The 35th Canadian Conference on Artificial Intelligence
DOI: 0

Canadian AI 2022
30 May–3 June 2022, Toronto

Group and Exclusive Sparse Regularization-based Continual Learning of CNNs

Basile Tousside*, Janis Mohr, Jörg Frochte
Bochum University of Applied Science, 42579 Heiligenhaus, Germany

Abstract

We present a regularization-based approach for continual learning (CL) of fixed capacity convolutional neural networks (CNN) that does not suffer from the problem of catastrophic forgetting when learning multiple tasks sequentially. This method referred to as Group and Exclusive Sparsity based Continual Learning (GESCL) avoids forgetting of previous tasks by ensuring the stability of the CNN via a stability regularization term, which prevents filters detected as important for past tasks to deviate too much when learning a new task. On top of that, GESCL makes the network plastic via a plasticity regularization term that leverage the over-parameterization of CNNs to efficiently sparsify the network and tunes unimportant filters making them relevant for future tasks. Doing so, GESCL deals with significantly less parameters and computation compared to CL approaches that either dynamically expand the network or memorize past tasks' data. Experiments on popular CL vision benchmarks show that GESCL leads to significant improvements over state-of-the-art method in terms of overall CL performance, as measured by classification accuracy as well as in terms of avoiding catastrophic forgetting.

Keywords: Continual Learning, Catastrophic Forgetting, Machine Learning, Convolutional Neural Networks

1. Introduction

Consider a standard image classification problem, where a CNN is given a data stream explicitly divided into a sequence of T tasks $\{\mathcal{T}_1, ..., \mathcal{T}_T\}$ with $D^t = \{x_n^t, y_n^t\}_{n=1}^{N^t}$ being the dataset of the t^{th} task and x_n^t, y_n^t respectively an instance and label among N^t examples. In a typical machine learning setup, the model is trained using data from all tasks at once. However, in real-world applications, additional policies have to be considered. In data privacy for example, it is required that old data is deleted after a given time to respect users's privacy concerns. This leads to a situation where data from old tasks is unavailable when learning future tasks. Another example is an online learning setting where data from new tasks arrive on-the-fly.

In such scenarios where the CNN can not access all training data at once, a performance drop occurs on the previous tasks, this phenomenon is referred to as *catastrophic forgetting* [1]. Overcoming catastrophic forgetting while limiting the models capacity, computational cost and memory footprint is the focus of continual learning also termed as lifelong- or sequential learning.

The main challenge in overcoming catastrophic forgetting is to resolve the *stability-plasticity dilemma* [2], which describes the problem of adapting to non-stationary data while at the same time preventing forgetting and is a common issue for both biological and artificial neural networks. When an artificial learning model is presented with data in sequence, it needs (i) stability to remember how to solve earlier tasks without re-training on previous data while it also requires (ii) plasticity to acquire new knowledge i.e., to learn new tasks.

If the model is trained without any immunity against the forgetting of past tasks, therefore focusing on the current task only, it will be very plastic but not stable, meaning that it

*basile.tousside@hs-bochum.de

can learn fast but also forgets quickly. On the other side, if the network mainly focus on being immune against forgetting, which is often achieved by identifying and freezing neurons or weights that were important in learning old tasks, it might lack plasticity i.e., enough capacity to learn future tasks, especially in the case of a fixed-capacity network, which is used in this paper.

To solve the stability-plasticity dilemma in fixed-capacity networks, typical approaches penalize the deviation of important weights for previous tasks during learning on new tasks. These approaches tend to achieve a high stability but lack plasticity, when the number of tasks becomes large. In this paper, we propose a dual strategy to train a model on a sequence of tasks. More precisely, during training on task t, in addition to constraining the weights of the network that are vital for previous tasks, we simultaneously sparsify the network using a model compression technique similar to [3], which leverage the over-parameterization of CNNs in terms of number of filters and feature maps. Therefore, after training on task t, our method identifies two sets of filters: (i) those that were crucial for learning tasks up to t, which we penalize to not deviate too much via a stability regularizer thus ensuring the stability of the network and (ii) filters that are unimportant in learning task t, which are reinitialized and used to learn future tasks, hence ensuring the plasticity of the network.

Specifically, to sparsify the CNN, we constrain convolutional kernels parameters with a sparsity regularizer, which we denote as plasticity regularizer, consisting of two parts: (i) an *exclusive sparsity* regularization term, which promotes feature discrimination by enforcing the features captured at each layer to be as different as possible, seconded by (ii) a *group sparisity* regularization term, which allows different layers to share important features. Moreover, the degree of both is adapted to allow more feature sharing at lower layers and feature discrimination at top layers, which makes sense for image classification which is the focus of this work.

For the remainder of the paper, we will outline three main contributions. First, we derive a sparsity regularizer combining exclusive- and group sparsity to ensure the plasticity of the CNN in regard to future tasks. Second, we combine the plasticity regularizer with a stability regularizer, which prevents performance deterioration on previous learned tasks. Finally, in the stability regularizer term we propose to adaptively update the importance of each filter based on its post-activation value, such that important filters are not all equal but differ in their degree of importance.

2. Related Work

Continual learning has gained much attention in recent years. In this section, we provide a brief survey of current state of the art approaches to address CL. These approaches are typically categorized in 3 mains groups. We will elaborate more on the first group since the work presented in this paper fall into this category. For an exhaustive survey see [4].

Regularization-based. The first group addresses continual learning by preventing significant changes to the parameters learned for previous tasks [5]. The key idea is to define a score indicating the importance of each parameter in the network for learning the previous tasks. When learning a new task, parameters with high importance are discouraged to deviate too much. As a notable work, Elastic weight consolidation (EWC) [6] uses a diagonal approximation of the Fisher Information Matrix as a proxy for parameter importance. In a similar vein, Memory Aware Synapses (MAS) [7] models the importance by the change in the function learned by the network rather than the loss. These approaches however focus on the stability of the network, therefore largely neglecting its plasticity for learning new tasks. Our work addresses this shortcoming via the introduced plasticity regularizer. Another notable difference in our approach is that prior methods typically work at a parameter level

(individual parameter of a convolution filter for example), whereas ours works on a group of parameters (the entire filter). More recent regularization-based approaches work on a group-level, similar to ours but have some limitations. For instance, HAT [8] proposes a per layer binary mask mechanism to capture important filters, but the method requires to know the number of tasks in advance. AGS [9] translates the notion of filter importance to active and inactive filters and implements a network sparsification strategy related to ours, but the method is restricted to group sparsity, which limits its plasticity capability.

Architecture-based. In this second group, the capacity of the network is dynamic and additional neurons/filters can be added when a new task arrives. The key idea is to devote different subsets of the final network to each task [10, 11]. The part of the network trained on old tasks can remain fixed therefore ensuring the stability whereas additional capacity comes up for new tasks, thus ensuring plasticity.

Memory-based. This final group stores data from past tasks or data representatives, which is then used during training on new tasks [12, 13]. It is the oldest technique to deal with continual learning and can often easily be combined with methods from other groups.

3. Method

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

3.1. Notation

As mentioned in Section 1, t indexes a task in a sequence of T tasks appearing to a CNN in an online fashion. Each task has a training and test dataset respectively denoted as D^t_{train} and D^t_{test}. Note that after training on task t, D^t_{train} is entirely discarded and becomes unavailable in the future. Let $i \in \{0, \ldots, L-1\}$ indexes a convolution layer of the CNN model. The parameters for i at task t reside in the kernel tensor $K^t_i \in \mathbb{R}^{k_i \times k_i \times c_i \times c_o}$ and the bias term $b^t_i \in \mathbb{R}^{c_o}$, where k_i is the kernel size, c_i and c_o the number of input and output channels respectively. We therefore denote by $\mathcal{F}^t_i = (K^t_i, b^t_i)$ the parameters of layer i at task t. Furthermore, we use $\mathcal{F}^t_{i,j} \in \mathbb{R}^{k_i \times k_i \times c_i}$ and $\mathcal{F}^t_{i+1,:j} \in \mathbb{R}^{k_i \times k_i \times c_i}$ to respectively denote the convolutional kernel of filter j in layer i and the channel j of tensor $\mathcal{F}^t_{i+1}$ that corresponds to filter$\mathcal{F}^t_{i,j}$. The set of all filters $\{\mathcal{F}^t_{i,j}\}$ in the network at task t is then given as Θ^t. In the rest of the paper, we omit the task index t when the context is clear.

3.2. Defining the Objective Function

Denoting by $f(\Theta^t)$ the CNN learner at learning task t, the goal of continual learning is to maximize the performance of $f(\Theta^t)$ at t while minimizing the forgetting on task $\mathcal{T}_1$ to $\mathcal{T}_{t-1}$, both evaluated on the test dataset $D^{t'}_{\text{test}}$ with $t' \in \{1, \cdots, t\}$. The objective function at learning $\mathcal{T}_t$ is then given as,

$$\mathcal{L}^t(\Theta^t, D^t_{train}) = \frac{1}{N^t} \sum_{n=1}^{N^t} l_n \left(f(x^t_n, \Theta^t), y^t_n \right), \tag{3.1}$$

where N^t is the number of samples in D^t_{train} and l_n is the loss (cross entropy loss) of training instance n. The final training objective for such a continual learning setting after learning all T tasks in sequence can then be written as,

$$\mathcal{L}(\Theta^T, D_{train}) = \sum_{t=1}^{T} \mathcal{L}^t(\Theta^t, D^t_{train}). \tag{3.2}$$

However, since we are interested in the continual learning setting where D^t_{train} becomes inaccessible after learning task t, the objective function in Eq. (3.2) can not be directly minimized. Therefore, when training on $\mathcal{T}_t$, the challenge is to stabilize $\sum_{t'=1}^{t-1} \mathcal{L}^{t'}(\Theta^{t'}, D^{t'}_{train})$ without explicitly measuring it, while at the same time estimating Θ^t by optimizing Eq. (3.1).

Stability Regularizer. To satisfy the above mentioned continual learning desiderata when learning a sequence of classification tasks, we constrain for each filter j, its convolution kernel parameters learned up to task $t-1$ (denoted $\hat{\mathcal{F}}^{t-1}_{i,j}$) such that their deviation during learning on task t is penalized, which results in stabilizing the performance of the network on previous tasks. To achieve this, we equip the training objective with a regularizer denoted *stability regularizer* $\mathcal{R}_S(\Theta)$, which forces the difference between $\hat{\mathcal{F}}^{t-1}_{i,j}$ and $\mathcal{F}^t_{i,j}$ to be minimal.

Our loss function in Eq 3.1, when training on task t can then be rewritten as,

$$\mathcal{L}^t(\Theta^t, D^t_{train}) = \frac{1}{N^t}\sum_{n=1}^{N^t} l_n(f(x^t_n, \Theta^t), y^t_n) + \underbrace{\sum_{\Theta^t} \|\mathcal{F}_{i,j} - \hat{\mathcal{F}}^{t-1}_{i,j}\|_2}_{\mathcal{R}_S(\Theta^t)}. \tag{3.3}$$

The stability regularizer $\mathcal{R}_S(\Theta^t)$ in Eq. (3.3) penalizes the deviation on all filters with the same hardness, treating them as being equally important. Such an assumption is however not true in modern CNN. To address this issue, we define the vector $\hat{\Gamma}^{t-1}$, which captures the importance of each filter in the network in learning tasks up to $t-1$. The importance of a specific filter j in layer i is denoted $\hat{\Gamma}^{t-1}_{\mathcal{F}_{i,j}}$. We will present later on how $\hat{\Gamma}^{t-1}_{\mathcal{F}_{i,j}}$ is computed, for now, let us assume such an importance measure is defined. The binarized version of $\hat{\Gamma}^{t-1}$ is a binary-valued mask vector denoted $\tilde{\Gamma}^{t-1}$. Each entry in the binary mask $\tilde{\Gamma}^{t-1} \in \{0, 1\}$ indicates whether or not the corresponding filter is important (1) or not (0) at learning tasks up to $\mathcal{T}_{t-1}$. Based on the latter and denoting multiplication with broadcasting by $\odot$, we can perform the operation $\tilde{\Gamma}^{t-1} \odot \Theta^{t-1}$, which divides Θ^{t-1} into 2 sets Θ^{t-1}_+ and Θ^{t-1}_- respectively containing important and unimportant filters for learning tasks up to $t-1$.

Incorporating $\hat{\Gamma}^{t-1}_{\mathcal{F}_{i,j}} \in \hat{\Gamma}^{t-1}$ into the stability constraint and constraining only filters, which are important for previous tasks, the stability regularizer in Eq. (3.3) becomes:

$$\mathcal{R}_S(\Theta^t) = \sum_{\Theta^{t-1}_+} \hat{\Gamma}^{t-1}_{\mathcal{F}_{i,j}} \|\mathcal{F}_{i,j} - \hat{\mathcal{F}}^{t-1}_{i,j}\|_2. \tag{3.4}$$

Plasticity Regularizer. Our loss function as defined in Eq. (3.3) (with the stability regularizer in Eq. (3.4)) encourages the network to be stable on previously learned tasks. However, its capacity is not yet optimally used for capturing upcoming knowledge, especially in cases where the number of tasks is large. In experiments, we observed that this loss function provides satisfactory results on small continual learning benchmarks like SVHN [14], which consists only of 5 tasks. However for a benchmark like ImageNet-50, which has 25 tasks, the CNN rapidly becomes inefficient at learning future tasks. Consequently, when training on task $\mathcal{T}_t$, the challenge is to make the network plastic enough for learning task $\mathcal{T}_{t+1}$ to $\mathcal{T}_T$ while maintaining its capacity fixed. Our idea to tackle this issue is inspired by model compression of convolutional neural networks [15]. This attractive area of research aims at generating compact CNNs by identifying and removing redundant filters from an over-parameterized network. The methods proposed to handle this typically start from a CNN that has been trained in a traditional manner, without taking into account such a future compression. In this work, we aim at considering compression during training on each task. Furthermore, instead of discarding unimportant or redundant filters as model

compression techniques do, we reinitialize them such that they can be used for learning future tasks.

Our main idea to account for compression during training is to empower the loss function with a regularizer, denoted *plasticity regularizer*, that will sparsify the CNN such that kernel of redundant filters are zeroed-out. The most common regularizer to boost sparsity is the l_1-norm $\sum_{\Theta^t}|\mathcal{F}_{i,j}|$. However, sparsification using l_1-norm usually results in accuracy drop in the context of deep CNN, since it tends to cancel out individual kernel parameters but not the complete filter. The most effective regularizer to sparsify CNNs is the group sparsity regularizer [16], which deactivates filters entirely at once, thus achieving structured sparsity. Using group sparsity, our plasticity regularizer can take the form:

$$\mathcal{R}_P(\Theta^t) = \sum_{\Theta^t_-} \|\mathcal{F}_{i,j}\|_2. \tag{3.5}$$

Group sparsity regularizer sparsify a CNN by highly promoting feature sharing among the layers. This is strongly desired in lower layers of a convolution neural network, where feature need to be shared and grouped into most representative geometry. On the other hand, in upper layers, which aim at differentiating between classes, feature discrimination will be a more appropriate perspective. A regularizer that promotes such a feature discrimination has been proposed in [17] and can be defined as $\sum_{\Theta^t}\|\mathcal{F}_{i,j}\|_1^2$. In a CNN this results in constraining the convolution filters to be as different as possible from each other. The filters therefore learn disjoint sets of feature, which removes redundancies among them. To allow both feature-sharing (at lower layers) and discriminance (at higher layers) we combine the group sparsity regularizer in Eq. (3.5) with an exclusive sparsity regularizer.

As a key ingredient, we apply both sparsity regularizers to the set of filters which has been identified as unimportant for previous tasks, i.e., $\mathcal{F}_{i,j} \in \Theta^{t-1}_-$, since filters, which are important in learning those tasks are constrained by the stability regularizer to not change much. The resulting plasticity regularizer can then be formulated as:

$$\mathcal{R}_P(\Theta^t) = \underbrace{\sum_{\Theta^{t-1}_-} \gamma\|\mathcal{F}_{i,j}\|_2}_{\text{Group sparsity}} + \underbrace{\sum_{\Theta^{t-1}_-} \zeta\|\mathcal{F}_{i,j}\|_1^2}_{\text{Exclusive sparsity}}, \tag{3.6}$$

with $\gamma = \psi_i$ and $\zeta = \frac{(1-\psi_i)}{2}$, where $\psi_i = 1 - \frac{i}{L-1}$ adjusts the degree of sharing and discriminating features at each layer, giving more weight to group sparsity in lower layers, whereas exclusive sparsity is dominating in higher layers.

Final loss term. During training on task $\mathcal{T}_t$ our network ensures both the stability on previous tasks $\mathcal{T}_{t'}, t' \in \{1, \cdots, t-1\}$ and the plasticity for future task $\mathcal{T}_{t''}, t'' \in \{t+1, \cdots, T\}$ by combining Eq. (3.3), (3.4) and (3.6) to form the final training objective as:

$$\begin{aligned}\mathcal{L}^t(\Theta^t, D^t_{train}) = \frac{1}{N^t}\sum_{n=1}^{N^t} l_n(f(x^t_n, \Theta^t), y^t_n) + \ \mu_S \sum_{\Theta^{t-1}_+} \hat{\Gamma}^{t-1}_{\mathcal{F}_{i,j}} \|\mathcal{F}_{i,j} - \hat{\mathcal{F}}^{t-1}_{i,j}\|_2 \\ + \ \mu_P \sum_{\Theta^{t-1}_-} \psi_i \|\mathcal{F}_{i,j}\|_2 + \ \mu_P \sum_{\Theta^{t-1}_-} \frac{(1-\psi_i)}{2} \ \|\mathcal{F}_{i,j}\|_1^2 \ ,\end{aligned} \tag{3.7}$$

where μ_S and μ_P are dimensionless strength of stability and plasticity regularizer.

3.3. **Solving the Optimization Problem**

To minimize our regularized learning objective defined in Eq. (3.7), we use a proximal gradient descent (PGD) approach, which is a broad class of optimization techniques for

separable objectives with both smooth and non-smooth terms, $\min_{\Theta} g(\Theta) + h(\Theta)$, where $g(\Theta)$ is convex differentiable and $h(\Theta)$ is potentially non-smooth [18]. The idea is to first take for each epoch $k \in \{0, \cdots, K-1\}$ a gradient step on $g(\Theta)$ followed by a "corrective" proximal gradient step to satisfy $h(\Theta)$:

$$\Theta^{k+1} := \text{prox}_{\alpha\ h}\ \left(\Theta^k - \alpha \nabla g(\Theta^k)\right), \tag{3.8}$$

where Θ^k is the k^{th} proximal update step and $\text{prox}_{\alpha\ g}$ is the proximal operator defined for a function g scaled by a scalar $\alpha > 0$ as:

$$\text{prox}_{\alpha\ h}(v) = \arg\min_{\Theta} \left(h(\Theta) + \frac{1}{2\alpha} \|\Theta - v\|_2^2 \right). \tag{3.9}$$

Let rewrite the training objective in Eq. (3.7) as:

$$\mathcal{L}^t(\Theta^t, D^t_{train}) = \mathcal{L}^t_{\text{CE}}(\Theta) + \mathcal{L}^t_{\text{Reg}}(\Theta), \tag{3.10}$$

where $\mathcal{L}^t_{\text{CE}}(\Theta)$ is the ordinary task-specific cross entropy loss on D^t_{train} (first term in Eq. (3.7)) and $\mathcal{L}^t_{Reg}(\Theta)$ is the convex regularization loss, which combines the stability and plasticity regularizer. The proximal gradient iteration as defined in Eq. (3.8) therefore results in minimizing the task specific cross-entropy loss $\mathcal{L}^t_{CE}(\theta)$ only for one epoch, with learning rate α, 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. (3.8) can then be rephrased for our training objective as:

$$\breve{\Theta}^{k+1} := \Theta^k - \alpha \nabla \mathcal{L}_{\text{CE}}(\Theta^k) \tag{3.11}$$

$$\Theta^{k+1} := \text{prox}_{\alpha\ \mathcal{L}_{\text{Reg}}}\ (\breve{\Theta}^{k+1}). \tag{3.12}$$

Since the convolution filters are non overlapping, the proximal gradient update in Eq. (3.12) can be applied independently for each filter in each layer. Let us now derive this update rule for each filter.

PGD iteration for a single filter. Consider the training of a single convolution filter $\mathcal{F}_{i,j}$ at task index t, which is omitted in the following for sake of readability. The regularization loss term $\mathcal{L}_{\text{Reg}}(\Theta)$ in Eq. (3.10), which consists of the 3 last terms in Eq. (3.7) can be written as:

$$\mathcal{L}_{\text{Reg}}(\mathcal{F}_{i,j}) = \begin{cases} \mu_S\ \hat{\Gamma}^{t-1}_{\mathcal{F}_{i,j}}\ \|\mathcal{F}_{i,j} - \hat{\mathcal{F}}^{t-1}_{i,j}\|_2 & \text{if } \mathcal{F}_{i,j} \in \Theta^{t-1}_{+} \\ \mu_P \left(\gamma \|\mathcal{F}_{i,j}\|_2 + \zeta \|\mathcal{F}_{i,j}\|_1^2\right) & \text{if } \mathcal{F}_{i,j} \in \Theta^{t-1}_{-}. \end{cases} \tag{3.13}$$

For small learning rates α, the proximal operator as defined in Eq. (3.9) converges to [1]:

$$\text{prox}_{\alpha\ h}(v) = v - \alpha \nabla h(v), \tag{3.14}$$

which let us rephrase Eq. (3.12) for a single filter as:

$$\mathcal{F}^{k+1}_{i,j} := \breve{\mathcal{F}}^{k+1}_{i,j} - \alpha \nabla \mathcal{L}_{\text{Reg}} \left(\breve{\mathcal{F}}^{k+1}_{i,j}\right), \tag{3.15}$$

where $\breve{\mathcal{F}}^{k+1}_{i,j}$ is the result of applying a gradient step via standard SGD optimizers (e.g., Adam).

The second term on the right hand side of Eq. (3.15) is the first order derivative of Eq. (3.13) with respect to $\breve{\mathcal{F}}^{k+1}_{i,j}$, which when derived translates Eq. (3.15) as follows:

[1] see Section 1.2 of [18] for more details.

- If $\mathcal{F}_{i,j} \in \Theta_{+}^{t-1}$

$$\begin{aligned} \mathcal{F}_{i,j}^{k+1} :&= \breve{\mathcal{F}}_{i,j}^{k+1} - \alpha \left(\mu_S \, \hat{\Gamma}_{\mathcal{F}_{i,j}}^{t-1} \, \frac{\breve{\mathcal{F}}_{i,j}^{k+1} - \hat{\mathcal{F}}_{i,j}^{t-1}}{\|\breve{\mathcal{F}}_{i,j}^{k+1} - \hat{\mathcal{F}}^{t-1}\|_2} \right) \\ &= (1-\beta) \; \breve{\mathcal{F}}_{i,j}^{k+1} + \beta \; \hat{\mathcal{F}}_{i,j}^{t-1} \\ &\text{with } \beta = \frac{\alpha \; \mu_S \; \hat{\Gamma}_{\mathcal{F}_{i,j}}^{t-1}}{\|\breve{\mathcal{F}}_{i,j}^{k+1} - \hat{\mathcal{F}}_{i,j}^{t-1}\|_2} \end{aligned} \tag{3.16}$$

- If $\mathcal{F}_{i,j} \in \Theta_{-}^{t-1}$

$$\begin{aligned} \mathcal{F}_{i,j}^{k+1} :&= \breve{\mathcal{F}}_{i,j}^{k+1} - \alpha \; \mu_P \Big(\psi_i \; \frac{\breve{\mathcal{F}}_{i,j}^{k+1}}{\|\breve{\mathcal{F}}_{i,j}^{k+1}\|_2} + (1-\psi_i) \; \|\breve{\mathcal{F}}_{i,j}^{k+1}\| \; sign\left(\breve{\mathcal{F}}_{i,j}^{k+1}\right) \Big) \\ &= (1-\xi) \; \breve{\mathcal{F}}_{i,j}^{k+1} - \eta \; sign\Big(\breve{\mathcal{F}}_{i,j}^{k+1}\Big) \\ &\text{with } \xi = \frac{\alpha \; \mu_P \; \psi_i}{\|\breve{\mathcal{F}}_{i,j}^{k+1}\|_2} \text{ and } \eta = \alpha \; \mu_P \; (1-\psi_i) \; \|\breve{\mathcal{F}}_{i,j}^{k+1}\|_1 \end{aligned} \tag{3.17}$$

From Eq. (3.16) we can observe how full stability on the filter (i.e. $\mathcal{F}_{i,j}^{k+1} = \hat{\mathcal{F}}_{i,j}^{t-1}$) is guaranteed if $\beta = 1$, whereas full plasticity (i.e. $\mathcal{F}_{i,j}^{k+1} = 0$) is achieved in Eq. (3.17) if $\xi = 1$ and $\eta = 0$.

Algorithm 1 Numerical Optimization Algorithm for task t

Input: learned parameter $\hat{\theta}^{t-1}$ up to $t-1$, learning rate α, regularizer strengh μ_s, μ_p
for each epoch k **do**
  $\breve{\theta}^{k+1} = \theta^k - \alpha \nabla \mathcal{L}_{CE}^t(\theta^k)$ ▷ Update parameters using SGD based on cross entropy loss
  for each filter j in layer i **do**
    compute ψ_i ▷ Adjust degree of sharing and discriminating features.
    Update $\theta_{\mathcal{F}_{i,j}}^{k+1}$ by computing (3.16) or (3.17)
  end for
end for

3.4. **Filters Pruning and Reinitialization**

We now present the computation of filters importance, which plays a crucial role in our approach.

Filter importance $\Gamma_{\mathcal{F}_{i,j}}^t$ at learning task t. Once the CNN is trained on task t and the plasticity regularizer has ensured network sparsification, we measure the importance of each filter in the CNN at learning task t. The importance $\Gamma_{\mathcal{F}_{i,j}}^t$ of a filter j in layer i at learning t is quantified as the average standard deviation of its post-activation value across all training samples of task t. More specifically, for each filter $\mathcal{F}_{i,j}^t$, we compute the standard deviation $\sigma_{\mathcal{F}_{i,j}}^t \in \mathbb{R}^{H_o \times W_o}$ of its output activation $a_{\mathcal{F}_{i,j}} \in \mathbb{R}^{H_o \times W_o}$ across all N^t training samples:

$$\sigma_{\mathcal{F}_{i,j}}^t = \sqrt{\frac{1}{N^t} \sum_{n=1}^{N^t} \big(a_{\mathcal{F}_{i,j}}(x_n^t) - \overline{a_{\mathcal{F}_{i,j}}}\big)^2}, \tag{3.18}$$

where $a_{\mathcal{F}_{i,j}}(x_n^t)$ is the Rectified Linear Unit (ReLU) activation value of filter $\mathcal{F}_{i,j}$ for the training instance $x_n^t \in D_{\text{train}}^t$ and $\overline{a_{\mathcal{F}_{i,j}}}$ is the average activation. The importance $\Gamma_{\mathcal{F}_{i,j}}^t \in \mathbb{R}$

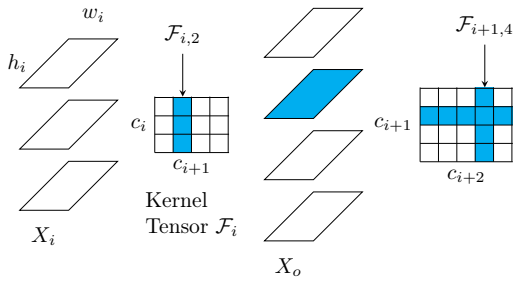


Figure 1. Processing unimportant filters. Filter 2 in layer i ($\mathcal{F}_{i,2}$) is unimportant. Consequently, its kernel weights get reinitialized and channel 2 of all filters in layer $i+1$ ($\mathcal{F}_{i+1,:2}$) is zeroed. The same applies to $\mathcal{F}_{i,4}$, which is also unimportant.

of $\mathcal{F}^t_{i,j}$ is then computed by averaging the element of the 2D tensor $\sigma^t_{\mathcal{F}_{i,j}}$ as:

$$\Gamma^t_{\mathcal{F}_{i,j}} = \frac{1}{H_o \times W_o} \sum_{q \in \sigma^t_{\mathcal{F}_{i,j}}} q \tag{3.19}$$

Intuition behind $\Gamma^t_{\mathcal{F}_{i,j}}$. The intuition behind the filter importance measure as presented in Eq. (3.19) with Eq. (3.18) is based on the observation that one of the reasons for the popularity of the ReLU activation is that it induces a sparsity in activation, which allows convolutional layers to act as feature detectors [19]. Hence, one can reasonably assume that if the output activation value of a filter is small, then the feature detected by this filter (and consequently the filter itself) is not important for learning the current task.

Filter importance $\hat{\Gamma}^t_{\mathcal{F}_{i,j}}$ at learning task $\mathcal{T}_1$ up to $\mathcal{T}_t$. Once the importance $\Gamma^t_{\mathcal{F}_{i,j}}$ of a filter $\mathcal{F}_{i,j}$ in learning task t only, is computed, we need to account for the importance of that filter in learning task $\mathcal{T}_1$ to $\mathcal{T}_{t-1}$ such that it remains stable at performing those tasks. For this purpose, we compute the importance $\hat{\Gamma}^t_{\mathcal{F}_{i,j}}$ of filter $\mathcal{F}_{i,j}$ for learning task $\mathcal{T}_1$ to $\mathcal{T}_t$ as:

$$\hat{\Gamma}^t_{\mathcal{F}_{i,j}} := \nu\, \hat{\Gamma}^{t-1}_{\mathcal{F}_{i,j}} + \Gamma^t_{\mathcal{F}_{i,j}}, \tag{3.20}$$

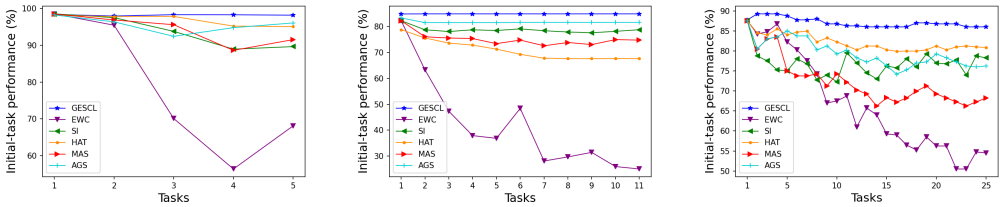
where $\hat{\Gamma}^{t-1}_{\mathcal{F}_{i,j}}$ is the importance of $\mathcal{F}_{i,j}$ at learning tasks $\mathcal{T}_1$ to $\mathcal{T}_{t-1}$ and ν is a hyperparameter balancing the filter importance before and after training on task t.

Processing unimportant filters. For filters with a score $\hat{\Gamma}^t_{\mathcal{F}_{i,j}} = 0$, i.e. filters $\mathcal{F}^t_{i,j} \in \Theta^t_-$, which have been identified as unimportant for learning task up to t, we perform the following two actions, which are illustrated in Figure 1:

- Kernel of $\mathcal{F}^t_{i,j} \in \Theta^t_-$ are randomly re-initialized. This allows them to be active and ready to be used by the training procedure for learning a new task.
- Kernel of channel j of tensor $\mathcal{F}_{i+1}$ that corresponds to filter $\mathcal{F}^t_{i,j}$ are set to zero, this prevents them from negatively affecting inference on task t.

4. Experiments

Continual Learning Scenario. In the experiments, we follow the typical continual learning (CL) scenario as presented in Section 1. To recall, in a CL scenario, a model is required to learn a sequence of tasks with unknown data distribution. We make the assumption that after training on task t, D^t_{train} becomes inaccessible whereas D^t_{test} is used to evaluate the



(a) Split SVHN *(b)* Split CIFAR-10/100 *(c)* ImageNet-50

Figure 2. Evaluation of catastrophic forgetting by measuring performance retention on the Initial task. Results show for each dataset how the classification accuracy of the first task evolves as further tasks are being learnt. Overall, GESCL show the strongest resilience against catastrophic forgetting. When facing new tasks, the performance it achieves on the initial task does not degrade as in concurrent approaches.

model performance on $\mathcal{T}_t$ after training on $\mathcal{T}'_t$, with $t' > t$. Furthermore, we assume that the model capacity is fixed.

Datasets. We conduct experiments on three different datasets, which are popular continual learning benchmarks: SVHN [14], CIFAR-10/100 [20] and ImageNet-50[2], which we generated as subset of the ImageNet dataset. It contains 50 classes grouped into 2 consecutive classes to form 25 tasks. Each class contains 1600 training images and 200 validation images. For SVHN, which contain 10 classes, we group 2 consecutive classes to get 5 tasks. In the case of CIFAR-10/100, the first task consists of all 10 classes of CIFAR-10, wheras CIFAR-100 is split into 10 tasks, which serve as the remaining CL tasks, resulting in 11 tasks in this experiment.

Baselines. We compare our work to prior state-of-the-art approaches including two reference methods EWC [6] and SI [21] as well as three recent, competitive ones, namely, MAS [7], HAT [8] and AGS [9]. We perform grid search to fairly select the best hyper-parameters for each approach.

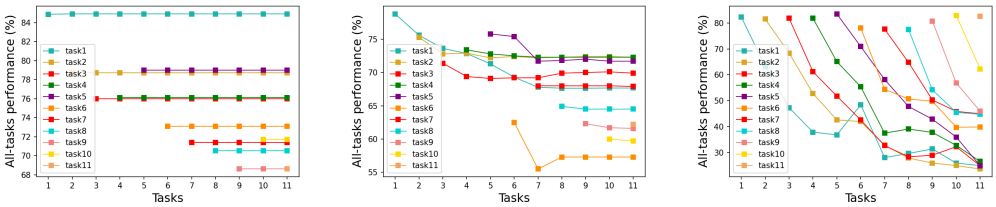
Network. For SVHN and CIFAR we use a CNN consisting of 3 blocks of 3×3 convolutions with $32, 64$ and 128 filters respectively, followed by ReLU and a 2×2 max-pooling. In ImageNet-50 experiments, we use a CNN similar to [22], which consists of two blocks of 2×2 convolution with 64 filters, followed by ReLU and a 2×2 max-pooling. For all experiments, we used a multi-headed network.

4.1. Is GESCL Able to not Catastrophically Forget?

As an initial experiment, we examine the ability of our method at addressing catastrophic forgetting. To assess catastrophic forgetting, a common metric is to evaluate how the accuracy of each task varies when learning the remaining tasks [23]. This is shown in Fig. 2, which illustrates for each dataset how the accuracy on the initial task evolves during the continual learning experiment. As can be seen, after sequentially training on all tasks, our method is the most stable and less forgetful, showing little to no forgetting on the ability to perform the first task.

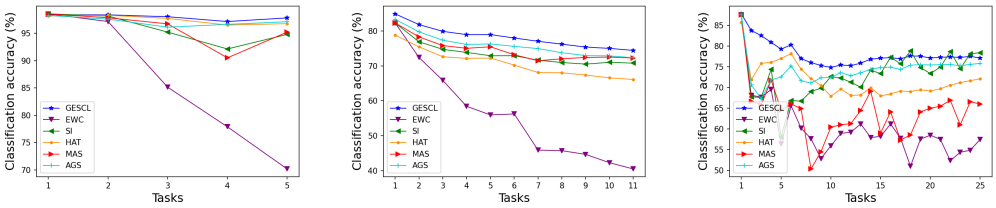
To further examine the catastrophic forgetting evaluation, Fig. 3 outlines for the CIFAR-10/100 dataset, how the test accuracy of each task changes after new tasks have been learnt. Here again, GESCL shows strong resilience against catastrophic forgetting.

[2]Our ImageNet-50 dataset can be downloaded here.



(a) GESCL *(b)* HAT *(c)* EWC

Figure 3. Evaluation of catastrophic forgetting by measuring performance retention on all task for the CIFAR-10/100 dataset. Results show how the test accuracy of each task evolves as further tasks are being learnt. Overall, GESCL suffers less from catastrophic forgetting.



(a) Split SVHN *(b)* Split CIFAR-10/100 *(c)* ImageNet-50

Figure 4. Evaluation of average test accuracy during the continual learning experiment of three datasets.

Overall, thanks to the filter importance adaptive stability regularizer (introduced in Section 3), which shrinks the change of vital parameters from task to task, GESCL outperforms state-of-the-art accuracy retention degrees. HAT [8] tends to be a strong competitor especially on small dataset like SVHN, also achieving high performance retention degree, wheras EWC in contrast, suffer from severe degradation of performance especially in ImageNet-50 and CIFAR-10/100. This points out the limitation of EWC when the number of tasks becomes large.

4.2. **Is GESCL a Competitive Continual Learning Model in Terms of Accuracy?**

In this experiment, we aim at validating the effectiveness of our method in terms of classification accuracy during the continual learning experiments. To evaluate the classification accuracy, we use the all-important average accuracy metric from [13], which is standard practice in the literature. More specifically, the classification accuracy at task t, is the average accuracy obtained from testing on task $1, \cdots, t$. For instance, denoting by $a_t^{t'}$ the accuracy (fraction of correctly classified images) evaluated on the test set of the t'-th task after the model is sequentially trained on $\mathcal{T}_1$ to $\mathcal{T}_t$, the average accuracy $A_t \in [0, 1]$ at task t is:

$$A_t = \frac{1}{t} \sum_{t'=1}^{t} a_t^{t'}. \tag{4.1}$$

Fig. 4 provides for different datasets, a view of how the average accuracy evolves from task to task during the continual learning experiments. The results indicate that GESCL outperforms baseline algorithms. Interestingly, SI [21] reaches the best performance on the last task on ImageNet-50, after being clearly outperformed by our method on earlier tasks. Another interesting observation is that AGS [9] is the strongest competitor to our method

regarding average accuracy while it is surpassed by HAT [8] on SVHN and ImageNet in terms of avoiding forgetting as illustrated in Fig. 2. For both average accuracy and forgetting, our method shows superior performance to the aforementioned CL reference baselines.

An empirical conclusion that can be made out of Fig. 2, 3 and 4 is that GESCL achieves strong overall continual learning results, thanks to the way it addresses catastrophic forgetting and learning of several new tasks via the stability- and plasticity regularizers pair embedded in the networks learning procedure.

4.3. **Ablation Study**

We performed an ablation study to examine the contribution of each component of GESCL in its overall performance on CIFAR-10/100 and ImageNet-50 datasets. Particularly, we are interested in how (1) the stability regularizer μ_s, (2) the plasticity regularizer μ_p, (3) the filter importance balance ν, contribute to the base model. We implement variants of GESCL with different combinations of those components, each component being either activated ($\surd$) or removed ($\times$). We report the results in Table 1, where A denotes the average accuracy as defined in Eq. (4.1), and F indicates the average forgetting defined as the decrease in performance at each task between the peak accuracy and the accuracy after all tasks have been learnt [23]. Results for those two metrics as shown in Table 1 demonstrate the effectiveness of each component at improving the classification accuracy and overall forgetting of the method.

Table 1. Average accuracy (A) and average forgetting (F) of GESCL variants on CIFAR-10/100 and ImageNet-50.

μ_s	μ_p	ν	CIFAR-10/100		ImageNet-50	
			A(%)	F	A(%)	F
$\surd$	$\surd$	$\surd$	74.5	0.0008	77.1	0.00007
$\surd$	$\surd$	$\times$	71.6	0.013	75.10	0.068
$\times$	$\surd$	$\surd$	41.9	0.44	57.38	0.349
$\surd$	$\times$	$\surd$	69.12	0.013	73.59	0.018

5. **Conclusion**

In this work, we presented a simple yet effective continual learning method for convolutional neural networks. The proposed algorithm derived a novel regularization term to deal with the stability-plasticity dilemma. Through experiments, the algorithm showed consistent performance across standard continual learning benchmarks and performed competitive or superior to existing state-of-the-art methods in both forgetting prevention and adaptability to new tasks.

Acknowledgements

This work was funded by the federal state of North Rhine-Westphalia and the European Regional Development Fund FKZ: ERFE-040021.

References

[1] M. McCloskey and N. J. Cohen. "Catastrophic interference in connectionist networks: The sequential learning problem". In: *Psychology of learning and motivation*. Vol. 24. Elsevier, 1989, pp. 109–165.

[2] W. C. Abraham and A. Robins. "Memory retention–the synaptic stability versus plasticity dilemma". In: *Trends in neurosciences* 28.2 (2005), pp. 73–78.

[3] J. Yoon and S. J. Hwang. "Combined group and exclusive sparsity for deep neural networks". In: *International Conference on Machine Learning*. 2017, pp. 3958–3966.

[4] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. "Continual lifelong learning with neural networks: A review". In: *Neural Networks* 113 (2019), pp. 54–71.

[5] J. Von Oswald, D. Zhao, S. Kobayashi, S. Schug, M. Caccia, N. Zucchet, and J. Sacramento. "Learning where to learn: Gradient sparsity in meta and continual learning". In: *Advances in Neural Information Processing Systems* 34 (2021).

[6] 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". In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

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

[8] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. "Overcoming catastrophic forgetting with hard attention to the task". In: *arXiv preprint arXiv:1801.01423* (2018).

[9] S. Jung, H. Ahn, S. Cha, and T. Moon. "Continual Learning with Node-Importance based Adaptive Group Sparse Regularization". In: *Advances in Neural Information Processing Systems* 33 (2020).

[10] O. Ostapenko, P. Rodriguez, M. Caccia, and L. Charlin. "Continual Learning via Local Module Composition". In: *Advances in Neural Information Processing Systems* 34 (2021).

[11] J. Yoon, S. Kim, E. Yang, and S. J. Hwang. "Scalable and order-robust continual learning with additive parameter decomposition". In: *arXiv preprint arXiv:1902.09432* (2019).

[12] C. Henning, M. Cervera, F. D'Angelo, J. Von Oswald, R. Traber, B. Ehret, S. Kobayashi, B. F. Grewe, and J. Sacramento. "Posterior meta-replay for continual learning". In: *Advances in Neural Information Processing Systems* 34 (2021).

[13] D. Lopez-Paz and M. Ranzato. "Gradient episodic memory for continual learning". In: *Advances in neural information processing systems* 30 (2017).

[14] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. "Reading digits in natural images with unsupervised feature learning". In: (2011).

[15] L. Liebenwein, C. Baykal, H. Lang, D. Feldman, and D. Rus. "Provable filter pruning for efficient neural networks". In: *arXiv preprint arXiv:1911.07412* (2019).

[16] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. "Learning structured sparsity in deep neural networks". In: *arXiv preprint arXiv:1608.03665* (2016).

[17] Y. Zhou, R. Jin, and S. C.-H. Hoi. "Exclusive lasso for multi-task feature selection". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 988–995.

[18] N. Parikh and S. Boyd. "Proximal algorithms". In: *Foundations and Trends in optimization* 1.3 (2014), pp. 127–239.

[19] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. "Pruning convolutional neural networks for resource efficient inference". In: *arXiv preprint arXiv:1611.06440* (2016).

[20] A. Krizhevsky, G. Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[21] F. Zenke, B. Poole, and S. Ganguli. "Continual learning through synaptic intelligence". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3987–3995.

[22] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. "Matching networks for one shot learning". In: *Advances in neural information processing systems* 29 (2016), pp. 3630–3638.

[23] S. I. Mirzadeh, M. Farajtabar, R. Pascanu, and H. Ghasemzadeh. "Understanding the role of training regimes in continual learning". In: *arXiv preprint arXiv:2006.06958* (2020).