IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Gating Mechanism in Deep Neural Networks for Resource-Efficient Continual Learning

HYUNDONG JIN[1], **KIMIN YUN**[2], **AND EUNWOO KIM**[1] **(Member, IEEE)**
[1]School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Korea.
[2]2Artificial Intelligence Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon 34129, Korea.

Corresponding author: Eunwoo Kim (eunwoo@cau.ac.kr)

This research was supported in part by the BK21 FOUR (Fostering Outstanding Universities for Research) Program funded by Ministry of Education of Korea (No.I20SS7609062), in part by the Chung-Ang University Graduate Research Scholarship in 2020, and in part by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-00004, Development of Previsional Intelligence based on Long-term Visual Memory Network).

ABSTRACT Catastrophic forgetting is well-known tendency in continual learning of a deep neural network to forget previously learned knowledge when optimizing for sequentially incoming tasks. To address the issue, several methods have been proposed in research on continual learning. However, theses methods cannot preserve the previously learned knowledge when training for a new task. Moreover, these methods are susceptible to negative interference between tasks, which may lead to catastrophic forgetting. It even becomes increasingly severe when there exists a notable gap between the domains of tasks. This paper proposes a novel method of controlling gates to select a subset of parameters learned for old tasks, which are then used to efficiently optimize a new task while avoiding negative interference. The proposed approach executes the subset of old parameters that provides positive responses by evaluating the effect when the old and new parameters are used together. The execution or skipping of old parameters through the gates is based on several responses across the network. We evaluate the proposed method in different continual learning scenarios involving image classification datasets. The proposed method outperforms other competitive methods and requires fewer parameters than the state-of-the-art methods during inference by applying the proposed gating mechanism that selectively involves a set of old parameters that provides positive prior knowledge to newer tasks. Additionally, we further prove the effectiveness of the proposed method through various analyses.

INDEX TERMS Continual learning, resource-efficient learning, task interference, gating mechanism.

I. INTRODUCTION

DEEP neural networks generally access the complete data of tasks when learning multiple tasks [1], [2]. A more challenging scenario of learning multiple tasks, known as continual learning [3]–[13], assumes that a task is observed at a specific time without accessing the data of the previous tasks. When tasks appear sequentially, a deep learning model prioritizes current tasks; however, it forgets the knowledge of previous tasks. This phenomenon is called catastrophic forgetting [3], [14], which represents a major obstacle to the success of continual learning.

The existing research on continual learning primarily addresses the problem of forgetting the knowledge of previous tasks. Recently proposed methods attempt to prevent the forgetting of previous knowledge while exploiting current information. Methods for continual learning can be categorized as: regularization [3]–[5], replay [6]–[8], dynamic architecture [9]–[11] and structural allocation [12], [13] approaches. The regularization-based strategy [3]–[5] identifies important parameters and prevents their update while learning the knowledge of the current task. However, the ability to curb changes in the parameter values is limited, especially when learning a long sequence of tasks. The replay-based strategy [6]–[8] stores a small number of training examples. The stored set is utilized to perform joint training with the set of the current task [6], [7] or seeking key parameters to retain the knowledge of previous tasks [8]. However, because the same examples are used for learning subsequent tasks, overfitting may occur [15]. The dynamic architecture-based strategy [9]–[11] generally introduces new learnable parameters when a current task is observed [9] or the network fails to achieve a predetermined criterion on loss or validation accuracy [16]. However, this approach incurs a higher computational cost than other approaches, which reduces its applicability. The structural allocation strategy [12], [13], [17] assigns task-specific parameters in a fixed network and prevents the update of the previous parameters while training a current task. In the approach proposed in [12], all parameters including the previous parameters are considered in training a current task. However, the indiscriminate use of previous parameters (knowledge) with the new ones may incur negative interference between tasks.

The proposed method is a type of structural allocation strategy. Unlike other kinds of strategies, the structural allocation approach assigns a disjoint set of parameters for a task and prevents the rewrite of previous parameters [12]. In other words, such methods do not update the previous parameter sets and thus do not forget the knowledge of the previous tasks. Despite their notable advantage of protecting the previous knowledge, indiscriminate use of the previous parameter set when learning a new task may negatively affect the optimization of the network. This aspect highlights the importance of associating previous parameters that provide a positive response for the network optimization while skipping other parameters that generate negative interference.

In this study, we establish a novel method to selectively skip previous parameters that negatively interfere with the current task based on the proposed gating mechanism. The proposed method includes a feature extractor consisting of units (disjoint sets of parameters) and multiple classification heads for sequential tasks. For each observed task, the proposed method first allocates parameters in the element-wise manner into disjoint groups through three steps: training, pruning, and retraining. When learning a new task, the gates are controlled to execute or skip the previous sets of parameters.

To this end, we exploit the previous parameters that yield a positive network response to the current task for the gate control. In particular, two main responses to the current task are considered, namely, (i) the effect of high-level feature and (ii) the amount of information in the lower-level features. The high-level features represent the response obtained from the end of the network, i.e., loss, and low-level features correspond to a feature map response from an intermediate layer of the network. By controlling the gates using both types of features, the proposed method effectively skips previous parameters that negatively interfere with the current task. Furthermore, the proposed method does not induce a memory overhead to store data of previous tasks, as in replay-based approaches [6]–[8].

We apply the proposed method to a range of continual learning scenarios using the CIFAR-100 [18] and ImageNet-50 [19] datasets. Experiments are conducted in which the number of tasks and size of the backbone model are varied for each dataset. Furthermore, we obtain results for additional scenarios involving different semantic information between tasks. Experiment results show that the proposed method outperforms existing continual learning approaches regardless of the similarity in the task domains. The contributions of this work can be summarized as follows:

- We propose a novel method to explore a task-specific network structure by controlling the gate of each unit guided by the high- and low-level responses from the network.
- The proposed method can effectively detect previously learned parameters that are helpful in accomplishing the current task from the responses.
- Experimental results show that the proposed method not only minimizes the harmful interference between tasks but also requires fewer parameters to perform the task.

We briefly introduce related works in continual learning in Section II. In Section III, we describe the proposed method with the gating mechanism. In Section IV, we show experimental results of the proposal with other compared approaches. Finally, we discuss the conclusion of this work in Section V.

II. RELATED WORK

This section introduces four different types of strategies in continual learning and gated neural network methods.

A. CONTINUAL LEARNING

Regularization strategies [3]–[5], [20] identify key older parameters that are strongly linked to a new task. The importance of the parameters is determined through the sensitivity measure [3], change in loss [4], or derivative of the output of the network [5]. However, the consolidation of important parameters weakens the learning efficiency when addressing a long sequence of tasks [21].

Replay strategies [6]–[8], [22] store a small set of training examples to replay when training on a current task. This line of methods employs the stored set to jointly train the network with current data [6], [7], [22]. Another approach [8] alternatively used the stored set for gradient estimation with important parameters of old tasks. However, the strategy incurs an additional memory overhead to store the examples. In addition, overfitting may occur due to repeated training with a small fixed number of replay data.

Dynamic expansion strategies [9]–[11], [16] increase the size of the model through a pre-determined criterion for loss or accuracy. [9] and [11] proposed new dynamic models to accommodate each incoming task. [10] and [16] attempted to expand the network size by adding new learnable parameters. However, the dynamic expansion strategy involves a notable limitation as it expands the size of the network whenever a pre-determined criterion is not satisfied. Consequently, the computational cost increases in proportion to the network size, rendering it difficult to apply this approach in practical problems.

Structural allocation strategies [12], [13], [17], [23] generally allocate the parameter sets in a single feature extractor through pruning [12], learnable binary masks [17] or the attention mechanism with gradient descent [13] for each task. To maintain previous knowledge, the approach updates the current set of parameters [12], [17] while preventing the previous ones from being updated or restricts the update of critical parameters of the previous tasks. However, if all the previous parameters are used to perform optimization for the current task [12], negative interference among tasks may be introduced, which may be especially severe for tasks of different domains.

B. GATED NETWORK

In gated networks [24]–[27], gates are controlled to identify the suitable computational path of the network for a given task. However, to search for a computational path or its corresponding subnetwork, additional networks or modules are required [24]–[27]. In addition, the approach may execute or skip the entire residual block [25], [26] or channels [24], [27] during inference. Moreover, these methods have been established for scenarios in which all data are simultaneously accessed. In contrast, the proposed method is applied to a more challenging continual learning scenario in which tasks appear sequentially. Because the domains of tasks are unknown in real-world scenarios, we attempt to determine an optimal computational path in a single backbone architecture for each task. In addition, the proposed method does not require additional search modules for the gating mechanism. Instead, the gates are controlled considering the low- and high-level information of the network collected within the network.

III. THE PROPOSED GATING MECHANISM IN CONTINUAL LEARNING

In this section, we describe the proposed gating strategy in continual learning. Section III-A introduces the framework of the proposed method and issues of existing works. Section III-B and III-C describe two responses for the proposed gating mechanism, respectively. We provide mathematical symbols used in this work in Table 1.

TABLE 1. Mathematical symbols used in this work.

Symbol	Meaning
T^i	i^{th} task
$f(\cdot)$	feature extractor
U_l	l^{th} unit in feature extractor
w^i	parameters of the i^{th} task-specific classifier
c_{w^i}	i^{th} task-specific classifier parameterized with w^i
θ^i, θ^i_l	i^{th} task-specific parameter in $f(\cdot)$ and U_l
$\tilde{\theta}^i, \tilde{\theta}^i_l$	i^{th} task-specific parameter in $f(\cdot)$ and U_l after retraining
$I_l^{(i,t)}$	average feature map information in the l^{th} unit when U_l is parameterized with $\tilde{\theta}^i_l \cup \theta^t_l$
$I_l^{(t)}$	average feature map information in the l^{th} unit when U_l is parameterized with θ^t_l
$L_l^{(i,t)}$	loss when U_l is parameterized with $\tilde{\theta}^i_l \cup \theta^t_l$
$L_l^{(t)}$	loss when U_l is parameterized with θ^t_l
$\hat{g}^i_l$	gate that controls $\tilde{\theta}^i_l$ in U_l

A. FRAMEWORK

The proposed method, based on the gating mechanism, aims to skip previously learned parameters that may cause negative task interference. The proposed method controls the gates using two responses of the network, which are collected from the intermediate part and end of the network. The framework follows the structural allocation strategy, in which each task-specific parameter set does not overlap those of other tasks.

The problem of interest pertains to the learning of sequentially incoming tasks $\mathcal{T} = \{T^1, T^2, \cdots\}$, where each task $T^i = \{\mathcal{X}^i, \mathcal{Y}^i\}$ contains data $\mathcal{X}^i = \{x^i_j\}^n_{j=1}$ and the corresponding labels $\mathcal{Y}^i = \{y^i_j\}^n_{j=1}$. The base network has a single feature extractor $f(\cdot)$ and task-specific classifiers $c_{w^i}(\cdot)$ parameterized with w^i. The feature extractor $f(\cdot)$ consists of L units, wherein a unit can be a layer or a or a group of layers. Specifically, we define the feature extractor with L units as

$$f(x) \equiv (U_L \circ \cdots \circ U_1)(x), \text{where } 1 \leq l \leq L \tag{1}$$

and $\circ$ is the function composition operator. Similarly, we define the output of the l^{th} intermediate unit as

$$f_l(x) \equiv (U_l \circ \cdots \circ U_1)(x). \tag{2}$$

Initially, all parameters in $f(\cdot)$ are initialized to α^0 such that $\alpha^0 = \bigcup_{l=1}^{L} \alpha^0_l$. When the first task is observed, α^0 is assigned to the first task as θ^1. T^1 is trained by updating θ^1 in the feature extractor f and w^1 in the classifier c_{w^1}. After training T^1, redundant parameters in θ^1 are removed from the feature extractor to provide space for the subsequent task. Discarded parameters are initialized as α^1, whereas the survived parameters are fine-tuned to produce $\tilde{\theta}^1$. Finally, the parameters in the feature extractor are composed of $\tilde{\theta}^1$ to perform the first task and α^1 to be allocated for the next task. For example, parameter sets of the l^{th} unit U_l becomes $[\tilde{\theta}^1_l, \alpha^1_l]$. When the i^{th} task is observed, we allocate initialized parameters α^{i-1} as θ^i, where U_l contains $[\tilde{\theta}^1_l, \cdots, \tilde{\theta}^{i-1}_l, \theta^i_l]$. From this, one can predict $\hat{y}^i$ for T^i using all previous parameters [12]:

$$\hat{y}^i = (c_{w^i} \circ f)(x^i), \text{ where } f(x^i) \equiv (U_L \circ \cdots \circ U_1)(x^i). \tag{3}$$

The objective function for the i^{th} task is

$$\underset{\theta^i, w^i}{\text{argmin}} \ \mathcal{L}(y^i, \hat{y}^i), \text{ where } \mathcal{L}(y^i, \hat{y}^i) = -\sum_{j=1}^{n} y^i \log \hat{y}^i. \tag{4}$$

This objective updates the newly allocated parameters θ^i and w^i, while the other parameters are maintained constant. After learning T^i, the sparse parameter set $\tilde{\theta}^i$ is obtained through pruning and retraining. Similarly, future tasks are treated with the train-prune-retrain strategy. However, the approach using all parameters of previous tasks [12] when learning the current task is exposed to potential risks of negative

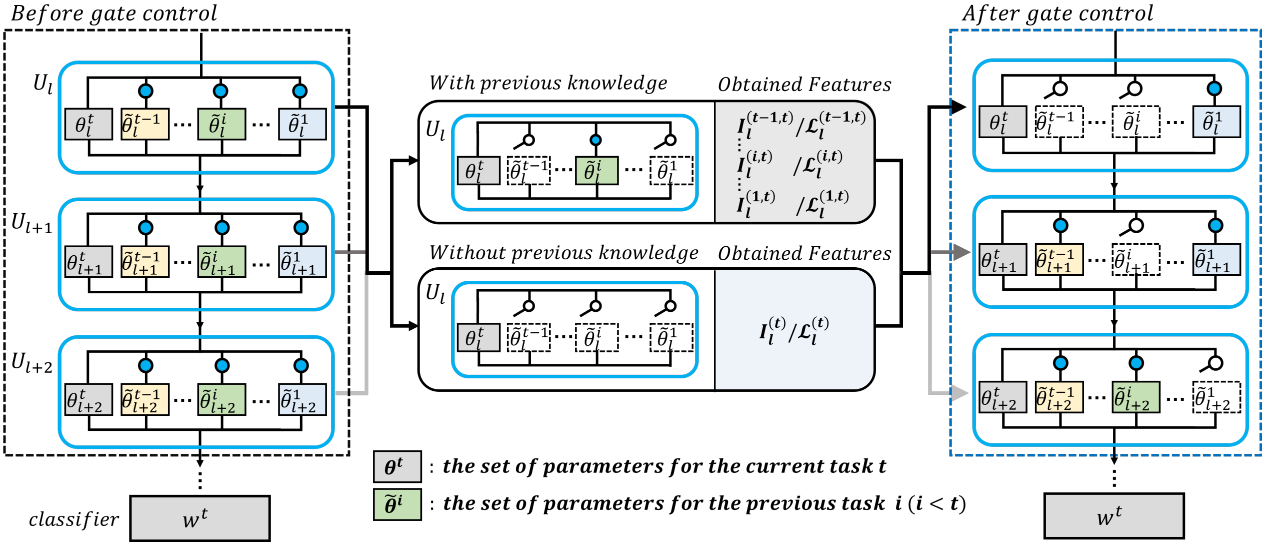

Before gate control
With previous knowledge
Obtained Features
Without previous knowledge
Obtained Features
After gate control
U_l
U_{l+1}
U_{l+2}
θ_l^t
$\tilde{\theta}_l^{t-1}$
$\tilde{\theta}_l^i$
$\tilde{\theta}_l^1$
$I_l^{(t-1,t)}/\mathcal{L}_l^{(t-1,t)}$
$I_l^{(i,t)}/\mathcal{L}_l^{(i,t)}$
$I_l^{(1,t)}/\mathcal{L}_l^{(1,t)}$
$I_l^{(t)}/\mathcal{L}_l^{(t)}$
classifier
w^t
θ^t : the set of parameters for the current task t
$\tilde{\theta}^i$: the set of parameters for the previous task i $(i < t)$

FIGURE 1. Graphical illustration of the proposed method representing the states of three consecutive intermediate units in the network.

interference between tasks. When the domains of the current and previous tasks are different, harmful information from the previous tasks may disturb the learning of the current task. To mitigate this phenomenon, the proposed method provides a parameter selection approach by introducing and controlling the gates.

Figure 1 represents conceptual illustration of the proposed gating mechanism. The figure on the left represents the network before controlling the gates. The operations of the unit l with and without the use of the previous knowledge are presented in the middle. In the units, we obtain information $I^{(i,t)}$ by executing the i^{th} previous set of parameters $\tilde{\theta}_l^i$ and t^{th} new set of parameters θ_l^t. Furthermore, we obtain loss $\mathcal{L}_l^{(i,t)}$ by further executing c_{w^t}. Similarly, we obtain information $I_l^{(t)}$ and loss $\mathcal{L}_l^{(t)}$ using θ_l^t (without $\tilde{\theta}_l^i$). The gated network of the l^{th} unit is shown in the figure on the right.

B. LOW-LEVEL RESPONSE

To select previous parameters that are helpful in accomplishing the current task, we first consider the outputs in the intermediate layers of the network. Specifically, we use low-level features of the network as feature maps and control the gates based on the information of feature maps generated by the previous sets of parameters.

The proposed method learns a new task by using the parameters that generate a feature map with rich information when the new task is given. To measure the relative amount of information with respect to different usages of parameters, the proposed method employs singular value decomposition (SVD) [28]. The singular values of feature maps can reflect the amount of information [29]. We denote the feature maps obtained using both current and the i^{th} previous parameter set and only the current parameter set in the l^{th} unit as $h_l^{(i,t)}(x^t)$ $(i = 1, 2, \cdots, t-1)$ and $h_l^{(t)}(x^t)$, respectively, where t is the index of the current task. The feature map $h_l^{(t)}(x^t)$ in the unit l can be decomposed through SVD as

$$\begin{aligned} h_l^{(t)}(x^t) &= (U_l \circ f_{l-1})(x^t), \\ &= \sum_{k=1}^{k'} u_k s_k v_k^T + \sum_{j=k'+1}^{K} u_j s_j v_j^T, \end{aligned} \tag{5}$$

where u_k and v_k denote the left and right singular vectors, respectively, and s_k is the singular value of $h_l^{(t)}(x^t)$. Similarly, the feature map $h_l^{(i,t)}(x^t)$ is obtained similar to the above equation, where U_l is parameterized with $\tilde{\theta}_l^i \cup \theta_l^t$. Note that $h_l^{(t)}(x^t)$ (or $h_l^{(i,t)}(x^t)$) can be divided into two terms by the k^{th} rank, where the left (low-rank) term $\sum_{k=1}^{k'} u_k s_k v_k^T$ contains a considerable amount of information and the right (high-rank) term $\sum_{j=k'+1}^{K} u_j s_j v_j^T$ contains relatively insignificant information. Consequently, the amount of feature information is dominant in the left term. We use the singular values that reflect information of feature maps as

$$S(h_l^{(t)}(x_j^t)) = \sum_{k=1}^{k'} s_k, \tag{6}$$

where s_k is a k^{th} singular value of feature map and $S(\cdot)$ is sum of singular values of a feature map for input image x_j^t. $I_l^{(t)}$ and $I_l^{(i,t)}$ represent the average feature map information in the l^{th} unit, calculated as

$$I_l^{(i,t)} = \frac{1}{n} \sum_{j=1}^{n} S(h_l^{(i,t)}(x_j^t)) \tag{7}$$

$$I_l^{(t)} = \frac{1}{n} \sum_{j=1}^{n} S(h_l^{(t)}(x_j^t)). \tag{8}$$

If $I_l^{(i,t)} > I_l^{(t)}$, the i^{th} parameter set provides richer positive information for the current task T^t. Considering this aspect, we define gate control g_l^i that applies the sets of previous parameters $\tilde{\theta}_l^i$ in the l^{th} unit as follows:

$$g_l^i = \begin{cases} 1 & \text{if } I_l^{(i,t)} > I_l^{(t)} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The feature extractor $f(x^t)$ incorporated with the gate control is denoted as

$$\begin{aligned} f(x^t) &= (U_L \circ \cdots \circ U_l \circ \cdots \circ U_1)(x^t), \\ \text{where } U_l &\leftarrow \left(\bigcup_{i=1}^{t-1} g_l^i \cdot \tilde{\theta}_l^i \right) \cup \theta_l^t. \end{aligned} \tag{10}$$

C. HIGH-LEVEL RESPONSE

The proposed method takes the loss as another response for the gate control and explores the sets of previous parameters that incur a small loss for a new task. The loss is employed as a measure to find relevant tasks [30], [31]. We seek the sets of previous parameters that further minimize the loss when used with the current parameter set. We denote the loss incurred when using the current parameters and both current and the i^{th} previous set of parameters in the unit l as $\mathcal{L}_l^{(t)}$ and $\mathcal{L}_l^{(i,t)}$, respectively. Formally, we can represent $\mathcal{L}_l^{(i,t)}$ as

$$\mathcal{L}_l^{(i,t)} = -\sum_{j=1}^{N} y^t \log(c_{w^t} \circ f)(x^t). \tag{11}$$

Likewise, $\mathcal{L}_l^{(t)}$ is obtained in a manner similar to the above equation, where U_l is parameterized with θ^t. Losses $\mathcal{L}_l^{(t)}$ and $\mathcal{L}_l^{(i,t)}$ are considered with the information in the feature maps $I_l^{(t)}$ and $I_l^{(i,t)}$ to control the gates associated with the previous parameters.

Finally, the gate control, an improved version from Eq. (9), for the i^{th} previous parameter set is defined as

$$\hat{g}_l^i = \begin{cases} 1 & \text{if } \dfrac{I_l^{(i,t)}}{\mathcal{L}_l^{(i,t)}} > \dfrac{I_l^{(t)}}{\mathcal{L}_l^{(t)}} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

The gate control is implemented by minimizing the loss of the current task while maximizing the information. After the gate control, the final network can be expressed as

$$\begin{aligned} f(x^t) &= (U_L \circ \cdots \circ U_l \circ \cdots \circ U_1)(x^t), \\ \text{where } U_l &\leftarrow \left(\bigcup_{i=1}^{t-1} \hat{g}_l^i \cdot \tilde{\theta}_l^i \right) \cup \theta_l^t. \end{aligned} \tag{13}$$

In summary, the proposed method mitigates the negative interference between tasks by the gate control that enriches the information of the feature maps while minimizing the loss for the new task.

IV. EXPERIMENTS

In this section, we compare the proposed method with other continual learning methods. The dataset used in the experiment and implementation details are discussed in Sections IV-A and IV-B, respectively. Section IV-C and IV-D show the results using ImageNet-50 and CIFAR-100, respectively. In Section IV-E, we analyze the proposed gate mechanism, including an ablation study.

TABLE 2. Datasets used in this work.

Dataset	# Train	# Test	# Class
ImageNet-50 [19]	65,000	2,500	50
CIFAR-100 [18]	50,000	10,000	100

TABLE 3. Notations for networks, the number of tasks, and notations for division types used in the experiments.

Network type	# Tasks	Division type
N / *W*	5 / 10 / 20 /25	*L* / *R* / *S*

A. DATASETS

We applied the proposal to various task-incremental scenarios in continual learning. The datasets used in the experiments included ImageNet-50 [19] and CIFAR-100 [18]. Following the method specified in [22], ImageNet-50 was resized to a resolution of 32×32 by randomly selecting 50 subclasses from the original ImageNet-1K [19] dataset. The CIFAR-100 dataset [18] was split in the order of the labels provided. In particular, we composed a scenario by splitting the dataset into 20 tasks (based on super-classes), each of which involved five classes. The characteristics of the datasets used in the experiments are summarized in Table 2.

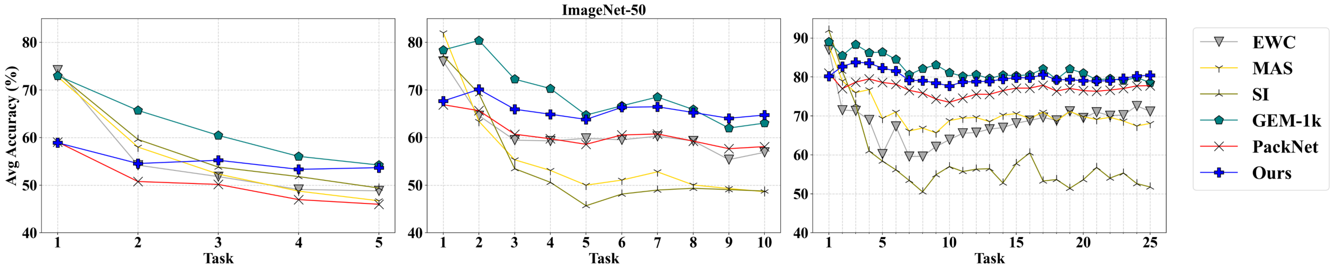
B. IMPLEMENTATION DETAILS

We used the ResNet-20 [32] and WideResNet-28-2 (WRN-28-2) [33] backbone networks. The proposed and existing methods have a classification head (fully connected layer) for each task. We defined each task as learning a set of classes at a time. Each experiment was conducted by dividing the dataset into sequential tasks according to the provided labels. Three criteria were considered to divide the dataset: by label order (L), random order (R), and super-class order (S). The random order strategy shuffled the labels and divided them in a specified order to produce sequential tasks. Table 3 represents the information for different scenarios. The experiment based on ResNet-20 (WRN-28-2) for 20 tasks constructed by the random order was denoted as $N-R-20$ $(W-R-20)$. To show the effectiveness of the proposed method, we compare it with EWC [3], SI [4], and MAS [5] in the regularization strategy, GEM [8] in the replay strategy, and PackNet [12] in the structural allocation strategy.

In experiment, we compared ours with a joint training method (Joint) using all the data and a fine-tuning method (Fine-tune) that is trained using data from only the current tasks. In addition, we compared the proposed approach with a naïve version of the replay method (Replay) [34], which randomly stores a part of the previous data and performs joint training with the current data.

TABLE 4. Continual learning results of the compared methods on ImageNet-50 with respect to average accuracy (%) using ResNet-20 (N) and WRN-28-2 (W).

ImageNet-50	Joint	Fine-tune	Replay-2K	EWC	SI	MAS	GEM-1K	PackNet	Ours
$N-R-5$	73.50	23.38	56.50	48.10	48.83	45.94	**54.08**	46.04	<u>53.67</u>
$N-R-10$	82.34	27.56	66.62	55.70	51.65	49.48	<u>63.22</u>	58.06	**64.73**
$N-R-25$	91.77	52.42	82.64	73.06	53.30	68.08	<u>78.56</u>	77.76	**80.44**
$W-R-5$	76.88(+3.38)	24.83(+1.45)	61.0(+4.5)	29.6(-18.5)	53.06(-5.77)	48.90(+2.96)	56.86(+2.78)	<u>54.36</u>(+8.32)	**58.72**(+5.05)
$W-R-10$	84.37(+2.03)	27.78(+0.22)	69.8(+3.18)	43.04(-12.66)	48.14(-3.51)	51.02(+1.54)	<u>64.94</u>(+1.72)	62.04(+3.98)	**65.66**(+0.93)
$W-R-25$	92.87(+1.1)	51.65(-0.77)	83.03(+0.64)	50.02(-23.04)	52.56(-0.74)	70.33(+2.25)	<u>78.64</u>(+0.08)	78.46(+0.88)	**81.26**(+0.82)



(a) 5 increments (b) 10 increments (c) 25 increments

FIGURE 2. Average accuracy on the ImageNet-50 dataset using the ResNet-20 backbone. The subfigures (a), (b), and (c) show the results of the compared methods for 5, 10 and 25 sequential tasks, respectively.

All datasets in the experiment were subjected to random horizontal flip and random cropping augmentation during training. The proposed method trained the network until convergence with an initial learning rate of 0.01 and stochastic gradient descent with a momentum of 0.9. The learning rate was multiplied by 0.1 at the 50 and 75 epochs. Note that PackNet and the proposed method discarded approximately 75% of the parameters based on the absolute values of the parameters during pruning in each task. We controlled the gates for the last three units of the network in the main scenarios, with each unit corresponding to a ResNet block [32]. The results for different numbers of units to be controlled in the ablation study were presented. All experiments were conducted using the PyTorch library [35] and NVIDIA 2080Ti GPU.

C. IMAGENET-50 RESULTS

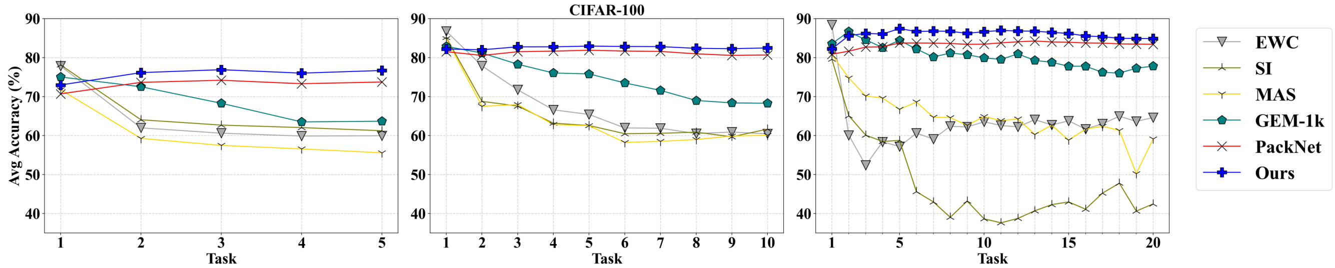
We applied the proposed method to ImageNet-50 which is divided into 5, 10 or 25 sequential tasks, respectively. The classes in the ImageNet-50 dataset were randomly selected from the original dataset [19]. We trained all the methods until convergence and derived the average accuracy of all tasks after training on the last task.

Table 4 (top) summarizes the results obtained using ResNet-20 [32] on ImageNet-50. Note that best and second best results are boldfaced and underlined, respectively. Overall, the average accuracy increases as the number of tasks increases because the number of classes in each task decreases. The fine-tuning method exhibits an inferior performance with average accuracies of 25% and 50%, respectively, for cases involving 10 and 25 tasks. This finding highlights the importance of retaining the helpful knowledge of previous tasks. For ImageNet-50 divided into five tasks, the GEM-1K and Replay-2K methods show slightly higher performance than other continual learning strategies. However, these methods require additional memory to store examples of previous tasks. The regularization strategies, EWC and SI, outperform PackNet by 2.06% and 2.79%, respectively; however, their performance is inferior to the proposed method (5.57% and 4.84% lower, respectively). The performance of the proposed method is the most similar to joint training and higher than the other competitors. Specifically, the proposed method achieves accuracies that are 1.51% and 1.88% higher than those attained using GEM-1K in cases involving 10 and 25 tasks, respectively. Furthermore, the proposal outperforms PackNet [12] when all previous parameter sets are adopted regardless of the number of tasks (performance enhancement of 7.63%, 6.67% and 1.88% in scenarios involving 5, 10, and 25 incremental tasks, respectively).

Table 4 (bottom) presents the average accuracy of ImageNet-50 using WRN-28-2, with the difference in the average accuracies between WRN-28-2 and ResNet-20 presented in parentheses. The performance of the methods is enhanced as the size of the network increased. However, the regularization methods, EWC and SI, achieve a lower accuracy when using the larger-size network. A similar trend has been reported in [15]. In contrast to EWC and SI, the performance of MAS is enhanced when WRN-28-2 is implemented. The replay approach, GEM-1K, shows the most competitive performance against the proposed method. Moreover, this approach exhibits a consistent performance improvement regardless of the number of tasks. The structural allocation strategy, PackNet and ours, improve performance as the larger-scale network is used. The proposed

TABLE 5. Continual learning results of the compared methods on CIFAR-100 with respect to average accuracy (%) using ResNet-20 (N) and WRN-28-2 (W).

CIFAR-100	Joint	Fine-tune	Replay-2K	EWC	SI	MAS	GEM-1K	PackNet	Ours
$N-L-5$	83.34	22.78	68.44	61.64	61.67	57.59	68.46	73.77	**75.76**
$N-L-10$	89.47	22.02	75.53	63.64	60.35	59.36	70.06	82.42	**82.73**
$N-L-20$	94.05	24.64	82.72	67.11	44.02	58.26	77.17	84.71	**85.9**
$W-L-5$	87.09(+3.75)	25.40(+2.61)	73.23(+4.79)	62.48(+0.83)	66.58(+4.9)	66.29(+8.7)	62.2(+6.25)	79.71(+5.93)	**81.07**(+5.3)
$W-L-10$	91.72(+2.25)	22.19(+0.17)	79.26(+3.73)	65.83(+2.18)	61.34(+0.99)	61.31(+1.95)	72.16(-2.09)	81.81(-0.6)	**86.22**(+3.48)
$W-L-20$	95.15(+1.10)	24.28(-0.35)	79.55(+3.17)	40.25(-26.86)	32.35(-11.67)	59.69(+1.42)	78.58(-1.4)	81.91(-2.79)	**87.78**(+1.87)



(a) 5 increments (b) 10 increments (c) 20 increments

FIGURE 3. Average accuracy on the CIFAR-100 dataset using the ResNet-20 network. The subfigures (a), (b), and (c) show the results of the compared methods for 5, 10 and 20 sequential tasks, respectively.

method outperforms PackNet by 4.36%, 3.62%, and 2.8% when the number of tasks is 5, 10, and 25, respectively. The results indicate that the negative interference between tasks can be reduced through the proposed gate mechanism for networks of different sizes.

In addition, we present the average accuracies associated with using ResNet-20 in Figure 2. The accuracy on the right is the same as the values presented in Table 4 as the average accuracy for all tasks is considered. The accuracies of the proposed method and PackNet are slightly lower than those of the other methods after learning on the first task. Unlike other methods that perform the first task using all network parameters, the structural allocation methods show a marginal decrease in the accuracy because certain parameters are pruned for the subsequent tasks. The proposed method can retain the previous knowledge and learns the current task using only the previous parameters that provide positive responses, thereby outperforming PackNet.

D. CIFAR-100 RESULTS

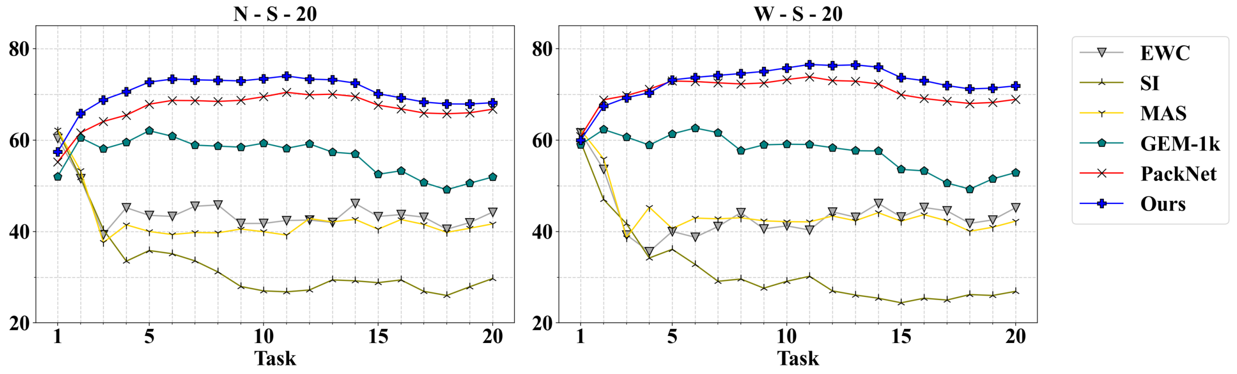
We validated the proposed method for the CIFAR-100 dataset [18]. Specifically, experiments were conducted with 5, 10, and 20 sequences of tasks on the ResNet-20 network [32]. We divided CIFAR-100 into tasks by the label order.

Table 5 (top) summarizes the results obtained using the compared approaches. We report the final accuracy which is obtained by averaging the accuracies for all tasks. Best and second best results are boldfaced and underlined, respectively. Similar to the previous experiments, the average accuracy increases as the number of tasks increases for most continual learning methods. GEM-1K, which exhibits the most competitive performance on ImageNet-50, performs lower than the structural allocation methods on CIFAR-100. EWC performs lower than SI in the sequence of five tasks but better than that for 10 tasks. MAS gives an unsatisfying performance, and its accuracies remain unchanged for different numbers of tasks. The performance of another regularization method, SI, deteriorates as the number of tasks increases. The structural allocation approaches outperform other regularization and replay methods. The proposed method consistently outperforms PackNet, with margins of 1.99%, 0.31%, and 1.19% for 5, 10, and 20 tasks, respectively. Notably, the proposed method also outperforms Replay-2K on CIFAR-100 even if it does not perform better than the competitor on ImageNet-50. The average accuracy pertaining to ResNet-20 is presented in Figure 3. The results at the rightmost point of each figure are identical to the results presented in Table 5 (top).

Table 5 (bottom) lists the average accuracies of CIFAR-100 using WRN-28-2, with the difference in the average accuracies between WRN-28-2 and ResNet-20 presented in parentheses. The performance of the replay and structural allocation methods is improved with the larger network. In contrast, the regularization methods do not show satisfactory performance for WRN-28-2 as they forget the previous knowledge. The replay approach, GEM-1K, shows better performance than the regularization method in most cases. However, GEM-1K achieves inferior results as the number of tasks increases (10 and 20 tasks), contrary to the case of the ImageNet-50 dataset. The structural allocation strategy, PackNet, outperforms other approaches but performs lower than ours by 1.36%, 4.41%, and 5.87% performance gap for 5, 10, and 20 tasks, respectively.

TABLE 6. Continual learning results of the compared methods with respect to average accuracy (%) on CIFAR-100 split by the super-class order.

CIFAR-100	Joint	Fine-tune	Replay-2K	EWC	SI	MAS	GEM-1K	PackNet	Ours
$N-R-5$	84.20(0.86)	23.89(1.11)	68.13(0.31)	60.75(0.89)	61.85(0.17)	56.68(1.10)	64.10(1.89)	<u>75.60</u>(1.82)	**76.56**(0.79)
$N-R-10$	89.12(0.35)	20.12(1.9)	74.43(1.1)	61.43(2.21)	59.12(1.41)	57.99(1.36)	69.29(0.76)	<u>78.83</u>(3.68)	**82.12**(0.33)
$N-R-20$	93.12(0.93)	25.89(1.25)	82.73(0.01)	64.90(2.21)	40.33(3.69)	57.58(0.67)	81.08(3.9)	<u>84.08</u>(0.62)	**85.24**(0.66)
$W-R-5$	87.15(0.06)	27.02(1.62)	73.34(1.11)	61.63(0.84)	67.30(0.71)	67.30(1.01)	62.0(0.2)	<u>77.3</u>(2.4)	**81.8**(0.73)
$W-R-10$	91.19(0.53)	22.20(0.01)	78.24(1.02)	62.26(3.57)	61.34(0.0)	59.90(1.41)	72.17(0.01)	<u>80.64</u>(1.17)	**85.43**(0.78)
$W-R-20$	94.96(0.19)	24.62(0.34)	83.17(3.62)	35.14(5.21)	30.29(2.06)	60.95(1.26)	77.90(0.67)	<u>82.76</u>(0.85)	**87.67**(0.11)
$N-S-20$	79.51(14.54)	24.13(0.51)	59.65(23.07)	45.20(21.91)	31.45(12.57)	42.20(16.06)	50.89(26.28)	<u>66.74</u>(17.97)	**68.16**(17.74)
$W-S-20$	81.79(13.36)	25.23(0.95)	61.64(17.91)	46.79(6.54)	25.66(6.69)	43.15(16.54)	52.79(25.79)	<u>68.89</u>(13.02)	**71.81**(15.91)



(a) ResNet-20

(b) WRN-28-2

FIGURE 4. Average accuracy on CIFAR-100 which is divided by the labels in the super-class order. The subfigures (a) and (b) show the results using the ResNet-20 and WRN-28-2 networks, respectively.

E. ANALYSIS

In the subsection, we discuss the effect of different division strategies for the proposed method (IV-E1), the parameter consumption of the network (IV-E2), negative interference (IV-E3),the effect of the number of gating units (IV-E4), and the ablation study of the proposal (IV-E5).

1) Effect of different division strategies

We analyzed the influence of different division strategies for the CIFAR-100 dataset on the considered methods. We report the final average accuracy of the dataset that is divided by the random label order to produce sequential tasks. Furthermore, we present the accuracy difference between random division and division by the label order.

Table 6 (top) reports the final average accuracy using ResNet-20. The final accuracy is obtained by averaging the accuracies for all tasks. Best and second best results are indicated in bold font and underline, respectively. The accuracy of the regularization strategy is lower than that of the other strategies. The replay strategy, GEM-1K, shows higher average accuracy than the regularization strategy. In particular, the improvement is greater than other methods when the number of tasks increases from 10 to 20. However, this approach is less accurate than the structural allocation strategy regardless of the number of tasks. The structural allocation methods, PackNet and the proposed, consistently achieve a higher average accuracy than other strategies for all numbers of tasks. Similar results and trends are observed in the middle of the table when the large-scale network, WRN-28-2 is implemented. Table 6 (bottom) presents the results for the CIFAR-100 split by the super-class order. The trend of the results is similar to that of the case in which CIFAR-100 is split by the label order, as shown in Table 5. The proposed method outperforms other continual learning approaches by a larger margin on average. Figure 4 reports the average accuracy in the tasks of the CIFAR-100 split by the super-class order.

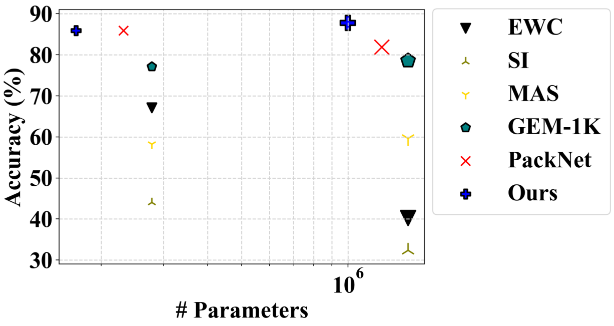


FIGURE 5. Results of the methods with respect to accuracy and the number of required parameters using ResNet-20 and WRN-28-2 (represented by small and large plots, respectively).

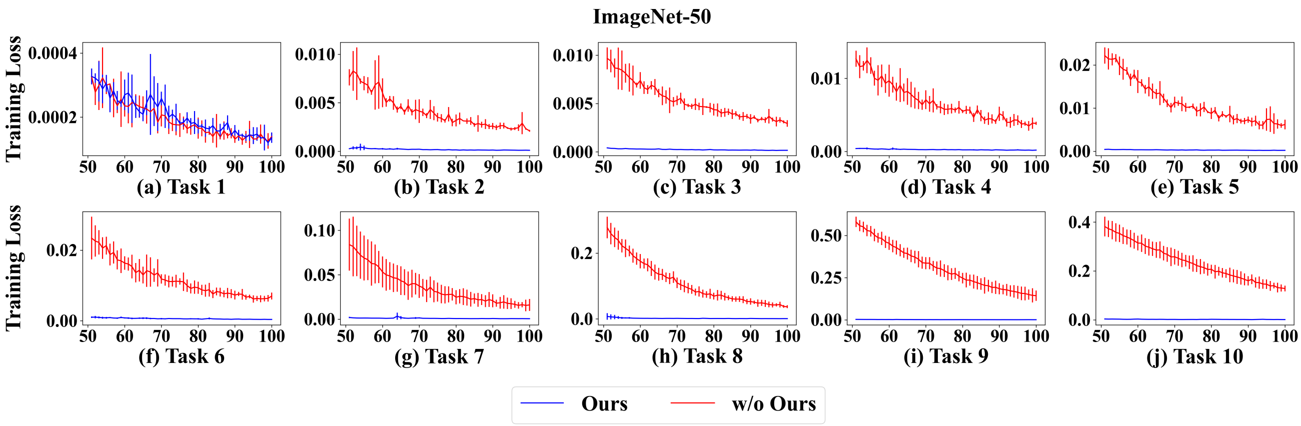


FIGURE 6. Loss behavior when learning each task using the proposed method with and without the gating mechanism.

2) Parameter consumption

We analyzed the number of parameters and the corresponding accuracy on CIFAR-100. In particular, we split the CIFAR-100 dataset into 10 tasks by label order. Figure 5 shows the average parameter consumption in each task and the corresponding average accuracy. The strategies except the structural allocation strategy use all parameters of the backbone network to perform the tasks. The structural allocation methods, PackNet and the proposed method, use fewer parameters than other approaches owing to the use of the pruning step. Even the proposed method performs better than PackNet with fewer parameters as shown in Figure 5.

3) Negative interference

We implicitly measure the negative interference among tasks. We compare the loss and standard deviation of the proposed method with and without the presented gate control mechanism. We used ResNet-20 for ImageNet-50 divided into 10 tasks with random label order. Figure 6 reports the training loss of each task. The loss of the proposed method with the gating mechanism is similar to that without it for the first task. The training loss of the proposal for the subsequent tasks is noticeably smaller and more stable than that without the gating mechanism. This shows that the proposed gate control mechanism selectively chooses the parameters of previous tasks that negatively affect the optimization of the current task.

4) Number of gating units

We investigated the performance of the proposed method under different numbers of gating units. Specifically, we applied the proposed gating module to the last three to nine units in ResNet-20, with a total of nine units. CIFAR-100 was divided into 10 tasks according to the label order. Figure 7 shows the results of the experiment. From the figure, we can observe that controlling a large number of units deteriorated performance. Controlling seven to nine units corresponds to an inferior performance than that in the case of controlling three to five units. This finding indicates that the application of a gate to deeper units representing better task-specific features is more effective and can enhance the performance.

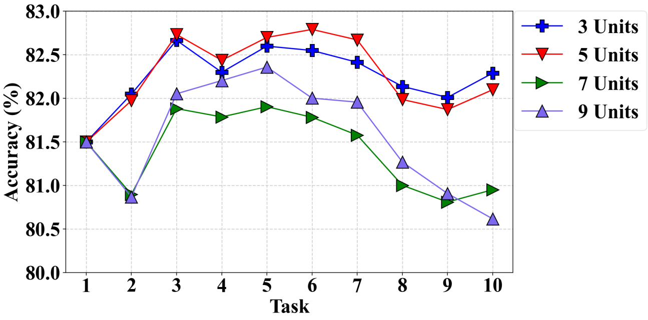


FIGURE 7. Average accuracy of the proposed method under different strategies of the gating module in ResNet-20.

5) Ablation study

An ablation study for the gate control was conducted. The study was performed under the same experimental setup as in the previous experiment. We compared the proposed method without considering each response (intermediate feature map or loss) or both responses, as described in the method section. As shown in Figure 8, the method without the loss response exhibits the lowest performance. The proposed approach without the intermediate responses also shows unsatisfying performance. In contrast, when both responses are used, the proposed approach outperforms all other methods, indicating that the two responses work complementarily to minimize the negative interference between tasks.

V. CONCLUSION

In this work, we have addressed the catastrophic forgetting issue in continual learning that prevents the efficient optimization of a deep neural network for sequential tasks. To alleviate the issue, we have established a novel method of selecting units with the gate control in structural allocation based continual learning. The proposed gated network employs the helpful previous parameters for the current task

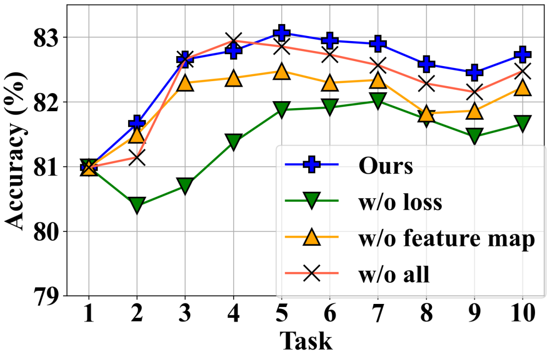


FIGURE 8. Average accuracy of the proposed method under different usages of the proposed responses for the gate control in ResNet-20.

using two responses from the intermediate layer end of the network. By selectively using the helpful parameters learned from the previous tasks, the proposed method effectively learns the current task by maximizing the information of feature maps and minimizing the loss. This framework also reduces the negative interference between tasks. A diverse set of experiments indicates that the proposal outperforms other continual learning competitors with different learning strategies. The effectiveness of the proposed approach under different learning scenarios was extensively evaluated. The proposed method exhibited a competitive performance among the considered approaches without requiring additional parameters. We have also provided thorough analyses of the proposed method under different experimental setups.

REFERENCES

[1] Eunwoo Kim, Chanho Ahn, and Songhwai Oh. Nestednet: Learning nested sparse structures in deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8669–8678, 2018.

[2] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1871–1880, 2019.

[3] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13):3521–3526, 2017.

[4] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 3987–3995, 2017.

[5] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In European Conference on Computer Vision, pages 144–161. Springer, 2018.

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

[7] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 374–382, 2019.

[8] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 6470–6479, 2017.

[9] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. arXiv preprint arXiv:1606.04671, 2016.

[10] JaeHong Yoon, Jeongtae Lee, Eunho Yang, and Sung Ju Hwang. Lifelong learning with dynamically expandable network. In International Conference on Learning Representations, 2018.

[11] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3366–3375, 2017.

[12] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7765–7773, 2018.

[13] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In International Conference on Machine Learning, pages 4548–4557. PMLR, 2018.

[14] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation, volume 24, pages 109–165. Elsevier, 1989.

[15] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. arXiv preprint arXiv:1909.08383, 2019.

[16] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In Advances in Neural Information Processing Systems, pages 13647–13657, 2019.

[17] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In Proceedings of the European Conference on Computer Vision (ECCV), pages 67–82, 2018.

[18] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255. Ieee, 2009.

[20] Zhizhong Li and Derek Hoiem. Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(12):2935–2947, 2017.

[21] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

[22] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11321–11329, 2019.

[23] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3931–3940, 2020.

[24] Chanho Ahn, Eunwoo Kim, and Songhwai Oh. Deep elastic networks with model selection for multi-task learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6529–6538, 2019.

[25] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In Proceedings of the European Conference on Computer Vision (ECCV), pages 3–18, 2018.

[26] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8817–8826. IEEE, 2018.

[27] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. You look twice: Gaternet for dynamic filter selection in cnns. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9172–9180, 2019.

[28] V. Klema and A. Laub. The singular value decomposition: Its computation and some applications. IEEE Transactions on Automatic Control, 25(2):164–176, 1980.

[29] Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song. Self-supervised knowledge distillation using singular value decomposition. In Proceedings

of the European Conference on Computer Vision (ECCV), pages 335–350, 2018.

[30] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In International Conference on Machine Learning, pages 9120–9132. PMLR, 2020.

[31] Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. In Thirty-Fifth Conference on Neural Information Processing Systems, 2021.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

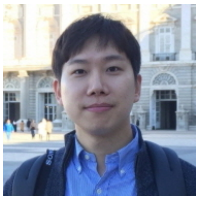
[33] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In British Machine Vision Conference 2016. British Machine Vision Association, 2016.

[34] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In NeurIPS Continual learning Workshop, 2018.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32, pages 8024–8035. 2019.

HYUNDONG JIN received the B.S. degree in Electrical and Electronics Engineering from Chung-Ang University, Seoul, Korea, in 2020. He is currently a M.S. student at School of Computer Science and Engineering, Chung-Ang University, Seoul, Korea. His research interests include deep learning, machine learning, continual learning, and computer vision.

KIMIN YUN received the B.S and the Ph.D degrees at the Department of Electrical and Computer Engineering from Seoul National University, Seoul, Rep. of Korea, in 2010 and 2017, respectively. Since 2017, he has been working with the Visual Intelligence Research Section in the Artificial Intelligence Research Laboratory at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea. His current research interests include machine learning, computer vision, visual event detection, moving object detection, and video analysis.



EUNWOO KIM received the B.S. degree in Electrical and Electronics Engineering from Chung-Ang University, Seoul, Korea, in 2011, and the M.S. and Ph.D. degrees in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea in 2013 and 2017, respectively. He is currently an assistant professor with the School of Computer Science and Engineering, Chung-Ang University, Seoul, Korea. From 2017 to 2018, he was a postdoctoral researcher with the Department of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea. From 2018 to 2019, he was a postdoctoral researcher with the Department of Engineering Science, University of Oxford, Oxford, UK. His research interests include machine learning, deep learning, model optimization, robotics, and computer vision.