# Supplementary Material: Category Adaptation Meets Projected Distillation in Generalized Continual Category Discovery


Grzegorz Rypeść*[1,2], Daniel Marczak[1,2], Sebastian Cygert[1,3], Tomasz Trzciński[1,2,4], and Bartłomiej Twardowski[1,5,6]

[1] IDEAS NCBR
[2] Warsaw University of Technology
[3] Gdańsk University of Technology
[4] Tooploox
[5] Autonomous University of Barcelona
[6] Computer Vision Center


## A CAMP training loss in detail

In this section, we describe the details of CAMP method, more specifically – representation learning during phase 1. Representation learning in CAMP consists of contrastive and entropy-based learning, which we describe below and conclude in a total loss function for CAMP at the end.

**Contrastive learning** We use unsupervised and supervised contrastive losses, namely SimCLR [5] and SupCon [12].

We calculate SimCLR [5] loss as:

$$\mathcal{L}_{SimCLR} = -\frac{1}{|B|}\sum_{i\in B} \log \frac{\exp\left(\boldsymbol{h}_i^\top \boldsymbol{h}_i'/\tau\right)}{\sum_i^{i\neq n} \exp\left(\boldsymbol{h}_i^\top \boldsymbol{h}_n'/\tau\right)}\,, \tag{1}$$

where $x_i$ and $x_i'$ are two views (random augmentations) of the same image in a mini-batch $B$, $\mathbf{h}_i = g(\mathcal{F}(x_i))$, and $\tau$ is a temperature value. $\mathcal{F}$ is the feature extractor, and $g$ is a multi-layer perceptron (MLP) projection head used in SimCLR method.

For data where labels are available ($T_L^t$) we can simply use supervised learning. Specifically, we employ SupCon [12] loss:

$$\mathcal{L}_{SupCon} = -\frac{1}{|B_L|}\sum_{i\in B_L}\frac{1}{|\mathcal{N}_i|}\sum_{q\in\mathcal{N}_i} \log \frac{\exp\left(\boldsymbol{h}_i^\top \boldsymbol{h}_q'/\tau\right)}{\sum_i^{i\neq n} \exp\left(\boldsymbol{h}_i^\top \boldsymbol{h}_n'/\tau\right)}\,, \tag{2}$$

where $B_L$ corresponds to the labeled subset of $B$ and $\mathcal{N}_i$ is the set of indices of other images that have the same label as $x_i$. We use the same projector as in SimCLR to produce representations $\mathbf{h}_i = g(\mathcal{F}(x_i))$ and the same temperature parameter $\tau$.


* Corresponding author, email: grzegorz.rypesc.dokt@pw.edu.pl

**Entropy based learning**

We also utilize two additional loss functions for training the feature extractor following [26] and applied in the self-distillation [1,4] fashion. We noticed that this improved results achieved by CAMP even though we do not utilize parametrical classifier like [26]. Formally, for a total number of categories $K$, we randomly initialize a set of prototypes $\mathcal{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K\}$, each representing one category. During training, we calculate the soft label for each augmented view $\boldsymbol{x}_i$ by performing softmax on cosine similarity between the hidden feature $\boldsymbol{z}_i = \mathcal{F}(x_i)$ and the prototypes $\mathcal{C}$ scaled by $1/\tau_s$:

$$\boldsymbol{p}_i^{(k)} = \frac{\exp\left(\frac{1}{\tau_s}(\boldsymbol{z}_i/||\boldsymbol{z}_i||_2)^\top(\boldsymbol{c}_k/||\boldsymbol{c}_k||_2)\right)}{\sum_{k'}\exp\left(\frac{1}{\tau_s}(\boldsymbol{z}_i/||\boldsymbol{z}_i||_2)^\top(\boldsymbol{c}_{k'}/||\boldsymbol{c}_{k'}||_2)\right)}, \tag{3}$$

and the soft pseudo-label $\boldsymbol{q}'_i$ is produced by another view $\boldsymbol{x}_i$ with a sharper temperature $\tau_t$ in a similar way. The loss functions are then cross-entropy loss $\ell(\boldsymbol{q}', \boldsymbol{p}) = -\sum_k \boldsymbol{q}'^{(k)} \log \boldsymbol{p}^{(k)}$ between the predictions and pseudo-labels:

$$\mathcal{L}_{pseudo} = \frac{1}{|B|}\sum_{i\in B}\ell(\boldsymbol{q}'_i, \boldsymbol{p}_i) - \varepsilon H(\overline{\boldsymbol{p}}), \tag{4}$$

or known labels:

$$\mathcal{L}_{CE} = \frac{1}{|B_L|}\sum_{i\in B_L}\ell(\boldsymbol{y}_i, \boldsymbol{p}_i), \tag{5}$$

where $\boldsymbol{y}_i$ denotes the one-hot label of $\boldsymbol{x}_i$. For the unsupervised objective, we additionally adopt a mean-entropy maximisation regulariser [1]. Here, $\overline{\boldsymbol{p}} = \frac{1}{2|B|}\sum_{i\in B}(\boldsymbol{p}_i + \boldsymbol{p}'_i)$ denotes the mean prediction of a batch, and the entropy $H(\overline{\boldsymbol{p}}) = -\sum_k \overline{\boldsymbol{p}}^{(k)} \log \overline{\boldsymbol{p}}^{(k)}$.

**Total loss for CAMP** On all data provided in a given task: $\mathcal{X}^t = \mathcal{X}_U^t + \mathcal{X}_L^t$, we calculate the following loss as $\mathcal{L}_{SSL} = \mathcal{L}_{SimCLR} + \mathcal{L}_{pseudo}$

Loss that we calculate only on the labeled data $\mathcal{X}_L^t$ is equal to $\mathcal{L}_{SL} = \mathcal{L}_{SupCon} + \mathcal{L}_{CE}$ Finally, the total loss function for CAMP is then equal to:

$$\mathcal{L}_{CAMP} = (1-\alpha)((1-\beta)\mathcal{L}_{SSL} + \beta\mathcal{L}_{SL}) + \alpha\mathcal{L}_{KD}, \tag{6}$$

where $\alpha, \beta \in [0, 1]$ are hyperparameters defining contribution of regularization and supervision respectively and $\mathcal{L}_{KD}$ is knowledge distillation loss calculated using a distiller and defined in Section 3 (main paper). In experimental section we set $\alpha$ to 0.5 or 0.1 (when exemplars are present) and $\beta$ to 0.35. In case of Class Incremental Learning scenario we set $\alpha$ to 0.9.

# B Experimental setup - details

## B.1 Generalized Continual Class Discovery

In order to fairly compare existing methods, which often were created for a very specific continual scenario, we evaluate them in a Generalized Continual

Category Discovery framework (GCCD). GCCD consists of an arbitrary number of disjoint tasks and can include exemplars. Each task consists of labeled and unlabeled data from known and novel classes. We extend well-established category incremental setting [3, 21] by adding unlabeled data and novel classes to each task. We formally define it in Section 3 of the main paper. We present differences between GCCD and other setting in Tab.1.

| Setting \ Characteristics | Methods | Partially labeled classes | Novel classes | Sequence of disjoint tasks | Arbitrary number of tasks |
|---|---|---|---|---|---|
| Class incremental learning | LwF [16], EWC [14], iCaRL [21], DER [3] | × | × | ✓ | ✓ |
| Self-supervised continual learning | CaSSLe [7], PFR [9], LUMP [17], POCON [8] | × | ✓ | ✓ | ✓ |
| Semi-supervised continual learning | NNCSL [11], CCIC [2], ORDisCo [25] | ✓ | × | ✓ | ✓ |
| Generalized category discovery | GCD [24], SimGCD [26] | ✓ | ✓ | × | × |
| Incremental generalized category discovery | IGCD [29] | ✓ | ✓ | × | ✓ |
| Continual generalized category discovery | PA [13] | ✓ | ✓ | ✓ | × |
| **GCCD (ours)** | | ✓ | ✓ | ✓ | ✓ |

**Table 1:** Generalized Continual Category Discovery is the most general setting. It includes partial labels, contrary to supervised and self-supervised learning, and requires discovering novel (unlabeled) classes, contrary to supervised and semi-supervised learning. Moreover, it works on a sequence of disjoint tasks, contrary to Incremental Generalized Category Discovery, which assumes that unlabeled data samples become labeled in the next task. Finally, unlike Continual Generalized Category Discovery, GCCD is not limited to only a single category incremental step.

Overall, the proposed setting in this paper is a generalization of the previous ones as we learn both known and novel classes in each of the learning stages and allow for partially-labeled classes, thus we make no distinction between the initial and subsequent learning stages. This scenario holds in many real-life applications, mostly when data comes sequentially and there are insufficient resources to label all images. Additionally, our experiments focus on equal split tasks (so without a sizeable first task) on many incremental steps and investigate the effect of different setting parameters, such as the fraction of novel classes or labeled data proportion, on models performance. For most GCCD experiments (Tab. 1, Fig.6, Fig.9 of the main body) we assume that methods know the number of novel classes due to simplicity purposes. We will publish the code of the framework and our method upon the acceptance of the manuscript. We present data splits which we used for evaluation of GCCD protocol in 2.

**Exemplars** In our experimental setting we utilize exemplars only for known classes as only they are given labels. We randomly choose which exemplars to store in the buffer. For each past, known class we store the same amount of exemplars.

### B.2 Class Incremental Learning

Here, we provide setup details for results presented in Tab.2 of the main body of the paper. Class Incremental Learning [16, 18] is a special case of GCCD, where

| Dataset | N | Known | Novel | Lab. known (%) |
|---|---|---|---|---|
| CIFAR100 | 5 | 16 | 4 | 50% |
| Stanford Cars | 4 | 40 | 9 | 50% |
| CUB200 | 5 | 32 | 8 | 50% |
| Aircrafts | 5 | 16 | 4 | 50% |
| DomainNet | 6 | 8 | 2 | 50% |

**Table 2:** Datasets we utilized in experiments, their splits and characteristics.

each data sample is labeled and there are no novel classes. To combat forgetting, typical approaches utilize exemplars [10, 21], expendable architectures [22, 27] or regularization [14, 16, 28] techniques. We compare CAMP to regularization baselines that consider a single neural network with constant number of parameters and no exemplars.

For the simplest baselines we utilize fine-tuning (no technique to combat forgetting) and joint that trains the network on the whole dataset. Then, we utilize LwF [16] which utilizes knowledge distillation, EWC [14] that regularizes weights of the neural network and PASS [32] as a prototype augmentation technique. Additionally, we use IL2A [31] and [20]. For all this methods we run their implementations in FACIL [18] and if the implementation is not available, we use PyCIL [30]. For all methods we use default hyperparameters and the same augmentations.

To compare CAMP to baselines, we utilize three commonly used benchmark datasets in the field of Continual Learning (CL): CIFAR-100 [15] (100 classes), ImageNet-Subset [6] (100 classes) and DomainNet [19] (345 classes, from 6 domains). DomainNet contains categories of different domains, allowing us to measure models' adaptability to new data distributions. We create each task with a subset of classes from a single domain, so the domain changes between tasks. We split datasets to $N$ equal tasks.

We train CAMP and CIL baseline methods from scratch. We compare all approaches with standard CIL evaluations using the classification accuracies after each task, and *average incremental accuracy*, which is the average of those accuracies [21].

## C Baseline methods details

In the following, we describe details of training feature extractors $\mathcal{F}$ of baseline methods.

**GCD** [24] combines contrastive unsupervised loss from Eq. 1 and supervised loss from Eq. 2 to train $\mathcal{F}^t$ on task $t$. Combined, the total $\mathcal{L}_{GCD}$ loss equals:

$$\mathcal{L}_{GCD} = (1-\beta)\mathcal{L}_{SimCLR} + \beta\mathcal{L}_{SupCon}, \tag{7}$$

where $\beta \in (0, 1)$ is a weighing parameter and is equal to 0.35 in our experiments following the original work [24].

**GCD+FD** GCD method was not designed to tackle continual scenarios as it suffers from catastrophic forgetting. In order to adapt it for the GCCD we follow most distillation-based continual learning methods [7, 9, 16]. We freeze previously trained feature extractor $\mathcal{F}^{t-1}$ and regularize currently trained feature extractor $\mathcal{F}^t$ with the outputs of $\mathcal{F}^{t-1}$:

$$\mathcal{L}_{FD} = \frac{1}{|B|}\sum_{i\in B} ||\mathcal{F}^t(x_i) - \mathcal{F}^{t-1}(x_i)||^2, \tag{8}$$

This form of distillation does not require labels and all the data from the current task can be used for a regularization.

The final loss function for feature extractor training is defined as follows:

$$\mathcal{L}_{GCD+FD} = (1-\alpha)\mathcal{L}_{GCD} + \alpha\mathcal{L}_{FD}, \tag{9}$$

where $\alpha \in [0, 1]$ is a hyperparameter defining the contribution of regularization.

**GCD+EWC** In this baseline method we improve GCD by enforcing Elastic Weight Consolidation regularization on $\mathcal{F}$ parameters using $\lambda$ parameter as described in [14]. We additionally add a linear head for training and change $\mathcal{L}_{SupCon}$ to cross entropy loss. In our experiments we set $\lambda$ to 5000 following the original work.

**SimGCD** [26] improves training $\mathcal{F}^t$ over GCD by adding two additional loss functions: popular cross entropy loss for labeled data which improves clustering capabilities and adapted mean-entropy maximisation regularisation [1] applied to all data present in the task. The loss is equal to

$$\mathcal{L}_{SimGCD} = ((1-\beta)\mathcal{L}_{SSL} + \beta\mathcal{L}_{SL}) \tag{10}$$

**IGCD** [29] uses the same loss function as SimGCD to train the feature extractor. However, it adapts SimGCD for continual settings by adding a replay buffer that helps to mitigate forgetting and provides for support sample selection. In each incremental step a random subset of data samples of each class in the task is added to the buffer. For classification IGCD utilizes Soft-Nearest-Neighbor classifier.

**PA** [13] utilizes proxy anchors to train $\mathcal{F}$ and a replay buffer to fight the forgetting. Proxy anchor loss is defined as:

$$\begin{aligned}\mathcal{L}_{SupPA} = & \frac{1}{|P^{0^+}|}\sum_{p\in P^{0^+}} \log\left(1 + \sum_{z\in Z_p^{0^+}} e^{-\mu(s(z,p)-\delta)}\right) \\ & + \frac{1}{|P^0|}\sum_{p\in P^0} \log\left(1 + \sum_{z\in Z_p^{0^-}} e^{\mu(s(z,p)+\delta)}\right)\end{aligned} \tag{11}$$

where $\mu > 0$ is a scaling factor and $\delta > 0$ is a margin which we set to 32 and 0.1 respectively, following the original work. Here, the function $s(\cdot,\cdot)$ denotes the cosine similarity score. $P^{0^+}$ represents same class PAs($e.g.$negative) in the batch.

Each proxy $p$ divides the set of embedding vector $Z^0$ as $Z_p^{0^+}$ and $Z_p^{0^-} = Z^0 - Z_p^{0^+}$. $Z_p^{0^+}$ denotes the same class embedding points with the proxy anchor $p$. The goal of the first term in the equation is to pull $p$ and its dissimilar but hard positive data together, while the last term is to push $p$ and its similar but hard negatives apart. The proxies are incrementally added in each task.

Following the original work [13], the total loss for PA method for training $\mathcal{F}$ is equal to:

$$\mathcal{L}_{PA} = \mathcal{L}_{SupPA} + \mathcal{L}_{KD} \tag{12}$$

# D Additional experiments

**Comparison to baselines**

We provide additional average accuracy after each task plots in Fig. 1. They represent results achieved by exemplar and exemplar-free methods on Stanford-Cars, CUB200, and FGVC Aircraft. CAMP achieves the best final accuracy, and results are consistent on all datasets. CAMP also achieves the best average accuracy after most of the tasks. However, on FGVCAircraft CAMP with 20 exemplars, is worse than GCD, with 20 exemplars after the second, third, and fourth tasks.

**Impact of $\beta$ hyperparameter** We verify the impact of *beta* hyperparameter (trade-off between SL and SSL losses) on CUB200. We measure average accuracy for all classes for $\beta$ equal to $0, 0.2, 0.4, 0.6, 0.8$ and 1.0 and present results in 2. CAMP achieves very low results for $\beta = 0$ as SL loss is not utilized in this case. Interestingly, accuracy for novel categories drops for $\beta > 0.6$ showcasing that the SSL part is crucial in obtaining good results for novel categories.

**Impact of number of exemplars on centroid adaptation** We verify how the number of exemplars (0, 5, 20 per each category) influences distance of memorized centroid to the real category centroid (denoted as distance to real-mean). We plot the results for CUB200 dataset split into 5 tasks in Fig. 3. We measure the distance to the real-mean before and after performing the centroid adaptation. Intuitively, the more exemplars are available, the better is the centroid estimation. This results are consistent for known and novel categories.

**Distillers vs adapters for known and novel classes**

We verify the impact of using different distillation functions and different adapters for CAMP on StanfordCars dataset split into four tasks. We utilize the same setup as in 4.4 (main body). We provide final all, known, novel accuracies in Fig. 3. The combination of MLP distiller and Linear adapter achieves 38.1%, 43.2%, and 15.0% on all, known and novel categories, respectively, which is the best result. This shows, that such combination improves results for known and novel classes. The results are consistent with Fig.5 (main paper). That proves the design choice of our architecture.

**Extended analysis of latent spaces in CAMP**

In Fig. 4, we present an analysis of latent spaces in different GCCD methods. We observe that using no distillation (GCD) leads to the best performance on the second task but the worst performance on the first task, as the model is

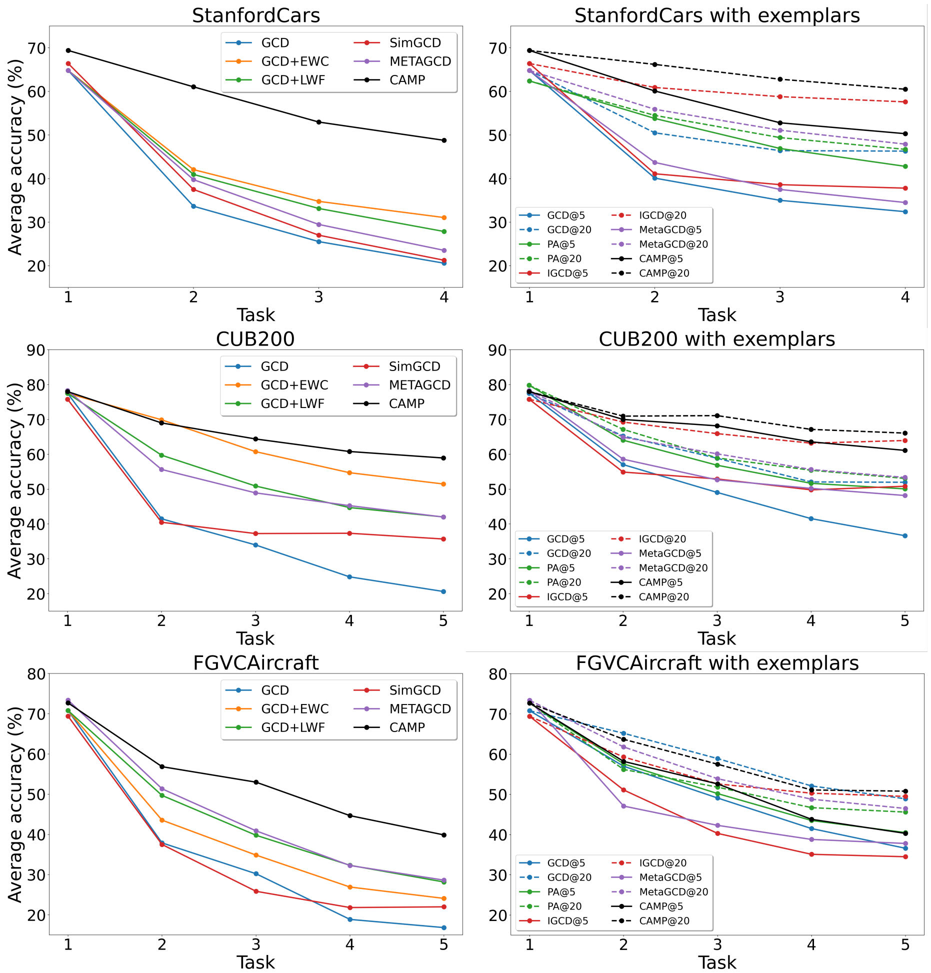


**Fig. 1:** Average accuracy after each task on three datasets. CAMP achieves the best accuracy after most of tasks.

optimized to fit the new data without regard for the past task. A rigid distillation (GCD + Feature Distillation) helps prevent forgetting but hinders learning new tasks. Moreover, we can see that feature distillation leads to the overlap between tasks. Our CAMP method uses projected knowledge distillation that enhances the ability to learn new tasks and learn representations that overlap less with those of old categories.

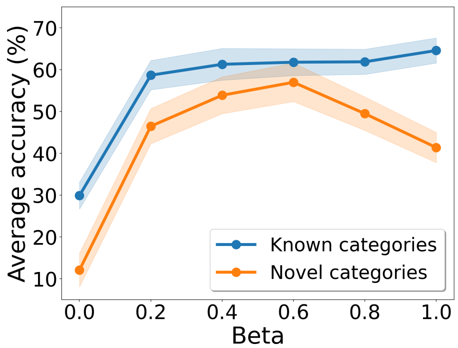


**Fig. 2:** Impact of $\beta$ hyperparametrer on known and novel accuracy achieved on CUB200.

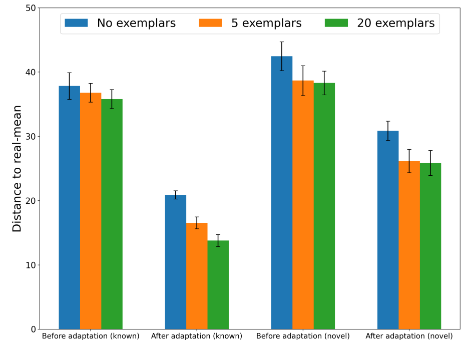


**Fig. 3:** Distance to real-mean before and after adaptation for 0, 5, 20 exemplars on CUB200.

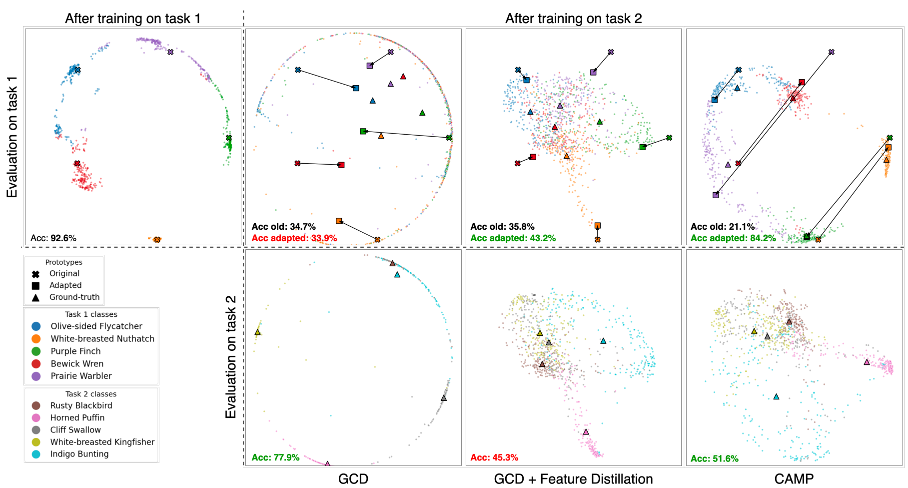


**Fig. 4:** CAMP utilizes a projected knowledge distillation that results in: (1) predictable drift in latent space that is revertible via centroids adaptation and leads to high performance on the first task and (2) high plasticity of the model and its ability to learn new tasks leading to high performance on the second task. Vanilla GCD fails to prevent forgetting. However, GCD with feature distillation reduces forgetting but diminishes the ability to learn new tasks. We report the nearest centroid classification accuracy. On the first task, we report accuracy using stored prototypes (Acc old) and adapted prototypes (Acc adapted).

| Adapter \ Distiller | None | FD | Linear | MLP | T-ReX |
|---|---|---|---|---|---|
| None | 20.6 | 28.3 | 25.2 | 22.5 | 21.3 |
| Linear | 25.6 | 28.5 | <u>35.5</u> | **38.1** | 26.5 |
| MLP | 26.6 | 26.0 | 34.4 | 33.7 | 23.1 |
| T-ReX [23] | 24.3 | 23.9 | 33.4 | 28.9 | 21.8 |
| SDC [28] | 23.8 | 28.1 | 29.4 | 31.0 | 23.8 |

| Adapter \ Distiller | None | FD | Linear | MLP | T-ReX |
|---|---|---|---|---|---|
| None | 23.9 | 32.1 | 28.9 | 25.8 | 24.5 |
| Linear | 29.4 | 32.5 | <u>40.2</u> | **43.2** | 30.1 |
| MLP | 30.3 | 30.1 | 39.5 | 38.8 | 26.5 |
| T-ReX [23] | 28.5 | 27.8 | 38.5 | 32.9 | 24.9 |
| SDC [28] | 28.0 | 32.4 | 33.3 | 35.5 | 27.0 |

| Adapter \ Distiller | None | FD | Linear | MLP | T-ReX |
|---|---|---|---|---|---|
| None | 5.8 | 11.3 | 8.2 | 7.7 | 7.0 |
| Linear | 8.6 | 10.2 | <u>14.3</u> | **15.0** | 10.5 |
| MLP | 9.7 | 7.4 | 11.4 | 11.1 | 8.2 |
| T-ReX [23] | 7.2 | 6.3 | 10.5 | 10.7 | 8.0 |
| SDC [28] | 6.9 | 10.3 | 9.8 | 11.0 | 9.6 |

**Table 3:** Impact of different adapters and distiller on our method on StanfordCars. We report all (top), known (middle) and novel (bottom) accuracy after the last task. Combination of MLP distiller and Linear adapter achieves the best results on all types of categories proving the design choice of CAMP.

## References


1. Assran, M., Caron, M., Misra, I., Bojanowski, P., Bordes, F., Vincent, P., Joulin, A., Rabbat, M., Ballas, N.: Masked siamese networks for label-efficient learning. In: European Conference on Computer Vision. pp. 456–473. Springer (2022)
2. Boschini, M., Buzzega, P., Bonicelli, L., Porrello, A., Calderara, S.: Continual semi-supervised learning through contrastive interpolation consistency. Pattern Recognition Letters (2021)
3. Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S.: Dark experience for general continual learning: a strong, simple baseline. Advances in neural information processing systems **33**, 15920–15930 (2020)
4. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021)
5. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations (2020)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
7. Fini, E., da Costa, V.G.T., Alameda-Pineda, X., Ricci, E., Alahari, K., Mairal, J.: Self-supervised models are continual learners. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022. pp. 9611–9620. IEEE (2022)
8. 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)
9. Gomez-Villa, A., Twardowski, B., Yu, L., Bagdanov, A.D., van de Weijer, J.: Continually learning self-supervised representations with projected functional regularization (2022)
10. Iscen, A., Zhang, J., Lazebnik, S., Schmid, C.: Memory-efficient incremental learning through feature adaptation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16. pp. 699–715. Springer (2020)
11. Kang, Z., Fini, E., Nabi, M., Ricci, E., Alahari, K.: A soft nearest-neighbor framework for continual semi-supervised learning. ICCV (2023)
12. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning (2021)
13. Kim, H., Suh, S., Kim, D., Jeong, D., Cho, H., Kim, J.: Proxy anchor-based unsupervised learning for continuous generalized category discovery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16688–16697 (2023)
14. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
15. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. . (2009)
16. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)

17. Madaan, D., Yoon, J., Li, Y., Liu, Y., Hwang, S.J.: Representational continuity for unsupervised continual learning. In: International Conference on Learning Representations (ICLR) (2022)
18. Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A.D., van de Weijer, J.: Class-incremental learning: survey and performance evaluation on image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)
19. Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., Wang, B.: Moment matching for multi-source domain adaptation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1406–1415 (2019)
20. 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)
21. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017)
22. Rypeść, G., Cygert, S., Khan, V., Trzciński, T., Zieliński, B., Twardowski, B.: Divide and not forget: Ensemble of selectively trained experts in continual learning. arXiv preprint arXiv:2401.10191 (2024)
23. Sariyildiz, M.B., Kalantidis, Y., Alahari, K., Larlus, D.: No reason for no supervision: Improved generalization in supervised models. In: ICLR 2023-International Conference on Learning Representations. pp. 1–26 (2023)
24. Vaze, S., Han, K., Vedaldi, A., Zisserman, A.: Generalized category discovery. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7492–7501 (2022)
25. Wang, L., Yang, K., Li, C., Hong, L., Li, Z., Zhu, J.: Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5383–5392 (2021)
26. Wen, X., Zhao, B., Qi, X.: Parametric classification for generalized category discovery: A baseline study. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16590–16600 (2023)
27. Yan, S., Xie, J., He, X.: Der: Dynamically expandable representation for class incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3014–3023 (2021)
28. Yu, L., Twardowski, B., Liu, X., Herranz, L., Wang, K., Cheng, Y., 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)
29. Zhao, B., Mac Aodha, O.: Incremental generalized category discovery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19137–19147 (2023)
30. Zhou, D.W., Wang, F.Y., Ye, H.J., Zhan, D.C.: Pycil: a python toolbox for class-incremental learning. SCIENCE CHINA Information Sciences **66**(9), 197101– (2023)
31. Zhu, F., Cheng, Z., Zhang, X.y., Liu, C.l.: Class-incremental learning via dual augmentation. Advances in Neural Information Processing Systems **34** (2021)
32. 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)