CLOSING THE STABILITY–PLASTICITY LOOP: ADAPTIVE CAPACITY RENEWAL FOR CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Class-Incremental Learning (CIL) requires a model to acquire new classes while preserving a usable representation for earlier classes. Existing approaches usually treat this as either a capacity problem (making parameters available for new classes) or a representation problem (constraining feature drift), leaving the two decisions weakly coupled. We formulate the problem as a closed-loop stability–plasticity control problem and propose Closing the Stability–Plasticity Loop (CSPL). CSPL uses one utility-and-reference loop: a maturity-aware utility signal identifies channels whose capacity has become ineffective; function-preserving replacement recycles only those channels; and a shared spatial context memory provides a task-agnostic reference for the resulting features. The reference is enforced through the task-agnostic/task-specific branches and feature distillation, so that capacity renewal is performed without changing the function seen by the stabilizing pathway. Experiments on CIFAR100, Tiny-ImageNet, ImageNet100, and CUB200 show consistent improvements over strong baselines. Under the 50–10 protocol, CSPL improves Last accuracy over the strongest listed baseline by 2.16 percentage points on CIFAR100 and 3.51 points on ImageNet100.

1 INTRODUCTION

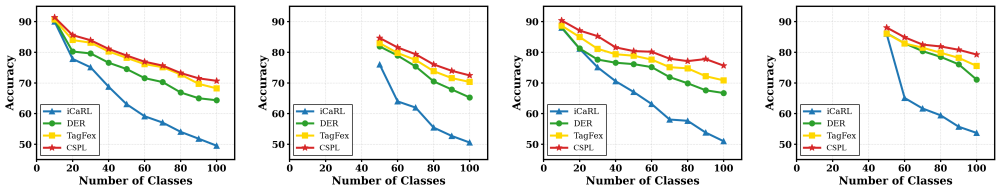
Continuous Learning (CL) enables artificial intelligence systems to adapt to evolving environments by integrating new knowledge over time, akin to human learning He et al. (2025); Wickramasinghe et al. (2024). A particularly challenging setting is Class-Incremental Learning (CIL), where a model must sequentially learn new classes without access to task identifiers during inference Fei et al. (2023). This capability is critical for real-world applications such as autonomous driving and robotics, where agents must continuously recognize novel objects while retaining knowledge of previously seen ones Mundt et al. (2022); Hayes et al. (2019).

However, CIL models face the fundamental stability-plasticity dilemma McCloskey & Cohen (1989). On one hand, models must possess sufficient plasticity to acquire new features; on the other, they must maintain stability to prevent catastrophic forgetting, where new learning overwrites existing representations Robins (1995). Additionally, models suffer from representation drift Aljundi et al. (2018), where the feature space becomes progressively distorted as network parameters are updated. To alleviate this, contrastive learning strategies have been introduced to learn more robust, drift-resistant representations during sequential updates Cha et al. (2021). These challenges are compounded by strict constraints on memory and computation, often precluding the retraining on historical data Ostapenko et al. (2022).

To address these issues, existing methods provide useful but separate mechanisms. Replay and regularization preserve old information, whereas dynamic architectures and neuron utility methods create capacity for new information. The missing design question is how a capacity change can be made observable to, and constrained by, a stable representation. In particular, resetting a channel can improve plasticity but can also introduce a discontinuity in the feature space; conversely, a strong representation constraint can preserve old features while leaving redundant capacity unused.

We therefore propose Closing the Stability–Plasticity Loop (CSPL) as a single capacity–reference loop for CIL. The loop has a clear division of state and action. A maturity-aware utility estimator

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) (b) (c) (d)

Figure 1: Accuracy curves for baselines and CSPL on CIFAR100 and ImageNet100. (a) CIFAR100 10–10. (b) CIFAR100 50–10. (c) ImageNet100 10–10. (d) ImageNet100 50–10.

measures whether a channel is both active and used by downstream computation. A replacement action is taken only for mature low-utility channels and is initialized with zero outgoing contribution, which makes the network function-invariant at the replacement instant. The same feature stream is then mapped to a shared, position-aware context memory and a task-agnostic reference branch. The context memory is not an auxiliary attention block: it is the slowly shared reference state against which the task-specific branch is distilled. Thus, channel renewal supplies controlled plasticity, while the reference pathway supplies the constraint that makes renewal safe. Figure 1 illustrates the resulting stability throughout incremental learning.

The main contributions of this work are summarized as follows:

- We formulate CIL capacity renewal and feature retention as a coupled stability–plasticity control problem. The formulation specifies when a capacity change is allowed and what reference must remain invariant during that change.
- We instantiate this control loop with maturity-aware utility tracking and function-preserving channel replacement. The replacement rule protects immature units and recycles only channels that are both mature and persistently ineffective, without injecting an instantaneous perturbation into the downstream function.
- We introduce a shared, position-aware context reference and a task-agnostic/task-specific distillation path that constrain the representation after renewal. This interface is what couples local capacity control to global spatial stability; SCM is therefore evaluated as part of the loop, not as an independent plug-in.
- We validate the complete loop through cross-dataset comparisons and targeted ablations that separately remove the capacity controller, the reference path, and their coupling.

2 Related Work

Class-incremental learning learns a sequence of tasks with newly introduced classes while alleviating catastrophic forgetting Fu et al. (2023); Belouadah et al. (2021). Existing methods are commonly grouped into rehearsal-based, regularization-based, and dynamic architecture-based approaches.

2.1 Rehearsal

Rehearsal methods store a small set of old samples (or generate pseudo-samples) and mix them with current data during training. iCaRL Rebuffi et al. (2017) keeps exemplars and classifies by nearest mean, while GDumb Prabhu et al. (2020) retrains from scratch on a fixed memory set. Despite strong performance, rehearsal is limited by memory budgets and sensitive to exemplar quality and diversity.

2.2 Regularization

Regularization methods preserve prior knowledge by constraining updates or aligning outputs/features. EWC Kirkpatrick et al. (2017) penalizes changes to important parameters via Fisher information, and LwF Li & Hoiem (2017) uses distillation to match old outputs. Furthermore, blending such distillation techniques with a small buffer of past experiences (e.g., dark knowledge replay) has

proven to be a highly effective baseline for maintaining structural knowledge Buzzega et al. (2020). They avoid explicit storage, but can weaken under large task shifts or domain changes.

2.3 Dynamic Architecture

Dynamic architecture methods expand networks or attach task-specific modules to increase plasticity. DER Yan et al. (2021) grows the model with distillation, DyTox Douillard et al. (2022) expands transformer tokens, and TagFex Zheng et al. (2025) fuses task-agnostic and task-specific features via attention. These methods improve capacity but often add architectural complexity and computational overhead.

2.4 Neuron Resetting Techniques

Neuron resetting techniques Dohare et al. (2024), akin to dynamic sparse training Evci et al. (2020) and iterative network pruning Mallya & Lazebnik (2018), maintain plasticity by selectively re-initializing low-utility units to refresh model capacity. However, these methods are primarily tailored for reinforcement learning rather than class-incremental tasks. Lacking specific protection mechanisms, they risk prematurely deleting newly initialized neurons before they mature. Furthermore, relying solely on internal plasticity without spatial anchoring often exacerbates representation drift, undermining discriminative stability across task boundaries.

3 Methods

3.1 Problem Formulation

Class-incremental learning (CIL) addresses a classification task sequence with continuously increasing new classes. In the CIL setting, assume a task sequence $\{T_1, T_2, \ldots, T_T\}$, where each task T_t contains a training dataset D_t and a corresponding class set C_t. Each data point in task T_t consists of input $x_i^{(t)}$ and label $y_i^{(t)}$, with $(x_i^{(t)}, y_i^{(t)}) \in D_t$ and $y_i^{(t)} \in C_t$.

The model is required, when learning task T_t, to effectively utilize information from previous tasks and incrementally expand its class space with new classes. The class sets satisfy $C_t \cap C_{t'} = \emptyset$ for $t \neq t'$, in other words, classes among tasks are mutually disjoint.

In rehearsal-based CIL, a fixed-size memory M stores selected samples from previous tasks. The memory size remains constant throughout training, and samples are updated as new tasks arrive. During inference, the task identity is unknown; the model must classify among all classes seen so far. Performance is therefore evaluated under the class-incremental setting rather than a task-incremental setting. The objective is to achieve strong accuracy on new tasks while minimizing degradation on earlier tasks, ensuring that learning new classes does not cause excessive forgetting.

3.2 Overview: A Capacity–Reference Control Loop

As illustrated in Figure 2, CSPL is organized as a capacity–reference control loop rather than as two parallel add-ons. At each update, the shared ResNet-18 backbone first exposes a feature state whose channel utility and maturity are tracked by the iCBP controller (Section 3.3). If a channel is replaced, its outgoing contribution is set to zero before its input weights are reinitialized; hence the action changes available capacity while preserving the function passed to the next layer at that instant. The resulting feature map is routed through the shared Spatial Context Memory (SCM; Section 3.4), which supplies a position-aware reference that is common to all incremental stages. The task-agnostic (TA) branch retains this reference representation, whereas the current task-specific (TS) branch adapts to new classes. The L_1 distillation term transfers the TA reference to the current TS branch, closing the loop: utility determines where capacity may be renewed, and the reference determines how the renewed representation is allowed to move. Previous TS branches remain frozen during task t.

The three losses have distinct control roles. L_{cls} drives acquisition of the current classes, L_{aux} keeps the intermediate TS features discriminative, and $L_{distill}$ constrains their deviation from the TA reference. The shared SCM parameters are updated by these losses and therefore accumulate information across stages; they are not a separately trained retrieval module. This state/action

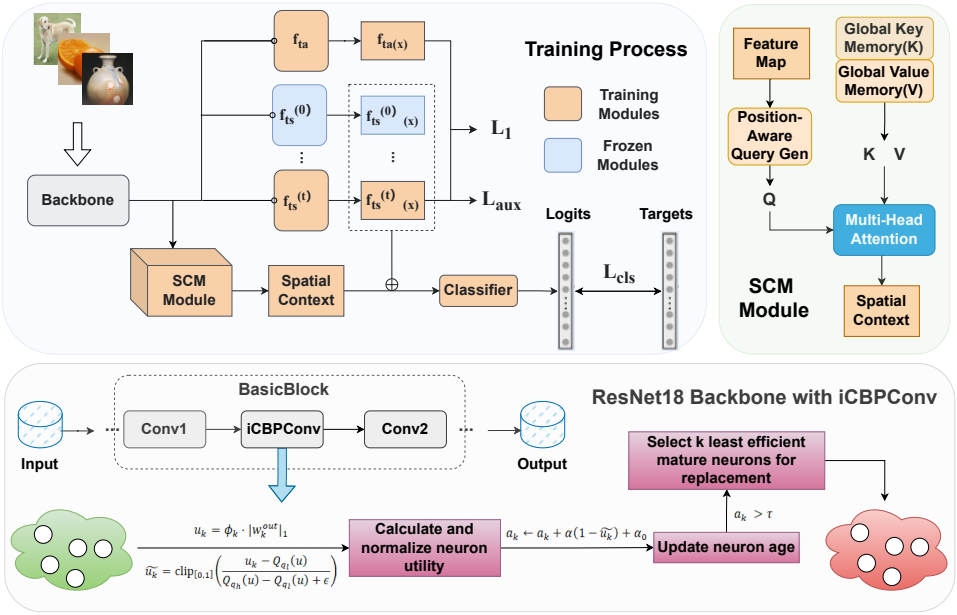


Figure 2: The CSPL capacity–reference control loop. Utility and maturity determine which channels can be renewed; zero outgoing weights make each renewal function-preserving. The renewed feature state is mapped to a shared, position-aware context reference, and the task-specific branch is constrained to this reference by $L_{distill}$ (with L_{cls} and L_{aux} providing adaptation and discrimination).

organization is the central design of CSPL and is used consistently in the analysis and ablations below.

Abstractly, one CSPL update can be written as a state transition

$$\big(F_t, a_t, r_t, K_t, V_t\big) \xrightarrow{u_t} \big(\theta_{t+1}, F_{t+1}, a_{t+1}, r_{t+1}, K_{t+1}, V_{t+1}\big), \tag{1}$$

where u_t determines which mature channels are eligible for renewal, the reset rule preserves the downstream function at the transition, and (K_t, V_t) provides the reference used by $L_{distill}$. This notation makes the coupling explicit: the capacity state and the reference state are updated in the same training loop and are evaluated jointly, rather than being optimized as independent modules.

3.3 Incremental Continuous Backpropagation

The capacity controller in CSPL is an incremental continuous backpropagation (iCBP) rule. It is not intended as an independent second model; it is the state/action component of the capacity–reference loop defined in Section 3.2. Building upon continuous backpropagation Dohare et al. (2024), the rule maintains representational diversity by selectively resetting neurons that contribute little to the network's performance. Its three stages—utility evaluation, maturity tracking, and selective replacement—are designed around the interface with the reference pathway: a replacement must be mature, persistently ineffective, and function-preserving at the moment it is made.

To ensure clarity, we define the key symbols used in the iCBP formulation in Table 1.

Utility Evaluation We define the "utility" of a neuron based on the intuition that a valuable feature must satisfy two conditions: it must be active for the current data distribution, and it must be utilized by downstream layers.

Formally, let $f_i(x)$ denote the activation of the i-th channel. We maintain a running estimate of the activation magnitude F_i to smooth out batch-wise variance. Given decay rate β, the update rule per step is:

$$F_i^{(t)} \leftarrow \beta F_i^{(t-1)} + (1-\beta)\mathbb{E}_{x,h,w}[|f_i(x)|], \tag{2}$$

Table 1: Notation used in the iCBP algorithm.

Symbol	Description
$f_i(x)$	Activation map of the i-th neuron/channel for input x
F_i	Exponentially smoothed activation cache for neuron i
ϕ_i	Mean activation magnitude of neuron i
$\|w_i^{\text{out}}\|_1$	L_1 norm of downstream weights connected to neuron i
$u_i, \tilde{u}_i$	Raw and quantile-normalized utility of neuron i
a_i	Age (maturity) of neuron i
τ	Maturity threshold for replacement eligibility
ρ, r	Replacement rate and accumulated fractional reset budget

If $\beta = 0$, we simply use the instantaneous mean $F_i = \mathbb{E}_{x,h,w}[|f_i(x)|]$.

The downstream weight magnitude $\|w_i^{\text{out}}\|_1$ is defined depending on the layer type. For a convolutional layer with kernel size $K \times K$, it is the mean absolute weight over output channels C_{out} and spatial kernels:

$$\|w_i^{\text{out}}\|_1 = \frac{1}{C_{out}K^2} \sum_{o=1}^{C_{out}} \sum_{p,q=1}^{K} |W_{o,i,p,q}|, \tag{3}$$

For a linear layer, it is the mean absolute weight over output units. The final neuron utility is:

$$u_i = F_i \cdot \|w_i^{\text{out}}\|_1, \tag{4}$$

Maturity Tracking. Directly resetting neurons based on instantaneous utility is unstable, as newly initialized neurons naturally have low utility. To prevent a reset loop where new neurons are killed before they can learn, we introduce a maturity mechanism.

First, to handle varying utility scales across layers, we normalize utilities to $[0, 1]$ using quantile scaling:

$$\tilde{u}_i = \text{clip}_{[0,1]} \left(\frac{u_i - Q_{q_\ell}(u)}{Q_{q_h}(u) - Q_{q_\ell}(u) + \varepsilon} \right), \tag{5}$$

where $Q_q(u)$ denotes the q-th quantile of the utility distribution in the layer.

Each neuron maintains an age parameter a_i. The aging process is adaptive:

$$a_i \leftarrow a_i + \alpha(1 - \tilde{u}_i) + \alpha_0, \tag{6}$$

where α (age_rate) accelerates aging for low-utility neurons, and α_0 (age_floor) ensures a baseline aging speed. A neuron is added to the mature set $\mathcal{M}$ only if its age exceeds a maturity threshold τ:

$$\mathcal{M} = \{i \mid a_i > \tau\}, \tag{7}$$

Selective Replacement. To ensure training stability, we strictly control the number of reset neurons using a fractional budget counter r. Given a replacement rate ρ:

$$r \leftarrow r + \rho|\mathcal{M}|, \quad k = \lfloor r \rfloor, \quad r \leftarrow r - k, \tag{8}$$

If $k > 0$, we select the k neurons with the *lowest* utility from $\mathcal{M}$.

For each selected neuron i, we perform a full reset to restore plasticity. The input weights are re-initialized using Kaiming Uniform initialization to preserve activation variance:

$$W_{i,:}^{\text{in}} \sim \mathcal{U}(-b, b), \quad b = \text{gain} \times \sqrt{\frac{3}{\text{fan_in}}}, \tag{9}$$

where gain $= \sqrt{2}$ for ReLU networks, and fan_in $= C_{\text{in}} \times K_h \times K_w$.

Crucially, the output weights are zeroed out:

$$W_{:,i}^{\text{out}} \leftarrow 0, \quad b_i^{\text{in}} \leftarrow 0, \quad a_i \leftarrow 0, \tag{10}$$

If the neuron is followed by Batch Normalization (BN), its running statistics and affine parameters are also reset: $\gamma_i \leftarrow 1, \beta_i \leftarrow 0, \mu_i \leftarrow 0, \sigma_i^2 \leftarrow 1$.

By setting the outgoing weights $W_{:,i}^{\text{out}}$ to zero, the reset operation is function-invariant at the moment of reset: $f(x;\theta_t) = f(x;\theta_{t+1})$, because the reset neuron contributes zero to the next layer. This is the interface that makes capacity renewal compatible with the stabilizing reference pathway. No random activation is injected into the downstream computation; the new channel can instead acquire utility gradually through gradient descent and will only become eligible for replacement after its age has matured.

The iCBP algorithm is integrated into ResNet-18 by replacing standard convolutional layers within BasicBlocks with iCBPConv layers. The input x is processed by Conv1-BN-ReLU to produce h_1, which then passes through the iCBP layer. The iCBP layer dynamically manages the kernels, producing an adjusted feature h_1' that feeds into Conv2-BN. Finally, the residual connection adds x to the output of Conv2-BN, followed by ReLU. This design enables continuous, controlled kernel resets, improving both efficiency and generalization in incremental learning.

3.4 Spatial Context Memory as the reference state

The second state of the control loop is a spatial reference. Channel renewal can free capacity, but without a reference it can also change the coordinates in which old and new classes are represented. We therefore use the Spatial Context Memory (SCM) module to provide a shared reference state for the feature stream. SCM is not added after training as an independent attention block: its output is consumed by both branches and its parameters are updated by the same objective that controls renewal.

In a standard incremental setup, backbone parameters change continually, causing the semantic meaning of feature maps to drift. SCM maintains a global set of key–value pairs that capture representative spatial patterns. The memory is shared across all tasks and is updated continuously, so it acts as the slowly accumulated reference state of the loop rather than a task-specific cache. Because the channel replacement rule is function-preserving, the reference does not have to absorb an artificial jump at a reset; it can adapt to genuinely useful new evidence.

The SCM module operates on the feature map $F \in \mathbb{R}^{H\times W\times C}$ produced by the backbone. It consists of three steps: Query Generation, Memory Retrieval, and Context Injection.

We define a learnable global memory consisting of a Key matrix $K \in \mathbb{R}^{M\times d}$ and a Value matrix $V \in \mathbb{R}^{M\times d}$. In our implementation, we set the memory size $M = 512$ and the context dimension $d = 256$. These matrices are randomly initialized at task $t = 0$. Crucially, to ensure the accumulation of task-agnostic patterns, K and V are shared across all tasks and are never frozen. They are continuously updated via backpropagation based on the gradients from the classification and distillation losses, allowing the memory to gradually capture universal spatial structures spanning all seen classes.

To incorporate spatial structure, we generate a specific query q_l for each spatial location $l \in \{(h,w) \mid 1 \le h \le H, 1 \le w \le W\}$. We introduce a learnable spatial position embedding $p_l \in \mathbb{R}^C$ for each grid location, initialized randomly and optimized during training. The query combines the local feature $f_l \in \mathbb{R}^C$ with this position embedding:

$$q_l = W_q \cdot (f_l + p_l), \tag{11}$$

where $W_q \in \mathbb{R}^{d\times C}$ is a projection matrix. The explicit addition of p_l ensures that the retrieval process is position-aware, allowing the model to distinguish between contexts appropriate for different image regions.

We retrieve relevant contexts using a Multi-Head Attention mechanism with $N_h = 8$ heads to capture diverse contextual patterns. For each head $j \in \{1,\dots,N_h\}$, we first project the query vector q_l and the global memory matrices K, V into a subspace of dimension $d_k = d/N_h = 32$:

$$q_l^{(j)} = W_Q^{(j)} q_l, \quad K^{(j)} = K(W_K^{(j)})^T, \quad V^{(j)} = V(W_V^{(j)})^T, \tag{12}$$

where $W_Q^{(j)}, W_K^{(j)}, W_V^{(j)} \in \mathbb{R}^{d_k\times d}$ are learnable projection matrices, and $K, V \in \mathbb{R}^{M\times d}$.

We then compute the attention scores $A_l^{(j)} \in \mathbb{R}^M$ over all memory slots and retrieve the subspace context vector $c_l^{(j)} \in \mathbb{R}^{d_k}$:

$$A_l^{(j)} = \text{softmax}\left(\frac{K^{(j)} q_l^{(j)}}{\sqrt{d_k}}\right), \tag{13}$$

$$c_l^{(j)} = (V^{(j)})^T A_l^{(j)}, \tag{14}$$

The final spatial context vector c_l is obtained by concatenating the outputs of all heads and projecting them:

$$c_l = W_o \cdot \text{Concat}(c_l^{(1)}, \ldots, c_l^{(N_h)}), \tag{15}$$

where $W_o \in \mathbb{R}^{d \times d}$ is the output projection matrix.

Finally, the retrieved context c_l is concatenated with the original feature f_l to form the augmented feature f_l':

$$f_l' = \text{Concat}([f_l, c_l]), \tag{16}$$

We employ concatenation rather than addition to explicitly preserve the original backbone features while providing the classifier with supplementary stability cues.

Distinct from standard self-attention or task-conditional prompts, SCM is used here as the reference state in a CIL control loop. A single shared memory and position-aware queries make the reference comparable across stages, while the TA–TS distillation term constrains the adaptive branch to remain in that reference coordinate system. The resulting division is causal: iCBP decides where capacity can be renewed, SCM and the TA branch define the admissible feature reference, and distillation feeds that reference back into the update. This is the mechanism by which CSPL couples plasticity and stability; neither operation is evaluated as a standalone plug-in.

3.5 Training Objective

The objective closes the capacity–reference loop defined above. Capacity renewal is useful only when the adaptive branch remains in the coordinate system accumulated by the shared reference; conversely, the reference must remain discriminative while new classes are learned. The three terms below therefore have control roles rather than being independent regularizers: classification drives acquisition, the auxiliary classifier preserves local discrimination, and feature distillation constrains the post-renewal representation to the task-agnostic reference. For incremental tasks, the distillation loss encourages consistency between the teacher model"s task-agnostic features f_{ta} and the model-extracted task-specific features $\{f_{ts}^i\}$. The distillation process employs the L_1 loss defined as:

$$L_{\text{distill}} = \left\| f_{ta} - f_{ts}^{\text{last}} \right\|_1, \tag{17}$$

where f_{ts}^{last} denotes the feature vector from the last task-specific network corresponding to the latest subtask. The total loss function is given by:

$$L = L_{\text{cls}} + \lambda_{\text{aux}} L_{\text{aux}} + \lambda_{\text{distill}} L_{\text{distill}}. \tag{18}$$

where L_{cls} and L_{aux} represent the cross-entropy loss and auxiliary classifier loss, respectively, and λ_{aux}, λ_{distill} are hyperparameters controlling the contributions of the auxiliary classifier loss and distillation loss.

4 Experiment

4.1 Implementation Details

Datasets and Evaluation Metrics We evaluate the proposed framework on four standard benchmarks: CIFAR100 Krizhevsky & Hinton (2009), ImageNet100 Russakovsky et al. (2015), TinyImageNet Le & Yang (2015), and the fine-grained CUB-200 Wah et al. (2011). Following standard

Table 2: Comparison with state-of-the-art methods on CIFAR100 and ImageNet100. Results are reported as the average of 3 independent runs with different random seeds. Bold font represents the best results.

Methods	Year	CIFAR100				ImageNet100			
		10–10		50–10		10–10		50–10	
		Last	Avg	Last	Avg	Last	Avg	Last	Avg
FineTune	–	8.18	23.11	8.13	20.68	9.46	27.08	9.42	24.68
iCaRL Rebuffi et al. (2017)	2017	49.42	64.67	50.51	60.17	51.29	67.02	53.65	62.32
BiC Wu et al. (2019)	2019	50.46	65.44	44.00	57.07	41.89	65.14	49.63	66.27
UCIR Hou et al. (2019)	2019	-	-	57.06	66.92	-	-	50.81	64.96
WA Zhao et al. (2020)	2020	52.08	66.78	55.90	64.94	55.28	68.90	56.72	65.96
PODnet Douillard et al. (2020)	2020	53.38	70.39	55.73	66.84	45.62	63.98	62.98	73.81
PASS Zhu et al. (2021)	2021	-	-	55.84	63.97	-	-	56.27	69.16
DER Yan et al. (2021)	2021	64.30	75.41	65.50	72.73	66.90	77.29	71.06	77.50
DyTox Douillard et al. (2022)	2022	60.43	73.77	-	-	67.66	76.59	-	-
FOSTER Wang et al. (2022)	2022	62.55	72.80	64.01	69.99	70.07	78.89	75.39	80.13
SSRE Zhu et al. (2022)	2022	-	-	56.97	65.48	-	-	60.19	70.05
BEEF Wang et al. (2023)	2023	60.97	71.98	63.59	70.81	68.91	77.56	70.97	77.43
TASS Liu et al. (2024)	2024	-	-	59.44	68.71	-	-	63.78	74.40
PASS++ Zhu et al. (2025)	2025	-	-	59.85	68.90	-	-	63.92	74.13
DCNet Wang et al. (2025)	2025	65.32	75.86	-	-	67.45	76.93	-	-
TagFex Zheng et al. (2025)	2025	68.17	78.37	70.12	75.66	70.78	79.34	75.60	80.85
CSPL	-	**71.05**	**79.06**	**72.28**	**77.71**	**76.09**	**81.63**	**79.11**	**82.77**

class-incremental learning protocols, we denote settings as B-I, where B is the number of base classes and I is the number of new classes per task. For the 100-class datasets (CIFAR100, ImageNet100), we use the 50-10 and 10-10 splits. For the 200-class datasets (TinyImageNet), we evaluate on 100-20, 100-10, and 100-5 settings. We report the Last Accuracy after the final task McCloskey & Cohen (1989) and the Average Accuracy across all phases Lopez-Paz & Ranzato (2017).

Training Details ResNet18 with iCBPConv is adopted as the backbone for CIFAR100, TinyImageNet, ImageNet100 and CUB200. The base training stage runs for 200 epochs; incremental stages run for 170 epochs each. Training uses SGD with momentum; the base stage employs an initial learning rate of 0.1, momentum 0.9, and weight decay 5e-4, with learning rate decayed by 0.1 at epochs 60, 120, and 170. Incremental stages use learning rate 0.1, momentum 0.9, weight decay 2e-4, with decay at epochs 80, 120, and 150. The memory size is fixed globally across all tasks, with a total of 2000 rehearsal samples, rather than allocating 2000 samples per task. Rehearsal samples are selected and updated using herding Rebuffi et al. (2017) to maintain a constant total memory size throughout incremental training.

4.2 Comparison With State-of-the-Art Methods

We test our proposed method in various offline class-incremental learning scenarios on CIFAR100, TinyImageNet, ImageNet100, and CUB200.

Results on CIFAR-100 and ImageNet100 We comprehensively compare our method with state-of-the-art class-incremental learning baselines, including iCaRL Rebuffi et al. (2017), BiC Wu et al. (2019), UCIR Hou et al. (2019), WA Zhao et al. (2020), PODnet Douillard et al. (2020), PASS Zhu et al. (2021), DER Yan et al. (2021), DyTox Douillard et al. (2022), FOSTER Wang et al. (2022), SSRE Zhu et al. (2022), BEEF Wang et al. (2023), TASS Liu et al. (2024), Pass++ Zhu et al. (2025), DCNet Wang et al. (2025), and TagFex Zheng et al. (2025). We also include a FineTune setting, which continues training on new data without anti-forgetting mechanisms. Experiments are conducted on CIFAR100 and ImageNet100 under the 10-10 and 50-10 protocols. As shown in Table 2, our method achieves the best accuracy across all settings on both datasets. DER Yan et al. (2021) mitigates forgetting by freezing previous parameters and adding task-specific extractors, but incurs unbounded model growth and increasing inference cost. TagFex Zheng et al. (2025) reduces feature interference by decoupling task-agnostic and task-specific representations via merge attention, yet its separated components limit efficient coupling between global context and local adaptation.

Table 3: Ablation study on CIFAR100. The first row shows the baseline dual-branch model without CSPL components.

L_1 Loss	SCM	iCBP	Last	Avg
			67.23	73.66
		✓	69.15	75.21
	✓		69.49	75.34
✓			70.34	75.89
	✓	✓	71.01	76.28
✓		✓	71.42	76.38
✓	✓		71.68	76.50
✓	✓	✓	72.28	77.71

In contrast, CSPL attains superior performance with a fixed-capacity backbone because the utility controller and reference pathway are updated under the same feature constraint. Specifically, renewal is admitted only for mature low-utility neurons, and the resulting representation is evaluated against the shared reference through distillation. This closed-loop design yields consistent gains over both DER and TagFex.

4.3 Ablation Studies

We conduct CIFAR100 ablations under the 50–10 protocol to test the proposed control loop rather than to count gains from independent add-ons. Table 3 starts from the dual-branch learner and then removes, in turn, the reference constraint (L_1 distillation), the shared reference state (SCM), and the capacity controller (iCBP). The relevant comparison is whether capacity renewal remains effective when its reference is removed, and whether the reference remains adaptive when capacity renewal is disabled. The full loop performs best: iCBP supplies controlled capacity renewal, SCM supplies a cross-stage reference, and distillation transmits that reference to the adapting branch. The ablation is evidence for the proposed interaction, not merely for the isolated usefulness of three modules.

5 Conclusion

In this paper, we proposed Closing the Stability–Plasticity Loop (CSPL), a capacity–reference control loop for class-incremental learning. The contribution is not a concatenation of a reset rule and an attention block: utility and maturity decide when capacity may be renewed, function-preserving replacement keeps the downstream function continuous, and a shared spatial reference together with TA–TS distillation constrains how the renewed representation can evolve. Across CIFAR100, TinyImageNet, ImageNet100, and CUB-200, the complete loop consistently improves Last and Average accuracy over the compared methods. The ablations further show that the gain depends on the interaction between capacity renewal and the reference path.

Limitations and Future Work. While CSPL is robust, its current implementation is specifically tailored for convolutional architectures, and extending the neuron-level utility mechanisms to token-based models like Vision Transformers requires further adaptation. The method also depends on hyperparameters such as the maturity threshold (τ), which may need retuning across substantially different domains. Future work will explore adapting iCBP for Transformer backbones and extending CSPL to broader settings, such as online continual learning.

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pp. 139–154, 2018.

Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Netw.*, 135:38–54, 2021. doi: 10.1016/j.neunet.2020.12.007.

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.

Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International conference on computer vision*, pp. 9516–9525, 2021.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024. doi: 10.1038/s41586-024-07711-7.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. PODNet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pp. 86–102, 2020.

Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. DyTox: Transformers for continual learning with dynamic token expansion. In *CVPR*, pp. 9275–9285, 2022. doi: 10.1109/CVPR52688.2022.00907.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *ICML*, pp. 2943–2952, 2020.

Zhu Fei, Xu-Yao Zhang, and Cheng-Lin Liu. Class incremental learning: A review and performance evaluation. *Acta Automatica Sinica*, 49(3):1–26, 2023.

Zhiling Fu, Zhe Wang, Xinlei Xu, Mengping Yang, Ziqiu Chi, and Weichao Ding. Semantic alignment with self-supervision for class incremental learning. *Knosys*, 282:111114, 2023.

Tyler L. Hayes, Nathan D. Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *ICRA*, pp. 9769–9776, 2019. doi: 10.1109/ICRA.2019.8793995.

Run He, Di Fang, Yizhu Chen, Kai Tong, Cen Chen, Yi Wang, Lap-pui Chau, and Huiping Zhuang. REAL: Representation enhanced analytic learning for exemplar-free class-incremental learning. *Knosys*, pp. 114901, 2025.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pp. 831–839, 2019. doi: 10.1109/CVPR.2019.00092.

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. *PNAS*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann Le and Xuan Yang. Tiny ImageNet visual recognition challenge, 2015. CS231N course project.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017. doi: 10.1109/TPAMI.2017.2773081.

Xialei Liu, Jiang-Tian Zhai, Andrew D. Bagdanov, Ke Li, and Ming-Ming Cheng. Task-adaptive saliency guidance for exemplar-free class incremental learning. In *CVPR*, pp. 23954–23963, 2024. doi: 10.1109/CVPR52733.2024.02261.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, volume 30, 2017.

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.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Martin Mundt, Iuliia Pliushch, Sagnik Majumder, Yongwon Hong, and Visvanathan Ramesh. Unified probabilistic deep continual learning through generative replay and open set recognition. *J. Imaging*, 8(4):93, 2022. doi: 10.3390/jimaging8040093.

Oleksiy Ostapenko, Timothée Lesort, Pau Rodriguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *CoLLAs*, pp. 60–91, 2022.

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. GDumb: A simple approach that questions our progress in continual learning. In *ECCV*, pp. 524–540, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, pp. 5533–5542, 2017. doi: 10.1109/CVPR.2017.587.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995. doi: 10.1080/09540099508915720.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. FOSTER: Feature boosting and compression for class-incremental learning. In *ECCV*, pp. 398–414, 2022.

Fu-Yun Wang, Da-Wei Zhou, Liu Liu, Han-Jia Ye, Yatao Bian, De-Chuan Zhan, and Peilin Zhao. BEEF: Bi-compatible class-incremental learning via energy-based expansion and fusion. In *ICLR*, 2023.

Tianqi Wang, Jingcai Guo, Depeng Li, and Zhi Chen. On the discrimination and consistency for exemplar-free class incremental learning. *arXiv preprint arXiv:2501.15454*, 2025.

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

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pp. 374–382, 2019. doi: 10.1109/CVPR.2019.00046.

Shipeng Yan, Jiangwei Xie, and Xuming He. DER: Dynamically expandable representation for class incremental learning. In *CVPR*, pp. 3013–3022, 2021. doi: 10.1109/CVPR46437.2021.00303.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pp. 13205–13214, 2020. doi: 10.1109/CVPR42600.2020.01322.

Bowen Zheng, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Task-agnostic guided feature expansion for class-incremental learning. In *CVPR*, pp. 10099–10109, 2025. doi: 10.1109/CVPR52734.2025.00944.

Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, pp. 5867–5876, 2021. doi: 10.1109/CVPR46437.2021.00581.

Fei Zhu, Xu-Yao Zhang, Zhen Cheng, and Cheng-Lin Liu. PASS++: A dual bias reduction framework for non-exemplar class-incremental learning. *TPAMI*, 47(8):7123–7139, 2025. doi: 10.1109/TPAMI.2025.3568886.

Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *CVPR*, pp. 9286–9295, 2022. doi: 10.1109/CVPR52688.2022.00908.