Expandable and Differentiable Dual Memories with Orthogonal Regularization for Exemplar-free Continual Learning

Hyung-Jun Moon[1] **and Sung-Bae Cho**[2]

[1]Department of Artificial Intelligence, Yonsei University
[2]Department of Computer Science, Yonsei University
axtabio@yonsei.ac.kr, sbcho@yonsei.ac.kr

Abstract

Continual learning methods used to force neural networks to process sequential tasks in isolation, preventing them from leveraging useful inter-task relationships and causing them to repeatedly relearn similar features or overly differentiate them. To address this problem, we propose a fully differentiable, exemplar-free expandable method composed of two complementary memories: One learns common features that can be used across all tasks, and the other combines the shared features to learn discriminative characteristics unique to each sample. Both memories are differentiable so that the network can autonomously learn latent representations for each sample. For each task, the memory adjustment module adaptively prunes critical slots and minimally expands capacity to accommodate new concepts, and orthogonal regularization enforces geometric separation between preserved and newly learned memory components to prevent interference. Experiments on CIFAR-10, CIFAR-100, and Tiny-ImageNet show that the proposed method outperforms 14 state-of-the-art methods for class-incremental learning, achieving final accuracies of 55.13%, 37.24%, and 30.11%, respectively. Additional analysis confirms that, through effective integration and utilization of knowledge, the proposed method can increase average performance across sequential tasks, and it produces feature extraction results closest to the upper bound, thus establishing a new milestone in continual learning.

Code — https://github.com/axtabio/EDD
Extended version — http://arxiv.org/abs/2511.09871

Introduction

Continual learning (CL) aims to learn a sequence of tasks while maintaining performance on previous tasks, but catastrophic forgetting (CF) remains a fundamental challenge (French 1995). This problem is exacerbated in exemplar-free settings, since without access to past examples, new training can override internal representations of prior knowledge or require sacrificing plasticity to preserve prior knowledge, resulting in degraded performance on previous tasks (Goswami et al. 2024; Moon and Cho 2025a).

Previous exemplar-free methods only partially mitigate forgetting. Regularization-based methods constrain parameter updates (e.g., penalties or knowledge distillation) to protect previously learned tasks (Sun, Mu, and Hua 2023; Tang et al. 2021). Methods based on dynamic architecture expansion or parameter isolation allocate dedicated neurons or layers for each new task and isolate them to prevent interference (Douillard et al. 2022; Ye and Bors 2023). However, these approaches either impose restrictive penalties that limit model plasticity to avoid interference with prior knowledge or lead to uncontrolled model growth, which is problematic in realistic CL scenarios (Arani, Sarfraz, and Zonooz 2022). A more significant problem is that they ignore relationships between tasks and fail to leverage useful weights or patterns acquired from previous tasks. In other words, they treat future tasks as completely independent of previously acquired knowledge, thereby hampering reuse of knowledge shared across tasks (Pham, Liu, and Hoi 2021; Moon and Cho 2025b).

In this paper, we propose a novel expandable, differen-

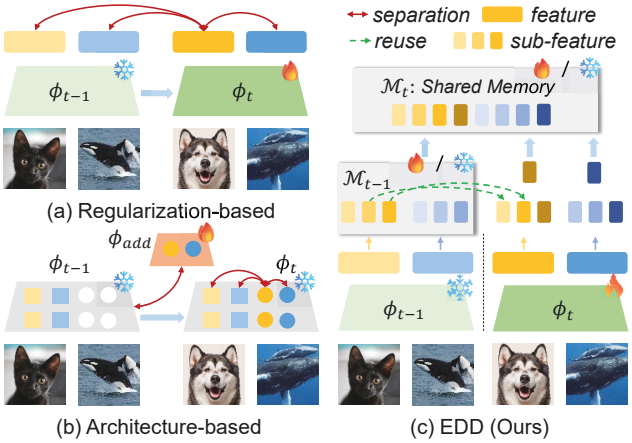

Figure 1: Comparison of the proposed method with regularization- and architecture-based approaches. (a) Regularization enforces new classes not to interfere with previous tasks. (b) Architecture-based methods freeze the parameters allocated to segregating past classes and isolate them from expanded parameters. (c) Our method encourages maximal mutual utilization of past and new knowledge rather than separation.

tiable dual memory (EDD) method to overcome these limitations. As illustrated in Figure 1(c), EDD decomposes past data features into small sub-features and stores them in memory, enabling the extraction of diverse patterns that may overlap with future incoming data. When new data relate to any stored sub-features, they are retrieved and reused to leverage prior knowledge. Simultaneously, sub-features containing information independent of past knowledge are newly stored for use in subsequent tasks. Unlike the approaches in Figures 1(a) and 1(b) that penalize newly learned information to preserve previously acquired knowledge, EDD reuses past knowledge to acquire knowledge about new data. Instead, it maximizes learning across all tasks by directly decomposing and reusing the intrinsic information of the data. To facilitate feature decomposition and knowledge reuse, the memories are fully differentiable, and to address capacity limitations as tasks continuously arrive, the method is designed to expand its capacity. In addition, inspired by complementary learning systems theory, the model incorporates two memories: one for shared knowledge and the other for task-specific knowledge. The shared memory encodes transferable representations that generalize for all tasks, while the task-specific memory, based on the shared knowledge, captures fine-grained discriminative features unique to each task.

In experiments on standard CL benchmarks, EDD outperforms 14 recent state-of-the-art methods. In particular, despite using without buffer, it surpasses approaches that employ buffers, demonstrating its superiority. Furthermore, as task sequences increase in complexity and length, it achieves relative improvements exceeding 26% over the previous SOTA. Compared to the upper bound joint-training strategy (i.e., training all tasks simultaneously), EDD's per-class feature representations most closely resemble those of the joint model, closer than those of any other contemporary methods, confirming its effectiveness at inter-task knowledge transfer.

Related Works

Continual Learning

Regularization-based methods aim to preserve prior knowledge by penalizing parameter changes or matching previous outputs. TwF (Boschini et al. 2022) is a hybrid method built on a frozen pretrained sibling network that continuously propagates source-domain knowledge through a layer-wise loss term. FDR (Benjamin, Rolnick, and Kording 2018) stores a tiny set of past samples and penalizes deviations in output logits on those exemplars, effectively distilling the old model's function without full rehearsal. These methods avoid large buffers but still require at least some exemplars or held-out splits and impose global constraints that can over-restrict learning.

Dynamic expansion approaches allocate new parameters for each task to avoid interference by design. PNN (Rusu et al. 2016) grows a frozen column of weights per task and add lateral connections for feature reuse, eliminating forgetting but leading to unbounded model growth. DEN (Yoon et al. 2017) selectively adds and prunes neurons based on task complexity, controlling expansion yet demanding careful pruning schedules. Parameter isolation methods constitute a mainstream approach in CL, where the upper layers of a previously trained neural network are frozen and only the remaining parameters are trained. PEC (Pernici et al. 2021) is a fixed-classifier method that pre-allocates multiple output nodes from the beginning of training and applies the classification loss to each of them. Dynamic expansion suffers from unbounded model growth and parameter proliferation, while parameter isolation limits plasticity and hinders knowledge reuse across tasks.

Dual Memory Approaches for Continual Learning

Several CLS-based CL methods mimic human cognitive and memory processes by configuring two complementary modules. DualNets (Pham, Liu, and Hoi 2021) uses an episodic buffer to train complementary fast and slow networks for supervised and self-supervised learning, achieving a plasticity–stability trade-off. CLS-ER (Arani, Sarfraz, and Zonooz 2022) maintains short-term and long-term semantic memories alongside an episodic buffer to align decision boundaries, but they tend to add learning rate adjustment loss between models or require additional self-supervised learning objectives rather than directly extracting knowledge shared across tasks. ICL (Qi et al. 2024) couples a vision transformer "fast thinker" with a frozen large language model "slow thinker" via attention and vMF-based routing, obviating exemplar storage but relying on heavyweight external models and a fixed backbone.

Rather than imposing rigid data- or class-based constraints or relying on external buffers and separate models, EDD integrates all knowledge end-to-end through a self-organizing memory. It decomposes inputs into sub-features that are stored in a fully differentiable memory, expanding to accommodate new tasks and pruning redundancies as needed. In doing so, it naturally extracts and shares inter-task information and resolves capacity limitations without any explicit analysis of the network itself or external dependencies.

Proposed Method

Differentiable Dual Memory

This paper addresses a class-incremental learning (CIL) scenario defined by a sequence of N tasks $\mathcal{T}_1, \ldots, \mathcal{T}_N$. Each task $\mathcal{T}_t$ introduces a dataset containing classes disjoint from those in previous tasks. The model F comprises a feature encoder E (e.g., a ResNet backbone) and a classifier C, producing predictions $y = C(E(x))$. Within the encoder, two complementary memories are used at intermediate layers, each placed after specific encoder blocks, and trained end-to-end with the rest of the network. The shared memory M^s captures representations that can be reused across tasks, while the task-specific memory M^t encodes features unique to each task. This design ensures that, upon each new input, the model can access its memory and retrieve relevant existing information. Figure 2 shows the overall architecture of EDD and its key components.

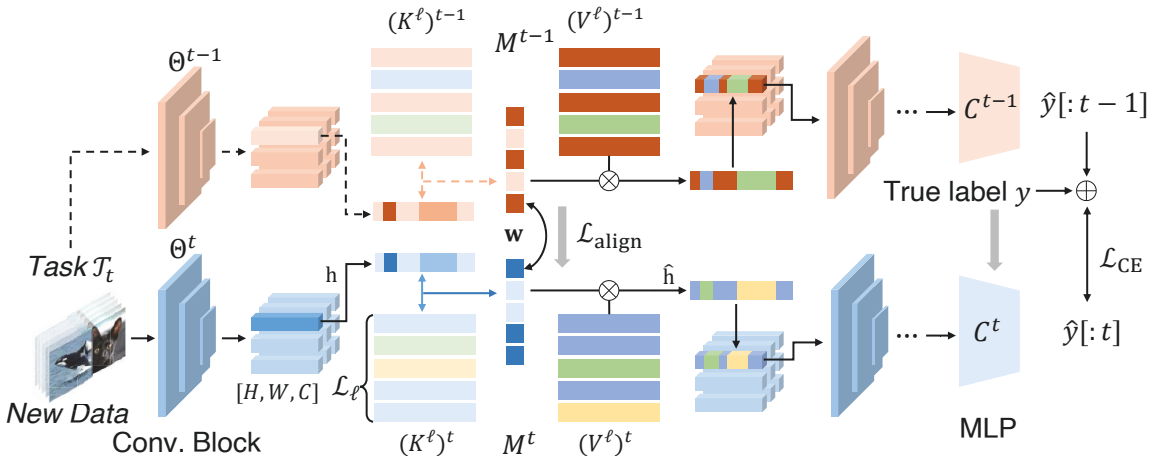

Figure 2: Overview of the proposed method.

Each memory M^ℓ (for $\ell \in s, t$) is a differentiable key–value memory with L_ℓ learnable slots. Let the memory keys K^ℓ be defined as

$$K^\ell = \left[\mathbf{k}_1^\ell, \ldots, \mathbf{k}_{L_\ell}^\ell\right]^\top \in \mathbb{R}^{L_\ell \times d}. \tag{1}$$

Similarly, the memory values V^ℓ are defined as

$$V^\ell = \left[\mathbf{v}_1^\ell, \ldots, \mathbf{v}_{L_\ell}^\ell\right]^\top \in \mathbb{R}^{L_\ell \times d'}. \tag{2}$$

Given an input's intermediate feature map $\mathcal{H} \in \mathbb{R}^{C \times H \times W}$ from the encoder, each spatial feature vector $\mathbf{h} \in \mathbb{R}^d$ (extracted from $\mathcal{H}$; e.g., $d = H \times W$ in Figure 2) serves as a query to the memory. The memory-read is computed via cosine-similarity attention over the keys. For each memory slot j, an attention weight is obtained as

$$w_j = \frac{\exp\left(\langle \mathbf{k}_j^\ell, \mathbf{h} \rangle\right)}{\sum_{i=1}^{L_\ell} \exp\left(\langle \mathbf{k}_i^\ell, \mathbf{h} \rangle\right)}, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product (assuming keys and queries are ℓ_2-normalized). The memory output is then the weighted combination of value vectors as

$$\hat{\mathbf{h}} = \sum_{j=1}^{L_\ell} w_j \, \mathbf{v}_j^\ell, \tag{4}$$

which has the same dimensionality as $\mathbf{h}$. The memory-guided feature $\hat{\mathbf{h}}$ is forwarded on its own to subsequent layers. The memory parameters (K^ℓ, V^ℓ) are jointly optimized with E and C via gradient descent, enabling the model to autonomously encode useful representations into these slots. In the next section, we will describe in detail how EDD extracts patterns from the data and manage the memory across tasks to balance stability and plasticity. Specifically, we cover expansion and pruning for memory consolidation, orthogonal regularization for feature disentanglement, and memory alignment for representation integration.

Memory Expansion with Knowledge Pruning

To accommodate new task information without unbounded incremental growth or forgetting, each memory self-organizes by pruning (freezing) critical slots and expanding with new slots at the end of each task. Pruning preserves knowledge by identifying memory slots that have captured the task's key representations and freezing them so that their weights no longer update in subsequent tasks. During the expansion phase, the same number of new slots as those frozen are allocated to ensure sufficient capacity (plasticity) for the next task. This strategy guarantees that previously learned features are retained (stability) while the model's capacity to learn new features grows only as needed (plasticity).

Formally, let $(K_j^\ell)^{(t-1)}$ be the key vector at the end of task $t-1$ (similarly for V). After training on task t, an importance score Δ_j^ℓ for each slot is computed as the total change in its parameters as

$$\Delta_j^\ell = |K_j^\ell - (K_j^\ell)^{(t-1)}|_2 + |V_j^\ell - (V_j^\ell)^{(t-1)}|_2. \tag{5}$$

A fraction of slots with the largest changes (i.e., most utilized to learn task t) are then pruned from further updates, which are denoted by $\mathcal{F}_t^\ell$ the set of indices of these newly frozen slots in memory M^ℓ. We choose the number of frozen slots proportional to the task's share of classes as

$$|\mathcal{F}_t^\ell| \approx \frac{|\mathcal{C}_t|}{|\mathcal{C}_{1:t}|} \cdot \text{unfrozen slots}, \tag{6}$$

ensuring that memory grows in pace with the diversity of learned classes. Once frozen, these slots contribute to inference but receive no gradient updates in subsequent tasks.

After pruning, the memory undergoes expansion to introduce new slots for the next task's learning. $|\mathcal{F}_t^\ell|$ new slots are added to memory M^ℓ. The new keys K_{new}^ℓ and values V_{new}^ℓ are initialized (e.g., with small random vectors), and they form the active part of the memory for learning task $t+1$. The total number of slots L_ℓ thus increases modestly over time, but through expansion and pruning, growth remains controlled. This memory adjustment yields a complementary memory system: older slots (frozen) specialize in previous tasks and are shielded from interference, while new slots (trainable) freely adapt to encode novel concepts. Figure 3(a) shows the expansion–pruning scheme.

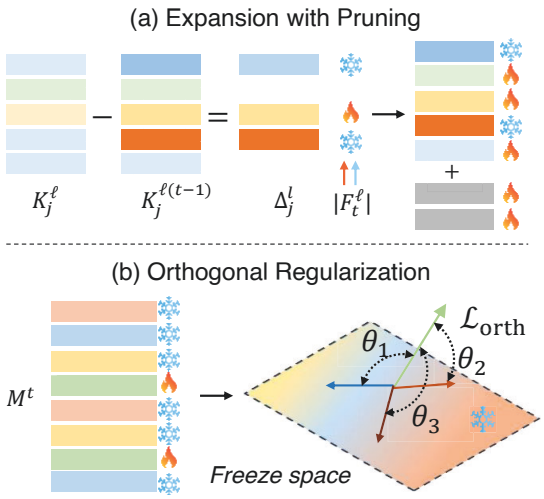

Figure 3: Schematic diagram of memory expansion, knowledge pruning and orthogonal regularization.

Orthogonal Regularization for Slot Separation

While freezing prevents direct parameter updates to old slots, the representational overlap between old and new knowledge can still cause interference. Orthogonal regularization is applied exclusively to the task-specific memory M^t, enforcing geometric separation between its frozen and newly expanded slots, which encourages the model to learn new task-specific features in directions orthogonal to all preserved task-specific features, further mitigating interference and promoting feature disentanglement.

Let $K_F^t \in \mathbb{R}^{|\mathcal{F}^t| \times d}$ and V_F^t be the matrix of frozen keys and values in M^t, and K_U^t, V_U^t their unfrozen (active) counterparts. We define the orthogonality loss only for M^t as

$$\mathcal{L}_{\text{orth}} = \left\| K_F^t \left(K_U^t\right)^\top \right\|_F^2 + \left\| V_F^t \left(V_U^t\right)^\top \right\|_F^2, \quad (7)$$

which penalizes any alignment between frozen and active task-specific slots. Equivalently, this encourages $\langle \mathbf{k}_i^t, \mathbf{k}_j^t \rangle = 0$ for every frozen–unfrozen pair (i, j) (and likewise for value vectors). In practice, each key/value vector is normalized to unit length and the average squared cosine similarity between every frozen–unfrozen pair is minimized. This orthogonal constraint on the task-specific memory directly shapes the fast learner's feature extractor, ensuring that newly encoded task details occupy an independent subspace and do not interfere with previously consolidated task–specific knowledge. During training, $\mathcal{L}_{\text{orth}}$ is added to the total loss with weight λ_{orth}.

Memory-Guided Representation Alignment

Finally, to consolidate inter-task knowledge, a memory-guided distillation aligns internal representations with those of the frozen network saved after the previous task. Rather than storing exemplars, the model uses its memory to transfer knowledge: for any input (including new-task samples), the pattern of memory activation in the current model is encouraged to match that of the previous model. This method ensures that important features learned in previous tasks are not forgotten but are reused and preserved in the new model's latent space.

Concretely, let $F^{(t-1)}$ be the previous model (after task $t-1$, kept fixed during learning of task t) and $f^{(t)}$ the current network. When training on data $x \in \mathcal{T}_t$, it is forwarded through both models to obtain their memory outputs. Denote by $A_{\text{old}}^\ell(x)$ and $A_{\text{new}}^\ell(x)$ the attention weight vectors (or equivalently, the normalized key affinity scores $[w_1, \ldots, w_{L_\ell}]$) produced by memory M^ℓ in the previous and current models, respectively. We define a memory alignment loss that penalizes differences in these activation patterns using a cosine embedding loss as

$$\mathcal{L}_{\text{align}} = \sum_{\ell \in \{s,t\}} \mathbb{E}_{x \sim \mathcal{T}_t} \left[1 - \cos\left(A_{\text{new}}^\ell(x), A_{\text{old}}^\ell(x)\right)\right] \quad (8)$$

where $\cos(u, v) = \frac{u \cdot v}{\|u\| \|v\|}$, averaging over training samples. This loss term drives $A_{\text{new}}^\ell(x)$ to be close to $A_{\text{old}}^\ell(x)$ for each memory. Thus, the current network retrieves a similar combination of memory slots as the previous model did for the same input. In effect, the current model is guided to mimic the previous model's internal memory representations, preserving the relative importance of learned features. This alignment is applied to both the shared and task-specific memory. The alignment loss is weighted by λ_{mem} and incorporated into the training objective.

Learning Process

For the initial task (i.e., training on the first dataset), the model is trained from scratch using only the standard classification loss. Since there is no previous knowledge to preserve, the memories are optimized jointly with the encoder and classifier, and cross task regularizers (memory alignment and orthogonality) are not applied. Upon completion of the first task, we save a copy of the resulting model, including its encoder, classifier, and learned memory slots, as the previous model for subsequent tasks.

For each subsequent task $t \geq 2$, we employ a previous model and current training procedure. Before training on the new task begins, the previous model remains frozen, and its batch normalization layers are calibrated to the new task distribution. Specifically, we forward the new task's data through the previous model for several epochs in training mode without gradient updates, allowing its running mean and variance to adapt to the new distribution. This step mitigates abrupt distribution shifts without altering the previous model's learned parameters.

Next, we initialize the current model by copying the previous model and train on the new task's data. The current model's objective combines the classification loss on the current classes with the memory alignment loss $\mathcal{L}_{\text{align}}$ and the orthogonal regularization $\mathcal{L}_{\text{orth}}$. After training on task t, we perform expansion and pruning. The updated current model is then designated as the previous model for the next task, and this process repeats for all tasks in the sequence.

Overall, the training loss for task t combines the standard classification loss and the proposed regularizers:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}}^{(t)} + \lambda_{\text{mem}} \mathcal{L}_{\text{align}} + \lambda_{\text{orth}} \mathcal{L}_{\text{orth}} \quad (9)$$

where $\mathcal{L}_{CE}^{(t)}$ is the cross-entropy on task t data plus an output-distillation term from the previous model $F^{(t-1)}$. By jointly optimizing this objective, the model achieves a balance between plasticity and stability: new task performance is driven by $\mathcal{L}_{CE}$, while $\mathcal{L}_{align}$ and $\mathcal{L}_{orth}$ act as complementary memory mechanisms that consolidate prior knowledge. Empirically, each component proves essential: memory expansion allocates capacity to new information while safeguarding prior features, orthogonal regularization enforces task-specific subspace separation, and memory-guided alignment preserves and reuses salient representations.

Experimental Results

Datasets and Implementation Details

To evaluate EDD in CIL, exemplar-free settings, we use three standard image classification benchmarks, each split into a sequence of tasks. CIFAR-10 is divided into 5 tasks of 2 classes each, CIFAR-100 into 10 tasks of 10 classes each and 20 tasks of 5 classes each, and TinyImageNet into 10 tasks of 20 classes each and 20 tasks of 10 classes each. EDD is implemented in PyTorch by extending a ResNet-18 backbone with dual memory inserted after the first and the second residual block. All baseline methods use the same ResNet-18 backbone and are trained with Adam optimizer using a batch size of 128 for 50 epochs. Full experimental settings and additional hyperparameter details are provided in Appendix A.

Accuracy Analysis

Table 1 summarizes the performance of state-of-the-art CL methods and dual-memory approaches under CIL across datasets of increasing difficulty and task lengths. EDD consistently outperforms all baselines, including exemplar-based methods such as LUCIR and DualNet, even in a strict exemplar-free setting. As we move from S-CIFAR-10 to S-Tiny-ImageNet (20 tasks), EDD's performance margin over the strongest competitor expands from 5.6% to 26.4%, demonstrating its robustness to both dataset complexity and an increasing number of tasks.

EDD outperforms the alternatives by separating shared and task-specific features into distinct memory and by securing stability and plasticity through expansion, pruning, and orthogonal regularization. As a result, the method exhibits much slower forgetting than other approaches, as shown in Figures 4 and 5, and even demonstrates modest accuracy gains on some intermediate tasks. Analysis reveals that whereas existing models sacrifice stability and lose past knowledge when adapting to new tasks, EDD preserves shared knowledge simply by resisting updates to frozen slots, thus maintaining stability, and simultaneously achieves plasticity by leveraging accumulated representations and expanded memory. This dual mechanism effectively solves the stability–plasticity dilemma: unlike other exemplar-free methods that lose most past knowledge over long task sequences, EDD consistently outperforms competing techniques from the middle tasks onward. Consequently, EDD consistently outperforms competing exemplar-free methods from intermediate tasks onward, effectively resolving the stability–plasticity dilemma.

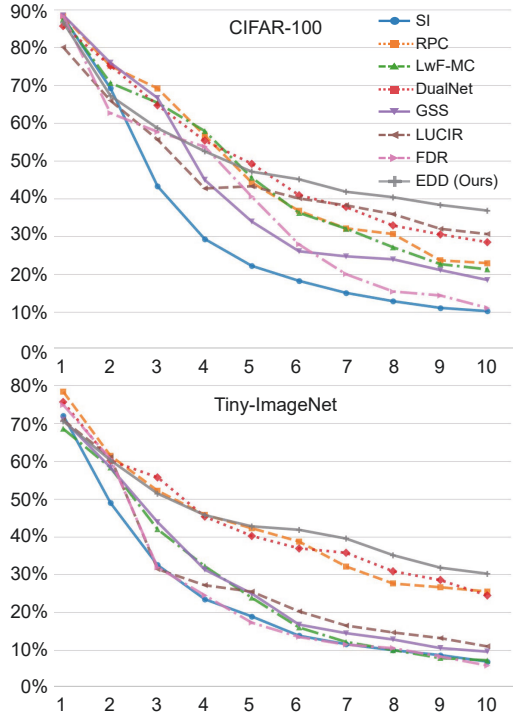

Figure 4: Task-wise accuracy for various CL methods on CIFAR-100 and TinyImageNet under 10 tasks. x-axis denotes the task index and y-axis indicates classification accuracy per task, illustrating differing forgetting dynamics across methods.

Ablation Study

Ablation study in Table 2 shows that each component contributes positively and their effects accumulate. Adding memory alignment to the baseline yields +2.35%p and +1.74%p; orthogonal regularization alone gives +1.48%p and +1.17%p. Put them all together, they produce +3.20%p and +2.65%p, demonstrating their complementarity. Incorporating batch adaptation (BA) further adds +1.57%p and +2.08%p by maintaining activation quality under distribution shifts. These consistent gains across moderate and high-difficulty datasets confirm that the fusion of memory adjustment, orthogonal regularization, and BA provides a robust, scalable exemplar-free solution for CL.

Comparison with Joint Learning

Figure 6 shows four alignment metrics between each method's representations and those from joint learning. Across all metrics, EDD achieves the closest match to the joint-learning model. It attains the highest average cosine similarity, indicating strong directional agreement, while simultaneously registering the lowest KL divergence and Wasserstein distance, together reflecting minimal distributional discrepancy. Furthermore, its average feature distance

Method	S-CIFAR-10 Acc (±)	S-CIFAR-100 Acc (±)		S-Tiny-ImageNet Acc (±)	
		10-task	20-task	10-task	20-task
JT (upper bound)	83.38±0.15	70.44±0.17	70.44±0.17	59.99±0.19	59.99±0.19
FT (lower bound)	18.35±0.81	4.43±0.42	2.91±0.15	5.84±1.67	2.51 ±0.37
SI (Zenke, Poole, and Ganguli 2017)	19.48±0.17	10.25±0.28	7.56±0.27	6.97±0.31	4.67±0.15
o-EWC (Kirkpatrick et al. 2017)	19.49±0.12	6.07±0.24	4.05±0.26	6.58±0.10	4.08 ±0.31
LwF (Li and Hoiem 2017)	19.61±0.05	9.24±0.33	6.98±0.11	8.46±0.22	5.99±0.22
A-GEM (Chaudhry et al. 2018)	19.64±0.57	10.33±0.29	5.65±0.13	7.81±0.04	5.22±0.12
HAL (Chaudhry et al. 2021)	19.99±0.28	8.33±0.18	6.77±0.22	8.89±0.94	5.97±0.15
GEM (Lopez-Paz and Ranzato 2017)	22.78±1.26	13.60±0.27	8.94±0.22	16.34±1.18	7.88±0.13
FDR (Titsias et al. 2019)	25.55±3.23	11.06±0.32	7.51±0.10	5.93±0.21	3.42±0.07
PNN (Rusu et al. 2016)	28.23±2.50	16.15±1.89	10.64±0.53	10.97±1.58	7.69±0.35
DualNet(Pham, Liu, and Hoi 2021)	41.69±0.27	28.96±0.19	15.91±0.29	24.48±0.25	12.56±0.17
LwF-MC (Rebuffi et al. 2017)	42.78±0.35	21.26±0.31	16.87±0.34	15.34±0.27	11.91±0.19
Lucir (Hou et al. 2019)	42.78±0.94	30.57±0.58	<u>19.99±0.58</u>	25.84±0.67	<u>14.51±0.37</u>
RPC (Pernici et al. 2021)	49.11±0.29	<u>31.08±0.33</u>	17.79±0.37	<u>25.51±0.75</u>	12.25±0.32
GSS (Aljundi et al. 2019)	49.21±0.78	18.44±0.27	14.95±0.26	10.53±0.16	8.81±0.49
PEC (Zajkac, Tuytelaars, and van de Ven 2023)	<u>52.19±0.18</u>	21.82±0.12	18.29±0.11	15.97±0.34	13.51±0.10
EDD	**55.13±0.21**	**37.24±0.29**	**21.68±0.10**	**30.11±0.22**	**18.34±0.22**

Table 1: Class-IL accuracy for various continual learning methods on S-CIFAR-10, S-CIFAR-100 (10/20 tasks) and S-Tiny-ImageNet (10/20 tasks).Bold denotes the best results, and underline values denote the second best. (buffer=500 if model has buffer).

Configuration	CIFAR-100	TinyImageNet
Naive $\mathcal{L}_{\text{CE}}$	32.47 ± 0.62	25.38 ± 0.54
$+\mathcal{L}_{\text{align}}$	34.82 ± 0.58	27.12 ± 0.47
$+\mathcal{L}_{\text{orth}}$	33.95 ± 0.71	26.55 ± 0.52
$+\mathcal{L}_{\text{align}} + \mathcal{L}_{\text{orth}}$	35.67 ± 0.49	28.03 ± 0.61
+ BA $+\mathcal{L}_{\text{align}} + \mathcal{L}_{\text{orth}}$	37.24 ± 0.29	30.11 ± 0.22

Table 2: Ablation study (%) on CIFAR-100 (10 tasks) and TinyImageNet (10 tasks). Mean ± std.

is the smallest, confirming that class representations cluster more tightly around those of joint learning. In contrast, competing approaches such as PEC, DualNet, LUCIR, LwF-MC, AGEN, RPC, and GEM fail to match EDD on at least one metric and exhibit sharp declines in feature distance and cosine similarity, indicating substantial divergence from the joint-learning baseline.

Figure 7 shows, for CIFAR-10, the per-class cosine similarity to joint learning at each task step, ordered from left to right. All other methods exhibit a gradual decline in similarity as new classes are added, reflecting reduced plasticity and an increasing departure from the ideal joint representation. In contrast, EDD maintains consistently high similarity across all classes and even shows slight increases for some tasks. This stability underscores its ability to integrate new information without compromising previously learned class representations and preserve close alignment with the joint-learning features throughout the entire sequence. In particular, the fourth class in the second task achieves the highest similarity score, indicating that its representation remains intact even after all subsequent tasks have been learned. By comparison, RPC and GSS converge to near-zero similarity by the final task, despite focusing on stability in the first task, while DualNet suffers from a steady decline in similarity over time.

In summary, our feature analysis demonstrates that EDD is proved to its design as a expandable differentiable dual memory method and learns, maintains and reuses shared transferable features and task-specific discriminative features across all tasks, thereby overcoming the barrier posed by independent task environments. This is supported by experimental results showing that EDD achieves the highest similarity to joint-learning across every task and class, and maintains a statistically significant level of similarity throughout the entire task sequence.

Time and Space Complexity

The time complexity of EDD is determined by the additional operations performed in each training step. During the forward pass, each memory performs a read operation, which has a complexity of $O(B \cdot HW \cdot C \cdot L)$, where B is the batch size. A key computational overhead lies in the orthogonal regularization loss. This involves a matrix multiplication with a complexity of approximately $O(L^2 \cdot C)$. The memory alignment loss attributes an additional $O(B \cdot HW \cdot L)$. In contrast, the memory adjustment occurs only once at the end of each task with a low amortized cost of $O(L \cdot \log(L))$. Therefore, while EDD introduces a principled computational overhead, it remains manageable for a moderate number of tasks, directly contributing to mitigating catastrophic forgetting. The space complexity is mainly influenced by the storage of the memories and a full copy of the previous network. The memories themselves require $O(L \cdot C)$ space for their keys and values.

These theoretical complexities do not present a bottleneck in practice. The space requirement is at most twice of

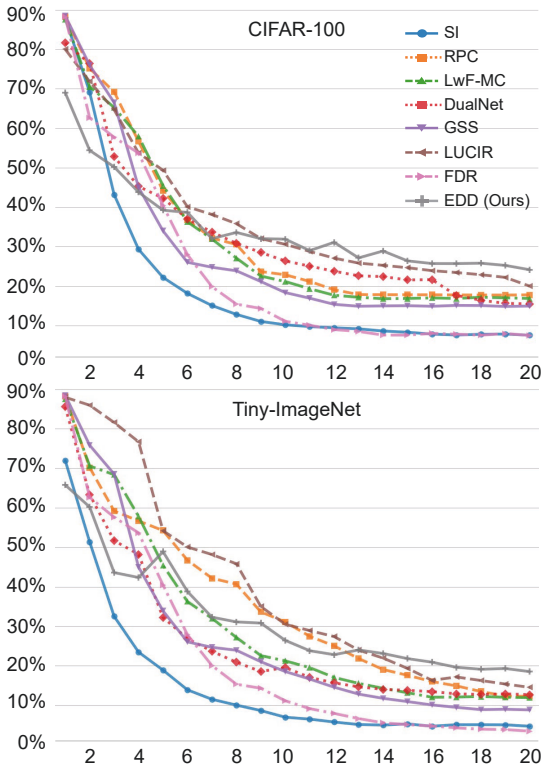

Figure 5: Task-wise accuracy for various CL methods on CIFAR-100 and TinyImageNet under 20 tasks.

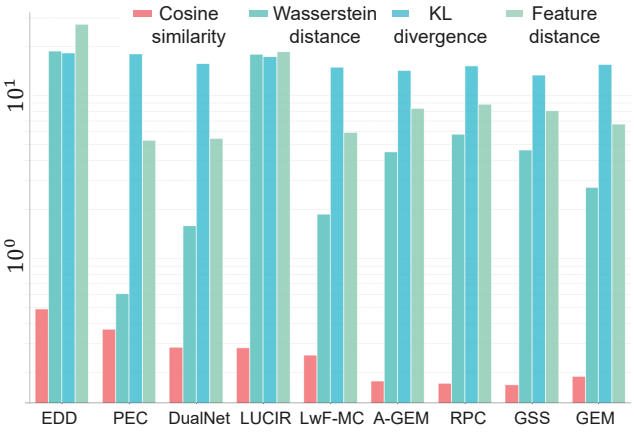

Figure 6: Alignment of class-incremental features with joint-learning representations. Metric values between each method and the joint-learning baseline across all tasks.

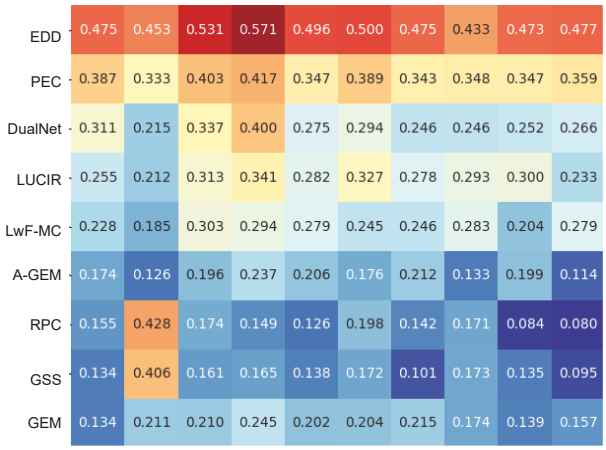

Figure 7: Per-class cosine similarity to joint learning for CIFAR-10. Each value shows how similarity evolves over the 5 two-class tasks.

the base model. Because the memory size L, is fixed, the per-step computational cost remains constant regardless of the number of tasks. The experimental results validate this, showing that EDD's runtime is, in fact, moderate and competitive compared to other continual learning methods (see Appendix B for details).

Limitation

Despite its advantages, EDD has certain limitations. First, while memory operations are computationally efficient for a moderate number of tasks, the cumulative overhead of managing and expanding the memory can increase as the task sequence grows very long. Second, under extreme task shift scenarios where successive tasks share minimal common structure, the method's generalizability may be constrained, since the shared memory might struggle to capture entirely divergent knowledge. Finally, scaling the dual-memory to tasks with extremely high-dimensional inputs or very large output spaces could present practical difficulties in terms of memory footprint and computational cost. Addressing these challenges may require further architectural optimizations.

Concluding Remarks

We have proposed a novel exemplar-free expandable differentiable dual-memory. By integrating orthogonal regularization and memory-guided distillation, EDD preserves a shared knowledge across tasks and avoids the representational fragmentation observed in previous approaches. This design substantially mitigates catastrophic forgetting and maintains feature-space coherence comparable to joint learning, all achieved without storing any past exemplars.

In future work, one promising direction is to adapt the dual-memory to transformer-based architectures such as vision transformers to assess its generality across different network backbones. Another extension is to apply the approach to significantly longer and more complex real-world task streams in order to rigorously evaluate its scalability and continual adaptation capabilities. Furthermore, we plan to explore modeling the differentiable memory as a class-level relational graph that evolves with each new task, which could capture rich inter-class relationships over time and further enhance knowledge transfer and retention.

Acknowledgments

This work was supported by the Yonsei Fellow Program funded by Lee Youn Jae, IITP grant funded by the Korea government (MSIT) (No. RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University), No. RS-2022-II220113, Developing a Sustainable Collaborative Multi-modal Lifelong Learning Framework).

References

Aljundi, R.; Lin, M.; Goujaud, B.; and Bengio, Y. 2019. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32.

Arani, E.; Sarfraz, F.; and Zonooz, B. 2022. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*.

Benjamin, A. S.; Rolnick, D.; and Kording, K. 2018. Measuring and regularizing networks in function space. *arXiv preprint arXiv:1805.08289*.

Boschini, M.; Bonicelli, L.; Porrello, A.; Bellitto, G.; Pennisi, M.; Palazzo, S.; Spampinato, C.; and Calderara, S. 2022. Transfer without forgetting. In *European conference on computer vision*, 692–709. Springer.

Chaudhry, A.; Gordo, A.; Dokania, P.; Torr, P.; and Lopez-Paz, D. 2021. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 6993–7001.

Chaudhry, A.; Ranzato, M.; Rohrbach, M.; and Elhoseiny, M. 2018. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*.

Douillard, A.; Ramé, A.; Couairon, G.; and Cord, M. 2022. DyTox: Transformers for Continual Learning With DYnamic TOken eXpansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9285–9295.

French, R. M. 1995. Interactive tandem networks and the sequential learning problem. In *Cognitive Science Society Conf.*, 1–6.

Goswami, D.; Soutif-Cormerais, A.; Liu, Y.; Kamath, S.; Twardowski, B.; Van De Weijer, J.; et al. 2024. Resurrecting old classes with new data for exemplar-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 28525–28534.

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z.; and Lin, D. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 831–839.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12): 2935–2947.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Moon, H. J.; and Cho, S. B. 2025a. Continual Learning by Contrastive Learning of Regularized Classes in Multivariate Gaussian Distributions. *International Journal of Neural Systems*, 35(6): 2550025.

Moon, H. J.; and Cho, S. B. 2025b. An Efficient Method of Lifelong Learning with Differentiable Memory for Edge Computing. In *58th Hawaii International Conference on System Sciences, HICSS 2025*, 7268–7276. IEEE Computer Society.

Pernici, F.; Bruni, M.; Baecchi, C.; Turchini, F.; and Del Bimbo, A. 2021. Class-incremental learning with preallocated fixed classifiers. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 6259–6266. IEEE.

Pham, Q.; Liu, C.; and Hoi, S. 2021. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34: 16131–16144.

Qi, B.; Chen, X.; Gao, J.; Li, D.; Liu, J.; Wu, L.; and Zhou, B. 2024. Interactive continual learning: Fast and slow thinking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12882–12892.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2001–2010.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Sun, Z.; Mu, Y.; and Hua, G. 2023. Regularizing second-order influences for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 20166–20175.

Tang, S.; Su, P.; Chen, D.; and Ouyang, W. 2021. Gradient regularized contrastive learning for continual domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 2665–2673.

Titsias, M. K.; Schwarz, J.; Matthews, A. G. d. G.; Pascanu, R.; and Teh, Y. W. 2019. Functional regularisation for continual learning with gaussian processes. *arXiv preprint arXiv:1901.11356*.

Ye, F.; and Bors, A. G. 2023. Self-evolved dynamic expansion model for task-free continual learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, 22102–22112.

Yoon, J.; Yang, E.; Lee, J.; and Hwang, S. J. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

Zajkac, M.; Tuytelaars, T.; and van de Ven, G. M. 2023. Prediction error-based classification for class-incremental learning. *arXiv preprint arXiv:2305.18806*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *International conference on machine learning*, 3987–3995. PMLR.