RECALL: Rehearsal-free Continual Learning for Object Classification

Knauer, Markus and Denninger, Maximilian and Triebel, Rudolph

RECALL: Rehearsal-free Continual Learning for Object Classification

Markus Knauer[1], Maximilian Denninger[1,2] and Rudolph Triebel[1,2]

Abstract— Convolutional neural networks show remarkable results in classification but struggle with learning new things on the fly. We present a novel rehearsal-free approach, where a deep neural network is continually learning new unseen object categories without saving any data of prior sequences. Our approach is called RECALL, as the network recalls categories by calculating logits for old categories before training new ones. These are then used during training to avoid changing the old categories. For each new sequence, a new head is added to accommodate the new categories. To mitigate forgetting, we present a regularization strategy where we replace the classification with a regression. Moreover, for the known categories, we propose a Mahalanobis loss that includes the variances to account for the changing densities between known and unknown categories. Finally, we present a novel dataset for continual learning (HOWS-CL-25), especially suited for object recognition on a mobile robot, including 150,795 synthetic images of 25 household object categories. Our approach RECALL outperforms the current state of the art on CORe50 and iCIFAR-100 and reaches the best performance on HOWS-CL-25.

I. INTRODUCTION

Humans are remarkable in extracting new knowledge about unknown things continuously throughout their lifetime. Thus, lifelong learning is a crucial capability in our daily life. Deep neural networks have shown excellent results on many problems, from recognition to reconstruction tasks in computer vision and more [1], [2], [3], [4]. Typically, these algorithms apply batch-wise training to large datasets, *e.g.*, ImageNet by Deng *et al*. [5], and then need many iterations over the whole dataset to obtain satisfactory performance. In contrast to humans, neural networks rely on a static dataset, which has to be fixed before the training starts. For new categories that have not been part of the original training set, the network has to be trained again, while the original training set has to be kept in memory (rehearsal) to prevent forgetting. It is not practical to keep relearning previously seen categories as it is time and memory-consuming. So in this work, we present a novel approach to do **Re**hearsal-free **C**ontinu**al** **L**earning called **RECALL**, which can be used to enable a network to learn new categories on-demand without the need of retraining the whole network. Therefore, the experiments are designed for a class-incremental learning (Class-IL) [6], schematized in Fig. 1, where each sequence contains a collection of images of a non-reoccurring fixed amount of categories. This means that a category is only present in one sequence. In order to get closer to a lifelong learning approach, we propose a rehearsal-free approach,

[1] Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Oberpfaffenhofen, Germany `first.last@dlr.de`
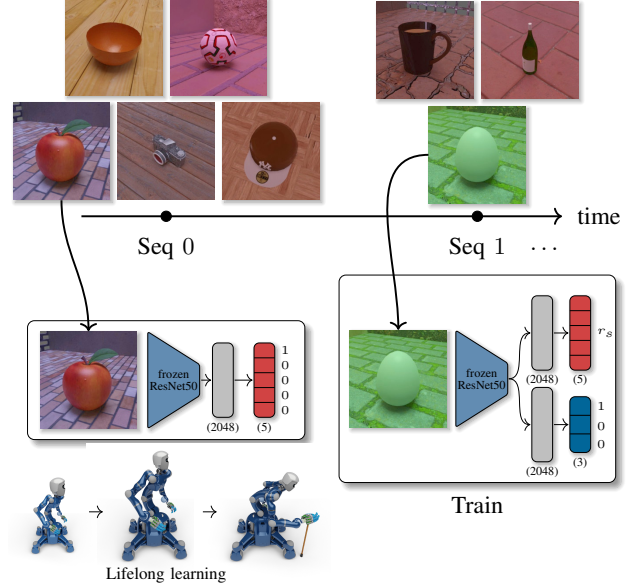[2] Technical University of Munich (TUM), Germany `triebel@in.tum.de`

Fig. 1: Continual learning without using previously seen examples. In each sequence, the model has to learn new categories while also ensuring not to forget previous ones.

which does not require storing any visual history like most other solutions in literature [7], [8]. The main contributions of this work are:

- Proposal of RECALL: A method to mitigate forgetting in a rehearsal-free continual learning environment.
- Analysis of the logit output distribution discrepancies of RECALL, which are caused by learning over different sequences. Evaluation and proposal of different solutions to cope with these discrepancies.
- Introduction of a novel dataset (HOWS-CL-25) with household objects and two suggested sequence structures in a continual learning manner.

II. RELATED WORK

Continual learning can be divided mainly into architectural, regularization, rehearsal, and a combination of those different strategies.

Progressive Neural Networks (PNN), by Rusu *et al*. [9], and Copy Weight with Reinit (CWR), by Lomonaco *et al*. [10], solve the problem of catastrophic forgetting by an **architectural strategy**. Both works let their network dynamically grow to accommodate new categories. However, in PNN, new sub-networks are repeatedly connected to the old frozen sub-networks, ensuring the usage of prior knowledge, while in CWR, new weights are trained for new

sequences, which are then frozen after training. In contrast to that, in our work, all weights are trained at the same time, enabling the network to learn those differences and prevent it from forgetting.

In literature, learning without forgetting (LwF) [11] is often used for comparison [12], [7], [13], [8], [14]. They propose a **regularization strategy** to stabilize the model accuracy on old tasks using knowledge distillation, proposed by Hinton *et al.* [15]. Here, the logits of the previous and current sequences are encouraged to be similar when applied to data from the new sequence. Further, they propose a form of knowledge distillation, where they record a set of label probabilities for each training image on the previous network weights. In LWF, they compensate for the distribution shift in the predicted probability distribution of different sequences by first freezing the old neurons and only training the new neurons in a warm-up step. Whereas in our work, we record the logits directly instead of modeling them through a probability distribution. Further, we propose to use a regression loss to compensate for distribution shifts, as it removes them completely.

Another regularization strategy is Elastic Weight Consolidation (EWC) [16], proposed by Kirkpatrick *et al.*, which tries to protect important weights from changing by using a Fisher weight importance matrix. Another one is Synaptic Intelligence (SI) [13], by Zenke *et al.*, where they propose calculating the weight importance on the fly, using Stochastic Gradient Descent. Both approaches are rooted in neural science, as they argue that a biological synapse accumulates task-relevant information over time and stores new memories without forgetting old ones. In this work, we change all weights on purpose to give the network the ability to learn all kinds of separations, and we tackle the forgetting by enforcing that the network's outputs don't change.

The most similar approach to our work is AR1 by Maltoni and Lomonaco [12]. It is a **combination** of an architectural strategy (CWR+) and a regularization strategy (SI). AR1 uses importance weighting and therefore needs the entire backbone during training. They also zero-init their last layer and propose to use mean-shifting to improve their continual learning performance. In RECALL, on the other hand, only the backbone is frozen and used to generate the features beforehand, thus decreasing training time. Furthermore, each sequence gets its own head to improve the overall capacity of the network and its accuracy instead of just one single layer in AR1. Lastly, AR1 uses a classification loss instead of a regression loss.

All previously mentioned approaches are rehearsal-free and therefore comparable to ours, as no examples of previous tasks are available here. Another way of continual learning is given by **rehearsal strategies**, where previous training examples or representations of them are stored and used in later training steps [17], [18], [19]. Two relevant methods are Incremental Classifier and Representation Learning (iCaRL), by Rebuffi *et al.* [7], and Gradient Episodic Memory (GEM) from Lopez-Patz and Ranzato [8]. In iCaRL, they propose a class-incremental algorithm to learn new classes over time.

It relies on a regularization strategy in a nearest-mean-of-exemplar classification and a rehearsal strategy by saving feature representations in each sequence. Like LWF, iCaRL uses a combination of knowledge distillation and classification loss to train the network. GEM uses the same strategies, but while iCaRL is designed to fill the whole memory after every batch, GEM uses a fixed amount of memory for each batch, a so-called episodic memory. Instead of keeping the predictions of past sequences invariant by using distillation, GEM uses the losses as an inequality constraint. This avoids their increase but allows their decrease in order to make the positive backward transfer possible. Like Rebuffi *et al.* [7], our approach focuses on keeping the predictions of past sequences invariant and does not consider the possibility of a positive backward transfer proposed in GEM. Another approach is Persistent Anytime Learning of Objects from Unseen Classes (PAL) by Denninger *et al.* [20]. Similar to us, they also use CNNs for feature extraction, but instead of fully connected layers, they use a random forest classifier, which also relies on saving previously seen data samples. In their online learning scenario, they found that the removal of trees leads to catastrophic forgetting and that the performance of learning new categories decreases over time.

These approaches reduce forgetting by saving samples from previous sequences. However, our overarching goal in this work is to get closer to a system that is able to learn continually without depending on storing previously seen examples. A lifelong learning strategy should not contain the requirement to store everything it has ever seen.

III. RECALL

This chapter introduces RECALL, our algorithm for rehearsal-free object classification on 2D images for continual learning. We selected ResNet50 as a feature extractor and backbone, which is pre-trained on Imagenet [1], [5].
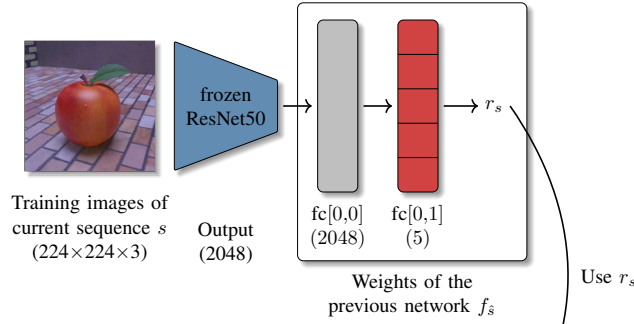
A. Architectural Strategies

Due to the continual procedure, where new categories are shown in sequences, we propose to let the network grow in each sequence. This is depicted in Fig. 1 and Fig. 2. Here, the first sequence is learned as in a classical CNN. For each following sequence, we propose to add a new head. Each newly added head has two fully-connected layers, called fc$[s, 0]$ and fc$[s, 1]$ with the sequence number s. Whereas each fc$[s, 1]$ has the same amount of outputs as categories C_s are in sequence s. During the training of this new sequence, all weights of all added heads are trained, and only the backbone is kept frozen. The input to each head is the same 2048 feature vector. By doing this, the network can better address the feature separation in specific sequences. This also avoids that the training of the categories of a new sequence changes any of the weights of prior sequences heads.

B. Regularization Strategies

The next step is to tackle catastrophic forgetting, which describes the decreasing performance on the categories of prior sequences when new sequences are trained. Our goal

① Prediction of the recall labels r_s before training a new sequence.

Training images of current sequence s (224×224×3) Output (2048)

fc[0,0] (2048) fc[0,1] (5)

Weights of the previous network $f_{\hat{s}}$

r_s Use r_s

② Training of the current sequence after expanding the network.

frozen ResNet50

Training images of current sequence s (224×224×3) Output (2048)

fc[0,0] (2048) fc[0,1] (5) L_s^{recall}

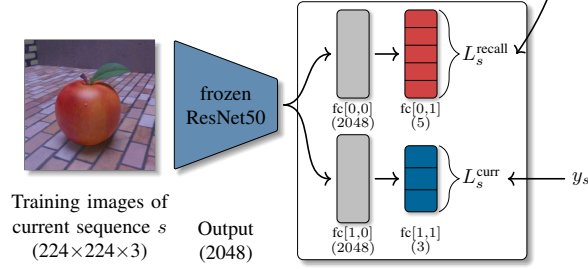fc[1,0] (2048) fc[1,1] (3) L_s^{curr} y_s

Fig. 2: Calculation of the recall label r: First, we calculate the logits for each training example of the current sequence s. After that, a new head is added (fc[1,0], fc[1,1]) and the recall labels r_s, are used for training in sequence s.

is now to prevent the logits of the categories of previous sequences from changing.

1) Recall label: In the first sequence, image-label pairs (x, y) are used for training, where the label y represents the one-hot encoded category of an object. In the following sequences, however, these one-hot encoded categories might lead to catastrophic forgetting, as the ground truth labels for categories, which are no longer present, are zero. To reach this zero value during training, the categories logit values of the network have to continually move into the negative, as they can only reach zero in the softmax if $x \to -\infty$ in e^x. This then changes the previously learned categories by forcing them to predict zeros regardless of the input. The logit value here is the input to the final softmax. In order to prevent this forgetting, we propose another label for already learned categories. As a label represents a value, which the network's output should target, we propose to use the original logit output on the training image. We call this value recall label r. This is similar to the approach from Li *et al.* [11], but they save the final softmax output and not the logit itself. We argue that the softmax output already has too much influence from the other classes making the decoupling harder.

But, the labels $y_{\hat{s}}$ from any previous sequence $\hat{s}$ are no longer present in the next sequence s. Thus, we only have the image-label pairs (x_s, y_s). Therefore, we propose replacing all zeros in the current y_s, which stand for the previous categories' labels with our recall label r_s. This recall label r_s is then compared to the output of fc[0, 1] at the current sequence s. The training procedure for this is shown in

Algorithm 1, where the calculation of the recall label happens in line three. This procedure is also depicted in Fig. 2.

Given the model $f(x)$ from our method, sequences $S \subseteq \mathbb{N}_0$ and categories $C \subseteq \mathbb{N}_0$, where each sequence $s \in S$ has its own categories $C_s \subseteq C$, where $C_{s_i} \cap C_{s_j} = \emptyset$ and $s_i \neq s_j$ for $s_i, s_j \in S$ and its network weights θ_s, we do:

- In the first step, the model f with its weights θ_0 is trained using the image-class pairs (x_0, y_0) of sequence $s = 0$ (see Algorithm 1, line six).
- For each new sequence $s \in S$, the recall labels r_s are calculated for each image of the current sequence s, using the weights $\theta_{\hat{s}}$ of the previous sequence (see Algorithm 1, line three).
- We create new network weights θ_s by adding a new head $f_{\hat{s}}$ (fc[1, 0], fc[1, 1]) according to the number of categories in the current sequence $|C_s|$.
- Now, the model f_s can be trained.

Algorithm 1 Network training procedure

1: **procedure** TRAIN_NETWORK(Network f, Sequence s, Categories C, Images x_s, Labels y_s)
2:     **if** $s \neq 0$ **then**
3:         $r_s = [f.\text{predictWithoutSoftmax}(X = i) \text{ for } i \text{ in } x]$
4:         $y_s = \text{concatenate}(r_s, y_s)$
5:     EXPAND_NETWORK($f, |(C_s)|$)
6:     $f.\text{train}(X = x_s, Y = y_s)$
7:     **return** $f(x)$

---

2) Loss function: In order to enable the network to learn how to reconstruct the previous network outputs and simultaneously learn new categories, we have to adapt the loss function. So, we define a new loss, as we are now using recall labels r_s for the categories of previous sequences and one-hot encoded classification labels for the current categories. Our loss function consists out of three parts:

a) 1. Loss on the previous categories $\boldsymbol{L}^{recall}$: The aim is to force the logit output space of the categories of previous sequences C_s^{prev} to only slightly adapt to the categories of the current sequence C_s, but to mainly stay the same, as those previous categories are not represented in the training anymore.

$$L_{i,s}^{\text{recall}} = \frac{1}{|C_s^{\text{prev}}|} \sum_{c \in C_s^{\text{prev}}} (o_i[c] - r_i[c])^2 \quad (1)$$

Equation (1) shows the loss on previous categories $L_{i,s}^{\text{recall}}$, where a regression loss is used instead of a classification loss. Namely, an L2-norm is calculated between the recall label r and the network's logit output o for a given training example i and a given category c. This is done for all c in C_s^{prev}:

$$C_s^{\text{prev}} = \begin{cases} \emptyset & \text{if } s = 0 \\ \bigcup_{s'}^{[0,\ldots,\hat{s}]} C_{s'} & \text{else} \end{cases} \quad (2)$$

With this loss function, previously shown categories are now prevented from being forgotten. Nevertheless, it is still necessary to ensure that the network learns new categories, which is tackled next.

b) 2. Loss on the current categories L^{curr}:: In order to learn the categories of the current sequence, a cross-entropy loss with softmax is used on the newly generated heads, which is possible as classification labels are used in a one-hot encoding style for the current sequence. The loss on new categories for one training example in the current sequence L^{curr}, can be described as:

$$L_{i,s}^{\text{curr}} = \frac{1}{|C_s|} \sum_{c \in C_s} \mathbb{1}_i[c] \log(p_i[c]) \tag{3}$$

With this loss L^{curr}, our network is able to learn new categories, where $p[c]$ is the softmax output for a class c. To balance those two-loss parts, a third loss is defined next.

c) 3. Loss over all categories L^{all}:: As the goal is to have a complete classifier for all categories, we now have to find a solution for the decoupling of the previous categories from the new ones. Here, also the cross entropy loss with softmax as in Eq. (3) is used, but this time on all logit outputs (all previous and new categories) $C_s' = \bigcup_{s'}^{[0,...,s]} C_{s'}$, where s is the current sequence:

$$L_{i,s}^{\text{all}} = \frac{1}{|C_s'|} \sum_{c \in C_s'} \mathbb{1}_i[c] \log(p_i[c]) \tag{4}$$

Important to note here is that this loss part again uses classification labels and a classification loss. Therefore, this loss function tries to lower the logit response of previous categories, as they are represented with a zero value in the labels, which might lead to forgetting. The recall loss L^{recall} prevents this. Combining the three formula parts, our loss function on all training examples of the current sequence s is defined in Eq. (5). The loss in the first sequence only uses Eq. (3) of the loss function as there are no previous categories given.

$$L_s = \begin{cases} L_{i,s}^{\text{curr}} & \text{if } s = 0 \\ L_{i,s}^{\text{recall}} + L_{i,s}^{\text{curr}} + L_{i,s}^{\text{all}} & \text{else} \end{cases} \tag{5}$$

3) Discrepancy in the output distribution: Continually training the network sequence after sequence causes the logit outputs o to rise steadily, which leads to forgetting. See the blue line in Fig. 3. This is caused as the previous categories try to keep their logit values through the L^{recall}, which means that to learn new categories, the logit values of those categories have to be higher than before. As the softmax uses all categories, this then increases the variance of each sequence. Caused through this unequal comparison inside of the softmax, the older sequences categories will be increasingly forgotten over time. We solve this problem of distribution discrepancy within already learned categories by dividing the difference between the output and the recall label by the variance per category.

$$\sigma[c] = \mathbb{E}[(o[c] - \mu[c])^2], \forall c \in C_s^{\text{prev}} \tag{6}$$

In Eq. (6), the calculation of the variance per category $\sigma[c]$ is shown, where $\mu[c]$ is the mean over the logit outputs o during one sequence s over all the image-label pairs (x_s, y_s)
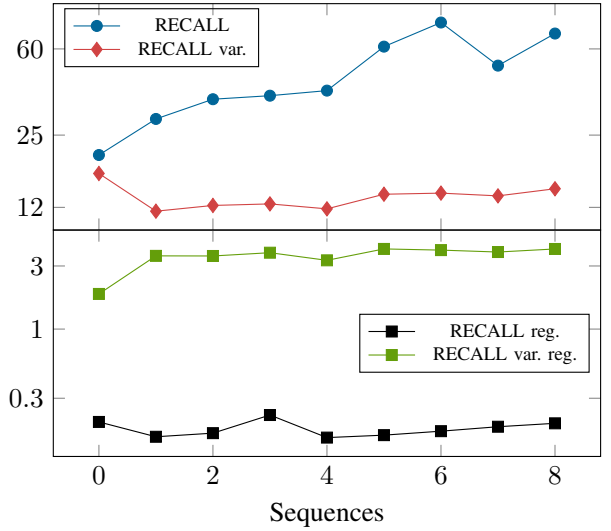

Fig. 3: Logarithmic variance over all logits per sequence and method. In blue, the output variance for the standard RECALL with catastrophic forgetting is shown. In red RECALL using dividing by variance, in black RECALL using full regression loss and in green RECALL using full regression loss in combination with dividing by variance is depicted. CORe50 [10] is used here.

for one class c. So, the loss on the previous categories from Eq. (1) is changed, resulting in a modified loss $\bar{L}^{\text{recall}}$, shown in Eq. (7). Inspired by a Mahalanobis distance, this equation shows the network output difference $o_i[c]$ and the recall label $r_i[c]$ of each neuron being divided by the variance of the respective category $\sigma[c]$. This reduces the discrepancy in the output of the neurons.

$$\bar{L}_{i,s}^{\text{recall}} = \frac{1}{|C_s^{\text{prev}}|} \sum_{c \in C_s^{\text{prev}}} \left(\frac{o_i[c] - r_i[c]}{\sigma[c]}\right)^2 \tag{7}$$

The red line in Fig. 3 shows that this technique smooths the variance.

4) Full regression loss: The division by the variance, however, does not solve the problem entirely as it only tries to patch up the problems caused by the softmax and cross-entropy.

As this is the root cause, we propose to replace the softmax and cross-entropy with a regression loss and clamp the output of the last layer to the range of zero to one, which is shown in Eq. (8).

$$\bar{o}_i[c] = \max(0, \min(o_i[c], 1)) \tag{8}$$

After that, we need to adapt L^{curr} and L^{all} by replacing the cross entropy with an L2 loss, see Eq. (9) and Eq. (10).

$$\tilde{L}_{i,s}^{\text{curr}} = \frac{1}{|C_s|} \sum_{c \in C_s} (\bar{o}_i[c] - \mathbb{1}_i[c])^2 \tag{9}$$

$$\tilde{L}_{i,s}^{\text{all}} = \frac{1}{|C_s'|} \sum_{c \in C_s'} (\bar{o}_i[c] - \mathbb{1}_i[c])^2 \tag{10}$$

Resulting in $\tilde{L}^{\text{curr}}$ and $\tilde{L}^{\text{all}}$, which are now used in Eq. (5). The black line in Fig. 3 depicts that our proposed full regression loss smooths the output distribution best. Nevertheless, we lose the advantage that the sum of all values is one. Finally, we also try using the variance here to shift the focus to categories, which have a high range of values in their corresponding logit output.

IV. HOWS-CL-25 DATASET

In order to better show the strength of our approach, we present a novel synthetic dataset for object classification for continual learning, created with BlenderProc, called **HOWS-CL-25** (**H**ousehold **O**bjects **W**ithin **S**imulation dataset for **C**ontinual **L**earning). BlenderProc, by Denninger *et al.* [21], is a procedural pipeline to generate images for deep learning. Our dataset contains 150,795 unique synthetic images using 25 different household categories with 925 3D models in total, see Fig. 4. We achieved that by creating a room with randomly textured floors, walls, and a light source with randomly chosen light intensity and color. After that, a 3D model is placed in the resulting room. This object gets customized by randomly assigning materials including different textures in order to achieve a diverse dataset. Moreover, each object might be deformed with a random displacement texture. For each RGB-D image, we also provide the corresponding segmentation map and normal image.

The images are organized in five sequences, containing five categories each. We also provide a long-version with twelve sequences containing three categories in the first and two in the following sequences. At the end, ten percent of the images are used for validation, whereas an object instance can either be in the training or in the validation set. This avoids that the network learns to recognize instances of certain categories. We created this dataset by taking 774 3D models from the ShapeNet dataset [22] and the others models from various sites. The dataset and the code are available online: https://github.com/DLR-RM/RECALL.

A. Dataset comparison

In our opinion, one of the most relevant applications for online learning is robotics, where a dataset consisting of household objects is more relevant than out of cars, planes, boats and dogs. This is the most important feature setting HOWS-CL-25 apart from iCIFAR-100 and ImageNet1K, making it similar to CORe50 but with a pronounced focus on category than on instance learning. Compared with CORe50, our dataset contains two and a half times more categories and over 18.5 times more instances, as well as a wider variety of backgrounds, lighting conditions, and camera positions. Furthermore, our dataset is non-handheld, as a CNN could learn the category of the object based on the grasp, which is not ideal if, after training, the approach is used in a general setting [25]. In Tab. I, we show datasets that other continual learning papers have used. This table shows that we provide the widest variety of objects and sessions from all datasets, and we are the only ones providing a segmentation map and

Fig. 4: All 25 categories used in the HOWS-CL-25 dataset are suited for mobile robotics as shown here. For training and testing, different instances of these categories are used.

normal image to each RGB-D image. Here, a session is a specific environment.

Rebuffi *et al.* [7] introduce another dataset that is often used in continual learning, called iCIFAR-100, which is an incremental version of the CIFAR-100 dataset proposed by Krizhevsky *et al.* [26] with 100 categories. This dataset splits the categories equally into different task sets (sequences). The advantage of this dataset is that it contains more categories than CORe50 or our dataset, but the problem is that it is easier for a network, which is pre-trained on ImageNet [5] as those contain similar or even the same categories. On top of that, it is not a household dataset.

OpenLORIS is a robotic vision dataset proposed by She *et al.* [24], which contains household objects recorded by a robot. The difference in our dataset is that OpenLORIS focus on instance learning, as each sequence contains the same 69 objects but in different conditions (occlusion, view change, and more), and additionally, this dataset uses the same instances and sessions for training and testing.

One of the best advantages of a synthetic dataset is that it is only a question of computation time to create an even more extensive dataset, whereas expanding the number of categories or instances in CORe50 is much more time-intensive, as these images are recorded manually. One might be concerned that using synthetic data will not generalize to real-world images. But, as shown from Hodaň *et al.* [27] and Denninger *et al.* [21], a generalization from synthetic to real-world images is possible.

TABLE I: Compared to other datasets used for continual learning, HOWS-CL-25 has the most objects, sessions and sequences.

Dataset	Type	Imgs.	Cat.	Obj.	Sess.	Seq.	Format	Setting
permuted MNIST [23], [16]	Category	70000	10	-	1	10	grayscale	hand written
OpenLORIS [24]	Instance	**1106424**	19	69	7	9	RGB-D	robot camera
CORe50 [10]	Instance	164866	10	50	11	9	RGB-D	hand hold
iCIFAR-100 [7]	Category	60000	**100**	600	600	10	RGB	mixed
HOWS-CL-25	Category	150795	25	**925**	**50265**	5 or **12**	**RGB-D, segmap, normal**	synthetic

V. RESULTS

A. Experimental setup

We evaluate our approach on iCIFAR-100 created by Rebuffi *et al.* [7], on CORe50, proposed by Lomonaco *et al.* [10], and finally on both versions of HOWS-CL-25, our dataset created with BlenderProc. In addition to the rehearsal-free methods (AR1, LwF, EWC, SI), we also compare our approach with iCaRL and A-GEM (see Sec. II) except for the CORe50 dataset, which is not supported out of the box for those approaches. All evaluated approaches use the same features generated by a frozen ResNet50 backbone on RGB input data. This ResNet50 is pre-trained on ImageNet. The only exception is AR1 as this approach actively changes the backbone while training, so using the features is impossible. For AR1, we use the code provided by the authors, which uses a MobileNet as backbone [28]. To improve comparability we also tested our approach with the same MobileNet backbone, shown in Tab. III. For generating the results on LwF, EWC, SI, iCaRL and A-GEM we use code provided by Van de Ven *et al.* [6] and adapt it so that all approaches us the same features.

B. Discussion

The results of our experiments are depicted in Tab. II and Fig. 5. Our approach reaches the best accuracy of all rehearsal-free methods with 71.45% on CORe50, as shown in the top plot of Fig. 5. RECALL even outperforms AR1, which is to the best of our knowledge state-of-the-art for rehearsal-free continual learning on this dataset.

Based on the results, it is quite clear that LwF suffers from catastrophic forgetting. The distribution shift in LwF still seems to be a problem, as the network forgets quite fast even though they use the final output of the softmax as target values. RECALL solves this by using recall labels r_s and a full regression loss. The bad performance of SI and EWC shows that only using an importance matrix seems insufficient. AR1, which combines SI with an architectural strategy, improves the performance by a decent margin.

On iCIFAR-100, RECALL reaches 61.15% in contrast to 42.39% with AR1, even though the implementation provided by the authors of AR1 reaches a better performance on iCIFAR-100 than the approx. 31% reported in their paper [12]. In comparison to the other rehearsal-free learning approaches, we are able to learn new categories without strong forgetting.

On HOWS-CL-25, our approach is also the best performing rehearsal-free method with 57.83% by a big margin, see

third plot of Fig. 5. For the long version of HOWS-CL-25, the best performing RECALL version is the one with full regression loss and dividing by variance with 40.65% compared to the standard one with 36.85%. This shows that replacing the probability distribution in the full regression loss mode works better on more complex tasks. However, in such a scenario with more sequences and complex categories, the limit of our approach is shown, as forgetting can't be prevented entirely, see the last plot in Fig. 5.

The experiments on the HOWS-CL-25 dataset show the advantage of rehearsal strategies, where training data is kept for later sequences. As HOWS-CL-25 contains more objects than CORe50 or iCIFAR-100, which seems to be harder for all rehearsal-free methods. This observation is even more vital for the challenging long version of HOWS-CL-25, see the bottom plot of Fig. 5. Here, it is shown that each rehearsal-free method forgets almost everything in the last sequence. On less challenging datasets like iCIFAR-100, RECALL even performs better than the rehearsal strategies, see the second plot of Fig. 5. It is interesting how iCaRL performs on this dataset, as it first suffers from forgetting but is able to relearn most of the categories in the last sequence. This is only possible because it saves previously seen training examples. Remarkably, RECALL performs almost as well as the rehearsal strategies on both versions of HOWS-CL-25, which shows that using the recall labels r_s helps the network to recall previous categories without using any memory.

The contrast in performance between RECALL and the other methods gets even bigger if one focuses solely on categorical continual learning tasks. Here, the method has to understand the semantic meaning of an object rather than recognizing one specific instance. This is visible for AR1, which is not able to learn the categories of HOWS-CL-25, as the category instances used for testing are different from the ones used for training. By testing RECALL with MobileNet features we find that this is not caused by the used backbone of AR1, see Tab. III. For CORe50, where the instances stay the same, the performance difference of AR1 is smaller.

In addition, RECALL is faster than AR1 as we can directly work on the backbone features. These are saved in a TFRecord file, so one complete run (training and validation) takes roughly three minutes[2]. The one-time conversion of all training images of CORe50 to ResNet50 features takes roughly 6 minutes[2]. In comparison, AR1 adapts the backbone and takes roughly 55 min. for each run of CORe50. For detailed sequence analysis, we refer to the appendix.

---

[1]This value is from [12], in our experiments we reach 57.67%.
[2]Using a NVIDIA GeForce RTX 2080 Ti

TABLE II: RECALL performs best on various datasets, compared to other rehearsal-free approaches, measured by the accuracy over all categories after the last sequence. This average accuracy and std. deviation is calculated over 40 runs.

Approach/Dataset	CORe50 (instance)	iCIFAR-100 (category)	HOWS-CL-25 (category)	HOWS-CL-25 long (category)
LwF [11]	34.14	27.93	25.13	10.29
EWC [16]	43.29	9.67	17.99	7.01
SI [13]	26.77	9.62	16.25	6.96
AR1 [12]	69.48[1]	42.39	8.59	8.41
RECALL	64.57($\pm$0.79)	**61.15($\pm$0.29)**	**57.83($\pm$0.18)**	36.85($\pm$0.23)
RECALL var.	50.36($\pm$1.06)	61.09($\pm$0.31)	55.68($\pm$1.27)	36.48($\pm$0.22)
RECALL reg.	63.13($\pm$2.46)	56.52($\pm$0.23)	57.05($\pm$0.53)	39.19($\pm$1.67)
RECALL var. reg.	**71.45($\pm$0.43)**	56.46($\pm$0.24)	56.82($\pm$0.95)	**40.65($\pm$1.37)**

TABLE III: The accuracies over all categories after the last sequence are depicted for various feature-extractors, strategies, and activation functions. The used hyperparameters in RECALL are marked with a star.

Dataset	Feature-Extractor					Strategy		Activation function	
	Mobile-Net[28]	ResNet50* [1]	ResNet50V2 [29]	Inceptionv3 [30]	Inception-ResNetV2 [31]	Exp. last layer	Adding head*	ReLU [32]	SIREN* [33]
CORe50	71.85	**72.00**	64.04	61.87	51.55	64.44	**72.00**	59.35	**72.00**
HOWS-CL-25	44.47	59.17	59.51	67.66	**69.19**	55.15	**58.31**	**59.17**	58.31

C. Ablation

As RECALL is fast to train, it was possible to do several ablation studies. In the following, we present hyperparameters, which have a strong influence on the results.

a) Feature extraction network: We evaluated the influence of the backbone on the performance, see Tab. III. We use ResNet50 as our default backbone in order to achieve better comparability over all datasets, even though Inception-ResNetV2, proposed by Szegedy *et al.* [31], performs better on the HOWS-CL-25 dataset. As HOWS-CL-25 is a category classification dataset, we assume that more advanced CNNs have an improved accuracy in contrast to CORe50, which is an instance-classification dataset. It can also be seen that ResNet50 proposed by He *et al.* [1] works better than the second version ResNet50V2 also proposed by He *et al.* [29]. On the CORe50 dataset, ResNet50 outperforms every other tested backbone. Other papers like Maltoni *et al.* [12] have partly confirmed these results, where they also found that ResNet50 outperformed GoogLeNet, proposed by Szegedy *et al.* [34] on the CORe50 dataset.

b) Using different heads per sequence: In Tab. III it can be seen that using a different head per sequence instead of only changing the last layer improves the results. Especially for the CORe50 dataset, this can be explained by the fact that separating different sequences reduces the possible influence of new output values on the first layer during training.

c) Activation function: In our approach, different activation functions are tested, see Tab. III. SIREN proposed by Sitzmann *et al.* [33] performs best for both datasets. As SIREN has been successfully used in reconstruction tasks, it is interesting to see that it also works best in a classification scenario [32]. But, as highlighted in their paper, it highly depends on its hyperparameter ω_0. In the case of the standard version of HOWS-CL-25, ReLU [32] performs better.

VI. CONCLUSION

This paper shows that the challenging task of online learning can be efficiently solved by combining several architectural and regularization techniques. We demonstrate SOTA performance on the two datasets CORe50 and iCIFAR-100, in a rehearsal-free setting. Our approach RECALL is able to adapt to new categories by adding a new head per sequence. In order to prevent forgetting, we introduce recall labels. However, the usage of those might lead to a discrepancy in the output distribution, for which we propose two different solutions. First, we propose to normalize the logits outputs by dividing with the variance per category and show top performance on iCIFAR-100. In the second solution, we replace the classification with a regression and outperform AR1 on CORe50. Further, we present a novel dataset for continual learning, especially suited for object recognition in a mobile robot environment, called HOWS-CL-25. On this, we show a strong improvement in comparison to all other rehearsal-free learning methods.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[2] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 699–715.

[3] E. Brachmann and C. Rother, "Learning less is more-6d camera localization via 3d surface regression," in *CVPR*. IEEE, 2018.

[4] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *IEEE International Conference on Machine Learning (ICML)*. PMLR, 2014, pp. 1764–1772.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009.

[6] G. M. Van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019.

[7] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *CVPR*, 2017.

[8] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6467–6476.
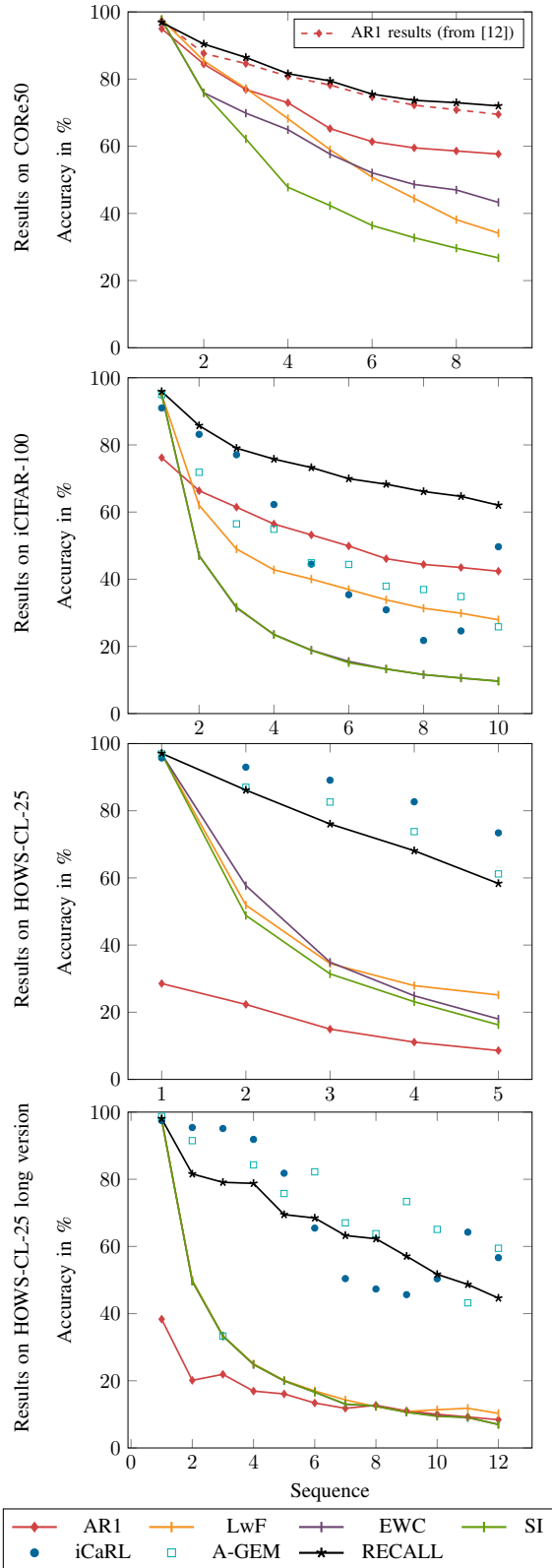
Fig. 5: Results on four datasets. iCaRL and A-GEM are not rehearsal-free and are here to show how strong our approach is. RECALL is SOTA for rehearsal-free learning in all plots. We always use the best run here. For the exact values, see the appendix.

[9] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *Computing Research Repository (CoRR)*, 2016.

[10] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," *Proceedings of Machine Learning Research (PMLR)*, 2017.

[11] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, 2017.

[12] D. Maltoni and V. Lomonaco, "Continuous learning in single-incremental-task scenarios," *Neural Networks*, 2019.

[13] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *ICML*, vol. 70, 2017, p. 3987.

[14] S. Stojanov, S. Mishra, N. A. Thai, N. Dhanda, A. Humayun, C. Yu, L. B. Smith, and J. M. Rehg, "Incremental object learning from contiguous views," in *IEEE CVPR*, June 2019.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[16] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, 2017.

[17] E. Belouadah and A. Popescu, "Il2m: Class incremental learning with dual memory," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[18] A. Ayub and A. R. Wagner, "Storing encoded episodes as concepts for continual learning," *ICML Workshop*, 2020.

[19] A. Ayub and A. Wagner, "Eec: Learning to encode and regenerate images for continual learning," in *International Conference on Learning Representations (ICLR)*, 2021.

[20] M. Denninger and R. Triebel, "Persistent anytime learning of objects from unseen classes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4075–4082.

[21] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodaň, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam, and A. Lodhi, "Blenderproc: Reducing the reality gap with photorealisitc rendering," *Robotics: Science and Systems (RSS)*, 2020.

[22] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv:1512.03012*, 2015.

[23] Y. LeCun, "The mnist database of handwritten digits," 1998.

[24] Q. She, F. Feng, X. Hao, Q. Yang, C. Lan, V. Lomonaco, X. Shi, Z. Wang, Y. Guo, Y. Zhang, *et al.*, "Openloris-object: A robotic vision dataset and benchmark for lifelong deep learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[25] J. Wang, Z. Zhang, C. Xie, Y. Zhou, V. Premachandran, J. Zhu, L. Xie, and A. Yuille, "Visual concepts and compositional voting," vol. 3.

[26] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[27] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. N. Sinha, and B. Guenter, "Photorealistic image synthesis for object instance detection," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 66–70.

[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*. Springer, 2016, pp. 630–645.

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Conference on Artificial Intelligence (AAAI)*, 2017.

[32] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[33] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *NeurIPS*, 2020.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

- Supplemental Material -
RECALL: Rehearsal-free Continual Learning for Object Classification

Markus Knauer[1]   Maximilian Denninger[1,2]   Rudolph Triebel [1,2]

October 7, 2022

1   Introduction

With *RECALL*, we introduce a novel algorithm for rehearsal-free object classification on 2D images for continual learning. This supplemental document includes the quantitative results of the graphs, which are shown in the result section of the paper. For more details, see Sec. 2. In Sec. 3, a study on catastrophic forgetting is provided. Furthermore, we give a more detailed overview of our HOWS-CL-25 dataset in Sec. 4. More ablation study results are presented in Sec. 5. And finally, we also discuss the release of our used hyperparameters in Sec. 6. In addition to this document, a video is provided, making the approach more accessible. For more details for downloading HOWS-CL-25 and running the code, check the accompanying code and README file. Please see the paper for more informations and sources.

2   Quantitative results

In this section, we provide the quantitative results of our experiments on the different datasets from Fig. 5 of the paper. Table 1 shows the accuracy of each tested approach on the CORe50 [10] dataset. In Tab. 2, the results of the iCIFAR [7] dataset are printed. The result of the different approaches on our HOWS-CL-25 dataset are shown in Tab. 3. Finally, the long version of the HOWS-CL-25 dataset is depicted in Tab. 4.

3   Qualitative sequence specific results

In the paper, we examine the average performance of the variants of RECALL over all sequences. In this section, we want to further investigate the performance for the categories of each sequence in each sequence. We therefore provide plots for all four different versions of RECALL in combination with CORe50 [10] (Fig. 2), iCIFAR-100 [7] (Fig. 3), HOWS-CL-25 (Fig. 4), and the long version of the HOWS-CL-25 dataset (Fig. 5). In each plot, it can be seen how the categories of a particular sequence perform in the following sequences. For example, the categories' performance of the first sequence is shown in red during the training. In the second sequence, the green categories are added and tracked throughout the training process. The same is true for the remaining sequences.

4   HOWS-CL-25

The HOWS-CL-25 dataset has two different versions: the standard one with five sequences and a long version with twelve sequences. Both versions contain the same images and categories. In Fig. 6, the stan-

---

Table 1: Comparison of RECALL and other approaches on CORe50 dataset, shown in Fig. 5 of the paper

Approach	Seq 0	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Seq 6	Seq 7	Seq 8
LwF [11]	97.85	85.44	77.22	68.23	58.97	50.78	44.46	38.14	34.14
EWC [16]	97.64	75.93	69.85	64.93	57.66	52.07	48.63	46.99	43.29
SI [13]	97.71	75.95	62.20	47.79	42.33	36.41	32.76	29.65	26.77
AR1 [12]	95.01	84.48	76.89	72.95	65.26	61.37	59.52	58.59	57.67
AR1 results from [12]	97.34	87.64	84.62	80.84	78.23	74.71	72.24	70.85	69.48
RECALL	97.78	91.11	86.15	78.94	72.98	68.19	64.76	66.22	67.01
RECALL var.	97.56	93.0	86.22	75.0	66.67	60.44	51.81	51.53	53.01
RECALL reg.	97.78	90.67	82.52	79.11	74.49	71.15	68.44	67.44	67.33
RECALL var. reg.	96.89	90.44	86.44	81.61	79.47	75.52	73.68	72.97	**72.02**

Table 2: Comparison of RECALL and other approaches on iCIFAR-100 dataset, shown in Fig. 5 of the paper

Approach	Seq 0	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Seq 6	Seq 7	Seq 8	Seq 9
LwF [11]	95.00	62.05	49.00	42.80	40.04	36.95	33.87	31.36	29.87	27.93
EWC [16]	95.00	47.05	31.47	23.58	18.88	15.53	13.30	11.59	10.60	9.67
SI [13]	95.00	47.00	31.70	23.48	18.80	15.13	13.24	11.61	10.55	9.62
AR1 [12]	76.20	66.35	61.46	56.45	53.20	49.90	46.11	44.40	43.50	42.39
RECALL	95.90	85.75	79.03	75.77	73.26	69.93	68.3	66.12	64.68	**62.04**
RECALL var.	94.90	85.15	78.77	75.25	72.80	69.50	68.04	65.89	64.38	61.76
RECALL reg.	95.20	85.50	78.87	74.27	69.72	64.78	63.3	61.33	59.56	57.30
RECALL var. reg.	94.80	85.90	78.73	74.25	69.68	64.55	63.17	61.26	59.64	57.39
A-GEM [8]	95.00	71.85	56.50	54.90	44.90	44.40	37.93	36.94	34.85	25.83
iCarl [7]	91.00	83.15	77.07	62.25	44.50	35.40	30.90	21.77	24.61	49.68
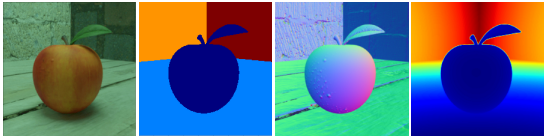
Figure 1: Each RGB image of an object in the HOWS-CL-25 dataset (left) has a corresponding segmentation map (second from left), a normal image (third from left), and a depth image (right).

Table 3: Comparison of RECALL and other approaches on HOWS-CL-25 dataset, shown in Fig. 5 of the paper.

Approach	Seq 0	Seq 1	Seq 2	Seq 3	Seq 4
LwF [11]	96.91	51.91	34.49	27.91	25.13
EWC [16]	96.90	57.72	34.91	24.90	17.99
SI [13]	96.91	48.83	31.42	23.12	16.25
AR1 [12]	28.54	22.33	14.97	11.09	8.59
RECALL	95.71	87.62	78.58	69.73	58.21
RECALL var.	97.07	86.15	76.06	68.10	**58.34**
RECALL reg.	97.05	84.01	73.92	66.85	58.13
RECALL var. reg.	96.73	84.57	74.12	66.94	58.31
A-GEM [8]	96.91	87.00	82.60	73.78	61.18
iCarl [7]	95.71	92.98	89.09	82.66	**73.41**

dard version is shown. Here in each sequence five new categories are introduced to the method. These categories are only available during this one sequence and are not stored for later usage. Furthermore, we use different instances of the same category for testing and training, requiring the method to understand the semantic meaning of a category, rather than just recognizing a certain instance of a category.

In the long version of the HOWS-CL-25 dataset, we start with three categories in the first sequence and then go down to two categories per sequence for the remaining sequences. This is depicted in Fig. 7. The long version contains twelve sequences in total, which is the highest amount of sequences of all datasets for continual learning, as far as the authors of this work are aware of. Making the learning even more challenging as it firstly contains more training steps, which means more risk to suffer from catastrophic forgetting. On top of that, the comparison in each sequence between the different categories is limited, as each sequence now only contains two different cat-

egories. RECALL tries to combat this with our so-called recall label r.

In contrast to existing datasets like CORe50, the network can not learn the instances by heart or even rely on the scene's background to get information about the object. This is especially difficult as the HOWS-CL-25 even contains categories like fork and spoon, which are similar by design. If one compares the images in sequence eight and eleven of the spoon and fork, one will see that this is particularly challenging. The design decision behind this was to lift continual learning to the next level, overcoming the datasets, where a network could learn based on the environment or a specific instance the matching category, as mentioned in the paper.

Finally, synthetic data makes it easier to randomize the environments, which removes the previous mentioned factor in the evaluation. This is achieved by using over 1000 different materials, which are randomly assigned to the room's floor and walls. These materials also have randomized physical properties, meaning that the specularity and other factors of the materials are randomly changed even to further increase the randomness of the environment.

For each RGB we also provide a depth, normal, and segmentation map. Please find an example in Fig. 1. A complete list of the different categories used in HOWS-CL-25 is shown in Tab. 5.

## 5 Additional ablation studies

5.1 Number of layers

In Tab. 6 the resulting accuracies on CORe50 and HOWS-CL-25 dataset are shown, when different amounts of fully-connected-layers are used. It is shown that the best results for both datasets are achieved when one fully-connected layer is used, which is our default option of RECALL, as more layers do not further improve the results. We speculate the reason for this is that the features from the ResNet50 are already separated well enough that further processing only hurts the overall performance.
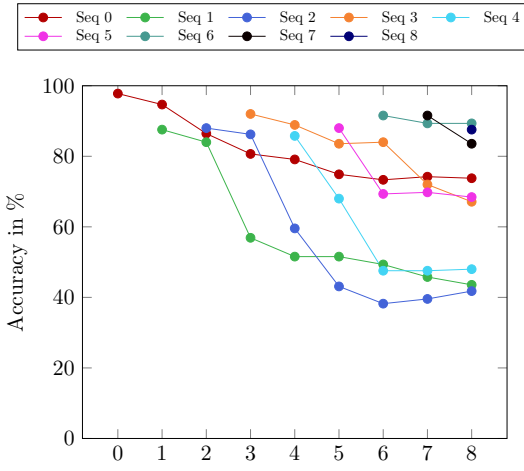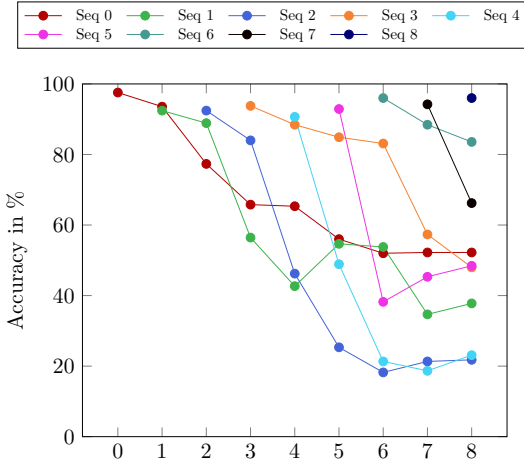
3
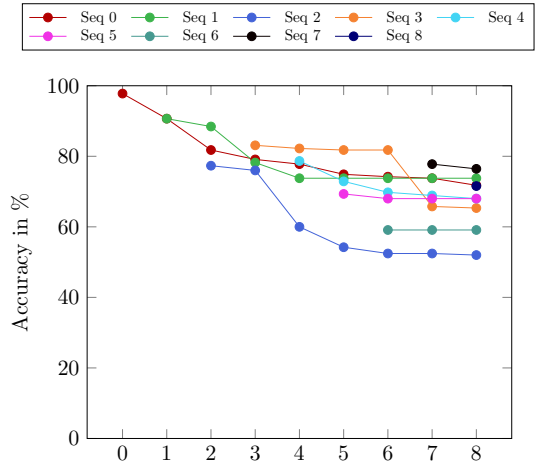
(a) CORe50 with RECALL

(b) CORe50 with RECALL reg.

(c) CORe50 with RECALL var.

(d) CORe50 with RECALL var. reg.

Figure 2: The results on the CORe50 dataset with our four methods.
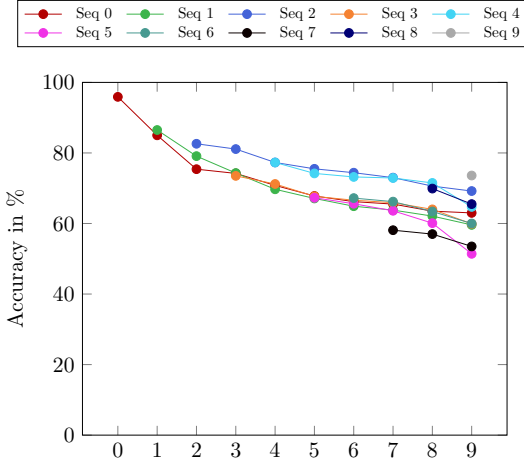
4
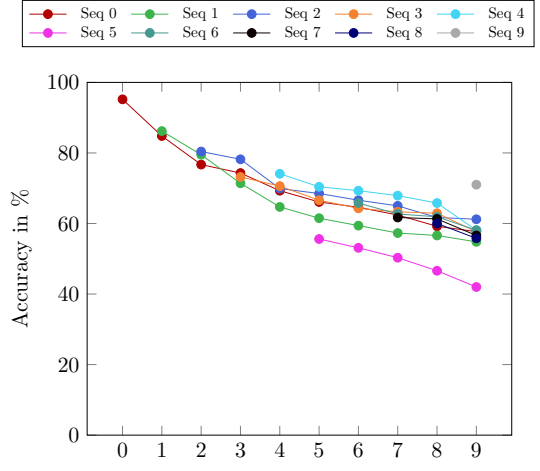
(a) iCIFAR-100 with RECALL

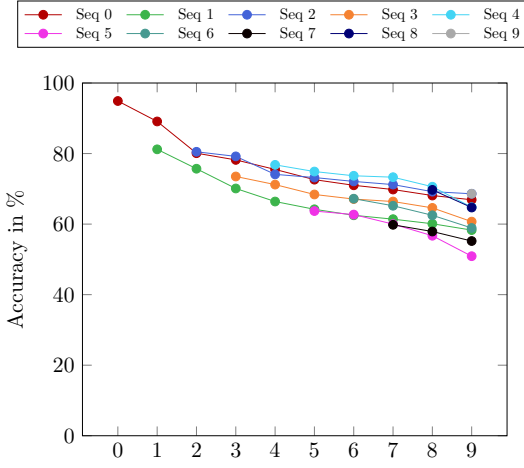(b) iCIFAR-100 with RECALL reg.

(c) iCIFAR-100 with RECALL var.
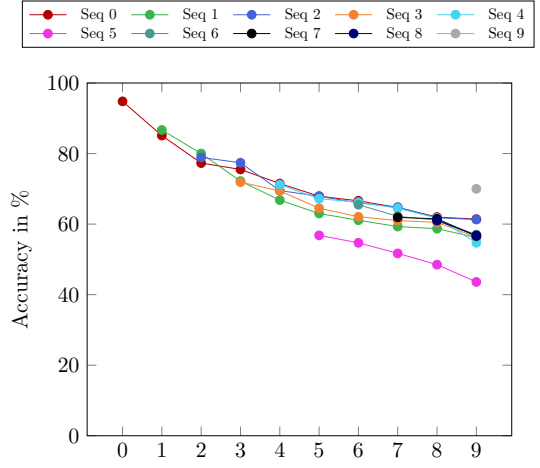
(d) iCIFAR-100 with RECALL var. reg.

Figure 3: The results on the iCIFAR-100 dataset with our four methods.
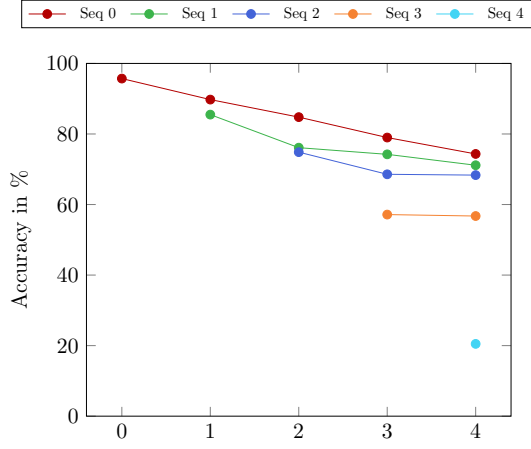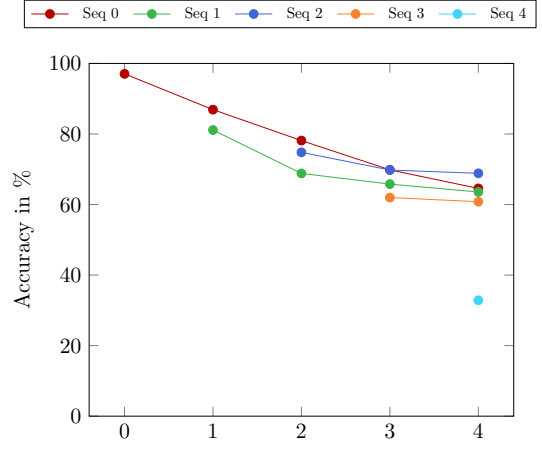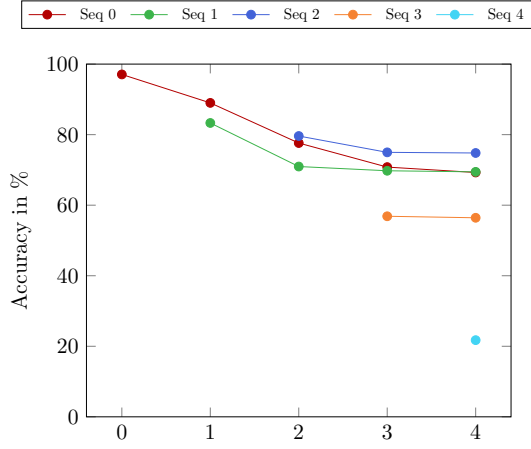
(a) HOWS-CL-25 with RECALL

(b) HOWS-CL-25 with RECALL reg.

(c) HOWS-CL-25 with RECALL var.
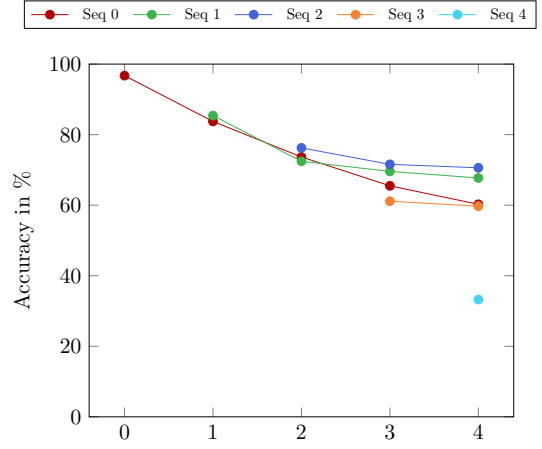
(d) HOWS-CL-25 with RECALL var. reg.

Figure 4: The results on the HOWS-CL-25 dataset with our four methods.
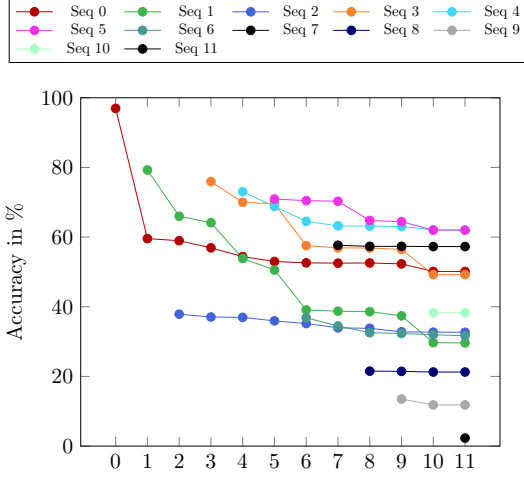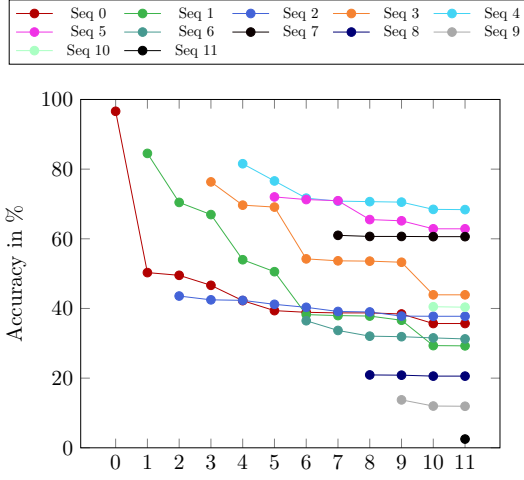
(a) HOWS-CL-25 with RECALL

(b) HOWS-CL-25 with RECALL reg.
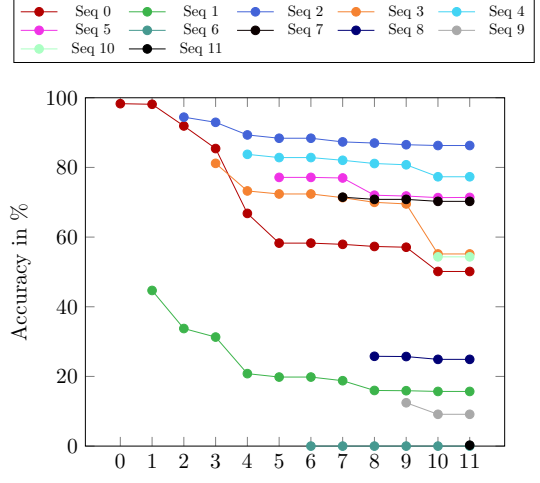
(c) HOWS-CL-25 with RECALL var.
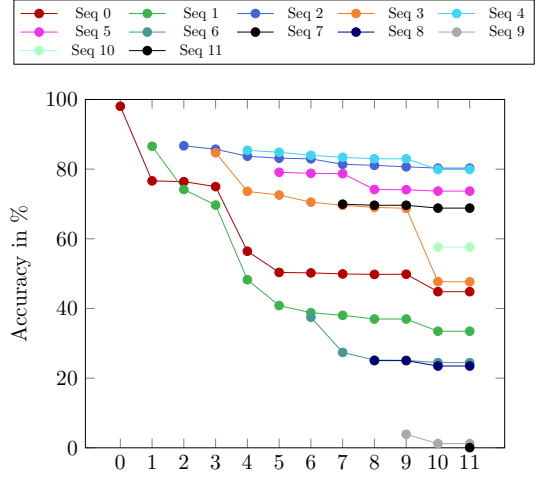
(d) HOWS-CL-25 with RECALL var. reg.

Figure 5: The results on the HOWS-CL-25 long dataset with our four methods.

7

Table 4: Comparison of RECALL and other approaches on HOWS-CL-25 long dataset, shown in Fig. 5 of the paper

Approach	Seq 0	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Seq 6	Seq 7	Seq 8	Seq 9	Seq 10	Seq 11
LwF [11]	97.82	49.90	33.41	25.04	20.11	16.98	14.30	12.29	10.85	11.38	11.82	10.29
EWC [16]	97.82	49.61	33.33	24.84	19.98	16.62	13.00	12.48	10.63	9.62	9.03	7.01
SI [13]	97.82	49.68	33.33	24.87	19.98	16.62	13.10	12.44	10.63	9.40	9.10	6.96
AR1 [12]	38.35	20.12	21.92	16.93	16.09	13.39	11.80	12.70	11.05	10.02	9.24	8.40
RECALL	96.95	69.42	54.26	58.51	57.64	58.14	50.89	50.97	46.8	43.11	40.59	37.36
RECALL var.	96.61	67.41	54.51	58.10	57.96	58.14	50.16	50.74	46.55	42.90	40.30	37.10
RECALL reg.	98.30	71.40	73.36	72.71	66.79	66.47	56.97	58.23	53.33	49.06	46.78	42.90
RECALL var. reg.	98.04	81.59	79.10	78.77	69.49	68.47	63.25	62.30	57.09	51.70	48.68	**44.62**
A-GEM [8]	98.72	91.50	33.30	84.29	75.76	82.22	67.02	63.80	73.36	65.08	43.22	**59.46**
iCarl [7]	97.48	95.42	95.12	91.87	81.80	65.45	50.42	47.32	45.65	50.38	64.25	56.66

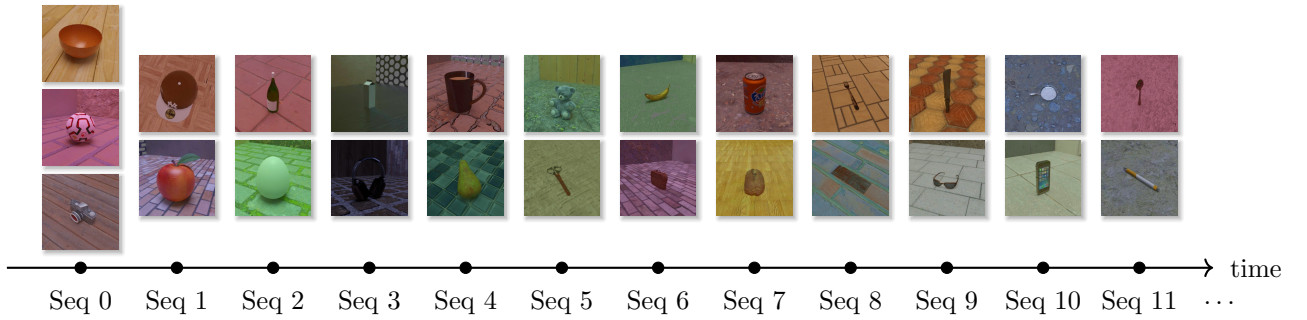Figure 6: Standard version of HOWS-CL-25 dataset with five sequences.

Figure 7: Long version of HOWS-CL-25 dataset with twelve sequences. A particular set of categories is introduced to the network in each sequence, whereas each category contains a set of instances. For example, in the first sequence, the categories camera, ball, and bowl are introduced. Different instances of the same category are used for testing, and these categories are only available in the first sequence.

Table 5: Overview of all categories of our RECALL dataset

Category	HOWS-CL-25	HOWS-CL-25 long
Apple	seq. 0	seq. 0
Bag	seq. 0	seq. 0
Ball	seq. 0	seq. 0
Banana	seq. 0	seq. 1
Bowl	seq. 0	seq. 1
Bread	seq. 1	seq. 2
Camera	seq. 1	seq. 2
Can	seq. 1	seq. 3
Cap	seq. 1	seq. 3
Computer keyboard	seq. 1	seq. 4
Egg	seq. 2	seq. 4
Fork	seq. 2	seq. 5
Glass bottle	seq. 2	seq. 5
Glasses	seq. 2	seq. 6
Headset	seq. 2	seq. 6
Knife	seq. 3	seq. 7
Milk canister	seq. 3	seq. 7
Mobile phone	seq. 3	seq. 8
Mug	seq. 3	seq. 8
Pan	seq. 3	seq. 9
Pear	seq. 4	seq. 9
Pen	seq. 4	seq. 10
Scissors	seq. 4	seq. 10
Spoon	seq. 4	seq. 11
Teddy bear	seq. 4	seq. 11

6 Hyperparameter and source code

The exact hyperparameter used to achieve all the reported results of the different RECALL versions and the implementation of our approach are available in our GitHub repository: https://github.com/DLR-RM/RECALL. The same is true for our HOWS-CL-25 dataset. Which can also be downloaded here: https://zenodo.org/record/7054171.

Table 6: Results of RECALL on different datasets when using different amounts of fully-connected layers. The default option is marked with a star.

Layer amount	CORe50	HOWS-CL-25
1*	**72.02**	**58.31**
2	71.65	52.11
3	70.32	52.21
4	69.95	53.56