28th *Computer Vision Winter Workshop*
Graz, Austria, February 12–14, 2025

DOI: 10.3217/978-3-99161-022-9-004

Incremental Learning with Repetition via Pseudo-Feature Projection

Benedikt Tscheschner[1,2] Eduardo Veas[1] Marc Masana[2,3]

[1]Know-Center Research GmbH
[2]Institute of Visual Computing, TU Graz
[3]SAL Dependable Embedded Systems, Silicon Austria Labs
{btscheschner, eveas}@know-center.at, mmasana@tugraz.at

Abstract

Incremental Learning scenarios do not always represent real-world inference use-cases, which tend to have less strict task boundaries, and exhibit repetition of common classes and concepts in their continual data stream. To better represent these use-cases, new scenarios with partial repetition and mixing of tasks are proposed, where the repetition patterns are innate to the scenario and unknown to the strategy. We investigate how exemplar-free incremental learning strategies are affected by data repetition, and we adapt a series of state-of-the-art approaches to analyse and fairly compare them under both settings. Further, we also propose a novel method (Horde), able to dynamically adjust an ensemble of self-reliant feature extractors, and align them by exploiting class repetition. Our proposed exemplar-free method achieves competitive results in the classic scenario without repetition, and state-of-the-art performance in the one with repetition.

1. Introduction

As autonomous agents and models in production systems are exposed to continuous streams of information, they are required to adapt to dynamic data distributions with potentially multiple tasks and integrate new information over time [3, 28, 43]. The practice of retraining the complete system whenever new data is available becomes unfeasible as the storage, computation and privacy constraints for data streams increase [31, 35, 39]. To address these constraints, incremental learning (IL) or continual learning has emerged as a promising approach [4].

IL aims to learn a model sequentially through a sequence of tasks introducing disjoint sets of information at each training step [8, 27, 42]. Generally, these scenarios enforce a strict no-repetition constraint [7] allowing access to the data distribution only once in the task sequence. Unlike humans, who can learn nearly inference-free between tasks, neural networks suffer from a phenomenon called *catastrophic forgetting* [10, 12]. When models are optimized sequentially on novel tasks, a swift forgetting of previously learned tasks is observed. To mitigate this forgetting, a delicate balance between preserving learned task knowledge (stability) and the ability to adapt to new information (plasticity) has to be reached, which is known as the stability-plasticity dilemma [29]. A popular approach to address this is to cache a representative subset of previously encountered data points in a buffer and replay them during the following training sessions [38, 41, 42]. Although such *rehearsal* addresses catastrophic forgetting effectively, data privacy concerns have been raised [14], and the scalability of an exemplar buffer in long-tailed incremental sequences is questionable [42] due to the large computational cost of complete retraining and significant storage requirements.

Nonetheless, the strict enforcement of no-class repetition becomes unrealistic for many real-world applications, as continuous streams are bound to repeat certain information [7] or be affected by semantic or covariate shifts [30]. For example in industrial defect detection, certain common defects and defect-free samples will repeat throughout production. The occurrence of repetition is further amplified in environments where an agent has the freedom to reexperience elements which are contained within the overall environment design. Thus the effects of catastrophic forgetting are likely exaggerated as an uncontrollable form of rehearsal occurs naturally. Previous incremental learning research has largely explored catastrophic forgetting under the assumption that new information has a single opportunity to be learned, since each class is only available within a single task throughout the sequence. The introduction of repetition into these scenarios enables the selection of more broad incremental training tasks and highlights the different dynamics within the plasticity-stability dilemma of learning new tasks while maintaining current knowledge [7]. The focus on catastrophic forgetting without repetition may limit the development of more realistic incremental learning agents, which involve different complex objectives like forward transfer [24] and efficiency for computational limitations in edge devices [9].

As such, we want to loosen the no-repetition constraint and explore the effects of natural repetition. To explore these new settings and effects, our contributions are:

1. a new variation of the class-incremental learning

CIFAR 50/10 scenario introducing class repetition,
2. benchmarking a broad selection of state-of-the-art exemplar-free class-incremental learning methods and investigate the effects of innate data repetition and their resiliency to repetition frequency bias,
3. a novel incremental learning method (Horde) that builds an ensemble of independent feature extractors for stability and utilizes pseudo-feature projection for plasticity (see Fig. 1).

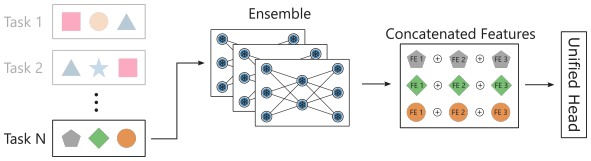


Figure 1. Overview of our proposed method (Horde). Each data sample is processed by an ensemble of independent feature extractors. The features from all extractors are concatenated before being passed into a unified head that can accommodate the dynamic input size through pseudo-feature projection.

2. Related Work

Class-incremental learning (CIL) addresses the challenge of training a model sequentially on a series of tasks, without access to previous or future data [36]. When training without any constraints, models fail to retain knowledge from previous tasks – a problem known as *catastrophic forgetting* [10, 12]. Usually, each incremental task contains a disjoint set of new classes, which increases the difficulty of discriminating between those which have not been learned together under the same task [8]. A key challenge in incremental learning lies in keeping the balance of the stability-plasticity dilemma [29], critical for mitigating catastrophic forgetting while ensuring the adaptability of the model to new tasks.

Incremental learning approaches include: weight regularization [2, 20], which preserves important weights by estimating their importance; knowledge distillation [18, 22], which focuses on protecting task representations rather than weights; rehearsal [38], which replay stored exemplars from previous tasks; mask-based approaches [26], which use task-specific masks to isolate parameters that can be updated; and dynamic network structures [11, 32], which expand the model architecture by adding new or contracting existing modules for each task. In this work, we concentrate on weight-regularization, knowledge distillation and dynamic network structure-based methods. These are the approaches that work on task-agnostic scenarios (do not require a task-ID during inference) and promote privacy preservation (do not store samples).

Incremental learning with repetition. In many practical applications (automated failure inspection, medical imaging, robotics), pattern repetition naturally arises, yet traditional CIL approaches assume that each class is encountered only once, imposing a strict no-repetition constraint [7]. This constraint focuses on the prevention of catastrophic forgetting but also diverges from real-world scenarios where classes may reappear or shift over time. To address this, Hemati et. al [15, 16] propose an extension to the class-incremental learning scenario which models the repetition of individual classes outside of a single task. Unlike joint incremental or rehearsal-based learning, this repetition is innate to the learning scenario and cannot be adjusted. This emphasizes an experience-based scenario [41], which favours shorter training tasks that can sometimes only cover a part of the class distribution. Moreover, covering scenarios that lie between the classic offline incremental and the online ones.

Class-incremental learning with repetition has received increased interest in the research community, being a central element in the challenge tracks of the last two CLVISION challenge tracks at CVPR 2023 and 2024 [1, 16]. In the 2023 edition, we competed with a base variant of our proposed method, although without elements for controlling ensemble growth (see Sec. 3.1), self-supervision (see Sec. D) or applicability to variable network architectures.

Class prototypes and pseudo-features. To enforce stability and alleviate class-recency biases in the classifier [27], Exemplar-Free Class Incremental Learning (EFCIL) methods [33, 40, 46–48] utilize class prototypes to simulate unavailable classes. These prototypes capture statistical properties of embedding representations of each class, which are usually modeled as a multivariate Gaussian distribution [40, 46, 47]. Specifically, the statistics typically include the mean and covariance of feature representations for each class, allowing to generate pseudo-features when class data is not available. To extract representations, the neural network is divided into two modules. A feature extractor (FE) that projects the input samples into their corresponding embedding representation; and a classifier head that uses these embeddings to solve the classification task. Therefore, prototype-based methods can generate embeddings even when no samples from past classes are available during subsequent tasks by sampling the stored distributions of each class. The sampled embedding representations are rehearsed alongside the current task data, thus promoting stability and mitigating class-recency bias. However, in order to maintain valid approximations of class distributions, the feature extractor needs to be either frozen or heavily regularized to prevent changes or drifts in the extracted features. Unlike rehearsal-based approaches, the use of prototypes does not violate data privacy due to the non-linearly projected representation in the embedding space [44, 47].

Feature translation. Instead of sampling the distribution approximated by class prototypes, FeTrIL [33] proposes to translate the features of available data classes to unavailable ones directly. Given a feature extractor $\mathbf{f}(x;\theta)$ being trained on current data $\{(x_i, y_i)\}$, its output embedding F is efficiently translated from one of the current

classes to the desired previously learned class $c \in \mathcal{Y}$ as

$$\hat{F}_c = \mathbf{f}(\boldsymbol{x}_i; \theta) + \mu_c - \mu_{\boldsymbol{y}_i} \,, \tag{1}$$

where μ_c and $\mu_{\boldsymbol{y}_i}$ represent the means of the old and current classes, respectively. The feature translation modifies the classifier, however, the feature extractor is requires to be frozen after the initial training so that the class means can be reliably extracted. This limits the continual learning process as the initial task constrains the diversity and robustness of the features that can be learned for new classes [4, 33]. In our proposed approach, we relax this restriction by allowing an ensemble of smaller feature extractors to be learned. This allows for unknown class prototypes to be estimated through pseudo-feature projection until the repetition of classes allows for an accurate extraction of class prototypes.

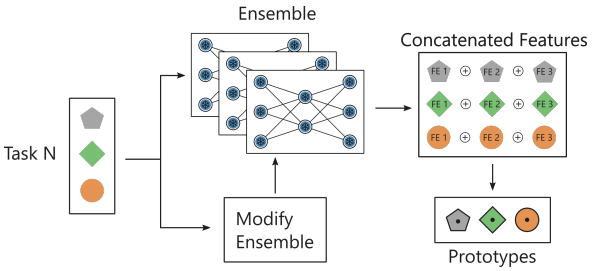
3. Method

In incremental learning scenarios with repetition, the reappearance of classes introduces uncertainty in task sequences, requiring strategies that handle dynamic class distributions. Our approach aims to: (a) capture information from the current task, (b) integrate it with knowledge from previously seen tasks, and (c) ensure the ability to discriminate between all encountered classes so far. To achieve this, we leverage zero-forgetting feature extractors (FEs), which are aggregated in an ensemble to overcome the limitation of a completely fixed feature space. Through this aggregation, we form a flexible feature representation space that can adapt (expand or contract) based on the incremental learning sequence (see Fig. 1 for an overview of the proposed method structure).

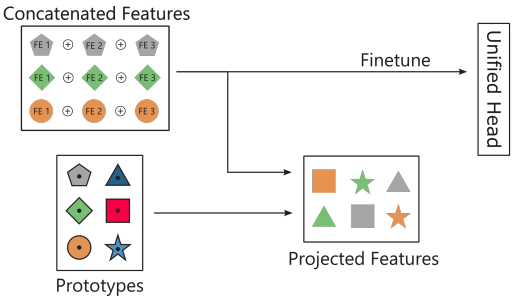
To effectively utilize this dynamic embedding space, we address the challenge of missing classes by constructing prototypes for all encountered classes. Class prototypes are used to train a unifying classification layer through an adjusted feature translation mechanism, termed *pseudo-feature projection*, ensuring continuous adaptation and robust performance across all classes. Concretely, the learning approach is divided into two steps: **(1st)** based on the difficulty of the current task and depending on how well new classes can fit into the ensemble embedding space, the ensemble is expanded with a new feature extractor (described in Sec. 3.1); **(2nd)** once the embedding space has been fixed for the current task, class prototypes are extracted and the unified classification layer is trained through the *pseudo-feature projection* (described in Sec. 3.2). These steps are performed for every incremental task and are summarized in Figure 2.

3.1. Feature Extractor Ensemble

The proposed aggregation framework consists of multiple individual feature extractors (FEs), each trained on a specific task and then frozen to preserve the learned representations. The motivation for this zero-forgetting strategy is to enforce stability, avoiding any catastrophic forgetting on the ensemble while providing some plasticity



(a) **Step 1**: The ensemble of feature extractors is adjusted based on the current task through either the addition or update of a self-reliant feature extractor. This step is only performed when estimated as necessary via a heuristic criteria.



(b) **Step 2**: Class prototypes are extracted or updated from the current task data. Incomplete class prototypes (those estimated before Step 1 extends or modifies a feature extractor) are updated and data for unavailable classes is simulated by pseudo-feature projection. An unbiased classification head is finetuned from the current training data and the projected features of unavailable classes.

Figure 2. Overview of the steps our proposed method (Horde) performs for each incremental task.

through the extension of the ensemble. Unlike FeTrIL, which freezes a single feature extractor after the initial training, the extension of the feature space through the ensemble relieves the dependence of an expressive initial feature extractor. The goal of each feature extractor is to build a diverse and expressive feature space that emphasizes high-quality representations rather than optimizing the performance of the individual incremental task. Further, we adopt the self-learning loss from PASS [47]. This self-learning loss enhances the learned feature representation by simultaneously classifying image orientation and categories (each image class now has 4 augmented labels depending on the image orientation). To further improve regularization on the feature space topology, we incorporate a metric learning head with contrastive loss [34] and hard-negative mining [37]. This promotes *spherical-shaped* clusters in the embedding space, which improves class discrimination between known and unknown distributions [25]. Additionally, the sphere-shaped structure aligns well with the properties of a multivariate Gaussian distribution, which relates to the pseudo-feature projection we propose. An ablation study of the effects of individual components is provided in the supplementary material (see Sec. D).

Ensemble Growth. To control the growth of the ensemble, we set a predefined budget B for the maximum num-

ber of FEs. For each incremental task, a decision is made whether the concatenated embedding space should be adjusted based on the following criteria:

- *constant feature representation*: when the current ensemble embedding representation is sufficient to handle the incremental task, no new FE is trained. New classes are learned using the existing ensemble representations without requiring additional feature extraction capacity.

- *dynamic feature adaption*: when the current ensemble of FEs cannot adequately represent the new task due to a significant change in the data distribution, task complexity or overlap with previous classes, a new FE is added.

To capture these criteria and guide the growth of the ensemble, we propose two heuristics to guide the modification of the ensemble (see Step 1 in Fig. 2a):

- **Class Set Maximisation (Horde$_m$)**: this heuristic aims to maximize the diversity of classes represented across the ensemble. Specifically, it ensures that each FE contributes to representing as much of a distinct set of classes as possible

$$\max \bigcup_{i \in B} \left| c \in F^i \right| , \tag{2}$$

  thereby increasing the overall coverage of the class space across all feature extractors. This maximization is tested at the start of each incremental task. Thus, when a larger class set is possible with the current incremental task data, a new FE is trained. The new FE either is added or replaces one in the ensemble.

- **Task Error Rate (Horde$_c$)**: At the start of the incremental task, the error rate e on the current incremental data is computed (before training). It is obtained from the confusion matrix (CM) by calculating the ratio of wrong predictions over all other predictions:

$$e = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} \left(\frac{\sum_{j \neq c} \mathrm{CM}_{c,j}}{\sum_i \mathrm{CM}_{c,i}} \right) . \tag{3}$$

  If e is too high, the incremental data cannot be classified with the current ensemble effectively. Therefore, we introduce a threshold or budget of the ensemble B which signals the need to train a new feature extractor based on e. After training the unified head (Step 2), an *improvement score* is calculated as the difference between the error rate at Step 1 (before any training is performed) and after Step 2. If the budget B has been exceeded the FE with the lowest improvement score is replaced.

3.2. Unified Classification Layer

To unify the feature representations from the ensemble and enable task-agnostic classification, we utilize a fully-connected layer. This layer has dynamic input and output sizes depending on the growth of the ensemble and the number of incrementally learned classes. To mitigate task-recency bias [8], we train this unified head using both data from the current task and projected class prototype features through our proposed pseudo-feature projection.

Pseudo-feature projection. Pseudo-feature projection, inspired by FeTrIL [33], extends feature translation by incorporating both the mean and standard deviation of class prototypes. This enhances the sampling of dimensions, reduces the chance of overlapping classes in the embedding space and leads to more accurate feature replay. With this projection, a data point from one class may be projected to a pseudo-feature representation of any other previously learned class. Our proposed projection extends the one from FeTrIL on Eq. (1) as

$$\hat{F}_c = \mu_c + \frac{\mathbf{f}(\boldsymbol{x}_i; \theta) - \mu_{\boldsymbol{y}_i}}{\boldsymbol{\sigma}_{\boldsymbol{y}_i}} \cdot \boldsymbol{\sigma}_c \, , \tag{4}$$

where $\hat{F}_c$ represents the pseudo-features of the latent representation of a data point $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ which is projected from the original class $\boldsymbol{y}_i$ to the desired class c. This transformation leverages the class prototypes; specifically the mean $\mu_{\boldsymbol{y}_i}$ and standard deviation $\sigma_{\boldsymbol{y}_i}$ to modify the latent representation $\mathbf{f}(\boldsymbol{x}_i; \theta)$. Class prototypes are updated during Step 2, before training the unified classification layer and after the ensemble has been adjusted.

We represent a complete class prototype as the concatenation of the individual class statistics from each FE in the ensemble:

$$\begin{aligned} \boldsymbol{\mu}_c &= (\boldsymbol{\mu}_{c,1}, \, \ldots \, , \boldsymbol{\mu}_{c,n}), \\ \boldsymbol{\sigma}_c &= (\boldsymbol{\sigma}_{c,1}, \, \ldots \, , \boldsymbol{\sigma}_{c,n}), \end{aligned} \tag{5}$$

where n determines the current size of the ensemble. Throughout the incremental sequence, the ensemble can be expanded until the feature extractor budget is exhausted ($n \leq B$). Once this limit has been reached, individual feature extractors need to be finetuned or replaced and their corresponding class prototype ($\boldsymbol{\mu}_{c,i}$, $\boldsymbol{\sigma}_{c,i}$) is reset.

Class prototypes of certain classes may be incomplete for newly added or modified FEs. When class statistics are unknown for a specific FE, estimates are required for pseudo-feature projection to calculate $\hat{\boldsymbol{\mu}}_{c,\mathbf{f}}$ and $\hat{\boldsymbol{\sigma}}_{c,\mathbf{f}}$. In the absence of statistical information, we fix the standard deviation to $\hat{\boldsymbol{\sigma}}_{\mathbf{c},\mathbf{f}} = \mathbf{1}$. This decision is based on the fact that the estimation of $\hat{\boldsymbol{\mu}}_{c,\mathbf{f}}$ already provides sufficient variance. Therefore, for the estimation of the mean component $\hat{\boldsymbol{\mu}}_{c,\mathbf{f}}$ we propose three heuristics:

1. **zeros**: clamping all $\hat{\boldsymbol{\mu}}_{c,\mathbf{f}}$ estimations to 0

$$\hat{\boldsymbol{\mu}}_{c,\mathbf{f}} = \mathbf{0} \, , \tag{6}$$

2. **random**: randomly sample $\hat{\boldsymbol{\mu}}_{c,\mathbf{f}}$ from a multivariate normal distribution

$$\hat{\boldsymbol{\mu}}_{c,\mathbf{f}} \sim \mathcal{N}(\mathbf{0}; \boldsymbol{\Sigma}) \, , \tag{7}$$

3. **original features**: estimate $\hat{\boldsymbol{\mu}}_{c,\mathbf{f}}$ with the original representation of the transforming sample and use them without modification

$$\hat{\boldsymbol{\mu}}_{c,\mathbf{f}} = \mathbf{f}(\boldsymbol{x}_i; \theta) \, . \tag{8}$$

Est. Method	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Avg. Acc. ↑
zeros	73.3	39.9	32.9	35.2	25.2	25.4	24.9	19.2	18.3	19.1	17.1	30.0
random-1.0	73.3	39.9	32.9	35.2	25.2	25.4	24.8	19.2	18.3	19.0	17.2	30.0
random-3.0	73.3	47.0	37.9	38.0	29.0	31.3	30.0	22.9	21.9	22.5	22.5	34.2
random-5.0	73.3	52.5	42.7	44.4	33.6	34.5	33.8	25.2	25.9	26.3	26.4	38.0
random-10.0	73.3	59.2	50.8	52.7	43.5	43.4	40.2	31.4	31.8	31.8	31.7	44.5
random-15.0	73.3	61.7	53.5	54.2	45.6	46.1	42.0	35.8	**36.1** ●	**34.8** ●	34.4	47.0
random-20.0	73.3	62.5	55.5	**54.8** ●	47.3	47.0	42.9	36.5	35.5	**34.8** ●	34.4	47.7
random-30.0	73.3	**62.9** ●	55.9	**55.1** ●	**48.8** ●	47.6	**43.7** ●	36.8	36.0	**34.8** ●	**34.7** ●	**48.1** ●
random-40.0	73.3	62.6	**58.0** ●	**54.8** ●	46.9	**47.8** ●	**44.9** ●	**38.6** ●	**36.5** ●	**35.9** ●	**34.9** ●	**48.6** ●
random-50.0	73.3	**64.1** ●	57.2	**54.8** ●	**47.6** ●	**47.9** ●	42.0	**37.6** ●	33.8	**34.8** ●	34.4	48.0
random-75.0	73.3	62.1	**57.9** ●	54.5	46.2	45.4	40.7	35.6	34.1	33.7	33.6	47.0
random-100.0	73.3	62.6	56.5	53.9	46.5	45.8	40.7	34.4	34.2	33.8	33.4	46.8
original features	73.3	**64.3** ●	**61.3** ●	**60.8** ●	**55.1** ●	**53.7** ●	**52.7** ●	**46.8** ●	**47.1** ●	**46.3** ●	**45.2** ●	**55.2** ●

Table 1. Results for class prototype estimation when the corresponding class prototype is not available during training. The evaluation is performed on a CIL 50/10 setup with Slim-Resnet-18 only. **1st** ●, **2nd** ● and **3rd** ● best metrics are marked accordingly.

We evaluate the proposed feature estimation heuristics in an empirical experiment on a class-incremental learning scenario with no repetition. The results on CIFAR 50/10 trained on a Slim-Resnet-18 are listed in Table 1 (see Sec. 4 for more details). This scenario requires the estimation of class prototype components (e.g., mean, variance) at each incremental task and the estimation is essential for the classification. The *original features* estimation performed best, and this heuristic is the one used in all subsequent experiments.

In EFCIR scenarios, the repetition of classes within incremental tasks enhances the performance of pseudo-feature projection as it aligns individual FE representation spaces by eliminating the need for estimating class prototype components. During class repetition they can be directly calculated from the available task.

4. Experimental Setup

Most incremental learning methods expect a different set of classes with all dataset samples for each class available when learning its corresponding task. However, when class repetition is introduced, the complexity of potential scenarios increases significantly, and where sequence length and repetition frequency become additional variables. To address this, we propose an analysis into the effects of class repetition within a setting that shares many characteristics of traditional incremental learning but incorporates longer sequences with class repetition. Code for the proposed scenarios and methods is available[1].

Overall, the proposed experiments aim to analyze a) the performance of IL methods in scenarios without repetition (baseline), b) the performance of CIL with small incremental tasks and class repetition, and c) the resilience of the methods against bias deviations in repetition frequency.

Ideally, we expect the average accuracy of our proposed method to be on par with state-of-the-art methods on (a) and to outperform them in (b) and (c). To validate this, method performance will be ranked based on average accuracy for all scenarios (a – c).

4.1. Compared Methods

We benchmark a total of 14 methods, which include two rehearsal-based approaches, five incremental learning methods, five state-of-the-art exemplar-free class-incremental learning (EFCIL) methods, and two variants of our proposed approach. The two rehearsal-based methods are excluded from the ranking and serve as an upper baseline (Joint [8]) and a reference point (Weight-Alignment (WA) [45]; $n = 2000$).

The five incremental learning methods consist of two baseline methods (Freezing (FZ) and Finetuning (FT) [27]), and three classic IL methods Elastic Weight Consolidation (EWC) [20], Memory Aware Synapses (MAS) [2] and Learning without Forgetting (LWF) [22]. These three methods were not originally proposed for CIL, thus, requiring the use of a task-ID at inference time. However, they are easily and commonly adaptable to task-agnostic settings. As such, we performed a grid search for their optimal hyperparameters based on the CIL 50/10 setting and used these for the repetition settings.

The five state-of-the-art, rehearsal-free, protoype-based methods comprise: Prototype Augmentation and Self-Supervision (PASS) [47], Class-Incremental Learning with Dual Supervision (IL2A) [46], Self-Sustaining Representation Expansion (SSRE) [48], Prototype Reminiscence and Augmented Asymmetric Knowledge Aggregation (PRAKA) [40] and Feature Translation for Exemplar-free Class Incremental Learning (FeTrIL) [33]. These methods were originally reported on the CIL CIFAR 50/10 setting. Therefore, since the proposed repetition scenarios are closely related to this setting, we use the hyperparameters proposed by the original authors.

Finally, we evaluate our proposed method with both ensemble growth heuristics (Horde_m and Horde_c). A de-

[1] www.github.com/Tsebeb/cvww_cir_horde

tailed overview of the used hyperparameters is provided in the supplementary material (see Sec. C).

4.2. Model Architecture

All methods employ the same base feature extractor, a ResNet-18 [13] model that has been adjusted to the CIFAR input dimensions [46, 47]. For our approach, which utilizes an ensemble of feature extractors, we employ a slimmed-down variant of ResNet-18 for incremental tasks. This variant reduces the number of channels/filters for convolutions while preserving the network's depth (see supplementary material Sec. B). With this reduced architecture, we construct an ensemble consisting of one full ResNet-18 and nine Slim-ResNet-18 models (making our budget $B = 10$). This configuration results in a total number of parameters and computational requirements (see Table 2) that are roughly equivalent to those of knowledge distillation approaches (or importance weight estimation [2, 20, 23]).

4.3. Scenarios

Experiments are conducted on the CIFAR-100 dataset [21], employing data augmentation in line with other CIL methods [40, 46]. These augmentations consist of a 4-pixel zero padding of the input image and a random cropping to the original 32×32 size. Followed by a random horizontal flip, image brightness jitter and image normalization.

To evaluate the effects of repetition on CIL methods, we organize the experiments in three scenarios. First, a baseline is established by evaluating (a) all methods on an incremental learning scenario without repetition.

(a) **CIL 50/10.** The classic task-agnostic class-incremental scenario consisting of an initial training session with 50 classes and followed by 10 incremental tasks, each containing five novel classes.

Second, we evaluate (b) performance on a modified CIL scenario where classes repeat in the task sequence. Specifically, the scenario is built by replacing the discrete incremental tasks with clear boundaries from CIL 50/10 with small (2,000 training samples per task) incremental tasks that can contain class repetition. Each class, old or new, has the same probability of being in an incremental task.

(b) **EFCIR-U 50/100.** Similarly to the CIL 50/10 scenario, the initial training also covers 50 classes. An essential element of repetition is a mixture of new and already seen samples. Therefore, we only provide 50% of the available training data samples for the initial training. Following the initial task, the scenario consists of 99 small, incremental tasks, with a limit of 2,000 training samples each. Both the initial 50 and incremental 50 classes have a fixed probability of 15% of being discovered or repeated in an incremental task so that tasks do not contain too many classes on average. The number of samples per class in a task are balanced as in the CIL 50/10 scenario.

Model	# Parameters
ResNet-18	11.307.956
Slim ResNet-18	1.109.240
Knowledge Distillation	22.615.912
Ensemble (ours)	21.291.116

Table 2. Number of parameters for different architectures.

In the third scenario, the aim is to assess the IL method's (c) resilience against biases in repetition frequency. To establish this bias during scenario creation we propose to draw the repetition probability of each class from a *Beta Distribution* [19]. An illustration of the repetition bias is provided in the supplementary material Sec. E.

(c) **EFCIR-B 50/100.** To test the resiliency against repetition frequency, we sample individual class repetition probabilities $p \sim \text{Beta}(\alpha, \beta)$ with parameters $\alpha = 3.5$ and $\beta = 20.0$. This way, the expectation $\mathbb{E}[\text{Beta}(3.5, 20.0)] \approx 0.15$ is similar to the uniform EFCIR-U scenario, implying that on average the same number of classes are present in each task.

In scenarios with repeated classes, the optimal learning rate and number of epochs depend on various factors (*e.g.* method, number of training samples, length of incremental sequence) and are highly influential. To address this, we split 10% of the available training data as a validation set. For all methods, we apply early stopping [5, 34] using this validation data for the classes present in the task. We monitor the validation loss (including regularization and auxiliary losses of the method) and allow for a patience period of 5 epochs. If no improvement is observed, we perform a learning rate decay step. Each decay step reduces the learning rate by a factor of 0.1, the model weights are reset to the best checkpoint before patience, and we do not perform more than 2 decay steps.

Evaluation. All scenarios are ranked by the average accuracy [8, 27, 33, 36, 47] achieved over the complete task sequence. Average accuracy is calculated by evaluating the model on the CIFAR test set based on the classes that have been seen up to each task. Complementary to the average accuracy, average *forgetting* [6, 8, 27] is also reported, which measures the drop in accuracy over the task sequence. Experimental results are averaged over 5 seeds.

5. Results

The summarized results of the experiments are listed in Table 3. Detailed plots and tables of the accuracy progression for all methods in each proposed scenario are provided in the supplementary material (Sec. G).

Scenario (a) CIL 50/10. The conducted baseline experiment confirms the reported results from other works [8, 40, 46, 47]. We observe a significant performance gain of approximately $10\text{-}15\%$ in average accuracy over the task

	(a) CIL 50/10		(b) EFCIR-U 50/100		(c) EFCIR-B 50/100	
Method	**Avg. A ↑**	**Avg. f ↓**	**Avg. A ↑**	**Avg. f ↓**	**Avg. A ↑**	**Avg. f ↓**
Joint	73.9	-	69.8	-	68.9	-
WA [45]	42.7 ± 2.3	33.2 ± 1.4	50.4 ± 0.2	16.7 ± 2.4	49.2 ± 0.7	18.0 ± 1.6
FT	14.2 ± 1.0	57.8 ± 1.2	36.2 ± 2.1	25.6 ± 2.7	34.2 ± 2.0	29.0 ± 2.6
FZ	52.6 ± 1.4	19.7 ± 0.9	40.2 ± 3.9	20.0 ± 1.6	41.7 ± 3.1	22.5 ± 1.8
EWC [20]	45.9 ± 2.9	25.7 ± 1.4	47.7 ± 3.2	**13.5 ± 1.5** ●	45.5 ± 3.2	**17.8 ± 1.8** ●
MAS [2]	45.9 ± 2.9	25.8 ± 1.4	**49.3 ± 2.6** ●	**12.0 ± 1.8** ●	**47.2 ± 2.3** ●	**16.1 ± 2.1** ●
LwF [22]	47.9 ± 1.8	24.1 ± 0.8	45.7 ± 1.9	**15.9 ± 2.8** ●	43.5 ± 0.8	19.8 ± 4.1
PASS [47]	62.1 ± 1.9	14.1 ± 0.4	30.2 ± 2.0	35.3 ± 2.1	30.6 ± 1.4	38.3 ± 1.6
PRAKA [40]	**63.1 ± 2.5** ●	**11.8 ± 2.2** ●	43.1 ± 2.1	22.3 ± 2.3	42.7 ± 3.2	25.6 ± 1.8
IL2A [46]	54.2 ± 1.4	19.1 ± 1.3	26.3 ± 3.0	32.2 ± 2.9	27.2 ± 2.5	37.2 ± 1.7
SSRE [48]	53.0 ± 2.7	**13.0 ± 0.8** ●	29.2 ± 3.5	25.4 ± 2.1	26.5 ± 2.2	26.4 ± 2.1
FeTrIL [33]	61.4 ± 0.4	**13.6 ± 0.8** ●	46.5 ± 0.7	22.9 ± 0.7	46.9 ± 0.9	23.8 ± 1.2
$Horde_m$	**62.9 ± 1.2** ●	15.2 ± 0.7	**54.4 ± 0.7** ●	16.4 ± 1.5	**54.3 ± 0.4** ●	**17.7 ± 1.0** ●
$Horde_c$	**62.9 ± 1.2** ●	15.3 ± 0.6	**53.4 ± 0.7** ●	17.6 ± 1.6	**53.1 ± 0.4** ●	18.5 ± 1.1

Table 3. Average Accuracy (Avg. A) and average Forgetting (Avg. f) for all 3 proposed scenarios. The listed results are averaged over 5 seeds (except incremental Joint). The 3 best results are marked with a **gold** ●, **silver** ● and **bronze** ● medal respectively.

sequence when comparing the state-of-the-art rehearsal-free (EFCIL) methods with EWC, MAS and LwF. While our proposed method is particularly designed towards repetition scenarios, where the estimation of class prototype components is not always required, it remains competitive in disjoint, no-repetition scenarios as well, showing comparable performance to the best EFCIL methods [33, 40, 46, 47].

Scenario (b) EFCIR-U. Introducing class repetition in small incremental tasks into the scenario leads to significant performance differences. Weight-regularization approaches and vanilla finetuning typically underperform compared to knowledge distillation or class prototype-based approaches in EFCIL [40]. However, in this scenario with repetition, we observe greatly improved performance for FT, EWC and MAS. The results for these methods surpass even the results from the CIL 50/10 scenario by leveraging data repetition effectively (see Fig. 3). In contrast, EFCIL methods (PASS, IL2A, SSRE, PRAKA) that rely on both class prototype rehearsal and knowledge distillation show a performance degradation under repetition. This decline is not observed in methods that use either knowledge distillation (LwF) or class prototype rehearsal with frozen feature extractors (FeTrIL, Ours).

We hypothesize that the estimation of class prototypes with incomplete class data distribution in the former methods leads to a suboptimal feature embedding space, which is then propagated through the incremental task sequence via knowledge distillation. Frozen feature extractors, on the other hand, avoid this issue since their representations remain fixed after the initial training, preventing catastrophic drift in the embedding space during the task sequence. This raises the question whether the assumption that the complete training data distribution of an individual class – as in traditional class-incremental learning – is a realistic assumption for continual learning scenarios.

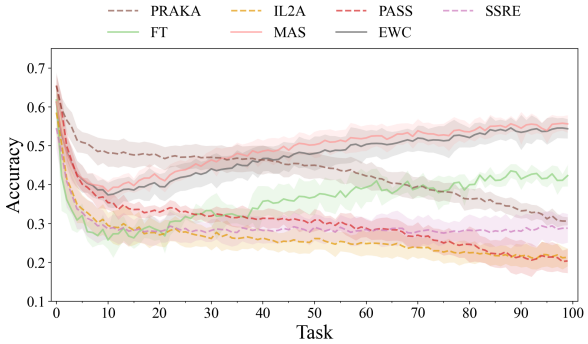
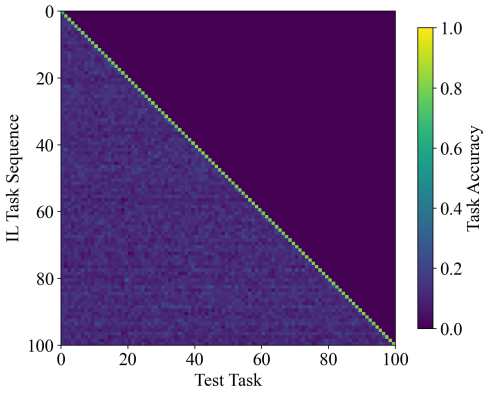


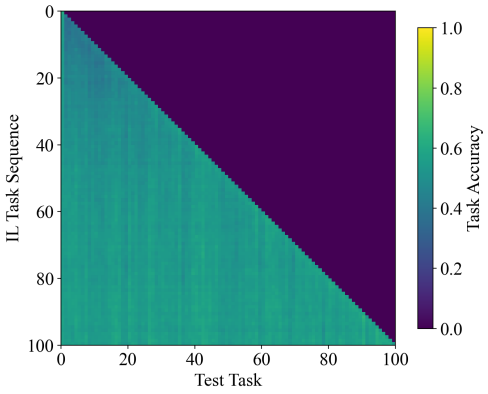
Figure 3. Accuracy curves for scenario (b) with equiprobable repetition frequency. Weight-regularized methods (*solid*) benefit directly from short tasks with class repetition, while prototype-based approaches (*dashed*) degrade in accuracy as the sequence advances.

Our ensemble-based approach (Horde), with both ensemble growth heuristics, establishes a new state-of-the-art for the repetition settings. Notably, when comparing our method with the closely related FeTrIL approach, we observe a performance increase under repetition. This suggests that our approach could extend the feature space of the base feature extractor by incorporating class combinations from smaller feature extractors. Over time, repetition aligns these representations, enabling the model to learn a unified classification head on a more diverse representation space provided by the ensemble.

The strong performance gains for weight-regularized approaches are only observed when the cross-entropy loss during training is limited to the classes that are present within the current task. Practically, this is achieved by freezing all weights associated with classes outside of the current task [27]. Figures 4 and 5 illustrate the consequences of backpropagating the loss through all weights of the classification head. In this case, regardless of



(a) Cross-entropy loss gradient applied to all class weights in the classification head. Significant task recency bias is visible (the diagonal is significantly higher, with sharp drops after each task).



(b) Cross-entropy loss gradient applied to only current task class weights in the classification head. Much of the task recency bias is alleviated by freezing the classifier weights for unavailable classes.

Figure 4. Depiction of the task accuracy progression of MAS over the scenario (b) sequence (averaged over 5 seeds). Accuracy is evaluated on the test set for the classes represented in the corresponding incremental training data within a task. Note, that for repetition there is always a certain overlap within tasks.

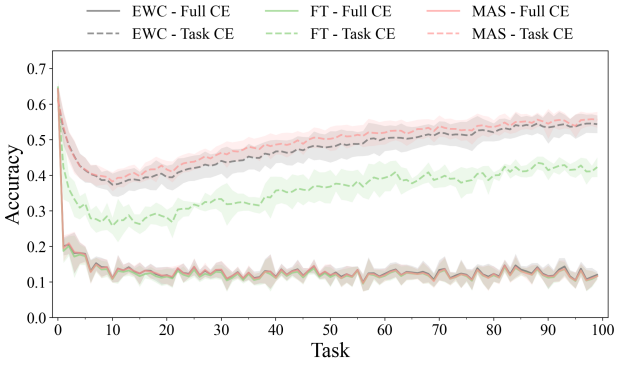


Figure 5. Accuracy results for finetuning and weight-regularization based methods. Solid lines indicate the backpropagation of the cross-entropy loss over all classes leading to catastrophic class recency bias. Dashed lines indicate the freezing of the weights related to the output of non-current classes.

whether weight-regularization is applied to the feature extractor, a strong class recency bias emerges in the classification head. As a result, the accuracy of all three methods collapses, with the model essentially forgetting classes proportionally to how long they were seen last (see Fig. 4). However, when the weights for unavailable classes in the classification head are frozen, a significant performance improvement is observed, as the model retains its ability to distinguish across earlier tasks without being overly biased towards the most recent ones.

Scenario (c) EFCIR-B. The bias in repetition frequency appears to have only a minor effect on the average accuracy of the approaches. All tested methods achieve similar results or experience only a slight drop of up to $\sim 2\%$ in average accuracy. This suggests that repetition frequency bias is a relatively minor challenge in the EFCIR-B 50/100 scenario. However, it is important to note that this setting only evaluates adjustments in repetition frequency while the sample distribution within a training task is kept balanced. Therefore, further investigation is needed to assess whether an imbalanced training data distribution in conjunction with biased repetition frequency would increase the difficulty. We leave this exploration to future work.

6. Conclusion

In this work, we conducted an exploratory evaluation of CIL methods in exemplar-free class-incremental learning with repetition scenarios and investigated their resiliency to biases in the repetition frequency of classes.

In the evaluated repetition scenarios, EFCIL methods that rely on class prototypes (PASS, PRAKA, IL2A, SSRE) severely underperform and are unable to benefit from the repetition of classes. Notably, weight-regularization-based approaches perform exceptionally well in repetition scenarios provided that training with cross-entropy is restricted to the classes present in each task, thereby mitigating the risk of class-recency bias in the classification head. The results from the repetition frequency bias from a beta distribution show only minimal performance differences, with either no effect on average accuracy or a slight drop of up to 2%. Thus, a bias in repetition frequency alone without a biased sample distribution within a training task is insufficient for significant classification bias.

Furthermore, we introduce a novel ensemble learning technique that takes advantage of class repetition. This method combines a dynamic set of independent feature extractors, which are aligned through a unified head in a process we call pseudo-feature projection. The proposed method demonstrates competitive performance in traditional no-repetition settings and establishes a new state-of-the-art for scenarios with repetition.

Acknowledgements

Marc Masana acknowledges the support by the "University SAL Labs" initiative of Silicon Austria Labs (SAL).

References

[1] 5th CLVISION CVPR workshop challenge, 2023. 2

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018. 2, 5, 6, 7, 12

[3] Eden Belouadah, Adrian Popescu, Umang Aggarwal, and Léo Saci. Active class incremental learning for imbalanced datasets. In *European Conference on Computer Vision*, pages 146–162. Springer, 2020. 1

[4] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54, 2021. 1, 3

[5] Christopher M. Bishop. Regularization and complexity control in feed-forward networks. International Conference on Artificial Neural Networks ICANN'95., 1995. 6

[6] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018. 6

[7] Andrea Cossu, Gabriele Graffieti, Lorenzo Pellegrini, Davide Maltoni, Davide Bacciu, Antonio Carta, and Vincenzo Lomonaco. Is class-incremental enough for continual learning? *Frontiers in Artificial Intelligence*, 5:829842, 2022. 1, 2

[8] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. 1, 2, 4, 5, 6

[9] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018. 1

[10] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1, 2

[11] Mudasir A Ganaie, Minghui Hu, AK Malik, M Tanveer, and PN Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115: 105151, 2022. 2

[12] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013. 1, 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6, 11

[14] Lu Yu He Li and Wu He. The impact of gdpr on global technology development. *Journal of Global Information Technology Management*, 22(1):1–6, 2019. 1

[15] Hamed Hemati, Andrea Cossu, Antonio Carta, Julio Hurtado, Lorenzo Pellegrini, Davide Bacciu, Vincenzo Lomonaco, and Damian Borth. Class-incremental learning with repetition. *arXiv preprint arXiv:2301.11396*, 2023. 2

[16] Hamed Hemati, Lorenzo Pellegrini, Xiaotian Duan, Zixuan Zhao, Fangfang Xia, Marc Masana, Benedikt Tscheschner, Eduardo Veas, Yuxiang Zheng, Shiji Zhao, Shao-Yuan Li, Sheng-Jun Huang, Vincenzo Lomonaco, and Gido M. van de Ven. Continual learning in the presence of repetition. *Neural Networks*, 183:106920, 2025. 2

[17] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 12

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[19] Norman L Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous univariate distributions, volume 2*. John wiley & sons, 1995. 6

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 2, 5, 6, 7, 12

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[22] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 2, 5, 7, 12

[23] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2262–2268. IEEE, 2018. 6

[24] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017. 1

[25] Marc Masana, Idoia Ruiz, Joan Serrat, Joost van de Weijer, and Antonio M Lopez. Metric learning for novelty and anomaly detection. In *British Machine Vision Conference (BMVC)*, 2018. 3

[26] Marc Masana, Tinne Tuytelaars, and Joost van de Weijer. Ternary feature masks: continual learning without any forgetting. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2021. 2

[27] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: Survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2023. 1, 2, 5, 6, 7

[28] Benjamin Maschler, Thi Thu Huong Pham, and Michael Weyrich. Regularization-based continual learning for anomaly detection in discrete manufacturing. *Procedia CIRP*, 104:452–457, 2021. 54th CIRP CMS 2021 - Towards Digitalized Manufacturing 4.0. 1

[29] Martial Mermillod, Aurélia Bugaiska, and Patrick BONIN. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4, 2013. 1, 2

[30] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019. 1

[31] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113: 54–71, 2019. 1

[32] Grégoire Petit, Adrian Popescu, Eden Belouadah, David Picard, and Bertrand Delezoide. Plastil: Plastic and stable memory-free class-incremental learning. In *Second Conference on Lifelong Learning Agents (CoLLAs)*, 2023. 2

[33] Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3911–3920, 2023. 2, 3, 4, 5, 6, 7, 12

[34] Simon J.D. Prince. *Understanding Deep Learning*. MIT Press, 2023. 3, 6

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1

[36] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 2, 6

[37] Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 3

[38] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 2

[39] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pages 4528–4537. PMLR, 2018. 1

[40] Wuxuan Shi and Mang Ye. Prototype reminiscence and augmented asymmetric knowledge aggregation for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1772–1781, 2023. 2, 5, 6, 7, 12

[41] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 1, 2

[42] Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1): 4069, 2020. 1

[43] Eli Verwimp, Rahaf Aljundi, Shai Ben-David, Matthias Bethge, Andrea Cossu, Alexander Gepperth, Tyler L Hayes, Eyke Hüllermeier, Christopher Kanan, Dhireesha Kudithipudi, et al. Continual learning: Applications and the road forward. *arXiv preprint arXiv:2311.11908*, 2023. 1

[44] Dejie Yang, Minghang Zheng, Weishuai Wang, Sizhe Li, and Yang Liu. Recent advances in class-incremental learning. In *Image and Graphics*, pages 212–224, Cham, 2023. Springer Nature Switzerland. 2

[45] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao 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. 5, 7, 12

[46] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, 34: 14306–14318, 2021. 2, 5, 6, 7, 12

[47] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5871–5880, 2021. 2, 3, 5, 6, 7, 11, 12

[48] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9296–9305, 2022. 2, 5, 7, 12

DOI: 10.3217/978-3-99161-022-9-004

Incremental Learning with Repetition via Pseudo-Feature Projection

Supplementary Material

A. Pseudo Code

The algorithm for the proposed Horde method can be separated into two parts: (1st) the training of an individual feature extractor (FE), which is listed in Algorithm 1 and (2nd) the overall assembly of the ensemble and training of the unified head through pseudo-feature projection (see Algorithm 2). The training of a feature extractor (1st part) can be freely adjusted (loss, network architecture) as long as a frozen feature extractor that can produce an embedding is the result.

Algorithm 1 FE Training

Initialize CE and ML Head
Initialize new FE (or transfer learned weights)
for training epoch **do**
  for $\boldsymbol{X}; \boldsymbol{Y}$ in Dataloader **do**
    $\boldsymbol{X}; \boldsymbol{Y} \leftarrow \text{SelfSupervision}(\boldsymbol{X}; \boldsymbol{Y})$
    Extract $\hat{\boldsymbol{X}} \leftarrow \text{FE}(\boldsymbol{X})$
    Predict $\hat{\boldsymbol{Y}} \leftarrow \text{Head}_{\text{CE}}(\hat{\boldsymbol{X}})$
    Project $\boldsymbol{A} \leftarrow \text{Head}_{\text{ML}}(\hat{\boldsymbol{X}})$
    Calculate $\mathcal{L}_{\text{CE}}$ (from $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$)
    Calculate $\mathcal{L}_{\text{ML}}$ (with Hard Neg. Pairs on $\boldsymbol{A}$)
    Backprop $\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{ML}}$
  end for
end for
Remove CE and ML head
Freeze FE

Algorithm 2 IL through pseudo-feature projection

for task **do**
  if Growth Condition (Horde$_{\boldsymbol{m}}$ or Horde$_{\boldsymbol{c}}$) **then**
    Train FE (Algorithm 1)
    Add / Replace FE in ensemble
  end if
  Calculate $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c$ for all current classes c
  for training epoch **do** ▷ Only Unified head
    for Batch **do**
      Calculate $\mathcal{L}_{\text{CE}}$
      Generate $\hat{F}_c$ from current Batch
      Calculate $\mathcal{L}_{\text{CE;P}}$ for $\hat{F}_c$
      Backprop $\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{CE;P}}$
    end for
  end for
end for

B. Details about the Model Architecture

The individual layers of a ResNet-18 are listed in Table 4 and a structural overview is depicted in Figure 6. There is no difference in the depth of the network or the type of layers between the ResNet-18 and the Slim-ResNet-18. The only difference is the number of base filters C_b for the convolutions in the *Basic Blocks*. The Slim-ResNet-18 uses $C_b = 20$ while the full ResNet-18 uses $C_b = 64$. This influences the number of channels for the following operations so that a compression to approximately a tenth of the original size can be achieved.

ResNet-18		
$C_b = 20$ for SlimResNet18 and $C_b = 64$ for ResNet-18		
Layer	**Stride**	**Dimension**
Conv 3×3	1	$C_b \times 32 \times 32$
BatchNorm	-	$C_b \times 32 \times 32$
ReLU	-	$C_b \times 32 \times 32$
BasicBlock $C_{in} = C_b, C_{out} = C_b$	1	$C_b \times 32 \times 32$
BasicBlock $C_{in} = C_b, C_{out} = 2 \cdot C_b$	2	$2 \times C_b \times 16 \times 16$
BasicBlock $C_{in} = 2 \cdot C_b, C_{out} = 3 \cdot C_b$	2	$3 \times C_b \times 8 \times 8$
BasicBlock $C_{in} = 3 \cdot C_b, C_{out} = 4 \cdot C_b$	2	$4 \times C_b \times 4 \times 4$
AvgPool 4×4	1	$4 \times C_b \times 1 \times 1$
Linear (Classification Head)	-	#classes

Table 4. The network structure is identical for both ResNet-18 and its SlimResNet-18 variant, besides a reduction in the number of base channels C_b.

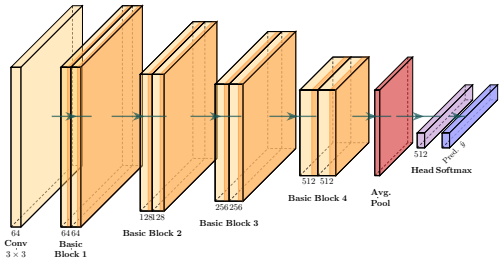


Figure 6. Visualization of the structure of a ResNet-18 [13].

C. Method Hyperparameters

The hyperparameters for all compared methods are listed in Table 5. For EWC, MAS and LWF, we perform a grid-search over their main hyperparameters on the CIL 50/10 scenario, and the one achieving the highest average accuracy are fixed for the repetition tasks. The remaining hyperparameters are the ones recommended by their original authors for the corresponding scenario.

D. Feature Extractor Training Components

Table 6 provides an overview of the effects of each component in the Feature Extractor and its effect on the average accuracy in the CIL scenario (a). The results have been averaged over 5 seeds. Both the self-supervision from PASS [47] as well as the training with the metric learning head are beneficial based on the overall average accuracy. The metric learning head alone without a cross-

Method	Hyperparameter
FT	-
FZ	freeze after 1st task
Joint	-
WA [45]	2000 exemplars, $\tau = 2$, $patience = 10$
EWC [20]	$\lambda = 40000$, $\alpha = 0.1$
MAS [2]	$\lambda = 10$, $\alpha = 0.1$
LwF [22]	$\lambda = 30$, $\tau = 2$
PASS [47]	$\tau_{\text{CE}} = 0.1$, $\tau_{KD} = 2$, $\lambda_{kd} = 10.0$, $\lambda_{aug} = 10.0$
IL2A [46]	$\tau_{\text{CE}} = 0.1$, $\lambda_{KD} = 10.0$, $\lambda_{seman} = 10.0$, $\#mixups = 4$
PRAKA [40]	$\tau_{\text{CE}} = 0.1$, $\lambda_{aug} = 15.0$, $\lambda_{KD} = 15.0$
SSRE [48]	$\tau_{\text{CE}} = 0.1$, $\lambda_{aug} = 10.0$, $\lambda_{KD} = 1.0$
FeTrIL [33]	AugMix [17] pre-train, fc head, 1-cosine translation
Horde (ours)	original features estimation, CE & ML Head, self-supervision, 1 Resnet18, 9 Slim Resnet18s

Table 5. Overview of approach-specific hyperparameters

CE-Head	ML-Head	Self-Supervision [47]	Avg. Acc ↑
✓	✗	✗	56.91 ± 0.88
✗	✓	✗	7.72 ± 0.85
✓	✓	✗	58.68 ± 0.79 ●
✓	✗	✓	60.62 ± 1.21 ●
✗	✓	✓	9.76 ± 1.15
✓	✓	✓	63.09 ± 1.19 ●

Table 6. Ablation study results on different variations of FE training. **1st** ●, **2nd** ● and **3rd** ● best metrics are marked accordingly.

entropy head is however insufficient for the training of a feature extractor.

E. Scenario Visualization

In the proposed experiments we differentiate between a fairly balanced repetition scenario and a biased scenario. The difference between the two repetition frequencies is visualized in Fig. 7 and Fig. 8. On average both scenarios have 15 classes in each incremental task.

F. Longer Task Sequence

The results from scenario (b) indicate a strong accuracy recovery/trend for weight regularization techniques. We further evaluate with even longer task sequences where the number of incremental tasks is increased from 99 to 149. The accuracy on later tasks is very strong on weight-regularization techniques as the overall accuracy trend continues. However, it is important to note that, already in the 100 task scenario, all available training data is used in the task sequence at least once, thus further tasks can only repeat samples and no longer provide any new/incremental training data. Although EWC and MAS both achieve a significant higher final accuracy in the longer task sequence, they are still slightly worse in terms of average accuracy across the whole sequence,

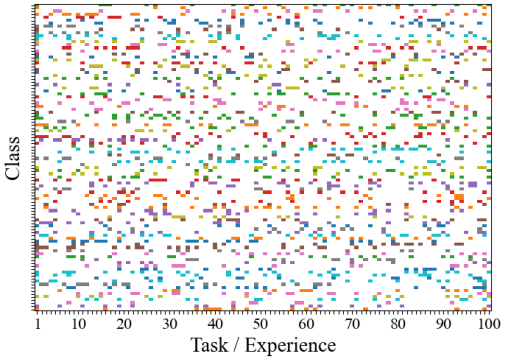


Figure 7. Class distribution visualization of scenario (b), with uniform class occurrence frequency. Each colored block indicates that the class is sampled in the corresponding task.

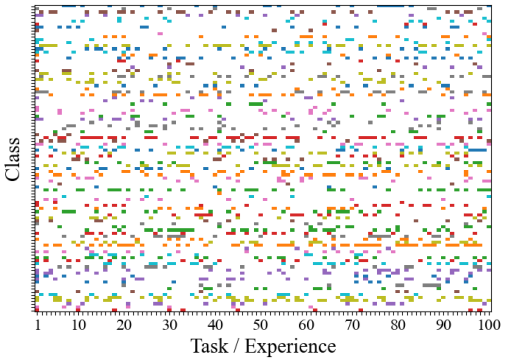


Figure 8. Class distribution visualization of scenario (c), with biased (beta) class occurrence frequency. Each colored block indicates that the class is sampled in the corresponding task.

since they are less stable in the initial tasks of the sequence. The compared average accuracies for the 100 and 150 task scenarios, as well as final test accuracy after the task sequence, are listed in Table 7. Furthermore, the accuracy progression is visualized in Figure 9.

Method	Avg. A_{100}	Avg. A_{150}	final A_{100}	final A_{150}
FT	36.2 ± 2.1	39.3 ± 2.1	42.3 ± 2.7	46.9 ± 1.1
EWC	47.7 ± 3.2	51.4 ± 0.9	54.4 ± 2.5 ●	57.2 ± 0.7 ●
MAS	49.3 ± 2.6 ●	52.5 ± 0.8 ●	55.6 ± 2.2 ●	59.0 ± 0.3 ●
$Horde_c$	54.4 ± 0.7 ●	53.2 ± 1.6 ●	55.1 ± 0.7 ●	54.4 ± 0.4 ●
$Horde_m$	53.4 ± 0.7 ●	53.8 ± 0.9 ●	54.0 ± 1.1	53.4 ± 1.9

Table 7. Comparison between unbiased repetition scenarios of 100 and 150 tasks. While our proposed method is more stable, especially in the initial phases of training. The trend of weight regularization methods continues and the final accuracy continues to increase.

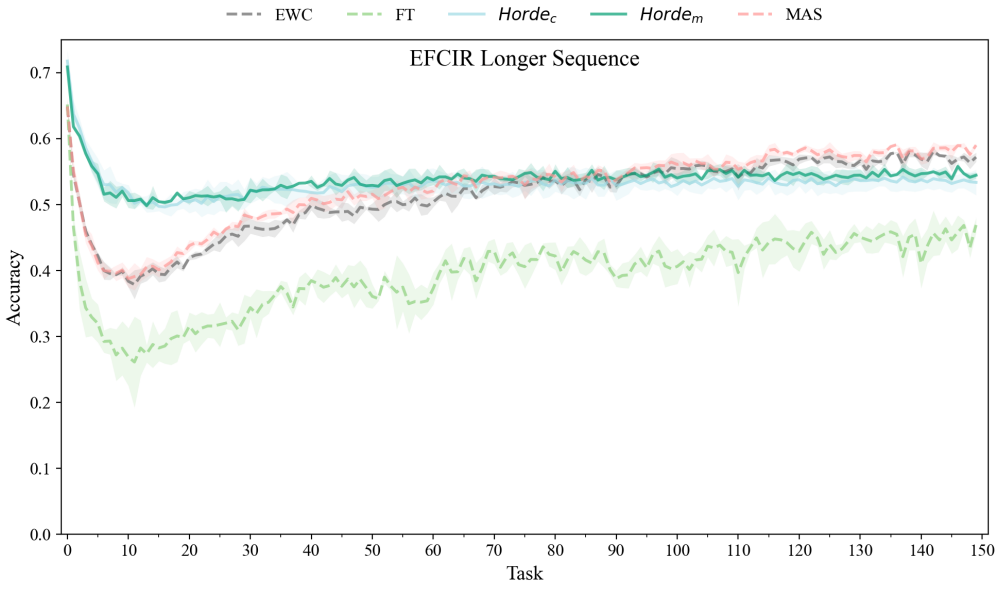


Figure 9. Average Accuracy over an even longer repetition scenario to analyse the trends between different methods. The performance increase of weight-regularization techniques continue

G. Scenario Results

The following figures visualize the detailed Average Accuracy development over the incremental task sequence. For each method the mean and one standard deviation have been plotted. The results for the class-incremental scenario (a) are listed in Table 8 and visualized in Figure 10. The unbiased repetition results fo scenario (b) can be found in Table 9 and Figure 11. The results of the biased class-repetition scenario (c) are shown in Table 10 and Figure 12.

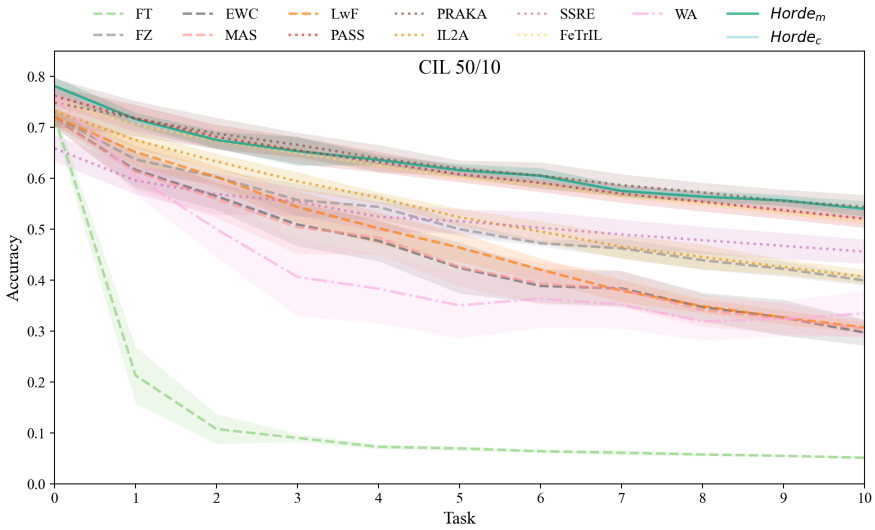


Figure 10. Accuracy development over the task sequence of scenario (a).

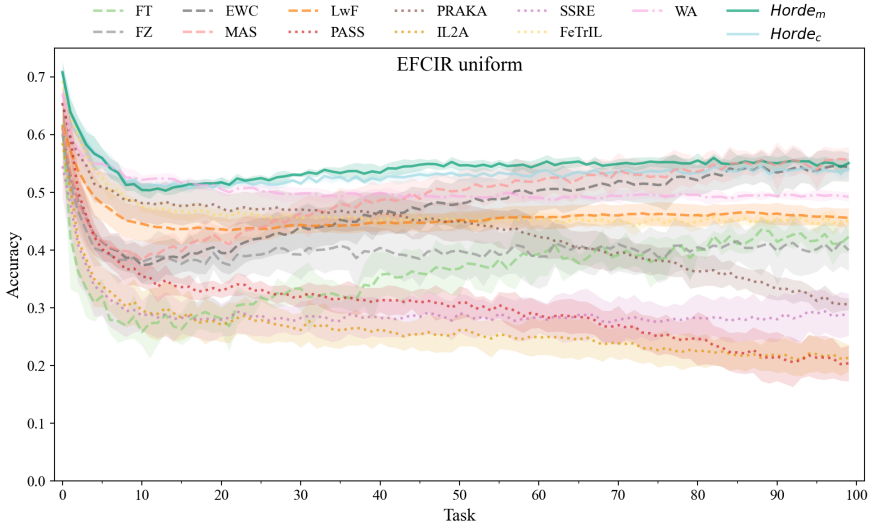


Figure 11. Accuracy development over the task sequence of scenario (b).

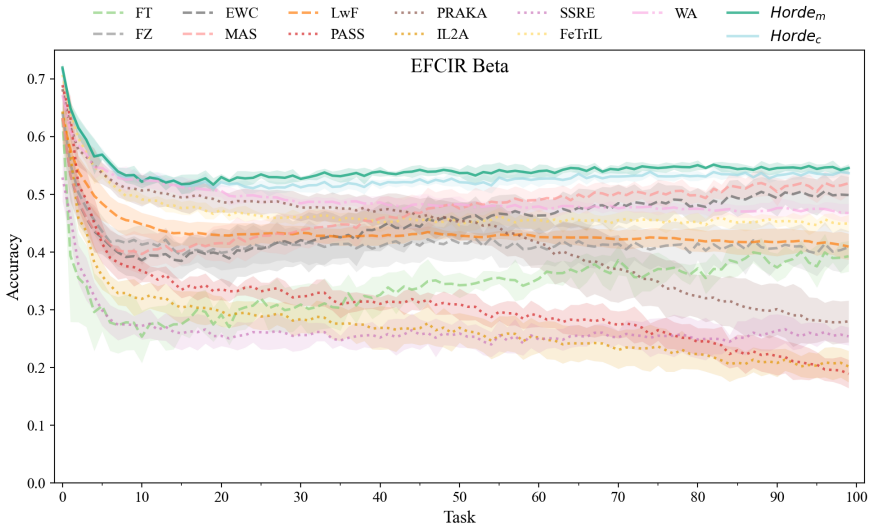


Figure 12. Accuracy development over the task sequence of scenario (c).

Method	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	Avg. $A\uparrow$	Avg. $f\downarrow$
WA	76.0 ± 1.5	60.4 ± 3.5	50.0 ± 5.4	40.7 ± 7.7	38.3 ± 6.7	35.1 ± 6.5	36.4 ± 5.7	35.2 ± 4.7	31.9 ± 3.7	32.4 ± 3.5	33.5 ± 4.5	42.7 ± 2.3	33.3 ± 1.5
FT	72.1 ± 1.7	21.3 ± 5.5	10.8 ± 2.9	9.0 ± 0.8	7.3 ± 0.4	7.0 ± 0.5	6.4 ± 0.3	6.1 ± 0.5	5.8 ± 0.3	5.5 ± 0.2	5.1 ± 0.3	14.2 ± 1.0	57.9 ± 1.2
FZ	72.1 ± 1.7	63.7 ± 2.7	60.3 ± 1.4	55.8 ± 3.2	54.4 ± 2.5	50.0 ± 1.4	47.3 ± 0.5	46.2 ± 1.9	43.9 ± 1.8	42.2 ± 1.5	39.9 ± 0.9	52.4 ± 1.4	19.7 ± 0.9
EWC	71.7 ± 1.6	61.6 ± 3.5	56.5 ± 2.6	50.9 ± 4.2	47.7 ± 4.0	42.4 ± 5.0	38.9 ± 3.3	38.4 ± 3.5	34.7 ± 2.8	32.6 ± 3.5	29.7 ± 2.5	45.9 ± 3.0	25.8 ± 1.5
MAS	71.7 ± 1.6	61.4 ± 3.6	55.9 ± 3.6	50.4 ± 5.3	48.4 ± 3.3	42.6 ± 3.8	39.4 ± 4.1	38.2 ± 3.6	34.1 ± 3.2	32.5 ± 3.0	30.5 ± 1.7	45.9 ± 2.9	25.8 ± 1.4
LwF	72.1 ± 1.7	65.2 ± 2.7	60.3 ± 2.0	54.4 ± 3.1	50.2 ± 3.1	46.4 ± 2.9	42.1 ± 2.2	37.9 ± 2.2	34.9 ± 1.3	32.7 ± 1.4	30.7 ± 1.1	47.9 ± 1.8	24.2 ± 0.8
PASS	**76.2 ± 2.0** ●	**71.8 ± 2.6** ●	**68.2 ± 2.3** ●	**65.5 ± 2.6** ●	63.2 ± 2.0	60.8 ± 1.5	59.1 ± 1.8	57.0 ± 1.7	55.4 ± 1.9	53.8 ± 1.7	52.1 ± 1.7	62.1 ± 1.9	14.1 ± 0.5
PRAKA	74.9 ± 4.7	**71.8 ± 3.4** ●	**68.8 ± 3.1** ●	**66.6 ± 2.4** ●	**63.9 ± 2.6** ●	**61.9 ± 1.6** ●	**60.5 ± 2.6** ●	**58.7 ± 2.1** ●	**57.2 ± 1.9** ●	**55.6 ± 2.2** ●	**54.4 ± 2.4** ●	**63.1 ± 2.6** ●	**11.8 ± 2.3** ●
IL2A	73.2 ± 0.8	67.5 ± 1.8	63.3 ± 1.5	59.4 ± 1.8	56.2 ± 1.1	52.4 ± 1.7	49.6 ± 2.1	46.6 ± 2.3	44.6 ± 2.4	42.6 ± 1.5	40.7 ± 1.4	54.2 ± 1.4	19.0 ± 1.3
SSRE	65.9 ± 2.6	59.6 ± 2.6	56.9 ± 4.0	55.4 ± 2.7	52.5 ± 2.9	51.6 ± 2.3	50.3 ± 3.2	49.0 ± 2.9	47.9 ± 2.7	46.7 ± 2.6	45.6 ± 2.4	52.9 ± 2.7	**13.0 ± 0.8** ●
FeTrIL	75.0 ± 1.2	70.6 ± 0.9	67.4 ± 0.9	64.9 ± 1.1	62.6 ± 0.6	60.2 ± 0.5	58.7 ± 0.5	56.5 ± 0.5	55.0 ± 0.5	53.2 ± 0.4	51.6 ± 0.6	61.4 ± 0.4	**13.6 ± 0.8** ●
$Horde_m$	**78.1 ± 1.7** ●	**71.6 ± 1.3** ●	67.5 ± 1.6	**65.3 ± 2.8** ●	**63.6 ± 1.0** ●	**61.6 ± 1.2** ●	**60.5 ± 1.5** ●	**57.5 ± 1.2** ●	**56.4 ± 1.1** ●	**55.7 ± 1.0** ●	**54.0 ± 1.5** ●	**62.9 ± 1.2** ●	15.2 ± 0.7
$Horde_c$	**78.2 ± 1.7** ●	71.2 ± 1.4	**67.7 ± 1.9** ●	65.2 ± 2.8	**63.4 ± 0.8** ●	**61.8 ± 1.2** ●	**60.2 ± 1.9** ●	**57.9 ± 1.0** ●	**56.6 ± 1.1** ●	**55.9 ± 1.1** ●	**53.8 ± 0.4** ●	**62.9 ± 1.2** ●	15.3 ± 0.7

Table 8. Results for the baseline CIL 50/10 scenario (a). **1st** ●, **2nd** ● and **3rd** ● best metrics are marked accordingly.

Method	A_0	A_{10}	A_{20}	A_{40}	A_{60}	A_{80}	A_{99}	Avg. $A\uparrow$	Avg. $f\downarrow$
WA	67.1 ± 2.4	52.2 ± 0.8	50.6 ± 1.1	49.9 ± 0.7	49.1 ± 0.8	49.6 ± 0.9	49.2 ± 0.8	50.4 ± 0.2	16.7 ± 2.4
FT	61.8 ± 4.3	25.8 ± 2.3	28.1 ± 2.2	35.7 ± 3.9	39.3 ± 3.8	40.0 ± 0.9	42.3 ± 2.7	36.2 ± 2.1	25.6 ± 2.7
FZ	60.1 ± 4.8	37.9 ± 3.7	37.9 ± 3.4	40.4 ± 3.5	38.9 ± 5.5	40.5 ± 3.6	40.0 ± 3.6	40.2 ± 4.0	20.0 ± 1.6
EWC	61.1 ± 4.1	37.4 ± 3.1	39.5 ± 3.3	46.7 ± 3.3	50.4 ± 3.5	52.1 ± 2.5	**54.4 ± 2.5** ●	47.7 ± 3.2	**13.5 ± 1.5** ●
MAS	61.3 ± 4.1	38.2 ± 2.6	41.6 ± 2.5	**48.6 ± 2.7** ●	**52.0 ± 3.3** ●	**53.6 ± 1.9** ●	**55.6 ± 2.2** ●	**49.3 ± 2.6** ●	**12.0 ± 1.8** ●
LwF	61.6 ± 4.2	44.5 ± 3.0	43.6 ± 2.2	44.7 ± 1.5	45.7 ± 2.0	46.1 ± 1.9	45.6 ± 1.7	45.7 ± 1.9	**15.9 ± 2.8** ●
PASS	65.5 ± 2.7	35.4 ± 1.8	32.9 ± 1.2	31.4 ± 2.1	28.8 ± 2.2	24.6 ± 4.6	20.4 ± 3.1	30.2 ± 1.9	35.3 ± 2.2
PRAKA	65.4 ± 3.3	**48.1 ± 3.5** ●	**47.2 ± 2.9** ●	46.4 ± 3.7	42.2 ± 3.0	36.3 ± 2.7	30.6 ± 1.0	43.1 ± 2.1	22.3 ± 2.4
IL2A	58.5 ± 5.8	29.5 ± 3.3	27.1 ± 3.0	26.3 ± 2.3	24.8 ± 3.0	22.5 ± 2.6	21.3 ± 2.3	26.3 ± 3.0	32.2 ± 2.9
SSRE	54.5 ± 5.3	28.7 ± 2.7	27.9 ± 2.8	28.2 ± 3.4	28.5 ± 2.8	28.3 ± 4.1	28.8 ± 3.7	29.2 ± 3.5	25.4 ± 2.1
FeTrIL	**69.3 ± 1.1** ●	47.5 ± 0.9	46.2 ± 1.0	45.2 ± 1.4	45.4 ± 1.1	45.0 ± 1.5	45.2 ± 0.3	46.5 ± 0.7	22.9 ± 0.7
$Horde_m$	**70.8 ± 1.7** ●	**50.4 ± 1.1** ●	**51.7 ± 1.0** ●	**53.4 ± 1.1** ●	**54.8 ± 1.1** ●	**55.5 ± 1.3** ●	**55.1 ± 0.7** ●	**54.4 ± 0.7** ●	16.4 ± 1.5
$Horde_c$	**70.9 ± 1.9** ●	**50.9 ± 1.0** ●	**51.7 ± 0.9** ●	**52.1 ± 0.7** ●	**53.6 ± 1.2** ●	**53.4 ± 1.4** ●	54.0 ± 1.1	**53.4 ± 0.7** ●	17.6 ± 1.6

Table 9. Results for the EFCIR-U scenario (b). **1st** ●, **2nd** ● and **3rd** ● best metrics are marked accordingly.

Method	A_0	A_{10}	A_{20}	A_{40}	A_{60}	A_{80}	A_{99}	Avg. $A\uparrow$	Avg. $f\downarrow$
WA	67.2 ± 1.8	52.4 ± 1.2	50.6 ± 1.1	48.6 ± 1.1	47.9 ± 1.6	47.5 ± 0.8	46.8 ± 1.6	49.2 ± 0.7	18.0 ± 1.6
FT	63.2 ± 4.3	25.3 ± 4.5	29.2 ± 3.0	32.6 ± 3.2	35.4 ± 1.5	37.2 ± 0.9	39.3 ± 2.7	34.2 ± 2.0	29.0 ± 2.6
FZ	64.2 ± 4.3	41.3 ± 3.4	39.9 ± 3.0	41.0 ± 3.5	41.0 ± 3.2	41.5 ± 2.9	40.9 ± 2.8	41.7 ± 3.1	22.5 ± 1.9
EWC	63.2 ± 4.3	38.6 ± 3.5	39.6 ± 4.6	44.2 ± 4.1	46.3 ± 3.2	48.1 ± 3.0	49.9 ± 3.2	45.5 ± 3.2	**17.8 ± 1.8** ●
MAS	63.2 ± 4.3	39.7 ± 3.5	41.6 ± 3.1	45.9 ± 3.7	**49.0 ± 2.1** ●	**49.9 ± 1.2** ●	**51.9 ± 2.0** ●	**47.1 ± 2.3** ●	**16.1 ± 2.1** ●
LwF	63.2 ± 4.3	44.8 ± 2.0	42.9 ± 2.1	42.7 ± 1.6	42.6 ± 0.7	42.0 ± 1.9	41.0 ± 2.2	43.5 ± 0.8	19.8 ± 4.1
PASS	68.9 ± 2.0	36.7 ± 1.9	33.6 ± 1.9	31.0 ± 1.2	28.9 ± 1.9	24.4 ± 2.4	18.9 ± 2.4	30.6 ± 1.4	38.3 ± 1.6
PRAKA	68.2 ± 2.2	**50.7 ± 2.7** ●	**48.6 ± 1.7** ●	**47.3 ± 2.5** ●	41.6 ± 4.3	32.3 ± 3.7	28.0 ± 3.6	42.6 ± 3.2	25.6 ± 1.9
IL2A	64.4 ± 4.1	31.9 ± 2.4	29.7 ± 2.6	26.9 ± 3.8	25.2 ± 2.6	22.3 ± 2.6	20.2 ± 2.7	27.2 ± 2.5	37.2 ± 1.7
SSRE	52.9 ± 4.1	27.2 ± 2.4	25.5 ± 1.7	25.2 ± 1.6	24.9 ± 2.3	24.9 ± 3.2	25.4 ± 1.5	26.5 ± 2.2	26.4 ± 2.1
FeTrIL	**70.7 ± 1.5** ●	49.1 ± 0.9	47.4 ± 0.5	45.1 ± 1.6	45.5 ± 1.8	45.2 ± 1.6	45.0 ± 1.5	46.9 ± 0.9	23.8 ± 1.2
$Horde_m$	**72.0 ± 0.8** ●	**52.2 ± 1.1** ●	**53.0 ± 0.6** ●	**54.0 ± 0.9** ●	**54.0 ± 1.2** ●	**55.1 ± 0.8** ●	**54.6 ± 0.7** ●	**54.3 ± 0.4** ●	**17.7 ± 1.0** ●
$Horde_c$	**71.7 ± 0.8** ●	**52.5 ± 0.7** ●	**52.1 ± 0.7** ●	**51.7 ± 0.7** ●	**52.7 ± 1.4** ●	**53.2 ± 0.8** ●	**53.7 ± 0.4** ●	**53.1 ± 0.4** ●	18.5 ± 1.1

Table 10. Results for the EFCIR-B scenario (c). **1st** ●, **2nd** ● and **3rd** ● best metrics are marked accordingly.