Helpful or Harmful: Inter-Task Association in Continual Learning

Hyundong Jin and Eunwoo Kim

School of Computer Science and Engineering
Chung-Ang University, South Korea
{jude0316, eunwoo}@cau.ac.kr

Abstract. When optimizing sequentially incoming tasks, deep neural networks generally suffer from catastrophic forgetting due to their lack of ability to maintain knowledge from old tasks. This may lead to a significant performance drop of the previously learned tasks. To alleviate this problem, studies on continual learning have been conducted as a countermeasure. Nevertheless, it suffers from an increase in computational cost due to the expansion of the network size or a change in knowledge that is favorably linked to previous tasks. In this work, we propose a novel approach to differentiate helpful and harmful information for old tasks using a model search to learn a current task effectively. Given a new task, the proposed method discovers an underlying association knowledge from old tasks, which can provide additional support in acquiring the new task knowledge. In addition, by introducing a sensitivity measure to the loss of the current task from the associated tasks, we find cooperative relations between tasks while alleviating harmful interference. We apply the proposed approach to both task- and class-incremental scenarios in continual learning, using a wide range of datasets from small to large scales. Experimental results show that the proposed method outperforms a large variety of continual learning approaches for the experiments while effectively alleviating catastrophic forgetting.

Keywords: continual learning, task association, model search

1 Introduction

Deep learning algorithms are generally optimized for a single task or multiple different tasks. However, it is difficult for them to apply to a more challenging scenario; learning sequentially incoming tasks without accessing the old ones. This is because a single deep network lacks the ability to accommodate both new and old knowledge using a limited set of shared parameters. Specifically, when a network learned from old tasks is further trained with a new task, it can easily forget the previously learned tasks. This phenomenon is called catastrophic forgetting [28,17], a major hindrance in learning sequential tasks with a deep neural network.

Many approaches have been recently proposed as a remedy for memorizing knowledge. They are generally classified into four categories: regularization

[2,5,21,17,42], replay [6,25,30,31,37,40,15,10], dynamic expansion [3,32,33,41], and structural allocation [1,26,27,36] approaches. The regularization approaches add a new penalty to consolidate old knowledge while training on a new task. However, this will not be effective when a long sequence of tasks is involved because the penalty does not sufficiently prevent the change of contributable parameters. Replay methods retain a small number of instances from old tasks and learn them with new task samples in a joint manner. For a large number of tasks, the number of retained instances of each task decreases due to the size limit to hold the old instances, causing performance degradation. Dynamic expansion methods generally expand the network when a new task comes in or when performance does not meet a predetermined criterion. As a result, they require a large amount of computation costs, constraining their applicability to real-world problems. The last category, structural allocation, constructs disjoint sets of parameters for tasks, respectively, and learns a new task using the older sets. However, it will not be helpful when we use unwanted old parameters that can negatively affect the new task.

The proposed method falls within the last category. Unlike other categories, it encourages to use a disjoint set of parameters for each task, preventing the network from containing the mixture of old and new knowledge. Despite its benefit, the strategy will not be promising if we use adversarial parameters that negatively affect a new task when using old parameter sets altogether [27]. From this, a question arises can we discover helpful knowledge from old tasks to enrich the knowledge of a new task for structural allocation based continual learning.

In this paper, we reformulate the forgetting problem into a task interference problem in continual learning and solve it using model (task) selection to discover cooperative tasks. To this end, we propose a novel approach to differentiate helpful and harmful knowledge from old tasks in a continual learning framework. Specifically, the proposed framework is based upon an architecture consisting of a global feature extractor and multiple heads (classifiers) corresponding to tasks. The proposed approach is based on structural allocation using a train-prune-retrain learning paradigm [27] to address sequential tasks so that the feature extractor consists of disjoint sets of parameters corresponding to tasks, respectively. To find older tasks and exploit their knowledge given a task, we present a model search framework and apply it to the disjoint sets of parameters learned so far (at a coarser level). The proposed method further discovers critical parameters in element-wise by measuring sensitivity to the loss of a new task from the searched old tasks (at a finer level). The discovered knowledge is leveraged when learning the new task. Due to the search mechanism, we select an optimal subnetwork that can accelerate the model inference as well as discover a cooperative relationship among tasks.

We validate our method for both task-incremental and class-incremental learning scenarios. A number of large-scale datasets, including ImageNet [8] and diverse fine-grained datasets, are used in task-incremental learning. In class-incremental learning, ImageNet-50 [30], Split CIFAR-10 [19], and Split MNIST [20] are used. Experimental results show that the proposed method achieves

higher performance than state-of-the-art continual learning methods for both learning scenarios while minimizing the computation cost. The contributions of this work are mainly four-fold:

- We redesign the forgetting problem into a task association problem by taking task interference into account in the structural allocation strategy.
- We propose a novel continual learning approach that can effectively learn a new task by differentiating and absorbing helpful knowledge from old tasks.
- The proposed approach elaborately selects contributable parameters from older tasks using a gradient-based model search at a coarser level and a sensitivity measure at a finer level.
- Experimental results show that the proposal achieves remarkable performance improvement over diverse continual learning competitors while minimizing the computation cost.

2 Related work

Regularization method. Regularization approaches [17,42,5,2,21] in continual learning typically append a regularization term to suppress updating of parameters that are important to the previously learned tasks. The importance can be calculated by the Fisher information [17], the change of loss [42], or the derivative of the network output [2]. This line of methods is memory-efficient because it does not require instances of previous tasks and does not expand the size of the network. However, the regularization term does not effectively prevent the change of previous knowledge for many sequential tasks, resulting in drastic performance drop [12].

Replay method. The approaches rely on some raw samples [6,25,31,40,23,10] or generated instances by a generative model [30,37]. [6] and [25] mitigate forgetting by controlling gradient produced from a few replayed samples. [31] picks some nearest neighbor samples to the average sample per task, and [24] parameterizes exemplar samples which can be optimized. However, they repeatedly train a small number of old instances, potentially leading to an overfitting problem. Also, since generated instances may not exactly imitate the distribution of real instances, we may encounter a performance drop [4].

Dynamic expansion. Methods falling into this category dynamically expand the network structure when a new task comes in [33,41,3,32,38]. Progressive neural network [33] produces an additional network to perform a new task, which inherits the knowledge of old tasks by connecting to previously learned networks. However, this reveals a drawback that the computation cost increases proportionally to the number of tasks. Dynamically expandable network [41] expands the network architecture by adding a predetermined number of parameters and learning with sparse regularization for size optimization. However, this method is not free from the forgetting issue as many parameters of old tasks are updated for the new task.

Structural allocation. This family of approaches [26,27,36,1] does not suffer from updating old knowledge with a new one. The key is to assign a disjoint

subset of parameters to each task. The disjoint sets of old tasks are fixed [26,27,1] or rarely [36] updated while training a new task. This is usually achieved by pruning parameters [27], learning a mask [26], or attention through gradient descent [36]. However, these methods do not discover relationships between tasks, which may cause negative task interference. Recently, [1] has proposed to select an internal network structure with the channel-gating module, but it produces a non-negligible amount of additional parameters unlike ours.

Contributions. We reformulate the catastrophic forgetting problem into a task interference problem under the structural allocation framework. To solve the problem, we introduce a task association strategy through a model search that effectively differentiates helpful and harmful information from old tasks. Since we explore an optimally shrunk subnetwork from an architecture, it is faster than the structural allocation baseline [27] that leverages all previous knowledge (i.e., using the entire architecture). The proposed task selection approach results in performance improvement over its strong competitors [27,26] while consuming a small number of model parameters.

3 Methodology

3.1 Framework

The proposed method aims to enrich the knowledge of a new task from old tasks without absorbing negative information. To this end, we propose a novel approach to whether each old task and its parameters help learn the new task or not. The proposal is a structural allocation method based on a train-prune-retrain framework [27], where a disjoint set of parameters is assigned to each task in a network. Let us denote incoming tasks as $T^1, ..., T^t$, where the t-th task $T^t = \{\mathcal{X}^t, \mathcal{Y}^t\}$ contains data $\mathcal{X}^t = \{x_j^t\}_{j=1}^n$ and corresponding labels $\mathcal{Y}^t = \{y_j^t\}_{j=1}^n$. The first task T^1 is learned using the feature extractor f_{θ^1} and the classifier $g_{w^1}^1$ parameterized by θ^1 and w^1, respectively. For T^1, we allocate the set of task-specific parameters θ^1 to Θ^1 in the feature extractor, i.e., $\Theta^1 = [\theta^1]$. After learning T^1, redundant parameters in θ^1 are pruned away to reserve parameters for the next task, and the set of survived parameters $\tilde{\theta}^1$ squeezed from θ^1 performs T^1.

For the i-th task T^i, we allocate the set of parameters θ^i to the pruned locations in the feature extractor after learning up to T^{i-1}. One way to train parameters θ^i is to utilize the entire set of parameters $\Theta^i = [\tilde{\theta}^1, \ldots, \tilde{\theta}^{i-1}, \theta^i]$ in the forward pass and then the set of parameters θ^i assigned to the i^{th} task is updated in the backward pass [27]:

$$\begin{aligned} &\text{Forward: } f_{\theta_{fwd}^i}(x^i) \text{ where } \theta_{fwd}^i = \Theta^i, \\ &\text{Backward: } f_{\theta_{bwd}^i}(x^i) \text{ where } \theta_{bwd}^i = \theta^i. \end{aligned} \tag{1}$$

After learning θ^i, we apply a prune-retrain approach to get $\tilde{\theta}^i$ and the set of reusable parameters θ^{i+1} for the next task. In summary, we train incoming tasks and sequentially produce disjoint sets of parameters, $\tilde{\theta}^1, \cdots, \tilde{\theta}^{t-1}, \theta^t$ for t tasks.

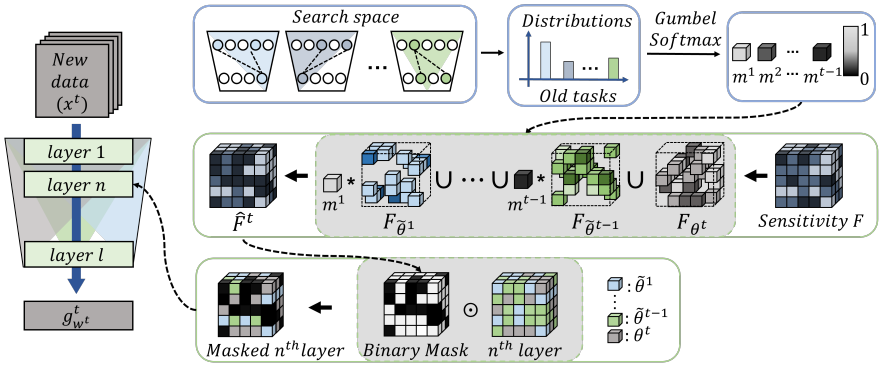

Fig. 1: A graphical illustration of the training process of the proposed method for the t-th task. It searches the old parameters to help optimize the t-th task. The architecture is obtained by the sampled variable m^i. The information from the proposed measure, $\hat{F}^t$, is computed by combining m^i and the sensitivity F. Helpful and harmful parameters are finally identified through a binary mask that is obtained from $\hat{F}^t$. From this, we get a masked layer obtained through the element-wise multiplication of the parameters in each layer and the corresponding mask. Best viewed in color.

To discover parameter sets that are critical for the new task t in the search space containing $\{\tilde{\theta}^1, \cdots, \tilde{\theta}^{t-1}\}$, in this work, we discover an association knowledge between tasks based on how each old task affects the new one. Note here that Eq. (1) may not avoid harmful interference between tasks because the forward pass takes the entire parameters from all previous tasks. By selectively using parameters of the previous tasks, which are positively cooperative to the new task, it can mitigate negative task interference. To discover helpful tasks with the corresponding parameters, we first search for an optimal subnetwork by a model search approach at a coarser level. Then, we further explore the parameters within the searched network that are sensitive to the loss of the new task at a finer level. Here, sensitive parameters can be understood as crucial old ones for the new task. By the search mechanism and the sensitivity measure, we can find the most promising architecture for the current task. The entire procedure of the proposal is illustrated in Fig. 1.

3.2 Search

We present a gradient-based search method to find previous tasks that are cooperative to the new task. We first define the search problem for the t-th task in continual learning as

$$\min_{a^t \in \mathcal{A}} \min_{\theta^t} \mathcal{L}(a^t, \theta^t). \tag{2}$$

Given the search space $\mathcal{A}$ containing a pool of candidate networks (corresponding to different combinations of parameters sets), we find an optimal net-

work structure a^t and update the set of parameters θ^t to minimize the loss. Since we aim to find cooperative old tasks for T^t, the search space is 2^{t-1}. As a naïve approach, during training task t, a previous parameter set $\tilde{\theta}^i$ is sampled and leveraged with the probability of

$$P_\alpha(\tilde{\theta}^i) = \text{softmax}(\alpha^i) = \frac{\exp(\alpha^i)}{\sum_{j=1}^{t-1}\exp(\alpha^j)}, \tag{3}$$

where $\alpha = (\alpha^1, \cdots, \alpha^{t-1})$ denotes learnable parameters that determine the sampling probability. Therefore, the output based on the probability over all previous sets becomes

$$\hat{\Theta}^t = \bigcup_{i=1}^{t-1}\left(m^i \cdot \tilde{\theta}^i\right) \cup \theta^t, \tag{4}$$

and it defines the structure a^t. Here, $m^i \in \{0, 1\}$ becomes 1 if softmax(α^i) is greater than pre-determined threshold and 0 otherwise.

However, we may not solve the problem in Eq. (2) using a gradient descent method due to (i) the discrete search space $\mathcal{A}$ and (ii) discontinuous relation between the sampling parameter α^i and m^i. For the discrete optimization problem, we relax it as the following problem:

$$\min_{\alpha}\min_{\theta^t} \mathbf{E}_{a^t\sim P_\alpha}[\ \mathcal{L}(a^t, \theta^t)\ ]. \tag{5}$$

To alleviate the latter problem, we present the Gumbel softmax trick [9,16] that produces the continuous random variable m^i.

$$m^i = \text{GumbelSoftmax}(\alpha^i|\alpha) = \frac{exp((\alpha^i + g^i)/\tau)}{\sum_{j=1}^{t-1} exp((\alpha^j + g^j)/\tau)}, \tag{6}$$

where g^i follows the Gumbel distribution [16] and τ is the control variable. While training, we discover the optimal architecture a^t of the feature extractor for T^t using Eqs. (5) and (6). We optimize the current set of parameters θ^t followed by updating α to select the contributable old parameters.

3.3 Sensitivity measure

To further explore useful parameters learned from the selected old tasks, we additionally introduce a promising measure that gives a sensitivity of each parameter to the loss of a new task, which can be calculated by the Fisher information [14] as

$$F(\theta) = \sum_n \mathbb{E}_{p_\theta(y|x_n)}[\nabla_\theta \log p_\theta(y|x_n) \nabla_\theta \log p_\theta(y|x_n)^\top], \tag{7}$$

where $\nabla_\theta \log p_\theta(y|x_n)$ denotes the gradient of the log-likelihood at all parameters θ. Fisher information captures the variability of the gradient and tells us which old parameters are sensitive (important) for the current task [17]. Note that

while [17] applies Fisher to prevent the update of sensitive old parameters, ours uses them to select helpful parameters.

Let us first express the sensitivity for the i^{th} task. $F_{\tilde{\theta}^i}$ denotes the element-wise Fisher values corresponding to $\tilde{\theta}^i$. For T^t, we gather all the Fisher information up to the task t as follows:

$$F^t(\theta) = \bigcup_{i=1}^{t-1} F_{\tilde{\theta}^i} \cup F_{\theta^t}. \tag{8}$$

Finally, θ^t_{fwd}, exploiting both sampled variable in Eq. (6) and the sensitivity in Eq. (8), is obtained as

$$\theta^t_{fwd} = \bigcup_{i=1}^{t-1} \left(\sigma_\phi(m^i \cdot F_{\tilde{\theta}^i}) \odot \tilde{\theta}^i \right) \cup \theta^t, \tag{9}$$

where $\odot$ is the element-wise multiplication. By applying the threshold function $\sigma_\phi(\cdot)$ to $m^i \cdot F_{\tilde{\theta}^i}$, we discover a final architecture a^t which is used in the forward pass.

3.4 Meta classifier

To perform the challenging class-incremental learning scenario, where task identity is unknown, we present a meta classifier to predict the task label of each sample and allocate the corresponding network. The meta classifier $h_{\theta^{mc}}(\cdot)$ has a few learnable parameters θ^{mc} but the total parameter increase is negligible. Features are extracted from the exemplar, x^t_e, corresponding to each task. By concatenating the features for t tasks, we compose data $\tilde{x}^t$ to learn the meta classifier, which is obtained by

$$\tilde{x}^t = \bigoplus_{i=1}^{t} flatten(\hat{a}^i(x^t_e)), \tag{10}$$

where $\bigoplus$ the concatenation operation and $\hat{a}^i$ is the architecture a^i excluding the head classifier $g^i_{w^i}$. Using the collected data, we learn the meta classifier and predict the task identity l to perform the chosen task l with the head classifier $g^l_{w^l}(\cdot)$, where

$$l \leftarrow h_{\theta^{mc}}(\tilde{x}^t),\ l \leq t. \tag{11}$$

To develop the meta classifier that learns discriminative task-wise features, we optimize it with the contrastive loss [7]. Unlike the contrastive loss used in self-supervised learning, which allows two different transformations to apply to an image as a positive pair, we treat samples in the same task as a positive pair. Formally, we compute the loss using positive pairs identified by their task IDs as

$$\begin{aligned} \mathcal{L}_{con} &= \sum_{i} \sum_{j \neq i} \pi(z_i, z_j) \\ \text{where } \pi(z_i, z_j) &= -\log \frac{\exp(sim(z_i, z_j)/\gamma)}{\sum_k \mathbb{1}_{[k \neq i]} \exp(sim(z_i, z_k)/\gamma)}, \end{aligned} \tag{12}$$

where $\mathbb{1}$, $sim(\cdot)$ and γ are the indicator function, cosine similarity, and temperature parameter, respectively. z is the extracted feature of $h_{\theta^{mc}}$, where z_i and z_j are extracted features from different tasks. The total loss with the cross-entropy loss $\mathcal{L}_{ce}$ for training $h_{\theta^{mc}}$ is

$$\mathcal{L}_{total} = \mathcal{L}_{con} + \lambda \mathcal{L}_{ce}, \tag{13}$$

where λ is a balancing factor between two losses.

4 Experiments

4.1 Datasets

We applied the proposed approach to discover **H**elpful or **H**armful parameters, named $\mathbf{H}^2$, to task-incremental and class-incremental learning scenarios. The datasets used in the task-incremental scenarios include ImageNet, diverse fine-grained datasets, CIFAR-10, and MNIST, where the fine-grained datasets are CUBS [39], Stanford Cars [18], Flowers [29], Wikiart [34], and Sketch [11]. ImageNet and the fine-grained datasets were used altogether in a task-incremental scenario and resized to 224×224 pixels. CIFAR-10 [19] contains $50,000$ training and $10,000$ test samples of the 32×32 pixels. This dataset was divided into five tasks, referred to as Split CIFAR-10, and each task is composed of two classes. MNIST [20] consists of $60,000$ training and $10,000$ test handwritten images of the 28×28 size. The dataset was divided into five tasks, referred to as Split MNIST, similar to Split CIFAR-10.

For class-incremental learning scenarios, we used three datasets: ImageNet-50 [8], Split CIFAR-10, and Split MNIST. The summary of all datasets used in the experiments is shown in Table 1. To construct ImageNet-50, we selected 50 classes randomly from the ImageNet dataset and resized to 32×32, following the practice in [30]. It contains five tasks, each of which has 10 classes and 13,000 images. We also have the same datasets, Split CIFAR-10 and Split MNIST, to task-incremental learning, but their task identities are unknown at test time in class-incremental learning.

Table 1: Datasets used in this work.

Dataset	ImageNet	CUBS	Stanford Cars	Flowers	Wikiart	Sketch	ImageNet-50	CIFAR-10	MNIST
# Train	1,287,167	5,994	8,144	2,040	42,129	16,000	65,000	50,000	60,000
# Test	50,000	5,794	8,041	6,149	10,628	40,000	2,500	10,000	10,000
# Class	1,000	200	196	102	195	250	50	10	10

Table 2: Classification accuracy (Acc, %) and the number of required parameters (Param, $\times 10^6$) on the ImageNet and find-grained datasets. Best results are indicated in bold font.

	ImageNet		CUBS		Stanford Cars		Flowers		Wikiart		Sketch		Avg	Avg
Method	Acc	Param	Acc	Param	Acc	Param	Acc	Param	Acc	Param	Acc	Param	Acc	Param
EWC-On	54.5	23.4	64.1	23.4	52.1	23.4	78.7	23.4	51.4	23.4	30.7	23.4	55.3	23.4
LwF	64.5	23.4	51.7	23.4	43.9	23.4	72.9	23.4	42.7	23.4	45.5	23.4	53.6	23.4
PackNet	75.7	**12.5**	80.4	15.6	86.1	18.0	93.0	19.7	69.4	21.0	76.7	22.0	80.2	18.1
Piggyback	**76.1**	23.4	81.5	20.5	89.6	19.7	94.7	22.3	71.3	16.3	**79.9**	18.0	82.2	20.0
H^2 (Ours)	75.7	**12.5**	**84.1**	**14.5**	**90.6**	**15.9**	**94.9**	**15.2**	**75.1**	**9.4**	76.2	**4.1**	**82.8**	**11.9**

4.2 Implementation details

We used the ResNet-50 [13] backbone architecture for the task-incremental learning scenario using the ImageNet and fine-grained datasets. The backbone models on Split MNIST and Split CIFAR-10 were a three-layer CNN and ResNet-18, respectively, for both task-incremental and class-incremental learning scenarios. All layers except the last convolutional layer are accompanied by the max pooling operation in the three-layer CNN. We used ResNet-18 for ImageNet-50. The meta classifier used in all class-incremental scenarios consists of three fully connected layers with the hyperparameters γ and λ of 0.5 and 1.6, respectively. The feature z is obtained by passing through two linear layers of the meta-classifier. For large-scale datasets, we resized every image into the size of 224×224 and applied a center crop. We applied random cropping and horizontal flip to augment ImageNet-50 and Split CIFAR-10 except for the Split MNIST dataset. We set the Gumbel softmax parameter τ for all experiments to 1.0. We set the threshold ϕ in Eq. (9) to 0.01 for all the experiments. We trained the backbone and the meta classifier using stochastic gradient descent. We report the average results over five independent runs for all the experiments.

4.3 Task-incremental learning

Large-scale datasets We first demonstrated the task association strategy of the proposed method compared with EWC-On [35], LwF [22], PackNet [27] and Piggyback [26]. We applied the compared approaches for six large-scale datasets in order of ImageNet, CUBS, Stanford Cars, Flowers, Wikiart, and Sketch, where we regard each dataset as a task. The scenario is challenging because the datasets are from different domains and rarely share many common classes. For a fair comparison, ours has the same pruning ratio to PackNet for each task.

Table 2 shows the experimental results of the methods on the six datasets. The proposed method, H^2, outperforms the compared approaches for most of the tasks. The regularization approaches, EWC-On and LwF, do not effectively address the large-scale tasks. PackNet does not give satisfactory results, especially for CUBS, Stanford Cars, and Wikiart. This is probably due to negative interference between tasks as the tasks may be irrelevant to one another. Piggyback

Table 3: Task-incremental learning results on the Split MNIST and Split CIFAR-10 datasets. We provide the accuracy for each task after training the last task, T^5, where T^i denotes the i-th task. Best results are indicated in bold font.

	Split MNIST						Split CIFAR-10					
Method	T^1	T^2	T^3	T^4	T^5	Avg	T^1	T^2	T^3	T^4	T^5	Avg
Joint	99.9	99.9	99.9	100.0	99.5	99.9	99.6	96.4	97.9	99.8	98.3	98.3
EWC-On	97.1	99.4	93.4	98.2	93.2	96.3	75.8	80.4	80.3	95.2	96.0	85.5
LwF	99.8	97.9	99.7	99.9	98.5	99.2	94.8	87.3	67.1	50.5	51.4	70.2
PackNet	99.7	99.4	99.8	99.8	98.2	99.3	98.8	93.4	95.7	98.7	97.8	96.8
Piggyback	**100.0**	99.6	**100.0**	99.7	97.0	99.2	96.1	85.4	91.8	96.6	96.4	93.3
HAT	99.9	99.6	99.9	99.8	99.0	99.7	98.8	91.1	95.3	98.5	97.8	96.3
TAC	**100.0**	99.4	**100.0**	99.9	99.3	99.7	**99.4**	91.7	95.0	98.3	97.8	96.4
H^2 (Ours)	**100.0**	**99.8**	**100.0**	**100.0**	**99.6**	**99.9**	98.8	**94.1**	**96.5**	**98.8**	**98.3**	**97.3**

performs better than PackNet but performs poorer than ours. The results show that finding helpful parameters by the proposed measure can alleviate destructive task interference. When it comes to the number of parameters, ours takes a much smaller number of parameters than the compared methods. Overall, the proposed method selects fewer parameters while allowing better performance than the existing approaches, showing its excellence.

Split CIFAR-10 and Split MNIST We conducted additional task-incremental learning experiments on Split MNIST and Split CIFAR-10, where we construct five incremental tasks for each dataset. We compared with EWC-On [35] and LwF [22] in the regularization category, PackNet [27], Piggyback [26], HAT [36], and TAC [1] in the structural allocation category. We also compared with a joint learning method that learns multiple tasks simultaneously. Note that since Piggyback requires a pretrained backbone, we pretrained a three-layer CNN by collecting 10K data samples from the original MNIST data. For Split CIFAR-10, we used the ImageNet pretrained ResNet-18 backbone for Piggyback.

The results for the Split MNIST are summarized on the left of Table 3. From the table, we can observe that H^2 performs better than other continual learning approaches on average and is almost similar to the joint learning results. The results for Split CIFAR-10 are shown on the right of Table 3. For both experiments, the structural allocation methods consistently perform better than the regularization methods. The proposed method gives higher accuracy than the regularization and other structural allocation methods with a larger margin. Compared to the recently proposed structural allocation method, TAC, ours achieves meaningful performance improvement with a margin of 0.9% for Split CIFAR-10.

4.4 Class-incremental learning

ImageNet-50 We evaluated H^2 to the class-incremental learning scenario using ImageNet-50 to compare with two class-incremental learning methods [31,40].

Table 4: Class-incremental learning results on ImageNet-50 in terms of the exemplar size.

Exemplar size	65,000	2,000					5,000				
Method	Joint	iCaRL	BiC	TAC	PackNet	H^2	iCaRL	BiC	TAC	PackNet	H^2
Acc (%)	58.7	43.6	21.2	40.4	48.0	**49.7**	47.3	35.2	42.9	48.2	**55.5**

While task-incremental learning knows task ID, class-incremental learning predicts it through the meta classifier described in Section 3.4. The backbone architecture for this scenario was ResNet-18. The dataset consists of five tasks of 10 classes each. For the scenario, the proposed method utilizes exemplars to learn the meta classifier and is compared with other replay-based methods, iCaRL [31] and BiC [40]. We also compared with two structural allocatioin methods, PackNet [27] and TAC [1]. Because PackNet does not address class-incremental learning, we applied the same meta classifier. Even if we make use of exemplars in class-incremental learning, we use them when we train the meta-classifier. Fine-tuning with a small number of exemplars can distort the knowledge learned on complete data of the previous tasks. We also compared H^2 with a joint learning method that has full accessibility to the previous data. We report the average accuracy of the learned tasks after learning the last task.

Table 4 shows the experimental results under two exemplar sizes. The proposed method yields better results than the replay-based methods. The proposed method achieves 6.1%, 28.5%, 1.7% and 9.3% higher accuracy than iCaRL, BiC, PackNet, and TAC, respectively, for the exemplar size of 2,000. For the exemplar size of 5,000, ours achieves more significant performance improvement over those methods with the margins of 8.2%, 20.3%, 5.8%, and 12.6%, respectively. The results show that ours is more efficient in terms of accuracy under the same memory sizes than the competitors for class-incremental learning.

Split CIFAR-10 and Split MNIST We also conducted additional class-incremental learning scenarios on the Split CIFAR-10 and Split MNIST datasets, respectively. In this scenario, we compared H^2 with the same approaches in the previous experiment. A three-layer CNN was used for Split MNIST, and ResNet-18 was used for Split CIFAR-10. We show the results under different exemplar sizes, from small to large, obtained after learning all tasks.

Fig. 2 (left) shows the results of the compared methods for Split MNIST. H^2 gives higher accuracy than other approaches for most exemplar sizes. The structural allocation methods, TAC and PackNet, perform better than the replay methods, iCaRL and BiC, because they do not update old parameter sets. However, these methods perform poorer than the proposed approach using the search mechanism for task association. The performance gap between the proposed method and the joint learning baseline decreases as the number of exemplars increases. Fig. 2 (right) shows the results of the methods on Split CIFAR-10. Notably, ours outperforms other compared approaches for all of the exemplar sizes. As the number of stored exemplars increases, the gap between H^2 and

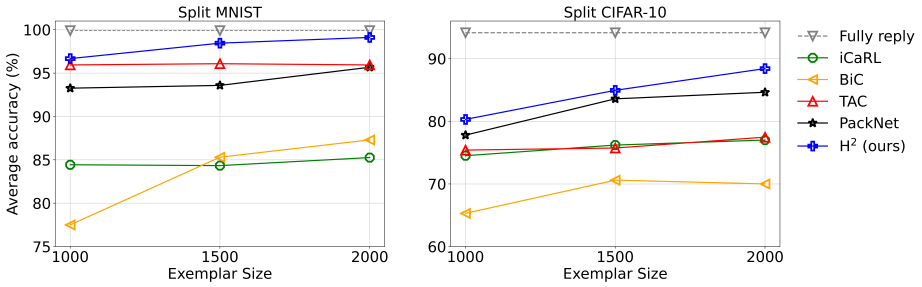

Fig. 2: Class-incremental learning results on the Split MNIST and Split CIFAR-10 datasets under different exemplar sizes.

TAC increases (4.9%, 9.2%, and 10.9% for 1,000, 1,500, and 2,000 exemplars, respectively). Similarly, ours outperforms PackNet equipped with the presented meta classifier for all exemplar sizes.

4.5 Analysis

Ablation study To demonstrate the efficiency of the proposed approach, we conducted an ablation study. We compared ours by removing the search method, the sensitivity measure, and both strategies, i.e., using all parameters [27] under the same learning framework as ours. In addition, we compared ours by removing the contrastive loss to show the effect on training the meta classifier. We also provide the results of the random selection, which chooses the parameters of the previous tasks randomly with the same ratio as the proposed method for a fair comparison. For the experiment, we conducted both task-incremental and class-incremental learning scenarios using Split CIFAR-10. We report the average accuracy of the tasks learned each time step.

Table 5 summarizes the ablation study. The proposed method of searching cooperative tasks and parameters gives higher accuracy than other strategies for both learning scenarios. Random selection performs poorly for most of the tasks in the scenarios, indicating that selectively incorporating helpful parameters from old tasks improves performance and arbitrarily selecting old parameters worsens the performance. The method using all parameters without search and sensitivity shows disappointing results (0.5% and 2.9% margins for task-incremental and class-incremental learning, respectively) compared to ours. This reveals that some parameters influence the current task negatively. We find that applying both approaches is the most promising way and is better than applying the individual method. Note that the performance gap is clearly shown in class-incremental learning that is more challenging than task-incremental learning.

We also show the parameter consumption, FLOPs, and memory requirement of the proposed method (except the meta classifier) in Fig. 3. We compute parameter consumption and FLOPs following the practice of *ShrinkBench*[1]. Since

[1] `https://github.com/jjgo/shrinkbench`

Table 5: Ablation study of the proposed method for Split CIFAR-10.

Method	Task-incremental					Class-incremental				
	T^1	T^2	T^3	T^4	T^5	T^1	T^2	T^3	T^4	T^5
Ours	**98.8**	**96.4**	**96.4**	**97.1**	**97.3**	**98.8**	**91.2**	**87.4**	**86.1**	**80.3**
w/o search	**98.8**	96.1	96.0	96.7	97.0	**98.8**	89.5	84.9	84.5	79.6
w/o sensitivity	**98.8**	96.2	96.1	96.7	96.9	**98.8**	90.0	83.8	82.6	78.7
w/o search and sensitivity [27]	**98.8**	96.1	95.9	96.6	96.8	**98.8**	89.3	83.5	81.1	77.4
w/o contrastive loss			N/A			**98.8**	90.7	86.7	83.2	77.1
Random search	**98.8**	95.1	95.1	95.9	96.3	**98.8**	86.3	80.8	77.9	68.7

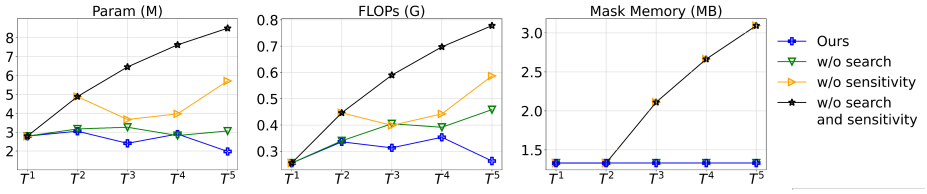

Fig. 3: Ablation study on parameter consumption, FLOPs, and memory of the proposed method for Split CIFAR-10. Best viewed in color.

the random selection method has the same number of parameters as the proposal, it was excluded from the experimental results. The proposed method takes the smallest number of parameters and FLOPs to perform tasks on average under the same threshold. The proposed methods without search and sensitivity consume a larger number of parameters than the proposal. Especially, the amount of parameters consumed to perform a task continues to increase (4.2× higher than ours for T^5) when we exclude the search and sensitivity approach, as shown on the left of the figure. The result shows that ours containing search and sensitivity makes it highly efficient for parameter consumption. In addition, we compute the memory of the mask to perform each task. Since the proposed method and the method without search use a binary mask, the memory for performing each task is constant. For example, when the backbone network has n parameters, the binary mask requires n bits. However, the memory of the mask to perform each task generally continues to increase for the other methods.

Task order To analyze the performance of the proposed method with respect to different task orders, we conducted an additional experiment using the ImageNet-50 (I) and Split CIFAR-10 (C) datasets, respectively. Specifically, the datasets were divided into five tasks as done in Sections 4.3 and 4.4, respectively. In this study, we performed both task- and class-incremental learning scenarios. We produced ten random sequences for each scenario and evaluated the compared methods using the average result. We used 1,000 and 2,000 exemplars for Split CIFAR-10 and ImageNet-50, respectively, in class-incremental learning.

Fig. 4 reports the average accuracies of the tasks trained up to each time step. Overall, the results on ImageNet-50 and Split CIFAR-10 show a similar

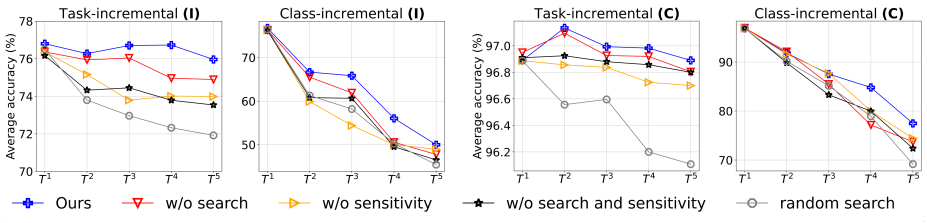

Fig. 4: Ablation study of the proposed method with respect to different orders of tasks for ImageNet-50 (I) and Split CIFAR-10 (C). Best viewed in color.

trend to those in Table 5. In Fig. 4 (left two subfigures), the proposed method outperforms other strategies without search, sensitivity, or both, by large margins of 1.06%, 1.97%, and 2.42%, respectively, in the task-incremental scenario. Similarly, the proposed method outperforms other methods for all time steps on average in the class-incremental scenario. The experiments using Split CIFAR-10 are shown in Fig. 4 (right two subfigures). The results show a similar trend to those on ImageNet-50. Note that compared to the results for task-incremental learning, the class-incremental learning performance degrades significantly due to its difficulty without knowing the task oracle.

5 Conclusions

We have proposed a novel approach to find an optimal architecture that exploits useful knowledge from old tasks in structural allocation-based continual learning. The proposed method simultaneously associates cooperative tasks and explores the optimal architecture by a model search with continuous relaxation. We have also presented element-wise parameter selection from the sensitivity measure incorporated with the model search. The proposed method takes fewer parameters than its competitors while avoiding negative interference. In experiments, we have compared with existing methods for task- and class-incremental learning scenarios using small to large-scale datasets. Experimental results show that the proposed method achieves excellent performance over diverse continual learning competitors while minimizing the computation cost.

Acknowledgement This work was supported in part by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT2002-05, and in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00124, Development of Artificial Intelligence Technology for Self-Improving Competency-Aware Learning Capabilities and No.2021-0-01341, Artificial Intelligence Graduate School Program(Chung-Ang University)).

References

1. Abati, D., Tomczak, J., Blankevoort, T., Calderara, S., Cucchiara, R., Bejnordi, B.E.: Conditional channel gated networks for task-aware continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3931–3940 (2020)
2. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: European Conference on Computer Vision. pp. 144–161. Springer (2018)
3. Aljundi, R., Chakravarty, P., Tuytelaars, T.: Expert gate: Lifelong learning with a network of experts. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3366–3375 (2017)
4. Belouadah, E., Popescu, A., Kanellos, I.: A comprehensive study of class incremental learning algorithms for visual tasks. Neural Networks (2020)
5. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: European Conference on Computer Vision. pp. 556–572. Springer (2018)
6. Chaudhry, A., Marc'Aurelio, R., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. In: 7th International Conference on Learning Representations, ICLR 2019. International Conference on Learning Representations, ICLR (2019)
7. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009)
9. Dong, X., Yang, Y.: Network pruning via transformable architecture search. Advances in Neural Information Processing Systems **32** (2019)
10. Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E.: Podnet: Pooled outputs distillation for small-tasks incremental learning. In: European Conference on Computer Vision. pp. 86–102. Springer (2020)
11. Eitz, M., Hays, J., Alexa, M.: How do humans sketch objects? ACM Transactions on Graphics **31**(4), 1–10 (2012)
12. Farquhar, S., Gal, Y.: Towards robust evaluations of continual learning. arXiv preprint arXiv:1805.09733 (2018)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
14. Hecht-Nielsen, R.: Theory of the backpropagation neural network. In: Neural networks for perception, pp. 65–93. Elsevier (1992)
15. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 831–839 (2019)
16. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. In: 5th International Conference on Learning Representations, ICLR (2017)
17. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences of the United States of America **114**(13), 3521–3526 (2017)

18. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: 2013 IEEE International Conference on Computer Vision Workshops. pp. 554–561. IEEE (2013)
19. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
20. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
21. Lee, S.W., Kim, J.H., Jun, J., Ha, J.W., Zhang, B.T.: Overcoming catastrophic forgetting by incremental moment matching. Advances in Neural Information Processing Systems **30**, 4652–4662 (2017)
22. Li, Z., Hoiem, D.: Learning without forgetting. In: European Conference on Computer Vision. pp. 614–629. Springer (2016)
23. Liu, Y., Schiele, B., Sun, Q.: Adaptive aggregation networks for class-incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2544–2553 (2021)
24. Liu, Y., Su, Y., Liu, A.A., Schiele, B., Sun, Q.: Mnemonics training: Multi-class incremental learning without forgetting. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 12245–12254 (2020)
25. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 6470–6479 (2017)
26. Mallya, A., Davis, D., Lazebnik, S.: Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In: Proceedings of the European Conference on Computer Vision. pp. 67–82 (2018)
27. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7765–7773 (2018)
28. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24, pp. 109–165. Elsevier (1989)
29. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing. pp. 722–729. IEEE (2008)
30. Ostapenko, O., Puscas, M., Klein, T., Jahnichen, P., Nabi, M.: Learning to remember: A synaptic plasticity driven framework for continual learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11321–11329 (2019)
31. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017)
32. Rosenfeld, A., Tsotsos, J.: Incremental learning through deep adaptation. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(3), 651–663 (2018)
33. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)
34. Saleh, B., Elgammal, A.: Large-scale classification of fine-art paintings: Learning the right metric on the right feature. International Journal for Digital Art History (2) (2016)
35. Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R.: Progress & compress: A scalable framework for continual

learning. In: International Conference on Machine Learning. pp. 4528–4537. PMLR (2018)
36. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: International Conference on Machine Learning. pp. 4548–4557. PMLR (2018)
37. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 2994–3003 (2017)
38. Tu, C.H., Wu, C.E., Chen, C.S.: Extending conditional convolution structures for enhancing multitasking continual learning. In: 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). pp. 1605–1610. IEEE (2020)
39. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
40. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 374–382 (2019)
41. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. In: International Conference on Learning Representations (2018)
42. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 3987–3995 (2017)