# Large Pre-trained Network for Class Incremental Learning

Wonseok Lee
DRL
dnjstjr1017@snu.ac.kr

Eungi Kim
VIP
kuman5262@snu.ac.kr

Kyubyung Chae
LAAL
kyubyung.chae@snu.ac.kr

## Abstract

*Deep learning has achieved remarkable success in supervised learning-based models, but the reliance on large amounts of labeled data poses challenges in real-world scenarios where new data keeps arriving. Continual and incremental learning methods aim to address these challenges. However, the issue of catastrophic forgetting, where a model forgets previously learned information, remains a significant hurdle. In this paper, we explore the use of pre-trained models as feature extractors to address the challenge of catastrophic forgetting in Class Incremental Learning (CIL). We evaluate the performance of pre-trained models on CIL tasks and compare them with state-of-the-art methods, demonstrating their ability to reduce forgetting. Additionally, we propose a novel method based on orthogonal projection to represent a good feature space for classification tasks while maintaining performance. We highlight the potential of pre-trained models for class incremental learning tasks and the negative impacts of full fine-tuning.*

## 1. Introduction

Deep learning has revolutionized various domains, achieving remarkable success in supervised learning-based models. However, reliable generalization performance requires access to large amounts of labeled data, posing significant challenges in real-world scenarios where new data keeps arriving, and reusing previously labeled data is often not feasible due to various reasons such as cost, time, or privacy concerns. For instance, in image classification tasks for products in a warehouse, the model must classify both new and existing products accurately, which requires the ability to learn continuously without forgetting previously learned knowledge.

To overcome these challenges, continual and incremental learning has garnered significant attention, but existing methods still struggle to preserve previously acquired knowledge over many cycles of short incremental learning steps. This phenomenon where a model forgets its previously learned information upon learning new tasks is known as *catastrophic forgetting* [20]. In this paper, we focus on *Class Incremental Learning*(CIL) [23, 2, 11, 12, 27, 21], where the classifier must learn classes by steps, in training cycles called tasks[8].

Previous studies on CIL generally assume that deep learning networks are trained from random initialization, which may not always be the case. Therefore, we investigate the effectiveness of employing large pre-trained models such as ResNet [10] and CLIP [22], which demonstrate exceptional performance as feature extractors across multiple domains, in the context of CIL. Specifically, we examine whether leveraging a proficient pre-trained feature extractor can overcome *catastrophic forgetting*.

While pre-trained weights provide a solid foundation as a good starting point, they do not guarantee optimal performance in the final model. Fine-tuning, although a valuable technique supported by studies [15, 5, 31], requires careful application, as finding the optimal balance can be challenging. In fact, some studies [17, 4] indicate that excessive fine-tuning in downstream tasks can lead to overfitting and hinder generalization. It is also important to note that fine-tuning incurs higher computational costs compared to pre-trained initialization.

Consequently, we challenge the effectiveness of fine-tuning and propose that zero fine-tuning may yield superior outcomes when utilizing a suitable pre-trained feature extractor. In essence, we determine whether a pre-trained feature extractor can outperform previous CIL benchmarks without undergoing any learning on the new dataset.

Furthermore, we present a novel method specifically tailored for CIL, which utilizes pre-trained models as feature extractors without engaging in the traditional fine-tuning method. Inspired by the concepts of *PROJECT and PROBE ($Pro^2$)* [4], our method proposes to learn a linear projection that not only maps pre-trained embeddings onto orthogonal directions but also maintains the predictability of labels. This projection effectively filters out unnecessary information while emphasizing the relevant features essential for accurate label prediction. By adopting this approach, we aim to enhance model performance through the refinement of pre-trained embeddings.

In our experiments, we evaluate the performance of pre-trained models compared to random initialization on the CIFAR100 dataset. Additionally, we conduct a comparative analysis of pre-trained models with and without fine-tuning to assess the efficacy of fine-tuning in the context of CIL. Lastly, we investigate the promising performance of our proposed method, which involves projecting pre-trained embeddings onto a lower-dimensional space while enforcing orthogonality.

Our results demonstrate that pre-trained models can be more effective than training from random initialization in CIL tasks, which has practical implications. We observe that good feature extractors can achieve superior accuracy and alleviate the catastrophic forgetting phenomenon, outperforming complex and resource-intensive methods. Moreover, our investigation reveals that fine-tuning pre-trained models may diminish the capabilities of the feature extractor. Conversely, our alternative approach, which fully utilizes the zero fine-tuned feature extractor, exhibits promising performance in mitigating the catastrophic forgetting phenomenon.

We summarize our motivation and main contributions as follows:

**Motivation**

- Why not use pre-trained feature extractor for CIL tasks? (Section 4)
- How can we utilize the pre-trained features better for CIL tasks? (Section 5)

**Contributions**

- We demonstrate the impacts of the feature extractor in CIL tasks by applying pre-trained weights.
- We investigate the potential negative effects of full fine-tuning on pre-trained feature extractors in CIL tasks.
- We propose a novel method using pre-trained weights for CIL tasks which effectively represents the feature space through orthogonal projection.

## 2. Related Works

The learning process of Class Incremental Learning (CIL) can be divided into two stages: an initial stage where the network is trained with the first given classes and an incremental stage where additional classes are learned. During the incremental stage, it is assumed that access to the entire dataset used in the previous stage is not possible [23]. The main objective of CIL is to address the issue of catastrophic forgetting that arises in such situations. To prevent forgetting, current CIL research employs three main approaches.

### 2.1. Data-Centric Approaches

Storing some of the previously learned data, referred to as exemplars [23, 27, 2], is considered to prevent catastrophic forgetting. Research has been conducted on how to utilize these exemplars to mitigate forgetting. Another approach involves training a separate network to learn the distribution of the data used in the previous training and generating exemplar images from it [25, 9]. In addition to using stored data for replay, the GEM [19] method sets constraint optimization based on stored data to facilitate network learning.

### 2.2. Model-Centric Approaches

The Model-Centric CIL method is an approach that focuses on the model used in the learning process. This involves adding to the structure of the model as the learning progresses. When there is not enough space to learn new classes, one approach is to add neurons [29], while another approach involves retraining the backbone network itself [28]. The Model-Centric approach is characterized by an increase in storage space for model-related information as the learning progresses. However, the information related to the model that increases with learning is not only used for model expansion. An alternative approach is to store previously learned model parameters and use them as regularizers when conducting incremental learning steps [14, 30, 3].

### 2.3. Algorithm-Centric Approaches

The Algorithm-Centric CIL method focuses on designing algorithms for class incremental learning. The *Knowledge Distillation* method aims to prevent forgetting by regularizing the output of the current training network with that of a previously trained network [18, 23, 8]. Another approach involves assuming an *Oracle* model that has all the information for every class and enforcing its characteristics on the class incremental learning step. This method aims to allow even a subset of data to follow the characteristics of the Oracle model by not deviating too much from its features, such as features, logit, and weight [12, 32, 1].

We focus on feature extractors, in contrast to the aforementioned CIL approaches (Section 2.1 - 2.3). We expect better performance when starting the learning process from a good initialization point, and therefore, we intend to investigate the significance of using pre-trained weights such as ResNet [10] and CLIP [22]. Furthermore, we highlight the limitations of traditional fine-tuning processes which deteriorate the effectiveness of the proficient feature extractor. Consequently, we propose a novel method inspired by *Pro*$^2$[4] that learns to effectively refine the pre-trained embeddings for CIL task in Section 5.1.

### 2.4. Fine-tuning and Linear Probing

When applying a pre-trained model to a downstream task, two commonly used methods are full fine-tuning, which involves updating all the model parameters, and linear probing, which only updates the last linear layer known as the *head*. While it is widely acknowledged that fine-tuning generally improves accuracy within the distribution (ID), Kumar et al. [17] theoretically demonstrate that fine-tuning can distort the pre-trained features. The results in this paper reveal that fine-tuning can actually lead to lower accuracy compared to linear probing when the pre-trained features are of high quality and when there is a significant distribution shift out-of-distribution (OOD). These findings support our approach of leveraging frozen pre-trained weights, as fine-tuning can exacerbate detrimental forgetting in the CIL tasks when dealing with the label shifts (class increments).

*PROJECT and PROBE (Pro$^2$)* [4] is one of the transfer learning frameworks that adapts a pre-trained model to distribution shifts. Its primary objective is to effectively learn from limited data and adapt a pre-trained model to a target distribution by leveraging feature interpolation with linear probing. In other words, Pro$^2$ enables the model to effectively handle distribution shifts. Specifically, when faced with few-shot labeled target data, it enhances robustness and generalization performance by mitigating the influence of spurious correlations. In this context, the distribution shift means covariate shifts. From the perspective of Class Incremental Learning (CIL), we can consider the exemplar set, which contains few images of complete class label information, as our target distribution. Therefore, we adopt the $Pro^2$ approach to tackle this challenge.

### 2.5. Contrastive Language-Image Pre-training

*CLIP (Contrastive Language-Image Pre-training)* [26], a widely recognized pre-trained network in the field of vision and language research, operates within the domain of learning visual representations using natural language guidance [6, 24, 13]. CLIP adopts a *shallow-interaction design*, where an independent visual encoder and a text encoder encode input images and text, respectively. The similarity score between the image and text is determined by taking the dot product of the outputs from both encoders. Pre-training CLIP utilizes contrastive loss, where the model learns to differentiate between aligned pairs and randomly sampled pairs. Notably, CLIP benefits from an extensive source of supervision comprising 400 million image-text pairs sourced from the internet, eliminating the need for human annotation. As a result, CLIP attains state-of-the-art performance in various image classification and image-text retrieval tasks, even in a zero-shot setting.

Currently, there are limited instances of utilizing CLIP in CIL tasks. Prior study [26] has demonstrated performance enhancements in zero-shot settings by leveraging both text encoders and image encoders. However, our approach solely focuses on utilizing image encoders and does not incorporate contrastive learning techniques. Moreover, the performance of previous work does not surpass that of the naive method that we experiment with, thus limiting the extent of our discussion on this aspect.

## 3. Preliminaries

**Problem Formulation** We now elaborate on Class Incremental Learning (CIL) setting, which aims to learn from a dynamic stream with incoming new classes. Our framework assumes a sequence of $B$ training tasks $\left\{\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^B\right\}$ with no overlapping classes. Each task, $\mathcal{D}^b = \left\{\left(x_i^b, y_i^b\right)\right\}_{i=1}^{n_b}$, represents the b-th incremental step with $n_b$ training instances. $x_i^b \in \mathcal{X}$ denotes an instance belonging to class $y_i^b \in Y^b$, where $Y^b$ represents the label space for task $\mathcal{D}^b$. Importantly, there are no shared classes between different tasks.

The ultimate goal of CIL is to continuously learn a classification model that encompasses all classes $Y = \bigcup_{b=1}^{B} Y^b$. The model must effectively learn from the current task $\mathcal{D}^b$ while retaining the knowledge acquired from previous tasks $\left\{\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^i\right\}$, where $i = b - 1$. By accomplishing this, the model adapts to the evolving nature of the stream while maintaining proficiency in classifying all previously encountered and newly introduced classes.

**Datasets and Evaluation protocol** We evaluate the model performance on a popular image classification dataset: CIFAR100 [16]. To validate our model and compare it with baseline approaches, we adopt the experimental protocol proposed by Hou et al. [12]. Specifically, we first train our model on 50% of the available classes in each dataset. Then, we perform incremental learning by dividing the remaining 50% of classes into equal portions and adding one portion at a time to train the model. At each step, we only train the model on the newly added classes.

We conduct a series of experiments to assess the model performance under different incremental learning scenarios. Specifically, we train our model using 5, 10, 25, and 50 incremental learning steps, with different numbers of classes added at each step. In the case of CIFAR100, we add 10, 5, 2, and 1 classes for each incremental step, respectively, across all four settings (i.e. iCIFAR100). In Table 1 and 3, we indicate the number of new classes (1, 2, 5, 10) added per incremental step below each step (50, 25, 10, 5 steps). This setting enables us to evaluate the model's performance under varying degrees of incremental learning complexities.

**Metrics** We adopt the global metric proposed by Rebuffi et al. [23], the *average incremental accuracy*, to ensure a

fair comparison with the baseline approaches. At the end of each incremental training step, we compute the model's accuracy on the test dataset, considering all the classes that have been trained previously. This metric considers the entire history of the run, including the first task, by averaging the accuracy computed at the end of each task $\mathcal{D}_b$:

$$\text{Avg. Incremental Accuracy}(\mathcal{D}_B) = \frac{1}{B}\sum_{i=1}^{B}\text{acc}(\mathcal{D}^{1:i})$$

# 4. Effects of feature extractors

We first show the superiority of utilizing a pre-trained network over complex methods with random initialization. Moreover, we illustrate in a straightforward manner that even when using a pre-trained model, zero fine-tuning may be more advantageous than conducting a full fine-tuning process.

## 4.1. Feature Extractor Replacement

In feature extractor replacement, we do not change the CIL method of the baselines but only substitute the feature extractor layers with pre-trained weights, such as CLIP[26] or ResNet [10]. Within the realm of CIL tasks, PODNet [8] stands out as one of the state-of-the-art methods, surpassing the performance of BiC and iCaRL [23]. By employing pre-trained weights from ImageNet instead of random initialization, we can attain superior performance using the same training method. This highlights the effectiveness of the feature extractor replacement technique, demonstrating the advantages of using pre-trained weights as a favorable starting point for CIL.

## 4.2. Naive method with Zero fine-tuning

To demonstrate the effectiveness of pre-trained weights in CIL task without the need for additional learning (zero fine-tuning), we propose a simple approach known as the *naive method*. In the naive method, our first step is to establish class prototypes, denoted as $\mu$, by calculating the average feature vectors of randomly selected exemplars for each class (lines 3-5 in Algorithm 1). These feature vectors are extracted from the training samples using a pre-trained image encoder, such as ResNet or CLIP. In practical implementation, we typically choose a value of $k$ equal to 20, aligning with the common approach used by most CIL methodologies that employ exemplars.

During the testing phase, we compute feature vectors for the test samples and utilize the prototypes to calculate Nearest Neighbor Scores for each class. Our predictions are made based on the class with the highest score (line 9 in Algorithm 1). Importantly, this process strictly prohibits access to the test samples during the training phase, and no fine-tuning is performed.

**Algorithm 1** Naive method with Zero fine-tuning

```
// Training Phase
```

**Require:** train set $\{x_i, y_i\}_{i=1}^{K}$, number of exemplars $k$, pre-trained network $f : \mathcal{X} \to \mathbb{R}^d$

1: $\Phi \leftarrow \varnothing$
2: **for** $b \leftarrow 1$**to**$B$ **do**
3: **for** $y^b$ in $Y^b$ **do**
4: $X \leftarrow$ image set $\{x_1, \cdots, x_k\}$ of class $y^b$
5: $\mu_{y^b} \leftarrow \frac{1}{n_b}\sum_{x\in X} f(x)$
6: **end for**
7: $\Phi \leftarrow \Phi \cup \left\{\mu_{y^b}\right\}_{y_b \in Y^b}$
8: **end for**
9: **return** set of prototypes $\Phi$

```
// Test Phase
```

**Require:** test image $x$, set of prototypes $\Phi$, pre-trained network $f : \mathcal{X} \to \mathbb{R}^d$

1: $y^* = \text{argmin}_{y\in Y}\|f(x) - \mu_y\|$
2: **return** class label $y^*$

To evaluate the efficacy of naive method, we conduct two experiments: *full samples* and *20 samples*, as outlined in Table 1. The *full samples* experiment utilizes all the available exemplars from the training dataset. In contrast, the *20 samples* experiment uses only 20 exemplars, enabling a fair comparison with iCaRL, as previously mentioned. iCaRL assumes the use of a limited number of exemplars, typically 20, due to memory constraints in real-world scenarios.

By adopting this approach, we can examine whether a well-performing feature extractor alone can surpass previous CIL methods in mitigating catastrophic forgetting, thereby showcasing its superiority in the CIL task.

**Table 1:** Comparison of CIL methods with pre-trained feature extractors on CIFAR-100 Dataset (* indicates the use of frozen pre-trained weights).

| Method | Pre-trained | CIFAR100 | | | |
|---|---|---|---|---|---|
| | | 50 steps 1 | 25 steps 2 | 10 steps 5 | 5 steps 10 |
| BiC | – | 47.09 | 48.96 | 53.21 | 56.86 |
| iCaRL | – | 44.20 | 50.60 | 53.78 | 58.08 |
| PODNet | – | 50.84 | 55.49 | 59.30 | 61.61 |
| PODNet | ResNet18 | 53.26 | 57.58 | 61.12 | 63.09 |
| Naive-random | CLIP-ViT-B/32* | 64.58 | 64.58 | 64.58 | 64.75 |
| Naive-full | CLIP-ViT-B/32* | 69.01 | 69.00 | 69.01 | 69.13 |
| Naive | CLIP-ViT-B/32* | 69.44 | 69.45 | 69.47 | 69.58 |

We now discuss the results presented in Table 1. First, we compare the performance between random initialization and utilizing pre-trained weights. In the third and fourth rows of Table 1, we observe a noticeable improvement in performance when replacing the feature extractors with pre-trained weights in PODNet method. Additionally, in

the last three rows of Table 1, we observe that the *Naive* methods demonstrate satisfactory performance, indicating the effectiveness of pre-trained weights. Specifically, the *Naive-random* method denotes randomly selected 20 samples, *Naive-full* represents the utilization of the full sample set, and *Naive* signifies 20 samples selected using prototypes. Consequently, the results highlight the significant impact on performance achieved by utilizing a pre-trained CLIP feature extractor, surpassing the effectiveness of complex methods such as iCaRL and PODNet.

### 4.3. Efficiency of zero fine-tuning

Full Fine-tuning requires substantial computation costs. The *zero fine-tuning* approach, however, efficiently achieves satisfactory results in CIL task by using pre-trained weights without additional training iterations. This eliminates the time-consuming process of fine-tuning the entire network on the target dataset. The absence of fine-tuning not only saves computational resources and training time but also mitigates overfitting and enhances the model's generalization and robustness.

Moreover, the zero fine-tuning effectively addresses catastrophic forgetting since there is no update in backbone network or linear layer. The model can retain previously learned knowledge while accommodating new classes or tasks by focusing on utilizing pre-trained weights and feature extraction. This adaptability makes it well-suited for incremental learning scenarios, where the model's capabilities can be expanded without sacrificing performance in previously learned classes.

**Table 2:** Comparison of fine-tune with iCarL + CLIP-ViT-B/32 on iCIFAR-100 Dataset.

| Fine-tune | Learning rate | 50 steps |
|---|---|---|
| Full | 0.01 | 4.90 |
| Full | 0.0001 | 34.97 |
| Full | 0.000001 | 70.06 |

Hyperparameter tuning is a labor-intensive process, particularly for CLIP due to its sensitivity to hyperparameters. Determining suitable hyperparameters, such as learning rate, batch size, regularization strength, and others, can be a non-trivial task. Finding the right combination of hyperparameters typically requires a comprehensive search through various possibilities. This search can be time-consuming and computationally expensive, as it often involves training and evaluating multiple models with different hyperparameter configurations.

*Zero fine-tuning* approach is robust to hyperparameters, while full fine-tuning is sensitive to them. As observed in the provided Table 2, in *zero fine-tuning* cases, there is little variation in performance with different learning rates. However, in full fine-tuning cases, the performance significantly differs based on the learning rate. Therefore, *zero fine-tuning* approach is less sensitive to hyperparameter tuning, whereas full fine-tuning is highly responsive to hyperparameter adjustments.

## 5. Project and Probe in CIL

In this section, we present our innovative method of utilizing pre-trained weights without fine-tuning. We notice, in Section 4, that employing the frozen pre-trained feature extractor produces encouraging results. Expanding on this discovery, we introduce a simple, computationally efficient, and data-efficient method named *PROJECT and PROBE (Pro$^2$)* inspired by Chen et al.[4], which incorporates *orthogonal projection* and linear probing.

### 5.1. Orthogonal projection method

Our main goal is to develop an approach that optimizes feature representation using samples of new classes on incoming datasets and limited examples for existing classes.

In order to explain the orthogonal projection method, we first describe some specific notations. The loss function $\mathcal{L}_{\mathcal{D}}$ is defined as the expected value of the *cross-entropy loss* function $l\left(w^T f(x), y\right)$, where $(x, y)$ is sampled from the dataset $\mathcal{D}$, i.e.

$$\mathcal{L}_{\mathcal{D}}\left(w^T f(x), y\right) \stackrel{\text{def}}{=} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[l\left(w^T f(x), y\right)\right]$$

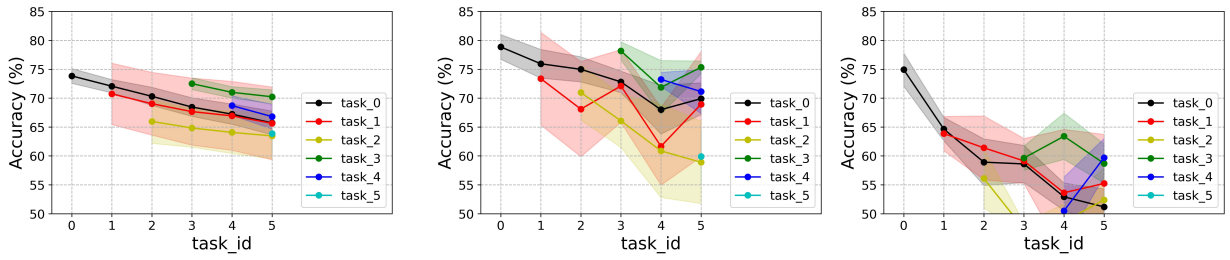
To denote the projection matrix for the subspace span $(\{\pi_i\}_{i=1}^m)$, we use the symbol $\Pi_m$, which is defined as the product of the matrices $[\pi_1, \cdots, \pi_m]$ and $[\pi_1, \cdots, \pi_m]^T$, i.e.

$$\Pi_m = [\pi_1, \cdots, \pi_m]\,[\pi_1, \cdots, \pi_m]^T$$

$Pro^2$ initially learns a projection of pre-trained embedding vectors that are optimized to extract a wide range of features, each of which is indicative of specific labels. Specifically, we project the pre-trained feature embeddings onto a set of predictive features using a source dataset. To ensure that each projected dimension contains unique information not found in other dimensions, we enforce orthogonality during the projection process. We anticipate that this acquired feature space will efficiently encompass a diverse collection of predictive features while eliminating non-predictive or irrelevant information.

Subsequently, $Pro^2$ trains a linear head to smoothly interpolate between the projected features. Both the linear projection and the head require minimal computational resources, making $Pro^2$ a practical approach for adapting to new class distributions.

**Figure 1:** In our experiments on iCIFAR-100 dataset, we evaluated class-incremental training and measured multi-class accuracies for all observed classes up to a specific time point. Our method consistently outperformed other approaches in this setting. In contrast, linear probing without the orthogonal projection resulted in the worst performance, demonstrating catastrophic forgetting.




**(a)** Naive **(b)** Ours **(c)** Growing basis

**Algorithm 2** Orthogonal projection method

**input:** input training task stream $\mathcal{D}^1, \cdots, \mathcal{D}^B$
**require:** pre-trained network $f : \mathcal{X} \to \mathbb{R}^d$

1: **Dummy classifier training**
$w_{dum} \leftarrow \text{argmin}_w \mathcal{L}_{\mathcal{D}^1}\left(w^T f(x), y\right)$
2: **Orthonormal basis training**
$\pi_1, \cdots, \pi_m \leftarrow \text{argmin}_\pi \mathcal{L}_{\mathcal{D}^1}\left(w_{dum}^T \Pi_m\left(f(x)\right), y\right)$
subject to $\pi_i \perp \pi_j$ for $\forall i, j$
3: **Classifier training**
4: **for** $b \leftarrow 1$**to**$B$ **do**
5: $w_{cls} \leftarrow \text{argmin}_w \mathcal{L}_{\mathcal{D}^{1:b}}\left(w^T \Pi_m\left(f(x)\right), y\right)$
6: **end for**

Our algorithm, as described in Algorithm 2, consists of three main stages: dummy classifier training, additional orthonormal basis training, and final classifier training.

**Dummy classifier training** In the first stage, we train a linear classifier that achieves the highest classification accuracy for the first task dataset $\mathcal{D}^1$. In this process, we utilize the full-dimensional features extracted by a pre-trained network.

**Orthonormal basis training** Proceeding to the second stage, we acquire orthonormal bases that optimize the classification accuracy when employing projected features in conjunction with the pre-trained dummy classifier. We compute the classification accuracy using the first task dataset $\mathcal{D}^1$, while ensuring orthogonality among $m$ orthonormal bases. To preserve the orthogonality constraint, we employ projected gradient descent, a technique introduced in the $Pro^2$ method [4].

**Classifier training** Lastly, we utilize an exemplar set $\mathcal{D}^{1:b}$ to train the ultimate classifier. All $\pi_1, ..., \pi_m$ are used as orthonormal bases for feature extraction. We train the final classifier based on the projected features to enhance the classification accuracy.

## 5.2. Implementation Details

The CLIP [26] model with ViT [7] backbone was employed as a pre-trained network. For the number of orthonormal bases, we select 250, which corresponds to five times the number of classes in the initial training task. We train the classifier linear weight over 20 epochs using a learning rate of 0.05. We conduct a separate training process for orthonormal bases over 20 epochs with a learning rate of 0.01 and weight decay set at 0.01.

## 5.3. Results

In this section, we address the following questions through empirical analysis: Can $Pro^2$ effectively determine a feature-space basis for CIL task? How does it compare to naive methods for feature extraction?

Our method, *orthogonal projection*, outperforms the naive approach of solely utilizing linear probing without fine-tuning the feature extractor, as demonstrated in Table 3.

**Table 3:** Comparison

| Method | Pre-trained | CIFAR100 50 steps 1 | CIFAR100 25 steps 2 | CIFAR100 10 steps 5 | CIFAR100 5 steps 10 |
|---|---|---|---|---|---|
| Naive-random | CLIP-ViT-B/32* | 64.58 | 64.58 | 64.58 | 64.75 |
| Naive-full | CLIP-ViT-B/32* | 69.01 | 69.00 | 69.01 | 69.13 |
| Naive | CLIP-ViT-B/32* | 69.44 | 69.45 | 69.47 | 69.58 |
| Ours | CLIP-ViT-B/32* | **72.97** | **72.95** | **72.73** | **72.72** |

Figure 1 reveals a stark contrast in performance between our method and the naive approach, demonstrating the effectiveness of our approach. In Subfigure 1a, it is evident that the naive method consistently experiences significant performance degradation whenever a new task is introduced. On the other hand, in Subfigure 1b, we observe that $Pro^2$ continuously improves its performance as knowledge accumulates, without forgetting previously learned tasks. Importantly, we achieve promising results even without employing the optimal learning methods such as additional

loss functions or regulatory terms. This highlights the potential and viability of our method in real-world scenarios.

## 6. Discussion and Future work

### 6.1. Growing numbers of otrhonormnal basis

**Algorithm 3** Growing orthonormal basis method

**input:** input training task $\mathcal{D}^b$
**require:** pre-trained network $f : \mathcal{X} \rightarrow \mathbb{R}^d$, orthonormal basis $\pi_1, \cdots, \pi_n$ from previous incremental step, exemplar dataset $\mathcal{D}^{1:b}$

1: **Dummy classifier training**
$w_{dum} \leftarrow \text{argmin}_w \mathcal{L}_{\mathcal{D}^b} \left(w^T f(x), y\right)$
2: **Orthonormal basis training**
$\pi_{n+1}, \cdots, \pi_m \leftarrow \text{argmin}_\pi \mathcal{L}_{\mathcal{D}^b} \left(w_{dum}^T \Pi_m \left(f(x)\right), y\right)$
subject to $\pi_i \perp \pi_j$ for $\forall i, j$
3: **Classifier training**
$w_{cls} \leftarrow \text{argmin}_w \mathcal{L}_{\mathcal{D}^{1:b}} \left(w^T \Pi_m \left(f(x)\right), y\right)$

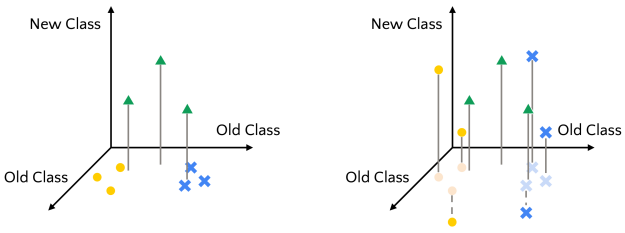
Currently, we train our orthogonal projections using only samples from task1 as a first step to improve performance. This initial finding suggests that the existing orthogonal space, which separates the 50 classes, might be sufficient for establishing decision boundaries in the subsequent tasks. However, it is important to continue our research beyond this point. We should explore the potential of learning enhanced orthogonal projections by incorporating features from the incremental class, as initially intended. For instance, we can investigate the inclusion of additional regularization terms to prevent the dispersion of the existing orthogonal space (Figure 2, left) or explore alternative methods for selecting examples that can improve the representation space. To this end, we proposed Algorithm 3. The key difference between Algorithm 3 and Algorithm 2 is that we conducted both dummy classifier training and orthonormal basis training for every learning task. The increase in feature dimension from class incremental, which can be considered as an increase in information, appeared to facilitate the discrimination of the growing classes.

**Table 4:** Comparison between two orthogonal projection algorithms.

| Method | CIFAR100 | | | |
|---|---|---|---|---|
| | 50 steps<br>1 | 25 steps<br>2 | 10 steps<br>5 | 5 steps<br>10 |
| Algorithm 2 | 72.97 | 72.95 | 72.73 | 72.72 |
| Algorithm 3 | 10.48 | 25.88 | 48.60 | 60.37 |

We evaluated the modified algorithm using the same experimental setup and presented the results in Table 4. Interestingly, the growing orthonormal basis method showed a decline in performance. As shown in the left image of Figure 2, our initial expectation was that the feature space would expand by adding a new axis to capture the unique characteristics of the new class, while the orthographic projection would remove any overlapping information from the previously learned classes' features.

**Figure 2:** Feature Space per Classes




However, we observed that as the basis is expanded, the well-distinguished features in the existing space were also disrupted, leading to poor performance (the right image of Figure 2).

### 6.2. Visualization of projected features

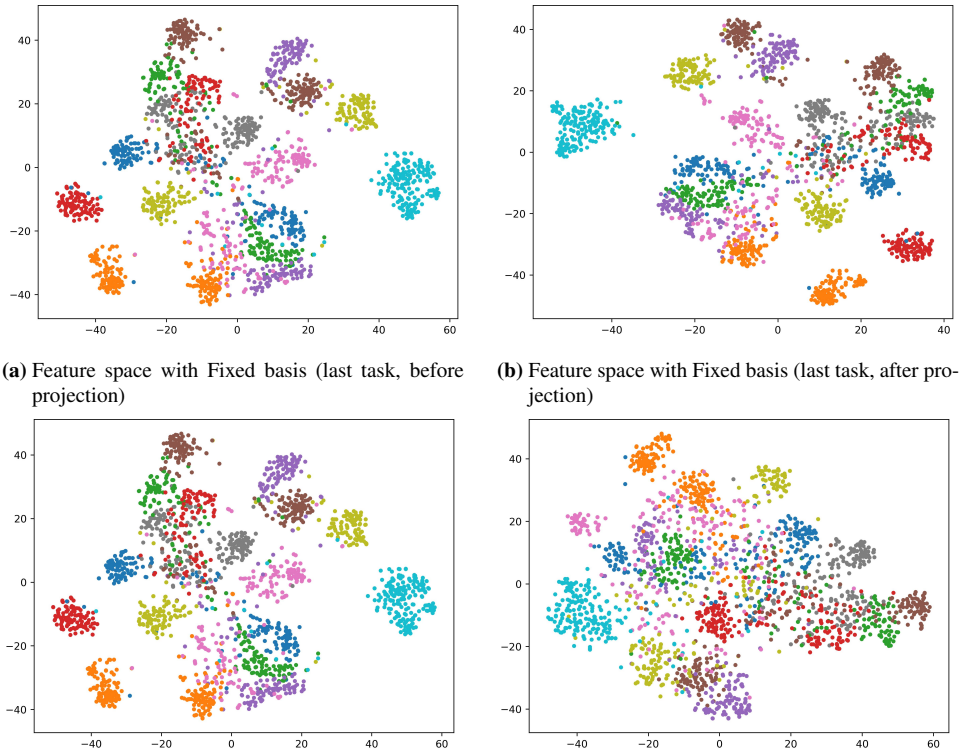
Figure 3 illustrates the learned projection in two scenarios. In the upper figure, the projection is learned solely based on task 1 and remains fixed. On the other hand, the lower figure represents the projection learned by gradually adding basis as the number of classes increases. It can be observed that as the number of basis increases, the process of learning the projection becomes more challenging, projecting the features into a complex space.

Our projection method involved projecting the high-dimensional features obtained from the CLIP model onto a low-dimensional space. While it is expected that there will be some loss of information, the t-SNE results indicate that the sample clusters are not severely disrupted. As a result, we suggest that it may be easier to discriminate them using a simple linear layer classifier. In contrast, the growing method resulted in a deterioration of the sample clusters during the process of moving them to the low-dimensional space, as observed in the t-SNE visualization.

## 7. Conclusion

In this paper, we focus on the use of pre-trained models as feature extractors in Class Incremental Learning (CIL) tasks. Our experiments on the CIFAR100 dataset showed that pre-trained models outperformed randomly initialized models, providing practical implications for real-world scenarios. We also discovered that fine-tuning can have negative effects on pre-trained feature extractors, leading to suboptimal performance.

**Figure 3:** t-SNE: embeddings of randomly selected 20 classes



**(a)** Feature space with Fixed basis (last task, before projection)

**(b)** Feature space with Fixed basis (last task, after projection)

**(c)** Feature space with Growing basis (last task, before projection)

**(d)** Feature space with Growing basis (last task, after projection)

We proposed a novel method inspired by orthogonal projection to tackle these challenges. This method effectively filters out irrelevant information, leading to improved model performance in Class Incremental Learning (CIL) tasks. Our contributions include challenging the assumption of random initialization, introducing a simpler and more computationally efficient approach, and highlighting the limitations of traditional fine-tuning.

In summary, our findings emphasize the importance of effectively utilizing pre-trained models in CIL tasks and provide insights into mitigating catastrophic forgetting. By leveraging pre-trained models as feature extractors and adopting our proposed method, we can achieve superior accuracy and overcome the limitations of fine-tuning, facilitating more efficient and effective continual and incremental learning.

## References


[1] H. Ahn, J. Kwak, S. Lim, H. Bang, H. Kim, and T. Moon. Ss-il: Separated softmax for incremental learning. In Proceedings of the IEEE/CVF International conference on computer vision, pages 844–853, 2021.

[2] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In Proceedings of the European conference on computer vision (ECCV), pages 233–248, 2018.

[3] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In Proceedings of the European conference on computer vision (ECCV), pages 532–547, 2018.

[4] A. S. Chen, Y. Lee, A. Setlur, S. Levine, and C. Finn. Project and probe: Sample-efficient domain adaptation by interpolating orthogonal features, 2023.

[5] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers, 2021.

[6] K. Desai and J. Johnson. Virtex: Learning visual representations from textual annotations, 2021.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[8] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16, pages 86–102. Springer, 2020.

[9] C. He, R. Wang, S. Shan, and X. Chen. Exemplar-supported generative reproduction for class incremental learning. In

BMVC, page 98, 2018.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[11] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Lifelong learning via progressive distillation and retrospection. In Proceedings of the European Conference on Computer Vision (ECCV), pages 437–452, 2018.

[12] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition, pages 831–839, 2019.

[13] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision, 2021.

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

[15] S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better?, 2019.

[16] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[17] A. Kumar, A. Raghunathan, R. M. Jones, T. Ma, and P. Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In International Conference on Learning Representations, 2022.

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

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

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

[21] A. Prabhu, P. H. Torr, and P. K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16, pages 524–540. Springer, 2020.

[22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021.

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

[24] M. B. Sariyildiz, J. Perez, and D. Larlus. Learning visual representations with caption annotations, 2020.

[25] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. Advances in neural information processing systems, 30, 2017.

[26] V. Thengane, S. Khan, M. Hayat, and F. Khan. Clip model is an efficient continual learner, 2022.

[27] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 374–382, 2019.

[28] S. Yan, J. Xie, and X. 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.

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

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

[31] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, L. Beyer, O. Bachem, M. Tschannen, M. Michalski, O. Bousquet, S. Gelly, and N. Houlsby. A large-scale study of representation learning with the visual task adaptation benchmark, 2020.

[32] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia. Maintaining discrimination and fairness in class incremental learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 13208–13217, 2020.