# Dynamic Capacity Expansion in Continual Learning: The eSECL Approach

Basile Tousside[1,2], Jörg Frochte[1], and Tobias Meisen[2]

[1] Bochum University of Applied Science, 42579 Heiligenhaus, Germany
{basile.tousside, joerg.frochte}@hs-bochum.de
[2] Bergische Universität Wupeprtal, 42119 Wuppertal, Germany
meisen@uni-wuppertal.de

**Abstract.** Humans excel at adapting to constantly changing environments, while artificial neural networks (ANNs) struggle with forgetting under dynamic conditions. Continual learning (CL) research aims to address this issue by enabling neural networks to sequentially acquire new knowledge, balancing stability (retaining past tasks) and plasticity (adapting to new tasks). Recently, promising work has emerged in this area. Notably, Sparsification and Expansion for CL (SECL) introduces a robust framework that sparsely uses a neural network's available capacity and expands it when this capacity is exhausted. We propose an enhanced version of SECL, dubbed eSECL, which addresses two key weaknesses of the original SECL algorithm. First, SECL was primarily investigated using convolutional neural networks (CNNs). Second, it does not properly address how much capacity to add when growing a network. We address the first issue by generalizing the algorithm to both dense neural networks and CNNs. For the second issue, we introduce a robust heuristic that monitors the model capacity at each layer and determines the necessary capacity to add. Experiments on popular CL datasets demonstrate the superiority of eSECL over state-of-the-art methods.

**Keywords:** Deep Learning · Continual Learning · Catastrophic Forgetting.

## 1 Introduction

Supervised machine learning models, including artificial neural networks (ANNs), rely on the assumptions that all data are available simultaneously and that the data are drawn i.i.d. from a stationary probability distribution. These assumptions are often invalid in real-world applications where data distributions change over time or where data is generated sequentially and hence, is not available all at once. For example, an image classification system initially trained to recognize cars (task 1) may need to adapt to new and evolving categories like scooters (task 2) as they become relevant in the dataset. Another example is data privacy. For instance, user data must often be deleted after a certain period of time to ensure compliance with data privacy regulations and to protect user privacy.

This can result in missing data for specific tasks that were previously trained, hindering the system's ability to learn new tasks.

In such scenarios, shifts in the distribution of inputs data over time lead to the forgetting of previously learned tasks when new tasks are learned by the neural network. This phenomenon, known as *catastrophic forgetting* (CF) [13], occurs as the model is updated with new-task gradients. For example, the hidden-layer representations, which encode information about prior tasks, become biased towards the new tasks, making it difficult for the model to accurately retain and apply knowledge from older tasks. Overcoming catastrophic forgetting while limiting computational cost and memory requirements is the focus of *Continual Learning* (CL), also known as lifelong- or sequential learning.

Continual Learning systems enable models to sequentially acquire new knowledge, such as new tasks or classes. The primary challenge in CL is managing the Stability/Plasticity dilemma: Stability refers to the ability to assimilate new information without disrupting the stability of previous knowledge, while plasticity ensures there is sufficient capacity to assimilate new knowledge. An effective CL system should excel in both stability and plasticity during sequential learning.

Stability is often achieved by identifying parameters crucial for past tasks and ensuring they do not change significantly during the learning of new tasks [17]. Sparsification and Expansion-based CL (SECL) [28] addressed this by operating on groups of parameters (entire filters) rather than individual kernel parameters. In this paper, the proposed enhanced SECL (eSECL) extends this notion to dense networks. Specifically, we define the notion of *units* as either a *filter* in a convolutional neural network (CNN) (along with its associated kernel parameters) or a *neuron* in a dense neural network (along with the parameters of its incoming connections). Therefore, to achieve stability, eSECL identifies important units and constrains changes to their parameters, preserving previously learned knowledge.

Existing continual learning research indicates that while improving stability to prevent catastrophic forgetting, a model's plasticity often decreases [15]. This reduction in plasticity hampers the model's performance when learning new tasks, especially in long task scenarios.

In long task scenarios, continual learning methods typically address plasticity by expanding the model's capacity. These approaches are collectively referred to as expansion-based methods. However, most of these methods freeze the old parts of the network, leading to underutilization of these parts [34]. Additionally, many expansion-based continual learning methods continuously introduce new parameters without optimizing the use of existing capacity [35]. SECL addresses these issues in an elegant way by using sparsity to manage network capacity pre- and post-expansion.

Specifically, SECL proposes to sparsely use the available network capacity before considering expansion. Given that neural networks are often overparameterized (see Meyes et al. [20]), they typically have more capacity than needed to learn a given task. This superfluous capacity is then used in SECL to train on future tasks without expanding the model for each task, as is done in popular

methods. A weakness of SECL is that it does not scale the amount of capacity to add at each layer. The enhanced version proposed here addresses this concern by monitoring the model capacity at each layer and determining the necessary capacity to add. This approach ensures that capacity is added in a scalable and efficient manner, enhancing the overall effectiveness of the method.

As mentioned earlier, the central contributions of this paper are twofold: first, it addresses the weakness of SECL being primarily investigated using CNNs and proposes a method that generalizes it for both CNNs and dense networks; second, it introduces a simple yet effective heuristic for automatically detecting the amount of capacity to add at each layer when additional capacity is required to learn new tasks. Both of these contributions are integrated into a robust framework, dubbed enhanced Sparsification and Expansion for Continual Learning (eSECL).

## 2 Related Work

Continual learning has gained increasing interest recently, with research divided into three main approaches: expansion-based, regularization-based, and memory-based. This section focuses on the first approach, which is particularly relevant to this paper. For a comprehensive review, see [36].

**Expansion-based Methods.** These methods address the stability/plasticity dilemma by dynamically expanding the model's architecture for new coming tasks [31,26,27,23]. A notable work is Sparsification and Expansion for CL (SECL) [29]. SECL endows a CNN's training objective with three key measures: stabilizing the network on past tasks, sparsifying it to maintain plasticity for future tasks, and extending it to learn additional tasks once its capacity saturates. The present paper enhances this method by addressing two of its key weaknesses, as described in Section 1.

Another notable method is Feature Boosting and Compression for Class-Incremental Learning (FOSTER) [31]. It combines gradient boosting techniques in two stages: feature boosting and feature compression. In feature boosting, the old model's parameters are frozen, and a new module fits the residuals between the target and the original model's output, enabling adaptation to new tasks. The compression stage manages long-term incremental learning by using a distillation strategy to transfer knowledge from the expanded model to a compact model. As FOSTER expands the network with each new task, the total network grows linearly with the number of tasks, raising scalability issues.

Such scalability concern was addressed by Progress and Compress (P&C) [26], which proposed distilling the new network back to the original one after each task, rather than to another compact model as in FOSTER. One of the main limitations of P&C is its fixed expansion locations.

Dynamic Token Expansion (DyTox) [9] tackles the expansion-based continual learning approach from a different angle, using a transformer architecture. It addresses the challenge of catastrophic forgetting by using a task-specific token

expansion mechanism. The architecture maintains shared self-attention layers across all tasks while using task-specific tokens to generate task-specialized embeddings. However, both P&C [26] and DyTox [9] focus on network stability, often neglecting its plasticity for learning new tasks. eSECL, like SECL, addresses this issue by introducing a sparsity regularizer that enforces a sparse architecture and preserves some of the network's available capacity for future tasks.

Other expansion-based methods [32,14] adopt a sparse architecture similar to SECL and eSECL, but have some drawbacks. For instance, Sparse neural Network for Continual Learning (SNCL) [33] uses variational Bayesian sparsity priors on neuron activations across all layers to selectively activate and utilize sparse neurons for learning both current and past tasks at any stage. However, it relies on full experience replay to effectively guide the learning of these sparse neuron activations in different layers. Sparse Continual Learning on the Edge (SparCL) [32] uses a dynamic masking strategy that is task aware to retain essential weights for both current and previous tasks, especially during transitions, but it assumes the continuous availability of a rehearsal buffer. Compact Picking and Growing (CPG) [14] gradually prunes a small fraction of weights and retrains the residuals to recover the original performance. The stopping criteria is triggered when a certain level of accuracy is reached, which requires prior knowledge of the task difficulty.

**Regularization-based Methods.** These methods use regularization techniques to penalize changes in representations learned from previous tasks [5,30,11,4]. This helps maintain stability by preventing the model parameters from deviating significantly from earlier representations, thereby mitigating the risk of forgetting previously acquired knowledge. A notable example is Elastic Weight Consolidation [17], which pioneered the use of regularization to selectively constrain important parameters, thereby preserving knowledge from earlier tasks. A major drawback of regularization-based methods is their fixed model capacity, which leads to a decrease in the model's plasticity as the number of previously learned tasks increases. On the other side, they often exhibit strong performance in maintaining stability over short task sequences.

**Memory-based Methods.** These methods are among the oldest techniques for addressing continual learning and can easily be combined with others methods. They involve either storing samples from previous tasks or generating them synthetically [12,7,6,22]. Knowledge is preserved by periodically replaying these stored samples along with data from new tasks. However, storing samples demands significantly more resources and computational overhead compared to other continual learning methods. Conversely, generating samples presents its own challenges, as the generative model is prone to forgetting and model collapse, where the model's performance degrades significantly. Furthermore, both storing and generating sample data are often constrained by data protection regulations.

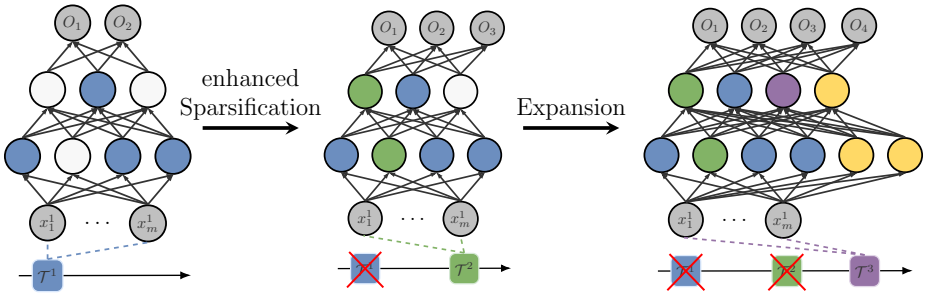



Fig. 1: Overview of eSECL. On the left, all units are initially unimportant. $\mathcal{T}^1$ is trained with the sparsity regularizer according to Eq. (4), identifying units critical for $\mathcal{T}^1$, shown in blue. In the middle, during learning of task $\mathcal{T}^2$, the network trains with both sparsity and stability regularizers to preserve knowledge from the first task; units important for $\mathcal{T}^2$ are highlighted in green. On the right, by the time the task $\mathcal{T}^3$ arrives, almost all units have been marked as important for previous tasks, necessitating the addition of new capacity (shown in yellow) to accommodate $\mathcal{T}^3$. At this stage, the network is further trained with the sparsity regularizer (Eq. (4)) to use the added capacity sparsely. The red "X" marks over $\mathcal{T}^1$ and $\mathcal{T}^2$ indicate the absence of data from these tasks. Adapted from [29].

## 3 Method

Before describing our approach, we briefly introduce the formalism and notation used throughout the paper.

### 3.1 Notation and Formalism

Consider $T$ sequentially arriving, previously unknown supervised learning classification tasks $\{\mathcal{T}^1, \cdots, \mathcal{T}^T\}$. Each task $t$ consist of a set of classes to be learned together and has a training dataset $\mathcal{D}^t_{train} = \{(x^t_s, y^t_s)\}_{s=1}^{m^t}$, where $x^t_s$ and $y^t_s$ represent an instance and its corresponding label among $m^t$ examples. Similary, the test dataset is denoted $\mathcal{D}^t_{test}$. Furthermore, let $l \in \{0, \cdots, L\}$ denote a layer in the neural network, with $N_l$ representing the number of its units. Moreover, $n_{l,i}$ (where $i \in \{1, \cdots, N_l\}$) refers to the $i^{th}$ unit in layer $l$, $\Theta^t_{l,i}$ and $\Omega^t_{l,i}$ respectively denotes its parameters and importance at task $t$. The activation of unit $n_{l,i}$ at task $t$, given an input $x_s$, is denoted by $a^t_{l,i}(x_s)$. $\Theta^t = \{\Theta^t_l\}_{1 \le l \le L}$, where $\Theta^t_l = \{\Theta^t_{l,i}\}_{1 \le i \le N_l}$, represents the neural network parameters for task $t$. Finally, the notations $\mathcal{T}^{1:t}$ and $\Omega^{1:t}$ respectively represent the tasks from 1 to $t$ and the importance of each unit in learning tasks from 1 to $t$.

As mentioned in Section 1, in a dense network, a unit corresponds to a neuron, with its parameters represented by its incoming connections. In convolutional neural networks (CNNs), a unit is represented by a 3D convolution filter, and a connection is denoted by a 2D kernel.

### 3.2 Enhanced Network Stability

**Unit importance at learning $\mathcal{T}^t$** To stabilize the network on previous tasks, minimizing significant deviations in the representations of past tasks while learning new ones is crucial. To achieve this, a measure of parameter importance for each task $t$ must be defined. Inspired by SECL [29], where importance is defined at the convolution filter level rather than at the individual kernel parameter level, we adapt this metric to dense networks.

Specifically, the importance $\Omega^t_{l,i} \in [0,1]$ of the $i^{th}$ unit in layer $l$ at learning task $t$ is quantified as the average standard deviation of the unit's post-activation value over all training samples of task $t$:

$$\Omega^t_{l,i} = \frac{1}{m^t} \sum_{n=1}^{N} \sigma\left(a^t_{l,i}(x_n)\right) \tag{1}$$

where $\sigma(\cdot)$ denotes the standard deviation.

Defining importance at the unit level reduces the complexity of tracking individual parameters and captures the collective influence of parameters within a unit. This approach provides a more holistic and robust assessment of each unit's contribution to the task.

A potential concern is whether the total activation of a unit accurately reflects its significance in learning a task. In CNNs, activations are frequently used to identify which filters are crucial for specific tasks in various contexts, such as neural pruning, see e.g. [3,2], or neural compression like e.g. [10,19]. Numerous empirical studies support this approach. For instance, [16] showed that removing the most active filters significantly degrades performance compared to removing an equal number of randomly selected filters. In this paper, this concept is extended to dense networks, considering neuron importance as defined earlier in this section.

**Unit importance at learning $\mathcal{T}^{1:t}$** Once the importance $\Omega^t_{l,i}$ of unit $i$ in layer $l$ in learning task $t$ only is computed, it is crucial to consider the importance of this unit in learning previous tasks $\mathcal{T}^{1:t-1}$ ($\mathcal{T}^1$ to $\mathcal{T}^{t-1}$) so that it remains stable in performing these tasks. For this purpose, after learning each task $t$, the importance $\Omega^{1:t}_{l,i}$ of the unit $i$ in layer $l$ in learning tasks $\mathcal{T}^1$ to $\mathcal{T}^t$ is updated as:

$$\Omega^{1:t}_{l,i} := \nu\ \Omega^{1:t-1}_{l,i} + \Omega^t_{l,i}, \tag{2}$$

where $\Omega^{1:t-1}_{l,i}$ is the importance at learning tasks $\mathcal{T}^1$ to $\mathcal{T}^{t-1}$ and $\nu$ is a hyperparameter balancing the importance before and after training on task $t$.

After learning each task $t$, the neural network's capacity, represented by its units, can be divided into units important for previous tasks, denoted by their parameters $\Theta_+$, and units unimportant for previous tasks, denoted by their parameters $\Theta_-$.

When learning $\mathcal{T}^t$, the regularizer, which constrains important representations from previous tasks $\mathcal{T}^{1:t-1}$ from changing significantly is then given as:

$$\mathcal{R}^t_{Stability}(\Theta^{1:t-1}, \Theta^t) = \sum_{l=1}^{L} \sum_{\substack{i=1 \\ \Theta^t_{l,i} \in \Theta^{1:t-1}_+}}^{N_l} \Omega^{1:t-1}_{l,i} \, \|\Theta^t_{l,i} - \Theta^{1:t-1}_{l,i}\|_2 \tag{3}$$

### 3.3 Enhanced Network Plasticity

Depending on its initial capacity, a neural network typically exhibits natural plasticity, as not all of its capacity is utilized for learning a given task. In fact, a neural network usually has more parameters than necessary for learning a specific task [20]. Therefore, parameters unused for previous tasks $\mathcal{T}^{1:t-1}$ can be reallocated for a new task $\mathcal{T}^t$. Thus, after training on a task $t$, two different sets of units can be identified:

1. Those crucial for learning task $t$, whose parameters, denoted $\Theta^t_+$, are constrained to ensure stability,
2. Units that are unimportant for learning task $t$, whose parameters, denoted $\Theta^t_-$, are reinitialized and used for future tasks, ensuring the network's plasticity.

To enhance this natural plasticity, eSECL, like SECL, adds a sparsity regularizer to the loss function but generalizes it to both dense networks and CNNs. This regularizer encourages a sparse architecture during training on each task, thereby maintaining sustained sparsity throughout the continual learning procedure. It consists of two components: group sparsity and exclusive sparsity. Group sparsity regularizers encourage feature sharing in the lower layers of the neural network, which is beneficial for capturing representative geometry, while exclusive sparsity promotes feature discrimination in the upper layers, making them more specialized for the task at hand:

$$\mathcal{R}^t_{Plasticity}(\Theta^t) = \underbrace{\sum_{l=1}^{L} \sum_{\substack{f=1 \\ \Theta^t_{l,i} \in \Theta^{1:t-1}_-}}^{N_l} \psi_l \|\Theta^t_{l,i}\|_2}_{\text{Group sparsity}} + \underbrace{\sum_{l=1}^{L} \sum_{\substack{f=1 \\ \Theta^t_{l,i} \in \Theta^{1:t-1}_-}}^{N_l} \zeta \|\Theta^t_{l,i}\|_1}_{\text{Exclusive sparsity}}, \tag{4}$$

where $\psi_l = 1 - \frac{l}{L-1}$ and $\zeta = \frac{(1-\psi_l)}{2}$, where $\psi_l$ adjusts the degree of sharing and discriminating features at each layer $l$, giving more weight to group sparsity at lower layers, while exclusive sparsity dominates at higher layers.

**Connection Rewiring** eSECL leverages connection rewiring for two main purposes: to mitigate negative transfer and to reactivate dead units. Dead units in neural networks are units that no longer respond to any input and produce zero or near-zero activations regardless of the data. This turns them into useless units, as they do not contribute to the learning process or the model's output.

Dead units can occur for several reasons, such as activation functions like the Rectified Linear Unit (ReLU) [24], which output zero for any input less than zero, or because units are unimportant for a task, for which they do not contribute significantly to the learning process and thus receive minimal activation. After training on task $t$, dead units (including unimportant units) are reinitialized to make them active and available for future tasks.

Negative transfer in continual learning refers to a situation where learning new tasks adversely affects the performance on previously learned tasks. This phenomenon occurs when the acquisition of new knowledge interferes with or disrupts existing knowledge. To prevent negative transfer in eSECL, connections from unimportant units to important units are set to zero, ensuring that future changes in an unimportant unit's functionality do not propagate to important units. This is particularly important in dense neural networks, where the dense connectivity can easily lead to disruptions in learned knowledge if not properly managed.

### 3.4 Enhanced Network Expansion

Even with the sparsification scheme presented in the previous section, network expansion may become unavoidable. This is especially true in two scenarios: (i) the number of learning tasks is extremely high. (ii) There is significant variability in tasks. For instance, the features needed to recognize handwritten digits differ significantly from those required to identify objects in images. Consequently, there are two initial situations when starting the training of a new task $t$: (i) the capacity limit has not yet been reached, allowing the available units to be used for learning this task; (ii) the capacity limit has been reached, and further learning will inevitably lead to forgetting, requiring additional units to be introduced.

However, growing a neural network in a continual learning setting presents specific challenges, which can be categorized into three main groups: "scalability", "heuristicity", and "handling additional parameters". These challenges were addressed in SECL for convolutional neural networks. In dense networks, these challenges are even more amplified as the dense connectivity increases the complexity and potential for interference among units.

The remainder of this section discusses each of these challenges in detail as addressed in eSECL for both dense networks and CNNs. Additionally, the new heuristic added in eSECL to determine the amount of capacity to add when expanding the network is examined.

**Scalability** To address the scalability drawbacks (increased training and storage costs) typically encountered with expansion methods, as discussed in Section 2, eSECL applies the sparsity regularizer from Eq. (4) to the parameters $\Theta^{\text{new}}$ added during network expansion. This approach maintains a more efficient and less redundant network architecture, significantly enhancing the scalability of eSECL. This is particularly effective in dense neural networks, where the dense connectivity can otherwise lead to substantial overhead in both computation and

storage. As a result, eSECL can handle an increasing number of tasks without a proportional increase in network complexity or resource consumption, which is crucial for an expansion-based continual learning system.

**Heuristicity** A key challenge in expansion-based CL is determining the optimal timing for network expansion. At the arrival of a task $t$, if previously learned parameters $\Theta^{t-1}$ effectively describe a new task $\mathcal{T}^t$, expansion may not be necessary. However, if they do not, additional capacity is essential. Thus, it is crucial to determine whether the existing capacity is adequate for a new task, requiring a robust heuristic to detect when network expansion is needed.

To achieve this, SECL proposed an expansion heuristic based on a dynamic component of the objective function, specifically the filters' importance vector $\Omega^{1:t}$. The resulting vector $\Omega^{1:t}$ quantifies the significance of each unit (neuron or filter) in the network after learning tasks up to $t$. By focusing on an intrinsic property of the network's learning process, eSECL simplifies the expansion mechanism and reduces the overhead associated with manually adjusting expansion hyper-parameters as done in others methods. The heuristic is outlined in the following three steps:

1. *Importance Thresholding*: After learning task $t$, the importance of each unit in learning tasks 1 to $t$, denoted as $\mathcal{T}^{1:t}$, is computed using Eq. (2).
2. *Layer-specific Evaluation*: Once the important units represented by their parameters $\Theta_+^{1:t} \subset \Theta^{1:t}$ are identified, their distribution across the network layers is assessed. If a layer $l$ has fewer important units $N_{l+}^{1:t}$ than a specified percentage of its total initial units, it indicates that the layer may be undercapacity for handling new task.
3. *Adaptive unit Addition*: Based on this evaluation, additional units $n_{\text{add}}^l$ are introduced into layers where the current capacity is insufficient. The number of added units $n_{\text{add}}^l \subset n_{\text{add}}$ in layer $l$ corresponds to the difference between the initial capacity $n_{\text{init}}^l$ and the remaining capacity $n_{\text{remain}}^l$ at layer $l$.

This process constitutes a heuristic, determining "when" and "where", "how many" additional units to add, thereby ensuring optimal resource allocation and improving overall network performance.

**Handling Additional Parameters** Adding units to a neural network in a continual learning context can degrade the well functioning of the overall network. To avoid this, care muss be taken to properly handle the added units and connections. eSECL addresses this issue effectively, as illustrated in Fig. 2, which depicts the addition of units to two consecutive layers within a dense network. In layer $l$, two units (filled in yellow) have been added, while in layer $l+1$, one unit has been added. Additionally, a specialized unit (dashed in yellow) has been added to the output layer, dedicated to the current task.

This expansion process results in the creation of three distinct parameter matrices, which are central to the network's expanded architecture:

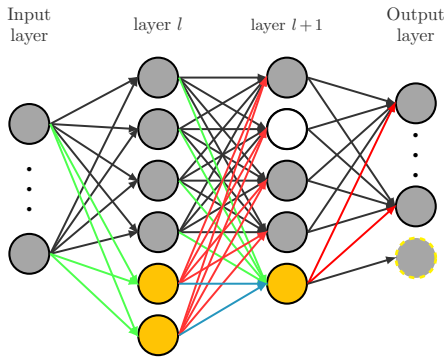



Fig. 2: Schematic overview of network expansion: unimportant units are shown in white, important units are shown in gray, and newly added units are shown in yellow. Adapted from [29].

1. $\Theta^{\mathsf{expand},1} = \{\Theta^{in}_{l-1,l}, \Theta^{in}_{l,l+1}\}$: This set of matrices, shown in green connections, encapsulates the incoming connections from the previous layer to each newly added unit.
2. $\Theta^{\mathsf{expand},2} = \Theta^{in,out}_{l,l+1}$: This matrix, shown in blue, represents the interconnections between the newly added units across the consecutive layers.
3. $\Theta^{\mathsf{expand},3} = \{\Theta^{out}_{l,l+1}, \Theta^{out}_{l+1,l+2}\}$: This set of matrices, shown in red connections, captures the outgoing connections from the newly added units to the existing units in the subsequent layer.

This organized structure allows for the added connections to be handled in an organized way during training. Specifically, $\Theta^{\mathsf{expand},1}$ (green connections) and $\Theta^{\mathsf{expand},2}$ (blue connections) can be trained without any restrictions as they do not have a direct influence on previously established knowledge in the network. However, care must be taken when dealing with $\Theta^{\mathsf{expand},3}$ (red connections). Without proper handling, the information flowing to important units in layers $l+1$ and $l+2$ may change during future training, resulting in catastrophic forgetting of previous knowledge.

To prevent this catastrophic forgetting occurrence, an additional regularization term, $\mathcal{R}_{\text{expand}}$, is introduced into the objective function:

$$\mathcal{R}^t_{\text{expansion}}(\Theta^t) = \sum_{\Theta^{\mathsf{expand},3}_+} \|\Theta^t_{l,i}\|_2^2 \tag{5}$$

This term strongly constrain the magnitude of $\Theta^{\mathsf{expand},3}$ (red connections) specifically on units deemed important for learning past tasks $\{\mathcal{T}^1, \cdots, \mathcal{T}^{t-1}\}$, thereby minimizing any potential interference with the retention of previously acquired knowledge. This strategy provides a balanced approach that allows for network expansion while maintaining the integrity of prior learning, resulting in a robust continual learning system that can adapt its architecture over time without losing its acquired knowledge.

**Combining Stability, Plasticity and Expansion to Enhance CL Efficiency**

To form the final eSECL training objective, the task-specific loss is combined with the stability, plasticity, and expansion regularizers defined above. This results in the following loss function:

$$\begin{aligned}\mathcal{L}^t_{\text{CL}}(\Theta^{1:t}, \mathcal{D}^t_{train}) = \mathcal{L}^t_{\text{task}}(\Theta^t, \mathcal{D}^t_{train}) + \lambda_s \mathcal{R}^t_{\text{Stability}}(\Theta^{1:t-1}, \Theta^t) \\ + \lambda_p \mathcal{R}^t_{\text{Plasticity}}(\Theta^t) + \lambda_e \mathcal{R}^t_{\text{Expansion}}(\Theta^t)\end{aligned} \tag{6}$$

where $\mathcal{R}^t_{\text{Stability}}(\Theta^{1:t-1}, \Theta^t)$, $\mathcal{R}^t_{\text{Plasticity}}(\Theta^t)$, and $\mathcal{R}^t_{\text{Expansion}}(\Theta^t)$ are defined in Eq. (3), Eq. (4), and Eq. (5) respectively. The coefficients $\lambda_s$, $\lambda_p$, and $\lambda_e$ represent the strengths of the stability, plasticity, and expansion regularizers.

When training on the first task, the plasticity regularizer, which is always active, ensures that only a minimal capacity of the network is utilized. The stability and expansion regularizers are inactive since there is no previous task to stabilize and no need for expansion, as the full network capacity is still available. Starting from the second task, the stability regularizer is introduced to balance the retention of knowledge from previous tasks with the integration of new information. The expansion regularizer becomes relevant once expansion is trigerred by the expansion heuristic presented in Section 3.4.

### 3.5 Solving the Resulting Optimization Problem

To minimize the regularized learning objective defined in Eq. (6), we employ a proximal gradient descent (PGD) approach. PGD is suitable for optimization problems with separable objectives that include both smooth and non-smooth components, represented as:

$$\min_{\Theta} g(\Theta) + h(\Theta), \tag{7}$$

where $g(\Theta)$ is convex and differentiable, while $h(\Theta)$ may be non-smooth [25]. The idea is to first take a gradient step on $g(\Theta)$ followed by a "corrective" proximal step to satisfy for $h(\Theta)$. The resulting parameter update after PGD is then given as:

$$\Theta^{k+1} := \text{prox}_{\alpha\ h}\ \left(\Theta^k - \alpha \nabla g(\Theta^k)\right), \tag{8}$$

where $\Theta^k$ is the $k^{th}$ proximal update step and $\text{prox}_{\alpha\ g}$ is the proximal operator. Let rewrite the training objective in Eq. (6) as:

$$\mathcal{L}^t(\Theta^{1:t}, D^t_{\text{train}}) = \mathcal{L}^t_{\text{task}}(\Theta^t, D^t_{\text{train}}) + \mathcal{L}^t_{\text{Reg}}(\Theta^{1:t-1}, \Theta^t), \tag{9}$$

where $\mathcal{L}^t_{\text{Reg}}(\Theta^{1:t-1}, \Theta^t)$ is the regularization loss, which combines the stability, the plasticity and the expansion regularizers.

The proximal gradient iteration as defined in Eq. (8) therefore results in minimizing the task specific (cross-entropy) loss $\mathcal{L}^t_{\text{task}}(\Theta)$ only for one epoch,

with learning rate $\alpha$, and from the resulting solution, applying the proximal operator of the regularizer loss $\mathcal{L}^t_{\text{Reg}}(\Theta)$. Assuming training on task index $t$, which is omitted in the following for the sake of readability, the gradient descent and proximal gradient step in Eq. (8) can then be rephrased for our training objective as:

$$\breve{\Theta}^{k+1} := \Theta^k - \alpha \nabla \mathcal{L}_{\text{task}}(\Theta^k) \tag{10}$$

$$\Theta^{k+1} := \text{prox}_{\alpha\ \mathcal{L}_{\text{Reg}}}(\breve{\Theta}^{k+1}). \tag{11}$$

Since the parameters of the units are non-overlapping (in a dense layer, each neuron has its own unique set of incoming connections, and in a convolutional layer, each filter has its own unique set of weights) the proximal gradient update from Eq. (11) can be applied independently to each unit in each layer.

**PGD Iteration for a Single Unit** Considering the training of a single unit $i$ in layer $l$ at task $t$, where index $t$ is omitted for the sake of readability, the regularization loss term $\mathcal{L}_{\text{reg}}(\Theta)$ from Eq. (9), can be rewritten as:

$$\mathcal{L}_{\text{reg}}(\Theta_{l,i}) = \begin{cases} \lambda_s \Omega^{1:t-1}_{l,i}\ \|\Theta_{l,i} - \Theta^{1:t-1}_{l,i}\|_2 & \text{if } \Theta_{l,i} \in \Theta^{1:t-1}_{+} \\ \lambda_e \|\Theta_{l,i}\|_2 & \text{if } \Theta_{l,i} \in \Theta^{\text{expand},3}_{+} \\ \lambda_p \left(\zeta_l \|\Theta_{l,i}\|_1 + \psi_l \|\Theta_{l,i}\|_2\right) & \text{if } \Theta_{l,i} \in \Theta^{1:t-1}_{-} \end{cases} \tag{12}$$

Consequently, the parameters update rules in Eq. (11) translate for a single unit into the following tree distinct cases (remember that $\breve{\Theta}^{k+1}_{l,i}$ is the result of a standard gradient step, such as SGD, on $\mathcal{L}_{\text{task}}(\Theta^k)$):

– If $\Theta_{l,i} \in \Theta^{1:t-1}_{+}$

$$\begin{aligned} \Theta^{k+1}_{l,i} :&= \text{prox}_{\alpha\ \mathcal{L}_{\text{reg}}}(\breve{\Theta}^{k+1}_{l,i}) \\ &= \Theta^{1:t-1}_{l,i} + \left(1 - \frac{\alpha\lambda_s \Omega^{1:t-1}_{l,i}}{\|\breve{\Theta}^{k+1}_{l,i} - \Theta^{1:t-1}_{l,i}\|_2}\right)_+ \left(\breve{\Theta}^{k+1}_{l,i} - \Theta^{1:t-1}_{l,i}\right) \\ &= \Theta^{1:t-1}_{l,i} + \beta(\breve{\Theta}^{k+1}_{l,i} - \Theta^{1:t-1}_{l,i}) \\ &= \Theta^{1:t-1}_{l,i}\ (1-\beta) + \beta\ \breve{\Theta}^{k+1}_{l,i} \\ &\text{with}\ \ \beta = \left(1 - \frac{\alpha\lambda_s \Omega^{1:t-1}_{l,i}}{\|\breve{\Theta}^{k+1}_{l,i} - \Theta^{1:t-1}_{l,i}\|_2}\right)_+ \end{aligned} \tag{13}$$

– If $\Theta_{l,i} \in \Theta^{\text{expand},3}_{+}$

$$\begin{aligned} \Theta^{k+1}_{l,i} :&= \text{prox}_{\alpha\ \mathcal{L}_{\text{reg}}}(\breve{\Theta}^{k+1}_{l,i}) \\ &= \left(1 - \frac{\alpha\lambda_e}{\|\breve{\Theta}^{k+1}_{l,i}\|_2}\right)_+ \breve{\Theta}^{k+1}_{l,i} \end{aligned} \tag{14}$$

– If $\Theta_{l,i} \in \Theta_{-}^{1:t-1}$

$$\begin{aligned}
\Theta_{l,i}^{k+1} :&= \text{prox}_{\alpha\ \mathcal{L}_{\text{reg}}}\left(\breve{\Theta}_{l,i}^{k+1}\right) \\
&= \text{prox}_{\alpha\lambda_p\zeta_l\|\breve{\Theta}_{l,i}\|_1}\left(\text{prox}_{\lambda_p\psi_l\|\breve{\Theta}_{l,i}\|_2}\left(\breve{\Theta}_{l,i}^{k+1}\right)\right) \\
&= \text{prox}_{\alpha\lambda_p\zeta_l\|\breve{\Theta}_{l,i}\|_1}\left[\left(1-\frac{\lambda_p\psi_l}{\|\breve{\Theta}_{l,i}^{k+1}\|_2}\right)_+\breve{\Theta}_{l,i}^{k+1}\right] \\
&= \{\text{sign}(u_i)\max\{|u_i|-\alpha\lambda_p\zeta_l,0\}\}_{i=1}^{n} \\
\text{with}\quad u_i &= \left(1-\frac{\lambda_p\psi_l}{\|\breve{\Theta}_{l,i}^{k+1}\|_2}\right)_+\breve{\Theta}_{l,i}^{k+1}
\end{aligned} \tag{15}$$

These Equations (13, 14, 15) provide the final update for each unit in each layer. From Eq. (13) it can be observed how full stability (i.e. $\Theta_{l,i}^{k+1} = \Theta_{l,i}^{1:t-1}$) is guaranteed if $\beta = 0$, whereas full plasticity (i.e. $\Theta_{l,i}^{k+1} = 0$) is achieved in Equation (15) if $u_i = 0$.

The complete process for updating the parameters of each unit in every layer is summarized by the following pseudocode:

**Algorithm 1:** eSECL: Numerical Optimization Algorithm

**Input:** learned parameters $\Theta^{1:t-1}$ up to $t-1$,
$\Theta_{+}^{\text{expand},3}$,
learning rate $\alpha$,
regularizer strength $\lambda_s, \lambda_p, \lambda_e$
**Result:** Updated parameters $\Theta^{k+1}$

```
for each epoch k do
    Θ̆^{k+1} = Θ^k − α∇L^t_task(Θ^k)   ▷ Standard GD on task specific loss;
    for each unit i in layer l do
        Compute ψ_l = 1 − i/(L−1)   ▷ Adjust degree of sharing and
                                      discriminating features as given in
                                      Eq. (4);
        Update Θ^{k+1}_{l,i} using Eq. (13), Eq. (14) or Eq. (15);
    end
    Connection rewiring on dead and unimportant units Θ_− ⊆ Θ
end
```

## 4 Experiments

**Continual Learning Scenario:** We follow the typical continual learning scenario, where it is assumed that after training on $\mathcal{T}^t$, $D_{\text{train}}^t$ becomes unavailable, while $D_{\text{test}}^t$ is used to evaluate the model's performance on $\mathcal{T}^t$ after training on $\mathcal{T}^{t'}$, with $t' > t$.

**Datasets:** We conduct experiments on three different datasets, which are popular CL benchmarks: CIFAR-100 consists of 100 classes of $32 \times 32$ images, divided into 20 tasks, with each task containing 5 different classes. Omniglot consists of thousands of handwritten characters from 50 different alphabets, both real and fictional, each with a different number of classes/characters, treated as a sequence of different classification problems. ImageNet-32 is a downsampled ($32 \times 32$) version of the popular ImageNet [8]. It is split into 200 tasks of 5 classes each, making it suitable for evaluating performance over very long sequence scenarios.

**Baselines:** eSECL is evaluated against state-of-the-art regularization-based (fixed capacity) and expansion-based CL methods. Regularization baselines highlight the limitations of fixed-capacity approaches and underscore the need for expansion methods such as eSECL. Specifically, eSECL is compared to EWC [17], and GESCL [30], which are fixed-capacity based as well as CPG [14], P&C [26], FOSTER [31], and DyTox [9], which are expansion-based methods.

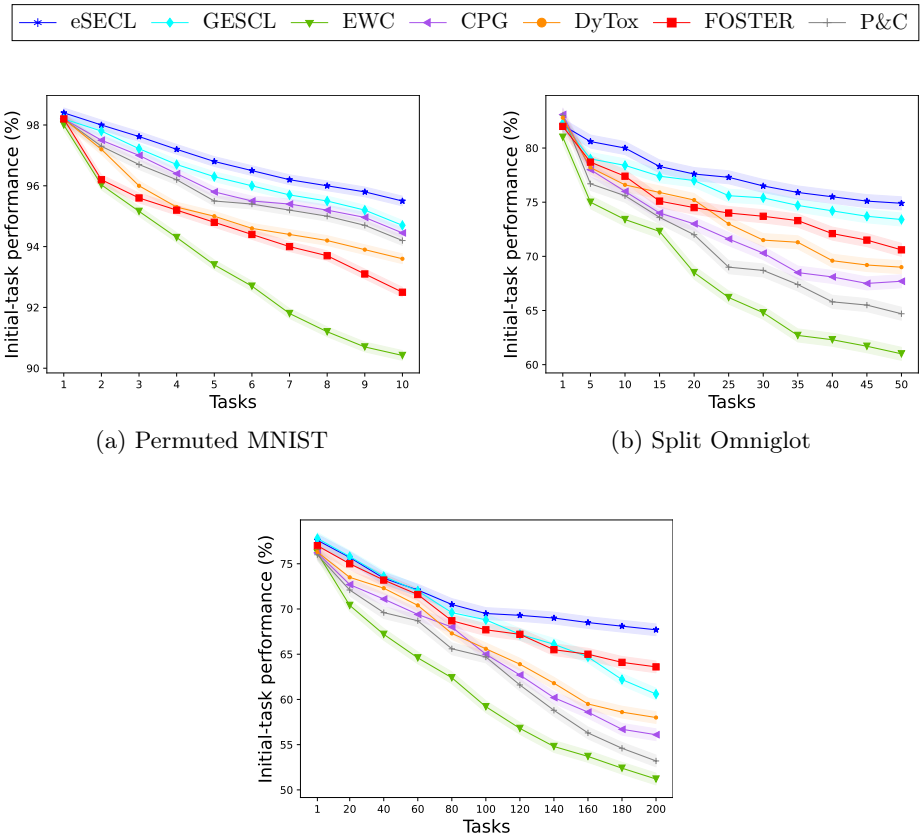
**Network:** In the case of Permuted MNIST, we use a three-layer multi-layer perceptron (MLP) with 400 units each. For Omniglot and ImageNet-32, a scaled-down version of the ResNet18 architecture, similar to that used in [18,1], is adopted as it corresponds to real-world scenarios and is frequently used in CL literature, facilitating meaningful comparisons.

### 4.1 Overcoming Catastrophic Forgetting

As a first experiment, the ability of eSECL to address catastrophic forgetting is investigated. A common measure to evaluate forgetting is to assess how the accuracy of the initial task changes as subsequent tasks are learned. [21]. This is visualized in Fig. 3, which shows the accuracy progression of the initial task across different data sets during the continual learning experiment. Specifically, it shows how the accuracy for the initial task shifts after sequentially learning 9, 49, and 199 tasks across the permuted MNIST, Split Omniglot, and Split ImageNet-32 datasets.

The results allow for several interesting observations. Most importantly, after sequential training on all tasks, eSECL remains the most stable with minimal forgetting, consistently maintaining strong performance on the initial task. This is particularly evident in the challenging Split ImageNet-32 dataset, where eSECL significantly outperforms its strongest competitors, GESCL [30] and FOSTER [31], especially from task 120 onward.

Interestingly, in Permuted MNIST, a domain incremental learning scenario (where each task is drawn from a different domain but all tasks share the same set of classes), the decline in accuracy on the first task is more pronounced compared to shorter task sequences, such as splitting CIFAR-10 into 5 tasks. Furthermore, FOSTER [31] does not perform well on such a domain incremental learning dataset. However, more experiments are needed on domain incremental dataset to support this finding.




(a) Permuted MNIST

(b) Split Omniglot

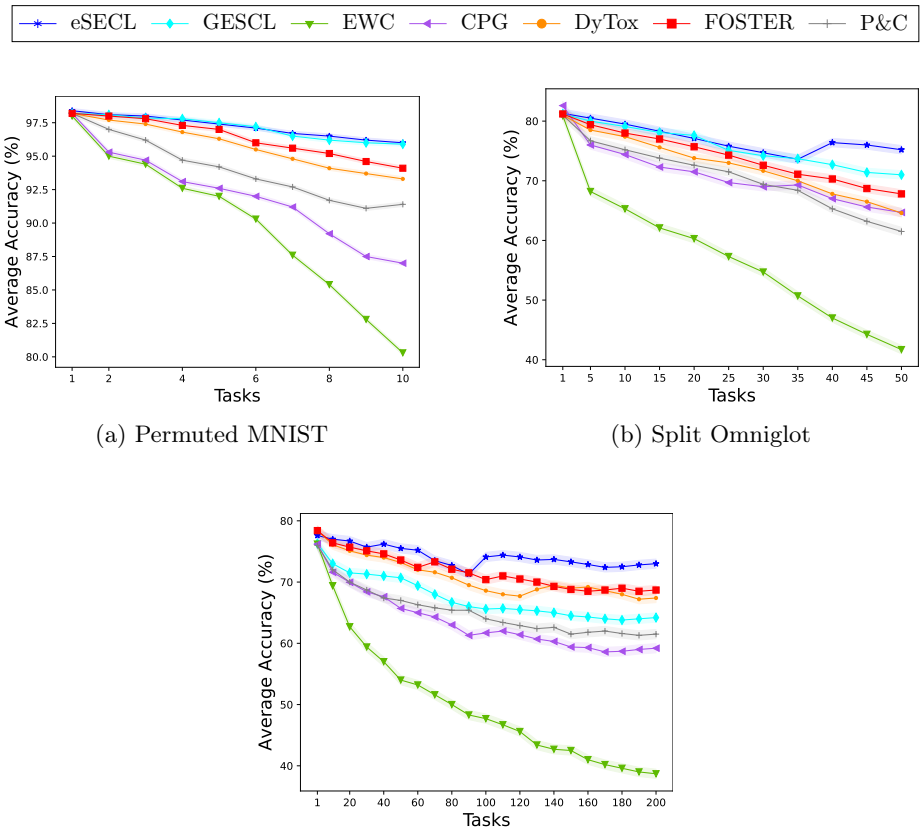
(c) Split ImageNet-32

Fig. 3: Assessing catastrophic forgetting by monitoring performance retention on the initial task across three datasets: Permuted MNIST, Split Omniglot, and Split ImageNet-32. Results show for each dataset how the classification accuracy of the first task evolves as new tasks are introduced. Overall, eSECL exhibits the strongest resistance to catastrophic forgetting, maintaining initial task performance more consistently than other methods.

Another notable observation is that while CPG performs well on Permuted MNIST (which has only 10 tasks), it struggles to retain performance in scenarios with a large number of tasks. This result may be due to Permuted MNIST's domain incremental nature, suggesting that CPG excels in domain incremental scenarios or a combination of both domain incremental and shorter task sequences. This will be investigated in more detail in future research.

Overall, eSECL's ability to maintain high performance on the initial task after learning a large number of tasks is attributed to its effective network expansion strategy, which employs a robust unit importance heuristic to overcome forgetting and strategically expand the neural network as needed.

### 4.2 Overall Classification Accuracy




(a) Permuted MNIST

(b) Split Omniglot

(c) Split ImageNet-32

Fig. 4: Evolution of average test classification accuracy during the continual learning experiments on three datasets. eSECL achieves significantly higher classification results than the competing continual learning methods.

In a second experiment, the effectiveness of eSECL is assessed in terms of classification accuracy during continual learning experiments. To evaluate the classification accuracy, the all-important average accuracy metric [18] is used. Specifically, the classification accuracy at $\mathcal{T}^t$, is the average classification accuracy obtained during testing on tasks $\mathcal{T}^1, \cdots, \mathcal{T}^t$ using $\mathcal{D}^1_{test}, \cdots, \mathcal{D}^t_{test}$ and is defined as:

$$ACC^t = \frac{1}{t}\sum_{t'=1}^{t} a^{t,t'} \tag{16}$$

where $a^{t,t'}$ denotes the model's accuracy on $\mathcal{D}^{t'}_{test}$ after training on task $t$.

Figure 4 illustrates the evolution of the average accuracy across tasks during the CL experiments for 3 datasets: Permuted MNIST, Omniglot, and ImageNet-32. The results indicate that eSECL shows strong performance in terms of average accuracy compared to other baselines on all benchmarks. An interesting observation can be made on ImageNet-32, which consists of 200 tasks and represents a longer sequence than typically studied in the CL literature. At the later stages of this sequence, eSECL exhibits a significant increase in accuracy, allowing it to outperform other expansion baselines. This improvement can be attributed to the strategic heuristic implemented in the eSECL learning procedure, which allows for targeted network expansion precisely where capacity is needed.

An empirical conclusion that can be drawn from Figures 3 and 4 is that eSECL achieves strong overall continual learning results, due to its effective handling of the stability-plasticity dilemma using built-in stability and plasticity regularizers. Additionally, the use of adaptive network expansion mechanisms, based on persistent sparsity in added units, proves to be highly effective.

### 4.3 Ablation Study

Table 1: Average accuracy (ACC) and average forgetting (F) of eSECL variants on Permuted Mnist, Omniglot, and ImageNet-32.

| $\lambda_s$ | $\lambda_p$ | $\lambda_e$ | Permuted MNIST | | Split Omniglot | | Split ImageNet-32 | |
|---|---|---|---|---|---|---|---|---|
| | | | ACC(%) | F | ACC(%) | F | ACC(%) | F |
| ✓ | ✓ | ✓ | 96.14 | 0.008 | 75.22 | 0.085 | 73.56 | 0.091 |
| ✓ | ✓ | | 93.06 | 0.083 | 71.16 | 0.01 | 64.28 | 0.29 |
| | ✓ | ✓ | 53.72 | 0.711 | 33.73 | 0.804 | 15.97 | 0.885 |
| ✓ | | ✓ | 77.55 | 0.048 | 58.9 | 0.053 | 50.4 | 0.069 |
| | | ✓ | 42.9 | 0.715 | 22.16 | 0.806 | 18.9 | 0.941 |
| ✓ | | | 75.04 | 0.12 | 42.70 | 0.144 | 35.8 | 0.206 |
| | ✓ | | 44.83 | 0.711 | 23.8 | 0.792 | 12.7 | 0.905 |

We evaluated the impact of each component in the eSECL framework on the permuted MNIST, Omniglot, and ImageNet-32 datasets. This ablation study specifically examined the effects of the stability regularizer ($\lambda_s$), the plasticity regularizer ($\lambda_p$), and the expansion regularizer ($\lambda_e$) on the base model. Various eSECL variants were created with different combinations of these components, either activated (indicated by ✓) or deactivated (indicated by an empty space). Additional variants of eSECL were generated by deactivating two of the three components simultaneously to assess their combined influence on the model's performance.

The results of the ablation study are presented in Table 1, using two primary metrics for evaluation: "ACC", which denotes the average classification accuracy

at $\mathcal{T}^T$ as described in Section 4.2, and "F", which represents the average forgetting, defined as the average decrease in performance on each task from its peak accuracy to the accuracy after all tasks have been learned [21].

The results for these two metrics, presented in Table 1 highlight the effectiveness of each component in enhancing average classification accuracy and reducing forgetting across tasks. The findings underscore that the interaction between the stability, plasticity, and expansion regularizers is crucial for the superior performance achieved by the eSECL framework.

## 5 Conclusion

This work introduces eSECL, a continual learning method that enhances SECL, a prominent expansion-based CL method primarily studied on CNNs, by generalizing it to both dense neural networks and CNNs. In addition, eSECL develops a robust heuristic to automatically determine the amount of capacity to add in each layer when expanding a network for new tasks. Extensive experiments on various datasets demonstrate the superiority of eSECL over a large variety of reference and state-of-the-art methods.

## References

1. Aljundi, R., Lin, M., Goujaud, B., Bengio, Y.: Gradient based sample selection for online continual learning. Advances in neural information processing systems **32** (2019)
2. Alqahtani, A., Xie, X., Jones, M.W., Essa, E.: Pruning cnn filters via quantifying the importance of deep visual representations. Computer Vision and Image Understanding **208**, 103220 (2021)
3. Anwar, S., Hwang, K., Sung, W.: Structured pruning of deep convolutional neural networks. ACM Journal on Emerging Technologies in Computing Systems (JETC) **13**(3), 1–18 (2017)
4. Asadi, N., Davari, M., Mudur, S., Aljundi, R., Belilovsky, E.: Prototype-sample relation distillation: towards replay-free continual learning. In: International Conference on Machine Learning. pp. 1093–1106. PMLR (2023)
5. Bonicelli, L., Boschini, M., Frascaroli, E., Porrello, A., Pennisi, M., Bellitto, G., Palazzo, S., Spampinato, C., Calderara, S.: On the effectiveness of equivariant regularization for robust online continual learning. arXiv preprint arXiv:2305.03648 (2023)
6. Bonicelli, L., Boschini, M., Porrello, A., Spampinato, C., Calderara, S.: On the effectiveness of lipschitz-driven rehearsal in continual learning. Advances in Neural Information Processing Systems **35**, 31886–31901 (2022)
7. Caccia, L., Aljundi, R., Asadi, N., Tuytelaars, T., Pineau, J., Belilovsky, E.: New insights on reducing abrupt representation change in online continual learning. arXiv preprint arXiv:2203.03798 (2022)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)

9. Douillard, A., Ramé, A., Couairon, G., Cord, M.: Dytox: Transformers for continual learning with dynamic token expansion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9285–9295 (2022)
10. Georgiadis, G.: Accelerating convolutional neural networks via activation map compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7085–7095 (2019)
11. Gurbuz, M.B., Dovrolis, C.: Nispa: Neuro-inspired stability-plasticity adaptation for continual learning in sparse networks. arXiv preprint arXiv:2206.09117 (2022)
12. Hao, J., Ji, K., Liu, M.: Bilevel coreset selection in continual learning: A new formulation and algorithm. Advances in Neural Information Processing Systems **36** (2024)
13. Hayes, T.L., Kafle, K., Shrestha, R., Acharya, M., Kanan, C.: Remind your neural network to prevent catastrophic forgetting. In: European Conference on Computer Vision. pp. 466–483. Springer (2020)
14. Hung, C.Y., Tu, C.H., Wu, C.E., Chen, C.H., Chan, Y.M., Chen, C.S.: Compacting, picking and growing for unforgetting continual learning. Advances in Neural Information Processing Systems **32** (2019)
15. Jung, D., Lee, D., Hong, S., Jang, H., Bae, H., Yoon, S.: New insights for the stability-plasticity dilemma in online continual learning. arXiv preprint arXiv:2302.08741 (2023)
16. Jung, S., Ahn, H., Cha, S., Moon, T.: Continual learning with node-importance based adaptive group sparse regularization. Advances in neural information processing systems **33**, 3647–3658 (2020)
17. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
18. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. In: Advances in neural information processing systems. pp. 6467–6476 (2017)
19. Luo, J.H., Wu, J.: An entropy-based pruning method for cnn compression. arXiv preprint arXiv:1706.05791 (2017)
20. Meyes, R., Lu, M., de Puiseau, C.W., Meisen, T.: Ablation studies in artificial neural networks. arXiv preprint arXiv:1901.08644 (2019)
21. Mirzadeh, S.I., Farajtabar, M., Pascanu, R., Ghasemzadeh, H.: Understanding the role of training regimes in continual learning. Advances in Neural Information Processing Systems **33**, 7308–7320 (2020)
22. Mohr., J., Tousside., B., Schmidt., M., Frochte., J.: Explainability and continuous learning with capsule networks. In: Proceedings of the 13th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KDIR,. pp. 264–273. INSTICC, SciTePress (2021). https://doi.org/10.5220/0010681300003064
23. Mohr, J., Tousside, B., Schmidt, M., Frochte, J.: Multiple additive neural networks: A novel approach to continuous learning in regression and classification. In: IJCCI. pp. 540–547 (2023)
24. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). pp. 807–814 (2010)
25. Parikh, N., Boyd, S.: Proximal algorithms. Foundations and Trends in optimization **1**(3), 127–239 (2014)

26. Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R.: Progress & compress: A scalable framework for continual learning. In: International Conference on Machine Learning. pp. 4528–4537. PMLR (2018)
27. Tousside, B., Friedrichsen, L., Frochte, J.: Towards robust continual learning using an enhanced tree-cnn. In: ICAART (3). pp. 316–325 (2022)
28. Tousside, B., Frochte, J., Meisen, T.: Cnns sparsification and expansion for continual learning. In: ICAART (2). pp. 110–120 (2024)
29. Tousside., B., Frochte., J., Meisen., T.: Cnns sparsification and expansion for continual learning. In: Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART. pp. 110–120. INSTICC, SciTePress (2024). https://doi.org/10.5220/0012314000003636
30. Tousside, B., Mohr, J., Frochte, J.: Group and Exclusive Sparse Regularization-based Continual Learning of CNNs. Proceedings of the Canadian Conference on Artificial Intelligence (may 27 2022), https://caiac.pubpub.org/pub/fgmh6oq8
31. Wang, F.Y., Zhou, D.W., Ye, H.J., Zhan, D.C.: Foster: Feature boosting and compression for class-incremental learning. In: European conference on computer vision. pp. 398–414. Springer (2022)
32. Wang, Z., Zhan, Z., Gong, Y., Yuan, G., Niu, W., Jian, T., Ren, B., Ioannidis, S., Wang, Y., Dy, J.: Sparcl: Sparse continual learning on the edge. Advances in Neural Information Processing Systems **35**, 20366–20380 (2022)
33. Yan, Q., Gong, D., Liu, Y., van den Hengel, A., Shi, J.Q.: Learning bayesian sparse networks with full experience replay for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 109–118 (2022)
34. Yan, S., Xie, J., He, X.: Der: Dynamically expandable representation for class incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3014–3023 (2021)
35. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 (2017)
36. Zhou, D.W., Wang, Q.W., Qi, Z.H., Ye, H.J., Zhan, D.C., Liu, Z.: Deep class-incremental learning: A survey. arXiv preprint arXiv:2302.03648 (2023)