FLY-CL: A FLY-INSPIRED FRAMEWORK FOR ENHANCING EFFICIENT DECORRELATION AND REDUCED TRAINING TIME IN PRE-TRAINED MODEL-BASED CONTINUAL REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Using a nearly-frozen pretrained model, the continual representation learning paradigm reframes parameter updates as a similarity-matching problem to mitigate catastrophic forgetting. However, directly leveraging pretrained features for downstream tasks often suffers from multicollinearity in the similarity-matching stage, and more advanced methods can be computationally prohibitive for real-time, low-latency applications. Inspired by the fly olfactory circuit, we propose Fly-CL, a bio-inspired framework compatible with a wide range of pretrained backbones. Fly-CL substantially reduces training time while achieving performance comparable to or exceeding that of current state-of-the-art methods. We theoretically show how Fly-CL progressively resolves multicollinearity, enabling more effective similarity matching with low time complexity. Extensive simulation experiments across diverse network architectures and data regimes validate Fly-CL's effectiveness in addressing this challenge through a biologically inspired design.

1 INTRODUCTION

Artificial neural networks have exhibited remarkable capabilities across various domains in recent years. Nevertheless, real-world applications often require continuous model adaptation to handle progressively emerging unseen scenarios, making updates based on sequential incoming data essential. This need has led to the development of Continual Learning (CL). Earlier research primarily focused on training models from scratch (Aljundi et al., 2018; Kirkpatrick et al., 2017; Li & Hoiem, 2017; Zenke et al., 2017). Pretrained models have recently become prominent in CL, owing to their robust generalization in downstream tasks for downstream tasks (Wang et al., 2022a;b) .

Popular CL methods utilizing pre-trained models can generally be classified into three categories: (1) prompt/adapter-based approaches (Jung et al., 2023; Smith et al., 2023; Tang et al., 2023; Wang et al., 2022a;b; 2025; Liang & Li, 2024; Yu et al., 2024), (2) mixture-based approaches (Chen et al., 2023; Gao et al., 2023; Wang et al., 2023a;b; Zhou et al., 2023b), and (3) representation-based approaches (McDonnell et al., 2023; Sun et al., 2025; Zhou et al., 2023a; 2024; Zhuang et al., 2024). All three paradigms operate without exemplars and significantly outperform traditional training-from-scratch methods. Despite their strengths, each approach has limitations. Prompt/adapter-based methods are constrained to transformer architectures, and updating prompts/adapters inherently risks propagating forgetting within the prompt/adapter space. Mixture-based approaches require storing previous models, resulting in significant storage overhead and increased computational complexity during model fusion. Compared with parameter-learning approaches, **representation-based methods perform better by reframing learning as similarity matching and avoiding dependence on a specific backbone**. They form class prototypes (CPs) by averaging features extracted by a frozen pretrained network, with each prototype acting as the centroid of its class. However, **insufficient separation between features can lead to ambiguous decision boundaries when distinguishing between prototypes, complicating the similarity matching process**. This challenge is formally recognized as the multicollinearity problem. Previous studies (McDonnell et al., 2023) have attempted to address this issue, but they still incur substantial computational costs, hindering real-world deployment.

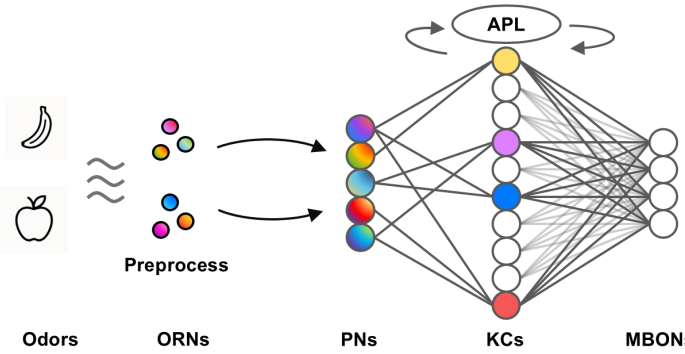


Figure 1: **Schematic of the Fly Olfactory Circuit.** Odors are first detected and pre-processed by olfactory receptor neurons (ORNs) in the antennal lobe, where feature extraction and normalization take place, before being transmitted to projection neurons (PNs). Expansion coding occurs as signals move from PNs to Kenyon cells (KCs), with an expansion ratio of approximately 40. Each KC connects to a fixed number of PNs (about 6). Lateral inhibition, mediated by an anterior paired lateral (APL) neuron, suppresses most weakly activated KCs, exemplifying a winner-take-all strategy. Finally, signals from KCs to mushroom body output neurons (MBONs) involve a dense down-projection that reduces dimensionality to select specific actions.

In biological systems, pattern separation is a well-recognized circuit motif across cerebellum-like and related networks, including the cerebellum, the hippocampus, and the fly olfactory system (Lin et al., 2014; Papadopoulou et al., 2011; Stevens, 2015; Zang & De Schutter, 2023). These neural circuits implement efficient decorrelation of overlapping sensory inputs. Figure 1 illustrates the information processing mechanism in the fly olfactory circuit. The process unfolds in several steps: in response to an odor stimulus, olfactory neurons extract and pre-process odor information represented as a 50-dimensional vector (PNs). These pre-extracted features are then randomly projected into expanded dimensions (KCs), selectively activating a small subset of KCs that receive the strongest excitation while zeroing out others. The high-dimensional features in KCs subsequently converge to low-dimensional MBONs for classification. Over the years, numerous studies have highlighted the role of the PN→KC transformation in producing decorrelated representations, and many strategies have been proposed to model this process. By contrast, the downstream KC→MBON transformation has received far less attention, and a clear theory of its contribution to decorrelation has yet to be established. To address this gap, **we examine whether the KC→MBON pathway also supports decorrelation**.

Inspired by the fly olfactory circuit, we propose Fly-CL, an efficient framework for progressive decorrelation in representation-based learning with pre-trained models. The pipeline starts with feature extraction and normalization using a pre-trained model, followed by random sparse projection into a high-dimensional space and a top-k operation for collective decorrelation, mimicking the PN-to-KC process. We then implement a similarity matching mechanism between class prototypes that mimics the KC-to-MBON structure during inference, and an efficient streaming ridge classification method to decorrelate parameter weights during training. Empirical results show substantially reduced time consumption versus baseline methods.

Our main contributions are as follows:

1. We propose an efficient and biologically plausible decorrelation framework that significantly reduces computational costs compared to current SOTA methods while achieving comparable or improved performance in CL.
2. Our method's effectiveness and robustness in decorrelation are validated by extensive experiments under various data setups and model architectures, supported by theoretical and empirical analyses.
3. The alignment of our framework with the fly olfactory circuit suggests that biological structures can inspire effective and efficient solutions to AI problems.

2 Related Work

Representation-based methods in CL using Pre-trained Models: Representation-based methods (McDonnell et al., 2023; Sun et al., 2025; Zhou et al., 2023a; 2024) demonstrate superior performance

over parameter-learning approaches by leveraging features extracted from a frozen pre-trained model to compute similarities with class prototypes. They are also more practical for resource-constrained deployments (e.g., edge computing scenarios), since they avoid updating the entire model. For instance, in a smart camera system, such methods enable the efficient addition of new object recognition capabilities without retraining the entire model. While showing promising progress, their computational overhead remains substantial, which may limit their applicability in scenarios requiring real-time responses.

Fly Olfactory Circuit: Information processing in cerebellum-like circuits, including the fly olfactory circuit, involves several stages (Lin et al., 2014; Papadopoulou et al., 2011; Stevens, 2015; Zang & De Schutter, 2023). Theoretical neuroscience studies suggest that most stages exhibit progressive feature separation effects (Hige et al., 2015). Algorithms inspired by the fly olfactory circuit have been applied across various AI domains, including locality-sensitive hashing (Dasgupta et al., 2017; Ryali et al., 2020; Sharma & Navlakha, 2018), word embedding (Liang et al., 2021), and federated learning (Ram & Sinha, 2022).

3 Background

3.1 Problem Statement

In this paper, we focus on CL within the context of image classification tasks. We denote sequentially arriving tasks as $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$, where each task $\mathcal{D}_t = \{(\boldsymbol{x}_t^i, y_t^i)\}_{i=1}^{n_t}$ consists of n_t samples. Each sample $\boldsymbol{x}_t^i$ within a task is drawn from the input space $\mathcal{X}_t$, and its corresponding label y_t^i belongs to the label space $\mathcal{Y}_t$. The training process involves sequential learning from $\mathcal{D}_1$ to $\mathcal{D}_T$, followed by class prediction on an unseen test set spanning the full label space.

To demonstrate the effectiveness and efficiency of our framework, we adopt Class Incremental Learning (CIL), a widely used experimental setup. Unlike traditional Task Incremental Learning, CIL does not provide access to the task ID during the testing process, making it more challenging. In CIL, the model learns mutually exclusive classes within each task, ensuring that the intersection of label spaces satisfies $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$. We denote that there are c_t classes for the first t tasks.

3.2 Representation-based paradigm in CL

We study on the basis of the recently popular representation-based paradigm using pre-trained models (McDonnell et al., 2023; Sun et al., 2025; Zhou et al., 2023a; 2024), which demonstrates superior performance for CL. Given an input image $\boldsymbol{x}_t^i$, it is first compressed into a d-dimensional feature $\boldsymbol{v}_t^i = f_\theta(\boldsymbol{x}_t^i) \in \mathbb{R}^d$ using a pre-trained encoder f_θ. For each class i in task t, we compute its prototype by averaging features over all training samples belonging to this class:

$$\boldsymbol{\mu}_t^i = \frac{1}{N_i} \sum_{j=1}^{|\mathcal{D}_t|} \mathbb{I}(y_t^j = i) f_\theta(\boldsymbol{x}_t^j) \in \mathbb{R}^d, \tag{1}$$

where $N_i = \sum_{j=1}^{|\mathcal{D}_t|} \mathbb{I}(y_j = i)$ denotes the cardinality of class i's training set, and $\mathbb{I}(\cdot)$ is the indicator function. During inference, for a test sample with feature vector $\boldsymbol{v}$, the predicted class $\hat{y}$ is determined by finding the maximum cosine similarity between $\boldsymbol{v}$ and all learned class prototypes:

$$\hat{y} = \arg\max_{t,i} \frac{\boldsymbol{v}^\top \boldsymbol{\mu}_t^i}{\|\boldsymbol{v}\| \cdot \|\boldsymbol{\mu}_t^i\|}. \tag{2}$$

However, significant inter-prototype correlations ($\mathbb{E}[{\boldsymbol{\mu}_{t_i}^{m_i}}^\top \boldsymbol{\mu}_{t_j}^{m_j}] \gg 0$, see Figure 3(a)), can severely compromise the discriminative power of the similarity measurement (Belsley et al., 2005). This phenomenon arises because high correlations reduce the effective angular separation between classes in the embedding space, leading to ambiguous decision boundaries. Specifically, when prototypes cluster near a dominant direction in $\mathbb{R}^d$, the cosine similarity metric becomes less sensitive to subtle but critical inter-class distinctions, thereby degrading classification performance.

4 FLY-CL

In this section, we detail the design motivation and functionality of each component of our Fly-CL framework. A schematic of the overall framework is provided in Figure 2, along with the pseudocode for the training and inference pipeline in Appendix A. The decorrelation effect of each component is visualized in Figure 3 using Pearson correlation coefficients of different class prototypes.

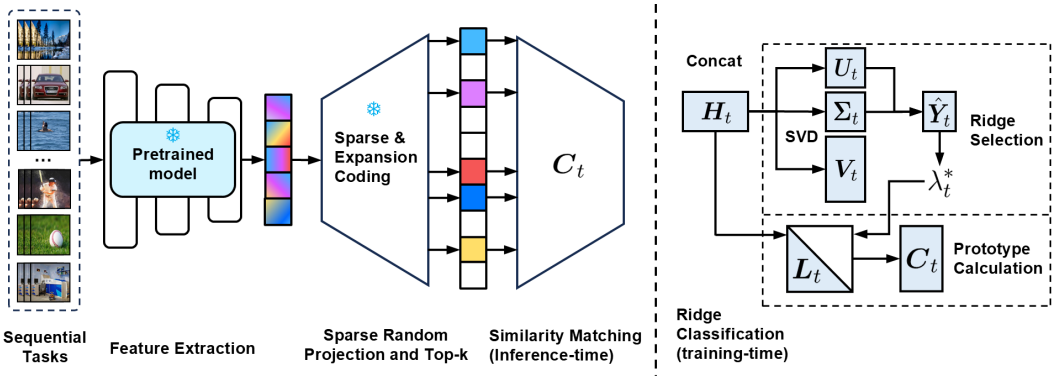


Figure 2: **Schematic of the Fly-CL Framework. Left:** Our framework extracts image embeddings using a frozen pre-trained model, projects them into a higher-dimensional space via a fixed sparse random projection, and filters them through a top-k operation (PNs $\rightarrow$ KCs). Then we utilize a learned down-projection for similarity matching during inference time (KCs $\rightarrow$ MBONs). **Right:** During the training phase, the parameter $\boldsymbol{C}_t$ is learned via a streaming ridge classification scheme.

4.1 SPARSE RANDOM PROJECTION AND TOP-K OPERATION

Building upon the representation-based paradigm, it is necessary to decouple different class prototypes. Inspired by the decorrelation mechanism of the fly olfactory circuit, we emulate the sparse expansion projection from PNs to KCs, followed by winner-take-all inhibition mediated by APL neurons. Given a feature embedding $\boldsymbol{v} \in \mathbb{R}^d$ extracted from the pre-trained encoder, we formulate the transformation $Z(\boldsymbol{v}) : \mathbb{R}^d \rightarrow \mathbb{R}^m$ as:

$$\boldsymbol{h}' = Z(\boldsymbol{v}) = \text{top-}k(\boldsymbol{h}) = \text{top-}k\left(\boldsymbol{W}\boldsymbol{v}\right), \tag{3}$$

where the fixed projection matrix $\boldsymbol{W} \in \mathbb{R}^{m \times d}$ (with $m \gg d$) implements weight sparsity: each row contains exactly p ($p < d$) non-zero entries independently sampled from $\mathcal{N}(0, 1)$. The top-k operator implements activation sparsity by preserving only the k largest components ($k < m$) while zeroing out others, formally defined as:

$$[\boldsymbol{h}']_i = \begin{cases} [\boldsymbol{h}]_i & \text{if the magnitude of } [\boldsymbol{h}]_i \text{ is among the top-}k \text{ values of } \boldsymbol{h}, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

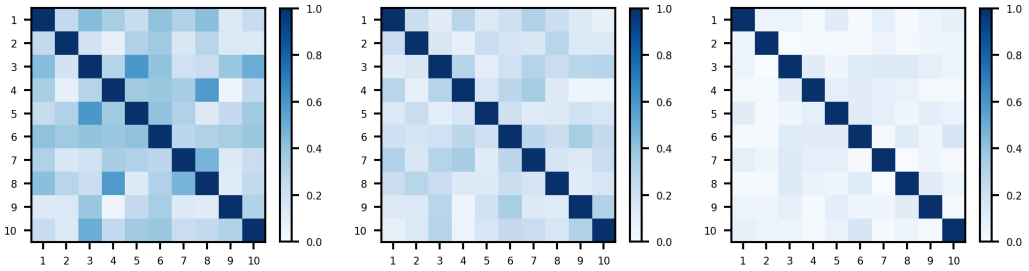
This two-stage process achieves effective decorrelation through the following properties, and its empirical effect is visualized in the transformation from Figure 3(a) to (b).

1. **High-Dimensional Embedding Enhances Linear Separability**: Random projection of low-dimensional features into an extremely high-dimensional space can improve the linear separability of the feature representations (Litwin-Kumar et al., 2017).

2. **Powerful Inhibition Suppresses Noisy Components**: The top-k operation imposes sparsity by suppressing noisy dimensions that may interfere with discrimination through dimensional competition, while enhancing separation by keeping the most discriminative dimensions (Metwally et al., 2006).

Considering computational efficiency, $\boldsymbol{W}$'s sparse pattern reduces the time complexity for random projection from $\mathcal{O}(mn_td)$ to $\mathcal{O}(mn_tp)$ while preserving the core representational capacity compared to dense projection. Similarly, the top-k operation reduces similarity matching complexity from $\mathcal{O}(mn_tc_t)$ to $\mathcal{O}(kn_tc_t)$, while simultaneously improving performance.

We further propose two theorems to demonstrate that strong sparsity does not significantly degrade performance. According to Theorem 4.1, as long as p and d are not extremely small, the matrix $\boldsymbol{W}$ retains full column rank with probability $1 - o(1)$, which is a common approach to demonstrate that



(a) After feature extraction (b) After random projection (c) After similarity matching

Figure 3: **Pearson Correlation Coefficients of Prototypes at Different Decorrelation Stages in Fly-CL.** Heatmaps display Pearson correlation coefficients for 10 randomly selected class prototypes at each stage of our pipeline (consistent across visualizations).

a sparse random projection does not result in severe information loss. Furthermore, in Theorem 4.2, by proving that the performance degradation between this sparsification operation and the original vector is bounded, we show that if k is not extremely small, it can preserve most of its performance. For a complete proof, please refer to Appendix B.

Theorem 4.1. *Given the matrix $\boldsymbol{W} \in \mathbb{R}^{m\times d}$, where $m > d$, with each row having exactly p non-zero entries, which are randomly sampled from $\mathcal{N}(0,1)$. Let $\mathcal{W} \in \mathbb{R}^{d\times d}$ be any square submatrix of $\boldsymbol{W}$. Then, for any $\epsilon > 0$, it holds that*

$$\mathbb{P}\left(|\det(\mathcal{W})| \geq \left(\frac{p}{d}\right)^{d/2} \sqrt{d!}\exp(-d^{1/2+\epsilon})\right) = 1 - o(1).$$

Thus, for sufficiently large p and d, any submatrix $\mathcal{W}$ is invertible with probability at least $1 - o(1)$.

Theorem 4.2. *For top-k sparsification in the expanded dimension m, the performance degradation is bounded by:*

$$\mathbb{E}\left[|L(\boldsymbol{h}, y) - L(\boldsymbol{h}', y)|\right] \leq M \cdot \sqrt{\frac{C}{k} \cdot \mathbb{E}[\|\boldsymbol{h}\|_2^2]},$$

where $L(\cdot)$ is a performance loss function for downstream tasks and C, M are constants. To ensure negligible performance degradation, we require:

$$\sqrt{\frac{C}{k} \cdot \mathbb{E}[\|\boldsymbol{h}\|_2^2]} \leq \mathcal{O}\left(\frac{1}{\sqrt{m^\alpha}}\right),$$

i.e., when $k = \Omega(m^\alpha)$ $(0 < \alpha < 1)$, the error bound decays polynomially with increasing dimension.

4.2 Streaming Ridge Classification

Previous studies on decorrelation in the fly olfactory circuit have primarily focused on the transformation from PNs to KCs, where sparse and decorrelated representations have been experimentally observed. In contrast, the downstream transformation from KCs to MBONs has received little attention in this regard, and the physiological evidence remains inconclusive. This gap motivates us to investigate whether the KC→MBON pathway can also facilitate decorrelation. To better align with the goal of CL, we model it using a streaming ridge classification framework with adaptive regularization, which naturally achieves decorrelation while ensuring computational efficiency and compatibility with sequential data. Ridge classification (Hoerl & Kennard, 1970) mitigates feature collinearity through ℓ_2-regularization, trading increased bias for reduced variance by shrinking correlated feature weights, thereby stabilizing prototype estimation in non-i.i.d. sequential learning scenarios. Let $\boldsymbol{H}_t \in \mathbb{R}^{n_t\times m}$ denote the concatenation of high-dimensional features $\boldsymbol{h}'$ for n_t samples in task t, and $\boldsymbol{Y}_t \in \{0,1\}^{n_t\times c_t}$ represent the corresponding one-hot label matrix for the total c_t classes. We maintain two streaming statistics: a Gram matrix $\boldsymbol{G} \in \mathbb{R}^{m\times m}$, which captures self-correlated statistics, and a matrix $\boldsymbol{S} \in \mathbb{R}^{m\times c_t}$, which accumulates cross-dimensional weights for each class prototype. During each task iteration t, these are updated as follows:

$$\boldsymbol{G}_t \leftarrow \boldsymbol{G}_{t-1} + \boldsymbol{H}_t^\top \boldsymbol{H}_t, \qquad \boldsymbol{S}_t \leftarrow \boldsymbol{S}_{t-1} + \boldsymbol{H}_t^\top \boldsymbol{Y}_t. \tag{5}$$

The classifier matrix $\boldsymbol{C} \in \mathbb{R}^{m \times c_t}$ is updated via regularized least squares accordingly:

$$\boldsymbol{C}_t = (\boldsymbol{G}_t + \lambda \boldsymbol{I}_m)^{-1} \boldsymbol{S}_t. \tag{6}$$

Prediction for preprocessed new samples $\boldsymbol{h}' \in \mathbb{R}^m$ follows:

$$\hat{y} = \arg \max_{i \in \{1,\ldots,c_t\}} \boldsymbol{h}'^{\top} \boldsymbol{C}_{\cdot,i}, \tag{7}$$

where $\boldsymbol{C}_{\cdot,i}$ denotes the i-th column of modulated prototypes.

Adaptive Regularization: Due to the inherent heterogeneity of different tasks, a fixed penalty coefficients λ will cause suboptimal performance. For adaptive regularization, vanilla λ selection via grid search and cross-validation incurs prohibitive computational costs of $\mathcal{O}(lm^3)$ for l candidates where the expanded feature dimension m is extremely large (McDonnell et al., 2023). To achieve our efficiency desideratum, we draw inspiration from an adaptive Generalized Cross-Validation (GCV) (Golub et al., 1979) framework that analytically approximates cross-validation error without explicit validation steps that require calculating large matrix inverses.

Given new task data $\boldsymbol{H}_t \in \mathbb{R}^{n_t \times m}$, we first obtain its singular value decomposition (SVD) as $\boldsymbol{H}_t = \boldsymbol{U}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^{\top}$, where $\boldsymbol{U}_t \in \mathbb{R}^{n_t \times r}$ and $\boldsymbol{V}_t \in \mathbb{R}^{m \times r}$ are semi-orthogonal column matrices that satisfy $\boldsymbol{U}_t \boldsymbol{U}_t^{\top} = \boldsymbol{I}_{n_t}$, $\boldsymbol{V}_t \boldsymbol{V}_t^{\top} = \boldsymbol{I}_m$, and $\boldsymbol{\Sigma}_t = \mathrm{diag}(s_1, \ldots, s_r) \in \mathbb{R}^{r \times r}$ contains non-zero singular values with $r = \mathrm{rank}(\boldsymbol{H}_t) = \min(n_t, m)$ (typically it's of full rank, since numerical computation is usually precise). The time complexity for SVD is $\mathcal{O}(n_t r m)$. For l candidate regularization coefficients $\lambda \in \Lambda = \{\lambda_{min}, \ldots, \lambda_{max}\}$ on a log scale, we use the following steps to compute the GCV criterion for each one:

First, in $\mathcal{O}(lr)$ time, we get the shrinkage matrix and calculate the effective degrees-of-freedom by:

$$\boldsymbol{D}_t = \frac{\boldsymbol{\Sigma}_t^2}{\boldsymbol{\Sigma}_t^2 + \lambda \boldsymbol{I}_r}, \quad \mathrm{df}(\lambda) = \mathrm{tr}(\boldsymbol{D}_t) = \sum_{i=1}^{r} \frac{s_i^2}{s_i^2 + \lambda}. \tag{8}$$

Then, we reconstruct the prediction value of ridge regression in $\mathcal{O}(ln_t r c_t)$ time by

$$\hat{\boldsymbol{Y}}_t = \boldsymbol{U}_t (\mathrm{vecdiag}(\boldsymbol{D}_t) \otimes \boldsymbol{1}_c^{\top}) \odot \boldsymbol{U}_t^{\top} \boldsymbol{Y}_t, \tag{9}$$

where the $\otimes$ denotes the outer product, $\odot$ denotes the Hadamard product, vecdiag denotes extracting the diagonal elements and concatenating them into a column vector. Finally, we can get the GCV value by

$$\mathrm{GCV}(\lambda) = \frac{\|\boldsymbol{Y}_t - \hat{\boldsymbol{Y}}_t(\lambda)\|_F^2}{n_t \left(1 - \frac{\mathrm{df}(\lambda)}{n_t}\right)^2}, \tag{10}$$

with time complexity being $\mathcal{O}(ln_t c_t)$. The optimal regularization parameter is then selected by:

$$\lambda_t^* = \arg \min_{\lambda \in \Lambda} \mathrm{GCV}(\lambda). \tag{11}$$

Considering projected dimension m is extremely large, we can make a mild assumption $m > n_t$, and $lc_t \ll m$, thus $r = \min(n_t, m) = n_t$. The original l loop complexity is $\mathcal{O}(ln_t r c_t) = \mathcal{O}(ln_t^2 c_t) \ll \mathcal{O}(n_t^2 m) = \mathcal{O}(n_t r m)$. Hence, the time complexity is reduced to being determined by SVD, at $\mathcal{O}(n_t^2 m)$. Compared to vanilla cross-validation that takes $\mathcal{O}(lm^3)$, time consumption is greatly reduced.

Accelerated Prototype Calculation: Upon determining the optimal regularization parameter λ through GCV, we solve Eq. 6 to obtain class prototypes $\boldsymbol{C}_t$. While vanilla matrix inversion via LU decomposition provides a baseline implementation, for the sake of computational efficiency, we exploit the inherent positive-definiteness of $\boldsymbol{G}_t + \lambda_t \boldsymbol{I}_t$ to achieve computational acceleration through Cholesky factorization by:

$$\boldsymbol{L}_t \boldsymbol{L}_t^{\top} = \boldsymbol{G}_t + \lambda_t^* \boldsymbol{I}_m, \qquad \boldsymbol{C}_t = \boldsymbol{L}_t^{-\top} (\boldsymbol{L}_t^{-1} \boldsymbol{S}_t), \tag{12}$$

where $\boldsymbol{L}_t$ denotes the lower-triangular Cholesky factor. This approach reduces theoretical complexity from $\mathcal{O}(\frac{2}{3}m^3)$ to $\mathcal{O}(\frac{1}{3}m^3)$ for factorization, with triangular solves requiring half the FLOPs of general linear system solutions. The numerical stability of this method is ensured by the condition number bound $\kappa(\boldsymbol{L}_t) \leq \kappa(\boldsymbol{G}_t + \lambda_t^* \boldsymbol{I}_m)$, making it particularly suitable for ill-conditioned streaming scenarios where $\boldsymbol{G}_t$ may accumulate numerical noise over tasks. The decorrelation effect of the streaming ridge classification is visualized via the transformation from Figure 3(b) to (c).

Table 1: **Performance Comparison on Pre-trained ViT-B/16 Models.** We report the average training time per task (τ_{train}), average post-extraction training time (τ_{post}), and overall accuracy ($\bar{A}$) across three benchmark datasets: CIFAR-100, CUB-200-2011, and VTAB.

Method	CIFAR-100			CUB-200-2011			VTAB		
	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$
L2P	263.54±0.10	183.56±0.36	87.74±0.46	52.02±0.04	37.03±0.07	77.48±1.43	42.10±0.04	36.45±0.02	81.24±0.67
DualPrompt	231.89±0.63	153.66±0.47	87.47±0.58	46.28±0.05	31.64±0.06	79.89±1.44	38.14±0.16	31.64±0.06	80.85±1.34
InfLoRA	220.82±0.44	140.31±0.41	91.10±0.36	45.31±0.19	30.97±0.22	80.65±0.73	35.80±0.28	29.26±0.17	88.73±0.57
SEMA	241.60±0.82	160.87±0.69	92.04±0.25	48.21±0.23	32.96±0.20	84.31±0.37	45.50±0.13	39.82±0.26	91.18±0.46
MoE-Adapter	187.91±0.61	106.37±0.53	90.43±0.46	37.89±0.15	22.61±0.12	79.65±0.32	32.28±0.19	26.06±0.14	86.39±0.77
EASE	621.26±1.05	583.4±0.76	92.96±0.25	138.78±0.36	122.48±0.27	89.56±0.43	108.35±0.97	94.73±1.11	94.02±0.15
RanPAC	98.62±0.42	84.42±0.44	**94.21±0.11**	37.95±0.15	33.83±0.13	92.67±0.20	63.68±0.18	61.44±0.17	94.16±0.32
F-OAL	71.05±0.23	57.13±0.19	91.96±0.29	6.24±0.09	2.04±0.05	91.13±0.15	3.19±0.05	1.03±0.02	94.68±0.32
Fly-CL	**19.07**±0.07	**5.38**±0.01	93.89±0.12	**4.43**±0.11	**0.35**±0.01	**93.84**±0.18	**2.48**±0.13	**0.34**±0.03	**96.54**±0.38

5 Experiments

5.1 Experimental Setup

Datasets and Backbones: We conduct experiments using various architectures, including transformer-based and CNN-based backbones. Specifically, we utilize the Vision Transformer (ViT-B/16) (Dosovitskiy et al., 2020) and ResNet-50 (He et al., 2016) as representative architectures. We test our method on five widely used datasets: CIFAR-100 (Krizhevsky et al., 2009), CUB-200-2011 (Wah et al., 2011), VTAB (Zhai et al., 2019), ImageNet-R (Hendrycks et al., 2021a), and ImageNet-A (Hendrycks et al., 2021b). Further details of the data setup are provided in Appendix E.

Baselines: We compare Fly-CL against eight baselines, including two prompt-based approaches: L2P (Wang et al., 2022b) and DualPrompt (Wang et al., 2022a), three lora/adapter-based approaches: InfLoRA (Liang & Li, 2024), SEMA (Wang et al., 2025), and MoE-Adapter (Yu et al., 2024), as well as three representation-based methods: EASE (Zhou et al., 2024), RanPAC (McDonnell et al., 2023), and F-OAL (Zhuang et al., 2024). In Fly-CL, we find that applying data normalization according to the specific combination of backbone and dataset is beneficial; a detailed analysis can be found in Appendix C.3. For a fair comparison, all baselines use the same data normalization strategy as Fly-CL. Comparisons among the different baselines and implementation details are provided in Appendices D and E, respectively.

Evaluation Metrics: To assess CL performance, we employ four metrics: average accuracy (A_t), last stage accuracy (A_T), backward transfer (BWT), and overall accuracy ($\bar{A}$). The average accuracy at stage t is defined as: $A_t = \frac{1}{t}\sum_{i=1}^{t} a_{t,i}$, where $a_{t,i}$ denotes the test accuracy on the i-th task after training on the t-th task. Specially, we refer to the average accuracy at the last stage T as the last stage accuracy (A_T). The backward transfer is denoted as $BWT = \frac{1}{t-1}\sum_{i=1}^{T}(a_{t,i} - a_{i,i})$. The overall accuracy is computed as the mean of A_t across all T tasks: $\bar{A} = \frac{1}{T}\sum_{i=1}^{T} A_t$. To evaluate computational efficiency, we introduce two time-related metrics: average training time per task (τ_{train}) and average post-extraction training time (τ_{post}). Here, τ_{train} represents the total training time amortized across all tasks, while τ_{post} is derived by subtracting the average feature extraction time for each task (using the pre-trained model) from τ_{train}. τ_{post} is a more precise metric for evaluating algorithm-specific time consumption, as it excludes the shared preprocessing overhead. [1]

5.2 Low Latency and High Accuracy

The main CL results across various datasets, architectures, and task settings are summarized in Tables 1, 2, and 5. Our framework's key strength is achieving CL accuracy comparable to or exceeding SOTA performance with significantly lower computational costs, as measured by both τ_{train} and τ_{post}. In Table 1, using ViT-B/16, Fly-CL reduces τ_{post} by 91% on CIFAR-100 with only a marginal accuracy drop of 0.32% compared to SOTA methods. On CUB-200-2011 and VTAB, Fly-CL achieves 83% and 67% reductions in τ_{post} versus the most efficient baseline while improving overall accuracy by

[1] τ_{train} and τ_{post} are measured in seconds (wall clock time); A_t and $\bar{A}$ are measured in %.

Table 2: **Performance Comparison on Pre-trained ResNet-50 Models.** We report the average training time per task (τ_{train}), average post-extraction training time (τ_{post}), and overall accuracy ($\bar{A}$) across three benchmark datasets: CIFAR-100, CUB-200-2011, and VTAB. The best results are highlighted in **bold**.

Method	CIFAR-100			CUB-200-2011			VTAB		
	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$
RanPAC	55.68±0.97	46.65±0.88	82.72±0.22	58.74±0.84	54.35±0.99	78.72±0.40	50.15±0.36	47.94±0.38	92.80±0.40
F-OAL	80.74±0.35	71.78±0.35	66.63±0.71	5.19±0.09	1.69±0.01	60.84±1.67	2.76±0.03	0.55±0.01	26.15±2.50
Fly-CL	**14.28**±0.04	**5.25**±0.01	**84.61**±0.16	**3.90**±0.31	**0.44**±0.08	**80.25**±0.10	**2.53**±0.10	**0.34**±0.02	**94.00**±0.15

Table 3: **Performance Comparison on Pre-trained ViT-B/16 Models using Online Learning Setting.** ° denotes methods in online mode. We report the average training time per task (τ_{train}), average post-extraction training time (τ_{post}), and overall accuracy ($\bar{A}$) across three benchmark datasets: CIFAR-100, CUB-200-2011, and VTAB. The best results are highlighted in **bold**.

Method	CIFAR-100			CUB-200-2011			VTAB		
	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$
RanPAC°	1236.74±1.36	1223.56±1.07	92.48±0.31	242.54±1.56	238.36±1.47	91.89±0.26	122.89±0.46	120.69±0.42	93.41±0.57
F-OAL°	164.58±0.71	151.27±0.64	91.48±0.42	31.34±0.32	27.20±0.28	91.60±0.22	11.49±0.14	9.47±0.16	95.28±0.21
Fly-CL°	**25.46**±0.32	**12.57**±0.26	**92.96**±0.14	**6.44**±0.08	**2.33**±0.05	**92.59**±0.13	**3.17**±0.05	**1.09**±0.04	**96.38**±0.24

1.17% and 2.38% over the best-performing methods, respectively. In Table 2, with ResNet-50, Fly-CL improves overall accuracy by 1.89%, 1.53%, and 1.20% on CIFAR-100, CUB-200-2011, and VTAB, respectively, while reducing τ_{post} by 93%, 74%, and 38% versus the most efficient baselines. These improvements align with transformer-based backbone trends. Notably, F-OAL exhibits significant performance degradation on CNN backbones, presumably due to error accumulation in its iterative update mechanism, but Fly-CL does not suffer from this issue. These results highlight Fly-CL's ability to balance computational efficiency and accuracy across diverse CL scenarios, demonstrating its robustness. Results on datasets with severe domain shifts are presented in Table 6.

Additionally, the time difference between τ_{train} and τ_{post} in Tables 1 and 2 indicates that feature extraction becomes the dominant time consumer in Fly-CL. For a fair comparison with the baselines, we do not apply additional acceleration techniques here. However, in practical applications, techniques like model quantization (e.g., INT8) can further reduce feature extraction time by around 4× without significant accuracy degradation, thereby enhancing the speedup ratio. For hardware-specific deployment, frameworks like TVM (Chen et al., 2018) can be utilized to maximize efficiency.

Furthermore, Fly-CL can be easily adapted to Online CL setups by updating the $\boldsymbol{G}$ and $\boldsymbol{S}$ matrices and solving Eq. 6 for each batch, without concatenating all batch embeddings within a task. The results in Table 3 indicate that batch-mode Fly-CL remains superior to other baselines in training time and is also competitive in accuracy.

5.3 Factors Contributing to Computational Speedup

Our analysis in Sections 4.1 and 4.2 demonstrates that the proposed framework achieves significant speedup over the vanilla implementation through component-level optimizations. To quantify these improvements precisely, we split the post-extraction training time into three key components (as illustrated in Figure 2) and evaluate Fly-CL against its vanilla implementation under the CUB-200-2011 setting in Table 4. The components include: (1) Random Projection: Acceleration via weight sparsity induced by sparse projection versus the dense version. (2) Ridge Selection: Time reduction achieved by GCV, which eliminates the need for explicit cross-validation. (3) Prototype Calculation: Optimization from LU decomposition to Cholesky factorization. Additionally, the inference stage also benefits from the activation sparsity induced by the top-k operation in similarity comparisons.

5.4 Ablation Study and Hyperparameter Sensitivity Analysis

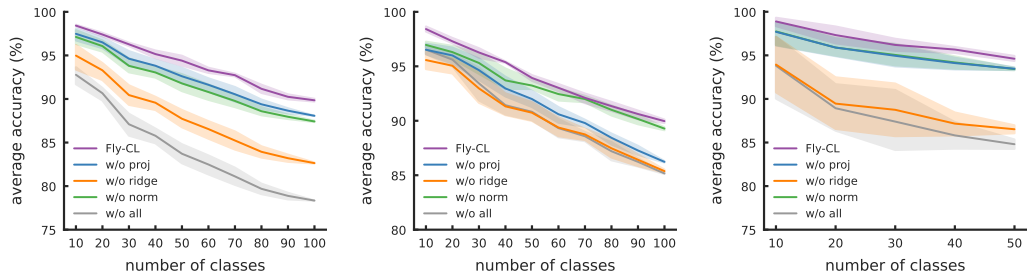
Using ViT-B/16 as the backbone, we conduct ablation studies by individually removing the projection layer (w/o proj), the streaming ridge classification (w/o ridge), and the data normalization components

Table 4: **Time Savings for Post-Extracting Components on CUB-200-2011.** We compare the theoretical time complexity per task (T_{theory}) and the actual runtime per task (T_{actual}) for each component on an NVIDIA GeForce RTX 3090 GPU. The optimized implementations demonstrate significant speedups across all components.

Method	Random Projection		Ridge Selection		Prototype Calculation		Similarity Comparison	
	T_{theory}	T_{actual}	T_{theory}	T_{actual}	T_{theory}	T_{actual}	T_{theory}	T_{actual}
vanilla	$\mathcal{O}(mn_td)$	0.22±0.03	$\mathcal{O}(lm^3)$	7.34±0.12	$\mathcal{O}(\frac{2}{3}m^3)$	0.20±0.01	$\mathcal{O}(mn_tc_t)$	0.21±0.01
optimized	$\mathcal{O}(mn_tp)$	**0.08**±0.02	$\mathcal{O}(mn_t^2)$	**0.14**±0.01	$\mathcal{O}(\frac{1}{3}m^3)$	**0.10**±0.01	$\mathcal{O}(kn_tc_t)$	**0.08**±0.01

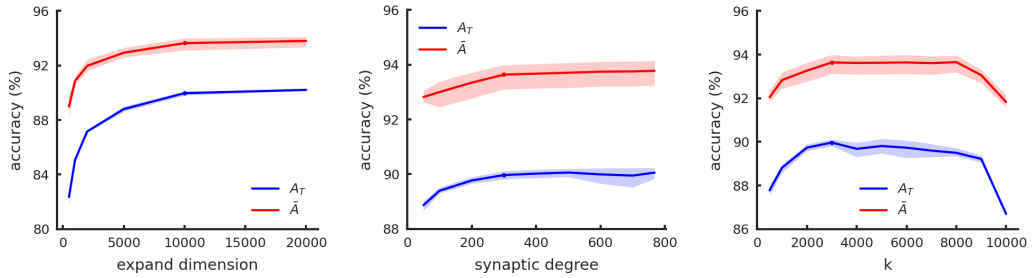
(w/o norm). The results in Figure 4 demonstrate that each component contributes significantly to overall performance. Removing any of these components, or all of them (w/o all), leads to noticeable performance degradation.

We also analyze the sensitivity of the key hyperparameters in Fly-CL: m (projection dimension), p (weight sparsity), and k (activation sparsity) in Figure 5. Increasing m improves accuracy, with performance saturating beyond $m = 10,000$. Notably, Fly-CL does not suffer from the curse of dimensionality, which can be attributed to the fact that random projection into a higher-dimensional space preserves pairwise distances between data points, as guaranteed by the Johnson-Lindenstrauss Lemma (Johnson et al., 1984). CL performance increases monotonically with p, and no significant performance drop occurs as long as p does not take an excessively small value. A sufficiently large k value avoids information loss, while a smaller value suppresses noisy dimensions. Thus, finding an appropriate trade-off can lead to optimal accuracy. Encouragingly, Figure 5(c) shows a broad plateau for optimal k selection. Based on our empirical results, we set $m = 10,000$, $p = 300$, and $k = 3,000$ as default values.



(a) CIFAR-100 (b) CUB-200-2011 (c) VTAB

Figure 4: **Accuracy Curves from Ablation Studies on Three Datasets.** We report average accuracy (A_t) for each stage. w/o refers to the removal of the specific component.



(a) Expanded dim m (b) Sparsity level p (c) Coding level k

Figure 5: **Sensitivity Analysis for Expanded dim m, Weight Sparsity p, and Activation Sparsity k on CUB-200-2011.** We report average accuracy in last task (A_T) and overall accuracy ($\bar{A}$). The dots denote the default values we use across experiments.

6 CONCLUSION

In this work, inspired by the decorrelation mechanism in the fly olfactory circuit, we propose an efficient CL framework, Fly-CL. Fly-CL significantly reduces computational overhead during training while achieving competitive performance compared to SOTA methods. This framework integrates several key components: data normalization, feature extraction, sparse random projection with top-k operation, and streaming ridge classification, each contributing to the overall efficiency and effectiveness of the system. This work establishes that neurobiological principles—particularly sparse coding and progressive decorrelation—can effectively address fundamental efficiency-accuracy trade-offs in artificial continual learning systems.

REFERENCES

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley & Sons, 2005.

Haoran Chen, Zuxuan Wu, Xintong Han, Menglin Jia, and Yu-Gang Jiang. Promptfusion: Decoupling stability and plasticity for continual learning. *arXiv preprint arXiv:2303.07223*, 2023.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022.

Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 578–594, 2018.

Stuart Coles, Joanna Bawa, Lesley Trenner, and Pat Dorazio. *An introduction to statistical modeling of extreme values*, volume 208. Springer, 2001.

Sanjoy Dasgupta, Charles F. Stevens, and Saket Navlakha. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796, 2017. doi: 10.1126/science.aam9868.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Ronald Aylmer Fisher and Leonard Henry Caleb Tippett. Limiting forms of the frequency distribution of the largest or smallest member of a sample. In *Mathematical proceedings of the Cambridge philosophical society*, volume 24, pp. 180–190. Cambridge University Press, 1928.

Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. In *ICCV*, pp. 11483–11493, October 2023.

Gene H Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

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.

Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8340–8349, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15262–15271, 2021b.

Toshihide Hige, Yoshinori Aso, Gerald M Rubin, and Glenn C Turner. Plasticity-driven individualization of olfactory coding in mushroom body output neurons. *Nature*, 526(7572):258–262, 2015.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pp. 709–727. Springer, 2022.

William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Dahuin Jung, Dongyoon Han, Jihwan Bang, and Hwanjun Song. Generating instance-level prompts for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11847–11857, 2023.

Pentti Kanerva. *Sparse distributed memory*. MIT press, 1988.

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.

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

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

Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems*, 35: 109–123, 2022.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23638–23647, 2024.

Yuchen Liang, Chaitanya K Ryali, Benjamin Hoover, Leopold Grinberg, Saket Navlakha, Mohammed J Zaki, and Dmitry Krotov. Can a fruit fly learn word embeddings? *arXiv preprint arXiv:2101.06887*, 2021.

Andrew C Lin, Alexei M Bygrave, Alix De Calignon, Tzumin Lee, and Gero Miesenböck. Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nature neuroscience*, 17(4):559–568, 2014.

Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and LF Abbott. Optimal degrees of synaptic connectivity. *Neuron*, 93(5):1153–1164, 2017.

Mark D McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *arXiv preprint arXiv:2307.02251*, 2023.

Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems (TODS)*, 31(3):1095–1133, 2006.

Maria Papadopoulou, Stijn Cassenaer, Thomas Nowotny, and Gilles Laurent. Normalization for sparse encoding of odors by a wide-field interneuron. *Science*, 332(6030):721–725, 2011.

Ameya Prabhu, Shiven Sinha, Ponnurangam Kumaraguru, Philip HS Torr, Ozan Sener, and Puneet K Dokania. Random representations outperform online continually learned representations. *arXiv preprint arXiv:2402.08823*, 2024.

Parikshit Ram and Kaushik Sinha. Federated nearest neighbor classification with a colony of fruit-flies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8036–8044, 2022.

Chaitanya Ryali, John Hopfield, Leopold Grinberg, and Dmitry Krotov. Bio-inspired hashing for unsupervised similarity search. In *International conference on machine learning*, pp. 8295–8306. PMLR, 2020.

Jaiyam Sharma and Saket Navlakha. Improving similarity search with high-dimensional locality-sensitive hashing. *arXiv preprint arXiv:1812.01844*, 2018.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *CVPR*, pp. 11909–11919, 2023.

Charles F Stevens. What the fly's nose tells the fly's brain. *Proceedings of the National Academy of Sciences*, 112(30):9460–9465, 2015.

Hai-Long Sun, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Pilot: A pre-trained model-based continual learning toolbox. *arXiv preprint arXiv:2309.07117*, 2023.

Hai-Long Sun, Da-Wei Zhou, Hanbin Zhao, Le Gan, De-Chuan Zhan, and Han-Jia Ye. Mos: Model surgery for pre-trained model-based class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20699–20707, 2025.

Yu-Ming Tang, Yi-Xing Peng, and Wei-Shi Zheng. When prompt-based incremental learning does not meet strong pretraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1706–1716, 2023.

Terence Tao and Van Vu. On random ± 1 matrices: singularity and determinant. *Random Struct. Algor.*, 28(1):1–23, 2006.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10087–10098, 2025.

Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. *arXiv preprint arXiv:2310.07234*, 2023a.

Yabin Wang, Zhiheng Ma, Zhiwu Huang, Yaowei Wang, Zhou Su, and Xiaopeng Hong. Isolation and impartial aggregation: A paradigm of incremental learning without interference. In *AAAI*, volume 37, pp. 10209–10217, 2023b.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *arXiv preprint arXiv:2204.04799*, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, pp. 139–149, 2022b.

Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23219–23230, 2024.

Yunliang Zang and Erik De Schutter. Recent data on the cerebellum require new models and theories. *Current Opinion in Neurobiology*, 2023.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *arXiv preprint arXiv:2303.07338*, 2023a.

Da-Wei Zhou, Yuanhan Zhang, Jingyi Ning, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Learning without forgetting for vision-language models. *arXiv preprint arXiv:2305.19270*, 2023b.

Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *CVPR*, 2024.

Huiping Zhuang, Yuchen Liu, Run He, Kai Tong, Ziqian Zeng, Cen Chen, Yi Wang, and Lap-Pui Chau. F-oal: Forward-only online analytic learning with fast training and low memory footprint in class incremental learning. *Advances in Neural Information Processing Systems*, 37:41517–41538, 2024.

A Algorithm Pseudocode

Algorithm 1 Fly-CL Training Pipeline

Input: Sequentially arriving data $\mathcal{D}_t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t}$ where $t = 1, \ldots, T$. Pre-trained encoder f_θ. Projection operator $Z(\cdot) : \mathbb{R}^d \to \mathbb{R}^m$. Penalty coefficient candidates $\Lambda = \{\lambda_1, \ldots, \lambda_l\}$. Zero-initialized matrices $\boldsymbol{G}_0 \in \mathbb{R}^{m \times m}$ and $\boldsymbol{S}_0 \in \mathbb{R}^{m \times c_t}$.
Output: Modulated Prototypes $\boldsymbol{C} \in \mathbb{R}^{m \times c_t}$.

for $t = 1, \ldots, T$ **do**
  $r = \min(n_t, m)$
  Get compressed embedding for each datum $\boldsymbol{v}_i^t = f_\theta(\boldsymbol{x}_i^t) \in \mathbb{R}^d$
  Transform to high-dim sparse embedding $Z(\boldsymbol{v}_i^t) = \text{top-}k\,(\boldsymbol{W}\boldsymbol{v}_i^t)$ ▷ $\mathcal{O}(mn_tp)$
  Concatenate $Z(\boldsymbol{v}_i^t)$ to get $\boldsymbol{H}_t \in \mathbb{R}^{n_t \times m}$
  $\boldsymbol{G}_t \leftarrow \boldsymbol{G}_{t-1} + \boldsymbol{H}_t^\top \boldsymbol{H}_t$ ▷ $\mathcal{O}(n_tm^2)$
  $\boldsymbol{S}_t \leftarrow \boldsymbol{S}_{t-1} + \boldsymbol{H}_t^\top \boldsymbol{Y}_t$ ▷ $\mathcal{O}(n_tc_tm)$
  $\boldsymbol{U}_t, \boldsymbol{\Sigma}_t, \boldsymbol{V}_t = \text{svd}(\boldsymbol{H}_t)$ ▷ $\mathcal{O}(n_trm)$
  for $\lambda \in \Lambda$ **do**
    $\boldsymbol{D}_t = \frac{\boldsymbol{\Sigma}_t^2}{\boldsymbol{\Sigma}_t^2 + \lambda \boldsymbol{I}_r}$ ▷ $\mathcal{O}(lr)$
    $\text{df}(\lambda) = \text{tr}(\boldsymbol{D}_t) = \sum_{i=1}^r \frac{s_i^2}{s_i^2+\lambda}$ ▷ $\mathcal{O}(lr)$
    $\hat{\boldsymbol{Y}}_t = \boldsymbol{U}_t(\text{vecdiag}(\boldsymbol{D}_t) \otimes \boldsymbol{1}_c^\top) \odot \boldsymbol{U}_t^\top \boldsymbol{Y}_t$ ▷ $\mathcal{O}(ln_trc_t)$
    $\text{GCV}(\lambda) = \frac{\|\boldsymbol{Y}_t - \hat{\boldsymbol{Y}}_t(\lambda)\|_F^2}{n_t\left(1 - \frac{\text{df}(\lambda)}{n_t}\right)^2}$ ▷ $\mathcal{O}(ln_tc_t)$
  end for
  Select $\lambda_t^* = \arg\min_{\lambda \in \Lambda} \text{GCV}(\lambda)$
  $\boldsymbol{L}_t \boldsymbol{L}_t^\top = \boldsymbol{G}_t + \lambda_t^* \boldsymbol{I}_m$ ▷ $\mathcal{O}(\frac{1}{3}m^3)$
  $\boldsymbol{C}_t = \boldsymbol{L}_t^{-\top}(\boldsymbol{L}_t^{-1}\boldsymbol{S}_t)$ ▷ $\mathcal{O}(m^2c_t)$
end for

Algorithm 2 Fly-CL Inference Pipeline

Input: Sequentially arriving data $\mathcal{D}_t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t}$ where $t = 1, \ldots, T$. Pre-trained encoder f_θ. Projection operator Z. Modulated class prototypes $\boldsymbol{C}_t \in \mathbb{R}^{m \times c_t}$.
Output: Predicted labels $\hat{y}$.

Get compressed embedding for each datum $\boldsymbol{v} = f_\theta(\boldsymbol{x}) \in \mathbb{R}^d$
Transform to high-dim sparse embedding $Z(\boldsymbol{v}) = \text{top-}k\,(\boldsymbol{W}\boldsymbol{v})$ ▷ $\mathcal{O}(mp)$
Compute prediction $\hat{y} = Z(\boldsymbol{v})^\top \boldsymbol{C}_t$ ▷ $\mathcal{O}(kc_t)$

B Complete Theoretical Analysis

In this section, we present a comprehensive theoretical analysis of the sparsification effects on both the weights and activations in the random projection operation, and demonstrate the consistency between the biologically plausible Hebbian learning rule and ridge classification in modeling the KC-MBON transformation..

B.1 Information Preserving for Sparse Connections in Random Projection Matrix

A common approach to demonstrate that sparse random matrix multiplication preserves information equivalently to its dense counterpart lies in proving the matrix's near-preservation of full column rank. For our sparse random matrix $\boldsymbol{W} \in \mathbb{R}^{m \times d}$ where $m > d$, we prove that $\boldsymbol{W}$ almost surely maintains rank d.

Theorem B.1. *Given the matrix $\boldsymbol{W} \in \mathbb{R}^{m \times d}$, where $m > d$, with each row having exactly p non-zero entries, which are randomly sampled from $\mathcal{N}(0, 1)$. Let $\mathcal{W} \in \mathbb{R}^{d \times d}$ be any square submatrix of $\boldsymbol{W}$.*

Then, for any $\epsilon > 0$, it holds that

$$\mathbb{P}\left(|\det(\mathcal{W})| \geq \left(\frac{p}{d}\right)^{d/2} \sqrt{d!}\exp(-d^{1/2+\epsilon})\right) = 1 - o(1).$$

Thus, for sufficiently large p and d, any submatrix $\mathcal{W}$ is invertible with probability at least $1 - o(1)$.

Proof. According to aformentioned definition, we have $\mathbb{E}(\boldsymbol{W}_{ij}) = 0$ and $\mathrm{Var}(\boldsymbol{W}_{ij}) = \frac{p}{d}$. Considering $\boldsymbol{R} = \frac{1}{\sigma}\mathcal{W}$, which satisfies $\mathbb{E}(\boldsymbol{R}_{ij}) = 0$, $\mathrm{Var}(\boldsymbol{R}_{ij}) = 1$, we can conclude, based on (Tao & Vu, 2006, Theorem 8.9), that

$$\mathbb{P}\left(|\det(\boldsymbol{R})| \geq \sqrt{d!}\exp(-d^{1/2+\epsilon})\right) = 1 - o(1). \tag{13}$$

By using $\sigma = \sqrt{\frac{p}{d}}$ and $\det(\boldsymbol{R}) = \sigma^{-d}\det(\mathcal{W})$ for substitution, we complete the proof. □

To better demonstrate the information preserving property of our construction, another line of validation is to utilize the random matrix's distance preservation property, which is theoretically guaranteed by the Johnson-Lindenstrauss (JL) lemma (Johnson et al., 1984). Specifically, we conduct an empirical simulation to verify that the normalization of our sparse projection matrix $\boldsymbol{\Phi} = \sqrt{\frac{d}{mp}}\boldsymbol{W}$ preserves pairwise Euclidean distances between feature vectors with high probability. For every pair of vectors $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, we calculate the Distortion Ratio defined as Ratio $= \frac{\|\boldsymbol{\Phi}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\|_2^2}{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}$. We randomly select 500 data points from the extracted features of CIFAR-100 dataset and compute the pairwise distances between all point pairs, with the results illustrated in Figure 6. By analyzing the distribution of these ratios across all vector pairs, the resulting histogram shows a strong concentration around the ideal value of 1. If we set ϵ to 0.03, then the vast majority of the ratios are empirically confined within the bounds of $[1 - \epsilon, 1 + \epsilon]$. This concentration strongly validates that our sparse projection structure-comprising only p non-zero $\mathcal{N}(0, 1)$ entries per row-effectively maintains the geometric structure of the high-dimensional data.

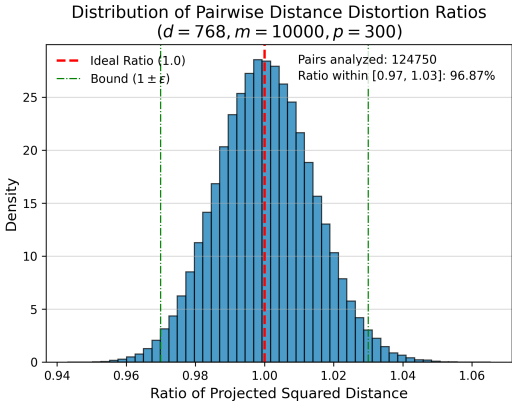


Figure 6: Empirical Verification of the Johnson-Lindenstrauss (JL) Property using Sparse Random Projection.

B.2 Robustness of Top-k Sparsification on High-Dimensional Embeddings

Let the high-dimensional embedding vector be $\boldsymbol{h} \in \mathbb{R}^m$. After applying the top-k operation, we obtain a sparsified vector $\boldsymbol{h}' \in \mathbb{R}^m$, where only the k largest absolute values in $\boldsymbol{h}$ are retained, and the remaining elements are set to zero. We aim to prove that when $k = \Omega(m^\alpha)$ (with $0 < \alpha < 1$, i.e., not overly sparse), the performance degradation is negligible.

According to statistic learning theory (Bartlett & Mendelson, 2002) and extreme value theory (Coles et al., 2001; Fisher & Tippett, 1928), we start with the following two widely-accepted assumption:

Assumption B.2. *Assume that the "energy" (squared ℓ_2-norm) of the embedding vector x is concentrated in a few dimensions, i.e., there exists a constant $C > 0$ such that:*

$$\mathbb{E}\left[\frac{\sum_{i=1}^k \boldsymbol{h}_i^2}{\|\boldsymbol{h}\|_2^2}\right] \geq 1 - \frac{C}{k},$$

where where $\boldsymbol{h}_i$ denotes the i-th largest value in $\boldsymbol{h}$.

Assumption B.3. *Assume that the performance loss function $L(\boldsymbol{h}, y)$ of the downstream task (e.g., classifier) is Lipschitz continuous with respect to input perturbations, i.e., there exists a constant $M > 0$ such that:*

$$|L(\boldsymbol{h}, y) - L(\boldsymbol{h}', y)| \leq M \cdot \|\boldsymbol{h} - \boldsymbol{h}'\|.$$

Regarding the approximation error of top-k operation, we show it is bounded.

Theorem B.4. *Under the Assumption B.2, the sparsification error satisfies:*

$$E\left[\|\boldsymbol{h}-\boldsymbol{h}'\|_2^2\right] \leq \frac{C}{k}\cdot\mathbb{E}\left[\|\boldsymbol{h}\|_2^2\right].$$

Proof. By Assumption B.2:

$$\mathbb{E}\left[\sum_{i=k+1}^{d}\boldsymbol{h}_i^2\right] \leq \frac{C}{k}\cdot\mathbb{E}\left[\|\boldsymbol{h}\|_2^2\right].$$

Thus,

$$\mathbb{E}\left[\|\boldsymbol{h}-\boldsymbol{h}'\|_2^2\right] = \mathbb{E}\left[\sum_{i=k+1}^{m}\boldsymbol{h}_i^2\right] \leq \frac{C}{k}\cdot\mathbb{E}\left[\|\boldsymbol{h}\|_2^2\right].$$

□

Then, we can quantify the upper bound of possible performance degradation.

Theorem B.5. *Under Assumption B.3, the performance degradation due to sparsification satisfies:*

$$\mathbb{E}\left[|L(\boldsymbol{h},y)-L(\boldsymbol{h}',y)|\right] \leq M\cdot\sqrt{\frac{C}{k}\cdot\mathbb{E}[\|\boldsymbol{h}\|_2^2]}$$

Proof. By the Cauchy-Schwarz inequality and Theorem B.4, we can derive that

$$\begin{aligned}\mathbb{E}\left[|L(\boldsymbol{h},y)-L(\boldsymbol{h}',y)|\right] &\leq M\cdot\mathbb{E}\left[\|\boldsymbol{h}-\boldsymbol{h}'\|_2\right]\\ &\leq M\cdot\sqrt{\mathbb{E}\left[\|\boldsymbol{h}-\boldsymbol{h}'\|_2^2\right]}\\ &\leq M\cdot\sqrt{\frac{C}{k}\cdot\mathbb{E}\left[\|\boldsymbol{h}\|_2^2\right]}.\end{aligned}$$

□

From Theorem B.5, we establish in Theorem B.6 that moderate sparsity does not result in significant performance degradation, as the error decreases exponentially with the increasing expanded dimension m.

Theorem B.6. *For top-k sparsification in the expanded dimension m, the performance degradation is bounded by:*

$$\mathbb{E}\left[|L(\boldsymbol{h},y)-L(\boldsymbol{h}',y)|\right] \leq M\cdot\sqrt{\frac{C}{k}\cdot\mathbb{E}[\|\boldsymbol{h}\|_2^2]},$$

where $L(\cdot)$ is a performance loss function for downstream tasks and C, M are constants. To ensure negligible performance degradation, we require:

$$\sqrt{\frac{C}{k}\cdot\mathbb{E}[\|\boldsymbol{h}\|_2^2]} \leq \mathcal{O}\left(\frac{1}{\sqrt{m^{\alpha}}}\right),$$

i.e., when $k=\Omega(m^{\alpha})$ $(0<\alpha<1)$, the error bound decays exponentially with increasing dimension.

For example:

- If $k=\Omega(d^{0.5})$, the performance degradation is $\mathcal{O}(d^{-0.25})$.
- If $k=\Omega(d^{0.8})$, the performance degradation is $\mathcal{O}(d^{-0.4})$.

B.3 Consistency between Ridge Classification and Hebbian Learning Rules

In this section, we show that a biologically plausible *local learning rule*—constructed from Hebbian and anti-Hebbian plasticity (Hebb, 2005)—converges to the same stationary point as ridge classification. This provides a theoretical bridge between synaptic dynamics in the KC→MBON pathway and the streaming ridge classification algorithm used in Fly-CL.

We begin by defining a synaptic update rule that depends only on variables locally available at each synapse. Let $\boldsymbol{h}' \in \mathbb{R}^m$ denote the pre-synaptic activity vector and $\hat{\boldsymbol{y}} = \boldsymbol{C}_t^\top \boldsymbol{h}'$ the post-synaptic response predicted by the current weights $\boldsymbol{C}_t \in \mathbb{R}^{m \times c_t}$. The synaptic update is defined as:

$$\Delta \boldsymbol{C} = \boldsymbol{C}_{t+1} - \boldsymbol{C}_t = \eta\big(\boldsymbol{h}'\boldsymbol{y}^\top \; - \; \boldsymbol{h}'\hat{\boldsymbol{y}}^\top \; - \; \lambda \boldsymbol{C}_t\big), \tag{14}$$

where $\eta > 0$ is a small learning rate, $\boldsymbol{y} \in \mathbb{R}^{c_t}$ is the one-hot ground-truth label. This rule consists of three biologically meaningful components:

1. **Hebbian term** $\boldsymbol{h}'\boldsymbol{y}^\top$, which strengthens synapses when both pre-synaptic activity and the target post-synaptic signal co-activate.
2. **Anti-Hebbian term** $-\boldsymbol{h}'\hat{\boldsymbol{y}}^\top$, which suppresses correlations between the input and the model's own prediction, implementing an error-correcting mechanism.
3. **Weight decay** $-\lambda \boldsymbol{C}_t$, corresponding to metabolic cost or homeostatic constraints.

This formulation is biologically plausible: it requires only pre-synaptic activity, post-synaptic activity, and a modulatory teaching signal—i.e., a standard *three-factor learning rule* commonly observed in neuromodulated plasticity.

Taking the expectation over data samples $(\boldsymbol{h}', \boldsymbol{y})$, we obtain:

$$\mathbb{E}[\Delta \boldsymbol{C}] = \eta\big(\mathbb{E}[\boldsymbol{h}'\boldsymbol{y}^\top] - \mathbb{E}[\,\boldsymbol{h}'\hat{\boldsymbol{y}}^\top\,] - \lambda \boldsymbol{C}_t\big) = \eta\big(\boldsymbol{S}_t - \boldsymbol{G}_t \boldsymbol{C}_t - \lambda \boldsymbol{C}_t\big), \tag{15}$$

where we define

$$\boldsymbol{G}_t = \mathbb{E}[\boldsymbol{h}'\boldsymbol{h}'^\top] \in \mathbb{R}^{m \times m}, \qquad \boldsymbol{S}_t = \mathbb{E}[\boldsymbol{h}'\boldsymbol{y}^\top] \in \mathbb{R}^{m \times c_t}.$$

Thus, the expected synaptic dynamics follow the linear recursion:

$$\boldsymbol{C}_{t+1} = \boldsymbol{C}_t + \eta\big(\boldsymbol{S}_t - (\boldsymbol{G}_t + \lambda \boldsymbol{I}_m)\boldsymbol{C}_t\big). \tag{16}$$

Taking the continuous-time limit $\eta \to 0$ yields the ordinary differential equation:

$$\frac{d\boldsymbol{C}_t}{dt} = \kappa\big(\boldsymbol{S}_t - (\boldsymbol{G}_t + \lambda \boldsymbol{I}_m)\boldsymbol{C}_t\big), \tag{17}$$

where $\kappa > 0$ rescales time. The stationary solution satisfies:

$$(\boldsymbol{G}_t + \lambda \boldsymbol{I}_m)\boldsymbol{C}_t^\star = \boldsymbol{S}_t,$$

and therefore:

$$\boldsymbol{C}_t^\star = (\boldsymbol{G}_t + \lambda \boldsymbol{I}_m)^{-1}\boldsymbol{S}_t, \tag{18}$$

which is *exactly* the closed-form solution of ridge regression.

In our representation-based CIL setting, since all samples in the same task are available simultaneously, we do not need to integrate the continuous-time dynamics in Eq. 17. Instead, we can directly compute the ridge solution to obtain the optimal stationary classifier.

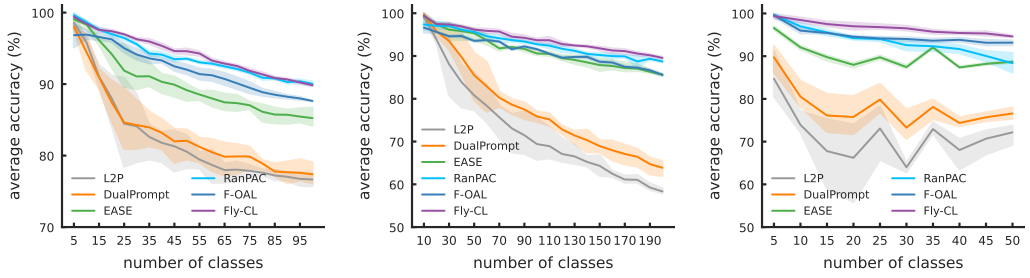
C Additional Results

C.1 Experiments in Longer Task Sequences

To evaluate the long-term stability of Fly-CL, we conduct experiments with task sequences twice as long as those in Table 1. Results are presented in Table 5 and Figure 7. Overall, both τ_{train} and τ_{post} are shorter than those in Table 1 due to fewer samples per task. Fly-CL improves overall accuracy by 0.54%, 1.21%, and 1.58% compared to SOTA methods, while significantly reducing average post-extraction training time by 89%, 74%, and 59% compared to the most efficient baselines. These results are consistent with the trends observed in Table 1 and Figure 8, demonstrating the robustness of Fly-CL across different task lengths.

Table 5: **Performance Comparison on Pre-trained ViT-B/16 Models with Longer Task Sequence.** We report the average training time per task (τ_{train}), average post-extraction training time (τ_{post}), and overall accuracy ($\bar{A}$) across three benchmark datasets: CIFAR-100, CUB-200-2011, and VTAB. The best results are highlighted in **bold**.

Method	CIFAR-100			CUB-200-2011			VTAB		
	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$
L2P	147.47±0.36	107.59±0.26	82.69±0.91	30.93±0.25	23.39±0.26	72.83±1.45	32.17±0.39	29.31±0.38	71.84±1.42
Dualprompt	130.12±0.09	91.05±0.07	83.42±0.86	27.66±0.18	20.28±0.18	77.93±0.91	29.44±0.24	26.62±0.23	78.46±1.14
InfLoRA	124.80±0.39	84.36±0.12	88.18±0.34	27.12±0.14	19.85±0.09	75.67±0.16	29.71±0.21	26.92±0.17	81.09±0.73
SEMA	128.95±0.46	88.57±0.19	88.93±0.52	27.98±0.25	20.52±0.18	80.82±0.11	30.36±0.27	27.55±0.20	84.56±0.48
MoE-Adapter	132.67±0.62	92.31±0.45	88.42±0.28	30.25±0.19	22.82±0.15	78.62±0.35	29.25±0.18	26.81±0.13	83.24±0.57
EASE	350.92±0.39	331.76±0.29	90.14±0.51	83.38±0.22	75.05±0.16	91.47±0.65	64.68±0.73	57.24±0.62	90.26±0.41
RanPAC	44.53±0.58	37.03±0.59	93.68±0.24	20.68±0.32	18.18±0.31	92.65±0.26	41.49±0.74	40.02±0.59	93.62±0.32
F-OAL	16.32±0.02	8.91±0.02	92.63±0.46	3.65±0.02	1.10±0.01	91.48±0.23	2.17±0.16	0.71±0.03	94.96±0.27
Fly-CL	**8.31**±0.04	**1.00**±0.01	**94.22**±0.09	**2.76**±0.06	**0.29**±0.01	**93.86**±0.27	**1.70**±0.10	**0.29**±0.02	**96.54**±0.38



(a) CIFAR-100 (b) CUB-200-2011 (c) VTAB

Figure 7: **Accuracy Curves of Different Methods on Pre-trained ViT-B/16 with Longer Task Sequence.** The average accuracy (A_t) is reported for each dataset. These results align with and extend the quantitative analysis presented in Table 5.

C.2 Experiments on Datasets with Severe Domain Shift

Table 6 summarizes the results on ImageNet-R and ImageNet-A under severe domain shift. Compared with existing continual learning baselines, Fly-CL achieves the best overall accuracy on both datasets (83.19% on ImageNet-R and 67.98% on ImageNet-A), comparable with the previous SOTA RanPAC. More importantly, Fly-CL attains these improvements with substantially lower computation cost. Its average training time per task is reduced by an order of magnitude compared to prompt-based methods (e.g., L2P, DualPrompt) and much faster than EASE, while its post-extraction training time is almost negligible (0.21s vs. 67.71s for RanPAC on ImageNet-R). These results demonstrate that Fly-CL is not only robust to severe distribution shifts but also highly efficient, making it especially suitable for practical continual learning scenarios where both accuracy and efficiency are critical.

C.3 Data Normalization Strategy

While data normalization is a well-established technique for improving classification performance in i.i.d. scenarios, its effectiveness in facilitating CL with frozen pre-trained encoders remains unclear. Our results indicate that applying proper architecture-specific normalization to input images significantly improves the learning performance compared to baseline CL methods (Table 7). The optimal normalization strategies for the included backbones differ. Across all tested datasets, ViT-B/16 (Dosovitskiy et al., 2020) benefits more from standard normalization that projects inputs into the $[-1, 1]$ range, while ResNet-50 (He et al., 2016) achieves optimal performance when normalized using ImageNet statistics.

We hypothesize that the imporved performance arises from a reduced feature distribution shift across tasks. Proper normalization preserves the geometry of the pre-trained feature manifold, which is crucial for prototype-based classification, where cosine similarity measures depend on the angular

Table 6: **Performance Comparison on Pre-trained ViT-B/16 Models with Severe Domain Shift.** We report the average training time per task (τ_{train}), average post-extraction training time (τ_{post}), and overall accuracy ($\bar{A}$) across two benchmark datasets: ImageNet-R and ImageNet-A. The best results are highlighted in **bold**.

Method	ImageNet-R			ImageNet-A		
	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$	$\tau_{\text{train}}(\downarrow)$	$\tau_{\text{post}}(\downarrow)$	$\bar{A}(\uparrow)$
L2P	131.97±0.46	110.56±0.42	76.13±0.21	56.28±0.32	48.92±0.27	48.86±0.08
Dualprompt	117.80±0.34	96.58±0.30	73.92±0.46	49.60±0.28	42.31±0.26	57.05±0.13
InfLoRA	62.32±0.33	41.05±0.23	82.15±0.41	25.71±0.30	18.50±0.24	62.32±0.28
SEMA	68.85±0.21	47.34±0.19	81.89±0.17	28.65±0.26	21.23±0.22	61.79±0.32
MoE-Adapter	66.37±0.29	45.02±0.24	81.76±0.33	26.27±0.41	18.89±0.37	61.72±0.21
EASE	311.00±0.29	274.36±0.25	81.69±0.24	80.80±0.19	73.47±0.22	65.03±0.28
RanPAC	76.25±0.35	67.71±0.28	83.02±0.12	32.43±0.13	28.86±0.11	67.28±0.09
F-OAL	16.51±0.11	8.80±0.04	80.62±0.25	3.99±0.07	1.05±0.02	63.99±0.30
Fly-CL	**7.55**±0.04	**0.21**±0.02	**83.19**±0.14	**3.10**±0.03	**0.15**±0.01	**67.98**±0.17

relationships between features. Our empirical results suggest that input normalization may serve as a fundamental defense against forgetting by anchoring the feature space topology to the original pre-training distribution.

Table 7: **Comparison of CL Performance across Pre-trained Models and Normalization Strategies.** We report overall accuracy ($\bar{A}$). Normalization methods includes: "None" (no data normalization), "ImageNet" (ImageNet statistics), and "Standard"(scaled to the $[-1, 1]$).

Backbone	CIFAR			CUB			VTAB		
	None	ImageNet	Standard	None	ImageNet	Standard	None	ImageNet	Standard
ViT-B/16	91.64±0.62	87.87±0.62	**93.89**±0.12	93.04±0.37	90.68±0.42	**93.84**±0.18	95.26±0.68	95.47±0.52	**96.54**±0.38
ResNet-50	80.66±0.48	**84.61**±0.16	83.09±0.48	75.08±1.23	**80.25**±0.10	76.78±1.08	92.45±0.71	**94.00**±0.15	92.76±0.54

C.4 Memory Consumption

We also compare the memory consumption of Fly-CL against other methods in Table 8 using ViT-B/16 with the same task sequence as in Table 1. For fairness, we use a batch size of 128 across all methods and datasets. The results show Fly-CL also has the minimal memory cost, strengthening the efficiency of our method.

Table 8: **Memory Usage (GB) of Different Methods on Pre-trained ViT-B/16.** We report highest peak memory usage of each methods. The best results are highlighted in **bold**.

Method	CIFAR-100	CUB-200-2011	VTAB
L2P	16.4GB	16.4GB	16.4GB
DualPrompt	13.6GB	13.6GB	13.6GB
InfLoRA	14.1GB	14.1GB	14.1GB
SEMA	12.9GB	12.9GB	12.9GB
MoE-Adapter	20.2GB	20.2GB	20.2GB
EASE	12.2GB	12.2GB	12.2GB
RanPAC	12.2GB	12.2GB	22.8GB
F-OAL	12.2GB	4.9GB	4.5GB
Fly-CL	**6.7GB**	**4.6GB**	**4.3GB**

C.5 Memory-time tradeoff in high-dim projections

We present the trade-off between memory, training time, and overall accuracy in Table 9. The overall accuracy gradually saturates as the dimension increases, while memory and training time

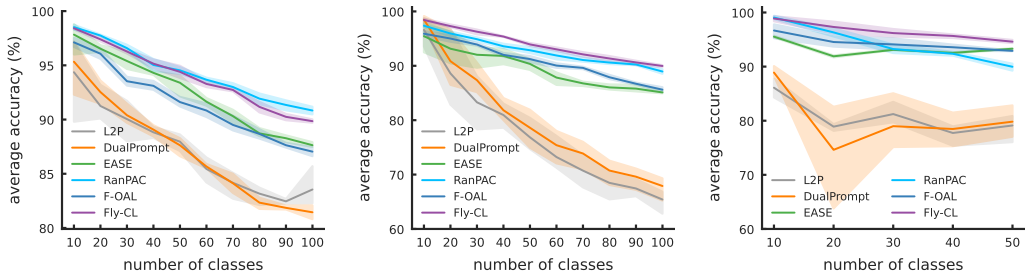
grow quadratically. Therefore, we chose 10,000 as the dimension in our simulations. As long as the dimension does not exceed 10,000, both memory and training time consumption remain lower than those of previous methods, as summarized in Table 1.

Table 9: **Memory-Time-Accuracy comparison with increasing projection dimension.** It's conducted on the CUB dataset using ViT B/16. We report highest peak memory usage of each methods. The best results are highlighted in **bold**.

Dimension	1000	2000	5000	10000	20000
Memory	2.8G	2.8G	3.0G	4.6G	9.3G
τ_{train}	4.19±0.02	4.20±0.05	4.25±0.07	4.43±0.11	5.13±0.08
$\bar{A}$	90.87±0.49	91.97±0.52	92.93±0.41	93.84±0.18	93.90±0.52

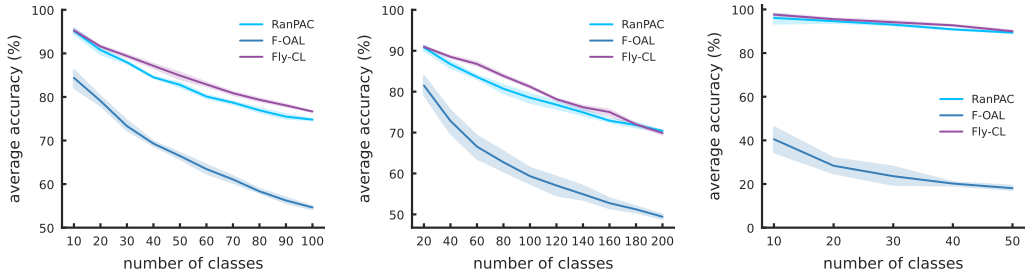
C.6 Additional Visualized Figures and Evaluation Metric During the Training Process

Here, we present a more detailed breakdown of the training processes for ViT-B/16 and ResNet-50. Results from Tables 1 and 2 are visualized in Figures 8 and 9. We list the average accuracy of different methods at different stages across three datasets. We additionally report the last-stage accuracy (A_T)



(a) CIFAR-100 (b) CUB-200-2011 (c) VTAB

Figure 8: **Accuracy Curves of Different Methods on Pre-trained ViT-B/16.** The average accuracy (A_t) is reported for each dataset. These results align with and extend the quantitative analysis presented in Table 1.



(a) CIFAR-100 (b) CUB-200-2011 (c) VTAB

Figure 9: **Accuracy Curves of Different Methods on Pre-trained ResNet-50.** The average accuracy (A_t) is reported for each dataset. These results align with and extend the quantitative analysis presented in Table 2.

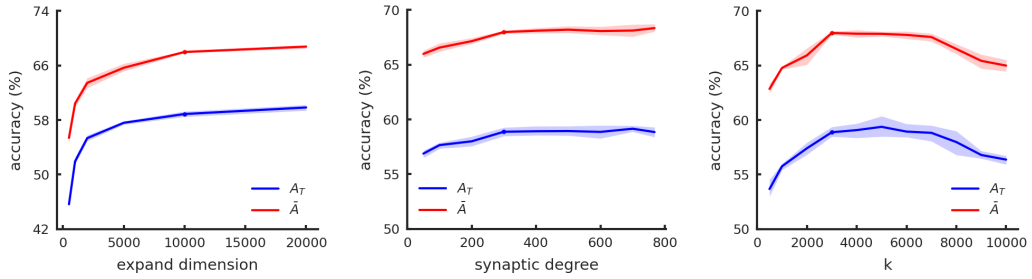
and the backward transfer score (a representative forgetting metric) of Table 1 in Table 10 to provide a more comprehensive evaluation.

Table 10: **Performance Comparison on Pre-trained ViT-B/16 Models.** We report the last stage accuracy (A_T), and backward transfer (BWT) across three benchmark datasets: CIFAR-100, CUB-200-2011, and VTAB.

Method	CIFAR-100		CUB-200-2011		VTAB	
	$A_T(\uparrow)$	$BWT(\uparrow)$	$A_T(\uparrow)$	$BWT(\uparrow)$	$A_T(\uparrow)$	$BWT(\uparrow)$
L2P	83.55±1.53	-6.23±0.41	65.41±1.84	-13.14±0.57	79.12±2.18	-8.04±0.78
DualPrompt	81.45±0.52	-7.06±0.20	67.90±1.89	-12.12±0.53	79.83±2.37	-7.63±0.75
InfLoRA	86.56±0.46	-5.07±0.16	69.45±0.56	-11.53±0.25	87.88±0.73	-4.59±0.24
SEMA	87.47±0.43	-4.70±0.13	73.66±0.36	-9.94±0.19	89.28±0.60	-4.06±0.22
MoE-Adapter	86.88±0.32	-4.97±0.15	68.11±0.41	-12.11±0.23	88.06±0.48	-4.51±0.19
EASE	87.63±0.20	-4.66±0.06	85.10±0.19	-5.68±0.08	93.30±0.07	-2.48±0.04
RanPAC	**90.83±0.41**	**-3.47±0.11**	88.95±0.48	-4.16±0.22	89.97±0.71	-3.79±0.26
F-OAL	87.04±0.50	-4.90±0.14	85.61±0.50	-5.43±0.24	92.91±0.07	-2.66±0.04
Fly-CL	89.85±0.17	-3.82±0.09	**89.97±0.18**	**-3.80±0.07**	**94.61±0.35**	**-2.01±0.15**

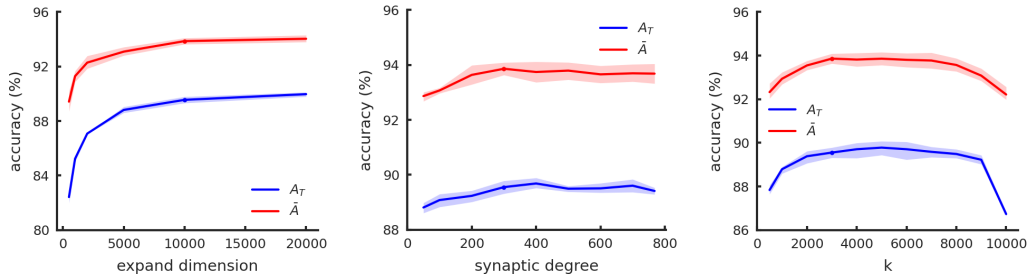
C.7 Additional Sensitivity Analysis Across Task Complexity, Number of Tasks/Classes, and Different Pretrained Backbones

We further conduct additional sensitivity analyses under three complementary settings: (i) ImageNet-A with 10 tasks and 20 classes per task using ViT-B/16 to examine the effect of task complexity (Figure 10); (ii) CUB-200-2011 with 20 tasks and 10 classes per task using ViT-B/16 to evaluate the impact of a larger number of tasks and classes (Figure 11); and (iii) CUB-200-2011 with 10 tasks and 20 classes per task using ResNet-50 to assess the influence of different backbone architectures (Figure 12). Across all settings, the observed trends remain consistent with those reported in Figure 5.



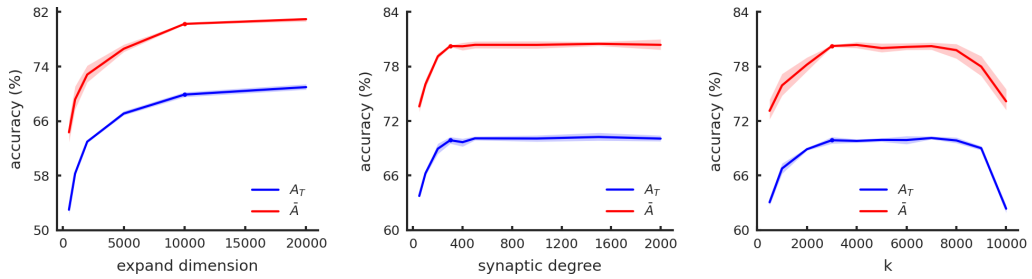
(a) Expanded dim m (b) Sparsity level p (c) Coding level k

Figure 10: **Sensitivity Analysis for Expanded dim m, Weight Sparsity p, and Activation Sparsity k on ImageNet-A.** We report average accuracy in last task (A_T) and overall accuracy ($\bar{A}$). The dots denote the default values we use across experiments.



(a) Expanded dim m (b) Sparsity level p (c) Coding level k

Figure 11: **Sensitivity Analysis for Expanded dim m, Weight Sparsity p, and Activation Sparsity k on CUB-200 with Longer Task Sequence.** We report average accuracy in last task (A_T) and overall accuracy ($\bar{A}$). The dots denote the default values we use across experiments.



(a) Expanded dim m (b) Sparsity level p (c) Coding level k

Figure 12: **Sensitivity Analysis for Expanded dim m, Weight Sparsity p, and Activation Sparsity k on CUB-200 with ResNet-50 as Backbone.** We report average accuracy in last task (A_T) and overall accuracy ($\bar{A}$). The dots denote the default values we use across experiments.

C.8 Additional Experiments on Larger Pre-trained Models

In our previous experiments, we primarily adopted ResNet-50 and ViT-B/16 as backbones for feature extraction. To further evaluate the scalability of Fly-CL on modern foundation models, we employ the vision encoder of Qwen2.5-VL-7B (Bai et al., 2025) to extract visual features. As shown in Table 11, increasing the scale of the pre-trained backbone consistently leads to improved performance. Furthermore, in Figure 13, we analyze the effect of expanding the projection dimension when using Qwen2.5-VL. The trend remains consistent with our earlier findings: performance improves steadily as the expanded dimension grows, but begins to saturate around $10,000$.

Table 11: **Performance Comparison on Different Scale Pre-trained Models.** We report overall accuracy ($\bar{A}$) across three benchmark datasets: CIFAR-100, CUB-200-2011, and VTAB.

Model	CIFAR-100	CUB-200-2011	VTAB
ResNet-50	84.61±0.16	80.25±0.10	94.00±0.15
ViT-B/16	93.89±0.12	93.84±0.18	96.54±0.38
Qwen2.5-VL-7B	95.06±0.23	94.68±0.21	97.45±0.24

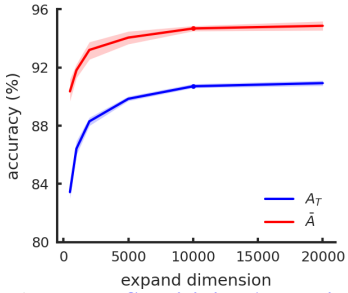


Figure 13: **Sensitivity Analysis for Expanded dim m on CUB-200 with Qwen2.5-VL-7B as Backbone.**

D Detailed Discussion of Related Work

D.1 Comparison with Several Representation-Based Methods

We highlight the main advantages of our proposed Fly-CL over several related representation-based methods, including RanPAC (McDonnell et al., 2023), F-OAL (Zhuang et al., 2024), and RanDumb (Prabhu et al., 2024).

Comparison with RanPAC. RanPAC employs several Parameter-Efficient Transfer Learning (PETL) approach (Chen et al., 2022; Jia et al., 2022; Lian et al., 2022) to adapt the pre-trained model to the downstream domain in the first task, alongside a ridge classification with explicit cross-validation for all ridge candidates. Although effective, these two components make the entire pipeline computationally expensive (see Table 1, 2, and 5). In contrast, Fly-CL eliminates the need for PETL and significantly optimizes the ridge classification process. Additionally, we introduce a sparse projection layer with a top-k operation, replacing the dense projection with ReLU, and analyze the impact of data normalization techniques. The speedup for each components can be refered to Table 4.

Comparison with F-OAL. F-OAL is originally designed for online CL and shares similarities with Fly-CL in feature extraction, random projection, and decorrelation. Although it can also be adapted to the CIL setting with batched data, it has several flaws under this circumstance. For instance, F-OAL lacks the top-k operation to filter noisy components after random projection, and its iterative analytic classifier may accumulate errors, leading to significant performance degradation on ResNet-50 (see Table 2). Moreover, while F-OAL is efficient on CUB-200-2011 and VTAB, its computational cost scales more rapidly with sample size compared to Fly-CL, making it less efficient on CIFAR-100 (see Table 1, 2, and 5).

Comparison with RanDumb. RanDumb shares a similar pipeline with F-OAL and is also designed for online CL. Like F-OAL, it does not utilize a top-k-like operation, and its fixed penalty coefficient λ may result in suboptimal performance. Crucially, RanDumb relies on StreamingLDA, which processes samples sequentially and cannot be parallelized for batch processing. This makes RanDumb significantly slower than all baselines evaluated in Table 1, 2, and 5.

D.2 Summary of Other Compared Baselines

L2P (Wang et al., 2022b) utilizes a prompt pool $\mathbf{P} = \{P_1, P_2, \cdots, P_M\}$ where M is the size of the pool, to store task-specific knowledge. Each prompt P_i is associated with a learnable key K_i for key-value selection. By optimizing the cosine distance $\gamma(p(x), k_i)$, where $p(x)$ is the feature selected by the query function during the training process, L2P can select the most appropriate prompt to provide information that is specific to the task.

DualPrompt (Wang et al., 2022a) extends the key-value selection and optimization methods of L2P by further encoding different types of information into a task-invariant prompt g and a task-specific prompt e. This is shown to be more effective in encoding the learned knowledge. It also decouples the higher-level prompt space by attaching prompts to different layers, which is crucial for the model to reduce forgetting and achieve effective knowledge sharing.

EASE (Zhou et al., 2024) first initializes and trains an adapter for each incoming task to encode task-specific information. It then extracts features of the current task and synthesizes prototypes of former classes to mitigate the subspace gaps between adapters. Finally, EASE constructs the full classifier and reweights the logits for prediction.

InfLoRA (Liang & Li, 2024) is a Interference-Free Low-Rank Adaptation method for continual learning. It injects a small set of parameters to constrain weight updates to a specific subspace. Critically, this subspace is designed to be orthogonal to the gradients of all past tasks while containing the gradient subspace of the new task, thereby eliminating interference and achieving an effective balance between model stability and plasticity.

SEMA (Wang et al., 2025) introduces a self-expansion mechanism by dynamically adding modular adapters only when significant distribution shifts are detected. Each adapter consists of a functional module and a representation descriptor, which acts as a novelty detector to determine whether existing adapters can handle the new task. SEMA further maintains an expandable mixture router to compose adapters through weighted combination, enabling flexible reuse of old modules while expanding only on demand. This design achieves a sub-linear parameter growth rate while improving the stability–plasticity balance across tasks.

MoE-Adapter (Yu et al., 2024) enhances continual learning by injecting a mixture-of-experts adapter structure into transformer blocks. Each MoE-Adapter contains multiple expert adapters, and a learned routing network selects or mixes experts conditioned on the input. This architecture allows the model to capture diverse task-specific patterns while mitigating forgetting through expert specialization. By leveraging the MoE structure, the method improves representational flexibility and achieves stronger adaptation capacity compared to using a single shared adapter.

D.3 Relationship with Kanerva's Sparse Distributed Memory

Kanerva's Sparse Distributed Memory (SDM) (Kanerva, 1988) is a classical high-dimensional computing framework in which memory addresses are distributed in a large binary space, and read/write operations are performed by activating only those locations within a neighborhood defined by Hamming similarity. To highlight its conceptual connection to Fly-CL, we rewrite the SDM

forward computation using the same notation as Fly-CL:

$$\hat{y} = \boldsymbol{C}^{\top}\big(f(\boldsymbol{W}\boldsymbol{v})\big), \tag{19}$$

where $\boldsymbol{W}$ is an address transformation matrix, $\boldsymbol{C}$ is a content transformation matrix, and $f(\cdot)$ denotes a binary activation function.

Under this formulation, SDM and Fly-CL share two high-level principles: (i) expansion into a high-dimensional representational space, which promotes separability and reduces interference; and (ii) sparse activation, which suppresses noises and enhances discrimination through selective addressing.

Despite these conceptual parallels, Fly-CL differs from SDM in several important ways. First, $\boldsymbol{W}$ in SDM is updated through iterative read/write operations, whereas Fly-CL employs a fixed, randomly initialized projection matrix motivated by the fly olfactory circuit. Second, SDM does not incorporate any decorrelation mechanism analogous to the Fly-CL ridge-based KC-to-MBON transformation. Third, SDM operates in a binary space: the activation function $f(\cdot)$ produces binary addresses and similarity is evaluated via Hamming distance, while Fly-CL performs real-valued projections and uses cosine similarity for downstream matching and classification.

These distinctions illustrate that although Fly-CL and SDM share the broad philosophy of high-dimensional sparse representations, their architectural assumptions, objectives, and operating regimes differ substantially.

E Training Details

E.1 Pre-trained Models

We use pre-trained ViT-B/16 and ResNet-50 models in our experiments. The ViT-B/16 checkpoint we used was first pretrained on ImageNet-21K and then fine-tuned on the ImageNet-1K dataset. All of which are loaded using the timm library. We list the dimensions of the extracted features and the download links for the checkpoints of each model in Table 12.

Table 12: **Information Related to the Pre-trained Models We Used in This Work.** We list the dimensions of the extracted features and provide corresponding download links for these pre-trained models.

Model	feature dimension	Link
ViT-B/16	768	Link
ResNet-50	2048	Link

E.2 Datasets

We evaluate our method on three benchmark datasets for CL tasks. Detailed information about these datasets, including download links, is provided in Table 13. For the experiments summarized in Tables 1, 2, and 3, we configure the number of training tasks as $T = 10$ for CIFAR-100 and CUB-200-2011, with 10 and 20 classes per task, respectively. For VTAB, we set $T = 5$ with 10 classes per task. In the longer task sequence experiments (Table 5), we double the task sequence length: for CIFAR-100 and CUB-200-2011, we set $T = 20$ with 5 and 10 classes per task, respectively, while for VTAB, we set $T = 10$ with 5 classes per task. For experiments in Table 6, we set $T = 10$ with 20 classes per task.

E.3 Experiment Setup

We reproduce the baseline results for L2P, DualPrompt, EASE, and RanPAC using the code provided by PILOT (Sun et al., 2023), ensuring that the learning parameters for each baseline align with the description in their original papers. For F-OAL, we adopt their official implementation for reproduction.

In our proposed Fly-CL, we set the expanded dimension m to $10,000$, p to 300, and k to $3,000$ across all experiments. For ViT-B/16, we apply standard data normalization, scaling each pixel value to the

Table 13: **Details of CIFAR-100, CUB-200-2011, VTAB Datasets.** We list the number of training, validation samples and classes for the following datasets, along with the download links.

Dataset	Training Samples	Validation Samples	Classes	Download Link
CIFAR-100 (Krizhevsky et al., 2009)	50000	10000	100	Link
CUB-200-2011 (Wah et al., 2011)	9430	2358	200	Link
VTAB (Zhai et al., 2019)	1796	8619	50	Link
Imagenet-R Hendrycks et al. (2021a)	24000	6000	200	Link
Imagenet-A Hendrycks et al. (2021b)	5981	1519	200	Link

range $[-1, 1]$. For ResNet-50, we normalize the input images using ImageNet statistics. Given the prior knowledge of high multicollinearity in this task, we explore the penalty coefficient range starting from larger values, specifically from 10^6 to 10^9 on a log scale for ViT-B/16 and 10^4 to 10^9 for ResNet-50. Since prompt-based methods and PETL techniques are limited to transformer-based architectures, we compare Fly-CL only with RanPAC and F-OAL in the ResNet-50 setting. For RanPAC, we remove PETL and incorporate data normalization following their original implementation (McDonnell et al., 2023) for ResNet-50. All experiments are conducted using five different random seeds, and we report the mean $\pm$ standard deviation.

E.4 Environments

All experiments were conducted on a Linux server running Ubuntu 20.04.4 LTS, equipped with an Intel(R) Xeon(R) Platinum 8358P CPU at 2.60GHz and 8 NVIDIA GeForce RTX 3090 GPUs, using CUDA version 11.7. For model loading, we employed the timm library (version 0.9.16).

F Limitations and Future Work

Our proposed Fly-CL is theoretically applicable to various scenarios requiring feature separation. Its lightweight design further suggests potential utility in a wide range of Continual Learning and Metric Learning tasks.

Recent neuroscience research (Dasgupta et al., 2017) indicates that the random projection layer in the fly olfactory circuit may not be entirely random. Biological experiments also suggest the presence of certain constraints within this projection layer. Inspired by these findings, a promising direction for future research is to explore structuring the projection layer as an entity with learnable parameters, potentially enhancing its adaptability and performance.

G Broader Impact

Our work provides a new perspective for enhancing the efficiency of CL using pre-trained models, which is crucial for real-world deployment, especially with increasingly large modern models. Fly-CL can help AI researchers and developers create more efficient CL algorithms.

On the other hand, the efficiency improvements in CL could potentially accelerate the development of AI systems that rapidly adapt to new domains without proper safeguards. This might lead to: (1) amplified propagation of biases present in sequential datasets, (2) reduced transparency as models continuously evolve beyond their initial training, and (3) potential misuse for generating tailored content at scale. We recommend implementing rigorous monitoring frameworks to track model behavior across learning phases.

H LLM Usage Declaration

During the preparation of this manuscript, a large language model was employed exclusively for language refinement. Its role was limited to rephrasing certain passages and enhancing the overall clarity and readability of the text. All conceptual contributions, theoretical derivations, experimental design, and analysis were independently developed and verified by the authors. The LLM was not

involved in generating research ideas, shaping methodologies, or producing novel scientific content. The authors bear full responsibility for the entirety of the paper.