AnaCP: Toward Upper-Bound Continual Learning via Analytic Contrastive Projection

Saleh Momeni[1], **Changnan Xiao**[2], **Bing Liu**[1]
[1] Department of Computer Science, University of Illinois Chicago
[2] ChangnXX.github.io
{smomen3, liub}@uic.edu changnanxiao@gmail.com

Abstract

This paper studies the problem of class-incremental learning (CIL), a setting within continual learning, where the model is required to learn a sequence of tasks, each composed of a distinct set of classes. Traditional CIL methods do not use a pre-trained model (PTM) and suffer from catastrophic forgetting (CF) due to their need to incrementally learn both feature representations and the classifier. The incorporation of PTMs into CIL has led to the development of computationally efficient methods that treat the PTM as a feature extractor paired with analytic classifiers. These methods have achieved state-of-the-art performance in CIL. However, they still face a major limitation: the inability to continually adapt or update feature representations to best suit the specific CIL tasks, leading to suboptimal performance. To address this, we propose AnaCP (Analytic Contrastive Projection), a novel method that preserves the efficiency of analytic classifiers while enabling incremental feature adaptation without gradient-based training, thereby eliminating the CF caused by gradient updates. Our experiments show that AnaCP not only outperforms existing baselines but also achieves the accuracy level of joint training, which is regarded as the upper bound of CIL.

1 Introduction

Continual learning (CL) learns a sequence of tasks incrementally, where each task introduces a set of new classes [1]. The key challenge in CL is catastrophic forgetting (CF) [2], a phenomenon where learning new tasks degrades the performance on previously learned ones. Among CL settings, class-incremental learning (CIL) [3] is particularly challenging as it needs to build a unified classifier to recognize all encountered classes, without task-id information at test time. This paper focuses on CIL. Early approaches typically trained models from scratch using regularization, experience replay, or architectural modifications to mitigate CF [4, 5]. However, they fall significantly short of **joint training**, which trains a model with the data from all tasks together as a single task, representing the **upper bound** for CIL performance [6].

Recent CIL methods increasingly exploit pre-trained models (PTMs) to improve accuracy by leveraging their strong feature representations [7, 8, 9]. PTM-based CIL methods can be broadly categorized into three main categories: (1) fine-tuning the PTM, either fully or with lightweight adapters [10], (2) training prompts while keeping the PTM frozen [11], and (3) using the PTM as a fixed feature extractor paired with an analytic classifier, such as the nearest class mean (NCM) method or more advanced variants [12, 13, 14]. We use the term **analytic** to describe methods with a **closed-form solution** that requires no gradient-based training. The first two groups require task-specific training, which is slow and prone to CF. The third group is highly efficient and immune to CF because it avoids gradient-based updates.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Analytic approaches often achieve state-of-the-art performance in CIL, outperforming methods that fine-tune the PTM or learn prompts [15, 14]. However, a **major limitation** of the analytic methods is their inability to adapt the PTM features to suit the downstream tasks, leading to suboptimal performance. Prior works have proposed to fine-tune the PTM on the first task, called first session adaptation (FSA) [16, 13]. However, the PTM must remain frozen afterward to maintain the integrity of analytic learning; otherwise, their closed-form solutions will not work.

This paper introduces **AnaCP** (Analytic Contrastive Projection), a novel method that enables analytic approaches to also perform feature adaptation. Our analytic method is related to Extreme Learning Machines (ELMs) [17, 18], as outlined in Section 3. AnaCP adapts features using a contrastive projection layer, inspired by contrastive learning [19, 20], which draws samples from the same class closer while pushing different classes apart to enhance separation. However, unlike standard contrastive learning, which requires gradient-based training for each task and suffers from CF, AnaCP achieves a similar effect analytically. With the adapted features, an NCM classifier can be easily constructed (see Section 4.3). However, we found that building another ELM classifier by sampling feature representations from distributions of previous tasks yields better performance, while remaining fully analytic. To summarize, this work makes the following contributions:

1. We propose a novel analytic approach that enables continual adaptation of feature representations across tasks, addressing the long-standing limitation of fixed representations in prior analytic methods.
2. Our method avoids CF entirely, as it requires no gradient-based training and instead relies on closed-form updates.
3. This approach is highly efficient, requiring no trainable parameters and incurring negligible computational overhead.
4. Through extensive experiments, we show that when paired with a strong PTM, our method achieves accuracy comparable to the joint training upper bound. This is particularly notable, as the inability to match joint training remains a key barrier to the practical adoption of CL.

2 Related work

Many CIL approaches have been proposed to deal with CF. These methods can be categorized into three main groups: (i) *Regularization* methods, which mitigate CF by penalizing updates to parameters that are important for old tasks [21, 22, 23, 24], or by aligning the current model with those of earlier ones using knowledge distillation [25, 26]. (ii) *Experience replay* methods, which retain a subset of past task samples in a buffer and use them in the new task training [27, 28, 29, 30, 31].A variation of this is *pseudo-replay*, where synthetic samples or feature representations are generated to simulate past data [32, 33, 34, 35]. Our method also replays feature representations in its final step, but they are sampled from each class's distribution represented by its mean and a shared covariance. (iii) *architecture-based* methods, which expand the network for each task [36, 37, 38], or isolate parameters through learning task-specific sub-networks via masking or orthogonal projections [39, 40, 41, 42, 43], and then use a task-id prediction method at test time [6, 44, 45, 46].

Early CIL methods learned both features and classification parameters for each new task without using PTMs, suffering from serious CF. Recent approaches leverage PTMs to significantly boost CIL accuracy [7, 12, 47, 13, 6, 14], primarily following three strategies. (1) fine-tuning the PTM, either by directly updating its parameters (e.g., SLCA [10]) or by adding adapters [6, 48, 49]. (2) learning prompts, where the PTM remains frozen and trainable prompts are introduced to condition its representations (e.g., L2P [11], DualPrompt [47], CODA-Prompt [50], as well as other variants [51, 52, 53]). (3) treating the PTM as a frozen feature extractor with an analytic classifier [13, 54, 14].

Our method aligns with the third strategy, which relies on closed-form solutions. The simplest method is NCM [55, 56], which computes a mean vector for each class and assigns each test sample to the class with the nearest mean. Another method is SLDA [57], which takes advantage of linear discriminant analysis (LDA) [58]. KLDA [14] applies an RBF kernel to expand the feature space before using LDA for classification. The regularized least squares used in ACIL [12] and GACL [54] is also within this paradigm. Analytic CIL methods can be enhanced by adopting the FSA strategy, where the PTM is fine-tuned on the first task and then kept frozen for subsequent tasks. APER [56] adopts this strategy with an NCM classifier. RanPac [13] has a similar approach but applies a random

projection [18] to the feature space, improving class separation. LoRanPAC [59] enables RanPAC to use a much higher dimensional random projection. FeCAM [15] uses a Mahalanobis distance classifier with normalized covariance matrices.

We follow the analytic CIL paradigm by using PTMs as frozen feature extractors. However, unlike previous methods, we introduce a contrastive projection layer to adapt the features for each task, which enables AnaCP to improve the accuracy while preserving the computational efficiency.

3 Background

Class-Incremental Learning: CIL seeks to sequentially learn a series of tasks, each introducing a new set of classes, while prohibiting access to data from previously encountered tasks. Specifically, each task t is defined by a training dataset $\mathcal{D}_t = \{(x_t^{(i)}, y_t^{(i)})\}_{i=1}^{n_t}$, where $x_t^{(i)} \in \mathcal{X}_t$ represents an input sample, and $y_t^{(i)} \in \mathcal{Y}_t$ is its class label. The class sets $\mathcal{Y}_t$ are mutually exclusive, and $\mathcal{Y} = \bigcup_{t=1}^{T} \mathcal{Y}_t$ defines the complete set of classes encountered so far. The core objective of CIL is to learn a single classifier $f : \mathcal{X} \to \mathcal{Y}$ capable of predicting the class label of any test instance without the knowledge of the task it belongs to.

Ridge Regression: Ridge regression is a closed-form method for training a linear classifier on fixed features using one-hot encoded targets [60]. Given a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$ from the PTM and a one-hot label matrix $\mathbf{Y} \in \mathbb{R}^{N \times C}$, the objective is to minimize the squared error with L_2 regularization:

$$\min_{\mathbf{W}} \ \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|^2 + \lambda \|\mathbf{W}\|^2, \tag{1}$$

where λ controls the strength of the regularization. This objective penalizes deviations from the target labels while encouraging smaller weights to prevent overfitting. The solution is obtained by setting the gradient for $\mathbf{W}$ to zero, yielding the closed-form optimal weights:

$$\mathbf{W} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y}, \tag{2}$$

often called a ridge classifier or regularized least squares. Here, $\mathbf{X}^\top \mathbf{X}$ is called the Gram matrix ($\mathbf{G}$) and $\mathbf{X}^\top \mathbf{Y}$ is the cross-product matrix ($\mathbf{H}$). This closed-form solution is attractive because it forgoes iterative training and directly computes a global optimum.

Analytic CIL: When tasks arrive sequentially, we can compute the ridge regression solution without revisiting past data [17, 12]. Let $(\mathbf{X}_t, \mathbf{Y}_t)$ denote the data from task t. Rather than storing all data, we can incrementally update the Gram and cross-product matrices as follows:

$$\mathbf{G}_t = \mathbf{G}_{t-1} + \mathbf{X}_t^\top \mathbf{X}_t, \quad \mathbf{H}_t = \mathbf{H}_{t-1} + \mathbf{X}_t^\top \mathbf{Y}_t, \tag{3}$$

where $\mathbf{G}_t$ and $\mathbf{H}_t$ are the gram and cross-product matrices after task t, respectively. Here, we slightly abuse the notation for $\mathbf{H}_t$ by allowing the number of classes to increase over time. To maintain dimensional consistency, both $\mathbf{H}_{t-1}$ and the one-hot labels in $\mathbf{Y}_t$ must be zero-padded. This allows us to compute the updated solution:

$$\mathbf{W}_t = (\mathbf{G}_t + \lambda \mathbf{I})^{-1} \mathbf{H}_t \tag{4}$$

Random Projection and Extreme Learning Machines: To improve the feature representations, the original d-dimensional features can be projected into a higher-dimensional space of dimension D before learning the analytic classifier in the projected space. [18] proposes to project the features via a random nonlinear layer:

$$\mathbf{Z} = \phi(\mathbf{X}\mathbf{R}), \tag{5}$$

where $\mathbf{R} \in \mathbb{R}^{d \times D}$ is a fixed matrix with random values and $\phi(\cdot)$ is a nonlinear activation, for which we use a GELU function. The classifier is learned analytically via ridge regression:

$$\mathbf{W} = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^\top \mathbf{Y} \tag{6}$$

This solution can be updated incrementally as before (Eqs. 3 and 4). The random projection acts like a kernel function, expanding the feature space and introducing nonlinear interactions among input features, which enhances class separability.

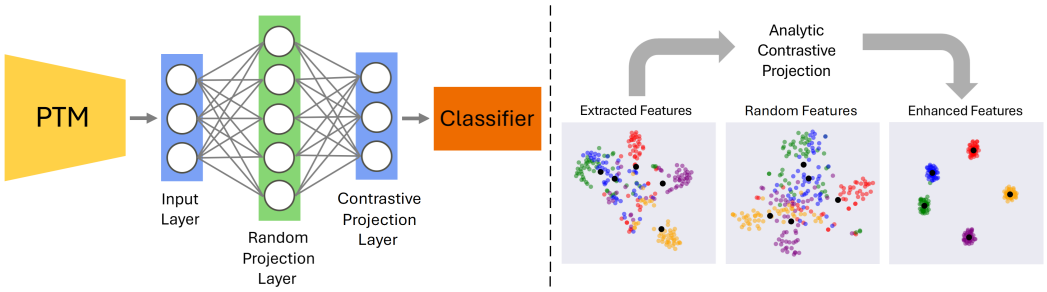

Figure 1: (**Left**) Architecture of AnaCP: The random projection layer uses fixed, randomly assigned weights, while the contrastive projection layer weights are computed analytically. Compared to the ELM architecture, AnaCP introduces an additional contrastive projection layer that adapts the feature representations. (**Right**) t-SNE visualization of the input features (first five classes of ImageNet-R) and their enhancement after contrastive projection using DINO-v2 as the PTM. Random features are generally more separable than input features due to their higher dimensionality, even though this is not easily observable in the 2D t-SNE map.

4 Methodology

Existing analytic CIL methods operate on fixed feature representations, allowing the classifier to be updated analytically without revisiting prior data. Freezing the PTM is essential because modifying it during training would alter the feature representations, undermining previous analytic updates and leading to CF. However, this rigid strategy also limits the capability to adapt feature representations to suit the downstream CIL tasks.

To overcome this limitation, we propose a novel analytic approach that also adapts feature representations for the continual learning tasks. The pipeline of our approach, illustrated in Figure 1 (left), begins with the PTM generating feature representations, termed the **input layer**. These features are transformed via a **random projection (RP) layer** into a higher-dimensional space [18] and then passed through the **contrastive projection (CP) layer**, which enhances the class separability. The CP layer draws inspiration from contrastive learning, where the samples from the same class (**positives**) are drawn closer, while samples from different classes (**negatives**) are pushed apart, typically using a contrastive loss. However, applying this directly to CIL would lead to CF due to iterative gradient updates. We achieve similar effects via an analytic CP layer that can be incrementally updated as new tasks arrive. The impact of this CP layer on feature distribution is depicted in Figure 1 (right). Finally, a **classifier layer** makes the final predictions based on these adapted features.

4.1 Positive Alignment via Prototype Regression

We begin with the ELM formulation, where ridge regression is employed to map the features from the RP layer to the target vectors, as previously defined:

$$\mathbf{W} = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^\top \mathbf{T}, \tag{7}$$

where $\mathbf{Z} \in \mathbb{R}^{N \times D}$ represents the random feature obtained from the RP layer, and $\mathbf{T}$ contains target vectors for inputs. When $\mathbf{T}$ is one-hot encoded for classification, the cross-product $\mathbf{Z}^\top \mathbf{T}$ reduces to a matrix where each column is the sum of random features belonging to a given class. This can be factorized as:

$$\mathbf{H} = \mathbf{Z}^\top \mathbf{T} = \mathbf{M}\mathbf{N} \tag{8}$$

Here, $\mathbf{M} \in \mathbb{R}^{D \times C}$ contains the **class prototypes** $\mathbf{m}_c$, defined as the means of the random features of class c, and $\mathbf{N} = \text{diag}(n_1, \ldots, n_C)$ holds the number of samples per class. Both $\mathbf{M}$ and $\mathbf{N}$ can be computed incrementally as each task arrives.

This formulation can be extended to arbitrary targets. Instead of using one-hot vectors, we can assign each class a **target prototype**, representing its ideal position in the projected space. These target prototypes can have any dimensionality; however, to preserve the geometric structure of the representations, we set their dimension to match that of the input layer, d. Let $\mathbf{P} \in \mathbb{R}^{C \times d}$ denote the target prototype matrix, where each row $\mathbf{p}_c$ specifies the desired projection for class c. For instance,

$\mathbf{p}_c$ can be set as the original class mean of the input layer, pulling the feature representations of class c toward their respective mean. In this case, the cross-product becomes:

$$\mathbf{H} = \mathbf{Z}^\top \mathbf{T} = \sum_c \left(\sum_{i \in c} \mathbf{z}_i \right) \mathbf{p}_c^\top = \sum_c \mathbf{m}_c \mathbf{n}_c \mathbf{p}_c^\top = \mathbf{MNP} \tag{9}$$

This decomposition enables efficient incremental updates by maintaining class prototypes $\mathbf{M}$ computed based on the features from the random projection layer, sample counts $\mathbf{N}$, and target prototypes $\mathbf{P}$. The learned projection $\mathbf{W}$ maps samples toward their corresponding target prototypes, effectively pulling intra-class samples together and enhancing positive alignment.

4.2 Negative Repulsion via Target-Prototype Separation

Aligning input samples to their respective class means (using class means as target prototypes) improves intra-class compactness but does not guarantee adequate inter-class separation. In particular, some class means may lie close to each other in the feature space, making them hard to distinguish. To address this, we refine the target prototypes by **shifting the class means** to enhance the separation between classes.

Let $\mathbf{C} = (\mu_1, \ldots, \mu_C) \in \mathbb{R}^{d \times C}$ denote the matrix of class means at the input layer. We normalize these means by whitening them with respect to the feature covariance matrix $\mathbf{\Sigma}$ (shared by all classes). This adjustment is essential because a large variance in some dimensions can give a false impression of class separability. For example, in a low-variance dimension even a small separation is meaningful, whereas in high-variance dimensions, a much larger separation is needed to achieve the same effect. Whitening corrects for this imbalance by accounting for the variance of each feature dimension. The covariance is computed incrementally as [15]:

$$\mathbf{\Sigma}_t = \frac{N_{t-1}}{N_t} \mathbf{\Sigma}_{t-1} + \frac{1}{N_t} \sum_{c \in \mathcal{C}_t} \sum_{i \in c} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^\top, \tag{10}$$

where N_t is the total number of samples seen up to task t and $\mathcal{C}_t$ is the set of classes introduced at task t. We can then transform the class means as:

$$\hat{\mathbf{C}} = \mathbf{\Sigma}^{-1/2} \mathbf{C}, \tag{11}$$

yielding whitened means with unit variance in all directions. We then perform singular value decomposition (SVD) on the whitened class means:

$$\hat{\mathbf{C}} = \mathbf{U} \mathbf{S} \mathbf{V}^\top, \tag{12}$$

where $\mathbf{U} \in \mathbb{R}^{d \times C}$ is an orthonormal basis of the input space, $\mathbf{V} \in \mathbb{R}^{C \times C}$ defines the class means subspace, and $\mathbf{S} \in \mathbb{R}^{C \times C}$ is a diagonal matrix containing the singular values.

Our objective is to enhance the separation between class means, quantified by the sum of their pairwise cosine similarities. In principle, maximum separation would be achieved if the means were arranged uniformly on the surface of a hypersphere. However, this strict configuration would significantly alter the original data geometry, distorting the feature representations. To maintain the structure while increasing class separation, we introduce an adjustment to the singular values in $\mathbf{S}$, which perturbs the class means in directions that reduce their pairwise cosine similarities:

$$\tilde{\mathbf{S}} = \mathbf{S} + \alpha Diag\{\delta_1, \ldots, \delta_C\}, \tag{13}$$

where α is a scaling factor, and δ_i is defined in the following Lemma.

Lemma 4.1. *Denote $w_1, \ldots, w_C \in \mathbb{R}^C$ as arbitrary vectors, and $e_1, \ldots, e_C \in \mathbb{R}^C$ as a set of orthogonal bases of $\mathbb{R}^C$. Denote $\langle x, y \rangle = x^\top y$ as the inner product and $\theta(x, y) = \frac{x^\top y}{||x|| \cdot ||y||}$ as the cosine similarity. There exist $\alpha > 0$ and*

$$\delta_i = \begin{cases} 1, \textit{ if } \sum_{j \neq i} \frac{1}{||w_j||} \left[(2 \cdot 1_{\langle w_i, w_j \rangle \geq 0}) \cdot \langle e_i, w_j \rangle \cdot ||w_i||^2 - \langle e_i, w_i \rangle \cdot |\langle w_i, w_j \rangle| \right] < 0, \\ 0, \textit{ if } \sum_{j \neq i} \frac{1}{||w_j||} \left[(2 \cdot 1_{\langle w_i, w_j \rangle \geq 0}) \cdot \langle e_i, w_j \rangle \cdot ||w_i||^2 - \langle e_i, w_i \rangle \cdot |\langle w_i, w_j \rangle| \right] = 0, \\ -1, \textit{ else,} \end{cases}$$

s.t.

$$\sum_i \sum_{j \neq i} |\theta(w_i + \alpha\delta_i e_i, w_j + \alpha\delta_j e_j)| \leq \sum_i \sum_{j \neq i} |\theta(w_i, w_j)| \tag{14}$$

See the proof in Appendix A. In Eq. 12, we denote $\mathbf{S}\mathbf{V}^\top = (w_1, \ldots, w_C)$ and $\mathbf{V}^\top = (e_1, \ldots, e_C)$. Each w_i in the subspace $\mathbf{S}\mathbf{V}^\top$ is perturbed by $\alpha\delta_i e_i$, effectively shifting it in an orthogonal direction with magnitude α. By Lemma 4.1, there exist coefficients $\delta_1, \ldots, \delta_C$ and a scalar $\alpha > 0$ such that Eq. 14 holds. The sign of each δ_i determines whether the shift occurs in the direction of e_i or its opposite. This condition arises from the proof of Lemma 4.1, where we analyze the derivative of the function $f_i(\alpha)$, defined as the sum of cosine similarities involving class i. By selecting δ_i accordingly, we ensure that the slope of $f_i(\alpha)$ at $\alpha = 0$ is negative, which guarantees that a small positive α decreases the cosine similarity. The choice of α is also important: if α is too small, the shift becomes negligible, whereas if it is too large, the reduction in cosine similarity may no longer be valid. In practice, we set $\alpha = 1$ in our experiments.

Defining $\mathbf{\Delta} = Diag\{\delta_1, \ldots, \delta_C\}$, the lemma guarantees that the average cosine similarity among the columns of $(\mathbf{S} + \alpha\mathbf{\Delta})\mathbf{V}^\top$ is smaller than that of $\mathbf{S}\mathbf{V}^\top$. Since the columns of $\mathbf{U} \in \mathbb{R}^{d \times C}$ form an orthogonal basis in $\mathbb{R}^d$, the transformations preserve inner products and norms:

$$\langle \mathbf{U}w_i, \mathbf{U}w_j \rangle = \langle w_i, w_j \rangle, \quad ||\mathbf{U}w_i|| = ||w_i||, \quad ||\mathbf{U}w_j|| = ||w_j|| \tag{15}$$

Thus, the cosine similarity between transformed vectors remains the same, i.e., $\theta(\mathbf{U}w_i, \mathbf{U}w_j) = \theta(w_i, w_j)$. Therefore, the average cosine similarity among the columns of $\mathbf{U}(\mathbf{S} + \alpha\mathbf{\Delta})\mathbf{V}^\top$ is also reduced compared to the original matrix $\hat{\mathbf{C}} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$.

Note that separability is measured as the sum of cosine similarities, but the class means are whitened. This procedure is closely related to Mahalanobis distance, which has been shown to outperform cosine similarity in related contexts [15]. By whitening, we effectively assess separation using Mahalanobis distance; after increasing the separation, we de-whiten the features to restore their original scale:

$$\hat{\tilde{\mathbf{C}}} = \mathbf{U}\tilde{\mathbf{S}}\mathbf{V}^\top, \quad \tilde{\mathbf{C}} = \hat{\tilde{\mathbf{C}}}\mathbf{\Sigma}^{1/2} \tag{16}$$

The separated class means are then used as target prototypes for the CP layer by setting $\mathbf{P}$ in Eq. 9 to $\tilde{\mathbf{C}}$, aligning samples with their respective classes while enhancing the separation between classes.

4.3 Classifier

The CP layer is an analytic module that adapts feature representations across tasks without modifying the underlying PTM. To increase its capacity, we extend it with multiple random projections. Since each RP is initialized independently, we learn a corresponding CP for each RP, all sharing the same set of target prototypes. Specifically, we define H such heads, where each CP head $\mathbf{W}^{(h)}$ maps the random features to the same target prototype space. Given an input $\mathbf{x}$, each head produces a projection and the output is obtained by averaging across heads:

$$\mathbf{u}^{(h)} = \phi(\mathbf{x}\mathbf{R}^{(h)})\mathbf{W}^{(h)}, \quad \mathbf{u} = \frac{1}{H}\sum_{h=1}^{H} \mathbf{u}^{(h)} \tag{17}$$

This aggregated representation $\mathbf{u}$ is then classified using NCM, which assigns it to the nearest target prototype in $\mathbf{P}$. As shown in Table 3 (row 3), AnaCP with an NCM classifier already outperforms all baselines. However, NCM is often surpassed by more expressive classifiers. To further improve performance, we place an ELM classifier after the averaged CP representation.

Unlike PTM features and their random projections, which remain fixed, the CP outputs evolve as new tasks are introduced due to the incremental computation of the CP layer. Directly updating the ELM classifier incrementally (Eq. 3) may therefore lead to CF. To mitigate this, we employ a pseudo-replay strategy. We model the PTM features (input layer) as a multivariate Gaussian distribution parameterized by the class means and a shared covariance matrix, from which we generate pseudo-replay samples for training. This approach resembles that of [10], which also trains the classifier on generated features. However, whereas [10] maintains a separate covariance matrix per class (leading to memory cost that grows with the number of classes), we use a single shared covariance matrix across all classes. This design significantly reduces memory usage while yielding comparable

performance. Specifically, we reuse the class means and shared covariance matrix from Eq. 10 for feature generation. The generated samples are then processed by the RP and CP layers to form the classifier input, upon which the ELM is trained using Eq. 6.

5 Experiments

5.1 Experimental Setup

Datasets: We conduct experiments on five publicly available datasets: CIFAR100 (100 classes) [61], ImageNet-R (200 classes) [62], CUB (200 classes) [63], TinyImageNet (200 classes) [64], and Cars (196 classes) [65]. For all datasets, we use the official train and test splits. Each dataset is split into 10 disjoint tasks by shuffling classes, and experiments are repeated with three different random seeds to account for variability in class-task assignments.

Baselines: We compare AnaCP with a range of baselines, including prompt-learning methods (L2P [11], DualPrompt [47], and CODA-Prompt [50]), fine-tuning method (SLCA [10]), and analytic methods that treat the PTM as a frozen feature extractor. The analytic baselines include SimpleCIL [55], SLDA [57], KLDA [14], GACL [54], APER [56], FeCAM [15], and RanPac [13], with some incorporating FSA. For a complete performance perspective, we also include results from joint linear probe, where a linear softmax classifier is trained on the PTM features using gradient descent with all tasks' data, and joint fine-tuning, which directly fine-tunes the PTM on the entire dataset and serves as the upper bound for CIL performance.

Implementation Details: For the PTMs, we use DINO-v2 [66] and MoCo-v3 [67], both of which are self-supervised PTMs. This choice helps prevent information leakage, as supervised PTMs are exposed to class labels during training, some of which may reappear in CIL, leading to unintended information leakage. MoCo-v3 is pre-trained on ImageNet-1k – a commonly used but relatively small dataset for pre-training in prior studies. However, because ImageNet-1k (or even ImageNet-21k) is considered limited for real-world applications, we also utilize DINO-v2, a more powerful model pre-trained on the substantially larger LVD-142M dataset. For joint fine-tuning, we employ LoRA adapters [68] instead of full fine-tuning, as we found this approach to be more accurate.

The prompt learning baselines are taken from the [50] repository, while SLCA, FeCAM, and RanPac are evaluated using their original implementations. All remaining analytic baselines are incorporated into a unified experimental setup, following their official implementations to ensure fair comparison.[1] We also adopt FSA following [56] before applying AnaCP to the frozen PTM as it helps improve accuracy. We use RP dimension $D = 5000$, number of CP heads $H = 3$, and number of generated feature representations per class for the classifier $R = 100$ as the default configuration; ablations on other variants are included. The regularization parameter λ in Eq. 6 for ELM is set to 10^2, and the coefficient α for class mean repulsion is set to 1. These values were optimized on a validation set derived from the CIFAR100 training set and are consistently used across all datasets and both PTMs. All experiments are conducted on a single NVIDIA A100 GPU with 80GB VRAM.

Evaluation Metric: We evaluate model performance using two primary metrics: Last Accuracy (A_{Last}), the accuracy after completing all tasks, and Average Incremental Accuracy (A_{Avg}), calculated as $A_{\text{Avg}} = \frac{1}{T}\sum_{t=1}^{T} A_t$, where A_t denotes the accuracy at the end of task t. Additionally, we measure running time and memory efficiency to assess the practicality of the methods. We also define relative error reduction as $\frac{A_I - A_0}{100 - A_0} \times 100$, where A_0 is the baseline accuracy and A_I is the improved accuracy.

5.2 Comparison with Baselines

Table 1 presents the main results of our experiments, with all methods using DINO-v2 as the PTM. AnaCP consistently outperforms all baselines across the evaluated datasets, achieving an absolute improvement in last accuracy ranging from 1.03% to 3.17% across different datasets, corresponding to a relative error reduction between 4.8% and 31.4%. We can observe that using replay features with a shared covariance (AnaCP) achieves similar results to class-specific covariances (AnaCP - Σ_y) while significantly reducing the number of parameters. AnaCP also surpasses the joint linear probe on all datasets and achieves comparable accuracy to joint fine-tuning, which represents the

[1] The code of AnaCP is available at https://github.com/SalehMomeni/AnaCP.

Method	CIFAR100		ImageNet-R		CUB		TinyImageNet		Cars	
	A_{avg}	A_{last}	A_{avg}	A_{last}	A_{avg}	A_{last}	A_{avg}	A_{last}	A_{avg}	A_{last}
Joint linear probe	-	$89.29_{\pm 0.00}$	-	$82.49_{\pm 0.02}$	-	$89.33_{\pm 0.01}$	-	$85.43_{\pm 0.01}$	-	$86.84_{\pm 0.01}$
Joint fine-tuning	-	$93.35_{\pm 0.00}$	-	$88.79_{\pm 0.01}$	-	$89.87_{\pm 0.01}$	-	$88.81_{\pm 0.02}$	-	$89.92_{\pm 0.02}$
L2P	$88.50_{\pm 0.38}$	$84.16_{\pm 0.35}$	$84.77_{\pm 0.64}$	$79.57_{\pm 0.13}$	$82.48_{\pm 1.25}$	$72.45_{\pm 0.25}$	$89.39_{\pm 0.18}$	$84.45_{\pm 0.45}$	$54.29_{\pm 2.02}$	$40.83_{\pm 1.12}$
DualPrompt	$88.13_{\pm 0.35}$	$83.46_{\pm 0.14}$	$84.59_{\pm 0.83}$	$79.61_{\pm 0.12}$	$82.13_{\pm 1.25}$	$72.33_{\pm 0.10}$	$89.50_{\pm 0.01}$	$84.64_{\pm 0.26}$	$53.31_{\pm 2.83}$	$39.15_{\pm 1.18}$
CODA-Prompt	$90.87_{\pm 0.39}$	$86.83_{\pm 0.01}$	$86.43_{\pm 0.61}$	$82.39_{\pm 0.01}$	$83.11_{\pm 1.07}$	$73.08_{\pm 0.25}$	$90.58_{\pm 0.07}$	$85.50_{\pm 0.52}$	$58.49_{\pm 2.32}$	$42.63_{\pm 0.71}$
SLCA	$93.03_{\pm 0.32}$	$88.80_{\pm 0.22}$	$\underline{88.50}_{\pm 0.51}$	$\underline{84.78}_{\pm 0.26}$	$92.83_{\pm 0.52}$	$88.97_{\pm 0.16}$	$89.38_{\pm 0.31}$	$85.31_{\pm 0.21}$	$90.70_{\pm 0.64}$	$85.76_{\pm 0.49}$
SimpleCIL	$91.18_{\pm 0.34}$	$86.20_{\pm 0.00}$	$80.94_{\pm 0.46}$	$75.11_{\pm 0.00}$	$91.20_{\pm 0.48}$	$86.87_{\pm 0.00}$	$87.80_{\pm 0.56}$	$83.17_{\pm 0.00}$	$66.75_{\pm 0.39}$	$55.36_{\pm 0.00}$
SLDA	$92.28_{\pm 0.31}$	$87.57_{\pm 0.00}$	$83.28_{\pm 0.14}$	$77.25_{\pm 0.00}$	$93.05_{\pm 0.44}$	$89.51_{\pm 0.00}$	$87.86_{\pm 0.69}$	$82.69_{\pm 0.00}$	$89.97_{\pm 0.70}$	$87.20_{\pm 0.00}$
KLDA	$92.97_{\pm 0.20}$	$88.64_{\pm 0.16}$	$82.37_{\pm 0.19}$	$77.75_{\pm 0.01}$	$93.11_{\pm 0.33}$	$\underline{89.54}_{\pm 0.11}$	$88.20_{\pm 0.64}$	$82.91_{\pm 0.08}$	$91.30_{\pm 0.63}$	$86.40_{\pm 0.03}$
GACL	$93.85_{\pm 0.21}$	$\underline{90.10}_{\pm 0.01}$	$84.91_{\pm 0.23}$	$81.74_{\pm 0.15}$	$\underline{93.22}_{\pm 0.47}$	$89.37_{\pm 0.10}$	$\underline{90.69}_{\pm 0.42}$	$\underline{86.70}_{\pm 0.03}$	$89.76_{\pm 0.48}$	$84.52_{\pm 0.18}$
APER	$93.05_{\pm 0.43}$	$88.59_{\pm 0.14}$	$86.32_{\pm 0.12}$	$80.61_{\pm 0.04}$	$91.18_{\pm 0.51}$	$86.85_{\pm 0.12}$	$89.26_{\pm 0.65}$	$84.50_{\pm 0.21}$	$75.40_{\pm 0.97}$	$65.11_{\pm 0.80}$
FeCAM	$93.27_{\pm 0.38}$	$89.08_{\pm 0.70}$	$84.61_{\pm 0.46}$	$79.27_{\pm 0.50}$	$91.59_{\pm 0.72}$	$87.63_{\pm 0.53}$	$89.47_{\pm 0.50}$	$84.75_{\pm 0.25}$	$90.64_{\pm 0.98}$	$85.72_{\pm 0.72}$
RanPAC	$\underline{94.10}_{\pm 0.34}$	$90.09_{\pm 0.79}$	$87.96_{\pm 0.19}$	$84.41_{\pm 0.20}$	$92.37_{\pm 0.68}$	$88.44_{\pm 0.37}$	$89.55_{\pm 0.48}$	$84.67_{\pm 1.04}$	$\underline{91.48}_{\pm 0.87}$	$\underline{87.48}_{\pm 0.11}$
AnaCP - Σ_y	$95.43_{\pm 0.18}$	$92.15_{\pm 0.11}$	$90.38_{\pm 0.09}$	$86.68_{\pm 0.03}$	$93.82_{\pm 0.28}$	$90.61_{\pm 0.13}$	$91.81_{\pm 0.26}$	$87.71_{\pm 0.29}$	$94.29_{\pm 0.59}$	$90.87_{\pm 0.20}$
AnaCP	$\mathbf{95.43}_{\pm 0.20}$	$\mathbf{92.15}_{\pm 0.09}$	$\mathbf{90.37}_{\pm 0.11}$	$\mathbf{86.60}_{\pm 0.04}$	$\mathbf{93.84}_{\pm 0.31}$	$\mathbf{90.57}_{\pm 0.15}$	$\mathbf{91.57}_{\pm 0.29}$	$\mathbf{87.35}_{\pm 0.29}$	$\mathbf{94.16}_{\pm 0.61}$	$\mathbf{90.65}_{\pm 0.24}$
Rel. Error	22.5% ↓	20.7% ↓	16.2% ↓	11.9% ↓	9.1% ↓	9.8% ↓	9.4% ↓	4.8% ↓	31.4% ↓	25.3% ↓

Table 1: Comparison of AnaCP with baselines using **DINO-v2** as the PTM. All methods are replay-free. AnaCP employs a shared covariance matrix for pseudo-replay feature generation, while AnaCP-Σ_y uses a separate covariance matrix for each class; the latter is reported only as an ablation since it requires more parameters. The best result (excluding AnaCP-Σ_y) is shown in bold, and the second-best is underlined. Rel. Error indicates the reduction in error rate achieved by AnaCP compared to the best baseline for the column.

Method	CIFAR100		ImageNet-R		CUB		TinyImageNet		Cars	
	A_{avg}	A_{last}	A_{avg}	A_{last}	A_{avg}	A_{last}	A_{avg}	A_{last}	A_{avg}	A_{last}
Joint linear probe	-	$83.81_{\pm 0.01}$	-	$59.16_{\pm 0.02}$	-	$63.19_{\pm 0.02}$	-	$81.12_{\pm 0.02}$	-	$41.95_{\pm 0.02}$
Joint fine-tuning	-	$87.45_{\pm 0.02}$	-	$73.54_{\pm 0.03}$	-	$79.24_{\pm 0.01}$	-	$82.54_{\pm 0.03}$	-	$81.98_{\pm 0.02}$
SLCA	$69.54_{\pm 0.58}$	$69.54_{\pm 0.89}$	$56.66_{\pm 1.12}$	$44.63_{\pm 0.76}$	$74.42_{\pm 0.51}$	$70.88_{\pm 0.36}$	$68.93_{\pm 0.59}$	$63.53_{\pm 0.26}$	$55.47_{\pm 0.88}$	$53.00_{\pm 0.41}$
SimpleCIL	$79.27_{\pm 0.47}$	$70.89_{\pm 0.00}$	$45.83_{\pm 0.83}$	$37.75_{\pm 0.00}$	$56.36_{\pm 0.61}$	$45.83_{\pm 0.00}$	$77.31_{\pm 0.46}$	$71.01_{\pm 0.00}$	$28.99_{\pm 0.36}$	$19.79_{\pm 0.00}$
SLDA	$86.68_{\pm 0.51}$	$79.14_{\pm 0.00}$	$65.37_{\pm 1.08}$	$58.22_{\pm 0.00}$	$\underline{79.61}_{\pm 0.60}$	$72.39_{\pm 0.00}$	$81.60_{\pm 0.78}$	$75.30_{\pm 0.00}$	$78.33_{\pm 0.32}$	$69.72_{\pm 0.00}$
KLDA	$88.27_{\pm 0.29}$	$81.15_{\pm 0.18}$	$63.64_{\pm 0.87}$	$56.92_{\pm 0.27}$	$78.49_{\pm 0.27}$	$71.39_{\pm 0.14}$	$82.46_{\pm 0.72}$	$75.67_{\pm 0.05}$	$77.33_{\pm 0.20}$	$69.11_{\pm 0.09}$
GACL	$\underline{88.89}_{\pm 0.31}$	$\underline{82.10}_{\pm 0.10}$	$64.02_{\pm 1.19}$	$56.80_{\pm 0.48}$	$76.12_{\pm 0.32}$	$67.43_{\pm 0.35}$	$\underline{84.72}_{\pm 0.58}$	$\underline{78.78}_{\pm 0.10}$	$72.48_{\pm 0.45}$	$61.55_{\pm 0.26}$
APER	$80.51_{\pm 1.25}$	$71.36_{\pm 1.36}$	$64.65_{\pm 0.18}$	$54.79_{\pm 0.68}$	$72.09_{\pm 1.59}$	$63.34_{\pm 0.57}$	$80.06_{\pm 0.74}$	$72.84_{\pm 0.31}$	$45.09_{\pm 2.78}$	$32.65_{\pm 2.06}$
FeCAM	$84.73_{\pm 1.24}$	$77.23_{\pm 2.00}$	$64.61_{\pm 0.70}$	$55.45_{\pm 0.69}$	$73.61_{\pm 0.80}$	$66.50_{\pm 0.13}$	$81.28_{\pm 1.44}$	$74.49_{\pm 1.04}$	$74.55_{\pm 0.64}$	$66.61_{\pm 0.87}$
RanPAC	$86.99_{\pm 1.46}$	$79.37_{\pm 1.84}$	$\underline{70.68}_{\pm 0.17}$	$\underline{62.50}_{\pm 0.81}$	$78.93_{\pm 0.75}$	$\underline{72.69}_{\pm 0.37}$	$82.97_{\pm 1.01}$	$76.25_{\pm 0.60}$	$\underline{80.39}_{\pm 0.23}$	$\underline{72.61}_{\pm 0.65}$
AnaCP - Σ_y	$90.16_{\pm 0.36}$	$83.87_{\pm 0.13}$	$74.88_{\pm 0.48}$	$67.91_{\pm 0.48}$	$84.37_{\pm 0.73}$	$79.05_{\pm 0.63}$	$85.88_{\pm 0.58}$	$80.02_{\pm 0.32}$	$85.08_{\pm 0.94}$	$79.24_{\pm 0.63}$
AnaCP	$\mathbf{89.91}_{\pm 0.43}$	$\mathbf{83.70}_{\pm 0.26}$	$\mathbf{74.60}_{\pm 0.54}$	$\mathbf{67.30}_{\pm 0.48}$	$\mathbf{84.21}_{\pm 0.76}$	$\mathbf{78.61}_{\pm 0.47}$	$\mathbf{85.43}_{\pm 0.49}$	$\mathbf{79.32}_{\pm 0.51}$	$\mathbf{84.78}_{\pm 0.87}$	$\mathbf{78.70}_{\pm 0.76}$
Rel. Error	9.1% ↓	8.9% ↓	13.3% ↓	12.7% ↓	22.5% ↓	21.6% ↓	4.6% ↓	2.54% ↓	22.3% ↓	22.2% ↓

Table 2: Comparison of AnaCP with baselines using **MoCo-v3** as the PTM. The prompt learning baselines are excluded here due to their much lower accuracy, which is also the case in Table 1.

upper bound for CIL. Notably, on two datasets, AnaCP even exceeds this upper bound, while on the remaining datasets, the largest accuracy gap is only around 2%. These results highlight that AnaCP's feature adaptation is highly effective, even without directly training the PTM on each task.

We also evaluate AnaCP using MoCo-v3 as the PTM, as presented in Table 2. AnaCP outperforms the baselines by a larger margin with MoCo-v3 than with DINO-v2, as the weaker features of MoCo-v3 make the CP layer more impactful. In this case, the gap to joint fine-tuning increases to 6%, which is expected since MoCo-v3 is a weaker PTM and benefits more from full fine-tuning. However, given that both PTMs have the same architecture (number of layers and hidden size), there is no reason to prefer a weaker PTM in real-world applications.

5.3 Analysis of Catastrophic Forgetting

The CP layer in our method is inherently immune to CF, as it operates on a frozen PTM and updates its statistics (prototypes and the Gram matrix) incrementally without overwriting prior information. In contrast, the final ELM classifier relies on pseudo-replay, which in principle could introduce some forgetting. In practice, however, we found this effect to be minimal. To verify this, we evaluate under the task-incremental learning (TIL) setting, where the task identity is provided at inference time. Specifically, we present the accuracy matrix $A[t][i]$, where each entry denotes the accuracy on task i after training on the first t tasks of CIFAR-100 (10-task split) using DINO-v2 as the backbone.

We adopt the TIL setting for this analysis because it offers a clearer view of forgetting. In CIL, accuracy inevitably decreases as more tasks are introduced, not necessarily due to forgetting, but because the classification problem becomes harder with a growing number of classes. By contrast, TIL keeps the classification difficulty fixed, since each task has a constant number of classes and the

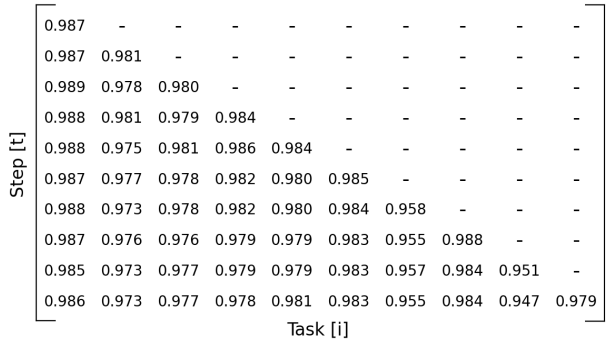

Figure 2: Accuracy matrix $A[t][i]$ on CIFAR-100 with 10 task splits using DINO-v2 as the backbone under the TIL setting. Each entry shows the accuracy on task i after training on the first t tasks.

task ID is known at inference. Under this setup, forgetting would appear as a decline in accuracy on earlier tasks. However, as shown in Figure 2, the accuracy for task 1 (first column) remains nearly constant across all steps. The same holds for other tasks, demonstrating that our method effectively preserves knowledge from prior tasks.

row	CLS	CP	NR	H	D	R	CIFAR100	ImageNet-R	CUB	TinyImageNet	Cars
1	NCM	✗	-	-	-	-	$88.13_{\pm 0.21}$	$81.60_{\pm 0.08}$	$86.95_{\pm 0.36}$	$83.63_{\pm 0.48}$	$69.51_{\pm 0.87}$
2	NCM	✓	✗	3	5000	-	$91.31_{\pm 0.03}$	$83.29_{\pm 0.45}$	$89.52_{\pm 0.10}$	$85.89_{\pm 0.26}$	$86.25_{\pm 0.25}$
3	NCM	✓	✓	3	5000	-	$91.86_{\pm 0.05}$	$86.01_{\pm 0.11}$	$89.88_{\pm 0.32}$	$87.02_{\pm 0.27}$	$89.48_{\pm 0.29}$
4	ELM	✗	-	-	5000	-	$91.12_{\pm 0.10}$	$85.14_{\pm 0.26}$	$89.75_{\pm 0.19}$	$86.48_{\pm 0.12}$	$89.32_{\pm 0.20}$
5	ELM	✓	✗	3	5000	100	$91.87_{\pm 0.02}$	$85.73_{\pm 0.19}$	$90.39_{\pm 0.14}$	$86.69_{\pm 0.28}$	$90.24_{\pm 0.20}$
6	ELM	✓	✓	3	5000	100	$92.15_{\pm 0.09}$	$86.60_{\pm 0.04}$	$90.48_{\pm 0.15}$	$87.71_{\pm 0.29}$	$90.65_{\pm 0.24}$
7	ELM	✓	✓	3	5000	20	$92.14_{\pm 0.06}$	$86.39_{\pm 0.04}$	$90.43_{\pm 0.18}$	$87.29_{\pm 0.32}$	$90.45_{\pm 0.23}$
8	ELM	✓	✓	3	5000	50	$92.13_{\pm 0.04}$	$86.47_{\pm 0.05}$	$90.46_{\pm 0.10}$	$87.49_{\pm 0.32}$	$90.57_{\pm 0.16}$
9	ELM	✓	✓	3	5000	100	$92.15_{\pm 0.09}$	$86.60_{\pm 0.04}$	$90.48_{\pm 0.15}$	$87.71_{\pm 0.29}$	$90.65_{\pm 0.24}$
10	ELM	✓	✓	1	5000	100	$91.99_{\pm 0.01}$	$85.72_{\pm 0.24}$	$90.36_{\pm 0.13}$	$87.12_{\pm 0.24}$	$90.35_{\pm 0.24}$
11	ELM	✓	✓	3	5000	100	$92.15_{\pm 0.09}$	$86.60_{\pm 0.04}$	$90.48_{\pm 0.15}$	$87.71_{\pm 0.29}$	$90.65_{\pm 0.24}$
12	ELM	✓	✓	5	5000	100	$92.20_{\pm 0.04}$	$86.62_{\pm 0.10}$	$90.51_{\pm 0.05}$	$87.76_{\pm 0.29}$	$90.68_{\pm 0.18}$
13	ELM	✓	✓	3	1000	100	$90.64_{\pm 0.10}$	$84.01_{\pm 0.29}$	$89.56_{\pm 0.15}$	$85.48_{\pm 0.17}$	$88.23_{\pm 0.21}$
14	ELM	✓	✓	3	2000	100	$91.35_{\pm 0.08}$	$85.15_{\pm 0.19}$	$89.97_{\pm 0.18}$	$86.24_{\pm 0.22}$	$89.93_{\pm 0.10}$
15	ELM	✓	✓	3	5000	100	$92.15_{\pm 0.09}$	$86.60_{\pm 0.04}$	$90.48_{\pm 0.15}$	$87.71_{\pm 0.29}$	$90.65_{\pm 0.24}$
16	ELM	✓	✓	3	10000	100	$92.75_{\pm 0.10}$	$86.86_{\pm 0.10}$	$90.60_{\pm 0.10}$	$88.11_{\pm 0.24}$	$90.73_{\pm 0.16}$

Table 3: Ablation studies on AnaCP with DINO-v2 as the PTM. All reported values are A_{last}. CLS indicates the method employed in the classifier (Section 4.3). CP indicates whether the proposed contrastive projection is applied. NR (negative repulsion) shows whether class mean separation is improved through Lemma 4.1; when not used, the original class means serve as target prototypes. The table also includes variations in the RP dimension (D), the number of heads in the CP layer (H), and the number of feature representations generated per class (R).

5.4 Ablation Studies

We conduct a series of ablations to evaluate the impact of key components, as shown in Table 3. **Rows 1-3** examine the effect of the CP layer. Removing CP and directly applying an NCM classifier to PTM features results in a significant drop in absolute accuracy between 2.93% to 19.97% (row 1 vs. row 3). This highlights the importance of CP in enhancing feature representations, as illustrated in Figure 1. The effect of negative repulsion (NR), which increases the separation between class means (Section 4.2) is also evident; disabling NR and relying only on positive alignment with the original class means as target prototypes leads to an accuracy reduction from 0.38% to 3.23% (row 2 vs. row 3). This confirms that NR is a critical component for improving class separability.

The role of classifier is explored in **Rows 4-6**, where NCM is replaced by ELM. We find that using an ELM classifier on average improves the performance by 0.66% (row 3 vs. row 6). Although this requires generating feature representations as pseudo-replay for training the ELM classifier, as explained in Section 4.3. An arbitrary number of feature representations can be sampled from the Gaussian distribution with negligible computational overhead, as detailed in the running time analysis. We observe that increasing the number of generated feature representations R improves performance on some datasets **(Rows 7-9)**. The effects of various numbers of CP heads H **(Rows 10-12)** and RP dimensions D **(Rows 13-16)** are also explored. Generally, increasing these values improves accuracy,

albeit at the cost of additional parameters. For the main results, we have used $D = 5000$ and $H = 3$. We note that even with $H = 1$, AnaCP outperforms all baselines.

5.5 Memory and Running Time Efficiency

AnaCP needs to save C class means, each of size d, and a shared covariance matrix of size $d \times d$ for feature whitening and pseudo-replay feature generation at the input layer (Figure 1). While a separate covariance matrix can be maintained for each class, this would result in a significant increase in memory as the number of classes grows. As shown in Table 1, using a shared covariance matrix does not compromise accuracy. Each CP head maintains the following: (1) an RP matrix of size $d \times D$, (2) a $D \times D$ Gram matrix, (3) C class prototypes (or means) in the random feature space, each of size D, and (4) a projection matrix W of size $d \times D$. We have H such heads in total. The target prototypes do not need to be stored, as they can be derived from the class means. The classifier layer also maintains an RP matrix of size $d \times D$ and W of size $C \times D$. Unlike the CP layer, this layer doesn't need to store the Gram matrix for incremental updates, as it is computed from generated feature representations after each task.

For a typical configuration with $C = 200$, $d = 768$, $D = 5000$, and $H = 3$, the total number of stored parameters amounts to approximately 106.6 million. The majority of these parameters belong to the $D \times D$ Gram matrices, which does not increase with the number of classes. Also, none of these parameters are trainable, making them suitable for storage in reduced-precision formats such as float8. This would further reduce the total memory usage to approximately 102 MiB.

While AnaCP may have a higher parameter count compared to some other methods, the design is balanced with exceptional computational efficiency. For example, CIFAR100 with the DINO-v2 PTM requires only 9 minutes and 18 seconds for training. 6 minutes and 5 seconds of this time are dedicated to the FSA, and 3 minutes and 8 seconds are spent passing inputs through the PTM to extract features. The core operations of AnaCP, including updating the CP layer and training the classifier with generated features, collectively take only 5 seconds. Thus, despite the larger parameter count, these operations are extremely fast. For comparison, a baseline such as CODA-Prompt that requires task-specific training takes 3 hours and 47 minutes to complete training on CIFAR100, and joint fine-tuning requires 1 hour and 58 minutes.

5.6 Discussion

Our results support the proposition that a strong PTM is the key to CL, and that representation-level forgetting is not the primary limitation. Even simple CIL strategies, when paired with a fixed PTM, can achieve near-optimal performance. This view aligns with neuroscience evidence that the human brain maintains stable representations, despite minor drift over time [69, 70]. Cortical activity also preserves a stable similarity structure that supports consistent perception and behavior, even as neural responses gradually shift [71, 70]. The human brain can thus be viewed as an innate PTM refined through evolution, which provides a stable foundation for continual adaptation. Similarly, large-scale pre-training in machine learning resembles this biological evolution and development, producing reusable representations that enable CL without any forgetting. For further discussion, see [72].

6 Conclusion

CIL poses a significant challenge in continual learning. Recent analytic methods with PTMs have shown promise. However, since they cannot learn or adapt features obtained from PTMs to suit specific CIL tasks, their performance is still suboptimal. This paper proposed a novel and principled method, called AnaCP, to adapt features extracted from PTMs analytically, which offers a highly efficient and accurate solution for CIL. Empirical evaluations demonstrate that AnaCP outperforms strong baselines, but more importantly, if paired with a strong PTM, its accuracy can be comparable to the joint training upper bound.

Limitations: When using a strong PTM like DINO-v2, AnaCP achieves accuracy on par with the joint fine-tuning upper bound. However, with a weaker PTM such as MoCo-v3, AnaCP cannot match joint fine-tuning accuracy due to the critical role of PTM features played in our method. While strong PTMs are readily available today, improving AnaCP's performance with weaker PTMs remains a meaningful goal. Additionally, our work currently focuses CIL. We believe AnaCP can be extended to TIL and domain-incremental learning (DIL) scenarios with appropriate adaptations.

Acknowledgments

The work of Saleh Momeni and Bing Liu was supported in part by three NSF grants (IIS-2229876, IIS-1910424, and CNS-2225427), and an NVIDIA's Academia Grant, which provides cloud compute via its Saturn Cloud.

References

[1] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

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

[3] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[4] Zixuan Ke and Bing Liu. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*, 2022.

[5] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[6] Haowei Lin, Yijia Shao, Weinan Qian, Ningxin Pan, Yiduo Guo, and Bing Liu. Class incremental learning via likelihood ratio based task prediction. *ICLR-2024*, 2024.

[7] Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[8] Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, and Bing Liu. Learnability and algorithm for continual learning. In *International Conference on Machine Learning*, pages 16877–16896. PMLR, 2023.

[9] Yutao Yang, Jie Zhou, Xuanwen Ding, Tianyu Huai, Shunyu Liu, Qin Chen, Yuan Xie, and Liang He. Recent advances of foundation language models-based continual learning: a survey. *ACM Computing Surveys*, 2024.

[10] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19148–19158, 2023.

[11] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022.

[12] Huiping Zhuang, Zhenyu Weng, Hongxin Wei, Renchunzi Xie, Kar-Ann Toh, and Zhiping Lin. Acil: Analytic class-incremental learning with absolute memorization and privacy protection. *Advances in Neural Information Processing Systems*, 35:11602–11614, 2022.

[13] Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems (NeurIP-2023)*, 36, 2023.

[14] Saleh Momeni, Sahisnu Mazumder, and Bing Liu. Continual learning using a kernel-based method over foundation models. In *AAAI-2025*, 2025.

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

[16] Aristeidis Panos, Yuriko Kobe, Daniel Olmeda Reino, Rahaf Aljundi, and Richard E Turner. First session adaptation: A strong replay-free baseline for class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18820–18830, 2023.

[17] Nan-Ying Liang, Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423, 2006.

[18] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, 2011.

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

[20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

[21] 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.

[22] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[23] Tianlin Liu, Lyle Ungar, and João Sedoc. Continual learning for sentence representations using conceptors. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 2019.

[24] Dingcheng Li, Zheng Chen, Eunah Cho, Jie Hao, Xiaohu Liu, Fan Xing, Chenlei Guo, and Yang Liu. Overcoming catastrophic forgetting during domain adaptation of seq2seq language generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

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

[28] Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. *Advances in Neural Information Processing Systems*, 33:9900–9911, 2020.

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

[30] Qingbin Liu, Xiaoyan Yu, Shizhu He, Kang Liu, and Jun Zhao. Lifelong intent detection via multi-strategy rebalancing. *arXiv preprint arXiv:2108.04445*, 2021.

[31] Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Elle: Efficient lifelong pre-training for emerging data. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2789–2810, 2022.

[32] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 226–227, 2020.

[33] Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):4069, 2020.

[34] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9296–9305, 2022.

[35] Zheng Gong, Kun Zhou, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. Continual pre-training of language models for math problem understanding with syntax-aware memory network. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5923–5933, 2022.

[36] Fu-Yun Wang, Da-Wei Zhou, Liu Liu, Han-Jia Ye, Yatao Bian, De-Chuan Zhan, and Peilin Zhao. Beef: Bi-compatible class-incremental learning via energy-based expansion and fusion. In *The Eleventh International Conference on Learning Representations*, 2022.

[37] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021.

[38] Chengwei Qin, Chen Chen, and Shafiq Joty. Lifelong sequence generation with dynamic module expansion and adaptation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6701–6714. Association for Computational Linguistics, 2023.

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

[40] Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A Smith, and Luke Zettlemoyer. Demix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 5557–5576, 2022.

[41] Binzong Geng, Fajie Yuan, Qiancheng Xu, Ying Shen, Ruifeng Xu, and Min Yang. Continual learning for task-oriented dialogue system with iterative network pruning, expanding and masking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL/IJCNLP)*, 2021.

[42] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

[43] Junpeng Liu, Kaiyu Huang, Hao Yu, Jiuyi Li, Jinsong Su, and Degen Huang. Continual learning for multilingual neural machine translation via dual importance-based model division. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12011–12027, 2023.

[44] Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, and Bing Liu. A theoretical study on solving continual learning. *Advances in neural information processing systems*, 35:5065–5079, 2022.

[45] Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. Rehearsal-free modular and compositional continual learning for language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 469–480, 2024.

[46] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13588–13597, 2020.

[47] 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. In *European Conference on Computer Vision*, pages 631–648. Springer, 2022.

[48] 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*, pages 23638–23647, 2024.

[49] 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, pages 20699–20707, 2025.

[50] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11909–11919, 2023.

[51] Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. *Advances in Neural Information Processing Systems*, 36:69054–69076, 2023.

[52] 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*, pages 11847–11857, 2023.

[53] Anurag Roy, Riddhiman Moulick, Vinay K Verma, Saptarshi Ghosh, and Abir Das. Convolutional prompting meets language models for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23616–23626, 2024.

[54] Huiping Zhuang, Yizhu Chen, Di Fang, Run He, Kai Tong, Hongxin Wei, Ziqian Zeng, and Cen Chen. GACL: Exemplar-free generalized analytic continual learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2024.

[55] Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. A simple baseline that questions the use of pretrained-models in continual learning. *arXiv preprint arXiv:2210.04428*, 2022.

[56] 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. *International Journal of Computer Vision*, 2024.

[57] Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 220–221, 2020.

[58] Shaoning Pang, Seiichi Ozawa, and Nikola Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE transactions on Systems, Man, and Cybernetics, part B (Cybernetics)*, 35(5):905–914, 2005.

[59] Liangzu Peng, Juan Elenter, Joshua Agterberg, Alejandro Ribeiro, and René Vidal. Loranpac: Low-rank random features and pre-trained models for bridging theory and practice in continual learning. *ICLR*, 2025.

[60] AK Md Ehsanes Saleh, Mohammad Arashi, and BM Golam Kibria. *Theory of ridge regression estimation with applications*. John Wiley & Sons, 2019.

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

[62] 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 ICCV*, pages 8340–8349, 2021.

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

[64] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7:7, 2015.

[65] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3973–3981, 2015.

[66] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2023.

[67] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9620–9629. IEEE Computer Society, 2021.

[68] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

[69] Walter G Gonzalez, Hanwen Zhang, Anna Harutyunyan, and Carlos Lois. Persistence of neuronal representations through time and damage in the hippocampus. *Science*, 365(6455):821–825, 2019.

[70] Zvi N. Roth and Elisha P. Merriam. Representations in human primary visual cortex drift over time. *Nature Communications*, 14(4422), 2023.

[71] David BT McMahon, Adam P Jones, Igor V Bondar, and David A Leopold. Face-selective neurons maintain consistent visual responses across months. *Proceedings of the National Academy of Sciences*, 111(22):8251–8256, 2014.

[72] Saleh Momeni and Bing Liu. Achieving upper bound accuracy of joint training in continual learning. *arXiv preprint arXiv:2502.12388*, 2025.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction match our claims and the paper's scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations is discussed in the conclusion.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: The only novel theoretical formula included in this paper is lemma 4.1, which is proven in the appendix.

   Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: We provided all information needed to reproduce the results and the code in included in the supplementary materials.

Guidelines:

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

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

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

Answer: [Yes]

Justification: The code is included in the supplementary materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We use the official data splits, and all the hyperparameters used are reported.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The standard deviation of the accuracies are reported.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The computer resources and the execution time of the experiments are reported.

Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research complies with NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: This paper studies a standard machine learning problem with no direct societal impact.

    Guidelines:

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

11. **Safeguards**

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

Answer: [NA]

Justification: There is no high risk misuse.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All the datasets and models are properly cited.

Guidelines:

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

13. **New assets**

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

Answer: [NA]

Justification: Paper does not release new assets.

Guidelines:

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

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

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

Answer: [NA]

Justification: The paper does not include human research experiments.

Guidelines:

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

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

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

Answer: [NA]

Justification: The paper does not include human research experiments.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: LLMs were only used for editing without any impact on the core methodology.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Technical Appendices

Lemma A.1 (Lemma 4.1)**.** *Denote $w_1, \dots, w_C \in \mathbb{R}^C$ as arbitrary vectors, and $e_1, \dots, e_C \in \mathbb{R}^C$ as a set of orthogonal bases of $\mathbb{R}^C$. Denote $\langle x, y\rangle = x^\top y$ as the inner product and $\theta(x,y) = \frac{x^\top y}{||x||\cdot||y||}$ as the cosine similarity. There exist $\alpha > 0$ and*

$$\delta_i = \begin{cases} 1, \textit{ if } \sum_{j\neq i} \frac{1}{||w_j||}\left[(2\cdot 1_{\langle w_i,w_j\rangle\geq 0})\cdot\langle e_i, w_j\rangle\cdot||w_i||^2 - \langle e_i,w_i\rangle\cdot|\langle w_i,w_j\rangle|\right] < 0, \\ 0, \textit{ if } \sum_{j\neq i} \frac{1}{||w_j||}\left[(2\cdot 1_{\langle w_i,w_j\rangle\geq 0})\cdot\langle e_i, w_j\rangle\cdot||w_i||^2 - \langle e_i,w_i\rangle\cdot|\langle w_i,w_j\rangle|\right] = 0, \\ -1, \textit{ else}, \end{cases}$$

s.t.

$$\sum_i\sum_{j\neq i}|\theta(w_i+\alpha\delta_i e_i, w_j+\alpha\delta_j e_j)| \leq \sum_i\sum_{j\neq i}|\theta(w_i,w_j)|. \tag{18}$$

Proof. Let

$$f_i(\alpha) = \sum_{j\neq i}\frac{|\langle w_j, w_i+\alpha\delta_i e_i\rangle|}{||w_j||\cdot||w_i+\alpha\delta_i e_i||}.$$

For simplicity, denote

$$p_{i,j}(\alpha) = |\langle w_i+\alpha\delta_i e_i, w_j+\alpha\delta_j e_j\rangle| = |\langle w_i, w_j\rangle + \alpha\delta_i\langle e_i, w_j\rangle|$$

and

$$q_i(\alpha) = ||w_i+\alpha\delta_i e_i|| = \sqrt{\langle w_i+\alpha\delta_i e_i, w_i+\alpha\delta_i e_i\rangle}.$$

Therefore, we have

$$f_i(\alpha) = \sum_{j\neq i}\frac{1}{||w_j||}\cdot\frac{p_{i,j}(\alpha)}{q_i(\alpha)}.$$

Since $\frac{d|x|}{dx} = 2\cdot 1_{x\geq 0} - 1$, we have

$$p'_{i,j}(\alpha)|_{\alpha=0} = (2\cdot 1_{\langle w_i,w_j\rangle\geq 0})\cdot\delta_i\langle e_i, w_j\rangle.$$

Since $\langle w_i+\alpha\delta_i e_i, w_i+\alpha\delta_i e_i\rangle = \langle w_i, w_i\rangle + 2\delta_i\langle e_i, w_i\rangle + \alpha^2$, we have

$$q'_i(\alpha)|_{\alpha=0} = \frac{1}{2}\frac{2\delta_i\langle e_i,w_i\rangle + 2\alpha}{\sqrt{\langle w_i+\alpha\delta_i e_i, w_i+\alpha\delta_i e_i\rangle}}|_{\alpha=0} = \frac{\delta_i\langle e_i, w_i\rangle}{||w_i||}$$

Therefore, we have

$$\begin{aligned} f'_i(\alpha)|_{\alpha=0} &= \sum_{j\neq i}\frac{1}{||w_j||}\cdot\frac{p'_{i,j}(\alpha)q_i(\alpha) - p_{i,j}(\alpha)q'_i(\alpha)}{q_i^2(\alpha)}|_{\alpha=0} \\ &= \sum_{j\neq i}\frac{1}{||w_j||||w_i||^2}\left[(2\cdot 1_{\langle w_i,w_j\rangle\geq 0}-1)\cdot\delta_i\langle e_i,w_j\rangle\cdot||w_i|| - |\langle w_i,w_j\rangle|\cdot\frac{\delta_i\langle e_i,w_i\rangle}{||w_i||}\right] \\ &= \frac{\delta_i}{||w_i||^3}\sum_{j\neq i}\frac{1}{||w_j||}\left[(2\cdot 1_{\langle w_i,w_j\rangle\geq 0}-1)\cdot\langle e_i,w_j\rangle\cdot||w_i||^2 - \langle e_i,w_i\rangle\cdot|\langle w_i,w_j\rangle|\right], \end{aligned}$$

which shows that δ_i's defined above guarantee $f'_i(\alpha)|_{\alpha=0} \leq 0$.

Let

$$g_j(\alpha;\hat{\alpha}) = \sum_{i\neq j}\frac{|\langle w_j+\alpha\delta_j e_j, w_i+\hat{\alpha}\delta_i e_i\rangle|}{||w_j+\alpha\delta_j e_j||\cdot||w_i+\hat{\alpha}\delta_i e_i||}.$$

Since $f'_i(\alpha)$ is a continuous function, we could take a sufficiently small $\hat{\alpha}$ s.t. $f'_i(\alpha')$ has the same sign when $\alpha' \in [0,\hat{\alpha}]$. Then for $\alpha' \in [0,\hat{\alpha}]$, the δ_i's defined above also guarantee $g'_j(\alpha;\alpha')|_{\alpha=0} \leq 0$. Choosing $\alpha = \alpha'$ gives the result. □