Hebbian Continual Representation Learning

Paweł Morawiecki
Polish Academy of Sciences
Institute of Computer Science
pawel.morawiecki@gmail.com

Maciej Wołczyk
Faculty of Mathematics and Computer Science
Jagiellonian University
maciej.wolczyk@gmail.com

Andrii Krutsylo
Polish Academy of Sciences
Institute of Computer Science
akrutsyl@office.ibspan.waw.pl

Marek Śmieja
Faculty of Mathematics and Computer Science
Jagiellonian University
smieja.marek@gmail.com

Abstract

Continual Learning aims to bring machine learning into a more realistic scenario, where tasks are learned sequentially and the independent and identically distributed assumption is not preserved. Although this setting is natural for biological systems, it proves very difficult for machine learning models such as artificial neural networks. To reduce this performance gap, we investigate whether biologically inspired Hebbian learning is useful for tackling continual challenges. In particular, we highlight a realistic and often overlooked unsupervised setting where the learner has to build representations without any supervision. By combining sparse neural networks with the Hebbian learning principle, we build a simple yet effective alternative (HebbCL) to typical neural network models trained via gradient descent. Due to Hebbian learning, the network have easily interpretable weights, which might be essential in critical application such as security or healthcare. We demonstrate the efficacy of HebbCL in an unsupervised learning setting applied to the MNIST and Omniglot datasets. We also adapt the algorithm to the supervised scenario and obtain promising results in the class-incremental learning.

Keywords: Continual Learning, interpretable neural network, Hebbian learning, unsupervised learning

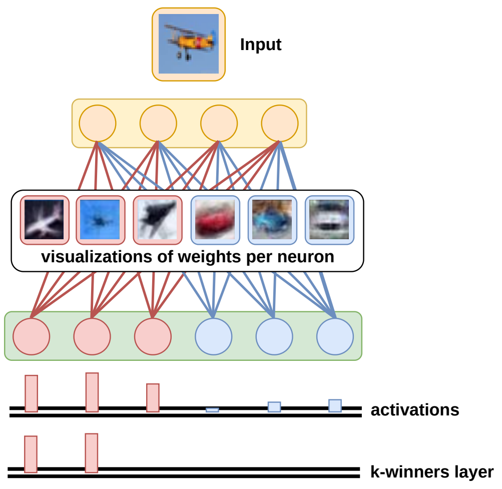

Figure 1. A one-layer network trained by the unsupervised HebbCL algorithm. Each visualization is a 3x32x32 weights of a neuron fully connected to the input layer. The more the neuron's weights are similar to the input sample, the more it will be activated. The final output is obtained by applying k-winners layer to the activation vector. Visualization of weights show that red neurons should recognize airplanes, whereas blue ones activate with a car as an input.

1. Introduction

Humans are able to learn new tasks in a dynamic sequential manner and continuously adapt to new domains. Previously learned knowledge and skills are not only preserved and not forgotten, but the gained experiences are used to build representations for learning future tasks. Although artificial neural networks have been inspired by the biological neural systems, typical learning algorithms fail to remember and reuse learnt knowledge without introducing additional mechanisms Kirkpatrick et al. (2017). To mitigate catastrophic forgetting, neural networks repeatedly process a fixed dataset several times in a shuffled order. If we aim to construct general artificial intelligence, which solves lifelong problems, then such a scenario is highly unrealistic.

Continual Learning (CL) focuses on training neural network models from non-stationary and non-i.i.d. samples, where tasks/classes appear sequentially Van de Ven and Tolias (2019). Several deep learning approaches to CL have been recently introduced with a particular focus on the supervised setting Hsu et al. (2018), Lomonaco and Maltoni (2017), and Van de Ven and Tolias (2019). Although in most cases biological systems are trained without supervision, the unsupervised CL has been mostly overlooked and only a few methods have been examined Lee et al. (2020) and Rao et al. (2019).

In this paper, we address the unsupervised CL scenario, in which labels and boundaries between tasks are not provided. Our model builds a representation in a continual setting, which carries the information about present and past tasks. Obtained representation can be later used for downstream tasks such as clustering or classification.

We develop our algorithm (HebbCL) relying on a few, biologically-inspired principles: sparsity of connections and activations, local Hebbian learning and a dynamic expansion of a network (adding new neurons during the training). Additionally, to limit the catastrophic forgetting, we isolate (freeze) the weights, which carry meaningful information. Although we focus on the unsupervised CL being the most challenging setting, our model can also be applied to the supervised case.

The proposed learning algorithm leads to the interpretable weights of the network that are easy to visualize and inspect. This feature might be essential for some critical applications, including human safety and healthcare.

We conducted extensive experiments to evaluate the properties of the obtained representation in both unsupervised and supervised cases. For the unsupervised setting, we apply HebbCL to the MNIST LeCun and Cortes (2010) and Omniglot Lake et al. (2015) datasets and evaluate the representation with cluster accuracy and k-Nearest Neighbors error. For most settings, we managed to improve the state-of-the-art for these datasets. In the supervised setting, we work in a class-incremental learning setting and show promising results even for a harder dataset such as CIFAR-10 Krizhevsky (2009). Finally, we carry out an ablation study that shows the importance of key steps in HebbCL.

2. Related Work

This work lies at the intersection of Hebbian learning and continual learning and as such we describe related works in both of these areas.

2.1. Hebbian Learning

Although backpropagation has proven to be an efficient method for calculating gradients in artificial neural networks, it is widely believed that it cannot be implemented in the brain Bengio et al. (2015). Because of that, several more biologically plausible alternatives were proposed, many of them inspired by the Hebbian rule of local learning which states that changes in synaptic strength should be only dependent on the presynaptic and postsynaptic neurons Hebb (2005).

The most closely related work and an inspiration for our algorithm is the Hierarchical Temporal Memory (HTM) framework, where both sparsity and Hebbian learning are essential Cui et al. (2017). In our paper, we apply a simplified version of the Krotov-Hopfield learning rule for unsupervised training Krotov and Hopfield (2019). This rule was previously used with success for the image retrieval problem Ryali et al. (2020) and embedding learning Liang et al. (2021).

A separate line of research incorporates Hebbian principle to a typical, gradient-based training Halvagal and Zenke (2022) and Thangarasa et al. (2020).

2.2. Continual Learning

Continual learning (CL) studies how to learn from non-stationary streams of data, where the i.i.d. assumption is not satisfied. This setting approximates learning conditions in the real world, where the environment is constantly changing Hadsell et al. (2020). Although biological systems perform very well in such scenarios, it has been shown that artificial neural networks, which excel in tasks involving i.i.d. data, generally fail in the face of non-stationarity. In particular, the catastrophic forgetting problem appears, i.e. the performance on previous tasks rapidly deteriorates. This is an important difference between human-learning and artificial learning which has to be addressed in case we want to approach human-level training.

In recent years the field of CL has grown considerably and numerous methods for overcoming catastrophic forgetting have been proposed. These approaches can be divided into three major families of methods: regularization-based, replay-based and parameter isolation-based Delange et al. (2021), where each is inspired by different mechanisms of biological brains. Regularization-based methods Aljundi et al. (2018), Li and Hoiem (2017), and Zenke et al. (2017) aim to penalize changes in the network which would reduce the performance on the previously seen tasks. For example, Elastic Weight Consolidation (EWC)

Kirkpatrick et al. (2017), one of the standard CL methods approximates the second-order curvature of the loss function of the previous task and uses it as a penalization term when training on a new task. This approach can be linked to the synaptic mechanisms of the biological brains Wixted (2004). Parameter isolation methods Mallya and Lazebnik (2018), Rusu et al. (2016), and Schwarz et al. (2018) take this approach further by completely freezing parts of the network and introducing modularity to the structure of the model. For example, PackNet Mallya and Lazebnik (2018) freezes a small fraction of weights crucial for each task, and Progressive Networks Rusu et al. (2016) introduce new neurons for each task. Similarly, modularity seems to be an important part of biological neural systems Meunier et al. (2009). Finally, replay-based methods Aljundi et al. (2018), Buzzega et al. (2020), and Chaudhry et al. (2018), Chaudhry et al. (2019) save examples from the past for further retraining which allows them to maintain performance on previous tasks, relating to the concept of human memory.

As such, the field of CL is heavily influenced by biological solutions. However, most of the existing literature focuses on the task of supervised CL Hsu et al. (2018), Lomonaco and Maltoni (2017), and Van de Ven and Tolias (2019), and particularly the setting of image classification. Although there have been some approaches to address other domains and settings Biesialska et al. (2020), Nekoei et al. (2021), and Wołczyk et al. (2021), the research on unsupervised learning is relatively scarce. We argue that this setting is especially important, as humans rarely learn from clearly defined tasks with labels. In practice, biological learning is less structured and lacks supervision. This is why we focus on unsupervised CL. Although the related work here is more scarce, promising approaches include CURL Rao et al. (2019) and CN-DPM Lee et al. (2020), which focus on building mixtures of experts or latent distributions which specialize in parts of the data. In this work, we show that a more biologically plausible approach achieves competitive results in practice.

3. HebbCL Algorithm

Throughout the paper we work in the class-incremental setting, that is classes are exposed to the model sequentially and during the inference time a task ID is not available. For the unsupervised setting — our main point of interest — we do not pass any knowledge of task boundaries to the learning algorithm. This makes the problem much more challenging than the standard benchmarks used for CL Van de Ven and Tolias (2019), which usually provide labels, task ID and the task boundaries.

Algorithm 1 Unsupervised HebbCL algorithm

```
Input: network (weights $W$), minibatch of training examples $X$
Hyperparameters: learning rate $\epsilon$, threshold $t$
Output: updated weights $W$

for all $x_i \in X$ do
    $m \leftarrow \arg\max(Wx_i)$   {identify the highest activation}
    $\Delta w \leftarrow x_i - W_m$   {difference between input and weight vector}
    $W_m \leftarrow W_m + \epsilon\Delta w$   {update weights}
end for
$W_m \leftarrow W_m/\phi$   {normalize weights}

for all row vectors $W_j \in W$ do
    for all $x_i \in X$ do
        $distances[i] \leftarrow d^2(W_j, x_i) \,/\, ||x_i||_1$   {calculate normalized Euclidean distance}
    end for
    $m \leftarrow \arg\min(distances)$   {identify the shortest distance}
    if $distance[m] < t$ then
        freeze $W_j$   {freeze a row vector of weights}
        add a new neuron   {add a new row vector of weights}
    end if
end for

return $W$
```

3.1. Algorithm overview

We develop the learning algorithm following a few biologically inspired principles. The first one is the local Hebbian rule for updating the weights. In a typical gradient-based training all the weights are updated (to a various degree), even those which are essential for previously learned tasks. In the long term this causes the catastrophic forgetting. Having local Hebbian weight changes plus freezing meaningful weights should limit the catastrophic forgetting. We also introduce a dynamic expansion of the network (adding new neurons) to balance frozen neurons and maintain the learning capacity of the network. Finally, we make the representation sparse, which turns out to be useful for downstream tasks such as clustering and classification.

HebbCL builds a representation using a single, wide, fully connected layer. Such the architecture has

Algorithm 2 Supervised HebbCL algorithm

```
Input: network (weights $W$), training examples $X$ from a single class $C_j$
Hyperparameters: learning rate $\epsilon$, EPOCHS, a number of new neurons $n$
Output: updated weights $W$

denote unfrozen neurons as neurons belonging to class $C_j$

for e in range(EPOCHS) do
    for all minibatches $B_k$ do
        for all $x_i \in B_k$ do
            $m \leftarrow \arg\max(Wx_i)$    {identify the highest activation}
            $\Delta w \leftarrow x_i - W_m$    {difference between input and weight vector}
            $W_m \leftarrow W_m + \epsilon \Delta w$    {update weights}
        end for
        $W_m \leftarrow W_m / \phi$    {normalize weights}
    end for
end for

freeze $W$    {freeze all the weights}
expand the network with $n$ new neurons    {add $n$ new row vectors to $W$}
return $W$
```

been proven to be successful for Hebbian learning in other contexts Krotov and Hopfield (2019) and Liang et al. (2021). Hebbian learning can also be applied to deep networks, but the accuracy of the network does not improve or even drops Amato et al. (2019). Any improvements to this research problem would also benefit HebbCL.

From a high-level perspective, HebbCL training consists of two main steps that are executed iteratively:

1. Update the weights W with the simplified Krotov-Hopfield rule
2. If any neuron converged:
    (a) Freeze weights of converged neuron
    (b) Add a new neuron (expand W)

Once the training is over, the representation vector y for the corresponding input vector x is constructed by applying

$$y = f(Wx), \tag{1}$$

where W is a matrix representing network weights and f is a function that zeros all activations but k highest activations (also known as 'k-winners takes all' function).

3.2. Unsupervised HebbCL

We train the network in a CL fashion, without any knowledge of boundaries between classes of training examples. Training examples are processed in minibatches and these steps are executed for each minibatch. Steps 2(a) and 2(b) are conditional, that is, weights are being frozen and a neuron is added only when certain conditions are matched. We give more details on this later in the section.

The weights are updated by a local Hebbian rule. We do not use any labels or a global cost function typical for gradient-based optimization such as the backpropagation algorithm. Concretely, we apply the (slightly simplified) Krotov-Hopfield rule Krotov and Hopfield (2019). First, for a given training example x, we calculate the activation vector $a = Wx$ and identify the index m of the most active neuron a_m, i.e.:

$$m = \underset{m'}{\arg\max}\,(Wx)_{m'}. \tag{2}$$

Let us denote the weights connected to the most active neuron by W_m (m-th row in the matrix W). Then the weights update from time step i to $i+1$ is defined as follows:

$$\Delta w = x - W_m^i, \tag{3}$$

$$W_m^{i+1} = W_m^i + \epsilon \Delta w / \phi \tag{4}$$

where ϵ is a learning rate and ϕ is a normalization factor equal to the largest absolute value of a weight in W.

What drives the update rule is the difference between the input and the weights, $\Delta w = x - W_m$. As the training proceeds, the difference Δw becomes smaller and weights W_i converge towards a particular input pattern.

To limit the catastrophic forgetting, we conditionally freeze some weights. Intuitively, we want to freeze those rows in W, for which the convergence to an input pattern is achieved and $\Delta w = x - W_j$ is very small. We calculate the squared Euclidean distance between x and each W_j and normalize it by the l_1 norm of x.

$$distance = d^2(W_j, x) \,/\, ||x||_1 \tag{5}$$

If $distance$ is smaller than a threshold t (hyperparameter), then we freeze the weights W_j and add a new neuron to the representation y. The threshold t is determined experimentally on a validation set.

New neurons are added to maintain the learning capacity of the network, otherwise there would be no trainable weights at some point of the training. Algorithm 1 shows the complete learning algorithm for a given minibatch.

Loops in the algorithm can be efficiently parallelized and implemented as a GPU-friendly code.

3.3. Supervised HebbCL

The proposed algorithm can be easily adapted to the supervised setting. If labels are known during training, they can be used to simplify freezing and expansion steps.

Let us recall that in the class-incremental setting, the classes/tasks appear sequentially. Once the learning of a given class C_j is over, we freeze all the weights and add a new portion of neurons for an upcoming class. This is in contrast to the unsupervised variant, where we do not have information about classes in training. To maintain the learning capability, we conditionally freeze and expand only one neuron at a time. Algorithm 2 shows the pseudocode for the supervised HebbCL.

During the training subsequent group of neurons are assigned to a given class. At the inference time, we sum activations for each group of neurons and the predicted class is the one with the highest total value.

4. Experiments

4.1. Continual Unsupervised Representation Learning

To examine the proposed algorithm, we follow the evaluation protocol proposed in Rao et al. (2019), where they introduced a method for the continual unsupervised representation learning called CURL. Following this paper, we test HebbCL on two datasets: MNIST LeCun and Cortes (2010) and Omniglot Lake et al. (2015). For MNIST, we use the default (LeCun et al.) split between the training and test sets. For Omniglot we treat 50 alphabets as 50 classes, where for each character (letter) 15 examples are given to the training set and 5 to the test set.

We compare the representations constructed by HebbCL and CURL. For CURL, we report the best results for two variants, that is, with and without a generative replay. We investigate whether the tested methods cluster the same classes together in the representation space. For that purpose, we use two metrics to evaluate the representation: cluster accuracy and k-Nearest Neighbours (k-NN) error. The cluster accuracy is calculated as follows. First, we cluster the obtained representations of the input dataset with the k-means algorithm. Then, we assign each cluster to its most represented class (true labels are available). With these newly assigned (predicted) labels and true labels we calculate the cluster accuracy for the test set. k-NN error is a percentage of wrongly classified examples from the test set by the k-NN classifier.

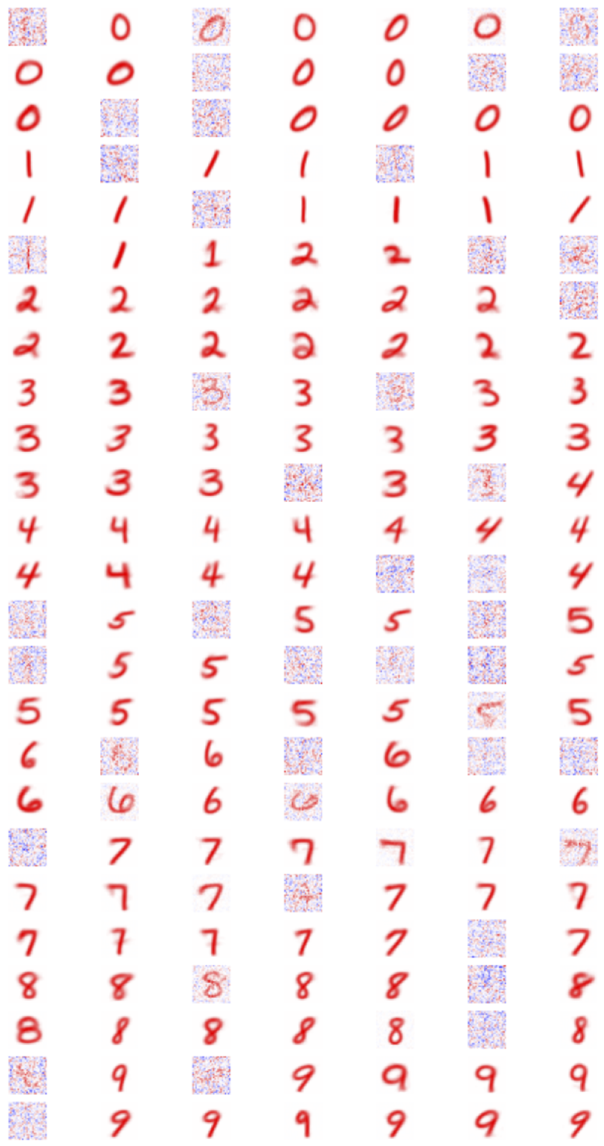
Figure 2. Visualization of weights trained on MNIST. Each square represents 28x28 weights related to a given neuron.

Table 1 summarizes the results. As expected, Omniglot, with 50 classes and much higher sample variance, turns out to be a more challenging dataset than MNIST. We obtain better results than CURL for Omniglot for both metrics. For MNIST, CURL and HebbCL reach similar performance. We also stress that the memory requirements for our algorithm are a few times lower than for CURL. HebbCL uses smaller network, e.g., for MNIST a single hidden layer with 500 neurons.

Fig. 2 shows the weights visualization after the

Table 1. The average cluster accuracy and 10-Nearest Neighbors (10-NN) error across all five tasks of the split MNIST and Omiglot protocols, evaluated after learning the whole sequence. Each value is the average of five runs (with standard deviations). For CURL the average number of clusters is not an integer, because it was found by the algorithm.

Benchmark	**MNIST**			**Omniglot**		
Method	# clusters	Cluster acc (%)$\uparrow$	10-NN error (%)$\downarrow$	# clusters	Cluster acc (%)$\uparrow$	10-NN error (%)$\downarrow$
HebbCL (our)	25	$74.23_{\pm 0.67}$	$6.48_{\pm 0.20}$	50	$14.07_{\pm 0.34}$	$71.62_{\pm 0.46}$
	50	$78.35_{\pm 0.73}$	$6.48_{\pm 0.20}$	100	$16.46_{\pm 0.25}$	$71.62_{\pm 0.46}$
CURL w/ MGR	$25.20_{\pm 2.23}$	$77.74_{\pm 1.37}$	$6.29_{\pm 0.50}$	$101.20_{\pm 8.45}$	$13.21_{\pm 0.53}$	$76.34_{\pm 1.10}$
CURL w/o MGR	$55.80_{\pm 1.94}$	$45.35_{\pm 1.50}$	$17.46_{\pm 1.25}$	$189.60_{\pm 9.75}$	$13.36_{\pm 1.06}$	$81.91_{\pm 1.36}$

Table 2. The average accuracy across all five tasks of the split MNIST and split CIFAR-10 protocols, evaluated after learning the whole sequence of tasks. Each value is the average of five runs (with standard deviations).

Method	MNIST	CIFAR-10
SGD	19.01 ± 0.04	15.56 ± 5.08
L2	18.88 ± 0.18	16.31 ± 3.52
EWC	18.90 ± 0.06	11.83 ± 4.10
Online EWC	18.89 ± 0.07	15.49 ± 5.20
Synaptic Intelligence	17.94 ± 0.57	17.38 ± 4.13
MAS	17.38 ± 4.19	11.79 ± 4.01
LwF	49.37 ± 0.68	13.89 ± 5.33
EfficientPackNet	99.42 ± 0.15	—
HebbCL (our)	93.25 ± 0.38	40.91 ± 0.30

training on the MNIST dataset. Most weights contain meaningful information. Every class is represented and variety of samples is also reflected. Although there are some "noisy" weights, they do not affect the final representations, as their activations are zeroed anyway due to the k-winners layer .

Omniglot has much higher sample variance within a class, which makes the problem more challenging for unsupervised learning methods. Generally, HebbCL captures all the alphabets (classes), however, not every letter/sign can be clearly recognized (see Fig. 3).

4.2. Continual Supervised Learning

We evaluated the supervised HebbCL on the split MNIST and split CIFAR-10 benchmarks, where the data are divided into five tasks, each classifying between two classes, and the model is trained on each task sequentially. At the inference time, we do not provide a task identification, that is we work in the class-incremental learning scenario Masana et al. (2020), which is the most challenging setting for supervised CL.

Since our method does not use a replay buffer or any generative processes, we restrict comparison to the regularization-based and parameter-isolation methods. For split MNIST HebbCL substantially outperforms the regularization methods and is close to the best parameter isolation approach such as EfficientPackNet Schwarz et al. (2021). CIFAR-10 is a more complex dataset and we use a wider network with a feature vector of 2000 units (compared to 640 for MNIST). Trained weights are still meaningful and easy to assign to a particular class; see Fig. 4. Here, we also obtain substantially better accuracy than regularization-based methods (Table 2). The authors of EfficientPackNet did not provide results for harder benchmarks (such as CIFAR-10) in the class-incremental setting.

4.3. Ablation study

HebbCL have four core elements — Hebbian learning, freezing weights, adding new neurons (expansion) and zeroing all but k highest activations. To assess the impact of these steps, we performed an ablation study for the unsupervised scenario measuring the cluster accuracy and the 10-NN error. A number of clusters for MNIST and Omniglot is 10 and 50, respectively. Table 3 indicates that removing one or several steps from the algorithm reduces the performance of both metrics. Figure 5 visualizes how changes are reflected in the generated representations.

In a variant without expansion, all neurons are initially available for training, which causes the model to exhaust its capacity after the first tasks. The model cannot learn representations of new classes when most neurons are frozen. In a variant without freezing, new classes tend to overwrite the stored representations, leading to forgetting earlier tasks. When both freezing and expansion are disabled, representations become less distinctive Interestingly, omitting the sparse representation (disabling k-winners layer) reduces the cluster accuracy greatly, particularly for MNIST.

For Omniglot, the accuracy drop is less consistent.

Table 3. Ablation study for HebbCL performed in an unsupervised setting. The sign ✓ indicates which steps were used in the experiment - (H)ebbian learning, (F)reezing weights, (E)xpansion and (K)-winners

Steps				MNIST		Omniglot	
H	F	E	K	Cluster acc (%)↑	10-NN error (%)↓	Cluster acc (%)↑	10-NN error (%)↓
✓	✓	✓	✓	$63.09_{\pm 1.53}$	$6.48_{\pm 0.21}$	$14.07_{\pm 0.34}$	$71.62_{\pm 0.46}$
✓	✓		✓	$28.96_{\pm 0.43}$	$8.79_{\pm 0.14}$	$9.30_{\pm 0.19}$	$81.95_{\pm 0.31}$
✓			✓	$48.56_{\pm 0.36}$	$7.99_{\pm 0.20}$	$14.50_{\pm 0.27}$	$79.81_{\pm 0.49}$
✓	✓	✓		$35.62_{\pm 0.14}$	$6.14_{\pm 0.05}$	$12.45_{\pm 0.21}$	$71.92_{\pm 0.53}$
✓				$32.2_{\pm 0.15}$	$6.8_{\pm 0.03}$	$14.42_{\pm 0.16}$	$60.18_{\pm 0.22}$

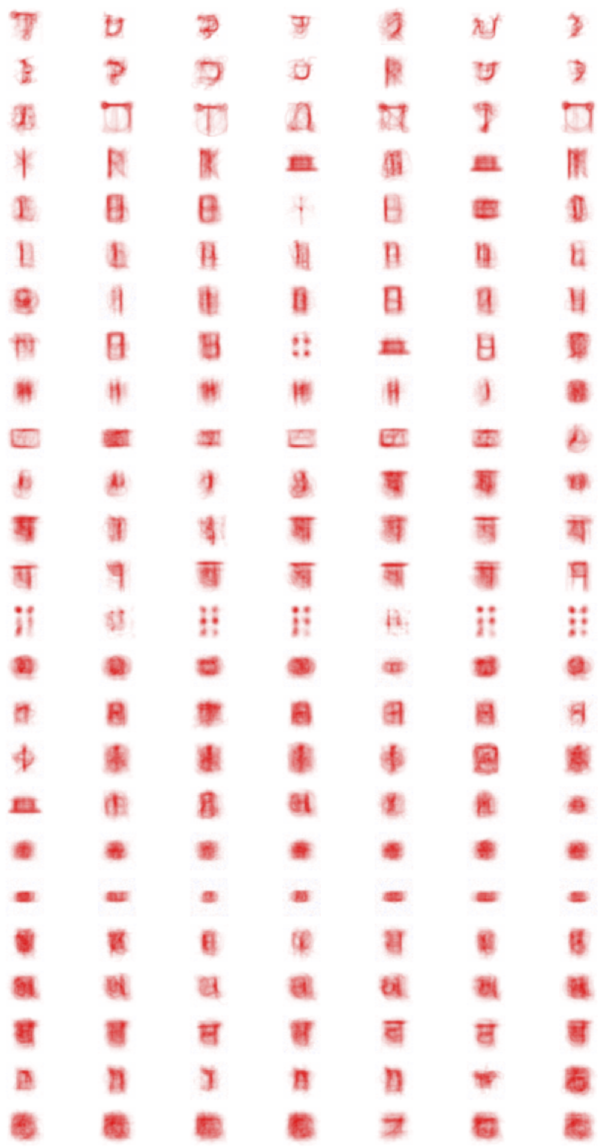
Figure 3. Visualization of weights trained on Omniglot. Each square represents 105x105 weights related to a given neuron.

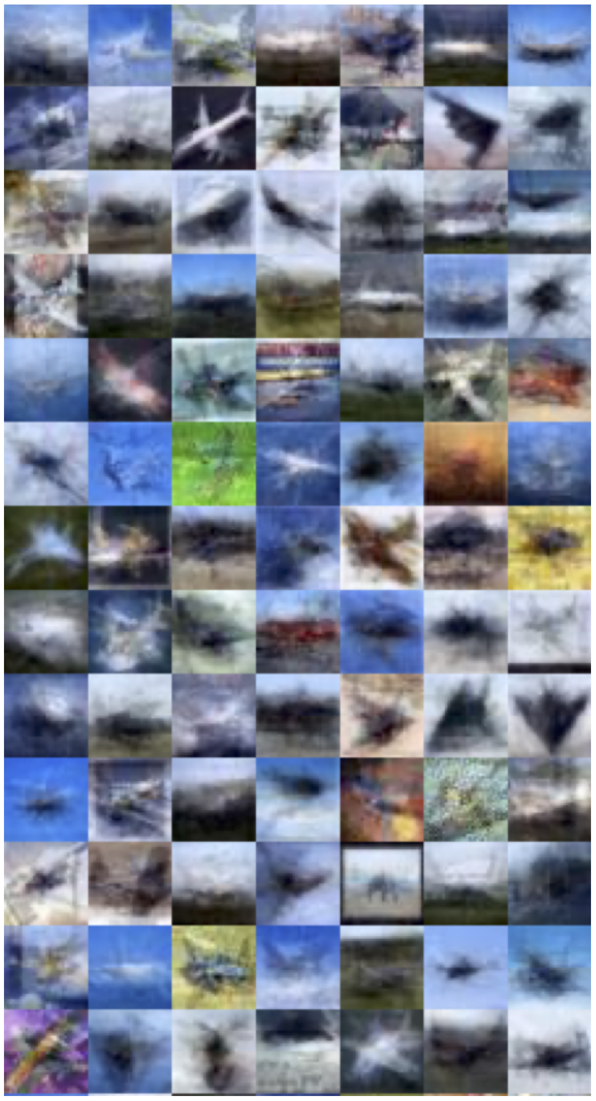
Figure 4. Visualization of weights trained on all tasks of CIFAR-10 in the supervised setting. Each square represents 32x32x3 weights connected to a given neuron. Here we show only neurons related to the first class (airplanes) with possible influence of other classes.

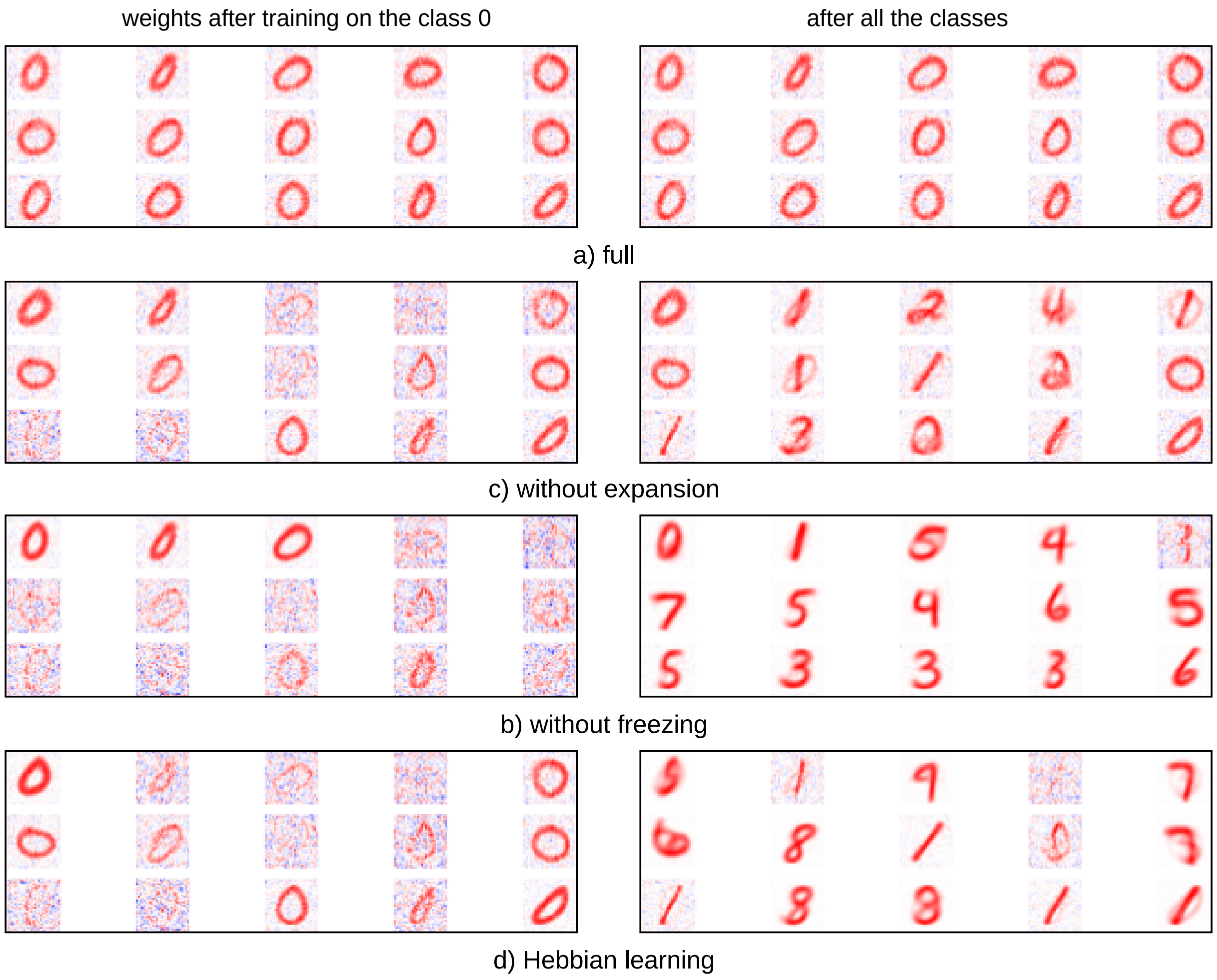

Figure 5. Investigation of catastrophic forgetting for different variants of HebbCL. Figures show visualizations of weights after training on the first class and the very same weights after a complete training on all classes. Clearly, in variants without freezing b) and d) the weights have been forgotten (overwritten) by more recent classes.

We hypothesise it is due to more complex nature of the dataset and the dynamics between elements of the algorithm is less predictable.

5. Conclusion

We developed a method called HebbCL to address the unsupervised CL problem, in which task labels and boundaries are unknown. The algorithm relies on a few biologically inspired principles: sparsity, Hebbian learning and dynamic growth of the network. For MNIST and Omniglot we improve state-of-the-art results in the unsupervised setting. We also adapted the method to the supervised scenario and showed competitive results. The proposed learning algorithm leads to easily interpretable weights, which is an important feature in many critical applications.

We used shallow but wide networks (up to 6 mln parameters). However, for more complex datasets, a deeper network is necessary. The main challenge and an open problem is how to scale HebbCL for deeper networks without losing the desired properties of the CL. We leave this research question for future work.

6. Acknowledgement

This research was carried out within the research project "Bio-inspired artificial neural networks" (grant no. POIR.04.04.00-00-14DE/18-00) within the Team-Net program of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., & Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. *Proceedings of the European Conference on Computer Vision (ECCV)*, 139–154.

Amato, G., Carrara, F., Falchi, F., Gennaro, C., & Lagani, G. (2019). Hebbian learning meets deep convolutional neural networks. In E. Ricci, S. R. Bulò, C. Snoek, O. Lanz, S. Messelodi, & N. Sebe (Eds.), *Image analysis and processing - ICIAP 2019 - 20th international conference, trento, italy, september 9-13, 2019, proceedings, part I* (pp. 324–334). Springer.

Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., & Lin, Z. (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Biesialska, M., Biesialska, K., & Costa-jussà, M. R. (2020). Continual lifelong learning in natural language processing: A survey. *arXiv preprint arXiv:2012.09823*.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., & Calderara, S. (2020). Dark experience for general continual learning: A strong, simple baseline. *Advances in neural information processing systems*, *33*, 15920–15930.

Chaudhry, A., Ranzato, M., Rohrbach, M., & Elhoseiny, M. (2018). Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*.

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

Cui, Y., Ahmad, S., & Hawkins, J. (2017). The htm spatial pooler—a neocortical algorithm for online sparse distributed coding. *Frontiers in Computational Neuroscience*, *11*. https://www.frontiersin.org/article/10.3389/fncom.2017.00111

Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., & Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hadsell, R., Rao, D., Rusu, A. A., & Pascanu, R. (2020). Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, *24*(12), 1028–1040.

Halvagal, M. S., & Zenke, F. (2022). The combination of hebbian and predictive plasticity learns invariant object representations in deep sensory networks. *bioRxiv*. https://doi.org/10.1101/2022.03.17.484712

Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.

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*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, *114*(13), 3521–3526.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

Krotov, D., & Hopfield, J. J. (2019). Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences*, *116*(16), 7723–7731.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338. https://doi.org/10.1126/science.aab3050

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. http://yann.lecun.com/exdb/mnist/

Lee, S., Ha, J., Zhang, D., & Kim, G. (2020). A neural dirichlet process mixture model for task-free continual learning. *arXiv preprint arXiv:2001.00689*.

Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, *40*(12), 2935–2947.

Liang, Y., Ryali, C., Hoover, B., Grinberg, L., Navlakha, S., Zaki, M. J., & Krotov, D. (2021). Can a fruit fly learn word embeddings? *International Conference on Learning Representations*. https://openreview.net/forum?id=xfmSoxdxFCG

Lomonaco, V., & Maltoni, D. (2017). Core50: A new dataset and benchmark for continuous object recognition. *Conference on Robot Learning*, 17–26.

Mallya, A., & Lazebnik, S. (2018). Packnet: Adding multiple tasks to a single network by iterative pruning. *Proceedings of the IEEE conference* *on Computer Vision and Pattern Recognition*, 7765–7773.

Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., & van de Weijer, J. (2020). Class-incremental learning: Survey and performance evaluation on image classification. https://arxiv.org/abs/2010.15277

Meunier, D., Lambiotte, R., Fornito, A., Ersche, K., & Bullmore, E. T. (2009). Hierarchical modularity in human brain functional networks. *Frontiers in neuroinformatics*, *3*, 37.

Nekoei, H., Badrinaaraayanan, A., Courville, A., & Chandar, S. (2021). Continuous coordination as a realistic scenario for lifelong learning. *International Conference on Machine Learning*, 8016–8024.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., & Hadsell, R. (2019). Continual unsupervised representation learning. *Advances in Neural Information Processing Systems*, *32*.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Ryali, C., Hopfield, J., Grinberg, L., & Krotov, D. (2020). Bio-inspired hashing for unsupervised similarity search. *International Conference on Machine Learning*, 8295–8306.

Schwarz, J., Jayakumar, S. M., Pascanu, R., Latham, P. E., & Teh, Y. W. (2021). Powerpropagation: A sparsity inducing weight reparameterisation. *CoRR*, *abs/2110.00296*. https://arxiv.org/abs/2110.00296

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., & Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. *International Conference on Machine Learning*, 4528–4537.

Thangarasa, V., Miconi, T., & Taylor, G. W. (2020). Enabling continual learning with differentiable hebbian plasticity. *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Van de Ven, G. M., & Tolias, A. S. (2019). Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Wixted, J. T. (2004). The psychology and neuroscience of forgetting. *Annu. Rev. Psychol.*, *55*, 235–269.

Wołczyk, M., Zając, M., Pascanu, R., Kucinski, L., & Miłoś, P. (2021). Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, *34*.

Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. *International Conference on Machine Learning*, 3987–3995.