELTA addresses this volatility by maintaining cumulative class priors over the entire data stream, improving stability. However, it neglects recency: two classes with identical cumulative counts may differ significantly in how recently they were observed. All three methods—DELTA, LAS, and PCR—treat class frequency as temporally static, failing to adapt reweighting based on how stale or current a class is in the buffer.

To overcome these limitations, we propose AA-RR, a method that explicitly regulates gradient propagation to ensure consistent, temporally adaptive, and balanced updates across all classes in CL.

4 Proposed Method

We propose Adaptive Augmented Reweighted Replay (AA-RR), a lightweight framework designed to improve generalization and robustness in rehearsal-based CL. AA-RR addresses the challenges of gradient imbalance, spurious correlation, and buffer distribution bias, which depend on which samples are retained and whether class and task balance are preserved. It does so through three core components: 1. an adaptive class-aware loss reweighting mechanism; 2. targeted data augmentation to mitigate spurious correlations; and 3. a representative sample selection strategy to improve memory quality.

In this section, we first describe the core formulation of our adaptive reweighting strategy, which directly addresses the gradient imbalance and recency bias inherent in buffer-based CL. We then introduce the augmentation and selection components, which further enhance generalization by reducing overfitting and improving buffer representativeness.

The overall AA-RR training procedure is summarized in Algorithm 1 in Appendix A.1, illustrating how these components are integrated into CL.

4.1 Adaptive Class-Aware Reweighting

We modify the standard objective in Eq. 2 by introducing a per-class scaling factor $\alpha_c > 0$, applied inside the softmax. For a training sample (x_i, y_i) with logits $\ell_{i,c}$, we compute the reweighted softmax cross-entropy loss as:

$$\mathcal{L}_{\text{AA-RR}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t\cup\mathcal{B}}\left[-\log\frac{e^{(\ell_y)}\cdot\alpha_y}{\sum_{j\in\mathcal{C}_{1:t}} e^{(\ell_j)}\cdot\alpha_j}\right]. \tag{10}$$

When all $\alpha_c = 1$, this reduces to the standard cross-entropy. Otherwise, α_c modifies both the softmax distribution and the gradient magnitude, correcting for class imbalance at training time.

Dynamic Per-Class Weighting. The class weights $\{\alpha_c\}$ are computed dynamically at each training iteration. Let t be the current task index, and $\mathcal{C}_t$ the set of classes introduced in task t. For each class c, we define:

$$\alpha_c = \left(\frac{1}{t-1}\right)^{\gamma_1}\cdot\left(\frac{1}{t-t_c}\right)^{\gamma_2}\cdot\left(\frac{f_c}{\bar{f}_{\mathcal{C}_t}+\hat{n}}\right)^{\gamma_3} \tag{11}$$

We apply Eq. 11 to all classes $c \in \mathcal{C}_{1:t}$, with the understanding that for current-task classes $c \in \mathcal{C}_t$, the reweighting terms are ignored and we set $\alpha_c = 1$. This defines the current task as the reference point, with no decay or correction applied.

For all previous-task classes $c \in \mathcal{C}_{<t}$, the three multiplicative terms in Eq. 11 serve distinct purposes. Where, t_c is the task number in which class c was introduced; f_c is the number of buffer samples for class c; $\bar{f}_{\mathcal{C}_t}$ is the average buffer count for current-task classes; $\hat{n} = \frac{n_{\text{task}}}{|\mathcal{C}_t|}$ is the expected per-class sample count for $\mathcal{D}_t$; $\gamma_1, \gamma_2, \gamma_3 \in \mathbb{R}_+$ are tuning parameters.

The three multiplicative terms address distinct aspects of training imbalance. **Term 1** $\left(\frac{1}{t-1}\right)^{\gamma_1}$ counteracts the growing disparity between buffered and current-task samples as training progresses. It adaptively amplifies the contribution of earlier-task classes by scaling their weights according to their diminishing presence in the training distribution, thus mitigating class imbalance caused by buffer dilution. In effect, this term addresses the imbalance between current and previous tasks. **Term 2** $\left(\frac{1}{t-t_c}\right)^{\gamma_2}$ applies a temporal decay based on the age of class c, giving higher priority to older classes at greater risk of forgetting. **Term 3** $\left(\frac{f_c}{\bar{f}_{\mathcal{C}_t}+\hat{n}}\right)^{\gamma_3}$ adjusts for repeated exposure of buffer samples. As training progresses, buffer samples are seen more frequently than current-task samples, which can cause the model to overfit them and lead to overconfident predictions.

In sum, our reweighted softmax loss provides a principled yet lightweight mechanism to balance class contributions during training. It mitigates the overfitting of recent classes and preserves performance on earlier tasks—without requiring architectural changes or auxiliary components.

4.2 Augmentation for OOD Robustness

A key insight from recent data-centric studies Yao et al. (2022); Noohdani et al. (2024) is that enhancing input diversity through strong, semantically preserving augmentations can substantially improve model robustness and reduce overfitting to spurious correlations. Following this principle, AA-RR adopts a data-centric augmentation approach to improve generalization under covariate shift in CL. By enriching the effective training distribution, the model becomes less sensitive to superficial, task-specific patterns and better equipped to handle OOD inputs.

Motivated by this perspective, we employ a strong augmentation pipeline inspired by Supervised Contrastive Learning (SupCon) Khosla et al. (2020). The pipeline includes transformations such as random cropping, horizontal flipping, color jittering, and grayscaling—augmentations known to produce diverse yet label-consistent views. These augmentations encourage the model to learn representations that are invariant to low-level appearance changes while preserving class-discriminative information.

For each training sample, we generate an augmented view using this pipeline and concatenate it with the original image in the training batch. To further mimic contrastive learning behavior, we replace the standard dot product in the softmax classifier with cosine similarity and introduce a temperature parameter τ to scale the logits. This yields the reweighted softmax cross-entropy objective applied to the combined batch:

$$\mathcal{L}_{\text{AA-RR}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t\cup\mathcal{B}}\left[-\log\frac{e^{\left(\cos(\tilde{w}_y^\top \tilde{z})/\tau\right)}\cdot\alpha_y}{\sum_{j\in\mathcal{C}_{1:t}} e^{\left(\cos(\tilde{w}_j^\top \tilde{z})/\tau\right)}\cdot\alpha_j}\right] \tag{12}$$

Here, $\tilde{z}$ is the ℓ_2-normalized feature representation of the input, $\tilde{w}_j$ is the normalized class weight vector, α_j is the class reweighting factor (defined in Section 4.1), and $\mathcal{C}_{1:t}$ denotes all classes observed up to task t.

This augmentation strategy strengthens representational invariance, expands the effective training distribution, and mitigates overfitting to spurious or task-specific patterns. Consequently, it significantly improves the model's ability to generalize to OOD samples—addressing a core limitation of existing rehearsal-based CL methods.

4.3 Correctness-Guided Buffer Management

We introduce a correctness-based strategy for exemplar selection and memory reallocation within the fixed-capacity buffer $\mathcal{B}$. Our approach measures how often each sample is predicted correctly during early training

and retains only those samples that are consistently well learned, while ensuring class- and task-balanced retention of informative exemplars across the learning sequence.

Correctness Scoring. During training on task $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$, we monitor each sample's prediction outcome over the first E epochs. Let $f_\theta^{(e)}$ denote the encoder parameters after epoch e, and w_c the classifier weight of class $c \in \mathcal{C}_{1:t}$. The correctness indicator for sample (x_i, y_i) at epoch e is defined as:

$$c_{e,i} = \mathbf{1}\left[\arg\max_{c \in \mathcal{C}_{1:t}} w_c^\top f_\theta^{(e)}(x_i) = y_i\right] \tag{13}$$

The cumulative correctness count of each sample is then, $r_i = \sum_{e=1}^{E} c_{e,i}$, where $r_i \in \{0, \ldots, E\}$ indicates the epochs number in which sample i was classified correctly. This score serves as a ranking criterion for buffer selection.

Task–Class Allocation. The memory buffer $\mathcal{B}$ maintains a fixed capacity B shared among all observed tasks. After completing task t, buffer capacity is distributed evenly across the t tasks, i.e. $m_t = \left\lfloor \frac{B}{t} \right\rfloor, \quad \sum_{i=1}^{t} m_i = B$. Each task quota m_t is further divided uniformly among its class set $\mathcal{C}_t$, i.e. $m_{t,c} = \left\lfloor \frac{m_t}{|\mathcal{C}_t|} \right\rfloor, \quad$ for each $c \in \mathcal{C}_t$. This ensures balanced memory allocation both across tasks and within their constituent classes.

Buffer Update Rule. At the end of training on $\mathcal{D}_t$, the buffer is updated according to the three steps; **1. Current-task selection:** For each class $c \in \mathcal{C}_t$, rank samples in $\mathcal{D}_t$ by their correctness counts r_i in descending order and select the top $m_{t,c}$ exemplars, $\mathcal{S}_{t,c} = \mathrm{Top}_{m_{t,c}}(\{(x_i, y_i) \in \mathcal{D}_t : y_i = c\}, r_i)$. **2. Retention from previous tasks:** For each prior task $i < t$ and class $c \in \mathcal{C}_i$, retain only the top $m_{i,c}$ samples in the existing buffer, ranked by their stored correctness values. **3. Merge and prune:** The updated buffer is obtained by combining the retained and newly selected subsets, $\mathcal{B} = \bigcup_{i=1}^{t} \bigcup_{c \in \mathcal{C}_i} \mathcal{S}_{i,c}, \quad |\mathcal{S}_{i,c}| = m_{i,c}$. If any class exceeds its quota $m_{i,c}$, the least-correct samples are removed.

This procedure guarantees that $|\mathcal{B}| = B$ at all times and that each class retains a balanced and correctness-prioritized subset of exemplars. By discarding less reliably learned samples when memory is full, the buffer evolves into a compact and stable repository of well-learned examples, effectively supporting CL without catastrophic forgetting.

5 Experiments

5.1 Setup

For fair and consistent comparison across CL methods, we extend the Mammoth framework following the experimental protocols described in Buzzega et al. (2020); Boschini et al. (2022). All experiments are conducted on a Quadro RTX 8000 GPU with CUDA 10.2.

Datasets. We evaluate all methods on three widely used class-incremental learning benchmarks. Split CIFAR-100 partitions the 100 classes into 10 sequential tasks of 10 classes each, with 32×32 pixel images Boschini et al. (2022). Split Mini-ImageNet follows a similar format, comprising 5 tasks of 20 classes each, with images resized to 32×32 pixels Sun et al. (2022) (results using the original 84×84 resolution are provided in the ablation study in Section 5.3). Split Tiny-ImageNet includes 200 classes divided into 10 tasks of 20 classes each, also resized to 32×32 pixels Buzzega et al. (2020). Each dataset provides 500 training samples per class. The number of test samples per class is 100 for Split CIFAR-100 and Mini-ImageNet, and 50 for Split Tiny-ImageNet.

Out-of-Distribution Evaluation. To assess robustness under natural distribution shifts, we construct an OOD evaluation set by applying a subset of corruption types from the ImageNet-C benchmark Hendrycks & Dietterich (2019). Specifically, we apply 11 corruptions: Gaussian noise, shot noise, impulse noise, defocus

blur, motion blur, zoom blur, snow, fog, elastic transform, pixelate, and JPEG compression (see Figure 4 in Appendix A.2). Each corruption is applied at a fixed severity level, and all are applied independently to the test set. This results in a test set that is 12× larger than the original, consisting of the clean images and 11 corrupted variants per example. We exclude the brightness and contrast corruptions due to direct overlap with training-time augmentations used in contrastive learning (via ColorJitter). In addition, we exclude frost and frosted glass blur to avoid over-representing spatially obstructive or blur-based corruptions. While these two corruptions introduce distinct visual effects—such as textured overlays (frost) and irregular spatial blur (frosted glass blur)—their perceptual impact overlaps partially with other selected corruptions, such as snow, fog, and defocus blur, which also reduce visibility or introduce high-frequency distortions. This selection ensures that the OOD evaluation covers a broad range of realistic distribution shifts while avoiding redundancy and overlap with training-time augmentations.

To quantify performance under these shifts, we compute the mean accuracy across all corruptions and original images. We define OOD generalization as the ability of a CL model, trained on clean, ordered tasks, to maintain high predictive accuracy when evaluated under distribution shift. These shifts manifest as test-time corruptions that preserve semantic labels but alter low-level appearance.

Let a model trained on a sequence of tasks $\{\mathcal{D}_1^{\text{train}}, \ldots, \mathcal{D}_T^{\text{train}}\}$, sampled from an in-distribution source $\mathbb{P}_{\text{train}}$. Let $\mathcal{D}_{\text{test}}^{\text{iid}} \sim \mathbb{P}_{\text{iid}}$ denote the clean test set. We define a family of corruption functions $\mathcal{Q} = \{q_1, \ldots, q_K\}$, each inducing a corrupted distribution $\mathbb{P}_{\text{ood}}^{(k)}$ by transforming inputs from the clean test set: $\mathbb{P}_{\text{ood}}^{(k)} = \{(x', y) \mid x' = q_k(x),\ (x, y) \sim \mathbb{P}_{\text{iid}}\},\ k = 1, \ldots, K$. For each corruption, we compute the model's classification accuracy: $\text{Acc}^{(k)} = \mathbb{E}_{(x,y)\sim\mathbb{P}_{\text{ood}}^{(k)}} \left[\mathbb{I}\{\arg\max_{c\in\mathcal{C}_{1:t}} w_c^\top f_\theta(x) = y\}\right]$. We include the clean test accuracy $\text{Acc}^{(0)}$ and define the overall OOD generalization as the mean accuracy across clean and corrupted variants: $\text{Acc}^{\text{ood}} = \frac{1}{K+1}\sum_{k=0}^{K} \text{Acc}^{(k)}$. High OOD generalization indicates that the model's performance is stable under covariate shift, reflecting robustness to unseen visual conditions that differ from the training distribution.

Metrics. We evaluate CL performance using two standard metrics: Average Accuracy (ACC) and Backward Transfer (BWT) Lopez-Paz & Ranzato (2017). ACC quantifies overall performance by averaging the test accuracy across all tasks after the final training stage. BWT measures forgetting by computing the average performance drop on past tasks once the model has completed training. Formally, these metrics are defined as, $\text{ACC} = \frac{1}{T}\sum_{j=1}^{T} \alpha_{T,j}, \quad \text{BWT} = \frac{1}{T-1}\sum_{j=1}^{T-1} \beta_{T,j}$. Where $\alpha_{i,j}$ denotes the accuracy on the test set of task j after training up to task i, and $\beta_{i,j} = \alpha_{i,j} - \alpha_{j,j}$ captures the degradation in task j's performance after subsequent training. T denotes the total number of tasks. All metrics are reported under the class-incremental setting, following prior works Boschini et al. (2022); Tong et al. (2025).

Baselines. Our evaluation includes eleven rehearsal-based CL methods: GEM Lopez-Paz & Ranzato (2017), A-GEM Chaudhry et al. (2018b), ER Chaudhry et al. (2019), GSS Aljundi et al. (2019), GDUMB Prabhu et al. (2020), MetaSP Sun et al. (2022), SOIF Sun et al. (2023), PCR Lin et al. (2023), ER-LAS Huang et al. (2023), DELTA Raghavan et al. (2024), and CSReL Tong et al. (2025).

Implementation Details. We adopt the implementations and hyperparameter configurations provided by the Mammoth framework Buzzega et al. (2020); Boschini et al. (2022), as well as those from MetaSP Sun et al. (2022), SOIF Sun et al. (2023), PCR Lin et al. (2023), ER-LAS Huang et al. (2023), DELTA Raghavan et al. (2024), and CSReL Tong et al. (2025). Additional details are included in Appendix A.3. All experiments employ ResNet-18 He et al. (2016) as the backbone architecture, trained from scratch. To ensure reliable comparison, each result is reported as the mean over five runs with fixed random seeds (0–4). We use a batch size of 32 for both current and memory samples. Training is conducted for 50 epochs per task using SGD, with learning rates of 0.1 on Split CIFAR-100 and 0.03 on both Split Mini-ImageNet and Split Tiny-ImageNet. Standard data augmentations, including random cropping and random horizontal flipping, are applied during training. For the reweighted softmax strategy, we configure the hyperparameters γ_1, γ_2, and γ_3 empirically by default as follows; For Ours$_\beta$, Ours$_\gamma$, and Ours$_\lambda$, all γ_1, γ_2, and γ_3 values are set to 1.0 on Split CIFAR-100 and Split Mini-ImageNet, while on Split Tiny-ImageNet we use $\gamma_1 = 1.0$ and $\gamma_2 = \gamma_3 = 0.5$. For AA-RR, we set $\gamma_1 = 1.0$ and $\gamma_2 = \gamma_3 = 0.5$ for Split CIFAR-100 and Split Mini-ImageNet, and $\gamma_1 = 1.0$, $\gamma_2 = 0.5$, $\gamma_3 = 0$ for Split Tiny-ImageNet. Sensitivity analyses for these hyperparameters are

Table 1: Results for class-incremental learning: Average Accuracy (ACC; higher is better) and Backward Transfer (BWT; less negative is better) under i.i.d. and OOD settings, averaged over 5 runs. The 'Mean' column shows the average performance of each method across all datasets. Best results are bolded, and second-best are underlined. Note: BWT for GDUMB is omitted due to computational constraints.

Method	Dataset	Split CIFAR-100						Split Mini-ImageNet						Split Tiny-ImageNet						*Mean*	
	Buffer	500		1000		2000		500		1000		2000		500		1000		2000			
	Setting	i.i.d.	OOD	i.i.d.	OOD	i.i.d.	OOD	i.i.d.	OOD	i.i.d.	OOD	i.i.d.	OOD	i.i.d.	OOD	i.i.d.	OOD	i.i.d.	OOD	i.i.d.	OOD
GEM	***ACC***	21.33	3.36	31.91	4.69	36.33	4.98	18.04	2.97	19.02	2.97	18.99	2.94	6.07	1.18	6.92	1.20	7.08	1.22	18.41	2.83
	BWT	-66.83	-18.85	-48.93	-15.75	-34.40	-12.81	-51.53	-9.09	-46.69	-8.33	-42.43	-7.12	-46.84	-8.75	-41.11	-7.80	-35.97	-6.59	-46.08	-10.57
A-GEM	***ACC***	9.32	2.31	9.23	2.39	9.25	2.36	14.65	2.72	14.69	2.82	14.74	2.86	6.30	1.30	6.30	1.33	6.23	1.29	10.08	2.15
	BWT	-85.68	-22.42	-85.05	-22.16	-86.07	-22.39	-66.56	-12.56	-66.79	-12.54	-66.60	-12.46	-64.18	-12.66	-64.71	-12.81	-64.61	-12.75	-72.25	-15.86
GDUMB	***ACC***	10.36	1.76	16.35	2.45	25.47	3.33	7.09	1.49	10.15	1.83	14.33	2.33	3.59	0.79	5.15	0.93	7.53	1.17	11.11	1.78
	BWT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	—	—
ER	***ACC***	19.29	3.32	24.46	3.87	31.94	4.53	18.02	3.16	20.78	3.39	24.72	3.67	8.90	1.44	10.75	1.61	13.88	1.86	19.19	2.98
	BWT	-75.53	-19.69	-69.29	-18.80	-61.18	-17.79	-63.62	-11.60	-60.07	-10.98	-55.12	-9.68	-68.53	-12.40	-67.12	-12.05	-63.47	-11.21	-64.88	-13.80
GSS	***ACC***	22.23	3.45	28.84	4.13	35.27	4.87	18.48	3.19	21.91	3.43	28.16	3.96	9.23	1.49	11.45	1.70	15.52	2.03	21.23	3.14
	BWT	-70.26	-16.85	-61.81	-15.86	-53.51	-13.79	-61.72	-11.43	-55.95	-10.57	-46.80	-8.80	-67.56	-11.64	-64.05	-11.00	-58.06	-9.74	-59.97	-12.19
MetaSP	***ACC***	21.87	3.78	26.23	3.94	29.74	4.49	22.92	3.57	27.84	4.03	33.39	4.61	11.34	1.70	14.72	1.95	19.22	2.36	23.03	3.38
	BWT	-71.01	-16.03	-65.53	-16.04	-59.90	-14.01	-60.40	-10.78	-53.28	-9.44	-44.64	-8.04	-67.63	-11.73	-63.03	-10.97	-56.61	-9.73	-60.23	-11.86
SOIF	***ACC***	18.85	3.18	26.19	3.96	25.36	4.03	18.75	3.23	22.85	3.56	23.46	3.68	9.10	1.46	11.66	1.69	11.58	1.65	18.64	2.94
	BWT	-71.47	-19.72	-62.91	-17.04	-62.11	-16.44	-65.70	-12.35	-58.65	-11.39	-54.76	-10.74	-69.63	-12.61	-65.65	-12.04	-63.64	-11.19	-63.84	-13.72
CSReL	***ACC***	26.61	3.95	32.27	4.51	35.18	4.89	23.31	3.84	28.25	4.23	34.54	4.67	11.50	1.63	15.57	2.00	21.15	2.52	25.38	3.58
	BWT	-61.31	-17.18	-51.37	-14.45	-44.39	-13.53	-57.60	-10.84	-48.51	-8.93	-36.68	-7.27	-65.44	-11.89	-60.05	-10.76	-50.83	-9.16	-52.91	-11.56
Ours_{α}	***ACC***	28.77	4.36	34.91	4.95	42.18	5.57	24.14	3.66	30.58	4.35	36.50	4.78	11.58	1.73	16.54	2.18	23.27	2.74	27.61	3.81
	BWT	-59.56	-16.31	-50.55	-13.98	-39.99	-12.54	-56.66	-10.06	-47.15	-8.47	-36.89	-6.94	-66.03	-11.72	-59.02	-10.28	-49.56	-8.73	-51.71	-11.00
PCR	***ACC***	23.34	11.61	33.03	15.01	37.63	17.03	17.08	8.28	20.20	9.01	24.77	10.62	10.05	4.57	13.36	5.42	16.37	6.14	21.76	9.74
	BWT	-58.22	-34.36	-48.70	-31.34	-49.98	-35.31	-61.13	-30.38	-57.65	-30.92	-52.41	-29.85	-60.48	-30.56	-55.96	-29.99	-51.24	-29.24	-55.09	-31.33
Ours_{β}	***ACC***	28.65	13.43	34.95	15.92	40.74	18.15	23.17	10.26	26.02	11.09	29.90	12.59	10.55	4.65	12.87	5.09	16.75	6.28	24.85	10.83
	BWT	-47.60	-21.31	-42.18	-19.51	-40.06	-20.16	-46.33	-19.80	-46.09	-21.64	-42.62	-21.75	-56.87	-26.53	-52.15	-24.64	-50.38	-26.31	-47.14	-22.41
DELTA	***ACC***	13.92	6.74	19.88	8.39	28.25	11.71	16.41	6.77	19.07	7.18	24.36	8.87	8.03	2.99	9.21	3.33	12.99	4.05	16.90	6.67
	BWT	-79.24	-42.02	-71.41	-38.70	-59.83	-31.11	-61.32	-24.59	-56.77	-23.04	-48.48	-19.86	-64.88	-26.55	-62.44	-25.18	-55.95	-21.85	-62.26	-28.10
Ours_{γ}	***ACC***	21.04	8.08	28.08	10.60	36.88	13.71	22.53	7.34	25.56	8.27	29.39	9.55	10.87	3.35	12.69	3.68	15.44	4.40	22.50	7.67
	BWT	-56.33	-22.62	-48.56	-19.91	-38.26	-15.64	-39.63	-13.65	-35.05	-12.22	-29.25	-10.00	-54.16	-19.76	-49.09	-17.20	-41.63	-14.14	-43.55	-16.13
ER-LAS	***ACC***	30.61	4.51	37.61	5.21	45.69	5.99	23.16	3.56	28.80	4.04	34.73	4.64	12.50	1.77	17.16	2.16	23.02	2.69	28.14	3.84
	BWT	-59.76	-14.84	-49.48	-12.09	-37.88	-10.11	-55.61	-9.71	-46.01	-8.25	-37.82	-6.44	-62.06	-11.02	-55.03	-9.24	-45.77	-7.67	-49.94	-9.93
Ours_{λ}	***ACC***	**36.56**	5.18	**43.60**	5.81	**48.67**	6.51	28.41	3.90	**33.14**	4.27	**37.42**	4.64	**16.37**	2.05	**20.23**	2.38	**24.51**	2.84	**32.10**	4.18
	BWT	-37.02	**-6.73**	-29.13	-5.82	-21.76	-4.72	-37.17	**-6.36**	**-29.64**	**-5.36**	**-23.03**	**-4.35**	**-41.23**	**-5.94**	**-33.99**	**-4.82**	**-27.15**	**-3.72**	**-31.13**	**-5.31**
AA-RR	***ACC***	33.91	**15.79**	38.16	**17.97**	41.34	**19.93**	**29.03**	**11.98**	30.69	**13.07**	34.26	**14.77**	12.77	**5.41**	16.96	**6.87**	19.97	**8.27**	28.57	**12.67**
	BWT	**-27.89**	-8.35	**-24.11**	**-5.48**	**-18.15**	**-2.67**	**-33.56**	-12.87	-34.60	-12.39	-29.42	-10.50	-50.38	-21.04	-42.28	-17.13	-37.80	-15.07	-33.13	-11.72

reported in Section 5.3 and visualized in Figure 2. In our sample selection strategy, the hyperparameter E is empirically set to 9 by default for both Ours_{α} and AA-RR. A sensitivity analysis of this setting is also provided in the ablation study (see Section 5.3, Figure 3).

5.2 Main Results

Table 1 reports ACC and BWT under both i.i.d. and OOD settings, comparing our approach against competitive baselines. To fairly isolate the effect of each component of our method, we evaluate four variants: Ours_{α}, Ours_{β}, Ours_{γ}, and Ours_{λ}. Specifically, Ours_{α} integrates our sample selection strategy (see Section 4.3) into ER, using the standard softmax cross-entropy loss as in ER, GSS, MetaSP, SOIF, and CSReL. All these baselines, except ER, incorporate buffer update strategies, enabling a fair comparison. As shown, our selection strategy consistently outperforms these methods on average.

In Ours_{β}, Ours_{γ}, and Ours_{λ}, we replace the reweighting mechanisms of PCR, DELTA, and ER-LAS, respectively, with our own reweighting strategy (see Section 4.1), while keeping reservoir sampling. Note that PCR and DELTA use contrastive learning and therefore operate on both original and augmented samples. This setup improves their OOD performance but often degrades i.i.d. performance compared to ER-LAS. Our approach achieves stronger average performance across both settings compared to these methods.

Finally, AA-RR integrates all components—sample selection, reweighting, and contrastive-style augmentation (see Section 4.2). It achieves the highest ACC in OOD settings, remaining competitive under i.i.d. conditions. Regarding BWT, some baselines report favorable values in OOD settings, but these often coincide with low ACC, suggesting that the models failed to learn sufficient knowledge to retain, much less forget.

Final class/task distribution in the buffer. We analyze the buffer at the end of training for each method, extracting the number of samples per class and task. Subsequently, we calculate the Coefficient of Variation (CV) for classes and tasks using the formula $\left(\frac{\text{Standard Deviation}}{\text{Mean}}\right) \times 100\%$. The results are presented in Table 2. A CV of 0.00 signifies perfect balance, with higher values indicating greater imbalance.

Only our buffer update policy (used in Ours$_\alpha$ and AA-RR) achieves perfectly class- and task-balanced buffer distributions, with a CV of 0.00.

Table 2: Coefficient of Variation (CV) of class/task distribution in the buffer at the end of training on Split CIFAR-100 (buffer: 1000).

Method	GEM	A-GEM	ER	GSS	MetaSP	SOIF	CSReL	Ours$_\alpha$	PCR	Ours$_\beta$	DELTA	Ours$_\gamma$	ER-LAS	Ours$_\lambda$	AA-RR
CV of Tasks	0.00	0.00	06.80	58.79	26.96	29.87	0.00	0.00	06.80	06.80	06.80	06.80	06.80	06.80	0.00
CV of Classes	28.91	29.77	31.43	68.38	53.85	63.97	44.25	0.00	30.56	28.98	33.59	31.75	31.43	29.70	0.00

Running Time. Table 3 Compares the per-hour running time of each method on Split CIFAR-100 (buffer size: 2000), measured on a Quadro RTX 8000 GPU. Integrating our modules adds only negligible overhead on the baselines. Notably, methods with buffer update policies tend to incur high computational costs. Interestingly, AA-RR is even faster than PCR, as it updates the buffer only once per task—after training—rather than batch-by-batch as in reservoir sampling. Consequently, the buffer remains empty during the first task, further reducing AA-RR's runtime.

Table 3: Running times per hour for different methods on Split CIFAR-100 (buffer size 2000) using a Quadro RTX 8000.

Method	GEM	A-GEM	ER	GSS	MetaSP	SOIF	CSReL	Ours$_\alpha$	PCR	Ours$_\beta$	DELTA	Ours$_\gamma$	ER-LAS	Ours$_\lambda$	AA-RR
Runtime (hours)	20.42	1.85	1.4	15.56	2.87	8.13	2.52	1.4	3.6	3.68	3.78	3.98	1.42	1.5	3.55

5.3 Ablation.

Correctness-Based Update Policies. We ablate the buffer update policy in Ours$_\alpha$ by varying the selection criterion at epoch E. In addition to our default strategy of storing the most consistently correct samples ("Highest"), we evaluate three alternatives: "Lowest," "Middle," and "Uniform" sampling, all under class- and task-balance constraints.

As shown in Table 4, the "Highest" strategy consistently yields the best performance—except on Split CIFAR-100 (BWT, buffer size 2000), where it is nearly matched by "Middle." These results highlight the importance of informed sample selection for improving generalization and retention.

Table 4: Ablation of buffer update policy in Ours$_\alpha$. "Highest" selects the most consistently correct samples, "Middle" selects mid-correct samples, and "Lowest" selects the least correct.

	Setting	**i.i.d.**									
	Dataset	**Split CIFAR-100**				**Split Mini-ImageNet**				***Mean***	
	Metric	**ACC**		**BWT**		**ACC**		**BWT**		**ACC**	**BWT**
	Buffer	**1000**	**2000**	**1000**	**2000**	**1000**	**2000**	**1000**	**2000**		
Method	**Uniform**	32.62	40.79	-52.74	-41.53	27.53	33.90	-50.87	-39.51	33.71	-46.16
	Lowest	20.45	26.80	-66.53	-58.67	19.70	23.65	-60.59	-53.15	22.65	-59.74
	Middle	33.99	41.58	-51.55	**-39.80**	28.02	34.68	-49.95	-38.95	34.57	-45.06
	Highest	**34.91**	**42.18**	**-50.55**	-39.99	**30.58**	**36.50**	**-47.15**	**-36.89**	**36.04**	**-43.65**

Hyperparameter sensitivity. We examine the sensitivity of the hyperparameters γ_1, γ_2, and γ_3 used in our softmax reweighting strategy of Ours$_\lambda$. In Figure 2, we fix $\gamma_1 = 1.0$ and vary γ_2 and γ_3 over $\{0.0, 0.5, 1.0, 1.5, 2.0\}$. For experiments varying $\gamma_1 \in \{0.0, 0.5, 1.0, 1.5, 2.0\}$, see Appendix A.4.

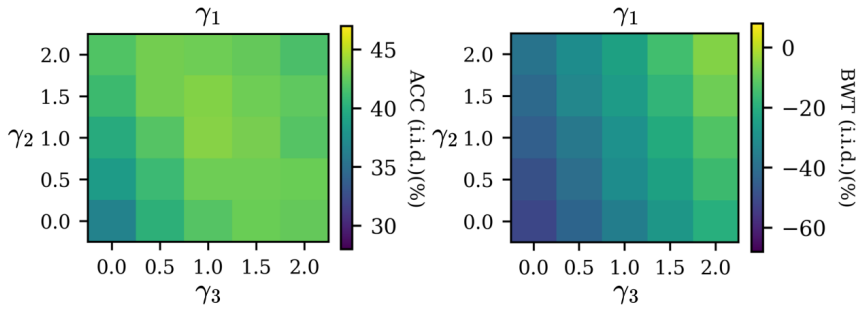

Figure 2: Sensitivity analysis of our softmax reweighting hyperparameters in Ours$_\lambda$.

In Figure 3, we examine the sensitivity of our hyperparameter E, which is integral to our memory update policy in Ours$_\alpha$. As illustrated, the performance of hyperparameter E remains stable across a wide range of epochs on all three datasets and buffer sizes, maintaining consistency in both ACC and BWT under i.i.d. conditions.

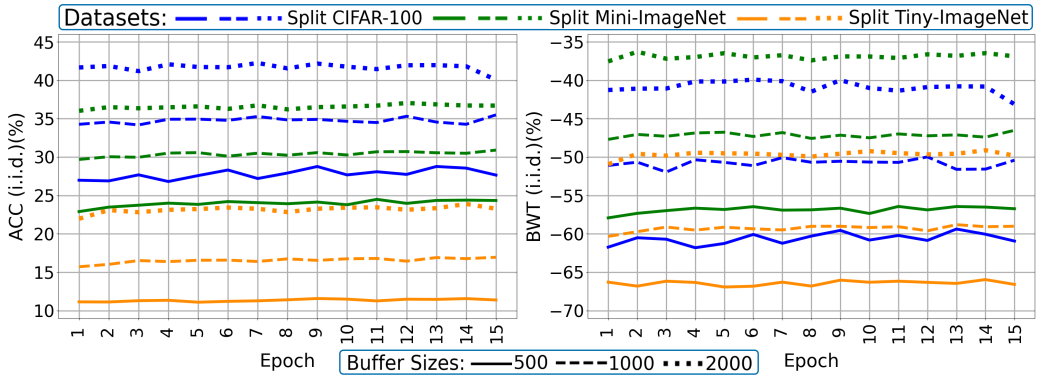

Figure 3: Sensitivity analysis of hyperparameter E in Ours$_\alpha$. Performance remains stable across a wide range of E values, with consistent ACC and BWT across Split CIFAR-100, Split Mini-ImageNet, and Split Tiny-ImageNet, and buffer sizes 500, 1000, and 2000 under i.i.d. conditions.

Performance on Split Mini-ImageNet (84×84 Resolution). Appendix A.5 presents results on Split Mini-ImageNet using the original image resolution of 84×84 pixels and a buffer size of 1000, to assess the impact of image downscaling on performance.

6 Conclusion

In this work, we addressed a critical yet underexplored limitation of rehearsal-based continual learning (CL)—its vulnerability to overfitting and poor generalization under distributional shift. We introduced AA-RR, a lightweight framework that integrates adaptive class-aware reweighting, principled sample selection, and targeted data-centric augmentation to mitigate the effects of gradient imbalance, spurious correlations, and buffer-induced class imbalance. Through extensive experiments, we showed that AA-RR not only improves in-distribution performance but also significantly enhances robustness to covariate shifts, outperforming existing state-of-the-art methods. Our findings underscore the importance of designing CL algorithms that not only retain past knowledge but also preserve generalization beyond the training distribution.

References

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.

Lorenzo Bonicelli, Matteo Boschini, Angelo Porrello, Concetto Spampinato, and Simone Calderara. On the effectiveness of lipschitz-driven rehearsal in continual learning. *Advances in Neural Information Processing Systems*, 35:31886–31901, 2022.

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.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018a.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018b.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019.

Cian Eastwood, Shashank Singh, Andrei L Nicolicioiu, Marin Vlastelica Pogančić, Julius von Kügelgen, and Bernhard Schölkopf. Spuriosity didn't kill the classifier: Using invariant predictions to harness spurious features. *Advances in Neural Information Processing Systems*, 36:18291–18324, 2023.

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

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Zhehao Huang, Tao Li, Chenhe Yuan, Yingwen Wu, and Xiaolin Huang. Online continual learning via logit adjusted softmax. *arXiv preprint arXiv:2311.06460*, 2023.

Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in neural information processing systems*, 32, 2019.

Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. *Advances in Neural Information Processing Systems*, 34:29193–29205, 2021.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

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.

Huiwei Lin, Baoquan Zhang, Shanshan Feng, Xutao Li, and Yunming Ye. Pcr: Proxy-based contrastive replay for online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24246–24255, 2023.

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

Fangrui Lv, Jian Liang, Shuang Li, Bin Zang, Chi Harold Liu, Ziteng Wang, and Di Liu. Causality inspired representation learning for domain generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8046–8056, 2022.

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*, pp. 7765–7773, 2018.

Fahimeh Hosseini Noohdani, Parsa Hosseini, Aryan Yazdan Parast, Hamidreza Yaghoubi Araghi, and Mahdieh Soleymani Baghshah. Decompose-and-compose: A compositional approach to mitigating spurious correlation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27662–27671, 2024.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 524–540, 2020.

Siddeshwar Raghavan, Jiangpeng He, and Fengqing Zhu. Delta: Decoupling long-tailed online continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4054–4064, 2024.

Dvir Samuel and Gal Chechik. Distributional robustness loss for long-tail learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9495–9504, 2021.

Jiang-Xin Shi, Tong Wei, Yuke Xiang, and Yu-Feng Li. How re-sampling helps for long-tail learning? *Advances in Neural Information Processing Systems*, 36, 2023.

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, pp. 9630–9638, 2021.

Qing Sun, Fan Lyu, Fanhua Shang, Wei Feng, and Liang Wan. Exploring example influence in continual learning. *Advances in Neural Information Processing Systems*, 35:27075–27086, 2022.

Zhicheng Sun, Yadong Mu, and Gang Hua. Regularizing second-order influences for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20166–20175, 2023.

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*, pp. 99–108, 2022.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Ruilin Tong, Yuhang Liu, Javen Qinfeng Shi, and Dong Gong. Coreset selection via reducible loss in continual learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Rahul Venkataramani, Parag Dutta, Vikram Melapudi, and Ambedkar Dukkipati. Causal feature alignment: Learning to ignore spurious background features. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 4666–4674, 2024.

Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9385–9394, 2021.

Huaxiu Yao, Yu Wang, Sai Li, Linjun Zhang, Weixin Liang, James Zou, and Chelsea Finn. Improving out-of-distribution robustness via selective augmentation. In *International Conference on Machine Learning*, pp. 25407–25437. PMLR, 2022.

Wenqian Ye, Guangtao Zheng, Xu Cao, Yunsheng Ma, and Aidong Zhang. Spurious correlations in machine learning: A survey. *arXiv preprint arXiv:2402.12715*, 2024.

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

Longhui Yu, Tianyang Hu, Lanqing Hong, Zhen Liu, Adrian Weller, and Weiyang Liu. Continual learning by modeling intra-class variation. *arXiv preprint arXiv:2210.05398*, 2022.

Michael Zhang, Nimit S Sohoni, Hongyang R Zhang, Chelsea Finn, and Christopher Ré. Correct-n-contrast: A contrastive approach for improving robustness to spurious correlations. *arXiv preprint arXiv:2203.01517*, 2022a.

Yaqian Zhang, Bernhard Pfahringer, Eibe Frank, Albert Bifet, Nick Jin Sean Lim, and Yunzhe Jia. A simple but strong baseline for online continual learning: Repeated augmented rehearsal. *Advances in Neural Information Processing Systems*, 35:14771–14783, 2022b.

A Appendix

A.1 Algorithm

The overall AA-RR training procedure is summarized in Algorithm 1, illustrating how our components are integrated into CL.

A.2 OOD Images

Figure 4 illustrates the various corruption types applied to generate OOD samples from in-distribution data. These corruptions simulate distributional shifts that commonly occur in real-world settings, enabling a more robust evaluation of model generalization. The figure includes noise-based corruptions such as Gaussian noise, shot noise, and impulse noise; blur types including defocus, motion, and zoom blur; as well as appearance-altering transformations like fog, snow, elastic distortion, pixelation, and JPEG compression. Each corrupted version is derived from the same original image (top-left), demonstrating the visual impact of each corruption type used for OOD data.

A.3 Implementation Details

For SOIF, we follow the original paper and set the probability of selecting an already-in-coreset sample to 0.5, the weight for second-order influence functions to 0.01, and the NTK regularization coefficient to 1e-3. For PCR and DELTA, we use the temperature value of 0.09, consistent with their respective implementations. For ER-LAS, we adopt the default settings from the original paper, setting the logit adjustment parameter and the sliding window length both to 1. For CSReL hyperparameters, we used the following configurations. On Split CIFAR-100, the main continual learner was trained with a learning rate of 2e-2, a loss factor of 4.0, and 40 selection steps. The selected subset was trained with a learning rate of 5e-3 for 7 steps. The holdout model was trained for 10 epochs with a learning rate of 3e-3 and 200 samples per previous task. For Split Mini-ImageNet and Split Tiny-ImageNet, we maintained the main learner's learning rate at 2e-2 and the loss factor at 4.0, but increased the selection steps to 250. The selected subset was trained with a learning rate of 5e-3 for 20 steps. The holdout model used the same learning rate (3e-3), but was trained for 20 epochs using 200 samples per previous task.

Algorithm 1 Adaptive Augmented Reweighted Replay (AA-RR)

Require: Task sequence $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$, buffer size B, temperature τ, reweighting parameters $\gamma_1, \gamma_2, \gamma_3$, early-epoch window E, task epochs H

Initialize model θ, classifier weights $\{w_c\}$, buffer $\mathcal{B} \leftarrow \emptyset$
for task $t = 1$ to T **do**
  Compute per-class reweighting factor α_c:

$$\alpha_c = \begin{cases} 1, & \text{if } c \in \mathcal{C}_t \\ \left(\frac{1}{t-1}\right)^{\gamma_1} \cdot \left(\frac{1}{t-t_c}\right)^{\gamma_2} \cdot \left(\frac{f_c}{\bar{f}_{\mathcal{C}_t} + \hat{n}}\right)^{\gamma_3}, & \text{if } c \in \mathcal{C}_{<t} \end{cases}$$

  for epoch $e = 1$ to H **do**
    for batch $(x, y, x_{\text{orig}}, i)$ in $\mathcal{D}_t$ **do**
      Current task:
      Apply strong SupCon-style augmentation: $x' \leftarrow \text{Augment}(x)$
      Concatenate: $x_{\text{cur}} \leftarrow [x; x']$, $y_{\text{cur}} \leftarrow [y; y]$
      Get embedding: $z_{\text{cur}} \leftarrow f_\theta(x_{\text{cur}})$
      if $e \leq E$ **then**
        Correctness tracking:
        For each $(x_i^{ori}, y_i) \in \mathcal{D}_t$, store correctness:

$$c_{e,i} \leftarrow \mathbf{1}\left[\arg\max_{c \in \mathcal{C}_{1:t}} w_c^\top f_\theta(x_i^{ori}) = y_i\right]$$

      end if
      Buffer:
      if $\mathcal{B} \neq \emptyset$ **then**
        Sample (x_b, y_b) from buffer
        Apply strong SupCon-style augmentation: $x_b' \leftarrow \text{Augment}(x_b)$
        Concatenate: $x_{\text{buf}} \leftarrow [x_b; x_b']$, $y_{\text{buf}} \leftarrow [y_b; y_b]$
        Get embedding: $z_{\text{buf}} \leftarrow f_\theta(x_{\text{buf}})$
        Combine: $z \leftarrow [z_{\text{cur}}; z_{\text{buf}}]$, $y \leftarrow [y_{\text{cur}}; y_{\text{buf}}]$
      else
        $z \leftarrow z_{\text{cur}}, \; y \leftarrow y_{\text{cur}}$
      end if
      Compute cosine logits: $\ell_c = \frac{\tilde{w}_c^\top \tilde{z}}{\tau}$
      Compute reweighted softmax cross-entropy loss:

$$\mathcal{L} = -\log\left(\frac{e^{\ell_y} \cdot \alpha_y}{\sum_{j \in \mathcal{C}_{1:t}} e^{\ell_j} \cdot \alpha_j}\right)$$

      Update model θ, and classifier weights $\{w_c\}$ via gradient descent
    end for
  end for
  Buffer Management:
    Compute total correctness scores: $r_i = \sum_{e=1}^{E} c_{e,i}, \quad \text{for all } x_i^{ori} \in \mathcal{D}_t$
    Allocate per-task buffer budget: $m_t \leftarrow \lfloor B/t \rfloor$
    For each class $c \in \mathcal{C}_t$: $m_{t,c} \leftarrow \lfloor m_t / |\mathcal{C}_t| \rfloor$
    Select top-$m_{t,c}$ current-task samples by r_i
    For each prior class $c \in \mathcal{C}_{<t}$, retain top-$m_{i,c}$ from existing buffer
    Update buffer: $\mathcal{B} \leftarrow \bigcup_{i=1}^{t} \bigcup_{c \in \mathcal{C}_i} \mathcal{S}_{i,c}$ where $|\mathcal{S}_{i,c}| = m_{i,c}$
end for
Output: Updated model parameters θ, classifier weights $\{w_c\}$, and buffer $\mathcal{B}$

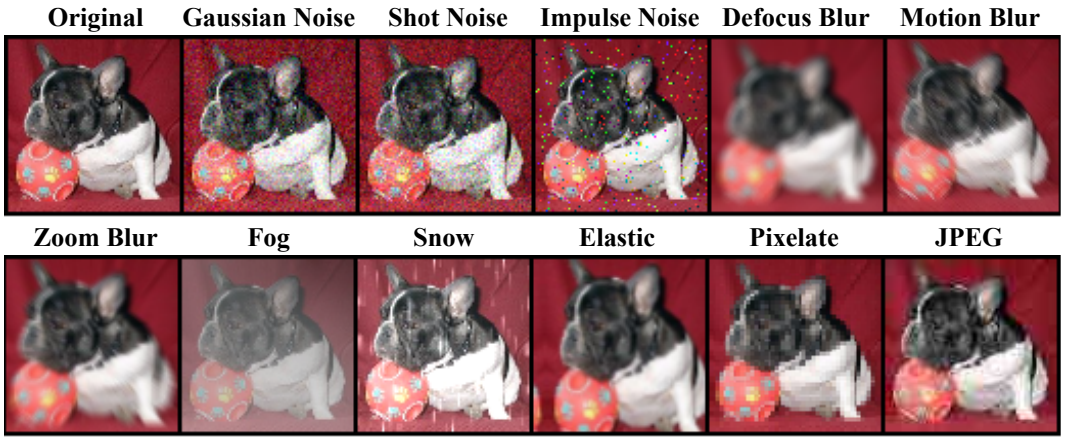

Figure 4: Examples of corruption types used to generate OOD samples. Starting from the original image (top-left), various corruptions such as Gaussian Noise, Shot Noise, Impulse Noise, different types of blur (Defocus, Motion, Zoom), and artifacts like Fog, Snow, Elastic distortions, Pixelation, and JPEG compression are applied to simulate distributional shifts.

A.4 Hyperparameter Sensitivity

Figure 5 presents a joint sensitivity analysis of the hyperparameters γ_1, γ_2, and γ_3 in our Ours$_\lambda$ model under the i.i.d. setting. We systematically vary each parameter over the range $\{0.0, 0.5, 1.0, 1.5, 2.0\}$ and report the corresponding changes in ACC and BWT. For each value of γ_1 (shown across columns), we visualize the impact of jointly varying γ_2 (vertical axis) and γ_3 (horizontal axis). The results indicate that performance is generally robust to moderate changes in these hyperparameters, with consistent improvements in backward transfer for higher values.

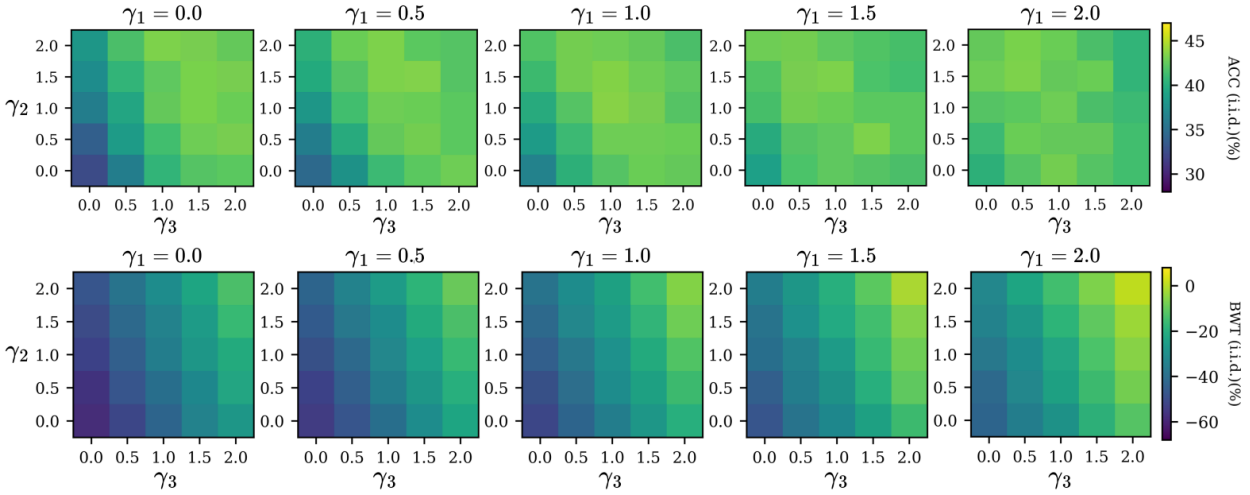

Figure 5: Joint sensitivity analysis of the hyperparameters γ_1, γ_2, and γ_3 in our Ours$_\lambda$ model under the i.i.d. setting. Each parameter is systematically varied over the range $\{0.0, 0.5, 1.0, 1.5, 2.0\}$, and the resulting effects on ACC and BWT are reported. For each value of γ_1 (arranged across columns), we visualize how changes in γ_2 (vertical axis) and γ_3 (horizontal axis) jointly affect performance. The results show that the model remains robust to moderate hyperparameter variation, with higher values generally leading to improved backward transfer.

A.5 Performance on Split Mini-ImageNet (84×84 Resolution)

Table 5 reports results on Split Mini-ImageNet using the original image resolution of 84×84 pixels with a buffer size of 1000. This evaluation allows us to examine the effect of image resolution on performance. We compare methods under both i.i.d. and OOD settings, reporting ACC and BWT. All hyperparameters match those used in the main experiments, except for Ours$_\gamma$, where we set $\gamma_1 = 1.0$, $\gamma_2 = 0.5$, and $\gamma_3 = 0.5$.

Table 5: Results on Split Mini-ImageNet with the original image resolution of 84×84 pixels and a buffer size of 1000. Methods are compared under both i.i.d. and OOD settings, with ACC and BWT reported.

Setting	Dataset	Split Mini-ImageNet [84 × 84] — buffer size: 1000															
	Method	GEM	A-GEM	GDUMB	ER	GSS	MetaSP	SOIF	CSReL	Ours$_\alpha$	PCR	Ours$_\beta$	DELTA	Ours$_\gamma$	ER-LAS	Ours$_\lambda$	AA-RR
i.i.d.	***ACC***	30.34	16.48	13.71	26.57	24.34	35.09	29.14	35.09	36.37	27.53	30.25	24.47	30.09	36.61	44.25	36.00
	BWT	-55.53	-76.84	-	-65.52	-66.45	-54.47	-63.29	-51.21	-49.77	-57.68	-48.72	-61.18	-35.84	-50.84	-28.89	-20.11
OOD	***ACC***	4.51	3.23	2.12	4.08	3.83	4.91	4.37	4.91	4.93	13.79	15.67	9.65	9.73	4.89	5.24	18.30
	BWT	-10.34	-13.43	-	-10.52	-11.26	-9.10	-11.47	-9.01	-8.21	-38.10	-27.82	-28.16	-14.23	-7.82	-4.54	-7.75