Supplementary Materials for Exemplar-free Continual Representation Learning via Learnable Drift Compensation

Alex Gomez-Villa[1,2], Dipam Goswami[1,2], Kai Wang[1⋆], Andrew D. Bagdanov[4], Bartlomiej Twardowski[1,2,3], and Joost van de Weijer[1,2]

[1] Computer Vision Center, Barcelona, Spain
[2] Universitat Autonoma de Barcelona, Barcelona, Spain
[3] IDEAS NCBR, Warsaw, Poland
[4] MICC, University of Florence, Florence, Italy
{agomezvi,dgoswami,kwang,btwardowski,joost}@cvc.uab.es,
andrew.bagdanov@unifi.it

1 Training Settings

1.1 Supervised CL settings

We implemented existing methods like LwF [6] (with NCM [8] and SDC [9]), PASS [12], FeTrIL [7] and FeCAM [5] and optimized them for small-start settings. We use the PyCIL [11] framework. Here we share the details of the settings for reproducibility.

For CIFAR-100, we use the CIFAR-10 policy augmentations from PyCIL and use it for all the methods to have a fair comparison. For all datasets, we follow the common practice of using random crop and random horizontal flip for the training set.

LwF: For all datasets in the first task, we follow PyCIL and use a learning rate of 0.1, momentum of 0.9, weight decay of 0.0005 and train with a batch size of 128 for 200 epochs. The learning rate is reduced by 10 after 60 , 120 and 180 epochs. For all new tasks in CIFAR-100 and ImageNet100, we use a learning rate of 0.05. For TinyImageNet, we use a learning rate of 0.001. We train for 100 epochs and reduce the learning rate by 10 after 45 and 90 epochs. The temperature scale is set to 2. The LwF regularization strength is set to 10 for CIFAR-100 and TinyImageNet and 5 for ImageNet100. We use the models trained with LwF and use them with a NCM classifier and SDC. For FeTrIL and FeCAM, we train the first task in the same way as LwF.

SDC: For SDC [9], we use $\sigma = 0.3$, which is the standard deviation of the Gaussian kernel for estimating the weights of the drift vectors. For ImageNet100, we set $\sigma = 1.0$.

PASS: Following the PyCIL implementation of PASS, we set $\lambda_{fkd} = 10$ and $\lambda_{proto} = 10$.

⋆ Corresponding author

Algorithm 1: Learnable Drift Compensation

At the end of each training session t
input : D_t;f_θ^{t-1};f_θ^t;Prototype pool of old task classes p^C
Phase 1: Train feature extractor f_θ^t in D_t using any learning strategy
Phase 2: Train forward projector p_F^t using D_t; f_θ^{t-1}; f_θ^t
begin
  `freeze(`f_θ^{t-1}`,`f_θ^t`)`
  for minibatch *in* training_batches **do**
    `minimize(MSE(`$p_F^t(f_\theta^{t-1}$(minibatch)),f_θ^t(minibatch)))`
  end
end
Phase 3: Update old prototypes with p_F^t
begin
  for prototype *in* p^C **do**
    $p^{\text{prototype}} = p_F^t(p^{\text{prototype}})$
  end
end

FeTrIL: For all incremental tasks, we follow the same settings as the FeTrIL implementation from PyCIL [11].

FeCAM: FeCAM trains the first task and use the frozen model for all subsequent tasks. FeCAM use the stored prototypes and distribution statistics (covariance matrix) for all old classes. Following the implementation from [5], we use the covariance shrinkage hyperparameters of (1,1) and 0.5 as the Tukey's normalization value.

1.2 Self-supervised CL settings.

We use the existing CL exemplar-free methods namely PFR [4], CaSSLE [2], and POCON [3]. We utilize the code provided for each method. Here we include additional details of the settings for reproducibility.

For each dataset, we follow the image augmentation pipeline of SimCLR [1]. All the methods use Barlow Twins [10] as base self-supervised learning strategy.

PFR: We conduct our experiments with 500 epochs per task on CIFAR100 and 400 for ImageNet100. As for CL hyperparameters, $\lambda = 25$ for both datasets; the learning rate, optimizer, and training schedules are the same as proposed by the authors.

CaSSLe: We use the same training epochs as PFR. We strictly follow the hyperparameters shared by the authors.

POCON: For both datasets, we employ the configuration CopyOP. The hyperparameters setup is the same as the authors proposed for the 10-task partition of each dataset.

Table 1: Accuracy comparison storing and projecting N features per class against storing and projecting only class prototypes.

Method	Mean	$N = 5$	$N = 50$	$N = 100$	$N = 500$
CaSSLe + LDC	35.0 ± 0.2	21.1 ± 0.01	32.9 ± 0.03	33.9 ± 0.02	35.05 ± 0.04

2 Additional Ablation Experiments

Feature Storage. In this work, we update only prototypes (one sample mean per class), but one can argue that storing a set of features per class is better than a single class mean. In Table 1, instead of projecting a prototype per class, we project N stored features and use them to compute the class prototype at the classification stage. The results show that storing and projecting features do not present a significant advantage over projecting the class means.

References

1. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020) 2
2. Fini, E., da Costa, V.G.T., Alameda-Pineda, X., Ricci, E., Alahari, K., Mairal, J.: Self-supervised models are continual learners. In: CVPR (2022) 2
3. Gomez-Villa, A., Twardowski, B., Wang, K., van de Weijer, J.: Plasticity-optimized complementary networks for unsupervised continual learning. In: IEEE/CVF Winter Conference on Applications of Computer Vision (2024) 2
4. Gomez-Villa, A., Twardowski, B., Yu, L., Bagdanov, A.D., van de Weijer, J.: Continually learning self-supervised representations with projected functional regularization. In: CVPRW (2021) 2
5. Goswami, D., Liu, Y., Twardowski, B., van de Weijer, J.: Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. Advances in Neural Information Processing Systems **36** (2024) 1, 2
6. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence (2017) 1
7. Petit, G., Popescu, A., Schindler, H., Picard, D., Delezoide, B.: Fetril: Feature translation for exemplar-free class-incremental learning. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 3911–3920 (2023) 1
8. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Conference on Computer Vision and Pattern Recognition (2017) 1
9. Yu, L., Twardowski, Bartlomiej, Liu, Xialei, Herranz, L., Wang, K., Jui, S., Weijer, J.v.d.: Semantic drift compensation for class-incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6982–6991 (2020) 1
10. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. arXiv preprint arXiv:2103.03230 (2021) 2

11. Zhou, D.W., Wang, F.Y., Ye, H.J., Zhan, D.C.: Pycil: a python toolbox for class-incremental learning. SCIENCE CHINA Information Sciences (2023) 1, 2
12. Zhu, F., Zhang, X.Y., Wang, C., Yin, F., Liu, C.L.: Prototype augmentation and self-supervision for incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5871–5880 (2021) 1