Supplementary Materials for Learning Equi-angular Representations for Online Continual Learning

Minhyuk Seo[1,†] Hyunseo Koh[1] Wonje Jeung[1] Minjae Lee[1] San Kim[1] Hankook Lee[2,3]
Sungjun Cho[2] Sungik Choi[2] Hyunwoo Kim[4,*] Jonghyun Choi[5,*]

[1]Yonsei Univ. [2]LG AI Research [3]Sungkyunkwan Univ. [4]Zhejiang Lab [5]Seoul National Univ.

{dbd0508, specific0924, reccos1020, nasmik419}@yonsei.ac.kr
khs8157@gmail.com, {sungik.choi,sungjun.cho}@lgresearch.ai
hankook.lee@skku.edu, hwkim@zhejianglab.com, jonghyunchoi@snu.ac.kr

Note: Blue characters denote the reference of the main paper.

1. Anytime Inference in Online CL (L164)

In online CL, new data continuously arrive in a stream rather than in a large chunk (*e.g.*, task unit). Several previous works [2][5] train the model only after a large chunk of new data accumulates, which leads to poor inference performance on new data during data accumulation, since the model is not trained during the accumulation period [6, 27].

However, inference queries can occur at any time including the data accumulation, *i.e.*, *anytime inference*, whose importance has been emphasized by recent research [1, 3] [17, 27, 36]. Consequently, we focus not only on A_{last}, which is measured after the learning has finished for all data, but also on A_{AUC}, which measures the average accuracy during training, *i.e.*, 'anytime inference' performance [27].

2. Analysis about Negative Transformation (L332)

Negative Transformation	Gaussian-Scheduled		Disjoint	
	A_{AUC} ↑	A_{last} ↑	A_{AUC} ↑	A_{last} ↑
Patch Permutation	66.37±0.36	69.04±1.39	75.34±0.88	63.73±2.05
Negative Cutmix	65.26±0.22	64.48±1.07	72.86±0.13	56.56±3.25
Gaussian Noise	64.83±0.23	66.97±1.40	74.18±0.22	63.32±1.52
Negative Rotation	**69.52±0.13**	**70.28±1.77**	**77.86±0.71**	**69.50±1.94**

Table 1. Comparison of various negative transformations to obtain preparatory data in CIFAR-10

Negative transformation is a transformation that modifies semantic information of original images and is widely used in self-supervised learning [4, 5] and out-of-distribution (OOD) detection [25, 47, 50]. We consider various negative transformations such as negative rotation (*i.e.*, rotation by 90, 180, and 270 degrees) [15, 18], patch permutation [4], negative CutMix, and Gaussian noise. Negative Cutmix is a kind of Cutmix [11] that finely divides an image into small patches and mixes it with another finely divided image. Gaussian noise refers to filling all the image pixels with a Gaussian random noise. Even though Gaussian noise is not a kind of transformation, it is a straightforward approach to generate synthetic input that can be distinguished from the existing images; thus, we also consider this.

Among them, we choose to use rotations 90, 180, and 270 degrees as the negative transform set T, because the rotation transformation is simple to implement, widely applicable from low- to high-resolution images, and also outperforms other transformations in our empirical evaluations, as we can see in Tab. 1. Unlike rotation transformation, which preserves image continuity while modifying semantic information, patch permutation or Negative Cutmix could cause discontinuities at the boundary of patches [6, 9], leading to loss of image features.

Furthermore, to compare the effect of various negative transformations, we compare the cosine similarities between the features and the corresponding ground truth classifier as shown in Fig. 1. As we can see in the cyan highlighting box, the rotation transformation promotes the convergence of $\hat{f}(x)$ for class 0 towards the ground truth classifier vector $\mathbf{w}_0$.

3. Effect of Negative Rotation Transformation (L304)

We use negative rotation, which rotates images by 90, 180, and 270 degrees [15, 18]. Since the vertical information of the image is more important than the horizontal information [22, 49] as mentioned in Sec.4.2, rotation by a large angle changes the semantic information [50]. To empirically verify that the negative rotation transformation alters the se-

†: Work done while interning at LG AI Research.
*: Indicates corresponding authors.

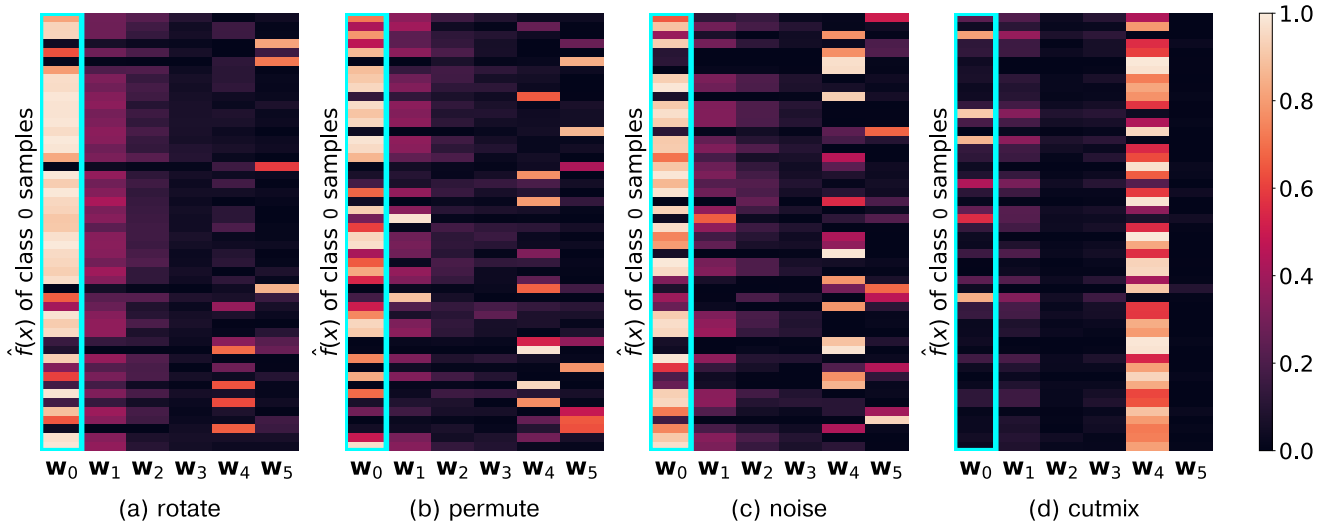


Figure 1. Cosine similarity between the features $\hat{f}(x)$ for class 0 and the ETF classifier vectors $\mathbf{w}_i$ at the 10000^{th} iteration after the introduction of class 0 in the Gaussian Scheduled CIFAR-10 setup.

mantic information of the image, we generate four datasets by rotating the original data by 0, 90, 180, and 270 degrees and train the model using each dataset. Subsequently, we performed inference on all four datasets to check whether they have different semantics.

Specifically, we used CIFAR-10-R90, CIFAR-10-R180, and CIFAR-10-R270, created by applying rotation transformations of 90, 180, and 270 degrees to CIFAR-10 dataset, respectively. We summarize the results in Tab. 2.

Train Dataset	Evaluation Dataset			
	CIFAR-10	CIFAR-10-R90	CIFAR-10-R180	CIFAR-10-R270
CIFAR-10	**92.9**	31.7	34.7	31.8
CIFAR-10-R90	32.1	**92.6**	31.9	35.3
CIFAR-10-R180	33.7	30.9	**92.5**	32.3
CIFAR-10-R270	32.4	34.3	31.4	**92.4**

Table 2. Comparison of the inference accuracy of the rotated datasets after training with each rotated dataset.

4. Prevent Forgetting with ETF Classifier

We compare the performance and forgetting between a fixed ETF classifier and a learnable classifier. We summarize the results in Tab. 3. The fixed ETF classifier not only effectively prevents forgetting, but also has high accuracy.

Classifier	CIFAR-10		CIFAR-100	
	A_{AUC} ↑	Forgetting ↓	A_{AUC} ↑	Forgetting ↓
Learnable	66.02±0.18	6.54±1.02	44.37±0.84	9.04±1.45
Fixed ETF	**69.61±0.35**	**3.75±2.21**	**47.78±0.69**	**7.41±1.02**

Table 3. Comparison of Forgetting and A_{AUC} between a learnable classifier and a classifier with fixed ETF structure on CIFAR-10/100.

5. Comparison of Baselines with Computational Constraint

Following [10][17], we measure the number of FLOPs required for each method to complete one iteration. We then calculate the relative FLOPs with respect to the ER [43], which is the simplest method, and adjust the total training iteration to align the total FLOPs used for the entire training process. With this computational constraint, we compare online CL methods on disjoint and Gaussian scheduled setup for CIFAR-10, CIFAR-100, TinyImageNet and ImageNet-200. Furthermore, we measure the Average Online Accuracy (AOA)[7, 17], which uses the newly encountered streaming data for evaluation before incorporating it into the training process. We plot the online accuracy on TinyImageNet in Fig. 2.

6. Comparison of Baselines on CLEAR-10/100

To evaluate the setups in which the domain of classes changes over time, we compare CL methods with the CLEAR benchmark and summarize the result in Tab. 5. In our experiments, we address a more realistic scenario, called the domain-class-IL setup, where both novel classes and domain shifts take place. This contrasts with domain-IL setups, where all classes are initially provided and only the data distribution changes over time.

EARL significantly outperforms the baselines in CLEAR-10/100 benchmarks and achieves high performance. We perform experiments under the computational constraints mentioned in Sec. 5.

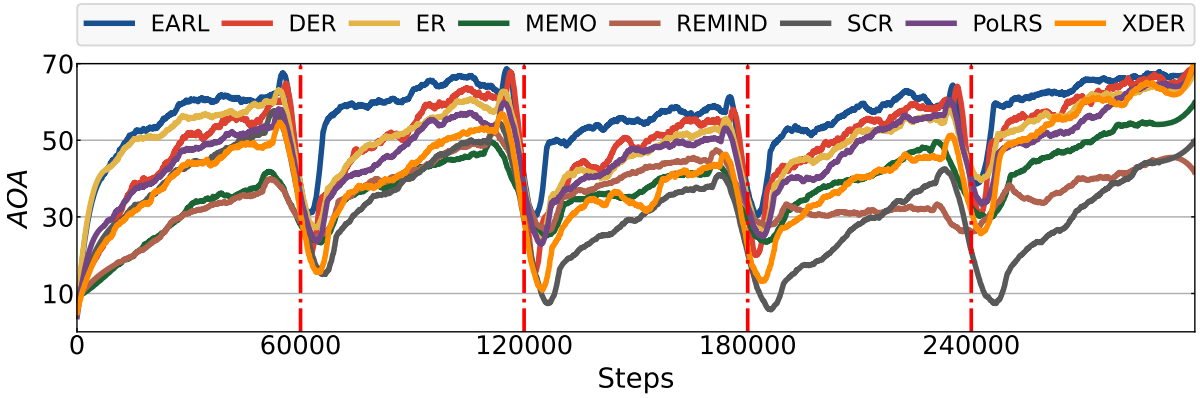


Figure 2. Online accuracy in TinyImageNet Disjoint. Red lines denote task boundary

Method	CIFAR-10						CIFAR-100					
	Disjoint			Gaussian-Scheduled			Disjoint			Gaussian-Scheduled		
	A_{AUC} ↑	A_{last} ↑	AOA ↑	A_{AUC} ↑	A_{last} ↑	AOA ↑	A_{AUC} ↑	A_{last} ↑	AOA ↑	A_{AUC} ↑	A_{last} ↑	AOA ↑
EWC (Kirkpatrick *et al.*, 2017)	77.30±0.91	62.78±1.25	88.70±0.89	60.91±0.30	63.56±3.04	83.22±0.16	52.94±1.00	43.27±0.41	62.04±0.52	40.89±0.40	42.80±0.89	55.29±0.18
ER (Rolnick et al., 2019)	76.89±0.96	63.92±2.36	88.47±0.78	60.71±0.10	66.71±2.49	82.47±0.17	53.13±1.28	41.97±0.21	62.06±0.73	41.30±0.22	44.10±0.14	55.28±0.33
ER-MIR (Aljundi *et al.*, 2019)	75.08±0.10	63.13±3.32	87.27±1.52	57.25±0.71	59.34±2.12	79.29±0.69	50.17±0.91	42.17±0.50	57.74±0.31	35.52±0.57	41.29±0.71	46.44±0.55
REMIND (Hayes *et al.*, 2020)	67.73±0.42	49.33±1.30	81.30±1.17	55.94±1.06	51.73±4.11	77.41±1.27	40.76±0.46	37.64±1.13	48.18±0.98	22.55±1.58	30.27±1.94	33.02±1.19
DER++ (Buzzega *et al.*, 2020)	75.71±1.46	58.49±3.16	**88.93±1.00**	59.14±1.15	66.16±3.83	82.14±0.76	41.39±0.80	42.03±0.81	57.00±0.30	28.63±1.93	37.10±2.23	47.39±1.38
SCR (Mai *et al.*, 2021)	75.89±0.47	57.90±2.45	87.91±1.82	60.38±0.25	63.02±3.71	78.73±0.48	36.74±1.72	30.54±0.99	34.56±1.26	26.53±1.00	27.59±0.96	29.88±0.93
ODDL (Ye *et al.*, 2022)	75.91±0.87	61.89±4.47	88.61±1.50	61.02±0.60	65.25±1.01	83.58±0.69	53.16±1.18	42.89±0.64	61.19±0.78	42.46±0.30	44.26±1.26	56.33±0.36
MEMO (Zhou *et al.*, 2023)	72.59±0.18	63.29±5.07	86.33±1.81	57.88±1.30	61.67±2.32	79.04±0.82	39.48±0.73	37.53±0.64	50.06±0.13	22.46±1.40	33.27±2.39	36.02±1.21
X-DER (Boschini *et al.*, 2023)	75.99±1.22	65.69±4.05	83.28±2.05	57.73±0.76	66.59±2.45	76.63±0.32	47.22±1.21	43.72±0.56	50.73±0.78	36.81±0.56	46.20±0.53	47.59±0.33
EARL (Ours)	**78.54±0.48**	**66.16±0.84**	88.10±0.58	**69.77±0.05**	**71.46±1.84**	**86.00±0.17**	**57.65±1.30**	**45.15±0.05**	**67.09±0.69**	**48.05±0.67**	**47.27±0.69**	**63.69±0.76**

Method	TinyImageNet						ImageNet-200					
	Disjoint			Gaussian-Scheduled			Disjoint			Gaussian-Scheduled		
	A_{AUC} ↑	A_{last} ↑	AOA ↑	A_{AUC} ↑	A_{last} ↑	AOA ↑	A_{AUC} ↑	A_{last} ↑	AOA ↑	A_{AUC} ↑	A_{last} ↑	AOA ↑
EWC (Kirkpatrick *et al.*, 2017)	37.57±0.83	27.43±0.89	49.58±0.42	26.35±0.84	25.61±0.23	37.77±0.25	41.85±0.64	31.57±0.76	64.10±0.26	32.22±0.27	32.71±0.94	54.68±0.31
ER (Rolnick et al., 2019)	37.29±0.81	27.04±0.40	49.66±0.39	26.37±0.90	25.92±0.38	37.66±0.50	41.65±0.62	32.18±0.22	64.22±0.23	32.43±0.51	32.85±0.27	54.72±0.50
ER-MIR (Aljundi *et al.*, 2019)	37.73±0.84	27.40±0.22	48.05±0.15	24.00±0.73	25.10±0.41	32.34±0.46	39.88±0.44	33.27±0.66	59.87±0.14	28.65±0.26	33.30±1.84	48.36±0.44
REMIND (Hayes *et al.*, 2020)	29.05±0.83	27.30±0.59	36.43±0.83	10.22±0.93	16.59±0.70	16.85±1.17	38.32±0.73	31.60±0.46	39.80±1.85	30.46±0.37	33.36±0.82	38.13±1.17
DER++ (Buzzega *et al.*, 2020)	39.24±0.80	29.77±1.16	49.30±0.81	27.27±2.13	31.40±0.89	40.19±1.47	44.67±0.17	32.65±0.54	65.74±0.21	37.11±0.28	37.85±0.74	59.81±0.33
SCR (Mai *et al.*, 2021)	34.07±1.16	24.24±0.43	33.28±0.93	25.62±0.90	24.28±0.24	29.36±0.41	41.94±0.31	28.53±0.37	61.97±0.40	33.27±0.46	31.41±0.31	54.12±0.48
MEMO (Zhou *et al.*, 2023)	27.84±0.38	27.52±0.52	37.35±0.26	9.45±0.60	17.30±0.65	17.50±0.49	38.68±0.39	31.70±0.53	58.92±0.53	32.09±0.26	35.95±1.40	50.88±0.10
X-DER (Boschini *et al.*, 2023)	35.68±0.38	27.15±1.66	41.36±1.41	23.51±2.43	24.80±4.06	32.17±3.71	44.03±0.37	33.58±1.17	56.40±0.08	34.92±0.24	38.29±1.12	53.75±0.23
EARL (Ours)	**42.19±1.18**	**29.50±0.78**	**56.41±0.08**	**34.79±0.67**	**31.68±0.21**	**49.77±0.13**	**45.01±0.36**	**34.25±0.69**	**67.08±0.34**	**39.04±0.22**	**38.95±0.50**	**60.80±0.13**

Table 4. Comparison of online CL methods on Disjoint and Gaussian Scheduled Setup for CIFAR-10, CIFAR-100, TinyImageNet and ImageNet-200 with computational constraint.

Method	CLEAR10			CLEAR100		
	A_{AUC} ↑	A_{last} ↑	AOA ↑	A_{AUC} ↑	A_{last} ↑	AOA ↑
EWC	70.88±1.15	69.46±2.40	76.96±1.16	45.74±0.40	47.61±0.54	46.15±0.28
ER	70.70±1.22	68.86±3.00	76.65±1.12	45.59±0.91	47.89±1.11	46.05±0.18
ER-MIR	68.21±0.94	65.58±2.33	74.53±1.10	43.21±1.03	46.60±1.21	42.24±0.47
REMIND	66.48±1.93	66.91±1.19	72.34±1.78	36.67±0.76	47.90±0.58	35.49±0.09
DER++	71.93±0.90	70.41±2.67	77.51±1.47	47.34±0.63	49.63±0.75	47.02±0.33
SCR++	73.32±0.85	70.81±1.69	77.90±1.11	44.67±0.77	44.70±0.72	46.15±0.28
MEMO	65.04±1.72	62.64±2.67	71.94±1.80	44.35±0.54	46.47±1.58	41.24±0.12
PoLRS	65.65±1.88	61.06±6.17	71.67±1.53	41.17±1.83	41.62±2.56	40.15±1.50
X-DER	69.77±0.85	68.67±3.04	74.76±1.23	44.76±1.21	49.01±1.31	43.60±0.65
EARL	**77.85±0.96**	**76.51±1.97**	**81.51±1.17**	**56.97±0.25**	**59.03±1.09**	**55.35±0.06**

Table 5. Comparison of online CL methods on CLEAR-10/100 with computational constraint.

7. Comparison of Computational Budget (L)

EARL requires an additional computation for residual correction, which calculates k nearest features, and FLOPs of additional cost can be formulated as $d\times N\times 3+k\times N$, where d is the dimension of stored features, N is the total number of feature-residual pairs, and k denotes top_k in k-NN. The first term is for calculating the distances between the stored features and the feature of the inference image and is multiplied by 3 since it involves subtraction, squaring, and addition operations. The second term is the cost of selecting top_k features among N features. Compared to the cost

incurred by the naive inference process, which requires forward flops of the model, it involves a very minimal amount of additional cost. Taking the example of ImageNet-200 with the ResNet-18 architecture that has $\frac{1}{3}$ GFLOPs in the model forward, only 0.5% of additional cost is consumed, since we use $N = 2,000 (= 10 \times 200)$ and $k = 15$.

8. Details About Experiment Setup (L404)

This paper focuses on online class-incremental learning in two types of setups: disjoint [35] and Gaussian scheduled [8] [45, 51]. In the disjoint setup, each class is assigned to a specific task, *i.e.*, tasks do not share any classes. On the other hand, in the Gaussian scheduled setup, the arrival time of each class follows a Gaussian distribution $\mathcal{N}(\mu_i, \sigma)$. Since the class distribution is shifted every time step, the Gaussian scheduled setup is a boundary-free setup. We set σ to 0.1 and μ_i, mean of the class i, to $\frac{i}{N}$, where N is the number of classes.

9. Effect of k

k, which is a hyperparameter that determines the number of the nearest features from the inference image used to weight-sum to calculate the residual, has a negligible impact, except for $k = 1$ as incorrect residual largely affects predictions. We illustrate the consistency of accuracy at various values of k in Fig. 3.

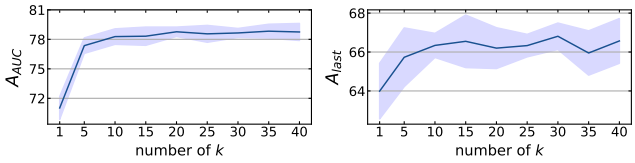


Figure 3. Performance variations of A_{AUC} and A_{last} with respect to the change in k in the CIFAR-10 disjoint setup.

10. Hyperparameters (L428)

For all methods, we use Adam optimizer [7] with a constant learning rate (LR) of 0.0003. For data augmentation, we use RandAugment [2]. For hyperparameters such as iteration, memory size, and number of tasks for each dataset, we follow prior works [2, 27, 40]. Specifically, we set the number of iterations as 1, 3, 3, 0.25, and 0.25, and memory sizes as 500, 2,000, 4,000, 4,000, and 10000 for the CIFAR-10, CIFAR-100, TinyImageNet, ImageNet-200, and ImageNet-1k datasets, respectively. To ensure a fair comparison among methods, we use the same batch size in all methods. Specifically, we use 16, 32, 32, 256, and 256 for CIFAR-10, CIFAR-100, TinyImageNet, ImageNet-200, and ImageNet-1k datasets, respectively. Note that the number of preparatory data is included in the batch size for EARL, *i.e.*, batch size = the number of retrieved samples from memory + the number of preparatory data.

To ensure that EARL does not depend on a specific dataset, we use the following hyperparameters for all datasets: $\tau = 0.9, k = 15, d = 4096$ and $\lambda = 1$, where τ refers to the temperature parameter during residual correction, k refers to the number of samples for top_k in k-NN during residual correction, d refers to the output dimension of the projection layer, and λ refers to the loss balancing parameter between real data training and preparatory data training. In addition, for N, the total number of stored feature-residual pairs in EARL, we used $N = 10 * |C|$ for all dataset, where $|C|$ is the number of classes seen so far.

11. Implementation of Baselines (L587)

Some of the baselines assumed multi-epoch training, *i.e.*, offline CL, so we modified them to be used in online CL for comparisons with our proposed method and other baselines.

REMIND. Remind is a feature-replay method and freezes front layers after base initialization, offline training phase using a subset of the data during the initial stage of training. We modified the base initialization of REMIND [20] to be suitable for use in online CL. In offline CL setup, REMIND freezes 7 layers after base initialization. However, as shown in Table 6, in online CL, it is not suitable to freeze many layers compared to offline CL, as the model is not sufficiently trained. We studied various numbers of frozen layers and revealed that freezing 6 layers (*i.e.*, 3 blocks) is the optimal number for freezing in ResNet-18 in CIFAR-10 and CIFAR-100. Compared to freezing 7 layers, which is the original proposed freezing criterion, REMIND performed comparatively well when freezing 6 layers in the online CL, despite the decrease in the number of stored features due to the larger sizes of stored features. Regarding the hyperparameters used in REMIND, such as the size of the codebook and the number of codebooks for product quantization, we performed additional hyperparameter search experiments with CIFAR-100, as shown in Table 7, and used the same hyperparameter in all remaining datasets.

For CIFAR-10, CIFAR-100, and TinyImageNet, which have a relatively small number of images, we increased the proportion of the initialization sample base from 10% to 60% of the total samples, since 10% images are not enough for the lower level layers to represent highly transferable features, leading to low performance. In ImageNet-200 and ImageNet-1k, we followed the original setting (*i.e.*, 10% base initialization samples and freezing 7 layers after base initialization), following [20].

MEMO. MEMO [63] retrieves samples that are relatively close to the class mean feature at every task boundary and

Frozen Layers	CIFAR-10				CIFAR-100			
	Disjoint		Gaussian-Scheduled		Disjoint		Gaussian-Scheduled	
	A_{AUC} ↑	A_{last} ↑	A_{AUC} ↑	A_{last} ↑	A_{AUC} ↑	A_{last} ↑	A_{AUC} ↑	A_{last} ↑
4 Layers	66.63±1.24	46.24±0.12	54.55±0.71	45.56±1.54	38.73±0.27	30.19±0.66	23.76±0.86	26.94±0.37
5 Layers	**70.11±0.66**	51.01±0.79	56.47±0.96	52.00±1.94	41.87±0.05	**38.81±0.41**	24.79±1.73	32.12±2.81
6 Layers	69.55±0.91	47.28±3.92	**57.15±0.71**	**53.40±0.70**	**41.92±0.06**	37.64±1.09	**25.56±1.10**	**34.08±1.02**
7 Layers	68.59±0.10	**53.71±2.27**	55.37±0.60	52.53±1.98	40.52±0.17	37.42±0.81	23.94±1.30	33.28±1.43

Table 6. REMIND performance as a function of the number of frozen layers in ResNet-18 with CIFAR-10 and CIFAR-100. Rather than freezing all the layers except the last layer (*i.e.*, freezing 7 Layers), continuously updating the last two layers and fixing the rest (*i.e.*, freezing 6 Layers) shows the best performance due to the limitation of online CL.

♯ of Codebooks	CIFAR-100			
	Disjoint		Gaussian-Scheduled	
	A_{AUC} ↑	A_{last} ↑	A_{AUC} ↑	A_{last} ↑
8	41.07±0.27	36.97±0.27	25.17±1.02	**34.14±1.18**
16	41.84±0.29	36.48±0.34	25.37±0.99	33.30±0.26
32	**41.92±0.06**	**37.64±1.09**	**25.56±1.10**	34.08±1.02
64	40.13±0.15	33.50±0.46	24.39±0.85	29.48±0.04

Codebook Size	CIFAR-100			
	Disjoint		Gaussian-Scheduled	
	A_{AUC} ↑	A_{last} ↑	A_{AUC} ↑	A_{last} ↑
256	**41.92±0.06**	**37.64±1.09**	25.56±1.10	**34.08±1.02**
512	41.48±0.19	36.98±0.51	**25.85±1.39**	33.01±1.99
1024	41.28±0.18	36.48±0.34	25.13±0.90	31.83±0.19
2048	41.11±0.21	36.10±0.33	25.05±0.90	31.33±0.11

Table 7. REMIND performance as a function of different codebook sizes and number of codebooks with CIFAR-100. Original hyperparameters (codebook size: 256, number of codebooks: 32) consistently show the best performance in the online CL setting. The same hyperparameters were used uniformly for all datasets.

stores them in episodic memory for replay. However, in online CL, the model can not access all data for the current task, *i.e.*, it continuously updates the memory using stream data from the current task. Therefore, the sampling strategy of MEMO is replaced by class-balanced random sampling, which is used to replace the sampling strategy of RM [2] for online CL in [27].

12. Comparison between NC-FSCIL and Vanilla ETF

NC-FSCIL [56] is a recently proposed offline CL method that attempts to induce neural collapse in few-shot class incremental learning (FSCIL), a CL setup with few training samples per class. NC-FSCIL uses the fixed ETF classifier, a backbone network f, and a projection layer. It freezes the backbone network after training the base task and further fine-tunes the projection layer in the following incremental tasks by using the replay memory $\mathcal{M}^{(t)}$, which stores the class-wise mean features $\mathbf{h}_c$ retrieved from the backbone network for each old class c as follows:

$$\mathcal{M}^{(t)} = \left\{ \mathbf{h}_c | c \in \bigcup_{j=0}^{t-1} C^{(j)} \right\}, \quad 1 \leq t \leq T, \qquad (1)$$

where $\mathbf{h}_c = \text{Avg}_i \left\{ f\left(x_i, \theta\right) | y_i = c \right\}$, T refers to the total number of tasks, and $C^{(j)}$ refers to the set of classes for task j.

Considering that our setup is neither an offline CL nor a few-shot class incremental learning setup, for a fair comparison with online CL methods, we modify the freezing strategy of NC-FSCIL. Instead of freezing the whole backbone, we freeze only 6 layers, which achieved the best performance in online CL as shown in Table 6.

Despite the high performance of NC-FSCIL in offline CL, it has a lower performance than that of vanilla ETF (*i.e.*, baseline of EARL without preparatory data training and residual correction) in online CL, as shown in Tab. 8.

13. Pseudocode for the Our Method (L428)

Algorithm 1 and Algorithm 2 provide detailed pseudocode for EARL.

14. Detailed analysis of ablation results (L509)

Note that exclusively relying on residual correction without the training of preparatory data may result in the addition of incorrect residuals due to bias problem, as illustrated in Fig. 4-(a). The bias in CL causes the features of novel classes and old classes to overlap. When the features of multiple classes are clustered together, as in Fig.4-(a), residuals of one class can be added to the residual-correcting term of other classes in the cluster, since we select the residuals using k nearest-neighbors of corresponding features. Thus, the residuals of old classes are often added to novel class samples and vice versa, hurting the accuracy of both old and novel classes.

In this context, the use of preparatory data not only accelerates the convergence of ETF during training, but also promotes accurate residual addition during inference. In conclusion, the combination of residual correction and prepara-

Methods	CIFAR-10				CIFAR-100			
	Disjoint		Gaussian-Scheduled		Disjoint		Gaussian-Scheduled	
	A_{AUC} ↑	A_{last} ↑	A_{AUC} ↑	A_{last} ↑	A_{AUC} ↑	A_{last} ↑	A_{AUC} ↑	A_{last} ↑
NC-FSCIL	68.09±0.69	48.05±1.69	53.12±0.77	46.11±0.36	39.54±0.79	34.83±1.27	30.91±1.46	35.73±1.13
Vanilla ETF	**75.27±0.77**	**62.10±4.12**	**65.80±0.25**	**67.79±0.78**	**52.91±1.05**	**41.08±0.60**	**33.34±0.55**	**44.45±0.48**

Table 8. Comparison between NC-FSCIL and Vanilla ETF (EARL w/o Preparatory data training and Residual correction) on CIFAR-10 and CIFAR-100.

Algorithm 1 Training Phase

Input model f_θ, Memory $\mathcal{M}$, Residual Memory $\mathcal{M}_{\text{RES}}$, Training data stream $\mathcal{D}$, ETF classifier $\mathbf{W}$, Negative transformation $\mathcal{R}_r$, Learning rate μ
for $(x, y) \in \mathcal{D}$ **do** ▷ Sample arrives from training data stream D
  Update $\mathcal{M} \leftarrow \text{ClassBalancedSampler}(\mathcal{M}, (x, y))$ ▷ Update memory
  Sample $(X, Y) \leftarrow \text{RandomRetrieval}(\mathcal{M})$ ▷ Get batch (X, Y) from memory
  Sample $(X', Y') \leftarrow \text{RandomRetrieval}(\mathcal{M})$ ▷ Get batch (X', Y') to make preparatory data
  $(X_{\text{p}}, Y_{\text{p}}) \leftarrow \mathcal{R}_r(X', Y')$ ▷ Negative transformation for preparatory data training
  $\hat{f}_\theta(X) = \frac{f_\theta(X)}{|f_\theta(X)|}, \hat{f}_\theta(X_p) = \frac{f_\theta(X_p)}{|f_\theta(X_p)|}$ ▷ Normalize model output
  $\mathbf{r} = \mathbf{W}_Y - \hat{f}_\theta(X)$ ▷ Calculate Residuals
  Update $\mathcal{M}_{\text{RES}} \leftarrow \left(\hat{f}_\theta(X), \mathbf{r}\right)$ ▷ Update feature-residual memory
  $\mathcal{L}(X, Y, X_p, Y_p; \theta, \mathbf{W}) = L_{DR}(\hat{f}_\theta(X), \mathbf{W}_Y) + L_{DR}(\hat{f}_\theta(X_p), \mathbf{W}_{Y_p})$ ▷ Calculate dot-regression loss
  Update $\theta \leftarrow \theta - \mu \cdot \nabla_\theta \mathcal{L}(X, Y, X_p, Y_p; \theta, \mathbf{W})$ ▷ Update model
end for
Output f_θ

Algorithm 2 Inference Phase

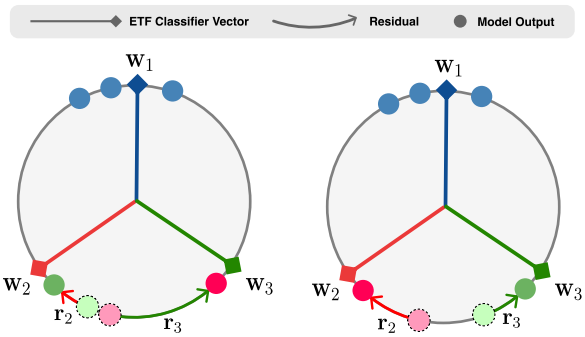
Input model f_θ, inference input x_{eval}, Residual Memory $\mathcal{M}_{\text{RES}}$, ETF classifier $\mathbf{W}$, number of nearest neighbors k, softmax temperature τ
$\{(\hat{h}_i, \mathbf{r}_i)\}_{i=1}^{|\mathcal{M}_{\text{RES}}|} \leftarrow \mathcal{M}_{\text{RES}}$ ▷ Get residual and features from residual memory
$\hat{f}_\theta(x_{\text{eval}}) = \frac{f_\theta(x_{\text{eval}})}{|f_\theta(x_{\text{eval}})|}$ ▷ Normalize model output
$\{n_1, n_2, ..., n_k\} \leftarrow k\text{-}\arg\min_i \left(\left|\hat{f}_\theta(x_{\text{eval}}) - \hat{h}_i\right|\right)$ ▷ Calculate k nearest neighbor features
$s_{n_i} = \frac{e^{-(\hat{f}(x_{\text{eval}}) - \hat{h}_{n_i})/\tau}}{\sum_{j=1}^{k} e^{-(\hat{f}(x_{\text{eval}}) - \hat{h}_{n_j})/\tau}}$ ▷ calculate residual weights
$\mathbf{r} = \sum_{i=1}^{k} s_{n_i} \mathbf{r}_{n_i}$ ▷ Calculate residual-correcting term
$\hat{f}_\theta(x_{\text{eval}})_{\text{corrected}} \leftarrow \hat{f}_\theta(x_{\text{eval}}) + \mathbf{r}$ ▷ Add residual on features
$y_{\text{pred}} = \arg\max_y(\text{CosineSimilarity}(\mathbf{W}_y, \hat{f}_\theta(x_{\text{eval}})_{\text{corrected}}))$ ▷ Predict class
Output y_{pred}

tory data training effectively aligns the model output with the corresponding ETF classifier, as demonstrated in Fig. 4-(b).

15. Properties of Neural Collapse

(NC1) Collapse of Variability: The last layer feature output of each data point collapses toward the class mean feature of its respective class. In other words, $h_{k,i}$, last layer feature of sample i in class k, collapse to $\mu_k = \sum_{i=1}^{n_k} h_{k,i}$ for $\forall k \in [1, K]$ where n_k is the number of samples for class k. By considering within-class covariance and between-class covariance

$$\Sigma_W = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{n_k}(\sum_{i=1}^{n_k}(h_{k,i} - \mu_k)),$$
$$\Sigma_B = \frac{1}{K}\sum_{k=1}^{K}(\mu_k - \mu_G), \tag{2}$$



(a) wrong residual added (b) correct residual added

Figure 4. Without preparatory data training and relying solely on residual correction, incorrect residuals can be added due to bias, which can potentially lead to decreased performance. On the other hand, the combination of preparatory data training and joint training leads to the addition of correct residuals by addressing the bias problem.

where $\mu_G = \sum_{k=1}^{K} \mu_k$, empirical variability can be measured as

$$NC1 := \frac{1}{K} trace(\Sigma_W \Sigma_B^{\dagger}). \tag{3}$$

(NC2) Convergence to simplex equiangular tight frame (ETF): Class means $\mu_k (k \in [1, K])$ centered by the global mean μ_G converge to vertices of a simplex ETF structure, i.e., matrix $M = [m_1\ m_2\ \cdots\ m_K]$ where $m_k = \frac{\mu_k - \mu_G}{\|\mu_k - \mu_G\|^2}$ satisfies the following equation:

$$\mathbf{M}\mathbf{M}^T = \frac{1}{K-1}(KI_K - \mathbf{1}_K \mathbf{1}_K^T). \tag{4}$$

The degree of convergence can be measured using:

$$\frac{\mathbf{M}\mathbf{M}^T}{\|\mathbf{M}\mathbf{M}^T\|_F} - \frac{1}{\sqrt{K-1}}(I_K - \frac{1}{K}\mathbf{1}_K \mathbf{1}_K^T). \tag{5}$$

(NC3) Convergence to self-duality: Classifier $\mathbf{W}$ converges to the simplex ETF $\mathbf{M}$ formed by recentered feature mean, and during this convergence, the classifier vector w_k aligns with their corresponding feature mean m_k where w_k means classifier weight for class k, $k \in [1, K]$, *i.e.*,

$$\frac{\mathbf{M}}{\|\mathbf{M}\|_F} = \frac{\mathbf{W}}{\|\mathbf{W}\|_F} \tag{6}$$

Duality can be measured by measuring:

$$\frac{\mathbf{W}\mathbf{M}^T}{\|\mathbf{W}\mathbf{M}^T\|_F} - \frac{1}{\sqrt{K-1}}(I_K - \frac{1}{K}\mathbf{1}_K \mathbf{1}_K^T). \tag{7}$$

References

[1] Soumya Banerjee, Vinay K Verma, Avideep Mukherjee, Deepak Gupta, Vinay P Namboodiri, and Piyush Rai. Verse: Virtual-gradient aware streaming lifelong learning with anytime inference. *arXiv preprint arXiv:2309.08227*, 2023. 1

[2] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 4

[3] Thang Doan, Seyed Iman Mirzadeh, and Mehrdad Farajtabar. Continual learning beyond a single model. *arXiv preprint arXiv:2202.09826*, 2022. 1

[4] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015. 1

[5] Chris Dongjoo Kim, Jinseo Jeong, Sangwoo Moon, and Gunhee Kim. Continual learning on noisy data streams via self-purified replay. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 537–547, 2021. 1

[6] Jun Kimata, Tomoya Nitta, and Toru Tamaki. Objectmix: Data augmentation by copy-pasting objects in videos for action recognition. In *Proceedings of the 4th ACM International Conference on Multimedia in Asia*, pages 1–7, 2022. 1

[7] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, San Diego, CA, USA, 2015. 4

[8] Hyunseo Koh, Minhyuk Seo, Jihwan Bang, Hwanjun Song, Deokki Hong, Seulki Park, Jung-Woo Ha, and Jonghyun Choi. Online boundary-free continual learning by scheduled data prior. In *The Eleventh International Conference on Learning Representations*, 2023. 4

[9] Sanghyeok Lee, Minkyu Jeon, Injae Kim, Yunyang Xiong, and Hyunwoo J Kim. Sagemix: Saliency-guided mixup for point clouds. *Advances in Neural Information Processing Systems*, 35:23580–23592, 2022. 1

[10] Minhyuk Seo, Hyunseo Koh, and Jonghyun Choi. Budgeted online continual learning by adaptive layer freezing and frequency-based sampling. 2023. 2

[11] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 1