CONTINUAL LEARNING WITH GROUP-WISE NEURON NORMALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning focuses on methods that accommodate the change in distribution and allow model adaptation and evolution while receiving data continuously. Importance and regularization -based weight update methods that rely on heuristics might not be effective. Recently, enhanced experience replay-based methods showed promising results but might add to the computational cost. In this paper, we propose simple parameter-free normalization over groups of distinct neurons at the penultimate layer of the used neural network and a straightforward experience replay algorithm. We argue that such normalization enables the network to balance its capacity for each task, reducing the chances of damaging interference between tasks and mitigating forgetting. Our evaluation shows that normalization over groups of neurons drastically impacts performance. We demonstrate improved retained accuracy and backward transfer with respect to related state-of-the-art methods while computationally efficient.

1 INTRODUCTION

Continual learning (CL) studies routines that allow the model to adapt when receiving data continuously. With the goal to accommodate the changes in the data distribution, the objective is to enable the model to generalize for unseen data while at the same time preserve performance for already observed past data. As appealing as continual learning is for computer vision, robotics, and machine learning applications, efficacy and efficiency are challenging. One of the main reasons is *catastrophic forgetting* (French, 1999; McClelland et al., 1995). It pinpoints a failure mode in the so-called stability-plasticity trade-off (Grossberg, 1980). It occurs when the data used for learning come sequentially (one after the another) within the different tasks that have distinct mutually independent probability distributions (Robbins, 2007; Bottou et al., 2018).

Many existing CL approaches address forgetting using importance based weight update (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Kolouri et al., 2019; Nguyen et al., 2017; Titsias et al., 2019; Ebrahimi et al., 2019; Ritter et al., 2018; Bennani et al., 2020), and replay-based rehearsal (Farajtabar et al., 2020; Riemer et al., 2019; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Javed & White, 2019; Gupta et al., 2020). (Srivastava et al., 2013) show that a biologically inspired local winner-take-all (LWTA) neuron activation can help to prevent forgetting. In the forward propagation, LWTA indices competition between a local subset of neurons over a layer. A fraction of the units are inactive for each input, utilizing only a subset of the network parameters. In a somewhat related line (Kaushik et al., 2021), show that a network with relevance mapping can learn an optimized representational overlap that overcomes the problem of catastrophic forgetting at the expense of adding additional memory. (Sokar et al., 2020; Abati et al., 2020) change/extend the neural network structure, and apply hand-crafted rules to grow the network which can lead to increase in memory. (Serrà et al., 2018) focus on learning hard attention mask concurrently by using regularization, while the authors in (Wen et al., 2020; Jung et al., 2020) constraints (adaptive/learned or manually rule-based constructed) by adding additional learning parameters/modules that increases the learning time. (Goodfellow et al., 2014; Mirzadeh et al., 2020), show that a dropout induces a gating mechanism such that for different tasks, different paths of the network are active. On the other hand, enhanced experience replay-based methods (Riemer et al., 2019; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Javed & White, 2019; Gupta et al., 2020) showed great potential since they avoid network growth and do not use importance based heuristics in the weighs update. Such methods use a small memory, dubbed episodic memory, that stores data samples from past tasks and

then replay them (Murre, 1992; Lopez-Paz & Ranzato, 2017) when training for future tasks. They focus on weights update with minimum interference between i) the gradient using the data from the current task and ii) the gradient using replayed data from the past tasks, to improve performance on the current task and avoid catastrophic forgetting.

In the past batch norm (Ioffe & Szegedy, 2015), group norm (Wu & He, 2018) and layer norm (Ba et al., 2016) were used for prediction tasks, but such normalization mechanisms/techniques (or even a simpler ones) were not much studied in the context of continual learning. To that end, in this paper, we propose a very simple normalization over groups of distinct neurons at the penultimate layer of the neural network and study its impact in a continual learning regime with straightforward experience replay (Murre, 1992). As a group-wise normalization we use normalization by the ℓ_2 norm, that is computationally efficient and parameter-free (we do not use additional parameters that can increase memory). It results in groups of neurons that have the same strength (unit norm), but different sparsity levels. As we show in the paper, this induces "soft" weighing during the back-propagation and promotes different learning dynamics compared to (Srivastava et al., 2013). In LWTA the errors back-propagate only through the active LWTA neurons. Whereas our normalization induces more strong weighing in the back-propagation trough the groups of neurons that are more sparse. We show that the group-wise neuron normalization triggers competition between the subset of weighs that connect the neurons at the penultimate layer and the layer before in terms of how much learning they receive (which is different from (Srivastava et al., 2013), where the competition is local, *i.e.*, within the group). We found that this property is crucial, since subset of network weights that connect to the groups of neurons which are more sparse receive "more learning". The corresponding gradients for these subsets of weights have similar sparsity level, but the gradients for the subsets of network weights that connect to the groups of neurons which are more sparse have higher strength (higher value of the ℓ_2-norm for the corresponding gradient), effecting the learning of the network weights. This seems important since when the learning is more focused on the subset of weights that directly connect to the corresponding sparse groups of neurons, we have less unwanted interference and neural network parameters override, since these parameters are considered/assumed to be more important (Srivastava et al., 2013) for the data samples (or batch of samples) during learning.

To verifiy our hypothesis that as we have sparser group of neurons, we have more learning for the directly connected corresponding subset of weights that affect all network parameters, we preform extensive empirical evaluation. We evaluate our approach on various benchmarks, including MNIST (rotation and permutation), CIFAR10, CIFAR100, and TinyImageNet. We empirically experiment in task-aware (single-pass and multi-pass) and task-agnostic continual learning setups, where we use NN architectures with convolutional and feed-forward layers. Our empirical results demonstrate that our simple normalization drastically impacts performance. We demonstrate that it leads and accommodates balanced use of the network capacity, achieves stable and efficient feature use, allows to discriminate more between the different representations and enables less forgetting. Differently than (Srivastava et al., 2013) that uses LWTA at each layer, we show that it is sufficient to use the normalization only at the penultimate layer. We show that our approach improves performance (or is on-pair) compared to existing state-of-the-art methods, while computationally efficient.

2 Related Work

In this section, we overview the resent, most closely related and prominent work, without pretending to be exhaustive. Regularization methods explicitly regularize the model during training so that it can avoid changes that may degrade the performance for already visited tasks. Commonly, such approaches include penalty for significant changes of important weights measured using Fisher information matrix or synaptic importance (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Kolouri et al., 2019). In Srivastava et al. (2013), the authors showed that LWTA neuron activation helps to prevent catastrophic forgetting. Differently then (Srivastava et al., 2013), during learning, our normalization induces "soft" weighting of the back-propagated error. In (Wen et al., 2020), the authors use out-of-network, task-dependent biasing units to accommodate/fight catastrophic forgetting. In (Kaushik et al., 2021), the authors introduce Relevance Mapping Networks (RMNs), showing that such network can learn an optimized representational overlap that overcomes the twin problem of catastrophic forgetting and remembering. In (Goodfellow et al., 2014), the authors empirically investigate the influence of common activation functions and dropout in a CL. Similarly, in (Mirzadeh et al., 2020), the authors argue that a stable network with dropout learns a

gating mechanism such that for different tasks, different paths of the network are active. In (Jung et al., 2020), the authors propose a regularization-based continual learning method where they add two explicit penalties, one for the task important and the other for the task non-important weights. In (Serrà et al., 2018), the authors propose a task-based hard attention mechanism to preserve previous tasks information without affecting the current task's learning. To do so, they learn a hard attention mask concurrently to every task. Here, we do not use regularization, neither we use any constraints (adaptively learned or manually rule-based constructed). We also do not add additional learning parameters/modules that can increase the learning time nor we use dropout. We focus on an algorithm that uses group-wise non-parametric normalization with experience replay. The authors in (Sokar et al., 2020), propose a sort of rule based controlled network expansion method referred as SpaceNet. Our approach focuses on balanced use of the network capacity, but differently than (Sokar et al., 2020) we do not use any rules to grow the network. The authors in (Abati et al., 2020) equip each convolutional layer with task-specific gating modules, and select which filters to apply on the given input in order to protect and promote important filters while ensuring no loss in the performance. In our approach, we do not change/extend the neural network structure, rather we only use the group-wise normalization as a special type of activation. Rehearsal techniques have achieved promising results by mitigating forgetting through replaying a subset of samples from previous tasks. Some methods in this category store random samples in a memory which are later used to optimize the model (Robins, 1995; Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017; Rolnick et al., 2018; Chaudhry et al., 2019). In pseudo-rehearsal techniques, a generative model is trained alongside the main model to generate images for tasks seen so far (Shin et al., 2017; Ayub & Wagner, 2021). In continual-meta learning, a meta-model is constantly updated over a sequence of tasks, and the goal is to quickly adapt the previous tasks with minimal optimization steps (He et al., 2019; Harrison et al., 2019; Antoniou et al., 2020). In (Riemer et al., 2019), the authors proposed an online version of meta-continual learning based on the first-order meta-learning algorithm (Nichol et al., 2018). The authors in (Gupta et al., 2020) proposed an online meta-learning approach based on (Finn et al., 2017) by using a look-ahead mechanism to learn the learning rate. We relay on first-order algorithm without this is computationally efficient.

3 PRELIMINARIES

We denote the input as $\mathbf{x}$, the target y and the neural network mapping as $h(\mathbf{x};\theta)$ with parameters θ containing the respective weighs and biases. We denote the weights connecting layers $l-1$ and l as $\mathbf{W}^l$, while the linear mapping as $\mathbf{z}^l = \mathbf{W}^l\mathbf{a}^{l-1}$ and the nonlinear activation as $\mathbf{a}^l = v(\mathbf{z}^l)$. In the past it was shown that linear activation $\mathbf{a}^l = \mathbf{z}^l$ is as powerful as a one-layer linear model (Goodfellow et al., 2016). While when the activation is ReLu, the negative values in $\mathbf{z}^l$ are set to zero, *i.e.*, $a_i^l = z_i^l$, if $z_i^l > 0$, and $a_i^l = 0$, otherwise, (Agarap, 2018). In common learning setups, not including continual learning, it was shown that ReLu overcomes the vanishing gradient problem, allowing models to learn faster and perform better (Agarap, 2018). To define the LWTA activation, we first extract distinct neurons into groups, *i.e.*, $\mathbf{z}^l = [(\mathbf{z}_1^l)^T, (\mathbf{z}_2^l)^T, ..., (\mathbf{z}_J^l)^T]^T$, where $\mathbf{z}_j^l$ are linear activation's that belong to group j. Similarly, $\mathbf{a}^l = [(\mathbf{a}_1^l)^T, (\mathbf{a}_2^l)^T, ..., (\mathbf{a}_J^l)^T]^T$, where we compute the local winner-take-all neuron activation as $a_{j,i}^l = z_{j,i}^l$, if $z_{j,i}^l = \max_i \mathbf{z}_j^l$ and $a_{j,i}^l = 0$ otherwise, (Srivastava et al., 2013). A fraction of the units within the group are inactive depending on the input, and might utilize only a subset of the network parameter. The authors in (Wu & He, 2018), propose generalized form that describes a family of feature normalization methods, including batch normalization (BN) (Ioffe & Szegedy, 2015), group normalization (GN) (Wu & He, 2018), layer normalization (LN) (Ba et al., 2016) and instance normalization (IN) (Ulyanov et al., 2016). During learning BN, GN, LN and IN normalize neuron activation's by mean and variance, and differ depending over which axis (batch, channel, height or width) the mean and variance are computed (Wu & He, 2018). Afterwards, BN, GN and LN learn linear transform (scale and shift) to compensate for the possible lost of representational ability. Regarding continual learning with experience replay, the authors in (Pham et al., 2022) propose continual normalization (CN) that take into account the strengths of BN and GN. However, CN might struggle when the normalization statistics become outdated. The existing normalization techniques used in common learning (iid regime) Ioffe & Szegedy (2015); Wu & He (2018); Ba et al. (2016) or continual learning (Pham et al., 2022) use learning parameters in the normalization. The advantage of the our approach is its simplicity; −we do not use additional learning parameters (scale and shift) and do not require additional continual learning strategy that might be necessary and could increase computation.

4 PROPOSED APPROACH

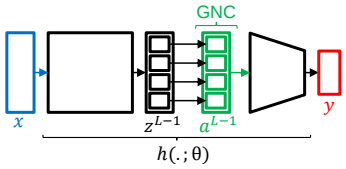

Figure 1: Schematic illustration of an NN with the proposed group-wise normalization at the penultimate $(L-1)$ layer.

Inspired by (Lopez-Paz & Ranzato, 2017; Gupta et al., 2020), we consider an online continual learning scenario, where the learning model continually observes training data sequences.

We propose parameter-free normalization over groups of neuron activation's (we visualize the schematic illustration of group-wise normalization at layer $L-1$ in Fig. A). We apply it at the penultimate layer of the neural network. During learning, we use straightforward experience replay characterized by a small memory (reservoir $\mathbf{R}$), where we store tuples data $\mathbf{x}_b$ and labels $\mathbf{y}_b$ from past tasks. We replay the episodic memory Murre (1992) when we train for future tasks. An advantage of experience replay is that it avoids network growth and does not use importance-based heuristics. In the following, we describe the proposed group-wise normalization and then present our simple learning algorithm.

4.1 GROUP-WISE NORMALIZATION

Similarly to the LWTA activation, we first extract distinct neurons into groups, *i.e.*, $\mathbf{z}^l = [(\mathbf{z}_1^l)^T, (\mathbf{z}_2^l)^T, ..., (\mathbf{z}_J^l)^T]^T$, where $\mathbf{z}_j^l$ are linear activation's that belong to group j. We define the group-wise normalization for $\mathbf{z}_j^l$ as:

$$\mathbf{a}_j^l = v_{gnor}(\mathbf{z}_j^l) = \frac{\mathbf{z}_j^l}{\|\mathbf{z}_j^l\| + \epsilon}, \forall j \in \{1, .., J\}, \tag{1}$$

where $\|\mathbf{z}_j^l\|$ is the ℓ_2-norm of $\mathbf{z}_j^l$ and ϵ is small constant (to regularize the denominator in case $\mathbf{z}_j^l$ is the zero-vector). Afterwards, we construct $\mathbf{a}^l$ as $\mathbf{a}^l = [(\mathbf{a}_1^l)^T, (\mathbf{a}_2^l)^T, ..., (\mathbf{a}_J^l)^T]^T$. Before the normalization, we can also apply nonlinear activation, such as ReLu Agarap (2018) $\mathbf{a}_j^l = v_{gnor}(\text{ReLu}(\mathbf{z}_j^l))$ or sparse coding transform (sT) (Ravishankar & Bresler, 2015) $\mathbf{a}_j^l = v_{gnor}(\text{sT}(\mathbf{z}_j^l))$, where $\text{sT}(\mathbf{z}_j^l) = sign(\mathbf{z}_j^l) \odot \max(|\mathbf{z}_j^l| - \lambda_j, 0)$, while $\lambda_j = \mathbf{1}^T|\mathbf{z}_j^l|$ is a sparsifying threshold, $|\mathbf{z}_j^l|$ denotes the vector of absolute values for $\mathbf{z}_j^l$.

The group-wise normalization (**gnor**) affects the max response (within the hole representation), so the max neuron activation before and after the normalization might not be at the same neuron location, which is in contrast to LWTA that has only a local effect within the group locality. After the normalization all groups of neuron activations have unit norm and differ by their sparsity level.

Without the normalization the sparsity level might be high, but the strength of the activations for the group of neurons might be low. Thus, the parameters connecting group of neurons at the penultimate layer and the previous layer (as well as the parameters of the network before the penultimate layer) might miss the required learning. As we describe in the following section, this very simple normalization seems to impose learning dynamics that surpasses this.

4.2 LEARNING ALGORITHM

We assume that there are T tasks. At each task t the model observes tuples $(\mathbf{x}_t, \mathbf{y}_t) \sim (\mathbf{X}_t, \mathbf{Y}_t)$ for that task, where $\mathbf{x}_t$ are the data and $\mathbf{y}_t$ are the labels. Per each observation, we also sample the memory reservoir $(\mathbf{x}_b, \mathbf{y}_b) \sim \mathbf{R}$. We denote the cross-entropy loss function as $\mathcal{L}(h(\mathbf{x}; \theta), \mathbf{y})$, where $\mathbf{x} = [\mathbf{x}_t, \mathbf{x}_b]$ and $\mathbf{y} = [\mathbf{y}_t, \mathbf{y}_b]$ and focus on learning the parameters θ that minimize it. In our learning algorithm, at task t, per one observation, one iteration with experience-replay consist of the following gradient-based update (Goodfellow et al., 2016):

$$\mathbf{W}^l = \mathbf{W}^l - \beta \frac{\partial \mathcal{L}(h(\mathbf{x}; \theta), \mathbf{y})}{\partial \mathbf{W}^l}, \tag{2}$$

where $\frac{\partial \mathcal{L}(h(\mathbf{x};\theta),\mathbf{y})}{\partial \mathbf{W}^l}$ is the gradient of $\mathcal{L}(h(\mathbf{x}; \theta), \mathbf{y})$ with respect to $\mathbf{W}^l$ using the data observation at the current task and the data from the memory buffer, and $\beta > 0$ is the learning rate .

Learning Dynamics. In order to see the influence of gnor, during back-propagation, we are interested in the gradient of the nonlinear activation at the penultimate layer ($l = L-1$) with respect

to its arguments, *i.e.*, $\mathbf{G}^{L-1} = \frac{\partial \mathbf{a}^{L-1}}{\partial \mathbf{z}^{L-1}}$[1]. In the case when we have linear activation at the penultimate layer, $\mathbf{G}^{L-1}$ is identity matrix. When using ReLu, $\mathbf{G}^{L-1}$ is again diagonal, but sparse, with values defined as follows $G_{ii}^{L-1} = 1$, if $z_i^l > 0$ and $G_{ii}^{L-1} = 0$, otherwise. When we use the LWTA, then $\mathbf{G}^{L-1}$ is block-diagonal matrix, consisting of J diagonal matrices $\mathbf{G}_j^{L-1}$. Moreover, $G_{j,ii}^{L-1} = 1$, if $z_{j,i}^{L-1} = \max_i \mathbf{z}_j^{L-1}$, and $G_{j,ii}^{L-1} = 0$, otherwise. When we use the gnor after linear activation, we also have block-diagonal matrix $\mathbf{G}^{L-1}$ consisting of $\mathbf{G}_j^{L-1}$ symmetric matrices ($\mathbf{G}_j^{L-1} = (\mathbf{G}_j^{L-1})^T$) that have diagonal ($G_{j,ii}^{L-1}$) and off-diagonal elements ($G_{j,ik}^{L-1} = G_{j,ki}^{L-1}$), *i.e.*,

$$G_{j,ii}^{L-1} = \frac{1}{\|\mathbf{z}_j^{L-1}\|}\left(1 - \frac{(z_{j,i}^{L-1})^2}{(\mathbf{z}_j^{L-1})^T\mathbf{z}_j^{L-1}}\right), G_{j,ik}^{L-1} = -\frac{1}{\|\mathbf{z}_j^{L-1}\|}\frac{z_{j,i}^{L-1}z_{j,k}^{L-1}}{(\mathbf{z}_j^{L-1})^T\mathbf{z}_j^{L-1}}. \tag{3}$$

Compared to the gradient of the Linear, the ReLu and the LWTA activation, equation 3 differs significantly. The main difference is the block-diagonal structure, and the terms with the highest contribution $\frac{1}{\|\mathbf{z}_j^{L-1}\|}$ and $1 - \frac{1}{2}\frac{(z_{j,i}^{L-1})^2}{(\mathbf{z}_j^{L-1})^T\mathbf{z}_j^{L-1}}$. In contrast to ReLu and LWTA, that act as hard "gating" with values ether 0 or 1, equation 3 actually acts as a soft "gating".

Interestingly, here the weighting induced by the group-wise normalization is "localized" per group in the back-propagation. The normalization induces back-propagation weighting with strength inversely proportional to the norm of the unnormalized neuron activation in the group. The local weighting values within the group differ more when the group sparsity is higher. Thus, the normalization is causing a suppression of back-propagation through groups of neurons that have strong activation's (before the normalization) but are not sparse. While at the same time the normalization pronounces back-propagation through groups of neurons that are sparse within the group locality, regardless of the intensity of the neuron activation before the normalization.

Competition Mechanism. This back-propagation weighting actually translates to competition between the subset of weights that connect to the corresponding groups. In fact, the weighting induced by the group of neurons with the highest sparsity causes the directly connected subset of weights to receive "most of the learning", i.e. win the competition. In our empirical evaluation, we demonstrate that the group of neurons at the penultimate layer with the highest sparsity induces weighing such that the directly connected subset of weights has the gradient with highest magnitude among these subsets of weights. Now, since the computations in the back-propagation at earlier layers depends on the later layers, the normalization influences not only the update in the network weights that connect layer $L-1$, but also the update for the remaining weights before it.

5 Empirical Evaluation

Our empirical evaluation includes ablations, study of the importance of the used components in our approach and the comparison with the state-of-the-art methods. We performed the evaluation using: (i) different CL regimes, including task-aware (multi-pass and single-pass) and task-agnostic, (ii) 5 computer vision data sets, (iii) multiple baselines, including 6 basic ones and 12 existing works, and 3 different neural networks.

5.1 Data Sets, Baselines and Setup

Datasets. We used MNIST Rotations, MNIST Permutations Lopez-Paz & Ranzato (2017), CIFAR-10, CIFAR-100 Krizhevsky (2009) and tinyImageNet Le & Yang (2015). The MNIST Rotations comprises of tasks that contain MNIST digits rotated by a different common angle in $[0, 180]$. The MNIST Permutations consists of tasks generated by shuffling the image pixels by a fixed random permutation. Both MNIST Permutation and MNIST Rotation have 20 tasks with 1000 samples per task. The CIFAR100 dataset Krizhevsky (2009) contains 60000 color images with dimension of 32x32. It includes 100, real-life classes. CIFAR10 dataset Krizhevsky (2009) is similar to CIFAR100, but it includes 10 classes. On CIFAR10 and CIFAR100, we use 5 classes per task, having 2 and 20 tasks in total, respectively. The TinyImageNet dataset Le & Yang (2015) consists of large

[1]We note that $\mathbf{g}^{L-1}$ appears in the back-propagation. To see this more obviously, we can express equation 9 in alternative form (Rumelhart et al., 1986), details in the Appendix

number of images and a total of 200 classes. The images are in color and with dimension of 64x64. On tinyImageNet, we use 5 classes per task, and have 40 tasks in total.

Baselines. We compare our approach with a model trained in iid setup (upper bound baseline), the iCARL (Rebuffi et al., 2017), ER, MER (Riemer et al., 2019), GEM (Lopez-Paz & Ranzato, 2017), AGEM (Chaudhry et al., 2019), La-MAML (Gupta et al., 2020), META-BGD (Zeno et al., 2019), LWTA (Srivastava et al., 2013), BN (Ioffe & Szegedy, 2015), GN (Wu & He, 2018), LN (Ba et al., 2016) and CN (Pham et al., 2022) methods. In the evaluation of our approach, we follow the continual learning setups as proposed by (Javed & White, 2019; Gupta et al., 2020). We use `https://github.com/montrealrobotics/La-MAML` as a code base for some of the baselines, while we implemented (in pythorch) the remaining baselines and our algorithm.

Continual Learning Setup. We consider a continual learning scenario where we learn by observing data from a set of sequentially arriving classification tasks. Using the CIFAR10, CIFAR100, and TinyImagenet, we evaluate task-aware experiments using a multi-headed model with known task identity. In the task-aware setup, we evaluate our approach under two training regimes. We consider the challenging (single-pass) regime Chaudhry et al. (2019). It consists of processing incoming data for every task in only one pass. Once we process the data samples, they are no longer accessible unless we add them to a replay memory. We also consider the multiple-pass regime, where once we have access to the data, we can perform offline training until convergence for every task. On the MNIST data sets (Rotations and Permutations), we use a single-head model and perform task-agnostic experiments, where the task identity is unknown at training and test-time.

Used Neural Networks and Hyper-parameters. We use three neural network architectures typically used for vision tasks, widely exploited in this domain. We use 2 layer feed-forward neural network for the MNIST data set and convolutional neural networks architectures for the CIFAR10, CIFAR100, and TinyImageNet data sets. The architectures are equivalent to the ones used by (Gupta et al., 2020), with the difference that at the penultimate layer, we use sT followed by gnor. We also experimented with gnor and LWAT applied i) at the penultimate layer of the architecture and ii) at all feed-forward layers in the architecture. As hyper-parameters, we have a learning rate (LR), batch size, memory buffer size, memory buffer batch size, the number of groups, and group type. We validated with LR's of [.1, .3, .4, .7] and found that a value of .3 and .4 works well. We set the batch size to be same as the one used in the comparing algorithms for the respective learning setups as well as run all methods with the same number of epochs for a fair comparison. In the experiments, we state the used memory buffer size and the memory buffer batch size for each experiment and per each data set. We present ablation results as well as we show results about the influence when we vary the size of the local group, the used nonlinearity, and the grouping type.

Performance Metrics and Computing Infrastructure. We use the retained accuracy (RA) metric to compare various approaches, computed as the average accuracy of the model across tasks at the end of training. We also report the backward-transfer and interference (BTI) values. The BTI measures the average change in the accuracy of each task. We define it as the average difference between i) the accuracy after learning on a single task and ii) the accuracy after learning on all tasks. A smaller BTI implies lesser forgetting during training. The results provided here are an average of three runs. As computing hardware, we use $1/5$ of the available resources from the NVIDIA DGX2 station with 3.3GHz CPU and 1.5TB RAM, which has a total of 16 1.75GHz GPUs, each with 32GB memory (using 2 GPUs). Computing all experiments using the proposed algorithm takes 7-8 days, while it needs additional time for the baselines.

5.2 Results

Why the Normalization Works. We consider continual learning in the single-pass regime using the CIFAR100 data set. As a model, we experimented with convolution neural network (Gupta et al., 2020), where we apply gnor at the penultimate layer and use 32 groups of 10 neural activation's each. We set the reservoir size to 200 and replay memory batch size to 10.

To see why the method works, we want to understand the connection between (a) the sparsity of group of neurons at the penultimate layer and (b) the average gradient sparsity and (c) the average gradient strength for subset of weighs that connect the corresponding group of neurons at the penultimate layer with the layer before. Therefore, we monitor/analyze such gradients in two setups. In the first setup, we compute the average gradient for the subset of weighs connecting to the

Table 1: The sparsity of group of neurons at the penultimate layer. The average gradient sparsity and the average gradient strength for subset of weighs that connect the corresponding group of neurons at the penultimate layer with the layer before when without (wo) and with (w) group-wise normalization.

GR SPASR		.933	.768	.688	.643	.599	.560	.521	.478	.440	.389	.322
WO NORM	$s_l(\Delta \mathbf{W}_j^l)$	.502	.502	.504	.502	.502	.503	.502	.502	.502	.502	.502
	$\|\Delta \mathbf{W}_j^l\|_2$	4.13	4.11	4.15	4.13	4.13	4.15	4.15	4.14	4.14	4.16	4.17
W NORM	$s_l(\Delta \mathbf{W}_j^l)$	.680	.681	.680	.681	.681	.680	.681	.680	.681	.680	.679
	$\|\Delta \mathbf{W}_j^l\|_2$	.237	.234	.231	.231	.229	.228	.227	.226	.224	.222	.218

TASK ID		1	3	5	7	9	11	13	15	17	19
WO NORM	$s_l(\Delta \mathbf{W}_j^l)$	.700	.860	.818	.654	.600	.475	.336	.268	.247	.219
	$\|\Delta \mathbf{W}_j^l\|_2$	1.55	1.13	2.26	3.50	3.35	4.65	4.41	4.65	7.21	8.52
W NORM	$s_l(\Delta \mathbf{W}_j^l)$	.650	.688	.690	.692	.691	.682	.682	.683	.669	.673
	$\|\Delta \mathbf{W}_j^l\|_2$	.509	.263	.244	.238	.226	.211	.180	.156	.202	.186

corresponding group of neurons over all tasks during learning. In the second setup, we compute the average gradient for the subset of weighs connecting to the corresponding group of neurons but per task and for all subsets of weights during learning. In the two setups, we considered the cases where we do continual learning (i) with group-wise normalization and (ii) without group-wise normalization. To show the influence of the sparsity of the group of neurons, we order the groups of neurons according to their sparsity level. Then compute the average for the sparsity of the gradient and the average for the strength of the gradient for the subset of the corresponding weighs, but sorted using the previously computed sort index for the sparsity of the group of neurons. We compute the sparsity using the method proposed by Hoyer (2004). In the following we present the results. In Tab. 1 on the top, we show the results for the first setup. In the first row, we have the sorted average sparsity for the groups of neurons. In the second row, we show the average gradient sparsity (ordered by the sort index for the sparsity of the group of neurons) for the subsets of the corresponding weighs that connect the penultimate layer and the layer before. We can see that the sparsity levels for the gradient of the subsets of the corresponding weighs do not change as the group sparsity level changes. The gradient strength for the subsets of the corresponding weighs do not change when we do not use normalization, while the gradient strength is higher for the groups with higher sparsity. We assume that a sparser activation for a group of neurons represents a more distinctive feature for the corresponding input sample (or batch of samples). Therefore, our argumentation is that when the learning is more focused on the subset of weights that directly connect to the corresponding sparse groups of neurons, we have less unwanted interference and neural network parameters override, since these parameters are considered/assumed to be more important for that sample (or batch of samples). On the contrarily when the learning is not focused (in the case when we do not use any other mechanisms) it might happen that we override parameters that are important for other samples (tasks). The results in Tab. 1 on the top verifies our hypothesis that as we have sparser group of neurons, we have more learning for the directly connected corresponding subset of weights. Thus, our approach with its learning dynamics, benefits CL in avoiding parameter overriding and forgetting. In Tab. 1 on the bottom, we show the results for the second setup. The results demonstrate that in the case when we use only experience replay, the sparsity level of the gradient for the subset of corresponding weights that connect to the penultimate layer increases, while the average strength of the same gradient decreases. In the case when we use normalization the average sparsity level of the gradient for the subset of corresponding weights that connect to the penultimate layer remains constant, while the average strength decreases.

Table 2: The override/overlap and the change in the neuron activation at the penultimate layer.

	CIFAR100	
	ER ONLY	GNOR
OVERLAP	12.3±184.	**3.19** ±6.12
CHANGE	11.5±.120	**4.30** ±.030

To further support our claims and further see the impact of the gnor, we also run experiments in a setup as the privuis experiment for our approach and the baseline without gnor, where we measured (i) the mean of the override/overlap between the representations from different tasks at the penultimate layer and (ii) the change in the average activation of the representation at the penultimate layer for the task samples during the continual learning over sequence of tasks. In Tab. 2, we can see that when we use gnor, we have less change in the representations and less overlap between the representations. We explain this using similar augmentation as earlier, it seems that it also impacts the override/overlap of the representations at the penultimate layer.

Ablations and Impact of gnor. Our ablation includes continual learning in the single-pass regime using the CIFAR100 data set. As a model, we experimented with convolution neural network (Gupta et al., 2020), where we apply gnor at the penultimate layer and use 32 groups of 10 neural activation's each. We set the reservoir size to 200 and replay memory batch size to 50.

Table 3: Ablation for gnor, we show retained accuracy and backward transfer.

	RA	BTI
BASELINE	34.84±0.86	-19.07±0.02
ONLY ER	54.74±0.80	-09.18±0.03
ONLY GNOR	55.02±0.91	-05.67±0.36
GNOR+ER	**60.21**±3.76	-05.81±1.67

In Tab. 3, we show the results for the gnor ablation. We see that the gnor drastically impacts both performance metrics, giving a huge gain in the retained accuracy. Our algorithm with only the group-wise normalization (wo er) at the penultimate layer has better performance than the experience replay-replay based one (wo gn), supporting the claims about the effectiveness of the used gnor.

Due to space limitations, we leave the results about the influence of the group size, the used nonlinear activation, and the grouping type in the appendix, together with the results about the impact of the replay memory batch size.

Table 4: Comparative results when using different normalization techniques, we show retained accuracy and backward transfer.

	CIFAR100	
NORM	RA	BTI
WO N	54.74±0.80	-09.18±0.03
BN	**61.01**±.984	-04.30±.319
GN	59.47±.134	**-02.20**±.026
LN	60.46±3.76	-02.46±.035
CN	60.18±.089	-04.50±2.65
ℓ_∞	48.97±1.07	-07.53±0.63
ℓ_1	56.00±1.09	-08.41±0.08
GNOR	**61.76**±2.03	**-03.99**±1.09

Comparison with Existing Work (Normalization Techniques). In Tab. 4, we show the results in the single-pass regime using the CIFAR100 data set. We use different normalization techniques (with/without learning parameters) at the penultimate layer in a setup identical as in the experiments above. We use reservoir batch size of 20. In GN, CN (we used implementation proposed by the authors) and gnor, we use 32 groups for fair comparison. In BN, GN, LN and CN we found that if we use ReLu activation before normalization the average retained accuracy over BN, GN, LN and CN drops by 3%, therefore, we do not apply non-linearity, whereas in gnor before normalization, we applied sT.

As we can see, despite the fact that BN, GN, LN and CN learn normalization parameters, gnor has slightly better retained accuracy. This results suggests that it might not be necessary to use normalization parameters, mitigating the challenges of the "usual" learning strategy (coping with outdated moments and sensitivity of the normalization parameters to forgetting). gnor outperforms ℓ_1 and ℓ_∞ normalization. We highlight that with ℓ_∞, we have strong drop of performance, but there we also do not have group-wise competition, since after the normalization with ℓ_∞ within each group of neurons, the max magnitude response is 1.

Table 5: Task-agnostic regime results, we show retained accuracy and backward transfer.

	MNIST ROTATION		MNIST PERMUTATION	
METHOD	RA	BTI	RA	BTI
ONLINE	53.38±1.53	-05.44±1.70	55.42±0.65	-13.76±1.19
ER	76.69±0.35	-08.58±0.4	72.93±0.16	-09.63±0.59
AGEM	61.02±0.15	-24.30±0.5	59.81±0.12	-21.13±0.12
MER	**84.13**±0.15	**-01.79**±0.13	**79.92**±0.02	**-03.40**±0.02
LA-MAML	76.65±0.11	-09.32±0.17	74.43±0.76	-07.41±0.75
LWTA	79.61±0.89	-05.99±1.0	73.73±0.65	-07.37±0.97
WO ER	40.92±2.24	-45.93±2.58	54.02±2.59	-28.46±2.25
GNOR_{50}	**81.56**±0.53	**-06.46**±0.58	**80.01**±0.03	**-04.78**±0.17

Comparison with Existing Work (Task-agnostic Regime). We used the MNIST-rotations and MNIST-permutations data sets in the task-agnostic continual learning regime. As a model, we used 2 layer feed-forward neural network (Javed & White, 2019), but applied group-wise normalization at the penultimate layer with configuration $N_b \times J = 10 \times 10$. The reservoir size is 1000 and we set batch size to 50. In Tab. 5, we present the results. In this regime, our approach has relatively good performance using replay buffer with small size, but we found out that our algorithm achieves better performance using medium buffer size.

Table 6: Task-aware regime, comparative results when using the CIFAR10 data set.

	MULTI		SINGLE	
METHOD	RA	BTI	RA	BTI
ER	75.07±6.02	-2.24±1.26	59.30±6.57	-2.11±0.61
MER	-	-	**63.87**±1.76	-2.15±0.04
AGEM	73.32±2.33	-5.32±0.38	58.84±0.91	-3.64±1.5
LA-MAML	**77.98**±3.31	**-1.54**±0.12	58.33±2.5	**-0.41**±0.55
LWTA	74.32±2.63	-5.01±1.14	58.62±3.3	-3.45±0.02
WO ER	79.63±2.60	-2.54±0.08	58.09±3.68	-2.05±5.06
GNOR_{10}	79.60±2.96	-2.66±0.11	64.23±2.85	-1.07±0.19
GNOR_{25}	**80.12**±3.83	**-2.04**±0.66	65.55±1.68	-1.22±0.27

Table 7: Task-aware regime (multi and single pass), comparative results. We show retained accuracy (RA) and backward transferability (BTI) using the CIFAR100 and TinyImageNet data sets.

	CIFAR100				TINYIMAGENET			
	MULTI		SINGLE		MULTI		SINGLE	
METHOD	RA	BTI	RA	BTI	RA	BTI	RA	BTI
IID	85.60±0.40	-	-	-	77.10±1.06	-	-	
ER	59.70±0.75	-16.50±1.05	47.88±0.73	-12.46±0.83	48.23±1.51	-19.86±0.70	39.38±0.38	-14.33±0.89
ICARL	60.47±1.09	-15.10±1.04	53.55±1.69	**-08.03**±1.16	54.77±0.32	**-03.93**±0.55	45.79±1.49	**-02.73**±0.45
GEM	62.80±0.55	-17.00±0.26	48.27±1.10	-13.70±0.70	50.57±0.61	-20.50±0.10	40.56±0.79	-13.53±0.65
AGEM	58.37±0.13	-17.03±0.72	46.93±0.31	-13.40±1.44	46.38±1.34	-19.96±0.61	38.96±0.47	-13.66±1.73
MER	-	-	51.38±1.05	-12.83±1.44	-	-	44.87±1.43	-12.53±0.58
META-BGD	65.09±0.77	-14.83±0.40	57.44±0.95	-10.60±0.45	-	-	50.64±1.98	-06.60±1.73
LA-MAML	**70.08**±0.66	**-09.36**±0.47	**61.18**±1.44	-09.00±0.20	**66.99**±1.65	-09.13±0.90	**52.59**±1.35	-03.70±1.22
LWTA	52.42±2.56	-24.93±1.11	49.72±0.43	-11.18±0.2	36.71±1.13	-23.85±0.37	47.84±0.24	-9.49±0.91
WO ER	64.55±0.18	-14.91±0.93	55.02±0.91	-05.67±0.36	36.47±5.85	-14.35±2.48	52.17±2.29	-06.61±5.40
GNOR_{10}	71.10±0.28	-09.38±0.49	60.24±0.54	-04.69±0.17	62.51±1.08	-06.65±0.32	58.70±0.81	**-02.71**±0.81
GNOR_{25}	**72.09**±0.85	-08.84±0.13	**61.48**±3.15	**-04.47**±0.65	65.51±0.81	**-05.76**±0.17	58.40±1.45	-03.82±1.25
GNOR_{50}	71.91±3.57	**-08.39**±2.03	60.21±3.76	-05.81±1.67	**66.00**±0.80	-05.97±0.03	**58.96**±0.09	-03.51±0.29

On the MNIST rotations, the algorithm has the second best RA result, while on the MNIST permutation, our approach is on-pair with a very slight gain in RA. We highlight that in comparing algorithms like la-maml, an effective look-ahead mechanism learns the learning rate, adding $\times 2$ increase in memory. In mer was used meta experience replay with extensive per sample meta updates-based algorithm that helps avoid forgetting but increases the learning time. However, comparing our approach to mer, it needs significantly less learning time.

Comparison with Existing Work (Task-aware Regime). We used the CIFAR10, CIFAR100, and TinyImageNet data sets in the single-pass and multi-pass learning regimes. As neural network models, we used CNNs, equivalent to the ones used in Gupta et al. (2020). We apply gnor at the penultimate layer and use configurations 10×32, 10×32 and 10×64, wherein $N_b \times J$, J is the number of local blocks and N_b its dimension. We set the reservoir size to $200, 200$ and 400 for CIFAR10, CIFAR100, and TinyImageNet, respectively, and experimented with a batch size of $10, 25$ and 50.

In Tab. 6, we show comparative results in the single- and multi-pass regime using the CIFAR10 data set (we use equivalent CNN that was used (Gupta et al., 2020) for the CIFAR100, with the difference in the activation at the penultimate layer). All comparing algorithms pass through the data only once (number of glances equals 1) in the single-pass regime. In gnor_Z, Z denotes the size of the buffer that uses the replay memory. As we can see from the results, the proposed algorithm outperforms the comparing ones by 2.2% (multi-pass) and 2.58% (single-pass) while been computationally efficient. It is interesting to note that the version with only gnor (no experience replay) is on-pair with the proposed algorithm in the multi-pass regime. In Tab. 7, we show comparative results in the single- and multi-pass regime using the CIFAR100 and TinyImageNet data sets. In the single-pass regime, in our algorithm, we set the number of glances to 1, while we leave the comparing algorithms in their respective configurations (Gupta et al., 2020). As we can see from the results, gnor on both setups has high performance. On the CIFAR100 data set, we outperform the comparing ones in the multi-pass regime, while we are on par in the single-pass regime. On the TinyImageNet data set, we outperform the comparing ones by a large margin in the single-pass regime, while we are on par in the multi-pass regime. We note that the variant with only group-wise normalization has solid performance, outperforming the majority of the algorithms, demonstrating the gnor effectiveness in avoiding possible forgetting.

6 CONCLUSION

In this paper, we proposed normalization over groups of distinct neurons at the penultimate layer of the neural network together with a straightforward replay-based algorithm. We showed that such normalization induces competition between the subset of weights that connect to the penultimate layer enabling the model to reduce the chances of damaging interference and mitigating forgetting. Our empirical results demonstrated that group-wise normalization drastically impacts performance, providing gains compared to the existing methods and having low computational complexity.

REFERENCES

Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. *CoRR*, abs/2004.00070, 2020. URL https://arxiv.org/abs/2004.00070.

Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018. URL http://arxiv.org/abs/1803.08375.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154, 2018.

Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. Defining benchmarks for continual few-shot learning. *arXiv preprint arXiv:2004.11967*, 2020.

Ali Ayub and Alan Wagner. {EEC}: Learning to encode and regenerate images for continual learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=lWaz5a9lcFU.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Mehdi Abbana Bennani, Thang Doan, and Masashi Sugiyama. Generalisation guarantees for continual learning with orthogonal gradient descent. *arXiv preprint arXiv:2006.11942*, 2020.

L. Bottou, Frank E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *ArXiv*, abs/1606.04838, 2018.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *ArXiv*, abs/1812.00420, 2019.

Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. *arXiv preprint arXiv:1906.02425*, 2019.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3762–3773. PMLR, 2020.

Chelsea Finn, P. Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

R. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3: 128–135, 1999.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6211.

S. Grossberg. How does a brain build a cognitive code. *Psychological Review*, 87:1–51, 1980.

G. Gupta, Karmesh Yadav, and L. Paull. Look-ahead meta learning for continual learning. In *NeurIPS*, 2020.

James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. *arXiv preprint arXiv:1912.08866*, 2019.

Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A Rusu, Yee Whye Teh, and Razvan Pascanu. Task agnostic continual learning via meta learning. *arXiv preprint arXiv:1906.05201*, 2019.

Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *CoRR*, cs.LG/0408058, 2004. URL http://arxiv.org/abs/cs.LG/0408058.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 448–456. PMLR, 07–09 Jul 2015.

Khurram Javed and Martha White. Meta-learning representations for continual learning. In *NeurIPS*, 2019.

Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Adaptive group sparse regularization for continual learning. *CoRR*, abs/2003.13726, 2020. URL https://arxiv.org/abs/2003.13726.

Prakhar Kaushik, Adam Kortylewski, Alex Gain, and Alan Yuille. Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. URL https://www.pnas.org/content/114/13/3521.

Soheil Kolouri, Nicholas Ketz, Xinyun Zou, Jeffrey Krichmar, and Praveen Pilly. Attention-based structural-plasticity. *arXiv preprint arXiv:1903.06070*, 2019.

A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *undefined*, 2009.

Ya Le and X. Yang. Tiny imagenet visual recognition challenge. 2015.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NIPS*, 2017.

James L. McClelland, B. McNaughton, and R. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102 3:419–457, 1995.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, and Hassan Ghasemzadeh. Dropout as an implicit gating mechanism for continual learning. *CoRR*, abs/2004.11545, 2020. URL https://arxiv.org/abs/2004.11545.

J. Murre. Learning and categorization in modular neural networks. 1992.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. URL http://arxiv.org/abs/1803.02999.

Quang Pham, Chenghao Liu, and Steven HOI. Continual normalization: Rethinking batch normalization for online continual learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vwLLQ-HwqhZ.

Saiprasad Ravishankar and Yoram Bresler. Sparsifying transform learning with efficient optimal updates and convergence guarantees. *IEEE Transactions on Signal Processing*, 63(9):2389–2404, may 2015.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

M. Riemer, Ignacio Cases, R. Ajemian, Miao Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *ArXiv*, abs/1810.11910, 2019.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. *arXiv preprint arXiv:1805.07810*, 2018.

H. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 2007.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7: 123–146, 1995.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. Publisher: Nature Publishing Group.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *CoRR*, abs/1801.01423, 2018. URL http://arxiv.org/abs/1801.01423.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/file/0efbe98067c6c73dba1250d2beaa81f9-Paper.pdf.

Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Spacenet: Make free space for continual learning. *CoRR*, abs/2007.07617, 2020. URL https://arxiv.org/abs/2007.07617.

Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. *arXiv preprint arXiv:1901.11356*, 2019.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. URL http://arxiv.org/abs/1607.08022.

Shixian Wen, Amanda Rios, Yunhao Ge, and Laurent Itti. Beneficial perturbation network for designing general adaptive artificial intelligence systems. *CoRR*, abs/2009.13954, 2020. URL https://arxiv.org/abs/2009.13954.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995. PMLR, 2017.

Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes, 2019.

A APPENDIX

In the following, we provide the Appendix as part of the supplementary material to the main paper.

In Appendix Section A., we prove the alternative form for the back-propagation to make more obvious at which part the gradient of the activation with respect to it arguments comes into use. **In Appendix Section B.**, we give additional comment about applying group-wise neuron normalization with nonlinear activation, providing additional results. **In Appendix Section C.**, we prove additional result regarding performance comparison of existing normalization techniques and our gnor. **In Appendix Section D.**, we give very simple pythorch code implementation of the proposed group-wise neuron normalization that induces group-wise neuron competition (gnor).

A. BACK-PROPAGATION ALTERNATIVE FORM

In the main text of the paper, we analyzed the influence of gnor that has during back-propagation, so we were interested in the gradient of the activation with respect to its arguments, *i.e.*, $\mathbf{g}^{L-1} = \frac{\partial \mathbf{a}^{L-1}}{\partial \mathbf{z}^{L-1}}$. In this appendix section, we give the full derivation of the equivalent form for the back-propagation, and pinpoint where $\frac{\partial \mathbf{a}^{L-1}}{\partial \mathbf{z}^{L-1}}$ comes to play.

Consider a neural network that maps inputs $\mathbf{x} \in \mathbb{R}^{N_0}$ to outputs $\mathbf{y} \in \mathbb{R}^{N_L}$. The forward pass reads as:

$$\begin{aligned} \mathbf{a}^0 &= \mathbf{x}, \\ \mathbf{z}^l &= \mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l, \qquad l \in \{1, \cdots, L\}, \\ \mathbf{a}^l &= \sigma(\mathbf{z}^l), \qquad l \in \{1, \cdots, L\}. \end{aligned} \tag{4}$$

Here, $\mathbf{W}^l \in \mathbb{R}^{N_l \times N_{l-1}}$ is the weight matrix of layer l (note that if we have convolution, we can express it similarly as a matrix product multiplication), $\mathbf{b}^l$ the corresponding bias vector. Where N_l denotes the dimension of the layer l. The activation function is denoted by σ, it processes the layer's weighted sum $\mathbf{n}^l$ to the layer's output $\mathbf{a}^l$.

Given data set, we define the training process of a neural network as an optimization over an objective function, usually called *loss*, that we denote as $\mathcal{L}$. In other word, we want to find the parameters (weights and biases) that minimize the cost function. The training is commonly done using a gradient based rule. Therefore, the update relies on the gradient of $\mathcal{L}$ with respect to weight $\mathbf{W}^l$ and the bias $\mathbf{b}^l$, that is it relies on $\nabla_{\mathbf{W}}\mathcal{L}$ and $\nabla_{\mathbf{b}}\mathcal{L}$, respectively. Back-propagation facilitates the computation of these gradients, and makes use of the chain rule to back-propagate the prediction error through the network (Rumelhart et al., 1986). We express the error vector at layer l as:

$$\delta^l = \nabla_{\mathbf{n}^l}\mathcal{L}, \tag{5}$$

and further use it to express the gradients as

$$\begin{aligned} \nabla_{\mathbf{W}^l}\mathcal{L} &= \delta^l (\mathbf{a}^{l-1})^{\mathrm{T}}, \\ \nabla_{\mathbf{b}^l}\mathcal{L} &= \delta^l. \end{aligned} \tag{6}$$

The output layer's error is simply given by

$$\delta^L = \nabla_{\mathbf{a}^L}\mathcal{L} \odot \frac{\partial \mathbf{a}^L}{\partial \mathbf{z}^L}, \tag{7}$$

where $\odot$ denotes the Hadamard element-wise product. Subsequent earlier layer's error are computed with

$$\delta^l = (\mathbf{W}^{l+1})^{\mathrm{T}} \delta^{l+1} \odot \frac{\partial \mathbf{a}^l}{\partial \mathbf{z}^l}, \qquad l \in \{1, \cdots, L-1\}. \tag{8}$$

where A usual parameter update takes on the form

$$(\mathbf{W}^l)_{\text{new}} = \mathbf{W}^l - \beta \nabla_{\mathbf{W}^l}\mathcal{L}, \tag{9}$$

where β is a positive learning rate.

B. ADDITIONAL COMMENT ON GROUP-WISE NEURON COMPETITION

B.1. RELEVANCE OF THE ORDER FOR THE OPERATIONS SPARSE CODING AND GNOR

Table 8: Results using the CIFAR100 data set in the single pass continual learning regime. At the penultimate layer we used two different operating orders (i) gnor followed by sT and (ii) sT followed by gnor, showing that the difference between the two is negligible.

sT ACTIVATION FOLLOWED BY GNOR		GNOR FOLLOWED BY sT ACTIVATION	
RA	BTI	RA	BTI
61.47±1.07	-04.34±-1.34	61.30±0.56	**-04.07**±-0.94

Existing Methods (Normalization Followed by Nonlinear Activation). In the existing normalization techniques like batch norm (Ioffe & Szegedy, 2015), group norm (Wu & He, 2018) and layer norm (Ba et al., 2016), it is common that we first apply the normalization and then the nonlinear activation, *i.e.*, $\mathbf{a}^l = ReLu(bn(\mathbf{z}^l))$, where $bn(\mathbf{z}^l)$ is the batch norm in this example. It turn out that when we use the existing normalization techniques (Ioffe & Szegedy, 2015; Wu & He, 2018; Ba et al., 2016) it makes a big difference whether (i) we first apply the nonlinear activation and then the apply the normalization, or (ii) we first apply the normalization and then apply the nonlinear activation. In the later case, using normalization with learnable parameters (Ioffe & Szegedy, 2015; Wu & He, 2018; Ba et al., 2016), we might have degradation of performance. We mention this in the paper as well as give additional results in Section C.

gnor. In our approach we found out that it dose not play a role if we first apply the nonlinear activation (at least in the case of sT) and then the gnor, or the other way around. In the following we will revisit the definition about gnor and give explanation.

In the main text, we construct $\mathbf{a}^l$ as $\mathbf{a}^l = [(\mathbf{a}_1^l)^T, (\mathbf{a}_2^l)^T, ..., (\mathbf{a}_J^l)^T]^T$ and define the group-wise normalization as:

$$\mathbf{a}_j^l = v_{gnor}(\mathbf{z}_j^l) = \frac{\mathbf{z}_j^l}{\|\mathbf{z}_j^l\|}, \forall j \in \{1, .., J\}, \tag{10}$$

where $\mathbf{z}_j^l$ are linear activation's that belong to group j and $\|\mathbf{z}_j^l\|$ is the ℓ_2-norm of $\mathbf{z}_j^l$. We also mentioned that before the normalization, we can apply nonlinear activation, such as ReLu Agarap (2018), so we have $\mathbf{a}_j^l = v_{gnor}(\text{ReLu}(\mathbf{z}_j^l))$, or with sparse coding transform sT (Ravishankar & Bresler, 2015) we have:

$$\mathbf{a}_j^l = v_{gnor}(\text{sT}(\mathbf{z}_j^l)). \tag{11}$$

Note that the group-wise neuron competition is about the max magnitude neuron response that should lay in the group with the highest sparsity. Therefore, when we apply the sT, we pronounce the sparsity (ether before or after) within the group of neuron responses, and this should help the group-wise neuron competition, since we remove (threshold out) some not-relevant "information".

In Tab. 8, we give results that evaluated the above line of tough. We experimented using the CIFAR100 data set in the task-aware single pass regime. We set the reservoir size of 200 and use a reservoir batch size of 20. As we can see in Tab. 8, the results suggest that the order of these operation dose not play a role. When we use the sT after the gnor, very slightly we have higher retained accuracy, while when we use the sT before the gnor, very slightly we have better backward transfer. So, the difference in results between the two are negligible.

B.2. SOFT SPARSE CODING TRANSFORM VS HARD SPARSE CODING TRANSFORM

Table 9: Results using the CIFAR100 data set in the single pass continual learning regime. At the penultimate layer we used two different sparse coding transforms (sT) (i) "hard" sT followed by gnor and (ii) "soft" sT followed by gnor.

HARD sT ACTIVATION		SOFT sT ACTIVATION	
RA	BTI	RA	BTI
61.47$_{\pm 1.07}$	-04.34$_{\pm 1.34}$	60.86$_{\pm 0.37}$	-**04.29**$_{\pm 0.18}$

Hard Sparse Coding Transform. We define the "hard" sparse coding transform (sT) (Ravishankar & Bresler, 2015), as,

$$\text{sT}(\mathbf{z}_j^l) = \mathbf{z}_j^l \odot \mathbf{t}_j^l, \tag{12}$$

where $t_{j,i}^l = 1$ if $|z_{j,i}^l| \geq \lambda_j$, and $t_{j,i}^l = 0$ otherwise, while $\lambda_j = \frac{J}{N_l} \sum_{i=1}^{N_l/J} |z_{j,i}^l|$ is a sparsifying threshold. We have $\mathbf{z}_j^l$ as the linear activation's that belong to group $j \in \{1, 2, ..., J\}$, $\mathbf{z}^l = [(\mathbf{z}_1^l)^T, (\mathbf{z}_2^l)^T, ..., (\mathbf{z}_J^l)^T]^T \in \Re^{N_l}$.

Soft Sparse Coding Transform. The "soft" sparse coding transform (sT) is similar to the "hard" sparse coding transform, *i.e.*:

$$\text{sT}_{soft}(\mathbf{z}_j^l) = sign(\mathbf{z}_j^l) \odot \max(|\mathbf{z}_j^l| - \lambda_j, 0), \tag{13}$$

where we define the sparsifying threshold as above. The difference between equation 13 and equation 12 is that in equation 13 in addition to the tresholding to zero the magnitude of the nonzero values are also reduced by the sparsifying threshold λ_j.

Although equation 13 might have better bias/variance properties in classical signal processing applications, we empirically found out that equation 12 works better in our continual learning regimes. In Tab. 9, we verify this and give comparative results between "hard" and "soft" sT. We experimented using the CIFAR100 data set in the task-aware single pass regime. We set the reservoir size of 200 and use a reservoir batch size of 20. We use sT ("hard" or "soft") followed by gnor. As we can see in Tab. 9, the "hard" sT has an edge over "soft" sT, with higher retained accuracy of .6%, while the backward trasferability score is very similar between the "hard" sT and the "soft" sT.

C. Additional Comparative Results with Normalization Methods

Table 10: Results using the CIFAR100 data set in the single pass continual learning regime. We show the performance when we use BN (Ioffe & Szegedy, 2015), GN (Wu & He, 2018), LN (Ba et al., 2016) and CN (Pham et al., 2022) at the penultimate layer. We experimented with ReLu before and after the normalization.

	Linear Activation Followed by Normalization		ReLu Activation Followed by Normalization		Normalization Followed by ReLu	
NORM	RA	BTI	RA	BTI	RA	BTI
BN	**61.01**±.984	-04.30±.319	56.90±2.05	-06.80±0.52	**60.65**±0.92	-06.93±0.54
GN	59.47±.134	**-02.20**±.026	56.49±0.50	-04.50±.020	58.73±0.67	**-04.66**±2.89
LN	60.46±3.76	-02.46±.035	**58.55**±0.58	-05.53±.013	58.14±1.72	-04.93±2.08
CN	60.18±.089	-04.50±2.65	58.26±0.68	**-04.50**±0.20	58.88±1.65	-07.75±3.08

Table 11: Results using the CIFAR100 data set in the single pass continual learning regime. We show the performance when we different activation functions.

	Activation Followed by GNOR		Activation Without GNOR	
f_n	RA	BTI	RA	BTI
LIN	60.14±1.80	-06.22±0.37	**55.89**±0.38	**-08.09**±1.06
ReLu	54.61±0.98	-10.04±0.00	53.50±0.05	-09.81±0.37
sT$_{soft}$	60.86±0.37	-04.29±-0.18	53.63±0.75	-09.25±-0.61
sT	**61.76**±2.03	**-03.99**±1.09	54.74±0.80	-09.18±0.03

In Tab. 10, we show the results in the single-pass regime using the CIFAR100 data set. We compare against existing normalization techniques, including including BN (Ioffe & Szegedy, 2015), GN (Wu & He, 2018), LN (Ba et al., 2016) and CN (Pham et al., 2022). We used equivalent convolution neural network as in (Gupta et al., 2020), and we apply the normalization at the penultimate layer. We use reservoir of size 200 and reservoir batch size of 20. In the GN, CN and gnor, we use 32 groups with 10 neuron activation's each for fair comparison.

As we can see in Tab. 10 on the left side on the top, when we do not apply non-linearity, BN, GN, LN and CN have good performance in both RA and BTI. While when we used ReLu activation followed by BN, GN, LN or CN we have lower retained accuracy (RA) and backward transferability (BTI). That is the retained accuracy over BN, GN, LN and CN drops by an average of around 3.5%. In the case when we preform the operations in the reversed order, that is BN, GN, LN or CN followed by ReLu the results have a bit higher retained accuracy compared to the previous ones. However, we still have lower retained accuracy compared to the linear activation by an average of around 2%. We also observed that when we used a sT activation followed by BN, GN, LN or CN, we had a strong degradation of performance (during training, the objective value got stuck at a certain value and was not able to improve/learn for the experimented normalization methods). We also tried out instance normalization (IN) (Ulyanov et al., 2016) in a equivalent setup as before, but similarly as above, we were not able to make the model to learn.

In Tab. 12 on the bottom, we show the influence of the used nonlinear activation function before the gnor. As activation function we experimented with linear, ReLu, and sT. As reported in the main text, we can see that not all combinations of activation function followed by gnor are useful.

In Tab. 10 and Tab. 12, we can also see that our approach with "hard" sT and gnor has the best performance.

Influence/Impact of Group Size, Nonlinear Activation, and grouping Type. To see the influence of the local group size for the gnor, we show the results with and without gnor in Tab. 12. We see that we achieve better performance when using groups with smaller size under gnor. We conclude that the used neural network with group size 10 has the highest performance. We contribute this to the weighting in error propagation induced by the gnor that is more localized. This appears to help avoid forgetting. In Tab. 12, we show the influence of the used nonlinear activation function before the

Table 12: Results using the CIFAR100 data set in the single-pass learning regime. **Top Left.** The impact on the that local block size N_b in the retained accuracy (RA) and the backward transferability (BTI). **Top Right.** The impact of the used grouping that results in non-overlapping blocks. **Bottom Right.** The impact of the used non-linearity. **Bottom Left.** Comparison between gnor and LWTA, using normalization 1. at the penultimate layer and 2. at at all feed-forward layers.

	W NORM		WO NORM	
$N_b \times J$	RA	BTI	RA	BTI
320×01	44.33±0.33	-09.54±2.83	54.87±0.97	-9.08±0.52
160×02	48.95±0.12	-10.09±1.48	54.61±0.50	-9.02±0.59
080×04	53.25±0.17	-09.12±0.28	54.76±0.36	-8.94±0.42
040×08	57.13±1.85	-07.58±0.12	54.87±2.69	**-8.48**±1.57
010×32	**60.21**±3.76	-05.81±1.67	54.74±0.80	-9.18±0.03
005×64	60.18±1.14	**-05.05**±0.03	**54.87**±0.52	-8.93±0.83

GR.	LINEAR		NONLINEAR	
TYPE	RA	BTI	RA	BTI
REG	60.14±1.80	-6.22±0.37	60.21±3.76	-5.81±1.67
SORT	60.38±1.27	-5.98±0.26	**60.49**±4.53	-5.54±1.71
RAN_s	60.38±3.05	-5.41±0.30	37.80±7.30	-4.21±1.17
RAN_{all}	**61.21**±1.20	**-5.15**±0.11	36.24±0.57	**-4.19**±0.43

	W NORM		WO NORM	
f_n	RA	BTI	RA	BTI
LIN	60.14±1.80	-06.22±0.37	**55.89**±0.38	**-08.09**±1.06
RELU	54.61±0.98	-10.04±0.00	53.50±0.05	-09.81±0.37
LWTA	32.70±1.30	-06.48±0.34	50.20±2.17	-10.46±0.16
sT	**60.21**±3.76	**-05.81**±1.67	54.74±0.80	-09.18±0.03

	W NORM		WO NORM	
	RA	BTI	RA	BTI
LWTA_l	32.70±1.30	-6.48±0.34	50.20±2.17	-10.46±0.16
GNOR_l	60.21±3.76	-5.81±1.67	**54.74**±0.80	-09.18±0.03
LWTA_{ll}	29.90±0.21	-4.41±0.11	48.18±0.38	-09.85±0.44
GNOR_{ll}	**61.43**±0.73	-5.26±0.00	54.33±1.70	**-08.77**±1.04

gnor. As activation function we experimented with linear, relu, LWTA and equation 10. The results show that not all combinations of activation function followed by gnor are useful. Interestingly, we can achieve good performance even without using any nonlinear activation function, signifying the role and importance of gnor in combating forgetting for continual learning. The performance drops when using the LWTA nonlinear activation function with gnor. Since LWTA with gnor neglects the magnitude of the activation within the local block, putting its value either -1 or 1, which represents a loss of important information related to the competitive nature. In Tab. 12, we show comparative results between convolutions neural networks that have sT and gnor on all of the feed-forward layers and only on the penultimate layer. As comparing nonlinear activation functions, we use LWTA and equation 10. As we can see from the results, applying the combination of nonlinear activation function followed by gnor on all feed-forward layers does not lead to big improvement. The identical network with the same nonlinear activation function and gnor only at the penultimate layer retains comparable accuracy. This result supports our claim that it is sufficient to have gnor only at the penultimate layer. In Tab. 12, we show the impact of the grouping type used to form the local groups. We apply fixed, regular grouping on non-overlapping groups of equal sizes. In addition, we compare the grouping into distinct neurons done on the neural activation at the penultimate layer sorted by their magnitude. We also compare with the grouping done on the neural activation at the penultimate layer but randomly permuted from their natural order. We apply either one random permutation at each update to all responses from the batch of data or as many random permutations as the number of data in the batch (one random permutation for each data response). Interestingly, we can see that the type of grouping does not play much of a role in achieving high performance. The group-wise normalization with random grouping achieves the highest RA.

Impact of the Replay Memory Batch Size. In Tab. 13, we give results about the impact of the used batch size for the replay memory. Chaining the balance between the available data from the current task and memory does not hurt performance. On the contrary, we can see that the performance of the algorithm slightly improves. The method gives results with slight difference and peaks performance in retained accuracy when using a batch size of 20. On the other hand, when we do not use gnor, as we increase the batch size, we have an increase in performance, but the results have lower accuracy compared to using the sT activation function followed by gnor. We assume that this is due to the used group-wise normalization. The update

Table 13: Ablation results, impact of the replay batch size.

BAT.	W NORM		WO NORM	
SIZE	RA	BTI	RA	BTI
10	60.24±0.54	-4.69±0.17	50.18±6.45	-11.71±6.83
15	61.23±2.14	-4.57±1.03	51.13±1.38	-10.42±0.77
20	**61.76**±2.03	**-3.99**±1.09	51.87±2.73	-10.25±2.41
25	61.48±3.15	-4.47±0.65	53.00±0.60	-09.18±0.08
30	61.43±0.77	-4.33±0.04	52.14±0.36	-10.46±1.19
35	59.72±0.68	-6.13±0.83	53.09±1.02	-10.01±0.12
40	60.61±1.27	-5.48±0.41	54.11±1.81	**-08.70**±1.43
45	59.91±0.37	-6.09±0.39	54.53±2.98	-08.95±1.04
50	60.21±3.76	-5.81±1.67	**54.74**±0.80	-09.18±0.03

of the network parameters is localized so that the strong update component that might lead to strong parameters overwrite is balanced out.

D. GROUP-WISE NEURON COMPETITION IMPLEMENTATION

D.1. GROUP-WISE NEURON COMPETITION (GNOR) PYTHORCH IMPLEMENTATION

In the following we prove a **pythorch** implementation of the group-wise normalization (gnor) that induces group-wise neuron competition between subsets of weighs.

```
def gnor(a, group_size, num_of_groups):
    # gnor implementation
    # assuming shape_1 = group_size*num_of_groups
    shape_0, shape_1 = a.shape
    a = a.reshape(shape_0, group_size, num_of_groups)
    a = torch.nn.functional.normalize(a, p=2, dim=1).reshape(shape_0, shape_1)

    return  a
```

We note that before applying the gnor, we also apply "hard" sparse coding transform (sT) (for more details please see Appendix Section B.2.).

D.2. HARD SPARSE CODING TRANSFORM (ST) PYTHORCH IMPLEMENTATION

In the following, we give the **pythorch** implementation for the "hard" sparse coding transform.

```
def st(z, group_size, num_of_groups):
    # sT implementation
    shape_0, shape_1 = z.shape
    # compute the threshold per group of neuron activation
    r = z.reshape(shape_0, group_size, num_of_groups ).detach().clone()
    tr = torch.abs( r ) >=  torch.mean( torch.abs(r), 1).unsqueeze(1)
    # set to zero the components below the threshold
    a = z * tr.float().reshape(shape_0, shape_1)

    return a
```