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

HYUNJONG JIN\textsuperscript{1}, KIMIN YUN\textsuperscript{2}, AND EUNWOO KIM\textsuperscript{1}, (Member, IEEE)

\textsuperscript{1}School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea
\textsuperscript{2}Artificial Intelligence Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea

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

This work 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 a 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, these 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 optimize a new task while avoiding negative interference efficiently. 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]. Note that 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], [23] 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. This may incur negative interference between tasks due to the indiscriminate use of previous parameters (knowledge) with the new ones.

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 to 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 an 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 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 the 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], [4], [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. References [9] and [11] proposed new dynamic models to accommodate each incoming task. References [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 proportionally 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 restricting the update of critical parameters of the earlier 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. GNN-CL: GATED NEURAL NETWORK IN CONTINUOUS LEARNING

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

#### A. FRAMEWORK

GNN-CL aims to skip previously learned parameters that may cause negative task interference. It 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 $T = \{T^1, T^2, \cdots\}$, where each task $T^i = \{X^i, Y^i\}$ contains data $X^i = \{x^i_j\}_{j=1}^n$ and the corresponding labels $Y^i = \{y^i_j\}_{j=1}^n$. 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 group of layers.

| Symbol | Meaning |
|--------|---------|
| $T^i$ | $i$-th task |
| $f(\cdot)$ | feature extractor |
| $U_l$ | $l$-th unit in feature extractor |
| $\theta^1_l, \theta^2_l$ | $l$-th task-specific parameter in $f(\cdot)$ and $U_l$ |
| $\theta_i^1, \theta_i^2, \theta_i'$ | $i$-th task-specific parameter in $f(\cdot)$ and $U_l$ after retraining |
| $\hat{Y}^i$ | average feature map information in the $i$-th unit |
| $\tilde{Y}_i$ | gate that controls $\theta_i^1$ in $U_l$ |

Specifically, we define the feature extractor with $L$ units as

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

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

$$
 f_l(x) \equiv (U_l \circ \cdots \circ U_1)(x).
$$

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

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

The objective function for the $i$-th task is

$$
 \text{argmin}_{\theta_{i, w^i}} \mathcal{L}(y^i, \hat{y}^i), \quad \text{where } \mathcal{L}(y^i, \hat{y}^i) = -\sum_{j=1}^n y^i_j \log \hat{y}^i.
$$

This objective updates the newly allocated parameters $\theta^l$ and $w^l$, 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 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 gating mechanism. The figure on the left represents the network before controlling the gates. The operations of the unit $i$ 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 $\theta_i^t$ and $i^{th}$ new set of parameters $\theta_i^u$. Furthermore, we obtain loss $\mathcal{L}_i^{(i,t)}$ by further executing $c_w$. Similarly, we obtain information $I_i^{(i)}$ and loss $\mathcal{L}_i^{(i)}$ using $\theta_i^t$ (without $\theta_i^u$). 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, it 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 $h_i^{(l,t)}(x^{i})$ ($i = 1, 2, \cdots, t-1$) and $h_i^{(l)}(x^{i})$, respectively, where $t$ is the index of the current task. The feature map $h_i^{(l,t)}(x^{i})$ in the unit $l$ can be decomposed through SVD as

$$h_i^{(l,t)}(x^{i}) = (U_l \circ f_{l-1})(x^{i}),$$

$$= \sum_{k=1}^{K} u_k s_k v_k^T + \sum_{j=k+1}^{K} u_j s_j v_j^T,$$  \hspace{1cm} (5)

where $u_k$ and $v_k$ denote the left and right singular vectors, respectively, and $s_k$ is the singular value of $h_i^{(l,t)}(x^{i})$. Similarly, the feature map $h_i^{(l)}(x^{i})$ is obtained similar to the above equation, where $U_l$ is parameterized with $\theta_i^t \cup \theta_i^u$. Note that $h_i^{(l,t)}(x^{i})$ (or $h_i^{(l)}(x^{i})$) can be divided into two terms by the $k^{th}$ rank, where the left (low-rank) term $\sum_{j=1}^{K} u_k s_k v_j^T$ 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

$$S(h_i^{(l,t)}(x_j^{i})) = \sum_{k=1}^{K} s_k,$$  \hspace{1cm} (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 $x_j^{i}$. $I_i^{(l,t)}$ and $I_i^{(l)}$ represent the average feature map information in the $l^{th}$ unit, calculated as

$$I_i^{(l,t)} = \frac{1}{n} \sum_{j=1}^{n} S(h_i^{(l,t)}(x_j^{i})).$$  \hspace{1cm} (7)

$$I_i^{(l)} = \frac{1}{n} \sum_{j=1}^{n} S(h_i^{(l)}(x_j^{i})).$$  \hspace{1cm} (8)

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

$$g_i^l = \begin{cases} 1 & \text{if } I_i^{(l,i)} > I_i^{(l)} \\ 0 & \text{otherwise.} \end{cases}$$

(9)

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

$$f(x^l) = (U_L \circ \cdots \circ U_l \circ \cdots \circ U_1)(x^l),$$

where $U_l \leftarrow \left( U_i^{(l,i)} \cdot g_i^l \right) \cup \bar{\theta}_i^l$. (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 $L_i^{(l,i)}$ and $L_i^{(l,i,t)}$, respectively. Formally, we can represent $L_i^{(l,i,t)}$ as

$$L_i^{(l,i,t)} = -\sum_{j=1}^{N} y_j \log(c_w \circ f)(x^l).$$

(11)

Likewise, $L_i^{(l,t)}$ is obtained in a manner similar to the above equation, where $U_l$ is parameterized with $\theta^l$. Losses $L_i^{(l,i)}$ and $L_i^{(l,i,t)}$ are considered with the information in the feature maps $I_i^{(l,i)}$ and $I_i^{(l,i,t)}$ 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

$$\hat{g}_i^l = \begin{cases} 1 & \text{if } \frac{I_i^{(l,i)}}{L_i^{(l,i)}} > \frac{I_i^{(l)}}{L_i^{(l)}} \\ 0 & \text{otherwise.} \end{cases}$$

(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

$$f(x^l) = (U_L \circ \cdots \circ U_l \circ \cdots \circ U_1)(x^l),$$

where $U_l \leftarrow \left( U_i^{(l,i)} \cdot \hat{g}_i^l \right) \cup \bar{\theta}_i^l$. (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

We compared the proposed method, GNN-CL, 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. All experiments were conducted using the PyTorch library [32] and NVIDIA 2080Ti GPU.

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 $32 \times 32$ 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 (super-classes), each of which involved five classes. The characteristics of the datasets used in the experiments are summarized in Table 2.

| Dataset       | # Train | # Test | # Class |
|---------------|---------|--------|---------|
| ImageNet-50   | 65,000  | 2,500  | 50      |
| CIFAR-100     | 50,000  | 10,000 | 100     |

B. IMPLEMENTATION DETAILS

We used the ResNet-20 [33] and WideResNet-28-2 (WRN-28-2) [34] 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 regularizaion strategy, GEM [8] in the replay strategy, and PackNet [12] in the structural allocation strategy.

In the 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 naive version of the replay method (Replay) [35], which

TABLE 2. Datasets used in this work.

| Dataset       | # Train | # Test | # Class |
|---------------|---------|--------|---------|
| ImageNet-50   | 65,000  | 2,500  | 50      |
| CIFAR-100     | 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      |
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).

|         | Joint | Fine-tune | Replay-2K | BWC | 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   | 53.67 |
| N - R - 10 | 82.34 | 27.56     | 66.62     | 55.70 | 51.65 | 49.48 | 63.22  | 58.06   | 64.73 |
| N - R - 25 | 91.77 | 52.42     | 82.64     | 73.06 | 53.30 | 68.08 | 78.55  | 77.76   | 80.44 |
| W - R - 5  | 76.88(+3.38) | 24.83(+1.45) | 61.0(+4.5) | 29.61(18.5) | 53.06(5.77) | 48.90(+2.96) | 56.86(+2.78) | 54.36(+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) | 64.94(+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) | 78.64(+0.08) | 78.46(+0.88) | 81.26(+0.82) |

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.

randomly stores a part of the previous data and performs joint training with the current data.

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 [33]. The results for different numbers of units to be controlled in the ablation study were presented.

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 [33] on ImageNet-50. Note that the 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 GNN-CL (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, GNN-CL 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 increases. 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
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     | 89.22     | 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) | 21.9(+0.17) | 79.26(+3.73) | 65.83(+2.18) | 61.34(+0.99) | 61.31(+1.95) | 72.16(+2.09) | 81.8(+0.6) | 86.2(+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) | 81.9(-2.79) | 87.7(+1.87) |

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.

allocation strategy, PackNet, and ours improve performance as the larger-scale network is used. The proposed 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 GNN-CL 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 accuracy because certain parameters are pruned for the subsequent tasks. The proposed method can retain the previous knowledge and learn 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 [33]. 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. The 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. GNN-CL consistently outperforms PackNet, with margins of 1.99%, 0.31%, and 1.19% for 5, 10, and 20 tasks, respectively. Notably, GNN-CL 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.

**E. ANALYSIS**

In this 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 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. The 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 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

### 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(1.17) | 56.68(1.10) | 64.10(1.89) | 75.60(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) | 78.83(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) | 84.08(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) | 73.32(4.1) | 81.8(0.73) |
| $W - R - 10$ | 91.19(0.53) | 22.00(0.01) | 78.24(1.02) | 62.26(3.57) | 61.34(0.0) | 59.90(1.41) | 72.17(0.01) | 80.64(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) | 77.39(0.67) | 82.76(0.85) | 87.67(0.11) |

**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.

**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).
H. Jin et al.: Gating Mechanism in Deep Neural Networks for Resource-Efficient Continual Learning

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 applying a gate to deeper units representing better task-specific features is more effective and can enhance the performance.
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 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] E. Kim, C. Ahn, and S. Oh, “NastedNet: Learning nested sparse structures in deep neural networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2018, pp. 8669–8676.
[2] S. Liu, E. Johns, and A. J. Davison, “End-To-End multi-task learning with attention,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 1871–1880.
[3] K. James, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, and A. Grabska-Barwinska, “Overcoming catastrophic forgetting in neural networks,” Proc. Nat. Acad. Sci. USA, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.
[4] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” in Proc. 4th Int. Conf. Mach. Learn., vol. 70, 2017, pp. 3987–3995.
[5] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in Proc. Eur. Conf. Comput. Vis. Springer, 2018, pp. 144–161.
[6] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “ICaRL: Incremental classifier and representation learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2001–2010.
[7] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, “Large scale incremental learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 374–382.
[8] D. Lopez-Paz and M. Ranzato, “Gradient episodic memory for continual learning,” in Proc. 31st Int. Conf. Neural Inf. Process. Syst., 2017, pp. 6470–6479.
[9] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” 2016, arXiv:1606.04671.
[10] J. Yoon, J. Lee, E. Yang, and S. I. Hwang, “Lifelong learning with dynamically expandable networks,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–11.
[11] R. Aljundi, P. Chakravarty, and T. Tuytelaars, “Expert gate: Lifelong learning with a network of experts,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 3366–3375.
[12] A. Mallya and S. Lazebnik, “PackNet: Adding multiple tasks to a single network by iterative pruning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 7765–7773.
[13] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, “Overcoming catastrophic forgetting with hard attention to the task,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 4548–4557.
[14] M. McCluskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” Psychol. Learn. Motivat., vol. 24, pp. 109–165, Dec. 1989.
[15] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, “A continual learning survey: Defying forgetting in classification tasks,” 2019, arXiv:1909.08383.
[16] C.-Y. Hung, C.-H. Tu, C.-E. Wu, C.-H. Chen, Y.-M. Chan, and C.-S. Chen, “Compacting, picking and growing for unforgetting continual learning,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 13647–13657.
[17] A. Mallya, D. Davis, and S. Lazebnik, “Piggyback: Adapting a single network to multiple tasks by learning to mask weights,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 67–82.
[18] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2009, pp. 248–255.
[20] R. Aljundi and D. Hoiem, “Learning without forgetting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
[21] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, “A continual learning survey: Defying forgetting in classification tasks,” IEEE Trans. Pattern Anal. Mach. Intell., early access, Feb. 5, 2021, doi: 10.1109/TPAMI.2021.3057446.
[22] O. Ostapenko, M. Pascas, T. Klein, P. Jahnichen, and M. Nabi, “Learning to remember: A synaptic plasticity driven framework for continual learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 11321–11329.
[23] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. E. Bejnordi, “Conditional channel gated networks for task-aware continual learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 2039–3940.
[24] C. Ahn, E. Kim, and S. Oh, “Deep elastic networks with model selection for multi-task learning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 6529–6538.
[25] A. Veit and S. Belongie, “Convolutional networks with adaptive inference graphs,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 3–18.
[26] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Ferris, “BlockDrop: Dynamic inference paths in residual networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8817–8826.
[27] Z. Chen, Y. Li, S. Bengio, and S. Si, “You look twice: GaterNet for dynamic filter selection in CNNs,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 9120–9132.
[28] V. Klima and A. J. Laub, “The singular value decomposition: Its computation and some applications,” IEEE Trans. Autom. Control, vol. 25, no. 2, pp. 164–176, Apr. 1980.
[29] S. Liu, D. H. Kim, and B. C. Song, “Self-supervised knowledge distillation using singular value decomposition,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 335–350.
[30] T. Standlee, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese, “Which tasks should be learned together in multi-task learning?” in Proc. Int. Conf. Mach. Learn., 2020, pp. 9120–9132.
[31] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn, “Efficiently identifying task groupings for multi-task learning,” in Proc. 45th Conf. Neural Inf. Process. Syst., 2021, pp. 1–14.
HYUNDONG JIN received the B.S. degree in electrical and electronics engineering from Chung-Ang University, Seoul, South Korea, in 2020, where he is currently pursuing the M.S. degree with the School of Computer Science and Engineering. His research interests include deep learning, machine learning, continual learning, and computer vision.

KIMIN YUN received the B.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering, Seoul National University, Seoul, Republic of Korea, in 2010 and 2017, respectively. Since 2017, he has been working with the Visual Intelligence Research Section, Artificial Intelligence Research Laboratory, 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 (Member, IEEE) received the B.S. degree in electrical and electronics engineering from Chung-Ang University, Seoul, South Korea, in 2011, and the M.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, in 2013 and 2017, respectively. From 2017 to 2018, he was a Postdoctoral Researcher with the Department of Electrical Engineering and Computer Science, Seoul National University. From 2018 to 2019, he was a Postdoctoral Researcher with the Department of Engineering Science, University of Oxford, Oxford, U.K. He is currently an Assistant Professor with the School of Computer Science and Engineering, Chung-Ang University. His research interests include machine learning, deep learning, model optimization, robotics, and computer vision.