Neural Networks 183 (2025) 106920

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

# Continual learning in the presence of repetition

Hamed Hemati [a,*,1], Lorenzo Pellegrini [b], Xiaotian Duan [c,d], Zixuan Zhao [c,d], Fangfang Xia [c,d], Marc Masana [e,f], Benedikt Tscheschner [e,g], Eduardo Veas [e,g], Yuxiang Zheng [h], Shiji Zhao [h], Shao-Yuan Li [h], Sheng-Jun Huang [h], Vincenzo Lomonaco [i], Gido M. van de Ven [j,*]

[a] Institute for Computer Science, University of St. Gallen, Rosenbergstrasse 30, St. Gallen, 9000, Switzerland
[b] Department of Computer Science, University of Bologna, Via dell'Università 50, Cesena, 47521, Italy
[c] The University of Chicago, 5801 S Ellis Ave, Chicago, 60637, United States
[d] Argonne National Laboratory, 9700 S Cass Ave, Lemont, 60439, United States
[e] Graz University of Technology, Rechbauerstraße 12, Graz, 8010, Austria
[f] TU Graz - SAL Dependable Embedded Systems Lab, Silicon Austria Labs, Graz, 8010, Austria
[g] Know-Center GmbH, Sandgasse 36/4, Graz, 8010, Austria
[h] MIIT Key Laboratory of Pattern Analysis and Machine Intelligence, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China
[i] Department of Computer Science, University of Pisa, Piano Secondo, Largo Bruno Pontecorvo 3, Pisa, 56127, Italy
[j] Department of Electrical Engineering, KU Leuven, Kasteelpark Arenberg 10, Leuven, 3001, Belgium

ARTICLE INFO

Dataset link: https://github.com/ContinualAI/clvision-challenge-2023, https://github.com/xduan7/clvis-chlg-2023, https://github.com/xduan7/clvis-chlg-2023

*Keywords:*
Continual learning
Class-incremental learning
Repetition
Competition

ABSTRACT

Continual learning (CL) provides a framework for training models in ever-evolving environments. Although re-occurrence of previously seen objects or tasks is common in real-world problems, the concept of *repetition* in the data stream is not often considered in standard benchmarks for CL. Unlike with the rehearsal mechanism in buffer-based strategies, where sample repetition is controlled by the strategy, repetition in the data stream naturally stems from the environment. This report provides a summary of the CLVision challenge at CVPR 2023, which focused on the topic of repetition in class-incremental learning. The report initially outlines the challenge objective and then describes three solutions proposed by finalist teams that aim to effectively exploit the repetition in the stream to learn continually. The experimental results from the challenge highlight the effectiveness of ensemble-based solutions that employ multiple versions of similar modules, each trained on different but overlapping subsets of classes. This report underscores the transformative potential of taking a different perspective in CL by employing repetition in the data stream to foster innovative strategy design.

## 1. Introduction

Designing systems that can learn and accumulate knowledge over time in non-stationary environments is an important objective in artificial intelligence research (Chen & Liu, 2018; Thrun & Mitchell, 1995; Verwimp et al., 2024). In traditional machine learning, however, the common practice is to build and train models based on statistical learning assumptions. Under these assumptions, a model has access to a static dataset with samples that are independent and identically distributed (IID). This implies that both the training set and test set come from the same distribution and remain unchanged throughout the training and evaluation processes. In reality, these assumptions are often violated, and the model is exposed to different forms of shift in the data. To address such shifts, continual learning (CL) offers a framework to simulate the "never-ending" learning setting (Mitchell et al., 2018). In CL, a model is exposed to a data stream consisting of a potentially unbounded sequence of experiences (De Lange et al., 2022; Lesort, Lomonaco, Stoian, Maltoni, Filliat, & Díaz-Rodríguez, 2020; Parisi, Kemker, Part, Kanan, & Wermter, 2019). Unlike under the statistical learning assumptions, in CL, the model does not have full access to a static dataset. Instead, it gets non-IID access to the data distribution in the form of a data stream, as data become partially available in an incremental manner. In a specific variant of CL, referred to as batch continual learning or task-based continual learning, the model gets *locally IID* access to part of the data distribution. These locally IID parts of the data stream have been referred to as tasks, contexts, or experiences.

* Corresponding authors.
*E-mail addresses:* hemati.hmd@gmail.com (H. Hemati), gido.vandeven@kuleuven.be (G.M. van de Ven).
[1] At the time of this research, the author was affiliated with the University of St. Gallen. The author is now affiliated with Stuttgart University of Applied Sciences.

https://doi.org/10.1016/j.neunet.2024.106920
Received 24 May 2024; Received in revised form 22 September 2024; Accepted 12 November 2024
Available online 20 November 2024
0893-6080/© 2024 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

A central objective in CL is to design models that mitigate *catastrophic forgetting* (McCloskey & Cohen, 1989; Ratcliff, 1990). Catastrophic forgetting refers to the phenomenon where a model's performance on a previously learned task rapidly and drastically declines as the model continues to train on data from other tasks. To tackle the issue of forgetting in neural networks, researchers have proposed various strategies. Some strategies add regularization terms to the loss to penalize changes to parts of the model important for past tasks (Kirkpatrick et al., 2017; Li & Hoiem, 2017; Zenke, Poole, & Ganguli, 2017). Other strategies replay samples from previous experiences by either employing an external buffer to store raw samples or by using a generative model to synthesize samples similar to those seen previously (Chaudhry et al., 2019; Rebuffi, Kolesnikov, Sperl, & Lampert, 2017; Rolnick, Ahuja, Schwarz, Lillicrap, & Wayne, 2019; Shin, Lee, Kim, & Kim, 2017; van de Ven, Siegelmann, & Tolias, 2020). While replay strategies often yield good performance, they come with certain drawbacks, such as data privacy concerns and high costs in terms of computation or memory, which can become prohibitive for extended streams. In another CL strategy, parameters are added to increase the capacity of the model to ease the learning of new tasks while avoiding interference with older tasks (Aljundi, Chakravarty, & Tuytelaars, 2017; Rusu et al., 2016; Xu & Zhu, 2018).

An aspect that is not often considered in the CL literature is the *repetition* of previously seen concepts over time. CL research often uses "academic scenarios" in which tasks, domains or classes are presented in a strictly sequential manner without repetition of previously seen ones (Hsu, Liu, Ramasamy, & Kira, 2018; van de Ven & Tolias, 2018). (The terms task-, domain- and class-incremental learning are sometimes even interpreted as excluding the possibility of repetition, but see van de Ven, Tuytelaars, and Tolias (2022).) However, in real-world problems, the re-occurrence of previously encountered concepts is common. For example, in a practical scenario where an agent explores a large environment consisting of many areas or rooms over time and learns about new concepts through observation, the agent may revisit similar concepts in different ways as it explores the world. Consequently, the repetition of concepts can naturally happen during the lifetime of the agent, which may lead to an improvement in understanding the concept as it re-appears in different contexts.

Adding the dimension of repetition to a CL problem raises questions such as '*Does repetition in a stream affect the behavior of CL strategies?*' and '*What type of CL strategies are better for streams with repetition?*'. It is possible that repetition impacts the performance of different CL strategies in different ways. For example, while replay is typically considered to be a very competitive approach for CL without repetition, it is conceivable that its relative effectiveness might be reduced when there is already naturally occurring repetition in the data stream. It is important to point out that there is a fundamental difference between repetition in the data stream and the replay of past experiences from a memory buffer. With replay, the strategy controls the re-occurrence of samples or concepts, and typically, samples from all previous experiences are replayed alongside new data. However, in data streams with repetition, it is the environment or the underlying generation factors of the stream that enforce the recurrence of a concept. Thus, the model might only be presented with a subset of the previously encountered concepts.

A number of recent works have started to empirically evaluate CL methods in data streams where repetition can happen. To better capture the temporal statistics of real-world data streams, authors have used very long data streams with many tasks in which past classes are reused (Lesort et al., 2022), data streams with blurry and stochastic task boundaries (Koh, Kim, Ha, & Choi, 2022; Moon, Park, Kim, & Park, 2023), and data streams modeled based on how infants interact with toys (Stojanov et al., 2019). The role and importance of repetition for learning have also been investigated from a neuroscience perspective. For instance, Zhan, Guo, Chen, and Yang (2018) explore how repetition can affect memory performance and brain activity for associative memory over time. Other studies focus on the spacing effect (Feng et al., 2019; Smith & Scarf, 2017), which concerns how spaced repetition can enhance memory retention compared to massed repetition, or on the Hebb repetition effect (Cho, Lee, Baek, & Paik, 2024), which involves the enhancement of memory after each repetition.

In recent years, several workshops on CL have organized challenges. For example, the first edition of the CLVision challenge at CVPR 2020 introduced new benchmarks based on the CORe50 dataset (Lomonaco et al., 2022). Although the 'New Instances and Classes' scenario (Maltoni & Lomonaco, 2019) that was used in that challenge includes some form of repetition of past classes, it lacks a principled approach to the study of the problem. In the 2021 edition of the CLVision challenge, the focus was mainly on supervised class-incremental classification and continual reinforcement learning (Normandin et al., 2021). In that same year, in a challenge at ICCV 2021, there was a track on domain-incremental object classification and detection based on a benchmark for autonomous driving called CLAD (Verwimp et al., 2023). Finally, the CLVision challenge at CVPR 2022 (Pellegrini et al., 2022) emphasized the design of strategies for object detection and classification using the large EgoObjects dataset (Zhu et al., 2023).

For the CLVision challenge at CVPR 2023, the primary goal was to explore the role of repetition in CL, a topic not often explored by the community. To do so, the participants were asked to design strategies that could exploit the repetition inherent in the stream to boost knowledge transfer and reduce forgetting without storing raw samples. This report provides a summary of this challenge. Details about the design of the challenge are provided in Section 2. Solutions proposed by finalist teams are presented in Section 3 to 5. Section 6 presents the results of the challenge, and Section 7 discusses the main lessons learned.

## 2. Challenge details

Participants in the challenge were asked to design strategies for a class of CL problems referred to as class-incremental with repetition (CIR). CIR encompasses a variety of streams with two key characteristics:

1. New classes can appear over time.
2. Previously encountered classes can re-occur with varying repetition patterns.

Since many existing strategies in the CL literature have only been tested in repetition-free settings, their efficacy and performance in the presence of repetition remain unclear. To investigate the influence of repetition in the data stream on the relative effectiveness of different strategies, a set of CIR benchmarks was generated using a sampling-based data stream generator controlled by four interpretable parameters (see Section 2.1). Participants were tasked with developing strategies that, upon completing training on the entire stream, could achieve high average accuracy on a test set containing unseen samples from the same classes present in the stream. To allow participants flexibility in strategy design without the need for significant computational resources, we used CIFAR-100 (Krizhevsky, 2009), a relatively small-scale dataset of natural images ($32 \times 32$ RGB images, 100 classes), as the base dataset for the stream generator.

The competition consisted of two phases:

*Pre-selection phase.* The pre-selection phase took place between the 20th of March 2023 and the 20th of May 2023. For this phase, there were three challenge data streams, which were released on the GitHub repository[2] of the challenge. The participants were asked to submit their solutions in the form of model predictions on the test set after training on each stream. Appendix B provides details on the participation over time in the pre-selection phase.

[2] https://github.com/ContinualAI/clvision-challenge-2023

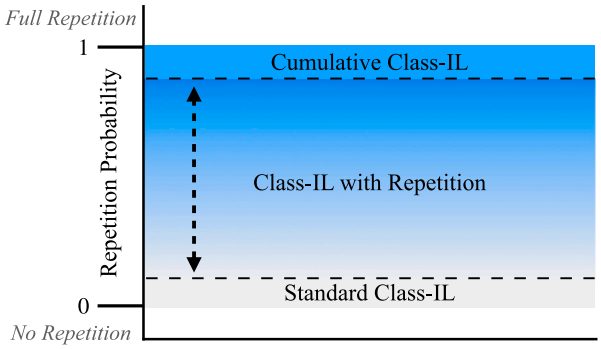



**Fig. 1.** Illustration of transitioning from "standard Class-IL" to "cumulative Class-IL" by increasing the probability of repetition for seen classes, which is set by control parameter $P_r$. Class-IL: class-incremental learning.

*Final evaluation phase.* The final phase started from the 20th of May 2023. In this phase, the five highest-ranking participants from the pre-selection phase were asked to send the source code of their strategies. Their strategies were tested on three new data streams, and the average accuracy on the test set after training on each of these three new streams was used as the metric for the final ranking. Participants were allowed to make changes to their strategies between the pre-selection phase and the final evaluation phase.

### 2.1. Stream generator

The sampling-based generator developed by Hemati et al. (2023) was used to generate the data streams for this challenge using four control parameters with clear interpretations:

- Stream length ($N$): Number of experiences in the stream.
- Experience size ($S$): Number of samples in each experience.
- First occurrence distribution ($P_f$): A discrete probability distribution over the experiences in the stream that determines the first occurrence of each class.
- Repetition probability ($P_r$): Per-class repetition probabilities that control the likelihood of each class re-appearing in future experiences in the stream after its first occurrence.

By default, an equal number of samples is assigned to all classes present in each experience. $P_f$, which controls the "first occurrence" property of the stream, determines when dataset classes make their initial appearance. For instance, in one stream, all classes might appear for the first time at the outset, whereas in another stream, new classes might appear gradually. Once a class has appeared for the first time, it re-appears based on a per-class repetition probability, which is controlled by $P_r$. Fig. 1 illustrates how increasing the repetition probability from 0 to 1 corresponds to a transition from "standard" class-incremental learning, where each class is present in only one experience, to "cumulative" class-incremental learning, where each class re-appears in each experience after its first occurrence. While the repetition probability can be dynamically set based on a time-varying probability mass function, in this challenge, the repetition probability for each class remains fixed over time. This means that, in this challenge, $P_r$ is a list of probability values, with one value assigned to each class in the dataset.

In Fig. 2, three different examples of streams generated using the generator are shown, with a dataset containing 30 classes and $N$ set to 50. In these examples, $P_f$ is varied while $P_r$ is kept constant to illustrate the effect of first occurrence on the generated streams.

*Selected challenge stream parameters.* The settings used for $P_f$ and $P_r$ in each of the challenge data streams are provided in Appendix C. The number of experiences per stream ($N$) was fixed as 50, and the number of samples per experience ($S$) was set to 2000. Following the default setting, the number of samples in each experience was equally distributed across the classes present in the experience.

### 2.2. Participation

Participation was in the form of teams. Team registrations and solution submissions were handled using the CodaLab platform (Pavao et al., 2023). Each team was allowed to create only one account on the platform to prevent parallel submissions. Participants were provided with a DevKit based on the Avalanche library (Carta, Pellegrini, Cossu, Hemati, & Lomonaco, 2023): https://github.com/ContinualAI/clvision-challenge-2023/. The DevKit contains all necessary scripts and codes to load the benchmarks for the challenge configurations. Researchers who are interested in evaluating the performance of their own strategies can use the DevKit for both pre-selection and final phase streams.

### 2.3. Rules and restrictions

**Submission:** Each submission in the pre-selection phase consisted of three sets of predictions. Each set of predictions was made by testing the model against the challenge's test set, after training the designed strategy on one of the three challenge streams.

**Strategy Design:** Within each experience in a data stream, users had full access to the training data from that experience, but not to the data from other experiences. In the default settings of the DevKit, the number of epochs per experience was set to 20. Participants had the flexibility to tweak and tailor the epoch iterations and dataset loading. For instance, a possible strategy would be to iterate for more epochs in the initial experiences and fewer in the final ones.

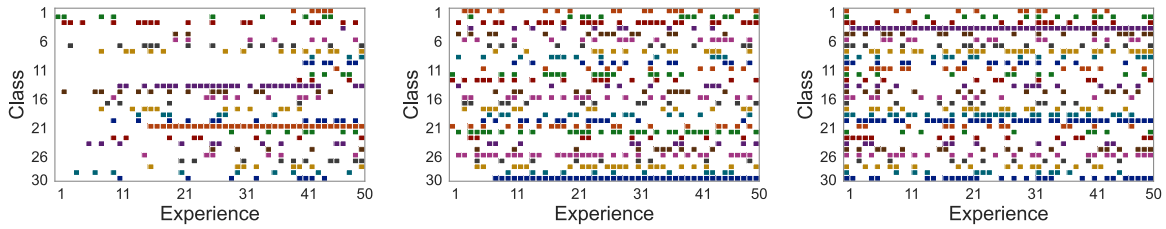
**Model Architecture:** All participants were required to utilize the (Slim-)ResNet-18 provided in the DevKit as the base architecture for their models. However, they were allowed to use multiple copies and to add additional modules, e.g. gating modules, provided they adhered to the maximum GPU memory usage (4000 MB) allowed for the competition during training. The GPU memory limit is set to almost eight times the memory size needed to train the Slim-ResNet-18 backbone with a batch size of 64. This allows for considerable flexibility in adding additional modules to the backbone.

**Replay Buffer:** Using a replay buffer to store dataset samples was not allowed. However, buffers could be used to store any form of data representation, such as the model's internal representations of data samples. Irrespective of the buffer type, the buffer size (i.e., the total number of stored exemplars) was not allowed to exceed 200.

**Hardware Limitations:** To encourage a comparison between strategies that use similar computational resources, participants were limited to using one GPU for training, and the maximum training time for each run was capped at 500 min. No computing limitations were specified for the inference time.

## 3. Strategy 1: HAT-CIR

The strategy proposed by team *xduan7* is called **HAT-CIR**. This strategy combines the strengths of network replicas and test-time decision-making, along with other elements, such as Hard Attention to the Task (HAT) (Serra, Suris, Miron, & Karatzoglou, 2018) and Supervised Contrastive Learning (SupCon) (Khosla et al., 2020).




(a) $P_f$ : Geom(0.01), $P_r$ : Zipf(0.5) (b) $P_f$ : Geom(0.3), $P_r$ : Zipf(0.5) (c) $P_f$ : Geom(0.9), $P_r$ : Zipf(0.5)

**Fig. 2.** Examples of generated streams with a Geometric distribution over the first occurrences of the dataset's classes, and a Zipfian distribution for the repetition of classes along the stream.

### 3.1. Motivation and related work

In CIR, as more generally with class-incremental learning, comparing model output logits across different classes is difficult because images from different classes are not jointly used for training. One potential solution is using a memory buffer and replaying samples to calibrate the logits, but due to restrictions on the storage of data, this approach is not well suited for this challenge. Another strategy involves parameter isolation. Parameter-isolation methods divide the neural network into segments, each responsible for a specific task or experience, preventing the network from forgetting knowledge acquired from prior experience. For example, HAT employs trainable binary masks to partition the network based on the experience identifier. This has proven effective, particularly in task-incremental learning problems where the experience identifier is provided during both training and test phases. However, the parameter-isolation strategy encounters challenges when applied to the CIR setting, where experience identifiers are unavailable during testing.

To address the experience identifier limitation in parameter-isolation methods within the CIR setting, OOD detection techniques can be used to enable "task-agnostic inference". Such techniques can detect samples that are not from the current experience, as for example demonstrated in Kim, Esmaeilpour, Xiao, and Liu (2022) and Kim, Liu, and Ke (2022). Another option is to use representation learning methods like SupCon to make the output logits more comparable across different classes. This has been explored in previous works (e.g., Kim, Esmaeilpour, et al., 2022) and has been shown to be effective when applied to the class-incremental learning setting.

In light of these existing works, and considering the specific challenges posed by the competition, the subsequent sections will elaborate on how to adapt and integrate these strategies to propose a novel solution for the CIR setting.

### 3.2. Method description

The proposed method comprises three core components: (i) structural design, featuring HAT-based partitioning and network replicas; (ii) a two-phase training strategy, including supervised contrastive learning and classification; and (iii) a momentum-based inference mechanism for test-time decision making. A schematic of the method is shown in Fig. 3.

#### 3.2.1. Structural design

*HAT-based partitioning.* To mitigate catastrophic forgetting, HAT isolates network parameters based on experience IDs. However, the original HAT method suffers from slow training and hyperparameter sensitivity when handling a large number of experiences. To overcome this, HAT-CL (Duan, 2023) is used, which initializes masks to ones and uses a cosine mask scaling curve to facilitate better alignment with network weights. By using the cosine mask scaling curve, each training epoch is divided into three phases:

1. Train weights while masks are mostly ones
2. Train masks and weights together to make masks more sparse
3. Fine-tune weights while masks are mostly ones again

These changes significantly improve the training speed and stability, and performance of HAT. Furthermore, the effect of the regularization term for the masks is gradually reduced. This step ensures full network capacity utilization and provides an accounting for the number of classes in each experience through variable regularization terms. It is important to note that the HAT-based partitioning was only used in the pre-selection phase; for the final phase, only network replicas were used, which led to higher performance.

*Network replicas.* Two types of network replicas, *fragments* and *ensembles*, are used to increase the model capacity and improve performance. Fragments refer to network replicas trained on different experiences. For example, if there are ten experiences, five fragments might be trained, each on two experiences. Ensembles are replicas trained on the same experiences. For each experience, multiple ensemble replicas will be trained on the same samples with different initializations and data augmentation. Thus, their predictions are averaged for final decision-making. These two replica types are complementary and can be combined with or without HAT-based partitioning. Importantly, adding fragment replicas does not increase the computational costs during training, while adding ensemble replicas does. Due to the challenge's restriction on memory usage and training time, the final version of the model used for the experiments employed 50 fragments and two ensembles. It is important to note that during the pre-selection phase, network replicas were not yet used.

#### 3.2.2. Two-phase training

To learn better feature representations, in each experience the network is first trained with supervised contrastive learning. The objective is to maximize the similarity between feature vectors of the same class and minimize those of different classes with the following loss function:

$$L = \sum_{i=1}^{N} \max\left(0, D(f(x_i^a), f(x_i^p)) - D(f(x_i^a), f(x_i^n)) + \alpha\right)$$

In this equation, $f(x)$ produces the embedded feature vector for input $x$, $D(x, y)$ denotes a distance function, $x_i^a$, $x_i^p$, and $x_i^n$ represent the anchor, positive, and negative samples, respectively, and $\alpha$ is a margin parameter. The number of samples in a batch is denoted by $N$. The training of hard attention masks occurs exclusively in this supervised contrastive learning phase, due to their sensitivity to learning rates and epoch numbers.

In the second training phase of each experience, the parameters of the network are trained further using a standard cross-entropy classification loss.

#### 3.2.3. Momentum-based test-time decision making

At test time, a predicted "score" for each possible class is calculated as a specific weighted average of the logits of that class from different model "snapshots", whereby a model snapshot is a fragment or a HAT-based partitioning. In particular, an average is taken over the logits

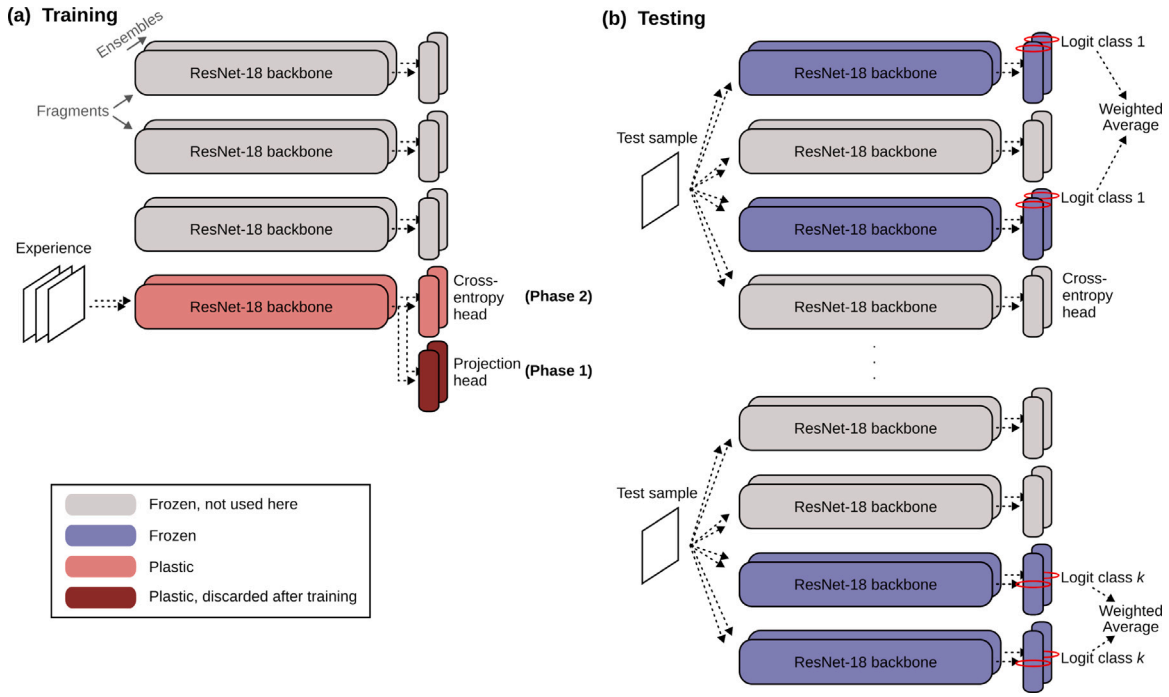



**Fig. 3.** Schematic of HAT-CIR during training and test time. HAT-CIR is illustrated here with network replicas and without HAT-based partitioning. **(a)** When training on a new experience, a new 'fragment' – consisting of multiple 'ensembles' – is added to the model and trained on the training data of the new experience in two phases. In the first phase, a projection head is used and a supervised contrastive loss is optimized; in the second phase, a softmax output layer is used and a cross-entropy loss is optimized. **(b)** During testing, a score for each possible class is computed as a weighted average of the logits from the most recent fragments that were trained on an experience in which that class appeared.

of the $N$ most recent model snapshots that were trained on an experience in which that class appeared. This proposed approach, called "momentum-based decision making", achieves strong performance in this challenge by setting $N = 3$, using the weights $\{1, 2, 3\}$ for the last three experiences in which each class appeared.

#### 3.2.4. Other details

Standard techniques such as data augmentation, early stopping, and learning rate scheduling were explored. Notably, it was found that data augmentation during both training and inference was crucial for improving performance. For the final submission, a combination of `RandomResizedCrop`, `RandomHorizontalFlip`, and `ColorJitter` transformations was used. Further details, consisting of an ablation study and an analysis of the influence of the number of network replicas, can be found in Appendix D. The implementation of HAT-CIR publicly available on GitHub: https://github.com/xduan7/clvis-chlg-2023/.

### 3.3. Discussion

In this section, the limitations and possible improvements for the method are critically assessed, as well as the implications of the findings for future research.

*Limitations.* A significant drawback of the approach arises when the number of classes is small in the initial experiences. In such cases, the network fails to learn effective class representations due to the limited diversity. This negatively impacts the performance of the momentum-based test-time decision-making strategy. Another inherent limitation is related to the use of HAT. The rigidity of the network structure requires careful hyperparameter tuning to match the expected total number of experiences, which is not always known in advance in CIR settings. As a result, suboptimal parameter allocation might be achieved, leading to compromised performance.

*Future directions.* The experiments show that the method's performance scales positively with the number of network replicas, implying that network capacity plays a significant role. Utilizing larger networks, possibly incorporating different structures with pre-trained weights, might offer avenues for further improvement in performance.

## 4. Strategy 2: Horde

The strategy proposed by team *mmasana* is called **Horde**. This strategy learns an ensemble of feature extractors (FEs) on selected experiences, which should provide robust features useful for discriminating between seen and unseen downstream classes. After training, the learned FEs are frozen to obtain a zero-forgetting ensemble. However, since not all classes are present in each experience, the outputs need alignment to balance them effectively. To do this, Horde uses pseudo-feature projection on the outputs to retain as much discrimination between classes as possible, while learning a unified head capable of discriminating between all classes seen so far. To further facilitate the pseudo-feature projection, FEs are trained with both the usual cross-entropy loss and an additional metric learning loss, which promotes alignment among the learned classes within each feature space.

### 4.1. Motivation and related work

An important motivation for the proposed approach is balancing the stability–plasticity dilemma (Mermillod, Bugaiska, & Bonin, 2013). This balance involves retaining useful knowledge from previous experiences while learning new ones from the sequence. For plasticity, robust and discriminative representations (Oquab, Bottou, Laptev, & Sivic, 2014) are important, as they help adapt to new classes and promote generalization (or forward transfer). Horde encourages the learning of such representations by combining a metric learning loss and the usual cross-entropy classification loss. Support for stability can be provided by zero-forgetting methods (Masana et al., 2023), which freeze the feature extractor part of model after training (Aljundi et al., 2017; Rajasegaran,

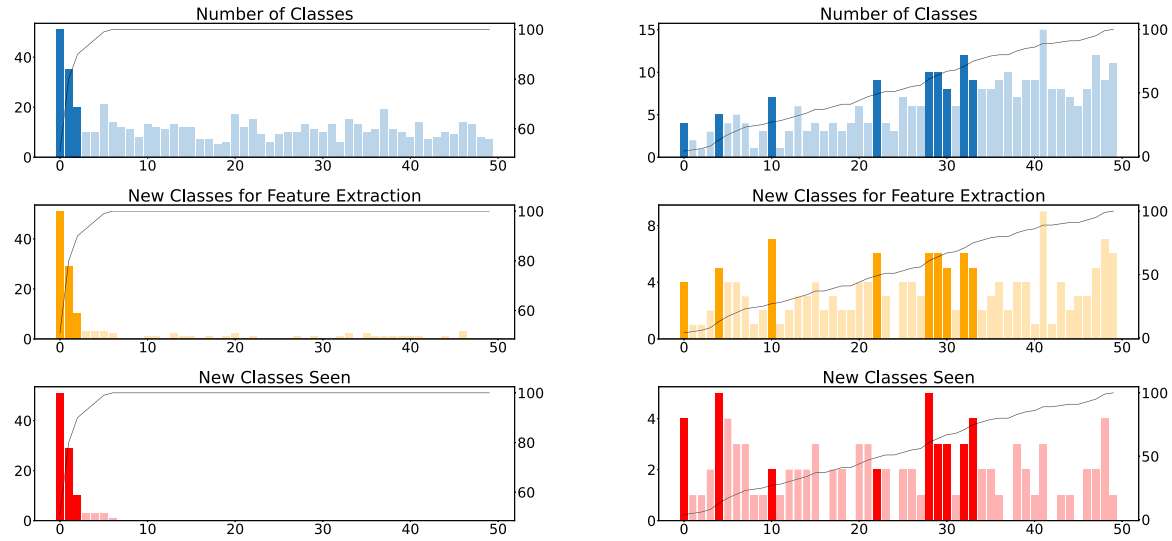



**Fig. 4.** The panels indicate for stream 1 (left) and stream 2 (right) in which experiences Horde decided to add a new FE (indicated by bright colors), while comparing the experiences based on three factors: the total number of classes in the experience (blue), the number of classes in the experience on which no FE had been trained yet (yellow), and the number of new, unseen classes in the experience (red). The black line represents the total amount of classes seen so far.

Hayat, Khan, Khan, & Shao, 2019; Rusu et al., 2016) or apply masks to the parameters or the intermediate representations (Fernando et al., 2017; Mallya & Lazebnik, 2018; Masana, Tuytelaars, & van de Weijer, 2021). The set up of the challenge excludes the use of the experience ID at test time and the use of pretrained models, both of which limit the direct application of existing zero-forgetting methods. However, the enforcement of the stability component via freezing is preserved for Horde and represented with the use of an ensemble of FEs that retain all learned knowledge.

Initial inspiration for Horde is provided by the existing methods Feature Translation for Exemplar-Free Class-Incremental Learning (FeTrIL) (Petit, Popescu, Schindler, Picard, & Delezoide, 2023) and Ternary Feature Masks (TFM) (Masana et al., 2021). FeTrIL uses a pretrained frozen backbone to benefit from the stability of having zero-forgetting for that part of the model, while the plasticity is introduced via the learning of a unified head. This method has a pseudo-feature generator that uses the representations from a single FE and applies a geometric translation to align both past and new classes. The pretrained backbone could be replaced by learning on the first experience, although, the method heavily relies on the initial training covering a majority of classes such that the resulting feature extractor is expressive and discriminative enough. TFM freezes the existing model and extends it at each experience while reusing the representations from previous experiences. This method applies masking on the outputs of each layer to control the flow of gradients so that previous knowledge can be used but its corresponding weights are not modified. However, the dynamic architecture requires the experience ID at test time. Both methods break the challenge rules in multiple ways when used as proposed in their original works.

Horde is inspired by these two methods by having a dynamic zero-forgetting architecture for the ensemble and making use of a pseudo-feature generation approach for learning the unified head. The approach is adapted to the challenge by extending the pseudo-feature generation with the usage of the standard deviation, and using the contrastive loss (Hadsell, Chopra, & LeCun, 2006; Kulis, 2013) to improve the learned shape of the representations, and avoiding the issues for the pseudo-feature generation. Furthermore, the proposed pseudo-feature generation method is adapted to the challenge scenarios that contain class repetition.

### 4.2. Method description

The proposed strategy Horde combines the feature representations of individual FEs into a single unified head capable of predicting all classes seen so far. This is achieved through a two-step training process: first, the learning of an FE (on selected experiences only), and second, the pseudo-feature alignment for adapting the unified head (on each experience). Each individual FE is an expert model trained on a single experience, after which it is frozen and added to the ensemble. In the second training step, data are passed through all the ensemble's models, and the unified head is fine-tuned. Training the unified head involves using representations directly from the expert FEs familiar with a class and the pseudo-feature projections from the representations of FEs not trained on that class.

**When to add a new FE to the ensemble:** For the specific settings of the competition, two constraints (heuristics) were designed to determine when to add an FE to the ensemble. First, to constrain the presence of overly overfitted FEs and to limit the size of the ensemble, experiences with fewer than five classes are not considered. Second, the addition of FEs to the ensemble is stopped after 85% of the classes have been seen, because once robust features for most classes have been learned, good performance on the remaining ones is expected. Additionally, an FE is always trained on the first experience. With these constraints, the proposed approach learns feature extractors based on the highlighted experiences shown in Fig. 4.

**Feature extractor training:** When an FE is trained on the current experience, learning occurs with the usual cross-entropy loss on a fully connected head with as many outputs as classes. In order to promote the learning of features in a more similarly distributed space, a contrastive loss with emphasis on the hard negative pairs is also included on a separate head, as shown in Fig. 5a. Both losses are balanced with an adaptive alpha, which is automatically computed based on the energy of each loss. After learning that experience, the heads are removed and the backbone is frozen and added to the rest of the ensemble.

**Pseudo-feature projection:** Regardless of an FE being trained and added to the ensemble for the current experience, the unified head that uses all the representations is always trained from all FEs and learns to discriminate between all seen classes (Fig. 5b). The proposed pseudo-feature projection extends feature translation (Petit et al., 2023) with the corresponding class standard deviation to allow a better sampling of the dimensions. It translates a representation from a class into a

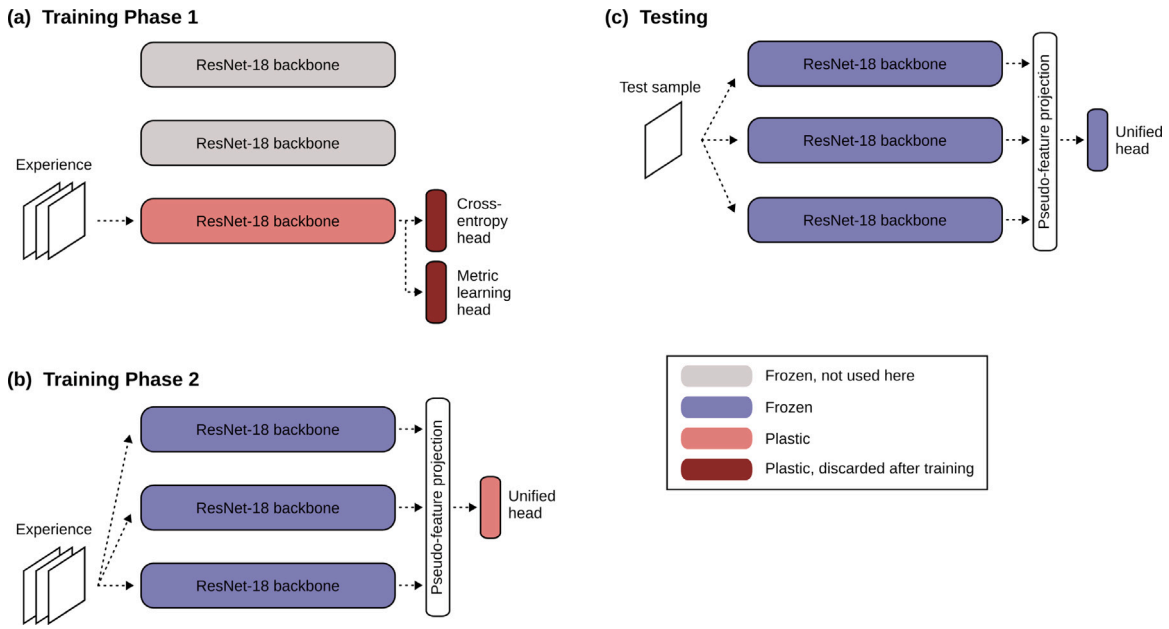



**Fig. 5.** Schematic of Horde during training and test time. **(a)** When training on a new experience, in the first phase, a new feature extractor might be trained using both a cross-entropy and a contrastive loss. Whether a new feature extractor is trained on the new experience is decided by a heuristic, see Fig. 4. **(b)** In the second training phase, which is performed on each new experience, a pseudo-feature projection is performed and a unified head is trained to discriminate between all seen classes based on the features from the ensemble. **(c)** At test time, a test sample is simply forwarded through all components of the model, and the predicted class is read out from the unified head.

projected representation of a different class. Let $a_i$ be the concatenated representation of all FEs outputs for a sample from the current experience belonging to class $i$. The proposed projection is then defined as

$$\hat{a}_{j,i} = \mu_j + \frac{a_i - \mu_i}{\sigma_i} \cdot \sigma_j,$$

where $\hat{a}_{j,i}$ is the estimated projection from class $i$ to class $j$. This projection is applied to each sample in the training batch while learning the unified head, and not used during evaluation. The target class $j$ is chosen at random from previously learned classes, and both the original representation and the projected one are added onto the loss. The class prototypes (i.e., the mean $\mu_i$ and standard deviation $\sigma_i$) are always updated before the unified head is trained by calculating the statistics over the available class data. However, since multiple FEs exist, access might not be available to the mean and standard deviation for each class, depending on when they were learned, or if those classes have appeared since that time. Therefore, if the class statistics are unknown for a feature extractor $e$, estimates are needed for $\hat{\mu}_{c,e}$ and $\hat{\sigma}_{c,e}$. We fix the estimation of the standard deviation to $\hat{\sigma} = 1$, as we do not have any information about the class shape for this feature extractor. For the estimation of the unknown mean statistic of a specific FE output, we resort to the simple heuristic of using the original representation of the current sample without modification: $\hat{\mu}_{c,e} = a_{i,e}$.

Pseudocode for Horde as well as further experimental analysis is presented in Appendix E. The implementation is publicly available on GitHub: https://github.com/mmasana/clvision-chlg-2023/.

## 5. Strategy 3: DWGRNet

The strategy proposed by team *pddbend* is called **D**ynamic **W**eighted **G**ated **R**epresentation **Net**work (**DWGRNet**). This strategy creates independent branches for each experience, and uses gating units to control which branches are active. During training, the branch corresponding to the current experience is activated by its gating unit, facilitating learning, while branches from previous experiences remain inactive. To improve the model's generalization and robustness, data augmentation techniques, such as AugMix (Hendrycks et al., 2019), are used to enhance the data samples. In the testing phase, the gating units control the sequence of predictions. Importantly, using predictions from all branches, as done in Yan, Xie, and He (2021), may not always yield optimal results. This is because many branches may not have been exposed to a particular class, leading to overconfident and potentially inaccurate predictions. This issue resembles an open-set recognition problem. To mitigate this, DWGRNet assigns weights based on entropy, feature norm, and the number of classes experienced by each branch. Specifically, the entropy of each branch's predicted probability distribution is assessed. High entropy indicates the possibility of the sample being an open-set item for that branch. Similarly, the feature norm is computed. A higher feature norm suggests that the sample is likely to be an open-set sample. Finally, it is posited that experiences with a larger number of classes will render the model's prediction more reliable. As a result, weighting can also adjusted based on the number of classes in each experience.

### 5.1. Motivation and related work

To achieve a good balance between stability and plasticity in class-incremental learning, Yan et al. (2021) introduced the method Dynamically Expandable Representation (DER). DER decouples the adaptivity of feature representation from the classification procedure. When faced with a new task that contains novel categories, the DER method first freezes the previously learned feature extractor. Then, it incorporates a new feature extractor to expand the main feature extractor network. Ultimately, the features extracted by all the extractors are combined and forwarded to the classifier module for predicting the category. This strategy exhibits notable efficiency in preserving existing knowledge while providing sufficient flexibility to capture new information.

However, the DER architecture cannot be directly employed in the context of this competition. This is because DER constructs a model ensemble that exploits replay buffers for raw sample storage. Another drawback of DER is the simultaneous execution of predictions across all branches, which has high computational complexity. In practice, DER applies predictions from each branch without distinction. A problem with this is that certain branches produce overconfident predictions for unseen classes, leading to sub-optimal performance.

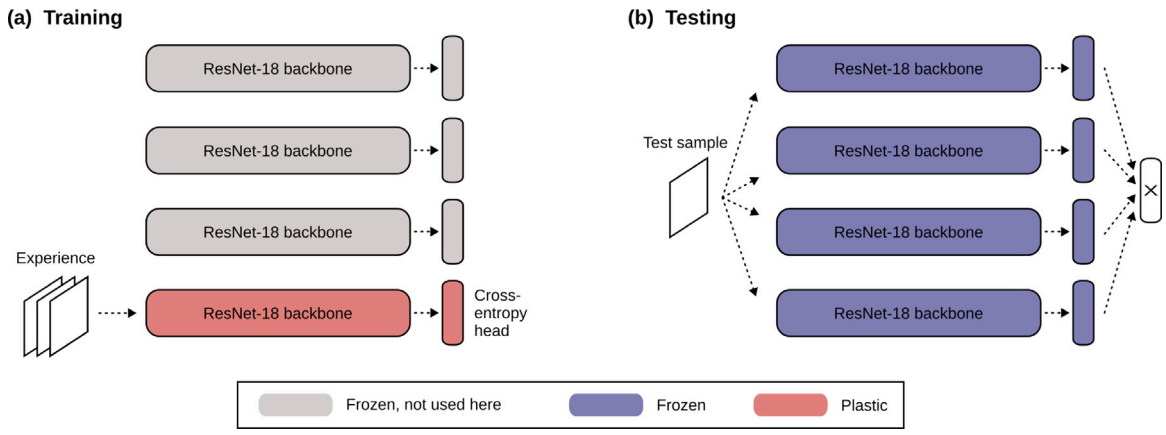



**Fig. 6.** Schematic of DWGRNet during training and test time. **(a)** When training on a new experience, a new model branch is trained on that experience, after which that branch is frozen and added to the model ensemble. **(b)** At test time, a test sample is forwarded through all model branches. The outputs of each branch are collected and combined according to Eq. (1) for the final prediction.

To fix the limitations of DER, DWGRNet implements an open-set approach (Geng, Huang, & Chen, 2020; Sun, Guo, & Li, 2021; Vaze, Han, Vedaldi, & Zisserman, 2022). Within this framework, each model is restricted to be exposed to a limited number of classes. This approach is inspired by recent advancements in open-set recognition literature that have elucidated the relationships between logit entropy, feature norm, and out-of-distribution (OOD) samples. Capitalizing on these findings, a scoring system is designed for OOD samples. This scoring system requires the evaluation of each model to determine whether a test sample falls within its OOD category, prior to the ensembling process. This evaluation allows two possible options: (1) to either suppress the OOD outputs or (2) to harness the OOD scores to assign weight to each sample. Therefore, it serves to alleviate the constraints of the DER method, with the overarching aim of augmenting both its performance and applicability in the realm of CIR.

### 5.2. Method description

DWGRNet uses gating units to control the activation of each branch. A new branch is added with each new experience and then activated, while the model parameters in the old branches are kept frozen. As illustrated in Fig. 6a, during the training phase, no special loss functions or replay buffers are used. Instead, the standard cross-entropy loss is used to train the model, while using AugMix to enhance the model's generalization and robustness. AugMix combines different data augmentation techniques. During the testing phase, the gating units could activate each branch one by one to avoid the need for a large GPU memory. Their outputs are collected and then used to make the final prediction. This process is illustrated in Fig. 6b.

This method is similar to a model ensemble. It is designed in a way that both complies with the competition rules and also speeds up training time and reduces memory space during training and inference. On the other hand, the "naive" model ensembling approach may not work properly as each model only learns limited knowledge. Therefore, the open-set idea is employed. If the problem is treated as an open-set problem, and asks each model whether a test sample is an OOD sample before ensembling, it is possible to either mask the OOD outputs or use the OOD scores to weigh each sample.

The method relies on the concept of entropy to make the strategy work. The assumption is that the entropy of a branch's outputs can indicate their confidence. For an in-distribution sample, the branch's outputs should have low entropy since the branch has seen the class and is confident in its prediction. For an OOD sample, the branch's outputs should have high entropy since the branch has not seen this class before, and lacks confidence in its output. While this phenomenon is not always accurate, it holds true in many circumstances.

**Table 1**
Results from the pre-selection phase for the top ten teams. Shown is the accuracy (as %) on the test set of CIFAR-100 after training on each of the three streams from the pre-selection phase.

| Team | Average accuracy ↑ | Accuracy $S_1$ | Accuracy $S_2$ | Accuracy $S_3$ |
|---|---|---|---|---|
| 1. pddbend | 44.77 | 52.32 | 40.31 | 41.67 |
| 2. linzz | 44.08 | 50.88 | 41.83 | 39.52 |
| 3. shelley | 42.53 | 45.21 | 41.80 | 40.58 |
| 4. xduan7 | 41.37 | 44.99 | 40.80 | 38.31 |
| 5. mmasana | 40.52 | 42.47 | 34.29 | 44.81 |
| 6. zys_tjut | 34.35 | 45.34 | 30.43 | 27.28 |
| 7. Vanixxz | 31.72 | 33.33 | 30.01 | 31.83 |
| 8. M1andy | 31.01 | 33.32 | 28.97 | 30.74 |
| 9. rainstar | 30.88 | 36.01 | 27.77 | 28.87 |
| 10. flyfishzzz | 29.97 | 31.31 | 31.31 | 27.28 |

Additionally, the aim is to further enhance the performance of the model ensemble. First, the reliability of each model branch is to be evaluated. The hypothesis is that the more classes a branch encounters, the more reliable it becomes. Therefore, the number of classes ($N_C$) is used to weight each branch's output. Finally, inspired by the idea of "a good closed-set classifier is all you need" in Vaze et al. (2022), feature norms are used to verify whether the branch perceives the sample as an in-distribution or OOD sample. The OOD samples tend to have a lower feature norm.

To calculate the final logits, the $k$-th output logit of the model ensemble is calculated as:

$$\text{ensemble_logit}_k = \max_{i \in M} \left( \frac{\text{logit}_{i,k}}{\text{entropy}_{i,k}} * {N_C}^{(i)} * \text{feature_norm}_{i,k} \right), \tag{1}$$

where $M$ is the set of saved model branches. An ablation study to investigate the role of each module can be found in Appendix F.

## 6. Results

### 6.1. Pre-selection phase

In the pre-selection phase of the challenge, the teams were ranked based on their average accuracy. The average accuracy for each team was computed as the mean accuracy on the test set after training on each of the three streams released for the first phase, namely $S_1$, $S_2$, and $S_3$. The top ten teams in the pre-selection phase are shown in Table 1. The top five teams proceeded to the final phase.

**Table 2**
Results from the final phase for the finalist teams, along with the performance of popular CL baselines. Team *shelley* is not included since they decided not to participate in the final phase after realizing their solution violated the challenge rules.

| Team | Average accuracy ↑ | Accuracy $S_4$ | Accuracy $S_5$ | Accuracy $S_6$ |
|---|---|---|---|---|
| 1. xduan7 | 62.75 | 63.64 | 68.04 | 56.57 |
| 2. linzz | 45.02 | 46.21 | 47.27 | 41.58 |
| 3. mmasana | 41.11 | 40.82 | 45.79 | 36.72 |
| 4. pddbend | 40.91 | 42.07 | 44.44 | 36.23 |
| Naive | 7.83 | 9.11 | 11.48 | 4.83 |
| EWC | 7.14 | 8.28 | 8.67 | 4.49 |
| LwF | 9.83 | 10.17 | 13.46 | 5.85 |
| ER (buffer: 200) | 9.87 | 8.98 | 13.48 | 7.17 |
| ER (buffer: 2000) | 21.91 | 18.56 | 27.58 | 19.63 |
| Joint | 65.12 | – | – | – |

### 6.2. Final phase

In the final phase of the challenge, three streams, namely $S_4$, $S_5$, and $S_6$, with configurations different from those in the pre-selection phase streams, were used to evaluate the solutions submitted by the finalist teams. The details of the challenge stream configurations are given in Table C.1 in Appendix C. The final phase demonstrated substantial variations in performance among the solutions of the finalist teams, as presented in Table 2. Team *xduan7* (Section 3) was selected as the challenge winner with an average accuracy of 62.75%, thereby substantially outperforming the other finalists. Team *linzz* earned the second spot with an average accuracy of 45.02%. However, they decided not to contribute their solution to this report. The other two teams, *mmasana* (Section 4) and *pddbend* (Section 5), achieved average accuracies of 41.11% and 40.91%, respectively. Recorded video presentation of the submitted solutions by the finalist teams can be accessed through https://sites.google.com/view/clvision2023/challenge/challenge-results.

*Baselines.* To put the performance of the solutions of the finalist teams in perspective, we also report in Table 2 the performance of several popular CL baselines strategies after training on each of the three data streams from the final phase. The approach 'Naive' refers to sequential fine-tuning the Slim-ResNet-18 backbone without any continual learning mechanism. For Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), a popular parameter regularization method, the lambda hyperparameter is set to 1; using higher or lower lambda values results in slightly lower average accuracy. For Learning without Forgetting (LwF) (Li & Hoiem, 2017), a popular functional regularization method, the alpha and temperature hyperparameters are set to their default values of 1 and 2, respectively. For Experience Replay (ER), two versions are run: one with a memory buffer with total capacity of 200 samples and another with total capacity of 2000 samples. For all baselines the implementation of the Avalanche library (version 0.3.1) is used. For 'Joint', the backbone model is trained in an *iid* fashion with access to samples from all classes at the same time.

The comparison with the CL baseline strategies shows that there is a significant gap between the performance of the finalist solutions and these baselines. In particular, even ER with a moderately-sized buffer (i.e., 2000 samples) performs substantially worse than all of the finalist solutions. Notable is that the solution of team *xduan7* approaches the performance of jointly training the backbone model on all training data at the same time.

### 6.3. Additional experiments

#### 6.3.1. No repetition

To test whether the repetition in the data stream plays an important role in the effectiveness of the finalist solutions, they are also evaluated after training on a data stream without repetition. For this, the "standard" Split CIFAR-100 class-incremental learning benchmark is used, with 20 experiences (or tasks) that are encountered one after the other, whereby each experience contains five distinct classes. The results in Table 3 indicate that the performance of the finalist solutions is significantly reduced when there is no repetition in the data stream. Importantly, while ER with buffer size of 2000 was clearly outperformed by each of the finalist solutions on the data streams with repetition, when there is no repetition, this version of ER performs better than all the finalist solutions. This shows that repetition in the data stream can change the relative effectiveness of different CL strategies.

**Table 3**
Results for class-incremental learning experiments on Split CIFAR-100 without repetition. Shown is the accuracy (as %) on the test set at the end of training. Team *linzz* is not included in these additional experiments as they decided not to participate in the writing of this report.

| Team | Accuracy ↑ |
|---|---|
| xduan7 | 24.30 |
| mmasana | 3.39 |
| pddbend | 7.59 |
| Naive | 1.67 |
| EWC | 1.71 |
| LwF | 1.82 |
| ER (buffer: 200) | 3.74 |
| ER (buffer: 2000) | 25.35 |
| Joint | 65.12 |

**Table 4**
Results for the finalist solutions after training on three CIR data streams constructed from the Tiny ImageNet (TIN) dataset. Team *linzz* is not included in these additional experiments as they decided not to participate in the writing of this report.

| Team | Average accuracy ↑ | Accuracy $S_4^{\text{TIN}}$ | Accuracy $S_5^{\text{TIN}}$ | Accuracy $S_6^{\text{TIN}}$ |
|---|---|---|---|---|
| xduan7 | 55.62 | 55.52 | 56.48 | 54.88 |
| mmasana | 34.01 | 34.94 | 38.14 | 28.94 |
| pddbend | 32.68 | 36.66 | 39.12 | 22.28 |

#### 6.3.2. Tiny ImageNet

To probe the generalizability of our results, in this section the finalist solutions are evaluated on three CIR data streams generated using the Tiny ImageNet dataset (Le & Yang, 2015). This dataset contains natural images of $64 \times 64$ pixels, divided over 200 classes, of which we only use a random subset of 100 classes. The configurations to generate the data streams are the same as those used for the data streams $S_4$, $S_5$ and $S_6$ in the final phase of the challenge. The evaluation results are shown in Table 4. These results indicate consistency in the performance of the three finalist solutions relative to the dataset used to generate the data streams.

## 7. Discussion

The challenge of class-incremental learning with repetition asks for a careful balance between learning new information and not forgetting previous knowledge. This report has described three solutions proposed by the finalists of the CLVision challenge at CVPR 2023. These solutions reflect diverse methodologies to address the challenge of *learning continually in the presence of repetition*. Interestingly, an approach that is shared by all finalist solutions is an ensemble-based strategy. This commonality suggests that the relatively simple approach of having separate (sub-)networks per experience might, from a practical perspective, be a very useful strategy for continual learning, especially when there is repetition in the data stream.

The suitability of the ensemble-based strategy for this challenge is illustrated well by the winning solution *HAT-CIR*. During the pre-selection phase, the solution submitted by this team used a single network with a HAT-based partitioning approach, which can be interpreted as a relatively soft version of an ensemble-based strategy. In

the final phase of the challenge, this team switched to using network replicas, a more explicit version of an ensemble-based strategy, which substantially improved the performance. The second solution discussed in this report, *Horde*, trained an ensemble of feature extractors, where a new feature extractor was added only when needed based on a heuristic. It also used a unified head to discriminate between classes that are observed in the current step of training. The third solution, *DWGRNet*, used independent branches for each experience, activated by gating units during training and incorporating data augmentation techniques for robustness. This solution further approached the problem from an open-set problem perspective, and to address open-set recognition issues, it employed a weighting strategy based on factors such as entropy, feature norm, and the number of classes in each experience.

In recent years, several variants of ensemble-based strategies have been explored for continual learning. Two broad types of ensemble-based strategies can be distinguished. On the one hand, a number of studies have explored using an ensemble of multiple models that are all continually trained on the full data stream (Doan, Mirzadeh, & Farajtabar, 2023; Rypeść, Cygert, Khan, Trzcinski, Zieliński, & Twardowski, 2024; Wang et al., 2023; Wang, Zhang, Li, Zhu, & Zhong, 2022). On the other hand, it is also possible to have an ensemble of multiple models whereby each model is trained on a different part of the data stream (e.g., to have a separate model for each task or experience) (Hess, Verwimp, van de Ven, & Tuytelaars, 2024; Vogelstein et al., 2020; Yan et al., 2021). In this challenge, the finalist teams predominantly used the second type of ensemble-based strategy, although the *ensembles* used by HAT-CIR are an example of the first type (while the *fragments* of HAT-CIR belong to the second type).

The results reported in this paper suggest that when there is repetition in the data stream, the ensemble-based strategy might be particularly effective for continual learning. Indeed, we found that while the ensemble-based solutions of the finalists teams clearly outperformed all baseline methods on the data streams with repetition, when the repetition was removed, the finalist solutions became markedly less effective, and they were outperformed by experience replay with a moderately-sized memory buffer. This thus establishes that repetition in the data stream affects the behavior of different CL strategies in different ways. An exciting question for future work is to map out exactly how the amount and the type of repetition affect the effectiveness of each approach.

Another intriguing question is why the ensemble-based approach is so effective for continual learning in the presence of repetition. We speculate that one reason is that the repetition allows each class to be observed in multiple experiences and in different pairings with other classes. This diversity of observed combinations of classes might allow for the development of a "rich" set of feature extractors, leading to more robust representations. Another perspective is that the repetition in the data stream can be interpreted as providing a natural form of bagging, on which an ensemble method can be effectively trained.

Overall, the solutions proposed by the winning teams of the CLVision challenge at CVPR 2023 offer diverse yet converging approaches for the problem of class-incremental learning with repetition. While the proposed solutions are primarily based on the idea of ensemble learning, they also leverage specialized discrimination techniques, data augmentation and weighting strategies. This report provides multiple avenues for future research to further refine and combine these approaches for more effective and robust class-incremental learning strategies, especially when repetition is present in the data stream.

## CRediT authorship contribution statement

**Hamed Hemati:** Writing – original draft, Software, Investigation, Conceptualization. **Lorenzo Pellegrini:** Software. **Xiaotian Duan:** Writing – original draft, Methodology. **Zixuan Zhao:** Writing – original draft, Methodology. **Fangfang Xia:** Writing – original draft, Methodology. **Marc Masana:** Writing – original draft, Methodology. **Benedikt Tscheschner:** Writing – original draft, Methodology. **Eduardo Veas:** Writing – original draft, Methodology. **Yuxiang Zheng:** Writing – original draft, Methodology. **Shiji Zhao:** Writing – original draft, Methodology. **Shao-Yuan Li:** Writing – original draft, Methodology. **Sheng-Jun Huang:** Writing – original draft, Methodology. **Vincenzo Lomonaco:** Conceptualization. **Gido M. van de Ven:** Writing – original draft, Writing – review & editing, Visualization, Supervision, Conceptualization.

## Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, some of the authors used ChatGPT (GPT-4) in order to improve the readability and language of certain sections. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was supported by the Exascale Computing Project, a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration [grant number 17-SC-20-SC]; by TU-Graz SAL DES Lab, part of the "University SAL Labs" initiative of Silicon Austria Labs (SAL) and its Austrian partner universities for applied fundamental research for electronic based systems; by the National Science and Technology Major Project [grant number 2022ZD0114801]; by a senior postdoctoral fellowship from the Research Foundation – Flanders (FWO) [grant number 1266823N]; and by a Marie Skłodowska-Curie fellowship under the European Union's Horizon Europe programme [grant number 101067759].

## Appendix A. Author contributions

In addition to the CRediT authorship contribution statement, here we provide additional information regarding to which authors have been responsible for the development and description of the different strategies reported on in this paper:

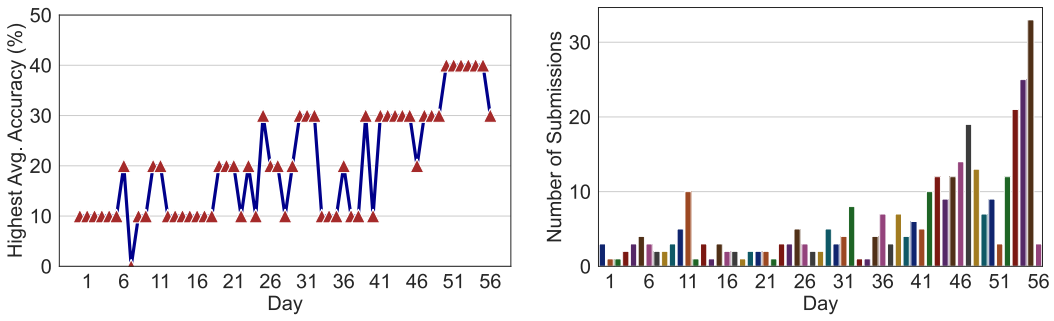
- **Strategy 1: HAT-CIR** developed by Xiaotian Duan, Zixuan Zhao, and Fangfang Xia.
- **Strategy 2: Horde** developed by Marc Masana, Benedikt Tscheschner, and Eduardo Veas.
- **Strategy 3: DWGRNet** developed by Yuxiang Zheng, Shiji Zhao, Shao-Yuan Li, and Sheng-Jun Huang.

## Appendix B. Participation over time

The graphs in Fig. B.1 show the number of submissions per day by all teams, and the maximum average accuracy achieved on each day. In summary, participants began with a low average accuracy of $\sim 10\%$. Over time, the highest average accuracy showed gradual improvement, peaking and stabilizing at $\sim 40\%$ towards the end. The challenge saw a steady increase in engagement, with daily submissions starting from as low as one to three in the beginning, and reaching above 30 towards the end.

## Appendix C. Stream configurations

The configurations of the streams used for both pre-selection and final phases are given in Table C.1.




(a) Highest average accuracy per day. (b) Number of submissions per day.

**Fig. B.1.** Participation over time in the pre-selection phase. Accuracy values are rounded by the competition platform.

**Table C.1**
Configurations of the challenge streams.

| Stream | First occurrence $P_f$ | Repetition Dist. $P_r$ |
|---|---|---|
| $S1$ | Geometric: $p = 0.5$ | Zipfian: $e = 0.7$ |
| $S2$ | Geometric: $p = 0.001$ | Zipfian: $e = 0.8$ |
| $S3$ | Geometric: $p = 0.2$ | Fixed at 0.04 |
| $S4$ | Geometric: $p = 0.6$ | Zipfian: $e = 0.8$ |
| $S5$ | Geometric: $p = 0.001$ | Zipfian: $e = 0.6$ |
| $S6$ | Geometric: $p = 0.1$ | Fixed at 0.05 |

## Appendix D. Appendix – HAT-CIR

### D.1. Ablation study for single-model settings

The ablation study, presented in Table D.1, indicates that various components of the proposed method were important for achieving robust performance in the pre-selection phase, when network replicas were not yet used. Notably, the modified HAT-CL technique significantly outperforms the traditional HAT method in the context of CIR, thus confirming its efficacy. The supervised contrastive learning phase also plays a crucial role in learning better feature representations, as its absence led to a marked decline in performance.

### D.2. Influence of number of network replicas

The results presented in Table D.2 demonstrate that both increases in ensemble replicas and increases in fragment replicas enhance performance. Yet, by adding more replicas, the added benefit starts to diminish. This suggests there is a sweet spot in balancing the number of fragments and ensembles for optimal performance without excessive training and/or test time. Importantly, adding ensemble replicas comes with the trade-off of both longer training and test time, while adding fragment replicas only comes with the trade-off of longer test time. Because the rules of the challenge predominantly put restrictions on computational resources during training, the final version of the model that was selected predominantly used fragment replicas.

## Appendix E. Appendix – Horde

### E.1. Algorithm

Inspired by zero-forgetting methods, Horde promotes stability on each task-specific Feature Extractor (FE) while leveraging class repetition for balancing plasticity by learning an aligned unified representation. Each extractor provides a discriminative representation for the seen classes of the task where it is learned. Then, pseudo-feature projection is used to achieve alignment on the unified head. A description of the pseudo-code of Horde is provided in Algorithm 1. Additionally, the feature extractor training steps are shown in Algorithm 2.

**Algorithm 1** Horde Algorithm

**for** experience **do**
  **if** Training Condition **then**
    Train FE and add to ensemble ▷ See Alg. 2
  **end if**
  Calculate $\mu_c$ and $\sigma_c$ for all classes $c$
  **for** training epoch **do** ▷ Only Unified head
    **for** Batch **do**
      Generate $\hat{a}_{j,i}$ from $a_i$ and old classes
      Calculate $\mathcal{L}_{CE}$ with both $a_i$ and $\hat{a}_{j,i}$
      Backprop $\mathcal{L}_{CE}$
    **end for**
  **end for**
**end for**

**Algorithm 2** FE Training

Initialize CE and ML Head
Initialize new or transfer learning model
**for** training epoch **do**
  **for** Batch **do**
    Calculate $\mathcal{L}_{CE}$
    Calculate $\mathcal{L}_{ML}$
    Backprop $(1-\alpha)\cdot\mathcal{L}_{CE} + \alpha\cdot\mathcal{L}_{ML}$
  **end for**
  Update $\alpha$ from avg. $\mathcal{L}_{CE}$ and $\mathcal{L}_{ML}$
**end for**
Remove CE and ML head
Freeze FE

### E.2. Additional experimental analysis

To further study the effects of the ensemble size, the Horde strategy is evaluated with different number of maximum ensemble size. The results are presented in Fig. E.1. Stream 1 seems to be invariant to the ensemble size, probably due to the first experience having 51 classes and, therefore, providing a robust representation from the beginning. This can be exploited by heavily enforcing stability and limited number of FEs, thus becoming unnecessary to extend the ensemble after enough classes are seen. Streams 2 and 3 seem to show a benefit of extending the ensemble over time, which accommodates the incorporation of new seen classes without the need of replacing older FEs.

**Table D.1**
Ablation study results for HAT-CIR when using a single model without using network replicas. Shown is the test accuracy (in %) after training on each data stream from the final phase of the challenge. The "Avg. Decrease from Baseline" column indicates how much performance is lost relative to the baseline (i.e., the version of HAT-CIR that was used in the pre-selection phase of the challenge). The term "Original HAT" refers to the original hard attention method, "SupCon" stands for supervised contrastive learning, and "Momentum & TTA" are abbreviations for momentum-based test-time decision-making and test-time augmentation, respectively.

| Method | Accuracy $S_4$ | Accuracy $S_5$ | Accuracy $S_6$ | Avg. Decrease from Baseline |
|---|---|---|---|---|
| Baseline | 39.08 | 42.89 | 34.59 | – |
| With Original HAT | 23.89 | 26.05 | 19.47 | 15.71 |
| Without SupCon | 35.39 | 28.93 | 22.16 | 10.02 |
| Without Momentum & TTA | 30.67 | 35.16 | 26.61 | 8.04 |

**Table D.2**
Effect of the number and types of network replicas on the performance of HAT-CIR on the data streams from the final phase of the challenge. The table shows the test accuracy (in %) achieved for each data stream with different numbers of fragments and ensembles.

| # of Replicas | # of Fragments | # of Ensembles | Accuracy $S_4$ | Accuracy $S_5$ | Accuracy $S_6$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 39.08 | 42.89 | 34.59 |
| 2 | 2 | 1 | 40.20 | 43.03 | 35.90 |
| 2 | 1 | 2 | 43.90 | 46.17 | 39.11 |
| 10 | 5 | 2 | 45.96 | 49.42 | 42.18 |
| 10 | 2 | 5 | 48.06 | 48.50 | 44.69 |
| 50 | 10 | 5 | 51.14 | 53.22 | 46.37 |
| 50 | 5 | 10 | 50.70 | 52.58 | 47.92 |
| 100 | 10 | 10 | 52.74 | 54.13 | 46.87 |
| **100** | **50** | **2** | **63.64** | **68.04** | **56.57** |

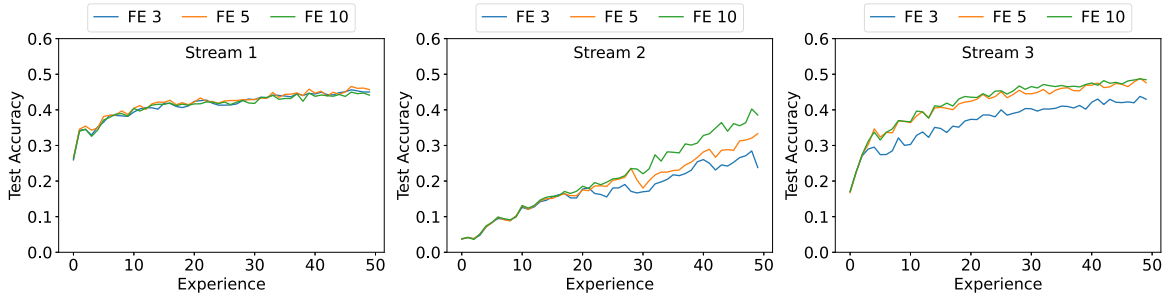



**Fig. E.1.** Impact of the maximum number of feature extractors allowed to be trained. The heuristic to decide when feature extractors are trained is not changed. Results are shown in the form of the test accuracy (as a proportion) after training on each of the three streams from the challenge's initial phase.

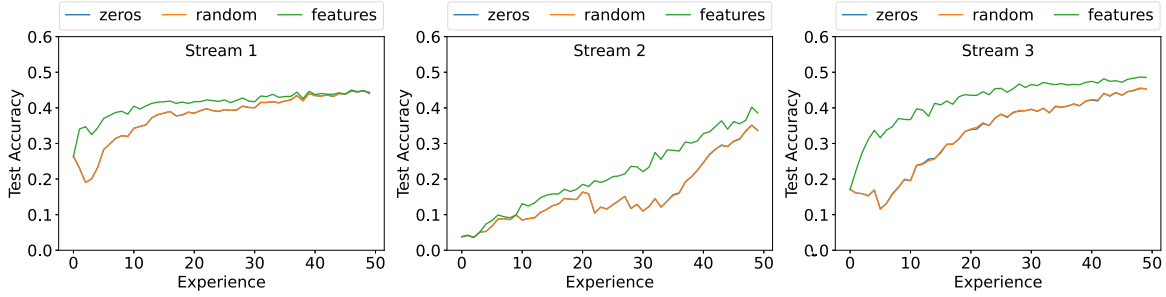



**Fig. E.2.** Impact of estimating unknown class statistics under different heuristics. Zeros and random heuristics had almost the same performance. Results are shown in the form of the test accuracy (as a proportion) after training on each of the three data streams from the initial phase of the challenge.

Another interesting direction to investigate is the different possibilities for estimating unknown class statistics for a feature extractor. Since multiple FEs exist, access might not be available to the mean and standard deviation for each class, depending on when they were learned, or if those classes have reappeared since last time. The complete class statistics $\mu_c$ and $\sigma_c$ can thus be written as a concatenation of the individual FEs where $n$ denotes the number of feature extractors.

$$\mu_c = (\mu_{c,1}, \ldots, \mu_{c,n})$$

$$\sigma_c = (\sigma_{c,1}, \ldots, \sigma_{c,n})$$

For the pseudo-feature projection step, given a class $c$ and a feature extractor $e$, estimates are needed for $\hat{\mu}_{c,e}$ and $\hat{\sigma}_{c,e}$. In cases we do not have any information about the class shape for this feature extractor, we fix the estimation of the standard deviation to $\hat{\sigma} = 1$. For the estimation of $\hat{\mu}_{c,e}$ we propose three heuristics:

1. **zeros**: setting all $\hat{\mu}_{c,e}$ estimations to 0.

**Table F.1**
Result from the ablation study for the DWGRNet strategy.

| Module | Average accuracy ↑ | Accuracy $S_1$ | Accuracy $S_2$ | Accuracy $S_3$ |
|---|---|---|---|---|
| Only mean | 18.34 | 20.22 | 12.08 | 22.71 |
| + Entropy | 41.61 | 49.16 | 38.14 | 37.52 |
| + $N_C$ | 42.79 | 50.11 | 38.62 | 39.64 |
| + feature_norm | 44.77 | 52.32 | 40.31 | 41.67 |

2. **random**: randomly sample $\hat{\mu}_{c,e}$ from a multivariate normal distribution $\mathcal{N}(0, 1)$.
3. **original features**: estimate $\hat{\mu}_{c,e}$ with the original representation of the current sample without modification.

Results are presented in Fig. E.2, showing that estimating with the original representations provides a clear advantage. Both zeros and random heuristics seem to perform similarly, and clearly underperform compared to the original features one.

## Appendix F. Appendix – DWGRNet

### F.1. Additional experimental analysis

An ablation study was conducted to investigate the role of each module. The results are presented in Table F.1. It can be seen that entropy plays the most significant role in the final accuracy. Adjusting each model's logits using the number of classes and the number of features also leads to improved accuracy. This verifies the effectiveness of each module.

## Data availability

We share code for the challenge (https://github.com/ContinualAI/clvision-challenge-2023) and for two of the described approaches (https://github.com/xduan7/clvis-chlg-2023 and https://github.com/xduan7/clvis-chlg-2023)

## References


Aljundi, R., Chakravarty, P., & Tuytelaars, T. (2017). Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3366–3375).

Carta, A., Pellegrini, L., Cossu, A., Hemati, H., & Lomonaco, V. (2023). Avalanche: A pytorch library for deep continual learning. *Journal of Machine Learning Research*, *24*(363), 1–6.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., et al. (2019). On tiny episodic memories in continual learning. arXiv preprint arXiv:1902.10486.

Chen, Z., & Liu, B. (2018). Lifelong machine learning, second edition. In *Synthesis lectures on artificial intelligence and machine learning*, Springer Cham.

Cho, S., Lee, H., Baek, S., & Paik, S.-B. (2024). Neuromimetic metaplasticity for adaptive continual learning. arXiv preprint arXiv:2407.07133.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., et al. (2022). A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(7), 3366–3385.

Doan, T., Mirzadeh, S. I., & Farajtabar, M. (2023). Continual learning beyond a single model. In *Proceedings of machine learning research*: *vol. 232*, *Proceedings of the 2nd conference on lifelong learning agents* (pp. 961–991). PMLR, URL https://proceedings.mlr.press/v232/doan23a.html.

Duan, X. (2023). HAT-CL: A hard-attention-to-the-task PyTorch library for continual learning. arXiv preprint arXiv:2307.09653.

Feng, K., Zhao, X., Liu, J., Cai, Y., Ye, Z., Chen, C., et al. (2019). Spaced learning enhances episodic memory by increasing neural pattern similarity across repetitions. *Journal of Neuroscience*, *39*(27), 5351–5360.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., et al. (2017). PathNet: Evolution channels gradient descent in super neural networks. arXiv preprint arXiv:1701.08734.

Geng, C., Huang, S.-j., & Chen, S. (2020). Recent advances in open set recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(10), 3614–3631.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *IEEE conference on computer vision and pattern recognition* (pp. 1735–1742).

Hemati, H., Cossu, A., Carta, A., Hurtado, J., Pellegrini, L., Bacciu, D., et al. (2023). Class-incremental learning with repetition. In S. Chandar, R. Pascanu, H. Sedghi, D. Precup (Eds.), *Proceedings of machine learning research*: *vol. 232*, *Proceedings of the 2nd conference on lifelong learning agents* (pp. 437–455). PMLR, URL https://proceedings.mlr.press/v232/hemati23b.html.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., & Lakshminarayanan, B. (2019). Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781.

Hess, T., Verwimp, E., van de Ven, G. M., & Tuytelaars, T. (2024). Knowledge accumulation in continually learned representations and the issue of feature forgetting. *Transactions on Machine Learning Research*, URL https://openreview.net/forum?id=aHtZuZfHcf.

Hsu, Y.-C., Liu, Y.-C., Ramasamy, A., & Kira, Z. (2018). Re-evaluating continual learning scenarios: A categorization and case for strong baselines. arXiv preprint arXiv:1810.12488.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., et al. (2020). Supervised contrastive learning. *Advances in Neural Information Processing Systems*, *33*, 18661–18673.

Kim, G., Esmaeilpour, S., Xiao, C., & Liu, B. (2022). Continual learning based on OOD detection and task masking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3856–3866).

Kim, G., Liu, B., & Ke, Z. (2022). A multi-head model for continual learning via out-of-distribution replay. In S. Chandar, R. Pascanu, & D. Precup (Eds.), *Proceedings of machine learning research*: *vol. 199*, *Proceedings of the 1st conference on lifelong learning agents* (pp. 548–563). PMLR, URL https://proceedings.mlr.press/v199/kim22a.html.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526.

Koh, H., Kim, D., Ha, J.-W., & Choi, J. (2022). Online continual learning on class incremental blurry task configuration with anytime inference. In *International conference on learning representations*. URL https://openreview.net/forum?id=nrGGfMbY_qK.

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*: *Technical report*, University of Toronto.

Kulis, B. (2013). Metric learning: A survey. *Foundations and Trends in Machine Learning*, *5*(4), 287–364.

Le, Y., & Yang, X. (2015). *Tiny imagenet visual recognition challenge*: *Technical report*, Stanford University.

Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., & Díaz-Rodríguez, N. (2020). Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, *58*, 52–68, URL https://www.sciencedirect.com/science/article/pii/S1566253519307377.

Lesort, T., Ostapenko, O., Misra, D., Arefin, M. R., Rodríguez, P., Charlin, L., et al. (2022). Scaling the number of tasks in continual learning. arXiv preprint arXiv:2207.04543.

Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(12), 2935–2947.

Lomonaco, V., Pellegrini, L., Rodriguez, P., Caccia, M., She, Q., Chen, Y., et al. (2022). CVPR 2020 continual learning in computer vision competition: Approaches, results, current challenges and future directions. *Artificial Intelligence*, *303*, Article 103635.

Mallya, A., & Lazebnik, S. (2018). PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7765–7773).

Maltoni, D., & Lomonaco, V. (2019). Continuous learning in single-incremental-task scenarios. *Neural Networks*, *116*, 56–73.

Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., & Van De Weijer, J. (2023). Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(5), 5513–5533.

Masana, M., Tuytelaars, T., & van de Weijer, J. (2021). Ternary feature masks: Zero-forgetting for task-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR) workshops* (pp. 3570–3579).

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*: *vol. 24*, (pp. 109–165). Elsevier.

Mermillod, M., Bugaiska, A., & Bonin, P. (2013). The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., et al. (2018). Never-ending learning. *Communications of the ACM*, *61*(5), 103–115.

Moon, J.-Y., Park, K.-H., Kim, J. U., & Park, G.-M. (2023). Online class incremental learning on stochastic blurry task boundary via mask and visual prompt tuning. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 11731–11741).

Normandin, F., Ostapenko, A., Caccia, M., Riemer, M., Khetarpal, K., Golemo, F., et al. (2021). Continual learning challenge, CVPR 2021. https://eval.ai/web/challenges/challenge-page/829/overview. (Accessed: 29 March 2024).

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1717–1724).

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, *113*, 54–71, URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

Pavao, A., Guyon, I., Letournel, A.-C., Tran, D.-T., Baro, X., Escalante, H. J., et al. (2023). CodaLab competitions: An open source platform to organize scientific challenges. *Journal of Machine Learning Research*, *24*(198), 1–6, URL http://jmlr.org/papers/v24/21-1436.html.

Pellegrini, L., Zhu, C., Xiao, F., Yan, Z., Carta, A., De Lange, M., et al. (2022). 3Rd continual learning workshop challenge on egocentric category and instance level object understanding. arXiv preprint arXiv:2212.06833.

Petit, G., Popescu, A., Schindler, H., Picard, D., & Delezoide, B. (2023). 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).

Rajasegaran, J., Hayat, M., Khan, S. H., Khan, F. S., & Shao, L. (2019). Random path selection for continual learning. *Advances in Neural Information Processing Systems*, *32*.

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, *97*(2), 285–308.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). Icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2001–2010).

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., & Wayne, G. (2019). Experience replay for continual learning. *Advances in Neural Information Processing Systems*, *32*.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., et al. (2016). Progressive neural networks. arXiv preprint arXiv:1606.04671.

Rypeść, G., Cygert, S., Khan, V., Trzcinski, T., Zieliński, B. M., & Twardowski, B. (2024). Divide and not forget: Ensemble of selectively trained experts in continual learning. In *International conference on learning representations*. URL https://openreview.net/forum?id=sSyytcewxe.

Serra, J., Suris, D., Miron, M., & Karatzoglou, A. (2018). Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning* (pp. 4548–4557). PMLR.

Shin, H., Lee, J. K., Kim, J., & Kim, J. (2017). Continual learning with deep generative replay. *Advances in Neural Information Processing Systems*, *30*.

Smith, C. D., & Scarf, D. (2017). Spacing repetitions over long timescales: a review and a reconsolidation explanation. *Frontiers in Psychology*, *8*, 962.

Stojanov, S., Mishra, S., Thai, N. A., Dhanda, N., Humayun, A., Yu, C., et al. (2019). Incremental object learning from contiguous views. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8777–8786).

Sun, Y., Guo, C., & Li, Y. (2021). ReAct: Out-of-distribution detection with rectified activations. *Advances in Neural Information Processing Systems*, *34*, 144–157.

Thrun, S., & Mitchell, T. M. (1995). Lifelong robot learning. *Robotics and Autonomous Systems*, *15*(1), 25–46, URL https://www.sciencedirect.com/science/article/pii/092188909500004Y.

van de Ven, G. M., Siegelmann, H. T., & Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, *11*, 4069.

van de Ven, G. M., & Tolias, A. S. (2018). Three continual learning scenarios. In *NeurIPS continual learning workshop*.

van de Ven, G. M., Tuytelaars, T., & Tolias, A. S. (2022). Three types of incremental learning. *Nature Machine Intelligence*, *4*(12), 1185–1197.

Vaze, S., Han, K., Vedaldi, A., & Zisserman, A. (2022). Open-set recognition: a good closed-set classifier is all you need? In *International conference on learning representations*.

Verwimp, E., Aljundi, R., Ben-David, S., Bethge, M., Cossu, A., Gepperth, A., et al. (2024). Continual learning: Applications and the road forward. *Transactions on Machine Learning Research*, URL https://openreview.net/forum?id=axBIMcGZn9.

Verwimp, E., Yang, K., Parisot, S., Hong, L., McDonagh, S., Pérez-Pellitero, E., et al. (2023). CLAD: A realistic continual learning benchmark for autonomous driving. *Neural Networks*, *161*, 659–669.

Vogelstein, J. T., Dey, J., Helm, H. S., LeVine, W., Mehta, R. D., Tomita, T. M., et al. (2020). Representation ensembling for synergistic lifelong learning with quasilinear complexity. arXiv preprint arXiv:2004.12908.

Wang, L., Zhang, X., Li, Q., Zhang, M., Su, H., Zhu, J., et al. (2023). Incorporating neuro-inspired adaptability for continual learning in artificial intelligence. *Nature Machine Intelligence*, *5*(12), 1356–1368.

Wang, L., Zhang, X., Li, Q., Zhu, J., & Zhong, Y. (2022). CoSCL: Cooperation of small continual learners is stronger than a big one. In *European conference on computer vision* (pp. 254–271). Springer.

Xu, J., & Zhu, Z. (2018). Reinforced continual learning. *Advances in Neural Information Processing Systems*, *31*.

Yan, S., Xie, J., & He, X. (2021). DER: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3014–3023).

Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International conference on machine learning* (pp. 3987–3995). PMLR.

Zhan, L., Guo, D., Chen, G., & Yang, J. (2018). Effects of repetition learning on associative recognition over time: Role of the hippocampus and prefrontal cortex. *Frontiers in Human Neuroscience*, *12*, 277.

Zhu, C., Xiao, F., Alvarado, A., Babaei, Y., Hu, J., El-Mohri, H., et al. (2023). EgoObjects: A large-scale egocentric dataset for fine-grained object understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*.