SELFLESS SEQUENTIAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequential learning studies the problem of learning tasks in a sequence with access restricted to only the data of the current task. In this paper we look at a scenario with a fixed model capacity, and postulate that the learning process should not be selfish, i.e. it should account for future tasks to be added and thus leave enough capacity for them. To achieve Selfless Sequential Learning we study different regularization strategies and activation functions that could lead to less interference between the different tasks. We find that learning a sparse representation is more beneficial for sequential learning than encouraging parameter sparsity. In particular, we propose a novel regularizer, that encourages representation sparsity by means of neural inhibition. It results in few active neurons which in turn leaves more free neurons to be utilized by upcoming tasks. As suppressing all other neurons in a layer can be too drastic, especially for complex tasks requiring strong representations, our regularizer only inhibits other neurons in a local neighbourhood, inspired by inhibition processes in the brain. We combine our novel regularizer, Sparse coding through Local Neural Inhibition (SLNI), with state-of-the-art sequential learning methods that penalize changes to important previously learned parts of the network. We show that our new regularizer leads to increased sparsity which translates in consistent performance improvement on diverse datasets.

1 INTRODUCTION

Sequential learning, also referred to as continual, incremental, or lifelong learning, studies the problem of learning a sequence of tasks, one at a time, without access to the training data of previous tasks, yet avoiding catastrophic interference with the previously learned tasks (French [1999] [Li & Hoiem [2016]]. Some methods exploit an additional episodic memory to store a small amount of previous tasks data to regularize future task learning (e.g. Lopez-Paz et al. [2017]). Others store previous tasks models and at test time, select one model or merge the models (Rusu et al. [2016]; Aljundi et al. [2016]; Lee et al. [2017]). In contrast, in this work we are interested in the challenging situation of learning a sequence of tasks without access to any previous task data and restricted to a fixed model capacity, as also studied e.g. in Kirkpatrick et al. [2016]; Aljundi et al. [2017]. This scenario not only has many practical benefits, including privacy and scalability, but also resembles more closely how the mammalian brain learns tasks over time.

However, while the mammalian brain is composed of billions of neurons, at any given time, information is represented by only a few active neurons resulting in a sparsity of 90-95% (Lennie [2003]). In neural biology, lateral inhibition describes the ability of an activated neuron to reduce the activity of its weaker neighbors. This creates a powerful decorrelated and compact representation with minimum interference between different input patterns in the brain (Yu et al. [2014]). On the contrary, artificial neural networks typically learn dense representations that are highly entangled and sensitive to changes in the input (Bengio et al. [2009]).

Based on these observations, we advocate for the use of sparse models, leading to Selfless Sequential Learning, i.e. leaving capacity for future tasks. We study several regularization approaches and activation functions proposed in the literature and evaluate their effect on sparsity and performance in a sequential learning context. Sparsity in neural networks can be thought of either in terms of the network parameters or in terms of the representation (i.e., the activations). In this paper we postulate, and confirm experimentally, that a sparse and decorrelated representation is preferable over parameter sparsity in a sequential learning scenario. There are two arguments for this: first, a sparse representation is less sensitive to new and different patterns (such as data from new tasks) and
Figure 1: The difference between parameter sparsity (a) and representation sparsity (b) in a simple two tasks case. Learning the first task utilizes parts indicated in red. Task 2 has different input patterns and uses parts shown in green. Orange indicates changed neurons. In (a), when an example from the first task is encountered again, the activations of the first layer will not be affected by the changes, however, the second and later layer activations are changed. Such interference is avoided when imposing sparsity on the representation (b).

second, the training procedure of the new tasks can use the free neurons leading to less interference with the previous tasks, hence reducing forgetting. In contrast, when the effective parameters are spread among different neurons, changing the ineffective ones would change the function of their corresponding neurons and hence interfere with previous tasks (see also Figure 1).

Based on the finding that we need sparse activations, we propose a new regularizer that exhibits a behavior similar to the lateral inhibition in biological neurons. The main idea of our regularizer is to penalize neurons which are active at the same time, leading to more sparsity. However, complex tasks may actually require multiple active neurons in a layer at the same time to learn a strong representation. Therefore, our regularizer, Sparse coding through Local Neural Inhibition (SLNI), only penalizes neurons locally. Furthermore, we don’t want inhibition to affect previously learned tasks, even if later tasks use neurons from earlier tasks. An important component of SLNI is thus to discount inhibition from neurons which have high neuron importance – a new concept that we introduce in analogy to parameter importance. When combined with a state-of-the-art important parameters preservation method (Aljundi et al., 2017; Kirkpatrick et al., 2016), our proposed regularizer leads to sparse and decorrelated representations of different input patterns which improves the lifelong learning performance.

Our contribution is threefold. First, we advocate to focus on Selfless Sequential Learning and study a diverse set of representation based regularizers, parameter based regularizers, as well as sparsity inducing activation functions to this end. These have not been studied extensively in the lifelong learning (LLL) literature before. Second, we propose a novel regularizer, SLNI, which is inspired by inhibition in the brain. Third, we show that our proposed regularizer consistently outperforms alternatives on three diverse datasets (Permuted MNIST, CIFAR, Tiny Imagenet) and we compare to and outperform state-of-the-art LLL approaches on an 8-task object classification challenge. SLNI can be applied to different regularization based LLL approaches, and we show experiments with MAS (Aljundi et al., 2017) and EWC (Kirkpatrick et al., 2016).

In the following, we first discuss related approaches to LLL and different regularization criteria from a LLL perspective (Section 2). We proceed by introducing Selfless Sequential Learning and detailing our novel regularizer (Section 3). Section 4 describes our experimental evaluation, while Section 5 concludes the paper.

2 RELATED WORK

Our work lies at the heart of lifelong learning where the goal is to learn a sequence of tasks without catastrophic forgetting of previously learned ones (Thrun & Mitchell, 1995). This paradigm has been revisited recently after the renaissance of neural networks. We can identify different approaches to introduce lifelong learning in neural networks, however, our work focuses on learning a sequence of tasks using a fixed model capacity. Under this setting, methods either follow a pseudo rehearsal approach, i.e. using the new task data to approximate the performance of the previous task (Li & Hoiem, 2016; Triki et al., 2017), or aim at identifying the important parameters used by the current set of tasks and penalizing changes to those parameters by new tasks. To identify the important parameters for a given task, Elastic Weight Consolidation (Kirkpatrick et al., 2016) uses an approximation of the Fisher information matrix computed after training a given task. Liu et al. (2018) suggest a network reparameterization to obtain a better diagonal approximation of the Fisher Information matrix of the network parameters. Path Integral (Zenke et al., 2017) estimates the importance of the network parameters while learning a given task by accumulating the contribution of each parameter to the change in the loss. Chaudhry et al. (2018) suggest a KL-divergence based generalization of Elastic
Weight Consolidation and Path Integral. Memory Aware Synapses (Aljundi et al., 2017) estimates the importance of the parameters in an online manner without supervision by measuring the sensitivity of the learned function to small perturbations on the parameters. This method is less sensitive to the data distribution shift, and a local version proposed by the authors resembles applying Hebb rule (Hebb, 2002) to consolidate the important parameters, making it more biologically plausible. However, for all these methods, learning a task could utilize a good portion of the network capacity leaving few "free" neurons to be adapted by the new task, which in turn leads to inferior performance on the newly learned tasks or forgetting the previously learned ones, as we will show in the experiments. Hence, we study the role of sparsity and representation decorrelation in sequential learning. This aspect has not received much attention in the literature yet. Very recently, (Serrà et al., 2018) proposed to overcome catastrophic forgetting through learned hard attention masks, and stored in an embedding, they have proposed to use $L_1$ regularization on the attention vectors.

The concept of reducing the representation overlap was suggested before in early attempts towards overcoming catastrophic forgetting in neural networks (French, 1999). This led to several methods with the goal of orthogonalizing the activations (French, 1992; 1994; Kruschke, 1992; 1993; Sloman & Rumelhart, 1992). However, these approaches were mainly designed for specific architectures and activation functions which makes it hard to integrate them in recent neural network structures.

The sparsification of neural networks was mostly studied for compression. SVD decomposition can be applied on top of neural networks to reduce the number of effective parameters (Xue et al., 2013). However, there is no guarantee that the training procedure would converge to a low rank weight matrix. Other works iterate between pruning and retraining of a neural network as a post processing step (Liu et al. [2015] Sun et al., 2016; Aghasi et al., 2017; Louizos et al., 2017)). While compressing a neural network by removing parameters leads to a sparser neural network, this does not necessarily lead to a sparser representation. In other words, a weight vector can be highly sparse but spread over the different neurons. This reduces the effective size of a neural network, from a compression point of view, but it would not be beneficial for later tasks as most of the neurons are already occupied by the current set of tasks. In our experiments, we show the difference between using a sparse penalty on the representation versus applying it to the weights.

3 Selfless Sequential Learning

One of the main challenges in single model sequential learning is to have capacity to learn new tasks at the same time avoid catastrophic forgetting of previous tasks as a result of learning new tasks. In order to prevent catastrophic forgetting, importance weight based methods such as EWC (Kirkpatrick et al., 2016) or MAS (Aljundi et al., 2017) introduce an importance weight $Ω_k$ for each parameter $θ_k$ in the network. While these methods differ in how to estimate the important parameters, all of them penalize changes to important parameters when learning a new task $T_n$ using $L_2$ penalty:

$$T_n : \min_\theta \frac{1}{M} \sum_{m=1}^M L(y_m, f(x_m, θ^n)) + \lambda Ω \sum_k Ω_k(θ^n_k - θ^{n-1}_k)^2$$  \label{eq:ewc}

where $θ^{n-1}_k$ are the optimal parameters that are learned so far, i.e. before the current task. $x_m$ is the input and $y_m$ the desired output of the network. $Ω_l$ is a trade-off parameter between the new task objective and the changes on the important parameters, i.e. the amount of forgetting.

In this work we introduce an additional regularizer $R_{SSL}$ which encourages sparsity in the activations $H_l = \{h^m_l\}$ for each layer $l$.

$$T_n : \min_\theta \frac{1}{M} \sum_{m=1}^M L(y_m, f(x_m, θ^n)) + \lambda Ω \sum_k Ω_k(θ^n_k - θ^{n-1}_k)^2 + \lambda_{SSL} \sum_l R_{SSL}(H_l)$$  \label{eq:ssl}

$λ_{SSL}$ and $λ_Ω$ are trade-off parameters that control the contribution of each term. When training the first task ($n = 1$), $Ω_k = 0$. For sparsity in parameters we instead regularize the parameters $R_{SSL}(θ_k)$.

3.1 Sparse coding through Neural Inhibition (SNI)

Our goal is to impose a regularization mechanism that encourages the learning procedure of a given task in a sequence to learn a sparse and decorrelated representation. This in turn minimizes the
interference with the previous and future tasks and utilizes fewest possible number of neurons. To promote sparsity in the representation, we can identify, in the literature, the use of $L_1$ norm on the activations (since minimizing the $L_0$ norm is an NP hard problem). This has been suggested by Glorot et al. (2011) to be combined with the rectifier activation function (ReLU) to control unbounded activations and to increase sparsity. However, $L_1$ norm imposes an equal penalty on all the active neurons leading to small activation magnitude across the network.

On the other hand, learning a decorrelated representation has been explored before with the goal of reducing overfitting. This is usually done by minimizing the Frobenius norm of the covariance matrix corrected by the diagonal as in Cogswell et al. (2015) or Xiong et al. (2016). Such a penalty results in a decorrelated representation but with activations that are mostly close to a non zero mean value. Merging the two objectives of sparse and decorrelated representation can be achieved by the following objective:

$$R_{SNI}(H_l) = \frac{1}{M} \sum_{i,j} h_i^m h_j^m, \quad i \neq j$$

(3)

where we consider a hidden layer $l$ with the activations $H_l = \{h_i^m\}$ for a set of inputs $X = \{x_m\}$ where $i \in 1, \ldots, N$ with $N$ the number of neurons in the hidden layer. This formula differs from minimizing the Frobenius norm of the covariance matrix in two simple yet important aspects:

1. In the case of a ReLU activation function, used in most modern architectures, a neuron is active if its output is larger than zero and zero otherwise. By assuming a close to zero mean of the activations, $\mu_i \approx 0 \forall \ i \in 1, \ldots, N$, it minimizes the correlation between any two active neurons.

2. By evaluating the derivative of the presented regularizer w.r.t. the activation, we get:

$$\frac{\partial R_{SNI}(H_l)}{\partial h_i^m} = \frac{1}{M} \sum_{j \neq i} h_j^m$$

(4)

i.e., each active neuron receives a penalty from every other active neuron that corresponds to that other neuron’s activation magnitude. In other words, if a neuron fires, with a high activation value, for a given example, it will suppress firing of other neurons for that same example. Hence, this results in a decorrelated sparse representation.

### 3.2 Sparse coding through Local Neural Inhibition (SNI)

The loss imposed by the SNI objective will only be zero when there is at most one active neuron per example. This seems to be too harsh for complex tasks that need a richer representation. Thus, we suggest to relax the objective by imposing a spatial weighting to the correlation penalty. In other words, an active neuron penalizes mostly its close neighbours and this effect vanishes for neurons further away. Instead of uniformly penalizing all the correlated neurons, we weight the correlation penalty between two neurons with locations $i$ and $j$ using a Gaussian weighting. This gives

$$R_{SNI}^{SLNI}(H_l) = \frac{1}{M} \sum_{i,j} e^{-\frac{(i-j)^2}{2\sigma^2}} \sum_{m} h_i^m h_j^m, \quad i \neq j$$

(5)

As such, each active neuron inhibits its neighbours, introducing a locality in the network inspired by biological neurons. While the notion of neighbouring neurons is not well established in a fully connected network, our aim is to allow few neurons to be active and not only one, thus those few activations don’t have to be small to compensate for the penalty. $\sigma^2$ is a hyper parameter representing the scale at which neurons can affect each other. Note that this is somewhat more flexible than decorrelating neurons in fixed groups as used in Xiong et al. (2016). Our regularizer inhibits locally the active neurons leading to a sparse coding through local neural inhibition, hence the name of our regularizer Sparse coding through Local Neural Inhibition (SLNI).

### 3.3 Neuron importance for discounting inhibition

Our regularizer is to be applied for each task in the learning sequence. In the case of tasks with completely different input patterns, the active neurons of the previous tasks will not be activated given the new tasks input patterns. However, when the new tasks are of similar or shared patterns, neurons used for previous tasks will be active. In that case, our penalty would discourage other neurons from being active and encourage the new task to adapt the already active neurons instead. This would
interfere with the previous tasks and could increase forgetting which is exactly what we want to overcome. To avoid such interference, we add a weight factor taking into account the importance of the neurons with respect to the previous tasks. To estimate the importance of the neurons, we use as a measure the sensitivity of the learned objective to their changes. This is approximated by the gradients of the loss w.r.t. the neurons outputs evaluated at each data point. To get an importance value, we then accumulate the magnitude of the gradients over the given data points obtaining importance weight $\alpha_i$ for neuron $n_i$:

$$\alpha_i = \frac{1}{M} \sum_{m=1}^{M} \left\| g_i(x_m) \right\|, \quad g_i(x_m) = \frac{\partial(L(y_m, f(x_m, \theta^t)))}{\partial n_i^m}$$

where $n_i^m$ is the output of neuron $n_i$ for a given input example $x_m$, and $\theta^t$ are the parameters after learning task $t$. This is in line with the estimation of the parameters importance in [Kirkpatrick et al., 2016] but considering the derivation variables to be the neurons outputs instead of the parameters. Instead of relying on the gradient of the loss, we can also use the gradient of the learned function, i.e. the output layer, as done in [Aljundi et al., 2017] for estimating the parameters importance. During the early phases of this work, we experimented with both and observed correlated estimation, hence to merge the estimation of the parameters importance with that of the neurons, in the experiments we utilize the gradient of the function when based on [Aljundi et al., 2017] as LLL method and the gradient of the loss when experimenting with EWC (Kirkpatrick et al., 2016). Then, we can weight our regularizer as follows:

$$R_{\text{SLNI}}(H_t) = \frac{1}{M} \sum_{i,j} e^{-(\alpha_i + \alpha_j)} e^{-\frac{(\alpha_i - \alpha_j)^2}{2\sigma^2}} \sum_m h_i^m h_j^m, \quad i \neq j$$

which can be read as: if an important neuron for a previous task is active given an input pattern from the current task, it will not suppress the other neurons from being active neither be affected by other active neurons. For all other active neurons, local inhibition is deployed. The final objective for training is given in Eq. 2 setting $R_{\text{SSL}} := R_{\text{SLNI}}$.

## 4 Experiments

In this section we study the role of standard regularization techniques with a focus on sparsity and decorrelation of the representation in a sequential learning scenario. We first compare different activation functions and regularization techniques, including our proposed SLNI on permuted MNIST (Sec. 4.1). Then, we compare the top competing techniques and our proposed method in the case of sequentially learning CIFAR-100 classes and Tiny Imagenet classes (Sec. 4.1). We also show how our regularizer improves the state-of-the-art performance on a sequence of object recognition tasks (Sec. 4.5). Our SLNI regularizer can be integrated in any importance weight-based lifelong learning approach such as [Kirkpatrick et al., 2016], [Zenke et al., 2017], [Aljundi et al., 2017]. Here we focus on Memory Aware Synapses (Aljundi et al., 2017) (MAS), which is easy to integrate and experiment with and has shown superior performance (Aljundi et al., 2017). However, we also show results with Elastic weight consolidation (Kirkpatrick et al., 2016) (EWC) in Sec. 4.3 and ablate the components of our regularizer in Sec. 4.4.

### 4.1 An In-depth Comparison of Regularizers and Activation Functions for Selfless Sequential Learning

We study possible regularization techniques that could lead to less interference between the different tasks in a sequential learning scenario either by enforcing sparsity or decorrelation. Additionally, we examine the use of activation functions that are inspired by lateral inhibition in biological neurons that could be advantageous in sequential learning.

**Representation Based methods:**
- **L1-Rep**: To promote representational sparsity, an $L_1$ penalty on the activations is used.
- **Decov** (Cogswell et al., 2015) aims at reducing overfitting by decorrelating neuron activations. To do so, it minimizes the Frobenius norm of the covariance matrix computed on the activations of the current batch after subtracting the diagonal to avoid penalizing independent neuron activations.
Figure 2: Comparison of different regularization techniques on 5 permuted MNIST sequence of tasks. Representation based regularizers are solid bars, bars with lines represent parameters regularizers, dotted bars represent activation function. Average test accuracy over all tasks is given in the legend.

Activation functions:
- **Maxout network** ([Goodfellow et al., 2013b]) utilizes the maxout activation function. For each group of neurons, based on a fixed window size, only the maximum activation is forwarded to the next layer. The activation function guarantees a minimum sparsity rate defined by the window size.
- **LWTA** ([Srivastava et al., 2013]): similar idea to the Maxout network with the difference that the non-maximum activations are set to zero while maintaining their connections. In contrast to Maxout, LWTA keeps the connections of the inactive neurons which can be occupied later once they are activated without changing the previously active neuron connections.
- **ReLU** ([Glorot et al., 2011]) The rectifier activation function (ReLU) used as a baseline here and indicated in later experiments as **No-Reg** as it represents the standard setting of sequential learning on networks with ReLU. All the studied regularizers use ReLU as activation function.

Parameters based regularizers:
- **OrthReg** ([Rodríguez et al., 2016]): Regularizing CNNs with locally constrained decorrelations. It aims at decorrelating the feature detectors by minimizing the cosine of the angle between the weight vectors resulting eventually in orthogonal weight vectors.
- **L2-WD**: Weight decay with **L**2 norm ([Krogh & Hertz, 1992]) controls the complexity of the learned function by minimizing the magnitude of the weights.
- **L1-Param**: **L**1 penalty on the parameters to encourage a solution with sparse parameters.

Dropout is not considered as its role contradicts our goal. While dropout can improve each task performance and reduce overfitting, it acts as a model averaging technique. By randomly masking neurons, dropout forces the different neurons to work independently but as a result encourages a redundant representation. As shown by ([Goodfellow et al., 2013a]) the best network size for classifying MNIST digits was when using dropout about 50% more than without it. Dropout steers the learning of a task towards occupying a good portion of the network capacity if not all of it which contradicts with the sequential learning needs.

**Experimental setup.** We use the MNIST dataset ([LeCun et al., 1998]) as a first task in a sequence of 5 tasks, where we randomly permute all the input pixels differently for tasks 2 to 5. The goal is to classify MNIST digits from all the different permutations. The complete random permutation of the pixels in each task requires the neural network to instantiate a new neural representation for each pattern. A similar setup has been used by ([Kirkpatrick et al., 2016], [Zenke et al., 2017], [Goodfellow et al., 2013a]) with different percentage of permutations or different number of tasks.

As a base network, we employ a multi layer perceptron with two hidden layers and a Softmax loss. We experiment with different number of neurons in the hidden layers {128, 64}. For SLNI we evaluate the effect of **λSLNI** on the performance and the obtained sparsity in Figure 4(b). In general, the best **λSLNI** is the minimum value that maintains similar or better accuracy on the first task compared to the unregularized case, and we suggest to use this as a rule-of-thumb to set **λSLNI**. For **λΩ**, we have used a high **λΩ** value that ensures the least forgetting which allows us to test the effect on the later tasks performance. Note that better average accuracies can be obtained with tuned **λΩ**. Please refer to Appendix A for hyperparameters and other details.

**Results:** Figure 2 presents the test accuracy on each task at the end of the sequence, achieved by the different regularizers and activation functions on the network with hidden layer of size 128. Results on a network with hidden layer size 64 are shown in the Appendix B. Clearly, in all the different tasks, the representational regularizers show a superior performance to the other studied techniques. For the regularizers applied to the parameters, L2-WD and L1-Param do not exhibit a clear trend and do not systematically show an improvement over the use of the different activation functions only. While OrthReg shows a consistent good performance, it is lower than what can be achieved by the representational regularizers. It is worth noting the L1-Rep yields superior performance over
While the previous section focused on learning a sequence of tasks with completely different input patterns and same objective, we move to study the case of learning different categories of one dataset. For this we split the CIFAR-100 and the Tiny ImageNet (Yao & Miller, 2015) dataset into ten tasks, respectively. We have 10 and 20 categories per task for CIFAR-100 and Tiny ImagNet, respectively. Further details about the experimental setup can be found in the appendix A. We compare the top competing methods from the previous experiments, L1-Rep, DeCov and our SLNI, and No-Reg as a baseline, ReLU in previous experiment. Figures 3(a) and 3(b) show the performance on each of the ten tasks at the end of the sequence. For both datasets, we observe that our SLNI performs overall best. L1-Rep and DeCov continue to improve over the non-regularized case No-Reg. These results confirm our proposal on the importance of sparsity and decorrelation in sequential learning.
We have shown that our proposed regularizer which contradicts our regularizer’s role. Also, since the network is pretrained, the locality introduced learning without forgetting (Li & Hoiem, 2016) as a base network, following the setting of Aljundi et al. (2017). More details are in Section 3, when tasks have completely different input patterns, the neurons that were activated on the previous task examples will not fire for new task samples and exclusion of important neurons is not mandatory. However, when sharing is present between the different tasks, a term to prevent the previous task examples from causing any interference is required. This is manifested in the reported results: for permuted Mnist, all the variants work nicely alone, as a result of the simplicity and the disjoint nature of this sequence. However, in the Cifar 100 sequence, the integration of the neuron importance in the SNI without neuron importance (SNI), SNI with neuron importance (SNI), SLNI without neuron importance (SLNI) and SLNI with neuron importance (SLNI) in addition to our full SLNI regularizer. As we explained in Section 3 when tasks have completely different input patterns, the neurons that were activated on the previous task examples will not fire for new task samples and exclusion of important neurons is not mandatory. However, when sharing is present between the different tasks, a term to prevent SLNI from causing any interference is required. This is manifested in the reported results: for permuted Mnist, all the variants work nicely alone, as a result of the simplicity and the disjoint nature of this sequence. However, in the Cifar 100 sequence, the integration of the neuron importance in the SNI and SLNI regularizers exclude important neurons from the inhibition, resulting in a clearly better performance. The locality in SLNI improves the performance in the Cifar sequence, which suggests that a richer representation is needed and few active neurons can be tolerated.

4.3 SLNI with EWC (Kirkpatrick et al., 2016)

We have shown that our proposed regularizer SLNI exhibits stable and superior performance on the different tested networks when using MAS as importance weight preservation method. To prove the effectiveness of our regularizer regardless of the used importance weight based method, we have tested SLNI on the 5 tasks permuted Mnist sequence and obtained a boost in the average performance at the end of the learned sequence equals to 3.1% on the network of h-layer=128 and a boost of 2.8% on the network of h-layer=64, detailed accuracies are shown in Appendix B. It is worth noting that with both MAS and EWC our SLNI was able obtain better accuracy using a network of 64 hidden size than when training without regularization No-Reg on network of double the size 128 indicating that SLNI allows to use neurons much more efficiently.

4.4 Ablation Study

Our method can be seen as composed of three components: the neural inhibition, the locality relaxation and the neuron importance integration. To study how these components perform individually, Table 1 reports the average accuracy at the end of the Cifar 100 and permuted Mnist sequences for each variant, namely, SNI without neuron importance (SNI), SNI with neuron importance (SNI), SLNI without neuron importance (SLNI) in addition to our full SLNI regularizer. As we explained in Section 3, when tasks have completely different input patterns, the neurons that were activated on the previous task examples will not fire for new task samples and exclusion of important neurons is not mandatory. However, when sharing is present between the different tasks, a term to prevent SLNI from causing any interference is required. This is manifested in the reported results: for permuted Mnist, all the variants work nicely alone, as a result of the simplicity and the disjoint nature of this sequence. However, in the Cifar 100 sequence, the integration of the neuron importance in the SNI and SLNI regularizers exclude important neurons from the inhibition, resulting in a clearly better performance. The locality in SLNI improves the performance in the Cifar sequence, which suggests that a richer representation is needed and few active neurons can be tolerated.

4.5 Comparison with the state of the art

To compare our proposed approach with the different state-of-the-art sequential learning methods, we use a sequence of 8 different object recognition tasks, introduced in Aljundi et al. (2017). The sequence starts from AlexNet (Krizhevsky et al., 2012) pretrained on ImageNet (Russakovsky et al., 2015) as a base network, following the setting of Aljundi et al. (2017). More details are in Appendix A.4. We compare against the following: Learning without Forgetting (Li & Hoiem, 2016) (LwF), Incremental Moment Matching (Lee et al. 2017) (IMM), Path Integral (Zenke et al., 2017) and sequential finetuning (FineTuning), in addition to the case of MAS (Aljundi et al., 2017) alone, i.e. our No-Reg before. Compared methods were run with the exact same setup as in Aljundi et al. (2017). For our regularizer, we disable dropout, since dropout encourages redundant activations which contradicts our regularizer’s role. Also, since the network is pretrained, the locality introduced in SLNI may conflict with the already pretrained activations. For this reason, we also test SLNI with randomly initialized fully connected layers. Our regularizers is applied with MAS as a sequential learning method. Table 2 reports the average test accuracy at the end of the sequence achieved by each method. SLNI improves even when starting from a pretrained network and disabling dropout. Surprisingly, even with randomly initialized fully connected layers, SLNI improves 1.8% over the state of the art using a fully pretrained network.
5 CONCLUSION

In this paper we study the problem of sequential learning using a network with fixed capacity – a prerequisite for a scalable and computationally efficient solution. A key insight of our approach is that in the context of sequential learning (as opposed to other contexts where sparsity is imposed, such as network compression or avoiding overfitting), sparsity should be imposed at the level of the representation rather than at the level of the network parameters. Inspired by lateral inhibition in the mammalian brain, we impose sparsity by means of a new regularizer that decorrelates nearby active neurons. We integrate this in a model which learns selflessly a new task by leaving capacity for future tasks and at the same time avoids forgetting previous tasks by taking into account neurons importance.

REFERENCES


Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.


APPENDIX

A DETAILS ON THE EXPERIMENTAL SETUP

In all designed experiments, our regularizer is applied to the neurons of the fully connected layers. As a future work, we plan to integrate it in the convolutional layers.

A.1 PERMUTED MNIST

The used network is composed of two fully connected layers. All tasks are trained for 10 epochs with a learning rate $10^{-2}$ using SGD optimizer. ReLU is used as an activation function unless mentioned otherwise. Throughout the experiment, we used a scale $\sigma$ for the Gaussian function used for the local inhibition equal to $1/6$ of the hidden layer size. For all competing regularizers, we tested different hyper parameters from $10^{-2}$ to $10^{-9}$ and report the best one. For $\lambda_\Omega$, we have used a high $\lambda_\Omega$ value that ensures the least forgetting. This allows us to examine the degradation in the performance on the later tasks compared to those learned previously as a result of lacking capacity. Note that better average accuracies can be obtained with tuned $\lambda_\Omega$.

In section 4.1, we estimated the free capacity in the network with the percentage of $\Omega_k < 10^{-2}$, with $\Omega_k$, the importance weight multiplier estimated and accumulated over tasks. We consider $\Omega_k < 10^{-2}$ of negligible importance as in a network trained without a sparsity regularizer, $\Omega_{ij} < 10^{-2}$ covers the first 10 percentiles.

A.2 CIFAR-100

As a base network, we use a network similar to the one used by Zenke et al. (2017) but without dropout. We evaluate two variants with hidden size $N = \{256, 128\}$. Throughout the experiment, we again used a scale $\sigma$ for the Gaussian function equal to $1/6$ of the hidden layer size. We train the different tasks for 50 epochs with a learning rate of $10^{-2}$ using SGD optimizer.

A.3 TINY IMAGE NET

We split the Tiny ImageNet dataset (Yao & Miller, 2015) into ten tasks, each containing twenty categories to be learned at once. As a base network, we use a variant of VGG (Simonyan & Zisserman, 2014). For architecture details, please refer to Table 3 below.

<table>
<thead>
<tr>
<th>Layer</th>
<th># filters/neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>64</td>
</tr>
<tr>
<td>Max Pooling</td>
<td>-</td>
</tr>
<tr>
<td>Convolution</td>
<td>128</td>
</tr>
<tr>
<td>Max Pooling</td>
<td>-</td>
</tr>
<tr>
<td>Convolution</td>
<td>256</td>
</tr>
<tr>
<td>Max Pooling</td>
<td>-</td>
</tr>
<tr>
<td>Convolution</td>
<td>256</td>
</tr>
<tr>
<td>Max Pooling</td>
<td>-</td>
</tr>
<tr>
<td>Convolution</td>
<td>512</td>
</tr>
<tr>
<td>Convolution</td>
<td>512</td>
</tr>
<tr>
<td>Fully connected</td>
<td>500</td>
</tr>
<tr>
<td>Fully connected</td>
<td>500</td>
</tr>
<tr>
<td>Fully connected</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 3: Architecture of the network used in the Tiny ImageNet experiment.

Throughout the experiment, we again used a scale $\sigma$ for the Gaussian function equal to $1/6$ of the hidden layer size.
A.4 8 TASK OBJECT RECOGNITION SEQUENCE

The 8 tasks sequence is composed of: 1. Oxford Flowers (Nilsback & Zisserman, 2008), 2. MIT Scenes (Quattoni & Torralba, 2009), 3. Caltech-UCSD Birds (Welinder et al., 2010), 4. Stanford Cars (Krause et al., 2013), 5. FGVC-Aircraft (Maji et al., 2013); 6. VOC Actions (Everingham et al.); 7. Letters (de Campos et al., 2009); and 8. SVHN (Netzer et al., 2011) datasets. We have rerun the different methods and obtain the same reported results as in Aljundi et al. (2017).

B  EXTRA RESULTS

B.1 PERMUTED MNIST SEQUENCE

In section 4.1, we have studied the performance of different regularizers and activation functions on 5 permuted Mnist tasks in a network with a hidden layer of size 128. Figure 5 shows the average accuracies achieved by each of the studied methods at the end of the learned sequence in a network with a hidden layer of size 64. Similar conclusions can be drawn. Maxout and LWTA perform similarly and improve slightly over ReLU. Regularizers applied to the representation are more powerful for sequential learning than regularizers applied directly to the parameters. Specifically, L1-Rep (orange) is consistently better than L1-WD (pink). Our SLNI is able of maintaining a good performance on all the tasks, achieving among the top average test accuracies. Admittedly, the performances of SLNI is very close to L1-Rep. The difference between these methods stands out more clearly for larger networks and more complex tasks.

Figure 5: Comparison of different regularization techniques on 5 permuted MNIST sequence of tasks, hidden size=64. Representation based regularizers are solid bars, bars with lines represent parameters regularizers, dotted bars represent activation function. See Figure 2 for size 128.

B.2 SLNI WITH EWC

To show that our approach is not limited to MAS (Aljundi et al., 2017), we have also experimented with EWC (Kirkpatrick et al., 2016) as another importance weight based method along with our regularizer SLNI on the permuted Mnist sequence. Figure 6 shows the test accuracy of each task at the end of the 5 permuted Mnist sequence achieved by our SLNI combined with EWC and by No-Reg, indicating EWC alone. It is clear that SLNI succeeds to improve the performance on all the learned tasks which validates the utility of our approach with different sequential learning methods.

B.3 CIFAR 100 SEQUENCE

In section 4.2, we have tested our SLNI and other representation regularizers on the Cifar 100 sequence. In Figure 3(a) we compare their performance on a network with hidden layer size 256. Figure 7 repeats the same experiment for a network with hidden size 128. While DeCov and SLNI continue to improve over No-Reg, L1-Rep seems to suffer in this case. Our interpretation is that L1-Rep here interferes with the previously learned tasks while penalizing activations and hence suffers from catastrophic forgetting. In line with all the previous experiments SLNI achieves the best accuracies and manages here to improve over 6% compared to No-Reg.
To avoid penalizing all the active neurons, our SLNI weights the correlation penalty between each two neurons based on their spatial distance using a Gaussian function. We want to visualize the effect of this spatial locality on the neurons activity. To achieve this, we have used the first 3 tasks of the Permutated Mnist sequence as a test case and visualized the neurons importance after each task. This is done using the network of hidden layer size 64. Figure 8, Figure 9 and Figure 10 show the neurons importance after each task. The left column is without locality, i.e. SNI, and the right column is SLNI. Blue represents the first task, orange the second task and green the third task. When using SLNI, inhibition is applied in a local manner allowing more active neurons which could potentially improve the representation power. When learning the second task, new neurons become important regardless of their closeness to first task important neurons as those neurons are excluded from the inhibition. As such, new neurons are becoming active as new tasks are learned. For SNI all neural correlation is penalized in the first task. And for later tasks, very few neurons are able to become active and important for the new task due to the strong global inhibition, where previous neurons that are excluded from the inhibition are easier to be re-used.
Figure 8: First layer neuron importance after learning the first task. Left: SNI, Right: SLNI.

Figure 9: First layer neuron importance after learning the second task. Left: SNI, Right: SLNI.

Figure 10: First layer neuron importance after learning the third task. Left: SNI, Right: SLNI.