Parameter-Level Soft-Masking for Continual Learning

Tatsuya Konishi [1] [†] **Mori Kurokawa** [1] **Chihiro Ono** [1] **Zixuan Ke** [2] **Gyuhak Kim** [2] **Bing Liu** [2]

Abstract

Existing research on *task incremental learning* in continual learning has primarily focused on preventing *catastrophic forgetting* (CF). Although several techniques have achieved learning with no CF, they attain it by letting each task monopolize a sub-network in a shared network, which seriously limits knowledge transfer (KT) and causes over-consumption of the network capacity, i.e., as more tasks are learned, the performance deteriorates. The goal of this paper is threefold: (1) overcoming CF, (2) encouraging KT, and (3) tackling the capacity problem. A novel technique (called SPG) is proposed that *soft-masks* (partially blocks) parameter updating in training based on the importance of each parameter to old tasks. Each task still uses the full network, i.e., no monopoly of any part of the network by any task, which enables maximum KT and reduction in capacity usage. To our knowledge, this is the first work that soft-masks a model at the parameter-level for continual learning. Extensive experiments demonstrate the effectiveness of SPG in achieving all three objectives. More notably, it attains significant transfer of knowledge not only among similar tasks (with shared knowledge) but also among dissimilar tasks (with little shared knowledge) while mitigating CF.

1. Introduction

Catastrophic forgetting (CF) and knowledge transfer (KT) are two key challenges of continual learning (CL), which learns a sequence of tasks incrementally. CF refers to the phenomenon where a model loses some of its performance on previous tasks once it learns a new task. KT means that tasks may help each other to learn by sharing knowledge.

This work further investigates these problems in the popular CL paradigm, *task-incremental learning* (TIL). In TIL, each task consists of several classes of objects to be learned. Once a task is learned, its data is discarded and will not be available for later use. During testing, the task id is provided for each test sample so that the corresponding classification head of the task can be used for prediction.

Several effective approaches have been proposed for TIL that can achieve learning with little or no CF. *Parameter isolation* is perhaps the most successful one in which the system learns to mask a sub-network for each task in a shared network. HAT (Serra et al., 2018) and SupSup (Wortsman et al., 2020) are two representative systems. HAT set binary/hard masks on neurons (not parameters) that are important for each task. In learning a new task, those masks block the gradient flow through the masked neurons in the backward pass. Only those *free* (unmasked) *neurons* and their parameters are trainable. Thus, as more tasks are learned, the number of free neurons left becomes fewer, making later tasks harder to learn, which results in gradual performance deterioration (see Section 4.2.1). Further, if a neuron is masked, all the parameters feeding to it are also masked, which consumes **a great deal of network capacity** (hereafter referred to as the "capacity problem"). As the sub-networks for old tasks cannot be updated, it has **limited knowledge transfer**. CAT (Ke et al., 2020) tries to improve KT of HAT by detecting task similarities. If the new task is found similar to some previous tasks, these tasks' masks are removed so that the new task training can update the parameters of these tasks for backward pass. However, this is risky because if a dissimilar task is detected as similar, serious CF occurs, and if similar tasks are detected as dissimilar, its knowledge transfer will be limited. SupSup uses a backbone network (randomly initialized) and finds a sub-network for each task. The sub-network is represented by a mask, which is a set of binary gates indicating which parameters in the network are used. The mask for each task is saved. Since the network is not changed, SupSup has no CF or capacity problem, but since each mask is independent of other masks, SupSup by design has no KT.

To tackle these problems, we propose a very different approach, named "Soft-masking of Parameter-level Gradient flow" (SPG). It is surprisingly effective and contributes in following ways:

[†]The work was done when this author was visiting Bing Liu's group at University of Illinois at Chicago. [1]KDDI Research, Inc., Fujimino, Japan. [2]University of Illinois at Chicago, Chicago, United States. Correspondence to: Tatsuya Konishi <tt-konishi@kddi.com>.

(1). Instead of learning hard/binary masks on neurons for each task and blocking these neurons in training a new task and in testing like HAT, SPG computes an importance score for each network parameter (not neuron) to old tasks using gradients. The reason that gradients can be used as importance is because gradients directly tell how a change to a specific parameter will affect the output classification and may cause CF. SPG uses the importance score of each parameter as a *soft-mask* to constrain the gradient flow in the backward pass to ensure those important parameters to old tasks have minimum changes in learning a new task to prevent CF of previous knowledge. To our knowledge, soft-masking of parameters has not been done before.

(2). SPG has some resemblance to the popular regularization-based approach, e.g., EWC (Kirkpatrick et al., 2017), in that both use importance of parameters to constrain changes to important parameters of old tasks. But there is a major difference. SPG directly controls each parameter (fine-grained), but EWC controls all parameters together using a regularization term in the loss to penalize the sum of changes to all parameters in the network (rather coarse-grained). Section 4.2 shows that our **soft-masking is markedly better than regularization**. We believe this is an important result.

(3). In the forward pass, no masks are applied, which encourages knowledge transfer among tasks. This is better than CAT as SPG does not need extra mechanism for task similarity comparison. Knowledge sharing and transfer in SPG are automatic. SupSup cannot do knowledge transfer.

(4). As SPG soft-masks parameters, it does not monopolize any parameters or sub-network like HAT for each task and SPG's forward pass does not use any masks. This reduces the *capacity problem*.

Experiments with the standard CL setup have been conducted with (1) similar tasks to demonstrate SPG's better knowledge transfer and (2) dissimilar tasks to show SPG's ability to overcome CF, and (3) deal with the capacity issue. None of the baselines is able to achieve all. The code is available at `https://github.com/UIC-Liu-Lab/spg`.

2. Related Work

Approaches in continual learning can be grouped into three main categories. We review them below.

Regularization-based: This approach computes importance values of either parameters or their gradients on previous tasks, and adds a regularization in the loss to restrict changes to those important parameters to mitigate CF. EWC (Kirkpatrick et al., 2017) uses the Fisher information matrix to represent the importance of parameters and a regularization to penalize the sum of changes to all parameters. SI (Zenke et al., 2017) extends EWC to re-

duce the complexity in computing the penalty. Many other approaches (Li & Hoiem, 2016; Zhang et al., 2020; Ahn et al., 2019) in this category have also been proposed, but they still have difficulty to prevent CF. As discussed in the introduction section, the proposed approach SPG has some resemblance to a regularization based method EWC. But the coarse-grained approach of using regularization is significant poorer than the fine-grained soft-masking in SPG for overcoming CF as we will see in Section 4.2.

Memory-based: This approach introduces a small memory buffer to store data of previous tasks and replay them in learning a new task to prevent CF (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019). Some methods (Shin et al., 2017; Deja et al., 2021) prepare data generators for previous tasks, and the generated pseudo-samples are used instead of real samples. Although several other approaches (Rebuffi et al., 2017; Riemer et al., 2019; Aljundi et al., 2019) have been proposed, they still suffer from CF. SPG does not save replay data or generate pseudo-replay data.

Parameter isolation-based: This approach is most similar to ours SPG. It tries to learn a sub-network for each task (tasks may share parameters and neurons), which limits knowledge transfer. We have discussed HAT, SupSup, and CAT in Section 1. Many others also take similar approaches, e.g., Progressive Networks (PGN) (Rusu et al., 2016), APD (Yoon et al., 2020), PathNet (Fernando et al., 2017), PackNet (Mallya & Lazebnik, 2018), SpaceNet (Sokar et al., 2021), and WSN (Kang et al., 2022). In particular, PGN allocates a sub-network for each task in advance, and progressively concatenates previous sub-networks while freezing parameters allocated to previous tasks. APD selectively reuses and dynamically expands the dense network. Those methods, however, depend on the expansion of network for their performance, which is often not acceptable in cases where many tasks need to be learned. PathNet splits each layer into multiple sub-modules and finds the best *pathway* designated to each task. PackNet freezes important weights for each task by finding them based on pruning. Although PathNet and PackNet do not expand the network along with continual learning, they suffer from over-consumption of the fixed capacity. To address this, SpaceNet adopts the sparse training to preserve parameters for future tasks but the performance for each task is sacrificed. WSN also allocates a subnetwork within a dense network and selectively reuses subnetworks for previous tasks without expanding the whole network. Nevertheless, those methods are still limited by the pre-allocated network size as each task monopolizes and consumes some amount of capacity, which results in poorer KT when learning many tasks.

In summary, parameter isolation-based methods suffer from over-consumption of network capacity and have limited KT, which the proposed method tries to address at the same time.

3. Proposed SPG

(a) Train task t until convergence

Forward pass

Parameter $\boldsymbol{\theta}_i$

Backward pass

Soft-masking: $\boldsymbol{g}'_i \leftarrow (1 - \boldsymbol{\gamma}_i^{\leq t-1})\boldsymbol{g}_i \quad \boldsymbol{g}'_{H_t} \leftarrow (1 - \bar{\boldsymbol{\gamma}}^{\leq t-1})\boldsymbol{g}_{H_t}$

(b) Compute importance after training task t

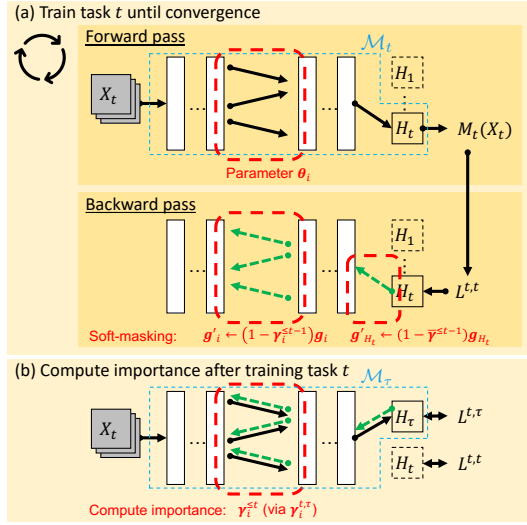Compute importance: $\boldsymbol{\gamma}_i^{\leq t}$ (via $\boldsymbol{\gamma}_i^{t,\tau}$)

Figure 1. When learning task t, SPG proceeds in two steps. Black (solid) and green (dashed) arrows represent forward and backward propagation, respectively. H_t denotes the head for task t. **(a)** Training of a model. In the forward pass, nothing extra is done. In the backward pass, the gradients of parameters in the feature extractor $\boldsymbol{g}_i$ are changed to $\boldsymbol{g}'_i$, based on the accumulated importance ($\boldsymbol{\gamma}_i^{\leq t-1}$). For parameters of the head for task t, their gradients $\boldsymbol{g}_{H_t}$ are changed to $\boldsymbol{g}'_{H_t}$ using the average of accumulated importance ($\bar{\boldsymbol{\gamma}}^{\leq t-1}$). **(b)** Computation of the accumulated importance $\boldsymbol{\gamma}_i^{\leq t}$.

As discussed in Section 1, the current parameter isolation approaches like HAT (Serra et al., 2018) and SupSup (Wortsman et al., 2020) are very effective for overcoming CF, but they hinder knowledge transfer and/or consume too much learning capacity of the network. For such a model to improve knowledge transfer, it needs to decide which parameters can be shared and updated for a new task. That is the approach taken in CAT (Ke et al., 2020). CAT finds similar tasks and removes their masks for updating, but may find wrong similar tasks, which causes CF. Further, parameters are the atomic information units, not neurons, which HAT masks. If a neuron is masked, all parameters feeding into it are masked, which costs a huge amount of learning capacity. SPG directly soft-masks parameters based on their importance to previous tasks, which is a more flexible and uses much less learning space. Soft-masking clearly enables automatic knowledge transfer.

Figure 1 and Algorithm 1 illustrate how SPG works. In SPG, the importance of a parameter to a task is computed based on its gradient. We do so because gradients of parameters directly and quantitatively reflect how much changing a parameter affects the final loss. Additionally, we normalize the gradients of the parameters within each layer to make relative importance more reliable as gradients in different layers

---

Algorithm 1 Continual Learning in SPG.

1: **for** $t = 1, \cdots, T$ **do**
2:     **# Training of task t. $\mathcal{M}_t$ is the model for task t (see Figure 1(a)).**
3:     **repeat**
4:         Compute gradients $\{\boldsymbol{g}_i\}$ and $\boldsymbol{g}_{H_t}$ with $\mathcal{M}_t$ using the task t's data $(\boldsymbol{X}_t, \boldsymbol{Y}_t)$.
5:         **for all** parameters of i-th layer **do**
6:             $\boldsymbol{g}'_i \leftarrow$ Equation (6)
7:         **end for**
8:         **for all** parameters of the task t's head **do**
9:             $\boldsymbol{g}'_{H_t} \leftarrow$ Equation (7)
10:        **end for**
11:       Update $\mathcal{M}_t$ with the modified gradients $\{\boldsymbol{g}'_i\}$ and $\boldsymbol{g}'_{H_t}$.
12:     **until** $\mathcal{M}_t$ converges.
13:     **# Computing the importance of parameters after training task t (see Figure 1(b)).**
14:     **for** $\tau = 1, \cdots, t$ **do**
15:         Compute a loss $\mathcal{L}^{t,\tau}$ in Equation (2).
16:         **for all** parameters of i-th layer **do**
17:             $\gamma_i^{t,\tau} \leftarrow$ Equation (1)
18:         **end for**
19:     **end for**
20:     **for all** parameters of i-th layer **do**
21:         $\gamma_i^t \leftarrow$ Equation (4), $\gamma_i^{\leq t} \leftarrow$ Equation (5)
22:     **end for**
23:     Store only $\{\gamma_i^{\leq t}\}$ for future tasks.
24: **end for**

---

can have different magnitude. The normalized importance scores are accumulated by which the corresponding gradients are reduced in the optimization step to avoid forgetting the knowledge learned from the previous tasks.

3.1. Computing the Importance of Parameters

This procedure corresponds to Figure 1(b). The importance of each parameter to task t is computed right after completing the training of task t following these steps. Task t's training data, $(\boldsymbol{X}_t, \boldsymbol{Y}_t)$, is given again to the trained model of task t, and the gradient of each parameter in i-th layer (i.e., each weight or bias of each layer) is then computed and used for computing the importance of the parameter. Note that we use $\boldsymbol{\theta}_i$ (a vector) to represent all parameters of the i-th layer. This process does not update the model parameters. The reason that the importance is computed after training of the current task has converged is as follows. Even after a model converges, some parameters can have larger gradients, which indicate that changing those parameters may take the model out of the (local) minimum leading to forgetting. On the contrary, if all parameters have similar gradients (i.e, balanced directions of gradients), changing
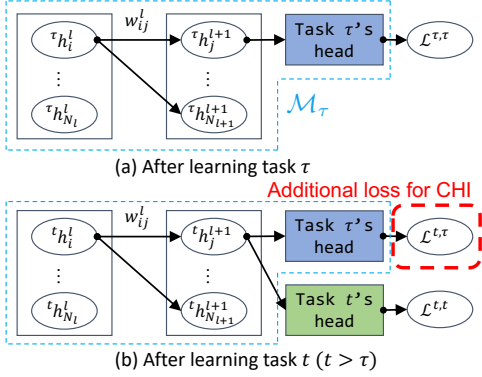
(a) After learning task τ

Additional loss for CHI

(b) After learning task t $(t > \tau)$

Figure 2. Cross-head importance (CHI). In the above figures, ${}^{t}h_i^l$ and w_{ij}^l represents the output of the i-th neuron in the l-th layer just after training task t and the parameter in l-th layer connecting between the neurons ${}^{t}h_i^l$ to ${}^{t}h_j^{l+1}$, respectively. **(a)** The importance of w_{ij}^l to task τ is computed based on its gradient, $\partial\mathcal{L}^{\tau,\tau}/\partial w_{ij}^l$ and then accumulated. **(b)** After learning task t $(t > \tau)$, the state of related parameters might have been changed. To reflect importance to task τ again with the current neurons' output (e.g., ${}^{t}h_i^l$ rather than old ${}^{\tau}h_i^l$), an additional loss, $\mathcal{L}^{t,\tau}$, is computed at task τ's head using task t's data as unlabeled data for task τ.

the parameters will not likely to change the model much to cause forgetting. Based on this assumption, we utilize the normalized gradients after training as a signal to indicate such dangerous parameter updates. The proposed mechanism in SPG has the merit that it keeps the model flexible as it does not fully block parameters using an importance threshold or binary masks. While HAT completely blocks important neurons, which results in the loss of trainable parameters over time, SPG allows most parameters remain "alive" even when most of them do not change much.

Additionally, computing the gradients based only on the model ($\mathcal{M}_t$) of the current task t does not deal with another issue. We use an example illustrated in Figure 2. For example, just after learning task τ, the gradient of a parameter is computed and normalized among the same layer's parameters to be accumulated. Even though during learning task t $(t > \tau)$, the parameter is not much changed considering its accumulated importance, at the end of learning task t, the state of related parameters might have been changed, by which the normalized importance may become less useful. To reflect the parameter's importance to task τ again in the current network state, we introduce *cross-head importance* (CHI) mechanism. In particular, an additional loss, $\text{Sum}(\mathcal{M}_\tau(\boldsymbol{X}_t))$, is computed with each previous task's head by substituting task t's data as unlabeled data for the previous tasks. By taking this loss, parameters affecting the logits more for previous tasks are regarded more important. Finally, both the normalized importance computed for the current task's head and the ones for previous tasks' heads in CHI are considered by taking element-wise maximum, as shown in Equation (4).

To put things together, the proposed method computes the normalized importance, $\boldsymbol{\gamma}_i^t$, of the parameters of the i-th layer, $\boldsymbol{\theta}_i$, using each task τ's model $(1 \leq \tau \leq t)$, $\mathcal{M}_\tau$.

$$\boldsymbol{\gamma}_i^{t,\tau} = \left|\tanh\left(\text{Norm}\left(\frac{\partial\mathcal{L}^{t,\tau}}{\partial\boldsymbol{\theta}_i}\right)\right)\right|, \tag{1}$$

$$\mathcal{L}^{t,\tau} = \begin{cases} \mathcal{L}\left(\mathcal{M}_\tau\left(\boldsymbol{X}_t\right), \boldsymbol{Y}_t\right) & (\tau = t) \\ \text{Sum}\left(\mathcal{M}_\tau\left(\boldsymbol{X}_t\right)\right) & (\tau < t) \end{cases}, \tag{2}$$

$$\text{Norm}(\boldsymbol{x}) = \frac{\boldsymbol{x} - \text{Mean}(\boldsymbol{x})}{\sqrt{\text{Var}(\boldsymbol{x})}}, \tag{3}$$

$$\boldsymbol{\gamma}_i^t = \max\left(\boldsymbol{\gamma}_i^{t,1}, \cdots, \boldsymbol{\gamma}_i^{t,t}\right), \tag{4}$$

where $\max(\cdot)$ and $\mathcal{L}$ mean element-wise maximum and a loss function, respectively. Equation (1) normalizes the gradients over the same layer to avoid the discrepancies caused by large differences of gradient magnitudes in different layers. For the current task's head (i.e., $\tau = t$), a normal loss function (e.g., cross entropy) is used as $\mathcal{L}^{t,t}$ in Equation (2). However, for each previous task's head (when $\tau < t$), since the current task data do not belong to any previous classes, the loss $\mathcal{L}^{t,\tau}$ is defined by $\text{Sum}(\mathcal{M}_\tau(\boldsymbol{X}_t))$ over previous classes' logits in the proposed CHI mechanism. Essentially, this operation computes the importance of parameters based on the data of task t's impact on all tasks learned so far. Finally, to prevent forgetting as much as possible, we take the accumulated importance, $\boldsymbol{\gamma}_i^{\leq t}$, as follows:

$$\boldsymbol{\gamma}_i^{\leq t} = \max\left(\boldsymbol{\gamma}_i^t, \boldsymbol{\gamma}_i^{\leq t-1}\right), \tag{5}$$

where an all-zero vector is used as $\boldsymbol{\gamma}_i^{\leq 0}$. This $\boldsymbol{\gamma}_i^{\leq t}$ depicts how important each parameter is to all the learned tasks.

Memory needed to save parameter importance: Regardless of the number of tasks, at any time only the accumulated importance $\boldsymbol{\gamma}_i^{\leq t}$ is saved after the learning of each task so that it can be used again in the next task for Equation (5). $\boldsymbol{\gamma}_i^{\leq t}$ has the same size as the number of parameters, $|\boldsymbol{\theta}_i|$.

3.2. Soft-Masking of Feature Extractor

This procedure appears in Figure 1(a). To suppress the update of important parameters in the backward pass in learning task t, the gradients of all parameters in the shared feature extractor are modified (i.e., *soft-masked*) based on the accumulated importance as follows (i.e., each parameter is soft-masked by a different amount according to its accumulated importance):

$$\boldsymbol{g}_i' = \left(1 - \boldsymbol{\gamma}_i^{\leq t-1}\right)\boldsymbol{g}_i, \tag{6}$$

where $\boldsymbol{g}_i$ and $\boldsymbol{g}_i'$ represent the original gradients of the parameters of the i-th layer (i.e., $\boldsymbol{\theta}_i$) and the modified ones, which will be used in the actual optimization, respectively.

3.3. Soft-Masking of Classification Head

We found that the above soft-masking may induce another problem. If only the feature extractor's parameters are soft-masked, the model will try to find an optimal solution mainly by updating the classification head since its parameters are not masked and thus can be updated more easily than the feature extractor. However, this discourages the learning of the feature extractor, which hinders knowledge transfer.

To achieve a balanced training of the feature extractor and the classification head, we need to **slow down** the learning of the head by reducing the gradients of the head's parameters based on how much the feature extractor's parameters are soft-masked. We still use the soft-masking idea, but all parameters in the head are soft-masked by an equal amount. Specifically, the gradients of all parameters in task t's head, g_{H_t}, are soft-masked by the average ($\bar{\gamma}^{\leq t-1}$) of the accumulated importance of all the parameters in the feature extractor (i.e., $\{\gamma_i^{\leq t-1}\}$). The modified gradients, g'_{H_t}, are used in optimization.

$$g'_{H_t} = \left(1 - \bar{\gamma}^{\leq t-1}\right) g_{H_t} \tag{7}$$

Note that SPG has no specific hyper-parameter and does not employ anything special in the forward pass except it needs the task id to locate the correct head, which follows the standard TIL scenario. To our knowledge, neither soft-masking of the parameters in the feature extractor nor in the classification head has been done by any existing work.

4. Experiments

Datasets: The proposed SPG is evaluated using 5 CL datasets. Their statistics are given in Table 1. Below, we use "-n" to depict that n tasks are created from each dataset (n takes 10 or 20). Classes in the first three datasets are split by task, so each task has a disjoint set of classes. On the other hand, all tasks in the last two datasets have the same set of classes. We refer to the tasks in the former datasets as "dissimilar tasks" in which CF is the essential problem to solve, while we regard the tasks in latter datasets as "similar tasks" as the ability of knowledge transfer is more important.

(1) CIFAR100-n (C-n): CIFAR100 (Krizhevsky & Hinton, 2009) is a dataset that has images of 100 classes. We split it into n tasks so that each task has $100/n$ classes. **(2) TinyImageNet-n (T-n)**: TinyImageNet (Wu et al., 2017) is a modified subset of the original ImageNet (Russakovsky et al., 2015) dataset, and has 200 classes. Each task contains $200/n$ classes. **(3) ImageNet-100 (I-100)**: ImageNet (Russakovsky et al., 2015) contains 1000 classes of objects. We split it to 100 tasks, each of which has 10 classes, to stress-test systems using a large number of tasks and classes. **(4) F-CelebA-n (FC-n)**: Federated CelebA (Liu et al., 2015) is a dataset of celebrities' face images with several attributes.

Table 1. Statistics of the CL datasets. n can take 10 and 20. Validation sets are used for early stopping.

Dataset	#Tasks	#Classes per task	#Train	#Validation	#Test
C-n	n	$100/n$	$45,000$	$5,000$	$10,000$
T-n	n	$200/n$	$90,000$	$10,000$	$10,000$
I-100	100	10	$1,000,000$	$100,000$	$50,000$
FC-n	n	2	$400n$	$40n$	$80n$
FE-n	n	62	$3100n$	$310n$	$620n$

We use it with binary labels indicating whether he/she in the image is smiling or not. Each task consists of images of one celebrity. **(5) F-EMNIST-n (FE-n)**: Federated EMNIST (Liu et al., 2015) is a dataset that has 62 classes of hand-written symbols written by different persons. Each task consists of hand-written symbols of one person.

Baselines: We use 16 baselines. 11 of them are existing classical and most recent task incremental learning (TIL) systems, **EWC** (Kirkpatrick et al., 2017), **A-GEM** (Chaudhry et al., 2019), **SI** (Zenke et al., 2017), **UCL** (Ahn et al., 2019), **TAG** (Malviya et al., 2022), **PGN** (Rusu et al., 2016), **PathNet** (Fernando et al., 2017), **HAT** (Serra et al., 2018), **CAT** (Ke et al., 2020), **SupSup** (Wortsman et al., 2020), and **WSN** (Kang et al., 2022). Additionally, three simple methods are used for references: multi-task learning (**MTL**) that trains all the tasks together, one task learning (**ONE**) that learns a separate model/network for each task and thus has no CF or KT, and naive continual learning (**NCL**) that learns each new task without taking any care of previous tasks, i.e., no mechanism to deal with CF. Since HAT, our main baseline and perhaps the most effective TIL system, adopts AlexNet (Krizhevsky et al., 2012) as its backbone, all our experiments are conducted with AlexNet. For other baselines, their original codes are used with switching their backbones to AlexNet for fair comparison. Furthermore, to compare our *soft-masking* with the *regularization-based approach* and our gradient-based importance with Fisher information matrix (FI) based importance in EWC, two more baselines **EWC-GI** and **SPG-FI** are created. EWC-GI is EWC with its FI based importance replaced by our gradient-based importance (GI) in Section 3.1, i.e., the same penalty/regularization in EWC is applied on our accumulated importance, $\gamma_i^{\leq t-1}$ in Equation (5) when learning task t (no soft-masking). SPG-FI is SPG with our gradient-based importance replaced by FI based importance in EWC. The network size of each method is presented in Appendix E.

Evaluation Metrics: The following three metrics are used. Let α_i^j be the test accuracy of task i task just after a model completes task j.

(1) *Accuracy*: The average of accuracy for all tasks of a dataset after learning the final task. It is computed by $1/T \sum_t^T \alpha_t^T$, where T is the total number of tasks.
(2) *Forward transfer*: This measures how much the learning of previous tasks contributes to the learning of the current task. It is computed by $1/T \sum_t^T (\alpha_t^t - \beta_t)$, where β_t repre-

Table 2. Accuracy results in percent. Best methods in each dataset are emphasized in **bold**, and second best methods are underlined.

| Model | Dissimilar tasks | | | | | | Similar tasks | | | | |
	C-10	C-20	T-10	T-20	I-100	(Avg.)	FC-10	FC-20	FE-10	FE-20	(Avg.)
(MTL)	$76.4_{\pm0.3}$	$78.4_{\pm0.4}$	$52.7_{\pm0.3}$	$59.6_{\pm1.2}$	$64.8_{\pm0.4}$	66.4	$87.5_{\pm0.7}$	$88.3_{\pm0.2}$	$86.2_{\pm0.7}$	$87.2_{\pm2.2}$	87.3
(ONE)	$66.9_{\pm3.1}$	$76.5_{\pm0.7}$	$43.5_{\pm3.0}$	$54.5_{\pm0.9}$	$49.3_{\pm0.4}$	58.1	$74.6_{\pm2.6}$	$78.8_{\pm2.3}$	$80.9_{\pm1.5}$	$79.7_{\pm1.4}$	78.5
NCL	$50.9_{\pm1.7}$	$54.2_{\pm5.3}$	$37.2_{\pm0.9}$	$41.1_{\pm1.0}$	$30.6_{\pm1.2}$	42.8	$84.4_{\pm1.1}$	$84.1_{\pm1.4}$	$86.1_{\pm0.8}$	$86.4_{\pm0.3}$	85.3
A-GEM	$50.8_{\pm1.0}$	$56.9_{\pm7.1}$	$36.2_{\pm0.6}$	$41.7_{\pm1.1}$	$32.1_{\pm1.1}$	43.5	$83.2_{\pm3.6}$	$83.2_{\pm1.9}$	$86.6_{\pm0.2}$	$86.9_{\pm0.3}$	85.0
PGN	$65.1_{\pm0.6}$	$75.5_{\pm0.4}$	$44.0_{\pm0.8}$	$53.5_{\pm0.4}$	$45.2_{\pm0.4}$	56.7	$74.7_{\pm3.5}$	$74.7_{\pm2.8}$	$82.5_{\pm1.0}$	$82.6_{\pm0.3}$	78.6
PathNet	$69.1_{\pm0.5}$	$75.5_{\pm1.0}$	$46.0_{\pm1.5}$	$51.9_{\pm1.6}$	$42.0_{\pm1.5}$	56.9	$79.3_{\pm1.2}$	$80.5_{\pm0.4}$	$84.3_{\pm0.2}$	$84.5_{\pm0.4}$	82.1
HAT	$62.8_{\pm0.7}$	$71.8_{\pm1.1}$	$45.5_{\pm1.0}$	$51.7_{\pm2.1}$	$45.3_{\pm1.9}$	55.4	$79.0_{\pm3.1}$	$81.9_{\pm0.7}$	$83.8_{\pm0.9}$	$84.6_{\pm0.8}$	82.3
CAT	$64.2_{\pm0.6}$	$73.9_{\pm1.1}$	$43.7_{\pm0.6}$	$50.9_{\pm0.8}$	N/A	N/A	$82.9_{\pm1.3}$	$82.9_{\pm3.7}$	$82.9_{\pm1.4}$	$84.1_{\pm0.9}$	83.2
SupSup	$66.2_{\pm0.2}$	$75.6_{\pm0.3}$	$44.0_{\pm0.2}$	$54.1_{\pm0.3}$	$48.6_{\pm0.1}$	57.7	$75.0_{\pm2.3}$	$78.1_{\pm1.4}$	$80.5_{\pm0.5}$	$79.7_{\pm0.2}$	78.3
UCL	$64.8_{\pm0.8}$	$74.0_{\pm0.6}$	$45.4_{\pm0.3}$	$55.1_{\pm0.5}$	$37.4_{\pm0.6}$	55.4	$86.2_{\pm0.5}$	$86.5_{\pm0.5}$	$85.1_{\pm0.7}$	$85.0_{\pm1.6}$	85.7
SI	$62.9_{\pm0.3}$	$70.3_{\pm0.7}$	$45.9_{\pm0.6}$	$52.6_{\pm0.7}$	$44.1_{\pm0.2}$	55.2	$86.4_{\pm0.8}$	$\underline{86.8}_{\pm0.3}$	$\underline{87.6}_{\pm0.3}$	$\mathbf{87.9}_{\pm0.2}$	$\underline{87.2}$
TAG	$60.6_{\pm0.7}$	$68.4_{\pm0.9}$	$43.0_{\pm0.8}$	$49.5_{\pm0.4}$	$44.9_{\pm0.3}$	53.3	$74.3_{\pm3.7}$	$77.3_{\pm2.1}$	$84.2_{\pm0.5}$	$83.8_{\pm0.4}$	79.9
WSN	$\mathbf{69.3}_{\pm0.2}$	$\mathbf{76.9}_{\pm0.5}$	$47.8_{\pm0.5}$	$\underline{57.8}_{\pm0.5}$	$51.9_{\pm0.4}$	$\underline{60.7}$	$83.9_{\pm1.2}$	$84.1_{\pm0.7}$	$85.5_{\pm0.2}$	$86.3_{\pm0.2}$	84.9
EWC	$61.6_{\pm0.9}$	$60.7_{\pm2.7}$	$36.5_{\pm1.1}$	$41.5_{\pm1.3}$	$25.4_{\pm1.4}$	45.1	$81.2_{\pm3.0}$	$86.1_{\pm0.9}$	$86.9_{\pm0.3}$	$86.8_{\pm0.6}$	85.2
EWC-GI	$63.3_{\pm1.2}$	$60.1_{\pm1.9}$	$\underline{48.3}_{\pm1.0}$	$48.6_{\pm1.9}$	$\underline{52.7}_{\pm0.2}$	54.6	$83.4_{\pm3.0}$	$84.6_{\pm1.7}$	$86.6_{\pm1.5}$	$\underline{87.4}_{\pm0.9}$	85.5
SPG-FI	$60.5_{\pm0.2}$	$67.7_{\pm1.0}$	$43.9_{\pm0.6}$	$51.2_{\pm0.8}$	$48.8_{\pm0.8}$	54.4	$\underline{86.7}_{\pm0.6}$	$86.2_{\pm0.8}$	$87.5_{\pm0.3}$	$\mathbf{87.9}_{\pm0.2}$	87.1
SPG	$\underline{67.7}_{\pm0.3}$	$\underline{75.9}_{\pm1.1}$	$\mathbf{48.4}_{\pm0.3}$	$\mathbf{59.1}_{\pm0.5}$	$\mathbf{58.1}_{\pm0.5}$	**61.8**	$\mathbf{87.0}_{\pm0.9}$	$\mathbf{87.1}_{\pm0.2}$	$\mathbf{87.7}_{\pm0.2}$	$\mathbf{87.9}_{\pm0.1}$	**87.4**

sents the test accuracy of task t in the ONE method, which learns each task separately.

(3) *Backward transfer*: This measures how the learning of the current task affects the performance of the previous tasks. Negative values indicate forgetting. It is computed by $1/(T-1)\sum_{t}^{T-1}\left(\alpha_t^T - \alpha_t^t\right)$.

4.1. Training Details

The networks are trained with SGD by minimizing the cross-entropy loss except for TAG, which uses the RMSProp optimizer as SGD-based TAG has not been provided by the authors. The mini-batch size is 64 except MTL that uses 640 for its stability to learn more tasks and classes together. Hyper-parameters, such as the dropout rate or each method's specific hyper-paramters, are searched based on Tree-structured Parzen Estimator (Bergstra et al., 2011). With the found best hyper-parameters, the experiments are conducted 5 times with different seeds, and the average accuracy and standard deviation are reported.

4.2. Results

Tables 2 to 4 report the accuracy, forward and backward transfer results, respectively. Since CAT takes too much time to train, proportionally to the square of the number of tasks, we are unable to get its results for I-100 (ImageNet with 100 tasks) due to our limited computational resources.

Dissimilar Tasks (C-n, T-n, I-100): MTL performs the best in all cases, but NCL performs poorly due to serious CF (negative backward transfer) as expected. While PGN, PathNet, HAT, CAT, and SupSup can achieve training with

no forgetting (0 backward transfer), on average SPG clearly outperforms all of them as their transfer is limited. Although SPG underperforms PathNet in C-10, its accuracy is markedly lower in the other settings due to PathNet's capacity problem (see Section 4.2.1). The backward transfer results in Table 4 show that SPG has slight forgetting. However, its positive forward transfer results in Table 3 more than make up for the forgetting and give SPG the best final accuracy in most cases. As we explained in Section 2, the regularization-based approach is closely related to our work. However, the representative methods, EWC, SI and UCL, perform poorly due to serious CF (see Table 4) that cannot be compensated by their positive forward transfer (see Table 3). Although TAG has almost no CF, its forward transfer is limited, resulting in poorer final accuracy. While SPG underperforms WSN in C-10 and C-20 slightly, SPG outperforms all baselines in other dissimilar task datasets; especially, in the more realistic and difficult dataset I-100, for which SPG is significantly better, outperforming WSN by 6.2%.

Comparing our gradient based importance (GI) and Fisher information (FI) matrix based importance, we observe that EWC-GI outperforms EWC except for C-20 (EWC is less than 1% better), and SPG significantly outperforms SPG-FI in all cases. Comparing soft-masking and regularization using the same importance measure, we can see SPG is markedly better than EWC-GI, and SPG-FI is also better than EWC except for C-10 (EWC is only 1% better). These results clearly demonstrate that our gradient based importance (GI) and soft-masking are much more effective than standard ***regularization methods***.

Similar Tasks (FC-n, FE-n): Table 2 shows that SPG achieves the best in all cases due to its strongest *positive forward* (see Table 3) and *positive backward knowledge transfer* ability (see Table 4). NCL, A-GEM, UCL, and SI also perform well with positive or very little negative backward transfer since the tasks are similar and CF hardly happens. TAG underperforms them due to it's limited transfer. PGN, PathNet, HAT, and WSN have lower accuracy as their forward transfer is limited (i.e., they just reuse learned parameters in the forward pass) and no positive backward transfer. SupSup, which does not have any mechanism for knowledge transfer, results in much lower performance. CAT is slightly better due to its stronger positive forward transfer and no negative backward transfer.

Comparing GI and FI, SPG-FI and EWC-GI perform similarly to SPG and EWC as suppressing updates of important parameters becomes less critical for similar tasks and thus the choice of GI or FI is not important. Comparing soft-masking and regularization, EWC-GI and EWC are worse than SPG and SPG-FI in all cases, indicating that soft-masking is still more effective as regularization may hinder the learning of new tasks more.

Summary: SPG markedly outperforms all the baselines. When tasks are dissimilar, its positive forward transfer capability overcomes its slight forgetting (negative backward transfer) and achieves the best overall results. It has the strong positive forward transfer even with dissimilar tasks, which has not been realized by the other parameter isolation-based baselines. When tasks are similar, SPG has both positive forward and backward transfer to achieve the best accuracy results. Keeping most parameters trainable in SPG promotes knowledge transfer. Moreover, we observe that soft-masking (SPG and SPG-FI) is better than regularization (EWC-GI and EWC) and that our gradient-based importance (SPG and EWC-GI) is better than FI (SPG-FI and EWC).

4.2.1. CAPACITY CONSUMPTION

The reason that PGN, PathNet, HAT, CAT, and WSN have lower accuracy on average than SPG despite the fact that they can learn with no forgetting (see backward transfer in Table 4) is mainly because they suffer from the capacity problem, which is also indirectly reflected in lower forward transfer in Table 3. Although SupSup does not suffer from this problem, its architecture prevents transfer and gives markedly poorer performances especially in similar tasks (see Table 2). As discussed in Section 1, since these parameter isolation-based methods freeze a sub-network for each task, as more tasks are learned, the capacity of the network left for learning new knowledge becomes less and less except for SupSup, which leads to poorer performance for later tasks. Table 5 shows the percentage of parameters in the whole network that are completely blocked by HAT

Table 3. Forward transfer results in percent.

Model	Dissimilar tasks						Similar tasks				
	C-10	C-20	T-10	T-20	I-100	(Avg.)	FC-10	FC-20	FE-10	FE-20	(Avg.)
NCL	−7.1	−2.7	−3.2	−4.3	−0.7	−3.6	+7.9	+6.2	+4.1	+5.9	+6.0
A-GEM	−3.2	−0.9	−3.7	−3.5	−0.5	−2.4	+8.4	+5.4	+4.3	+6.4	+6.1
PGN	−1.8	−1.0	+0.5	−1.0	−3.9	−1.4	+0.2	−4.1	+1.6	+2.9	+0.1
PathNet	+2.2	−0.9	+2.5	−2.6	−7.1	−1.2	+4.7	+1.7	+3.4	+4.8	+3.7
HAT	−4.0	−4.6	+2.0	−2.8	−3.8	−2.7	+4.4	+3.2	+2.9	+4.9	+3.8
CAT	−2.7	−2.5	+0.1	−3.6	N/A	N/A	+8.4	+4.1	+2.0	+4.4	+4.7
SupSup	−0.6	−0.8	+0.5	−0.3	−0.4	−0.4	+0.5	−0.8	−0.4	+0.1	−0.2
UCL	+3.9	**+6.1**	+4.9	+7.8	+2.7	+5.1	+8.3	+5.6	+3.0	+4.7	+5.4
SI	**+7.5**	+3.5	**+13.3**	**+8.6**	**+12.6**	**+9.1**	+7.1	+5.8	**+5.9**	+7.5	+6.6
TAG	−5.6	−6.6	−0.2	−4.2	−5.1	−4.3	−0.5	−1.8	+3.5	+4.1	+1.3
WSN	+2.4	+0.5	+4.3	+3.3	+2.9	+2.7	+9.3	+5.3	+4.6	+6.6	+6.4
EWC	+0.4	−2.5	−3.5	−5.6	−1.2	−2.5	+7.8	+5.8	+4.7	+6.7	+6.3
EWC-GI	−1.5	+2.3	+6.3	+6.8	+12.1	+5.2	+8.8	+5.7	+4.2	+6.0	+6.2
SPG-FI	+3.2	+1.0	+1.5	+4.4	+6.6	+3.4	+9.6	**+6.5**	+5.4	+7.4	+7.2
SPG	+5.5	+5.0	+8.9	+7.9	+10.2	+7.5	**+9.8**	+5.6	+5.9	+7.8	**+7.3**

Table 4. Backward transfer results in percent.

Model	Dissimilar tasks						Similar tasks				
	C-10	C-20	T-10	T-20	I-100	(Avg.)	FC-10	FC-20	FE-10	FE-20	(Avg.)
NCL	−9.9	−20.5	−3.4	−9.6	−17.9	−12.3	+2.1	−0.9	+1.2	+0.9	+0.8
A-GEM	−14.3	−21.8	−4.0	−9.7	−16.6	−13.3	+0.2	−1.0	+1.5	+0.9	+0.4
PGN	**0.0**	**0.0**	**0.0**	**0.0**	0.0	**0.0**	0.0	0.0	0.0	0.0	0.0
PathNet	**0.0**	**0.0**	**0.0**	**0.0**	0.0	**0.0**	0.0	0.0	0.0	0.0	0.0
HAT	**0.0**	**0.0**	**0.0**	**0.0**	0.0	**0.0**	0.0	0.0	0.0	0.0	0.0
CAT	**0.0**	**0.0**	**0.0**	**0.0**	N/A	N/A	0.0	0.0	0.0	0.0	0.0
SupSup	**0.0**	**0.0**	**0.0**	**0.0**	0.0	**0.0**	0.0	0.0	0.0	0.0	0.0
UCL	−6.6	−9.0	−3.4	−7.7	−14.5	−8.2	+3.7	+2.2	+1.2	+0.7	+2.0
SI	−12.8	−5.7	−12.1	−11.0	−17.6	−11.8	**+5.2**	+2.3	+0.9	+0.7	**+2.3**
TAG	−0.8	−1.6	−0.4	−0.8	+0.9	−0.5	+0.4	+0.4	−0.3	0.0	+0.1
WSN	**0.0**	**0.0**	**0.0**	**0.0**	0.0	**0.0**	0.0	0.0	0.0	0.0	0.0
EWC	−6.4	−13.9	−3.9	−7.8	−22.7	−10.9	−1.3	+1.6	+1.4	+0.4	+0.5
EWC-GI	−1.4	−19.6	−1.6	−13.4	−8.6	−8.9	0.0	+0.1	**+1.6**	**+1.7**	+0.9
SPG-FI	−10.7	−10.3	−1.3	−8.1	−7.0	−7.5	+2.8	+1.0	+1.4	+0.8	+1.5
SPG	−5.3	−5.9	−4.4	−3.4	−1.2	−4.0	+2.9	**+2.8**	+0.9	+0.5	+1.8

and SPG (layer-wise results are given in Appendix B). SPG blocks much fewer parameters than HAT does in all cases. This advantage allows SPG to have more flexibility and capacity to learn while mitigating forgetting, which leads to better performances.

Figure 3 plots the forward transfer of the limited datasets due to the space limitations (all results are presented in Appendix A). A positive value in the figures means that a method's forward test result (the test result of the task obtained right after the task is learned) is better than ONE, benefited by the forward knowledge transfer from previous tasks. We can clearly observe a downward trend of these baselines for dissimilar tasks ((a)-(c)). We believe that this is due to the capacity problem, i.e., as more tasks are learned, they gradually lose their learning capacity. On the other hand, SPG shows a upward trend in all cases, even for dissimilar tasks, thanks to its positive forward transfer, which indicates SPG has a much higher capacity to learn. For Figure 3(d), the difference is not obvious as the tasks are similar and the capacity problem is less serious. However, SPG still keeps the best forward transfer.

Table 5. Each cell reports how many percentage of parameters are completely blocked (i.e., importance of 1) just after learning task t. T is the total number of tasks (e.g., $T = 10$ for C-10).

t	Model	Dissimilar tasks					Similar tasks			
		C-10	C-20	T-10	T-20	I-100	FC-10	FC-20	FE-10	FE-20
1	HAT	1.9±0.5	15.9±1.1	2.9±0.2	0.2±0.1	23.5±0.6	0.3±0.1	6.3±0.9	23.9±1.0	19.0±1.1
	SPG	0.1±0.0	0.2±0.0	0.1±0.0	0.1±0.0	0.1±0.0	0.1±0.0	0.2±0.0	0.1±0.0	0.1±0.0
$T/2$	HAT	22.4±1.1	98.2±0.3	36.7±1.9	21.1±1.4	99.8±0.0	3.4±0.3	65.7±2.4	86.8±0.8	94.0±1.6
	SPG	0.9±0.1	2.9±0.3	0.5±0.1	1.4±0.2	4.1±0.4	0.8±0.0	1.3±0.1	0.6±0.3	1.0±0.2
T	HAT	41.8±1.3	99.6±0.1	57.5±1.8	38.6±2.1	99.9±0.1	9.8±1.1	79.5±0.9	97.9±0.3	97.5±0.9
	SPG	2.0±0.2	5.2±0.6	0.9±0.1	3.1±0.2	6.3±0.6	1.4±0.1	2.0±0.3	1.1±0.5	1.5±0.3

Table 6. The results for pruning parameters based on the gradient-based importance.

Pruning strategy	Dissimilar tasks					Similar tasks			
	C-10	C-20	T-10	T-20	I-100	FC-10	FC-20	FE-10	FE-20
Nothing	$75.5_{\pm1.0}$	$78.9_{\pm1.9}$	$46.7_{\pm0.8}$	$49.2_{\pm1.2}$	$48.8_{\pm1.2}$	$74.5_{\pm3.0}$	$87.8_{\pm2.6}$	$86.0_{\pm0.5}$	$85.4_{\pm0.8}$
Lowest 10%	$73.6_{\pm2.4}$	$75.4_{\pm4.8}$	$41.2_{\pm0.9}$	$45.8_{\pm1.5}$	$43.3_{\pm2.6}$	$74.0_{\pm2.9}$	$85.5_{\pm1.9}$	$83.1_{\pm1.7}$	$84.3_{\pm0.4}$
Random 10%	$69.5_{\pm2.1}$	$70.5_{\pm4.2}$	$34.7_{\pm1.1}$	$44.8_{\pm1.6}$	$40.8_{\pm1.2}$	$71.3_{\pm2.0}$	$84.3_{\pm3.7}$	$70.8_{\pm4.6}$	$72.3_{\pm3.1}$
Highest 10%	$19.4_{\pm8.5}$	$24.6_{\pm8.1}$	$6.2_{\pm1.1}$	$13.2_{\pm2.4}$	$17.1_{\pm3.6}$	$39.0_{\pm0.6}$	$49.3_{\pm4.1}$	$55.5_{\pm9.7}$	$19.8_{\pm8.4}$
Lowest 20%	$68.6_{\pm5.1}$	$72.7_{\pm6.7}$	$36.4_{\pm3.3}$	$43.1_{\pm1.8}$	$39.4_{\pm4.1}$	$73.3_{\pm3.9}$	$85.5_{\pm3.7}$	$61.5_{\pm2.7}$	$81.6_{\pm3.5}$
Random 20%	$58.3_{\pm2.2}$	$58.7_{\pm7.4}$	$20.4_{\pm0.6}$	$37.2_{\pm3.1}$	$32.3_{\pm2.6}$	$68.5_{\pm3.0}$	$85.0_{\pm3.6}$	$37.2_{\pm3.4}$	$43.5_{\pm5.4}$
Highest 20%	$10.8_{\pm1.0}$	$22.5_{\pm5.6}$	$5.5_{\pm0.6}$	$10.2_{\pm0.2}$	$11.7_{\pm1.3}$	$38.8_{\pm0.0}$	$49.3_{\pm4.1}$	$20.7_{\pm5.8}$	$7.8_{\pm1.5}$

(a) C-20   (b) T-20
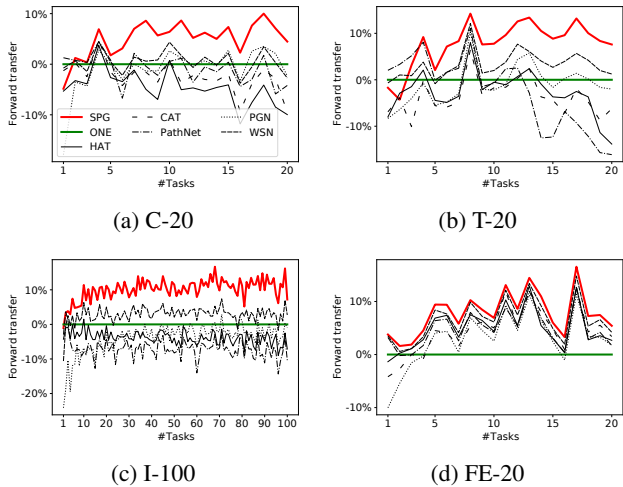
(c) I-100   (d) FE-20

Figure 3. The forward transfer of each task along with the number of tasks learned. (a) to (c) are the plots with dissimilar tasks, while (d) is the one with similar tasks.

4.2.2. VALIDITY OF GRADIENT-BASED IMPORTANCE

We further analyze how the proposed gradient-based importance metric indicates the contribution of parameters with different importance values to the performance of each task. Specifically, we evaluate the accuracy on the first task of each dataset (no continual learning) after pruning some parameters based on their importance so that we can confirm whether the importance metric is co-related with the performance change.

The following four strategies for choosing which parameters to prune are compared – **(1) Nothing**: we do not prune any parameters. **(2) Random n%**: n% of parameters are randomly pruned regardless of their importance. **(3) Lowest n%**: the parameters with the lowest n% of importance are pruned. **(4) Highest n%**: the parameters with the highest n% of importance are pruned.

The average results over 5 different seeds are presented in Table 6. It can be clearly observed that pruning parameters with higher importance degrades the performance more (e.g., "Lowest 10%" is much better than "Highest 10%").

(a) Fine-tuning for TinyImageNet after CL for C-10   (b) Fine-tuning for CIFAR100 after CL for I-100
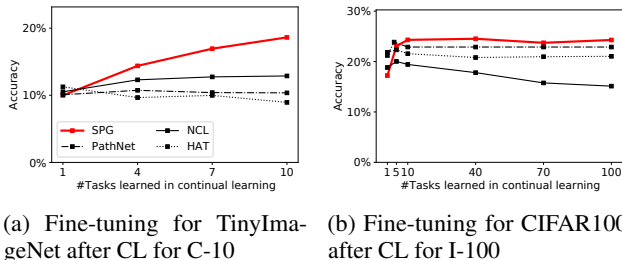
Figure 4. The learning of representation through continual learning. The x-axis means the number of tasks learned in continual learning (CL). The pair of a CL/non-CL dataset for (a) and (b) is C-10/TinyImageNet and I-100/CIFAR100, respectively.

When it is "Highest 20%", the accuracy is almost like random chance classification (e.g., it is 10.8% for C-10 while the random chance also gives 10%). We believe this observation implies that our gradient-based importance metric effectively indicates the importance of parameters.

4.2.3. BETTER REPRESENTATION LEARNING OF SPG

We found that SPG's stronger performance is manifested in its strong representations learning. We conduct additional experiments from the perspective of representation learning. In particular, a model that has just incrementally learned some tasks of a CL dataset (e.g., 5 tasks of C-10) are used as a frozen feature extractor to learn another dataset (not split into tasks), which we call "non-CL dataset", by fine-tuning a new classifier using the dataset, e.g., CIFAR100 or TinyImageNet (a non-CL dataset contains all classes of the dataset shown in Table 1). We evaluate the test accuracy for a non-CL dataset. Three pairs of a CL/non-CL dataset are tested: (1) C-10/TinyImageNet, (2) I-100/CIFAR100, and (3) T-10/CIFAR100. We here only show the results for (1) and (2) due to the space limitations (all the results are provided in Appendix D).

Figure 4(a) shows that SPG learns better representations in continual learning than baselines (we use NCL and two strong performing baselines from Table 2). HAT even deteriorates, which indicates that hard-masking of some parameters/units leave less network capacity to learn good features.

8

Table 7. The results for the ablation studies.

Ablation	Dissimilar tasks						Similar tasks				
	C-10	C-20	T-10	T-20	I-100	(Avg.)	FC-10	FC-20	FE-10	FE-20	(Avg.)
SPG	**67.7**$_{\pm 0.3}$	**75.9**$_{\pm 1.1}$	**48.4**$_{\pm 0.3}$	**59.1**$_{\pm 0.5}$	**58.1**$_{\pm 0.5}$	**61.8**	**87.0**$_{\pm 0.9}$	**87.1**$_{\pm 0.2}$	$\underline{87.7}_{\pm 0.2}$	87.9$_{\pm 0.1}$	**87.4**
SPG (w/o CHI)	65.4$_{\pm 0.6}$	70.4$_{\pm 2.2}$	46.9$_{\pm 0.7}$	55.3$_{\pm 0.8}$	$\underline{57.6}_{\pm 0.4}$	59.1	$\underline{86.2}_{\pm 0.3}$	$\underline{86.9}_{\pm 0.5}$	**87.9**$_{\pm 0.3}$	**88.1**$_{\pm 0.3}$	$\underline{87.3}$
SPG (w/o SMH)	$\underline{66.1}_{\pm 1.0}$	$\underline{74.8}_{\pm 0.3}$	$\underline{47.2}_{\pm 0.5}$	$\underline{56.9}_{\pm 0.4}$	55.2$_{\pm 0.8}$	$\underline{60.0}$	85.8$_{\pm 0.7}$	86.8$_{\pm 0.5}$	87.2$_{\pm 0.2}$	87.5$_{\pm 0.2}$	86.8
SPG (w/ hard-masking)	63.6$_{\pm 0.4}$	73.2$_{\pm 1.2}$	46.8$_{\pm 0.4}$	53.7$_{\pm 0.2}$	51.1$_{\pm 0.3}$	57.7	84.3$_{\pm 0.4}$	86.3$_{\pm 0.5}$	86.8$_{\pm 0.0}$	$\underline{88.0}_{\pm 0.2}$	86.3

Figure 4(b) shows that while NCL significantly degrades due to its serious forgetting in such an extreme continual learning on I-100 (with 100 tasks), SPG keeps the best after learning a few tasks. These clearly confirm that SPG learns better representations to enable better continual learning.

4.3. Ablation Studies

As SPG has three core mechanisms that contribute to its performance, we evaluate whether each of them is beneficial. The first one is *cross-head importance* (CHI), which is introduced in Equation (4) with the motivation of suppressing the update of important parameters for previous tasks more. In the ablation **SPG (w/o CHI)**, γ_i^t is replaced with $\gamma_i^{t,t}$ in Equation (4) and the previous tasks' heads are not used for computing importance. The second one is *soft-masking* of each *head* (SMH), which is introduced in Equation (7) for the purpose of balancing the training of the feature extractor and the classification head. In the ablation **SPG (w/o SMH)**, Equation (7) is not employed. The last one is soft-masking (not hard-masking), which is the core technique of SPG to keep most parameters trainable while mitigating CF. While the reported results so far are based on soft-masking, it is also possible to convert the importance to binary (0 or 1) masks using a pre-defined threshold. In the ablation **SPG (w/ hard-masking)**, if the threshold is 0.6, this variant treats importance larger than 0.6 as 1 (blocking updates of parameters), otherwise 0 (not blocking). We search for the best threshold for each dataset from $\{0.2, 0.4, 0.6, 0.8\}$.

The ablation results are presented in Table 7. For CHI, it improves the performance especially in dissimilar tasks (up to 5.5% for C-20) by blocking more gradient flow to mitigate CF. On the other hand, CHI does not contributes much for similar tasks, which is reasonable as blocking parameters becomes less important in similar tasks. Nevertheless, SPG with CHI still works the best for similar tasks on average and CHI does not cause side effects. More quantitative analysis of how CHI contributes to suppressing parameter updates is given in Appendix C. For SMH, it delivers positive performance gains both in dissimilar and similar tasks. These results indicate that the lack of balance in the training between the feature extractor and head, which can happen without SMH, adversely affects the performance. The promotion of knowledge transfer by SMH becomes most prominent for I-100 (up to 2.9%) as the effectiveness of knowledge

transfer is more important in such an extreme case with more tasks. For whether masking should be soft or hard, SPG with hard-masking is significantly worse than SPG with soft-masking, which demonstrates that it is difficult to find a good threshold for hard-masking. Soft-masking is more flexible and effective. These results clearly confirm that SPG enjoys all of the three different mechanisms.

5. Conclusion

To overcome the difficulty of balancing forgetting prevention and knowledge transfer in continual learning, we proposed a novel and simple method, called SPG, that blocks/masks parameters not completely but partially to give the model more flexibility and capacity to learn. The proposed soft-masking mechanism not only overcomes CF but also performs knowledge transfer automatically. Although it is conceptually related to the regularization approach, as we have argued and evaluated, it markedly outperforms the regularization approach. Extensive experiments demonstrate that SPG markedly outperforms all the strong baselines.

Acknowledgements

References

Ahn, H., Cha, S., Lee, D., and Moon, T. Uncertainty-based Continual Learning with Adaptive Regularization. In Proc. of NeurIPS, 2019.

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., and Page-Caccia, L. Online Continual Learning with Maximal Interfered Retrieval. In Proc. of NeurIPS, 2019.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for Hyper-Parameter Optimization. In Proc. of NeurIPS, 2011.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient Lifelong Learning with A-GEM. In Proc. of ICLR, 2019.

Deja, K., Wawrzyński, P., Marczak, D., Masarczyk, W., and Trzciński, T. BinPlay: A Binary Latent Autoencoder for Generative Replay Continual Learning. In Proc. of IJCNN, 2021.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. PathNet: Evolution Channels Gradient Descent in Super Neural Networks, 2017.

Kang, H., Mina, R. J. L., Madjid, S. R. H., Yoon, J., Hasegawa-Johnson, M., Hwang, S. J., and Yoo, C. D. Forget-free Continual Learning with Winning Subnetworks. In Proc. of ICML, 2022.

Ke, Z., Liu, B., and Huang, X. Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks. In Proc. of NeurIPS, 2020.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. In Proc. of NAS, 2017.

Krizhevsky, A. and Hinton, G. Learning Multiple Layers of Features from Tiny Images, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In Proc. of NeurIPS, 2012.

Li, Z. and Hoiem, D. Learning without Forgetting. In Proc. of ECCV, 2016.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep Learning Face Attributes in the Wild. In Proc. of ICCV, 2015.

Lopez-Paz, D. and Ranzato, M. Gradient Episodic Memory for Continual Learning. In Proc. of NeurIPS, 2017.

Mallya, A. and Lazebnik, S. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In Proc. of CVPR, 2018.

Malviya, P., Ravindran, B., and Chandar, S. TAG: Task-based Accumulated Gradients for Lifelong Learning. In Proc. of CoLLAs, 2022.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. iCaRL: Incremental Classifier and Representation Learning. In Proc. of CVPR, 2017.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference. In Proc. of ICLR, 2019.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive Neural Networks, 2016.

Serra, J., Suris, D., Miron, M., and Karatzoglou, A. Overcoming Catastrophic Forgetting with Hard Attention to the Task. In Proc. of ICML, 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual Learning with Deep Generative Replay. In Proc. of NeurIPS, 2017.

Sokar, G., Mocanu, D. C., and Pechenizkiy, M. SpaceNet: Make Free Space For Continual Learning. Neurocomputing, 439:1–11, 2021.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. Supermasks in Superposition. In Proc. of NeurIPS, 2020.

Wu, J., Zhang, Q., and Xu, G. Tiny ImageNet Challenge, 2017.

Yoon, J., Kim, S., Yang, E., and Hwang, S. J. Scalable and Order-robust Continual Learning with Additive Parameter Decomposition. In Proc. of ICLR, 2020.

Zenke, F., Poole, B., and Ganguli, S. Continual Learning Through Synaptic Intelligence. In Proc. of ICML, 2017.

Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., and Kuo, C. C. J. Class-incremental Learning via Deep Model Consolidation. In Proc. of WACV, 2020.

A. Forward Transfer

Figure 3 in the main paper shows the forward transfer plots for some datasets but not all due to space limitations. Figure 5 presents the forward transfer results for all datasets. It can be clearly seen that SPG has the best positive forward transfer and keeps or even grows it constantly in all cases. On the other hand, the other parameter isolation-based methods, PGN, PathNet, HAT, CAT, and WSN lose their ability for the forward transfer in later tasks for the dissimilar task (i.e., (a) to (e)).
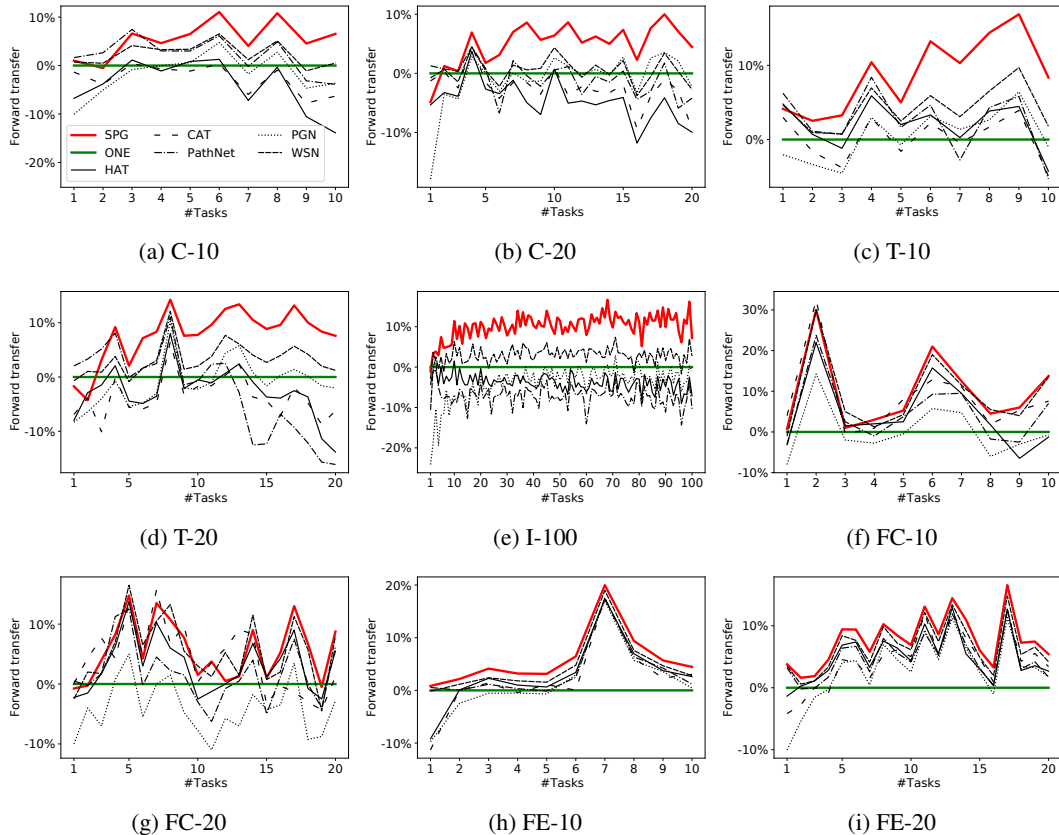

Figure 5. Forward transfer results. (a) to (e) are for dissimilar tasks, and (f) to (i) are for similar tasks.

B. Capacity Consumption at Each Layer

We show the percentage of parameters in the whole network that are fully blocked in Table 5 in the main text of the paper. Here Table 8 presents the same result for each layer. We use AlexNet as the backbone, and it has three convolution layers followed by two fully-connected layers.

As we described in Section 4.2.1, SPG blocks much fewer parameters than what HAT does in all cases. Additionally, we can see from Table 8 the significant difference between HAT and SPG in their layer-wise tendency. HAT blocks more parameters in earlier layers (e.g., after learning task 5 of C-10, 77.8% of parameters in the 1st convolution layer are completely blocked while 52.4% of the ones in the 2nd convolution layer are), which is reasonable given that the earlier layers are supposed to extract basic features and thus changing their parameters without being blocked could easily cause more forgetting than in later layers. On the other hand, SPG contrarily tends to completely block more parameters in later layers (e.g., after learning task 5 of C-10, 0.0% of parameters in the 1st convolution layer are completely blocked while 0.6% of ones in 2nd convolution layer are). Since SPG computes parameters' importance based on their gradients with regard to the loss through normalization, this result implies that later layers are likely to have more parameters on which some of the tasks highly depend. It can be said that SPG keeps earlier layers alive with less blocking for better basic feature learning (i.e., leading to positive knowledge transfer) while it blocks some specific parameters in later layers that are supposed to be important for previous tasks, which is different from what HAT does.

Table 8. The percentage of parameters that are completely blocked for each layer. T is the total number of tasks (e.g., $T = 10$ for C-10).

(a) Results for the 1st convolution layer.

t	Model	Dissimilar tasks					Similar tasks			
		C-10	C-20	T-10	T-20	I-100	FC-10	FC-20	FE-10	FE-20
1	HAT	$25.0_{\pm5.3}$	$24.4_{\pm8.4}$	$31.6_{\pm6.0}$	$14.4_{\pm5.5}$	$60.0_{\pm4.4}$	$17.8_{\pm4.4}$	$40.0_{\pm8.7}$	$75.0_{\pm3.1}$	$67.2_{\pm9.9}$
	SPG	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$
$T/2$	HAT	$77.8_{\pm2.7}$	$98.4_{\pm1.7}$	$83.4_{\pm3.6}$	$79.1_{\pm2.5}$	$100.0_{\pm0.0}$	$73.4_{\pm7.6}$	$100.0_{\pm0.0}$	$98.8_{\pm1.2}$	$99.4_{\pm0.8}$
	SPG	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$
T	HAT	$95.6_{\pm1.5}$	$100.0_{\pm0.0}$	$97.2_{\pm1.2}$	$95.3_{\pm1.0}$	$100.0_{\pm0.0}$	$98.8_{\pm1.8}$	$100.0_{\pm0.0}$	$100.0_{\pm0.0}$	$100.0_{\pm0.0}$
	SPG	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$

(b) Results for the 2nd convolution layer.

t	Model	Dissimilar tasks					Similar tasks			
		C-10	C-20	T-10	T-20	I-100	FC-10	FC-20	FE-10	FE-20
1	HAT	$5.6_{\pm1.6}$	$7.3_{\pm2.9}$	$8.9_{\pm2.3}$	$1.5_{\pm1.1}$	$33.9_{\pm5.0}$	$2.2_{\pm0.6}$	$12.0_{\pm2.0}$	$55.7_{\pm6.6}$	$45.7_{\pm12.1}$
	SPG	$0.1_{\pm0.0}$	$0.1_{\pm0.1}$	$0.0_{\pm0.0}$	$0.1_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$	$0.0_{\pm0.1}$	$0.0_{\pm0.0}$
$T/2$	HAT	$52.4_{\pm2.5}$	$98.0_{\pm1.7}$	$63.7_{\pm6.2}$	$56.2_{\pm4.5}$	$100.0_{\pm0.0}$	$41.3_{\pm4.7}$	$98.3_{\pm1.2}$	$98.4_{\pm1.0}$	$99.2_{\pm1.0}$
	SPG	$0.6_{\pm0.1}$	$2.1_{\pm0.4}$	$0.3_{\pm0.1}$	$0.9_{\pm0.3}$	$1.5_{\pm0.3}$	$0.1_{\pm0.1}$	$0.2_{\pm0.1}$	$0.5_{\pm0.5}$	$0.6_{\pm0.2}$
T	HAT	$81.0_{\pm2.0}$	$99.8_{\pm0.3}$	$90.2_{\pm2.8}$	$86.4_{\pm4.5}$	$100.0_{\pm0.0}$	$86.9_{\pm2.2}$	$99.4_{\pm0.3}$	$100.0_{\pm0.0}$	$100.0_{\pm0.0}$
	SPG	$1.1_{\pm0.2}$	$3.1_{\pm0.5}$	$0.4_{\pm0.1}$	$1.5_{\pm0.3}$	$1.8_{\pm0.4}$	$0.2_{\pm0.1}$	$0.3_{\pm0.2}$	$0.8_{\pm0.7}$	$0.9_{\pm0.3}$

(c) Results for the 3rd convolution layer.

t	Model	Dissimilar tasks					Similar tasks			
		C-10	C-20	T-10	T-20	I-100	FC-10	FC-20	FE-10	FE-20
1	HAT	$4.8_{\pm1.3}$	$8.7_{\pm1.7}$	$7.5_{\pm1.1}$	$0.7_{\pm0.3}$	$30.6_{\pm2.8}$	$1.0_{\pm0.2}$	$8.0_{\pm1.5}$	$43.7_{\pm6.5}$	$36.4_{\pm6.2}$
	SPG	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.2_{\pm0.1}$	$0.0_{\pm0.0}$	$0.0_{\pm0.0}$
$T/2$	HAT	$42.6_{\pm3.6}$	$98.7_{\pm0.7}$	$60.3_{\pm4.1}$	$47.1_{\pm2.3}$	$100.0_{\pm0.0}$	$14.7_{\pm1.2}$	$88.7_{\pm4.2}$	$98.0_{\pm0.7}$	$99.1_{\pm0.4}$
	SPG	$0.7_{\pm0.1}$	$2.1_{\pm0.4}$	$0.3_{\pm0.1}$	$1.0_{\pm0.2}$	$1.2_{\pm0.2}$	$0.5_{\pm0.1}$	$1.0_{\pm0.3}$	$0.3_{\pm0.2}$	$0.3_{\pm0.1}$
T	HAT	$66.8_{\pm2.7}$	$99.7_{\pm0.5}$	$87.1_{\pm2.2}$	$74.1_{\pm3.4}$	$100.0_{\pm0.0}$	$43.1_{\pm4.0}$	$95.3_{\pm1.9}$	$99.8_{\pm0.5}$	$99.8_{\pm0.2}$
	SPG	$1.2_{\pm0.2}$	$3.1_{\pm0.7}$	$0.4_{\pm0.1}$	$1.5_{\pm0.3}$	$1.8_{\pm0.2}$	$0.9_{\pm0.2}$	$1.4_{\pm0.4}$	$0.5_{\pm0.3}$	$0.5_{\pm0.1}$

(d) Results for the 4th fully-connected layer.

t	Model	Dissimilar tasks					Similar tasks			
		C-10	C-20	T-10	T-20	I-100	FC-10	FC-20	FE-10	FE-20
1	HAT	$2.7_{\pm0.8}$	$12.6_{\pm1.6}$	$4.0_{\pm0.3}$	$0.2_{\pm0.1}$	$25.7_{\pm1.4}$	$0.3_{\pm0.0}$	$6.5_{\pm0.9}$	$27.7_{\pm2.3}$	$22.7_{\pm2.1}$
	SPG	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.1}$	$0.0_{\pm0.0}$	$0.1_{\pm0.0}$	$0.2_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$
$T/2$	HAT	$29.7_{\pm1.4}$	$98.3_{\pm0.6}$	$47.7_{\pm2.1}$	$29.4_{\pm2.2}$	$99.9_{\pm0.1}$	$3.8_{\pm0.4}$	$70.9_{\pm3.7}$	$90.2_{\pm1.1}$	$95.6_{\pm1.4}$
	SPG	$0.8_{\pm0.2}$	$2.4_{\pm0.2}$	$0.4_{\pm0.1}$	$1.4_{\pm0.2}$	$1.6_{\pm0.2}$	$0.8_{\pm0.1}$	$1.4_{\pm0.1}$	$0.6_{\pm0.2}$	$0.8_{\pm0.1}$
T	HAT	$52.4_{\pm1.1}$	$99.7_{\pm0.3}$	$72.3_{\pm1.6}$	$50.6_{\pm2.5}$	$99.9_{\pm0.1}$	$12.1_{\pm1.3}$	$84.0_{\pm1.8}$	$98.5_{\pm0.4}$	$98.2_{\pm0.7}$
	SPG	$1.6_{\pm0.2}$	$4.1_{\pm0.3}$	$0.7_{\pm0.1}$	$2.5_{\pm0.2}$	$2.3_{\pm0.4}$	$1.5_{\pm0.2}$	$2.2_{\pm0.3}$	$0.9_{\pm0.4}$	$1.2_{\pm0.2}$

(e) Results for the 5th fully-connected layer.

t	Model	Dissimilar tasks					Similar tasks			
		C-10	C-20	T-10	T-20	I-100	FC-10	FC-20	FE-10	FE-20
1	HAT	$1.3_{\pm0.4}$	$17.9_{\pm1.0}$	$2.1_{\pm0.2}$	$0.1_{\pm0.0}$	$22.0_{\pm0.5}$	$0.2_{\pm0.0}$	$6.0_{\pm1.0}$	$20.8_{\pm0.5}$	$16.2_{\pm1.3}$
	SPG	$0.1_{\pm0.0}$	$0.2_{\pm0.0}$	$0.1_{\pm0.0}$	$0.0_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$	$0.1_{\pm0.0}$
$T/2$	HAT	$17.6_{\pm1.5}$	$98.2_{\pm0.1}$	$29.9_{\pm2.0}$	$15.4_{\pm1.2}$	$99.8_{\pm0.0}$	$2.1_{\pm0.2}$	$61.8_{\pm1.8}$	$84.6_{\pm0.8}$	$92.8_{\pm1.8}$
	SPG	$1.0_{\pm0.1}$	$3.2_{\pm0.4}$	$0.5_{\pm0.1}$	$1.5_{\pm0.2}$	$5.5_{\pm0.6}$	$0.7_{\pm0.0}$	$1.2_{\pm0.2}$	$0.7_{\pm0.3}$	$1.1_{\pm0.2}$
T	HAT	$34.9_{\pm2.0}$	$99.6_{\pm0.1}$	$48.5_{\pm2.1}$	$30.6_{\pm2.0}$	$99.9_{\pm0.1}$	$6.1_{\pm1.1}$	$76.4_{\pm0.6}$	$97.4_{\pm0.3}$	$97.1_{\pm1.1}$
	SPG	$2.2_{\pm0.3}$	$5.8_{\pm0.7}$	$1.1_{\pm0.2}$	$3.4_{\pm0.3}$	$8.6_{\pm0.7}$	$1.3_{\pm0.0}$	$1.9_{\pm0.2}$	$1.2_{\pm0.6}$	$1.8_{\pm0.3}$

Table 9. Quantitative contribution of CHI in learning task t. T is the total number of tasks (e.g., $T = 10$ for C-10).

	C-10				C-20				T-10			
t	F-each	G-each	F-total	G-total	F-each	G-each	F-total	G-total	F-each	G-each	F-total	G-total
2	$0.64_{\pm0.02}$	$0.15_{\pm0.02}$	$0.42_{\pm0.02}$	$0.13_{\pm0.00}$	$0.52_{\pm0.09}$	$0.10_{\pm0.03}$	$0.35_{\pm0.06}$	$0.09_{\pm0.02}$	$0.49_{\pm0.01}$	$0.13_{\pm0.00}$	$0.32_{\pm0.01}$	$0.12_{\pm0.00}$
$T/2$	$0.84_{\pm0.01}$	$0.31_{\pm0.02}$	$0.24_{\pm0.01}$	$0.04_{\pm0.00}$	$0.88_{\pm0.01}$	$0.33_{\pm0.02}$	$0.15_{\pm0.03}$	$0.01_{\pm0.00}$	$0.77_{\pm0.01}$	$0.28_{\pm0.01}$	$0.24_{\pm0.01}$	$0.04_{\pm0.00}$
T	$0.90_{\pm0.03}$	$0.39_{\pm0.04}$	$0.15_{\pm0.01}$	$0.01_{\pm0.00}$	$0.90_{\pm0.00}$	$0.34_{\pm0.01}$	$0.05_{\pm0.00}$	$0.00_{\pm0.00}$	$0.87_{\pm0.01}$	$0.36_{\pm0.01}$	$0.13_{\pm0.00}$	$0.01_{\pm0.00}$

	T-20				I-100				FC-10			
t	F-each	G-each	F-total	G-total	F-each	G-each	F-total	G-total	F-each	G-each	F-total	G-total
2	$0.52_{\pm0.01}$	$0.14_{\pm0.01}$	$0.34_{\pm0.02}$	$0.12_{\pm0.01}$	$0.47_{\pm0.01}$	$0.11_{\pm0.00}$	$0.30_{\pm0.02}$	$0.11_{\pm0.01}$	$0.53_{\pm0.04}$	$0.14_{\pm0.01}$	$0.38_{\pm0.03}$	$0.12_{\pm0.00}$
$T/2$	$0.88_{\pm0.02}$	$0.37_{\pm0.02}$	$0.14_{\pm0.00}$	$0.01_{\pm0.00}$	$0.91_{\pm0.00}$	$0.36_{\pm0.00}$	$0.02_{\pm0.00}$	$0.00_{\pm0.00}$	$0.83_{\pm0.06}$	$0.25_{\pm0.01}$	$0.25_{\pm0.01}$	$0.04_{\pm0.00}$
T	$0.92_{\pm0.01}$	$0.41_{\pm0.02}$	$0.05_{\pm0.00}$	$0.00_{\pm0.00}$	$0.91_{\pm0.01}$	$0.37_{\pm0.00}$	$0.02_{\pm0.00}$	$0.00_{\pm0.00}$	$0.85_{\pm0.02}$	$0.28_{\pm0.01}$	$0.15_{\pm0.02}$	$0.01_{\pm0.00}$

	FC-20				FE-10				FE-20			
t	F-each	G-each	F-total	G-total	F-each	G-each	F-total	G-total	F-each	G-each	F-total	G-total
2	$0.41_{\pm0.07}$	$0.10_{\pm0.01}$	$0.29_{\pm0.04}$	$0.10_{\pm0.01}$	$0.51_{\pm0.15}$	$0.08_{\pm0.02}$	$0.34_{\pm0.11}$	$0.07_{\pm0.02}$	$0.49_{\pm0.08}$	$0.09_{\pm0.01}$	$0.33_{\pm0.08}$	$0.08_{\pm0.01}$
$T/2$	$0.79_{\pm0.08}$	$0.24_{\pm0.01}$	$0.12_{\pm0.01}$	$0.01_{\pm0.00}$	$0.72_{\pm0.09}$	$0.17_{\pm0.04}$	$0.25_{\pm0.07}$	$0.02_{\pm0.00}$	$0.75_{\pm0.04}$	$0.26_{\pm0.02}$	$0.13_{\pm0.01}$	$0.01_{\pm0.00}$
T	$0.84_{\pm0.06}$	$0.30_{\pm0.01}$	$0.08_{\pm0.01}$	$0.00_{\pm0.00}$	$0.86_{\pm0.02}$	$0.24_{\pm0.05}$	$0.11_{\pm0.01}$	$0.01_{\pm0.00}$	$0.89_{\pm0.00}$	$0.31_{\pm0.01}$	$0.09_{\pm0.03}$	$0.00_{\pm0.00}$
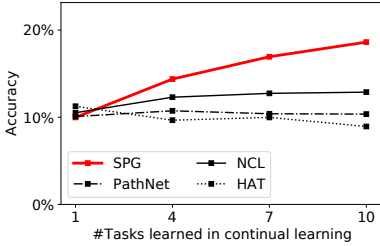
(a) Fine-tuning for TinyImageNet after CL for C-10
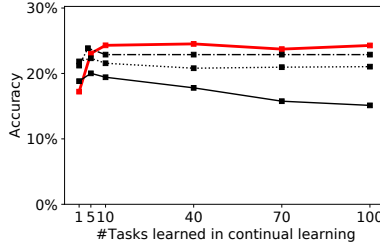
(b) Fine-tuning for CIFAR100 after CL for I-100
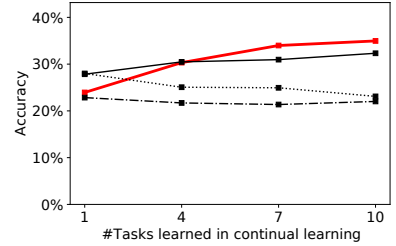
(c) Fine-tuning for CIFAR100 after CL for T-10

Figure 6. The learning of representation through continual learning. The x-axis means the number of tasks learned in continual learning (CL). The pair of a CL/non-CL for (a), (b) and (c) is C-10/TinyImageNet, I-100/CIFAR100, and T-10/CIFAR100, respectively.

C. Quantitative Analysis on Cross-Head Importance (CHI)

We analyze in detail how CHI quantitatively contributes to suppressing parameter updates with the following four metrics.

(1) Overwrite _Frequency_ at _each_ task (F-each): How often does the importance from CHI have a larger value than the one from the current task (1 means that it always happens)? It corresponds to cases where $\gamma_i^{t,\tau} > \gamma_i^{t,t}$ for any $\tau (1 \leq \tau < t)$ in Equation (4).

(2) Overwrite _Gap_ at _each_ task (G-each): When the cases of F-each happen, how much is the difference of overwriting on average? It is defined by the average of $\max(\gamma_i^{t,1}, \cdots, \gamma_i^{t,t-1}) - \gamma_i^{t,t}$.

(3) Overwrite _Frequency_ in _total_ (F-total): How often does the importance from CHI actually overwrite the accumulated importance through the maximum operation? It corresponds to cases where $\gamma_i^{t,\tau} > \gamma_i^{\leq t-1}$ for any $\tau (1 \leq \tau < t)$ in Equation (5).

(4) Overwrite _Gap_ in _total_ (G-total): When the cases of F-total happen, how much is the difference of overwriting on average? It is defined by the average of $\max(\gamma_i^{t,1}, \cdots, \gamma_i^{t,t-1}) - \gamma_i^{\leq t-1}$.

The result is presented in Table 9. We can clearly observe that CHI adds more importance to some parameters (e.g., in C-10, about 15-42% of parameters constantly update their accumulated importance by the ones from CHI), which is denoted by F-total. Since we introduce CHI to further mitigate forgetting by accumulating more importance, this expectation is consistent with the observed results. Although CHI also overwrites the accumulated importance in similar tasks as frequently as in dissimilar tasks (see F-each and F-total), it happens with a smaller gap overall (see G-each and G-total), which is reasonable as the tasks are similar thus parameters can have similar gradients among different tasks.

D. Representation Learning in Continual Learning

All results (i.e., the pairs of a CL and non-CL dataset are C-10/TinyImageNet, I-100/CIFAR100, and T-10/CIFAR100) are presented in Figure 6. We can see that SPG learns better representations in continual learning than baselines in all cases.

E. Network Size

The number of learnable parameters of each system is presented in Table 10. Note that all approaches adopt AlexNet as their backbone, and the number of parameters vary depending on their additional structures such as attention mechanisms or sub-modules. It also depends on datasets because each dataset has a different number of tasks and in TIL, each task has a different classification head and the number of units in each classification head depends on the number of classes in each task. It can be seen that CAT and SupSup need more parameters than SPG and other approaches.

Table 10. The number of learnable parameters of each model. "M" means a million $(1,000,000)$.

Model	Dissimilar tasks					Similar tasks			
	C-10	C-20	T-10	T-20	I-100	FC-10	FC-20	FE-10	FE-20
(MTL)	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
(ONE)	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
NCL	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
A-GEM	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
PGN	6.7M	6.7M	6.7M	6.7M	8.3M	6.0M	6.6M	7.5M	8.9M
PathNet	6.6M	6.8M	6.7M	6.6M	8.4M	6.4M	6.4M	7.8M	8.7M
HAT	6.8M	6.8M	7.0M	7.0M	9.0M	6.6M	6.7M	7.8M	9.1M
CAT	39.5M	39.7M	40.8M	40.9M	N/A	38.5M	38.9M	46.2M	55.1M
SupSup	65.2M	130.2M	65.4M	130.4M	652.0M	65.0M	130.1M	65.8M	131.7M
UCL	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
SI	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
TAG	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
WSN	6.7M	6.7M	6.8M	6.8M	8.5M	6.5M	6.5M	7.7M	8.9M
EWC	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
EWC-GI	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
SPG-FI	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M
SPG	6.7M	6.7M	6.9M	6.9M	8.6M	6.5M	6.6M	7.7M	9.0M