Adversarial Pseudo-replay for Exemplar-free Class-incremental Learning

Supplementary Material

1. Implementation Details

Table 9 shows the detailed parameters used in the benchmarks (Main Paper Table 1 and 2). The rows above the horizontal line are the parameters that are also used for the existing methods, and the rest are the ones for our proposed method. Gradient accumulation is used for ImageNet-Subset due to GPU memory constraints and the effective batch size is displayed. At the initial task stage, the model is trained with SGD as an optimizer with learning rate of 0.1 and weight decay of 5e-4 for 200 epochs. At the following incremental stages the model is trained with learning rate of 0.01 and weight decay of 2e-4 for 100 epochs (60 epochs for ImageNet-Subset). A cosine decay schedule is adopted and batch size is set to 64. For warm-start settings, the learning rate in the incremental stages is set to 0.001. All experiments are repeated three times and averaged, varying both the class-shuffling seed (1993 [5], 2993, 3993) and the randomness seed (0, 1000, 2000).

To avoid over-fitting to the test data, the shrinkage parameters γ_1 and γ_2 in Main Paper Eq. 9 are determined by validation protocol, splitting the validation dataset (N = 50 per class) from the training dataset for each setting (including ablation study). The candidates for γ_1 and γ_2 are discrete and the search space is (1, 3, 8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96, 104, 112, 120). Table 10 shows the tuned γ_1 and γ_2 for FeCAM, AdaGauss and our APR.

For AdaGauss [4], we re-formulated the anti-collapse loss. The original loss L_{AC} involves Cholesky decomposition of covariance matrix calculated from training mini-batch. Cholesky decomposition fails when the target covariance is not positive-definite, especially at the early training stage. We alternatively adopt eigenvalues of covariance as loss target instead of diagonal elements of Cholesky decomposed matrix. Our implementation is numerically stable and yields results empirically comparative to the original L_{AC}. The output feature dimension is set to 64 following the original setting.

2. Data Augmentation Parameters

The list of augmentation parameter is shown in Table 8. The random parameter values used in candidate selection are recorded and applied during new task training in a deterministic manner. During the candidate selection, AutoAugment [1] policies for CIFAR10 and ImageNet are applied for CIFAR100 and ImageNet-Subset respectively, following random crop and horizontal flipping. In AutoAugment, a policy is randomly selected from 25 policies, each of which consists of two successive transforms. The magnitude and occurrence probability of each transform are fixed as the default values of AutoAugment. The augmentation pipelines are as follows:

- CIFAR100: RandomCrop, RandomHolizontalFlip, ColorJitter, CIFAR10Policy.
- TinyImageNet: RandomCrop, RandomHolizontalFlip.
- ImageNet-Subset: RandomResizedCrop, RandomHolizontalFlip, ImageNetPolicy.

Augmentation	Parameter
RandomCrop	x, y, width, height
RandomResizedCrop	x, y, width, height
RandomHolizontalFlip	do or not
ColorJitter	function indices
ColorJitter	magnitude values
CIFAR10Policy	policy index
ImageNetPolicy	policy index
Autoaug policy 1	do or not
Autoaug policy 2	do or not
Autoaug ShearX	magnitude value
Autoaug ShearY	magnitude value
Autoaug TranslateX	magnitude value
Autoaug TranslateY	magnitude value
Autoaug Rotate	function index value
Autoaug Color	magnitude
Autoaug Contrast	magnitude value
Autoaug Sharpness	magnitude value
Autoaug Brightness	magnitude value

Table 8. Augmentation random parameter list. The data type for "do or not" is `bool` and `int64` otherwise.

3. Candidate Selection

In candidate sampling before each incremental task, k data indices and augmentation policy parameters are sampled among those with the smallest feature-prototype distances. For example, 200 image indices are sampled from 5,000 new-task images for each class. Some images are assigned to multiple prototypes, especially at later stages. In this section we limit the number of assignments per new-task image and compare it with the main result setting where no limit is applied.

To achieve this, we first calculate an (N, M) distance matrix d_{mn}, where N is the number of new-task images and M is the number of classes (prototypes). We sort all the values of d_{mn} and iteratively assign the image-prototype pair from the smallest distance. If the number of assigned

Parameter	CIFAR100 cold				CIFAR100 warm				TinyImageNet cold				TinyImageNet warm				ImageNetSubset cold
Number of tasks (T)	5	10	6	11	5	10	6	11	5	10	6	11	5	10	6	11	10
Number of initial classes	20	10	50	50	40	20	100	100	20	10	50	50	40	20	100	100	10
Number of incremental classes	20	10	10	5	40	20	20	10	20	10	10	5	40	20	20	10	10
1st conv of extractor	stride=2				stride=2				stride=2				stride=2				stride=1
Gradient accumulation	1				1				1				1				2
Number of epochs $T > 1$	100				100				100				100				60
KD loss weight λ_{kd}	10				10				10				10				10
CE loss temperature	1				1				1				1				1
Learning rate ($t = 0$)	0.1				0.1				0.1				0.1				0.1
Learning rate ($t > 0$)	0.01		0.001		0.01		0.001		0.01		0.001		0.01		0.001		0.01
Weight decay ($t = 0$)	5e-4				5e-4				5e-4				5e-4				5e-4
Weight decay ($t > 0$)	2e-4				2e-4				2e-4				2e-4				2e-4
Batchsize B_t ($t = 0$)	64				64				64				64				64
Batchsize B_t ($t > 0$)	32				64				64				64				64
Batchsize B_{APR} ($t > 0$)	64				64				64				64				32
APR magnitude α	64				64				64				64				64
APR iterations	4				6				2				2				2
APR number of candidates	200				200				200				200				200
ADC magnitude	6.32				6.32				3.16				3.16				3.16
ADC iterations	9				6				3				3				3
ADC batchsize	64				64				16				16				16
ADC number of candidates	1000				1000				1000				1000				1000
Transfer W learning rate	1e-4				1e-4				1e-4				1e-4				1e-4
Transfer W epochs	64				64				64				64				64

Table 9. Hyperparameter settings. The rows below the horizontal line correspond to the settings specific to our method.

Parameter	CIFAR100 cold				CIFAR100 warm				TinyImageNet cold				TinyImageNet warm				ImageNetSubset cold
Number of tasks (T)	5	10	6	11	5	10	6	11	5	10	6	11	5	10	6	11	10
Number of initial classes	20	10	50	50	40	20	100	100	20	10	50	50	40	20	100	100	10
Number of incremental classes	20	10	10	5	40	20	20	10	20	10	10	5	40	20	20	10	10
FeCAM [2]	3	1	3	3	3	1	3	3	3	1	3	3	3	1	3	3	1
AdaGauss [4]	3	3	1	3	8	8	3	8	3	3	1	3	8	8	3	8	8
APR (ours)	16	24	40	32	64	40	72	72	16	24	40	32	64	40	72	72	40

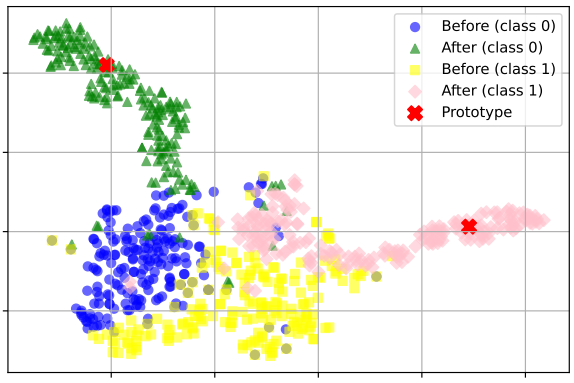
Table 10. Covariance shrinkage parameters γ_1 ($= \gamma_2$) tuned by the validation protocol, where the validation dataset (N=50 per class) is split from the training dataset.

classes for the image reaches N_{cap} (e.g. 4) or the number of assigned images for the prototype reaches k (e.g. 200), the distance value is skipped. The assignment is done when the numbers of assigned images for all the prototypes reach k. The no-limit setting is considered as the special case of the above assignment where $N_{cap} = \infty$.

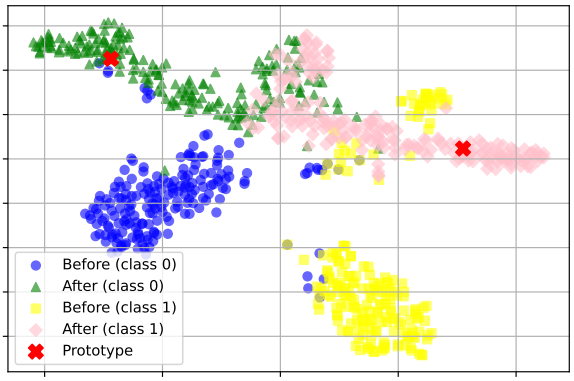
Table 11 shows the comparison between $N_{cap} = 4$ and ∞. We choose $N_{cap} = 4$ because it is the minimum requirement for CIFAR100 setting at the final incremental stage: 200 image indices $\times$ 90 classes from 5,000 new-task images. As a result, there are no significant differences in performance, suggesting that diversity of sampled images does not affect the results because the pseudo-replay samples are diversified by adversarial attack toward various class prototypes (with noise).

4. Covariance Decomposition

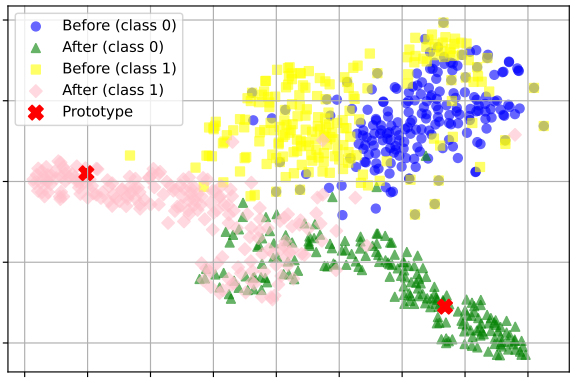
For the covariance decomposition experiment in Sec. 4.5 of Main Paper, we use the ablation study setting without prototype augmentation noise ($r = 0$) as the baseline, because the noise magnitude r depends on covariance matri-



(a) $t = 1$



(b) $t = 5$



(c) $t = 9$

Figure 6. t-SNE feature distribution analysis during $t = 1, 5, 9$ on CIFAR100. The perturbation effect persists both in the early and late incremental stages.

	NCM		Mahalanobis	
N_{cap}	A_{inc}	A_{last}	A_{inc}	A_{last}
4	68.95	55.95	69.92	57.65
∞	**69.00**	**56.29**	**69.96**	**57.94**

Table 11. Effect of candidate sampling limit per new-task image for CIFAR100 $T = 10$ setting. The baseline results from Main Paper Table 1 are highlighted in bold.

ces (Eq. 4 in Main Paper). To avoid irrelevant fluctuations, we use the saved network checkpoint at each task of the baseline run, and only the covariance decomposition setting is changed (with and without decomposition, different k values). The matrices U, S, V are recomposed into a full covariance matrix only during covariance calibration and Mahalanobis distance calculation at test time. Covariance shrinkage parameters γ_1 and γ_2 are determined for each k setting similar to the benchmark settings.

5. Storage Calculation

Required storage in Table 7 (Main Paper) is calculated with the following assumptions:

- Benchmark: ImageNet-Subset, $T = 10$, $t = 9$ (final stage)
- Prototypes: 90 classes, 512-dim (64-dim for AdaGauss)
- Covariances: 90 classes, 512-dim (64-dim for AdaGauss)
- Sample indices: 200 candidates $\times$ 90 classes, in `int64`
- Aug. params: (10 `int64` and 3 `bool` params) $\times$ 13k images
- Actual image data (for comparison, Sec. 4.5): 200 samples $\times$ 90 classes, each of shape (3, 224, 224) in `float32`

6. Training time

We compare detailed wall-clock measurements among methods in Table 12. Due to the online adversarial attack, APR's training time at incremental tasks is larger than the other methods.

Component	FeCAM	ADC	AdaGauss	APR
Init task $t = 0$	2.8	2.8	2.8	2.8
Inc. task $t = 9$	0.0	1.1	1.1	2.9
Calibration (90 cls)	0.00	0.10	0.84	0.36
Total benchmark	2.9	13.5	20.1	31.0

Table 12. Comparison of training time (hour) at $t = 0$, $t = 9$ and calibration on ImageNet-Subset $T = 10$, averaged across three-seed runs.

7. Visualization

Fig. 6 visualizes the t-SNE plot of feature distributions before and after perturbation at $t = 1, 5, 9$ on CIFAR100. The perturbation effect persists in the late incremental stage.

8. Reproducibility checklist

Our implementation guarantees reproducibility including seed control and recording function. The following the reproducibility checklist based on [3].

- Specification of dependencies: **Yes. We provide docker environment for reproducible dependencies.**

- Training code: **Yes.**
- Evaluation code: Yes.
- Config files: **Yes. We provide hierarchical configs.**
- (Pre-)trained model(s): **Yes.**
- README file includes table of results accompanied by precise command to run: **Yes.**
- Seed control: **Yes, deterministic training with a class-shuffling seed and a randomnesses seed is possible.**
- Exact augment parameter recording format: **Yes. We record all the config parameters, precise command, wall time, git SHA and all the necessary metrics (at every incremental stage) in csv format.**

References

[1] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 1

[2] 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, 2024. 2

[3] J. Pineau et al. The machine learning reproducibility checklist (version 2.0). https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf, 2020. Accessed: Sep. 2025. 3

[4] Grzegorz Rypeść, Sebastian Cygert, Tomasz Trzcinski, and Bartłomiej Twardowski. Task-recency bias strikes back: Adapting covariances in exemplar-free class incremental learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 1, 2

[5] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2024. 1