Received December 31, 2021, accepted January 28, 2022, date of publication February 7, 2022, date of current version February 14, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3149598

Incremental Learning With Adaptive Model Search and a Nominal Loss Model

CHANHO AHN[1], (Graduate Student Member, IEEE), EUNWOO KIM[2], (Member, IEEE), AND SONGHWAI OH[1], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, ASRI, Seoul National University, Seoul 08826, South Korea
[2]School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea

Corresponding author: Songhwai Oh (songhwai@snu.ac.kr)

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) funded by the Korean Government, Ministry of Science and Information & Communications Technology (MSIT), ([SW Star Lab] Robot Learning Efficient, Safe, and Socially-Acceptable Machine Learning) under Grant 2019-0-01190, in part by the Information & Communication Technology (ICT) Research and Development Program of MSIT/IITP (Development of Cloud Robot Intelligence Augmentation, Sharing and Framework Technology to Integrate and Enhance the Intelligence of Multiple Robots) under Grant 2020-0-00857, and in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2021-0-01341, Artificial Intelligence Graduate School Program (Chung-Ang University)).

ABSTRACT This paper addresses an incremental learning problem, in which tasks are learned sequentially without access to the previously trained dataset. Catastrophic forgetting is a significant bottleneck to incremental learning as the network performs poorly on previous tasks when it is trained on a new task. We propose an adaptive model search method that uses a different part of the backbone network depending on an input image to mitigate catastrophic forgetting. Our model search method prevents forgetting by minimizing the update of critical parameters for the previous tasks while learning a new task. This model search involves a trainable network that selects the model structure for an input image. We also propose a method for approximating the loss function of previous tasks without the previous dataset. The critical parameters for the previous tasks can be found, considering the relationship between the approximated loss function and the parameters. The proposed framework is the first method of model search that can consider the performance of both current and previous tasks in the incremental learning problem. Empirical studies and theoretical analysis show that the proposed method outperforms other competitors for old and new tasks while requiring less computation.

INDEX TERMS Artificial neural network, computer vision, incremental learning, model selection, object recognition.

I. INTRODUCTION

Icremental learning sequentially trains incoming tasks without accessing datasets of previous tasks. It has attracted attention due to its wide range of practical applications [1], [2], such as life-long learning and personalized learning. Datasets for previous tasks may not be provided for various reasons, such as the privacy, security, and massive size of the dataset in real-world applications. Without the previous dataset, the deep neural network has a chronic problem called catastrophic forgetting; That is, the network quickly loses existing information about the previous task while learning a new task [3]. Catastrophic forgetting is easily solved by using independent networks for each task, but this way is inefficient in terms of memory and inference speed. Actually, mobile robots and embedded devices include low memory resources, and loading a different network each time perform a task entails unnecessary computation. This paper aims to develop an incremental learning algorithm applicable to various personal devices. Therefore, this paper covers methods involving a limited number of networks, regardless of the number of tasks.

To alleviate the catastrophic forgetting problem with a few networks, incremental learning researches preserve previous knowledge through generative [4]–[6], architectural [7]–[9], functional [10], [11], and structural regularized [12]–[14] approaches. The generative approaches produce pseudo data for previous tasks to derive the proxy loss function for the previous tasks. However, the performance of these methods depends on the quality of the pseudo data. Also, learning the generative model is another challenging problem. The architectural approaches assign additional parameters to a new

The associate editor coordinating the review of this manuscript and approving it for publication was Wenbing Zhao.

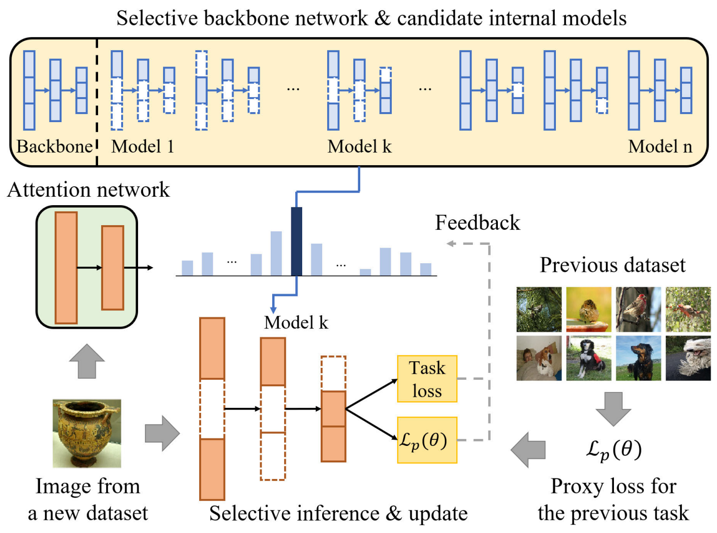


FIGURE 1. An overview of the proposed framework. We produce multiple internal models from a backbone network. An attention network selects a proper internal model for handling an input image. A proxy loss for the previous task is used to train the attention network and the backbone network to preserve the previous knowledge.

task while freezing the trained parameters for the previous tasks. The functional approaches maintain input-output mappings using the outputs of the previously learned networks. As the number of tasks increases, computationally intractable problems arise with the architectural and functional methods; A memory to store parameters or computation to calculate outputs of the previous networks increases. The structural regularized methods penalize parameter changes from previously learned parameters. It can handle sequential tasks without increasing memory or computation. Nevertheless, often this penalty interferes with learning new tasks.

Moreover, class-incremental learning has been studied to solve incremental learning problems without task identification. In the class-incremental learning problem, a weight imbalance problem occurs in which the classifier more predicts recently learned classes. Recent advances [15]–[17] in incremental learning resolve this problem by using sample images of previous tasks. However, we only deal with situations where we do not use data from previous tasks. In addition, a recent study [18] shows that a network trained only on sample images can achieve a comparable performance to that of other class-incremental learning methods. This result indicates that using sample images is more critical than other techniques in the class-incremental learning problem. Therefore, the class-incremental learning is outside the scope of this paper, and we are focusing on the task-incremental learning problem.

This paper proposes a task-incremental algorithm with a fixed network size and training cost that does not grow as the number of tasks increases. To this end, we propose an adaptive model search that selectively uses and updates the network parameters according to each input image (see Figure 1). The proposed model search selects a model structure suitable for learning a new task without compromising the performance of previous tasks. The model search is based on a trainable neural network, called an attention network, which outputs a probability of model structure. The attention network is trained to minimize the loss function of the previous and current tasks. This learning strategy is differentiated from existing incremental learning methods [19], [20]. The existing methods find a model structure composed of parameters that are rarely used or have little effect on the previous tasks to learn the current task. This distinction contributes to our model search avoiding catastrophic forgetting without losing too much ability to learn new tasks.

Since the previous datasets are not accessible, previous loss functions can not be derived. To derive the proxy loss for the previous task without the dataset, we propose a nominal loss model for previous tasks inspired by a nominal plant model in control theory [21]. The nominal loss model defines the nominal parameter set, which is assumed to be an optimal point for the previous task. Then, the proxy loss is calculated as the difference between the current parameter set and the nominal parameter set. Unlike similar methods using approximated loss function for the previous task [12]–[14], the nominal parameter set is designed to be free from interference with other tasks. The proposed nominal parameter set has a lower loss value than the learned parameter set, and we prove it theoretically and empirically.

To the best of our knowledge, our work is the first model search method which considers both the performance of the previous and the current tasks. Since the proposed model search can selectively update parameters for new tasks, it has the potential to prevent catastrophic forgetting. Our method relieves losing the ability to learn new tasks by finding the optimal network model, which minimizes a loss function for whole tasks. Besides, a nominal loss model proposed to learn the model search provides a nominal parameter set which can more accurately approximate the previous loss function than the existing incremental learning methods. Compared to the existing works [12]–[14], [22], the proposed method achieves outstanding performance on benchmark datasets in incremental learning. Additionally, the proposed method requires fewer computations (FLOPs) than other competing algorithms while showing favorable performance.

In summary, the main contributions of our work are:

- We propose a model search method that enables selectively using and updating the network parameters. It is the first model search method considering the previous and current tasks in the incremental learning problem.
- We provide a method to obtain an approximate loss function that is more accurate than our competitors. The superiority of the proposed approximation method is demonstrated theoretically.
- The proposed framework shows outstanding performance on the various incremental learning scenarios. Our method achieves higher accuracy on most tasks with fewer computations than other competitors.

In Section II, we provide a literature review of related work. Section III covers the proposed method, including a network structure, a loss function, and an algorithm. Empirical demonstrations of the benefit of the proposed method are presented in Section IV. Finally, Section V summarizes the overall framework and mentions the future work.

II. RELATED WORK

Incremental learning methods with a generative model create pseudo images of the previous tasks [4]–[6], or a parameter set that correspond to the local minimum of the previous task [23]. However, these methods are greatly affected by the performance of the generative model, and learning the generative model can be a challenging problem. Architectural approaches [7], [8], [24] in incremental learning provide additional parameters to train new tasks. As the number of tasks increases, the network size gradually increases, making it difficult to handle many tasks. Functional approaches [10], [11], [25] preserve the previous knowledge by maintaining the input-output mapping of the previously trained network. They also introduce an additional computational overhead by calculating the previous network's input-output mapping for instances of the current task. Structural regularized approaches [12]–[14] give a quadratic penalty on the difference between parameters for the current task and the old task. While they are computationally tractable methods based on Taylor quadratic approximation with diagonal Hessian matrices [26], the quadratic penalty often hinders learning a new task.

Network model search approaches based on the backbone structure exploit an efficient model consisting of a subset of the backbone network's parameters. Compared to the conventional network architecture search method [27], the backbone-based model search methods [28], [29] can reduce the ample space of model structures to be explored. The model structures are explored by removing certain filters of layers of the backbone architecture. Recent methods [30]–[32] which allow for tighter model search, suggest a strategy for selecting a model structure for each instance. These studies have rarely been applied to incremental learning. There is a model selection method that deals with incremental learning [20], which repeats the process of reducing the parameters and continuously learning new tasks on the removed parameters. However, this method cannot deal with a large number of tasks since the model has to be continuously compressed each time it learns a new task.

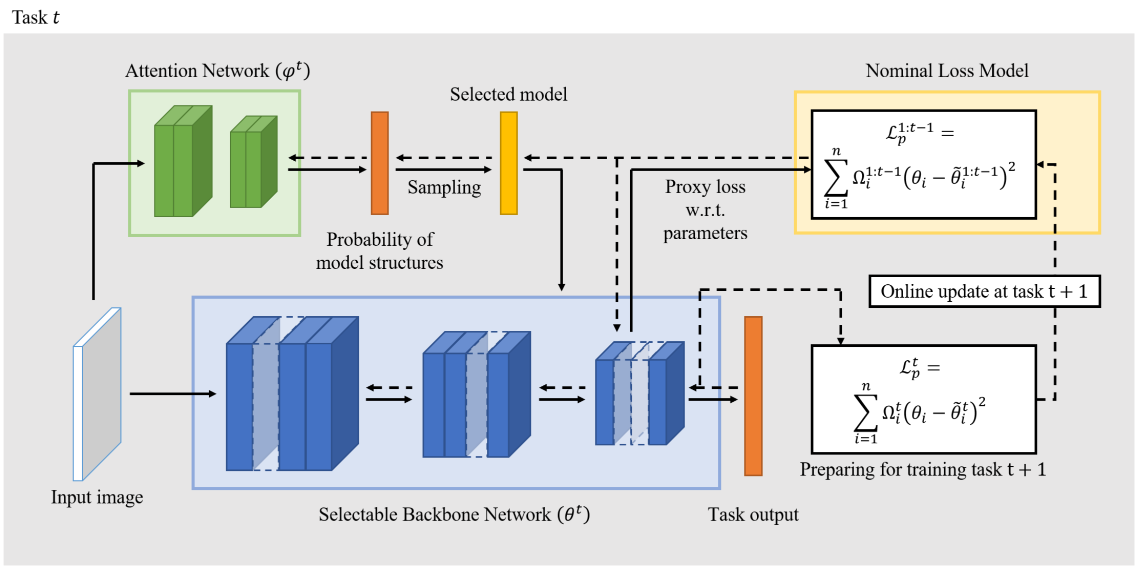
III. METHODS

This paper addresses incremental learning (strictly, task-incremental learning) problems in which multiple tasks are given sequentially and assume a situation where previously trained datasets are no longer accessible. We aim to provide a scalable incremental learning method free from an increase in memory or training cost even if the number of tasks increases. To this end, we propose an instance-wise model search method that refers to a backbone network and uses a part of the network according to the input image. The proposed method can prevent catastrophic forgetting by updating an appropriate part of the backbone that can increase the performance of the previous and current tasks. To measure the performance of the previous tasks, we propose a nominal loss model which can approximate the loss function of the previous tasks without accessing previous datasets. The following sub-sections describe the overall network architecture, the nominal loss model, and the learning strategy in order. The overall framework is illustrated in Figure 2.

A. NETWORK ARCHITECTURE

We propose a backbone-based model search method which does not require more parameters or computation than the backbone network. To design a backbone network from which multiple internal models can be derived, we define an internal model using some parameters of the backbone network; We call this backbone network as a selective backbone network (SBN). Since the combination of network parameters to define the internal model causes computationally intractable cases, we divide the network parameters into multiple groups to define the internal model as a combination of parameter groups. Besides, we define structurally clustered parameters (i.e., convolution channels or layers) into groups that can be considered for practical memory efficiency [33].

We divide the parameters of two consecutive convolution layers into g groups. The channels of a hidden layer between two convolution layers are split into g sets so that the original convolution operation is represented as the sum of separated convolution operations (we call them sub-convolutions)



Task t
Attention Network (φ^t)
Selected model
Sampling
Probability of model structures
Nominal Loss Model
$\mathcal{L}_p^{1:t-1} = \sum_{i=1}^{n} \Omega_i^{1:t-1}(\theta_i - \tilde{\theta}_i^{1:t-1})^2$
Proxy loss w.r.t. parameters
Online update at task t + 1
$\mathcal{L}_p^{t} = \sum_{i=1}^{n} \Omega_i^{t}(\theta_i - \tilde{\theta}_i^{t})^2$
Preparing for training task t + 1
Input image
Selectable Backbone Network (θ^t)
Task output

FIGURE 2. **The attention network outputs the probability distribution for internal models for training and inference of the current task. By sampling the internal model based on the probability distribution, the selected model structure is used to learn the input image. To preserve the performance of the previous tasks, the proxy loss function for the previous tasks is also minimized. The proxy loss function for the current task is derived to learn future tasks. The dotted arrows indicate elements required to update the object it points to.**

as follows:

$$\sigma(W^2 * \sigma(W^1 * I)) = \sigma\left(\sum_{i=1}^{g} W_i^2 * \sigma(W_i^1 * I)\right) \rightarrow \sigma\left(\sum_{i=1}^{g} \mathbb{I}_i \cdot W_i^2 * \sigma(W_i^1 * I)\right), \quad (1)$$

where W^1 and W^2 are original two convolution layers, W_i^1 and W_i^2 are i-th sub-convolution layers, σ is an activation function, I is an input layer, and $\mathbb{I}_i$ denotes a binary value that determines whether the i-th sub-convolution is used for the internal model. Where there are p consecutive convolution pairs in the backbone network, the SBN can produce $2^{p\times(g-1)}$ internal models.[1] The concept of the SBN can be applied most well-known convolution networks and fully connected networks by assuming a fully connected layer as a 1×1 convolution layer.

In details, for two consecutive convolution layers, the output of the first convolution layer is divided into g tensors as follows:

$$\sigma(W^1 * I) = \text{cat}\left[\sigma(W_1^1 * I),\ \sigma(W_2^1 * I) \cdots \sigma(W_g^1 * I)\right], \quad (2)$$

where $I \in \mathbb{R}^{h\times w\times c_i}$, h, w, and c_i are the size of height, width, and the number of channels, respectively, $W^1 \in \mathbb{R}^{s\times s\times c_i\times f}$, s is the size of convolution kernel, f is the number of output channels, $W_i^1 \in \mathbb{R}^{s\times s\times i\times f_i}$ is a divided convolution layer which satisfies $\sum_{i=1}^{g} f_i = f$, and cat$[\cdot]$ denotes a concatenate operation with respect to the last dimension. Then, the output of the second convolution layer is calculated as the sum of divided convolutions on the separate inputs:

$$\sigma(W^2 * I') = \sigma\left(\sum_{i=1}^{g} W_i^2 * I_i'\right), \quad (3)$$

where $W^2 \in \mathbb{R}^{s\times s\times f\times c_o}$, c_o is the number of output channels, $I' = \sigma(W^1 * I)$, $I_i' = \sigma(W_i^1 * I)$, and $W_i^2 \in \mathbb{R}^{s\times s\times f_i\times c_o}$ is the divided convolution layer which refers divided layer (I_i'). In most case, $f_i = \lfloor f/g \rfloor$ for $i < g$, and $f_g = f - \sum_{i=1}^{g} f_i$.

We propose a trainable network model, called an attention network (AN), to find a proper internal model by selecting sub-convolutions to use. The AN returns probability values for whether each sub-convolution is used based on the input image; The output dimension of the AN is $p(g-1)$ and the AN takes an image of the same size as the backbone network. Note that our input-wise model selection can handle tasks in a single-head network without task identification (see Experiment Results). After the model structure is determined through the AN, the task is performed using the corresponding internal model. To build the whole framework to be memory efficient, the AN is designed to be much smaller than the SBN. Indeed, choosing a small internal model from the SBN can reduce the actual computational amount. We also design the AN identically to the structure of the smallest-sized internal model from the SBN to alleviate the difficulty of designing the AN (but the AN and the SBN do not share parameters).

B. NOMINAL LOSS MODEL

We use proxy loss functions w.r.t. parameters of the network to represent the sum of the loss functions for all tasks without

[1]Cases where no sub-convolution is used are excluded.

accessing the previous datasets:

$$\min_{\theta} \sum_{t=1}^{T} \mathcal{L}^t(\theta; \mathcal{D}^t) \approx \mathcal{L}^T(\theta; \mathcal{D}^T) + \lambda \sum_{t=1}^{T-1} \mathcal{L}_p^t(\theta), \tag{4}$$

where $\mathcal{L}^t$ and $\mathcal{D}^t$ are the loss function and dataset for task t, respectively, θ is a parameter set of the network, T is the number of tasks, $\mathcal{L}_p^t$ is the proxy function for the loss function of task t, and λ is a weighting factor which controls the strength of the proxy losses. The existing methods using the proxy loss [12]–[14] assume that the proxy loss function has a minimum value at the parameter set trained for the previous task. Then, they define the proxy loss function as a quadratic function which has a minimum value at the trained parameter set for the previous task [12]–[14]:

$$\mathcal{L}_p^t(\theta) = \sum_{i=1}^{n} \Omega_i(\theta_i - \hat{\theta}_i^t)^2, \tag{5}$$

where n is the number of parameters, $\Omega_i \in \mathbb{R}_+$ represents an intensity for the i-th parameter, and $\hat{\theta}^t$ denotes a trained parameter set for task t. The proxy loss in (5) penalizes the parameter set as it moves away from the trained parameter set considering importance of the parameters. Although these methods show computational ease and competitive performance for incremental learning scenarios, the assumption that the parameter set trained for the previous task has a minimum proxy loss value may not be correct. In the incremental learning scenarios, the parameter set is trained not for the current task loss function but also for the proxy loss function for the past tasks while learning the current task. The proxy loss term for previous tasks may prevent the parameter from reaching the minimum of the current task loss function. Actually, learning with the proxy loss often results in lower performance in the current task than single task training (see Experimental Results).

To mitigate the interference of proxy loss, we define a nominal loss model that refers to a nominal parameter set closer to the optimal point of the current loss function than the trained parameter set. The proposed nominal loss model calculates the proxy loss function by calculating the intensities for each parameter and the nominal parameter set, as follows:

$$\mathcal{L}_p^t(\theta) = \sum_{i=1}^{n} \Omega_i(\theta_i - \tilde{\theta}_i^t)^2, \tag{6}$$

where $\tilde{\theta}^t$ is the nominal parameter set for task t.

The proposed nominal loss model reflects the amount of change in the loss function according to each parameter change. While learning task t from the trained parameter set for task $t-1$, the amount of change in the loss function for each parameter change is accumulated as follows [14]:

$$\mathcal{L}^t(\hat{\theta}^{t-1}; \mathcal{D}^t) - \mathcal{L}^t(\hat{\theta}^t; \mathcal{D}^t) \approx \sum_{j=1}^{m} (\nabla_\theta \mathcal{L}^t)_j^T (\Delta\theta)_j, \tag{7}$$

where m is the number of iterations for training the task, $(\nabla_\theta \mathcal{L})_j$ and $(\Delta\theta)_j$ are the gradient and the amount of change in

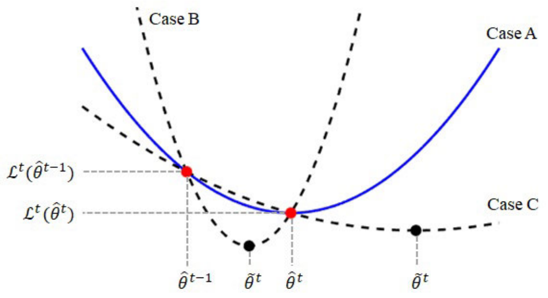


FIGURE 3. Schematic diagram of the proxy loss function. The solid blue line indicates the case where $\alpha = 0$ (Case A), and the dotted lines represent the function when the signs of the gradient and parameter change are the same (Case B) and the function when the signs are different (Case C). In Case B, the nominal parameter set ($\tilde{\theta}^t$) is between $\hat{\theta}^{t-1}$ and $\hat{\theta}^t$. In Case C, $\tilde{\theta}^t$ exists outside the boundary.

the parameters at the j-th iteration, respectively. The amount of change in the loss function according to the parameter change determines the intensity of each parameter on the proxy loss. The proxy loss is expressed as the sum of terms independent of each parameter in (6) so that the proxy loss can capture the characteristics of (7) by satisfying the following condition for each parameter (Condition 1):

$$\mathcal{L}_{p,i}^t(\hat{\theta}_i^{t-1}) - \mathcal{L}_{p,i}^t(\hat{\theta}_i^t) = \Delta\mathcal{L}_i^t := \sum_{j=1}^{m} (\nabla_{\theta_i} \mathcal{L}^t)_j \cdot (\Delta\theta_i)_j,$$
$$s.t.\ \ \mathcal{L}_{p,i}^t(\theta) = \Omega_i(\theta_i - \tilde{\theta}_i^t)^2. \tag{8}$$

The nominal loss model also refers to the gradient value of the current loss function at the trained parameter set to find a parameter set that has a smaller loss function value for the current task than the trained parameter set. Then, the proxy loss preserves the gradient of the original loss function at the trained parameter set (Condition 2):

$$\nabla_{\theta_i} \mathcal{L}_{p,i}^t(\hat{\theta}_i^t) = \alpha \nabla_{\theta_i} \mathcal{L}^t(\hat{\theta}^t), \tag{9}$$

where $\alpha \in [0, 1]$ is a scalar value which relaxes the gradient value. The relaxation is necessary to adjust the balance between the first and second conditions. When the gradient value is greater than the amount of change of the loss function for the parameter, Ω in (5) is calculated as a negative number.

The nominal loss model which derives the proxy loss function satisfying two conditions includes the following nominal parameter set and intensities:

$$\Omega_i = \frac{\Delta\mathcal{L}_i^t + \alpha \nabla_{\theta_i} \mathcal{L}(\hat{\theta}^t)(\Delta\theta_i)}{\Delta\theta_i^2},$$
$$\tilde{\theta}_i = \hat{\theta}_i^t - \frac{\Delta\theta_i^2 \alpha \nabla_{\theta_i} \mathcal{L}(\hat{\theta}^t)}{2(\Delta\mathcal{L}_i^t + \alpha \nabla_{\theta_i} \mathcal{L}(\hat{\theta}^t)\Delta\theta_i)}, \tag{10}$$

where $\Delta\theta_i$ is $\hat{\theta}_i^t - \hat{\theta}_i^{t-1}$. The derivation of (10) is in the appendix. According to (10), the nominal loss model includes the SI [14] method as a special case ($\alpha = 0$). When $\alpha > 0$, it can be interpreted that the each nominal parameter value moves as much as a value proportional to $\nabla_{\theta_i} \mathcal{L}(\hat{\theta}^t)/\Omega_i$ from the trained parameter values ($\hat{\theta}^t$). Since this is identical to the Newton method using an approximated Hessian (Ω), the nominal parameter set can be interpreted as moving toward decreasing the loss function value of the current task in the trained parameter set. Besides, for a sufficiently small α, the nominal parameter set $\tilde{\theta}^t$ is guaranteed to have a smaller loss value than the trained parameter set $\hat{\theta}^t$:

Theorem 1: There exists $\alpha^* > 0$, where the following equation is always satisfied for α smaller than α^*: $\mathcal{L}^t(\tilde{\theta}^t) < \mathcal{L}^t(\hat{\theta}^t)$, s.t. $\tilde{\theta}$ is calculated using (10).

The proof is in the appendix. This theorem illustrates that the proposed method is less affected by the proxy loss function of the previous tasks, so that the nominal parameter set which shows better performance in the current task can be found. Figure 3 shows the proxy loss function and nominal parameter values for several cases. As in Theorem 1, each nominal parameter value has a loss value lower than $\mathcal{L}^t(\hat{\theta}^t)$ (equality condition is when $\alpha = 0$). Moreover, if the sign of the gradient and the sign of the change of the parameter value ($\hat{\theta}_i^{t-1} - \hat{\theta}_i^t$) are the same, the nominal parameter value is between $\hat{\theta}_i^{t-1}$ and $\hat{\theta}_i^t$ and the corresponding intensity (Ω_i) increases, and vice versa.

C. ALGORITHM

In this section, we introduce the learning strategy of the SBN and the AN, using the nominal loss model. For the model structure $z \in \{0, 1\}^{p\times(g-1)}$ based on the output of the AN, (4) is expanded as follows:

$$\min_{\theta} \mathbb{E}_{(x,y)\in\mathcal{D}^T}\left[\mathcal{L}^T(\theta, z; x, y) + \lambda \sum_{t=1}^{T-1} \mathcal{L}_p^t(\theta, z)\right] \quad s.t.\ z \sim f(\phi; x), \tag{11}$$

where x and y are an image and a label from the dataset $\mathcal{D}^T$, respectively, and ϕ is a parameter set of the AN (f). Since AN produces the probability value for the model structure, we should minimize the following mean loss function:

$$\min_{\theta,\phi} \mathbb{E}_{(x,y)\in\mathcal{D}^T, z\sim f(\phi;x)}\left[\mathcal{L}_{tot}^T(\theta; x, y, z)\right], \tag{12}$$

where $\mathcal{L}_{tot}^T$ indicates the total loss function including the loss for the task T and the proxy losses for the previous tasks. However, it is computationally intractable to calculate the expected value for all possible z. Inspired by [34], we approximate the expectation to the average of several examples sampled from the probability distribution based on the output of the AN:

$$\min_{\theta,\phi} \mathbb{E}_{(x,y)\in\mathcal{D}^T}\left[\sum_{z\in Z} \log p(z; f(\phi; x)) \frac{\mathcal{L}_{tot}^T(\theta; x, y, z)}{|Z|}\right], \quad s.t.\ Z \sim f(\phi; x), \tag{13}$$

where Z is a set of the sampled model structures from the probability $f(\phi; x)$, and $p(z; f(\phi; x))$ is the probability value for the model structure z under $f(\phi; x)$. The derivation is in the appendix. (13) can be computed efficiently and differentiated for ϕ and θ to be minimized with a broad optimizer of a deep neural network.

Since storing all the nominal loss models for the previous tasks is a waste of memory, we provide an online update method of the nominal loss model. It is enough to show that we can express two nominal loss models as one nominal loss model by mathematical induction. Assuming that $\Omega^{1:t-1}$ and $\tilde{\theta}^{1:t-1}$ represent the nominal loss model of the previous tasks and Ω^t and $\tilde{\theta}^t$ are the nominal loss model of the current task, the sum of two models can be represented as a single nominal loss model, as follows:

$$\sum_{i=1}^{n} \Omega_i^{1:t-1}(\theta_i - \tilde{\theta}_i^{1:t-1})^2 + \Omega_i^t(\theta_i - \tilde{\theta}_i^t)^2 + \text{Const.}$$
$$= \sum_{i=1}^{n}(\Omega_i^{1:t-1} + \Omega_i^t)\left(\theta_i - \frac{\Omega_i^{1:t-1}\cdot\tilde{\theta}_i^{t-1} + \Omega_i^t\cdot\tilde{\theta}_i^t}{\Omega_i^{1:t-1} + \Omega_i^t}\right)^2. \tag{14}$$

The algorithm of the proposed framework is summarized in Algorithm 1. For the first task, we train the SBN and the AN, alternately. When a new task comes in, to derive reasonable performance for the new task, the SBN is trained for a few epochs (less than 5), and then the AN is trained. Finally, the SBN is trained based on the learned AN.

Algorithm 1 Continual Learning With Adaptive Network Model Search and Nominal Loss Model

```
1: Input: $\mathcal{D}_{1:T}$, $S$, $\lambda$
2: Initialize: $\theta$, $\varphi \leftarrow$ Xavier-initializer
3: update $\theta$, $\phi$ minimizing only a task loss function
4: for $t$ = 2 to $T$ do
5:     Initialize $\Delta\mathcal{L} = 0,\ \Delta\theta = 0$
6:     for $s$ = 1 to $S$ do
7:         Update $\theta$ minimizing only a task loss function
8:     end for
9:     repeat
10:        Update $\phi$ w.r.t. (13)
11:    until Convergence
12:    repeat
13:        Save $\theta^s \leftarrow \theta$
14:        Update $\theta$ w.r.t. (13)
15:        Update $\Delta\mathcal{L} \leftarrow \Delta\mathcal{L} - \nabla_\theta\mathcal{L}^t \cdot (\theta^s - \theta)$
16:        Update $\Delta\theta \leftarrow \Delta\theta + (\theta - \theta^s)$
17:    until Convergence
18:    Derive $\Omega^t$ and $\tilde{\theta}^t$ using (10)
19:    Online update $\Omega^{1:t}$ and $\tilde{\theta}^{1:t}$ using (14)
20: end for
```

TABLE 1. Summary of the datasets.

Dataset	size	# dataset	# c.p.t.
permuted MNIST	32	60,000 / 10,000	10
SVNH / MNIST	32	133,257 / 36,032	10
CIFAR-10/100	32	100,000 / 20,000	10
mini-ImageNet	84	50,000 / 10,000	10

TABLE 2. Summary of setup for the scenarios.

Scenario	Backbone network	Epochs per each task	# task	Learning rate	Batch size	α in (10)	λ in (11)
10-permuted MNIST	Fc-net	1/1/15	10	$1e-3$	256	$1e-2$	$1e-1$
20-permuted MNIST	Fc-net	1/1/15	20	$1e-3$	256	$1e-2$	$1e-1$
SVNH / MNIST	Conv-net	1/1/15	2	$1e-3$	256	$1e-2$	$1e-1$
11-split CIFAR-10/100	Conv-net	1/2/60	11	$1e-3$	256	$1e-2$	$1e-2$
10-split mini-ImageNet	Conv-net	1/2/60	10	$1e-3$	256	$1e-2$	$1e-1$

IV. EXPERIMENTAL RESULTS

A. EXPERIMENTAL SETUP

We evaluated the proposed framework on several benchmark datasets in incremental learning scenarios: permuted MNIST [35], SVHN [36] / MNIST, 11-split CIFAR-10/100 [37], and 10-split mini-ImageNet.[2] Detailed information for each dataset is shown in Table 1, where # dataset is the number of images in the train and test sets, and # c.p.t. denotes the number of classes per each task. For the permuted MNIST scenario, we randomly generated a permutation matrix that changes the order of pixels in an image for each task. We tested two incremental learning problems involving 10 and 20 tasks in the permuted MNIST scenario. In the SVHN / MNIST scenario, the SVHN and MNIST datasets were trained sequentially. Since these two scenarios deal with similar classification problems for each task, we used a single-head network which shares the last fully-connected layers over the different tasks in the scenarios. The classes in the dataset were divided into n tasks in n-split scenarios. In particular, in the 11-split CIFAR-10/100 scenario, CIFAR-10 was learned as the first task and 10 classes extracted from CIFAR-100 were learned from the second task. The proposed method (OURS) was compared with our method without model search (OURS-), structural regularized approaches including EWC [13], SI [14], and MAS [12], a functional approach: LwF [11], a generative approach: GR [6] and a baseline (Base) which independently learns each task.

Table 2 shows the detailed experimental settings for learning the proposed algorithm, where # task is the number of tasks in the scenario. Permuted MNIST created several tasks by zero-padding a 28×28 image with a size of 32×32 and changing the order of pixels (i.e., 10-permuted MNIST contains 10 permutation matrices). The backbone network used the same network model in the previous study [14]. The selective backbone network (SBN) was first trained for one epoch, and then the attention network (AN) was trained by considering the loss function of the current task and the nominal loss model for previous tasks. The AN was learned faster than SBN, so it was possible to learn with a small training cost of 1 or 2 epochs. We fixed the relaxation degree of the gradient (α) as $1e-2$. Also, we set λ, which controls the balance between the current loss function and the proxy loss function for previous tasks, as $1e-1$ or $1e-2$.

[2]https://github.com/yaoyao-liu/mini-imagenet-tools

B. EXPERIMENTAL RESULTS

We measured the average performance (ACC) after learning all tasks and the ability to learn new tasks (AL) in the incremental learning scenarios. The AL is defined as the difference in performance between the method which learns each task independently (Base) and the method which learns a new task along with previous tasks (incremental learning method):

$$\mathrm{AL} = \sum_{t=2}^{T} \frac{C_t - B_t}{T-1}, \tag{15}$$

where B_t represents the accuracy of the task t using Base, C_t is the accuracy of the task t measured after learning task t using the incremental learning method, and T is the number of the tasks. Note that C_t denotes the accuracy before it is reduced by learning newer tasks ($> t$). If the knowledge of the previous tasks has a positive effect on learning the new task, the AL is measured as a positive number.

1) PERMUTED MNIST

For this scenario, the backbone network consisted of two 2000-dimensional fully connected layers and the 10-dimensional fully connected layer. ReLU activation was used after each layer. To configure the backbone network as the SBN, we divided the first two fully connected layers into eight sub-fully connected layers with 250-dimensional hidden layers. We set λ in (11) to 0.1. The results on the permuted MNIST with 10 and 20 tasks are in Table 3. In the scenario with 20 tasks, the proposed method has outstanding performance compared to other methods, while most methods show similar performance for 10 tasks. The proposed method is robust for cases with large numbers of tasks in the scenario. Besides, the regularization-based methods have low AL values. It is evidence that regularization of penalizing changes in parameters often hinders learning new tasks. Nevertheless, the proposed method has the highest AL value among regularization methods.

2) SVHN/MNIST

From this scenario, we used a backbone network consisting of four convolution layers and two fully connected layers. The first two convolution layers output 32 channels and the after two convolution layers output 64 channels. After every two convolution layers, max-pooling and dropout were performed. The sizes of output dimensions of fully connected layers are 512 and 10, in order. The first and subsequent two convolution pairs are each split into four sub-convolutions to design the SBN, and we set λ to 0.1. Table 3 shows the

TABLE 3. Comparison of average accuracy (ACC) and learning ability (AL) on five incremental learning scenarios.

Scenario	10-permuted MNIST		20-permuted MNIST		SVHN / MNIST		11-split CIFAR-10/100		10-split mini-ImageNet	
Method	ACC	AL	ACC	AL	ACC	AL	ACC	AL	ACC	AL
Base	98.28	-	98.17	-	96.27	-	72.50	-	56.68	-
LwF	88.85	**-0.13**	79.08	**-0.30**	89.90	-0.28	52.25	-1.63	48.01	**-2.77**
GR	95.58	-0.99	77.76	-1.17	85.21	**0.08**	49.58	-4.20	31.50	-3.13
EWC	**97.06**	-0.49	88.17	-0.54	92.14	-0.39	70.24	-0.22	46.14	-9.72
SI	96.35	-0.53	90.14	-0.53	91.83	-0.15	73.11	4.84	46.43	-5.27
MAS	97.02	-0.28	87.57	-0.53	92.83	-0.22	69.83	0.05	47.54	-7.71
OURS-	96.96	-0.40	92.68	-0.46	93.22	-0.46	74.24	5.44	47.11	-5.76
OURS	96.95	-0.35	**93.01**	-0.41	**94.04**	-0.35	**74.28**	**5.49**	**49.01**	-4.86

experimental results for SVHN / MNIST. OURS and OURS- exhibit the best performance. In the case of the AL, GR shows the best performance. The positive AL value of GR indicates that the generated SVHN images improve the learning ability of the MNIST dataset.

3) 11-SPLIT CIFAR-10/100

This scenario used the same backbone network as the previous scenario. However, this scenario was not configured as a single-head network; Each task has its own the last fully connected layer. We set λ to 0.01. In Table 3, OURS and OURS- show the best results in both ACC and AL. The proposed method even has outstanding performance compared to Base. The positive value of the AL with our methods represents that the previously trained network has a positive effect on learning new tasks in this scenario.

4) 10-SPLIT mini-ImageNet

In this scenario, we experimented with larger-sized images. We used the same backbone network and λ as the previous scenario. In Table 3, the proposed method shows the highest performance. The proposed method shows the best AL among regularization-based methods and the best ACC in the overall scenarios. The performance of GR is much lower compared to other methods. It is interpreted to be because mini-ImageNet is challenging to generate, unlike simple images which are relatively easy to learn a generative model.

C. ANALYSIS

1) FLOPs

To verify the efficiency of the proposed model search, we listed the FLOPs of the network for all scenarios in Table 4, where S.M. is the FLOPs of the selected model and C.R. is the compression ratio compared to the backbone network. Since the proposed method has to pass both the AN and the selected model, the compression ratio was calculated by comparing the sum of the FLOPs in the two networks with the FLOPs in the backbone network. The proposed method can perform tasks with average FLOPs of 64.28% to 82.61%. Interestingly, there is a positive correlation between the difficulty of a task (measured as average performance) and the compression ratio. Note that other competitors have the same or more FLOPs than backbone.

TABLE 4. FLOPs of networks in incremental learning scenarios.

Scenario	Backbone	AN	S.M.	C.R.
1	12.14M	1.54M	6.26M	64.28%
2	12.14M	1.54M	6.88M	69.34%
3	53.16M	7.59M	25.97M	63.12%
4	53.16M	7.59M	31.74M	73.99%
5	366.24M	52.28M	250.27M	82.61%

TABLE 5. Ablation results for permuted MNIST and split CIFAR dataset.

Dataset	random-wise	task-wise	OURS
20-permuted MNIST	76.21	91.80	93.01
11-split CIFAR-10/100	72.06	72.72	74.28

2) INPUT-WISE MODEL SEARCH

We compared the proposed input-wise model search with random-wise model search (random model structure for every input instances) and task-wise model search. We evaluated the experiment on 20-permuted MNIST and 11-split CIFAR-10/100 scenarios. The ACC for the methods is shown in Table 5. The random-wise model search shows comparable performance with the task-wise model search in the split CIFAR scenario. This indicates that training the network with all parameters at all times is not a favorable condition for preventing catastrophic forgetting. Besides, compared to the task-wise model search, our method shows much higher accuracy.

3) PERFORMANCE ACCORDING TO α

We compared the performance of the proposed methods with different α to verify whether gradient relaxation is necessary for the proposed algorithm and can increase the performance of each task. We measured the average performance for all tasks using the method without performing a model search for comparison with SI [14] ($\alpha = 0$). The experimental result is shown in Figure 4. If α does not sufficiently alleviate the gradient value ($\alpha > 0.2$), it can be seen that catastrophic forgetting cannot be prevented. In this case, as described in the paper, α is higher than the amount of change in the current loss function, so Ω is incorrectly calculated. For a sufficiently small α, the proposed method shows a consistently reliable performance, and as the α decreases, we verify that the performance is similar to that of SI.

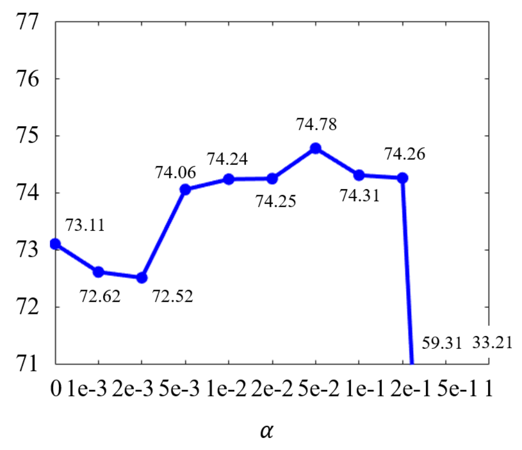


FIGURE 4. Comparison of average accuracy on 11-split CIFAR-10/100 dataset. We measured the accuracy for the proposed methods with different α. In this experiment, the proposed method did not contain the model search.

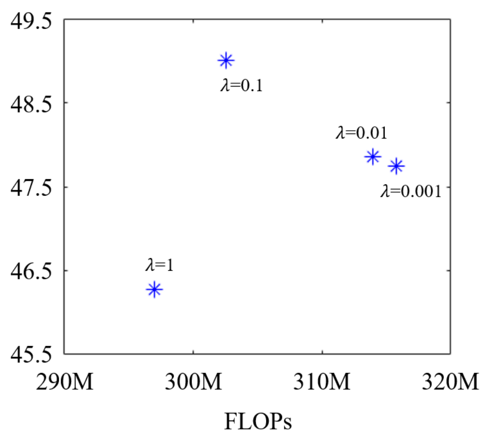


FIGURE 5. Comparison of average accuracy on 10-split mini-ImageNet dataset. We measured the accuracy for the proposed methods with different λ.

4) PERFORMANCE ACCORDING TO λ

We investigated the relationship between model selection and λ which controls the balance between the proxy loss for the previous tasks and the loss function for the current task. The FLOPs and average performance for each λ are shown in Figure 5. In this experiment, λ was set to 0.1 when learning SBN to confirm the relationship between model selection and λ clearly; Like other structural regularized methods, SBN's performance is sensitive to λ. It can be seen that the larger λ, that is, the more emphasis is placed on the proxy loss for the previous tasks, and the fewer parameter groups are selected when learning a new task. The original backbone network includes 366.24M FLOPs, and the proposed method can perform tasks using only 81% to 87% of the FLOPs of the backbone network. When λ is 0.1, it shows the best performance while using a smaller-sized model than when $\lambda = 0.01$, and 0.001. This result suggests that using a small model can be effective for incremental learning.

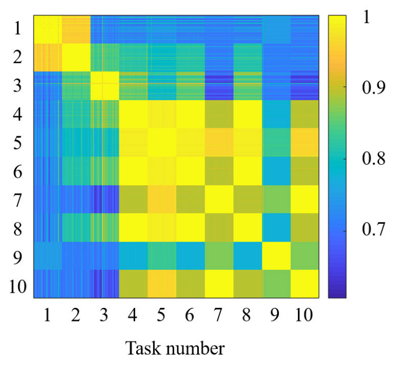


FIGURE 6. Correlation of model structure of sample images belonging to each task in 10-split mini-ImageNet.

5) CORRELATION OF MODEL STRUCTURES

We calculated the correlation for images sampled from mini-ImageNet (see Figure 6). The model structures are obtained by the trained AN in the 10-split mini-ImageNet scenario. Images belonging to the same task and images belonging to consecutive tasks have a high correlation. Since images belonging to the same task refer to the same nominal loss model and the nominal loss model is accumulated in subsequent tasks, it can be seen that similar models are often selected to prevent catastrophic forgetting.

V. CONCLUSION

In this study, we have proposed an adaptive model search method for incremental learning. The proposed framework consists of a selectable backbone network which can instantiate multiple internal models and an attention network (AN) which selects a suitable internal model according to a given input image. The AN is trained to select the internal model that minimizes the loss function of the current and previous tasks. The proposed model search is the first model search method which considers the current task performance so that our method can preserve the ability to learn new tasks compared to other methods. Meanwhile, we have designed a nominal loss model which derives the proxy loss function of the previous task without the previous dataset. The nominal loss model refers to the parameters with a lower loss value than the previously trained parameters, which has been analyzed theoretically. Besides, since the proposed framework performs the task using the internal model smaller than the backbone network, it enables memory-efficient inference. The proposed approach has been validated on incremental learning scenarios, while it has shown outstanding performance compared to existing incremental learning methods.

Future research should consider the potential effects of using rehearsal images from previous tasks. The proposed model search method can be applied to class-incremental learning problems using rehearsal images. Rehearsal images can be assisted in designing the approximated loss function and determining the importance of each parameter.

APPENDIX A
DERIVATION OF EQUATION (10)

According to the paper, the two conditions that the proxy loss for θ_i satisfies are:

$$\bullet\ \mathcal{L}^t_{p,i}(\hat{\theta}^{t-1}_i) - \mathcal{L}^t_{p,i}(\hat{\theta}^t_i) = \Delta\mathcal{L}^t_i$$
$$\bullet\ \nabla_{\theta_i}\mathcal{L}^t_{p,i}(\hat{\theta}^t) = \alpha\nabla_{\theta_i}\mathcal{L}^t(\hat{\theta}^t). \quad (16)$$

$\mathcal{L}^t_{p,i}(\theta_i)$ whose quadratic coefficient is Ω_i with the gradient value of the above condition at $\hat{\theta}^t_i$ is expressed as the following quadratic function:

$$\mathcal{L}^t_{p,i}(\theta_i) = \mathrm{C} + \alpha\nabla_{\theta_i}\mathcal{L}^t(\hat{\theta}^t)(\theta_i - \hat{\theta}^t_i) + \Omega_i(\theta_i - \hat{\theta}^t_i)^2, \quad (17)$$

where C is a constant value independent of θ_i. Substituting (17) for the first condition in (16), Ω_i is calculated as:

$$\begin{aligned}\Delta\mathcal{L}^t_i &= \mathrm{C} - \alpha\nabla_{\theta_i}\mathcal{L}^t(\hat{\theta}^t)\Delta\theta_i + \Omega_i\Delta\theta_i^2 - \mathrm{C}\\ \Omega_i &= \frac{\Delta\mathcal{L}^t_i + \alpha\nabla_{\theta_i}\mathcal{L}^t(\hat{\theta}^t)\Delta\theta_i}{\Delta\theta_i^2}, \end{aligned}\quad (18)$$

where $\Delta\theta_i$ is the amount of change in the parameter $(\hat{\theta}^t_i - \hat{\theta}^{t-1}_i)$. The nominal parameter set, $\tilde{\theta}_i$, is derived by substituting (18) for (17):

$$\begin{aligned}\mathcal{L}^t_{p,i}(\theta_i) &= \Omega_i((\theta_i - \hat{\theta}^t_i)^2 + \frac{\alpha\nabla_{\theta_i}\mathcal{L}^t(\hat{\theta}^t)}{\Omega_i}(\theta_i - \hat{\theta}^t_i)) + \mathrm{C}\\ &= \Omega_i(\theta_i - \hat{\theta}^t_i + \frac{\alpha\nabla_{\theta_i}\mathcal{L}^t(\hat{\theta}^t)}{2\Omega_i})^2 + \mathrm{C}'\\ &= \Omega_i(\theta_i - \tilde{\theta}^t_i)^2 + \mathrm{C}'\\ s.t.\ \tilde{\theta}^t_i &= \hat{\theta}^t_i - \frac{\alpha\nabla_{\theta_i}\mathcal{L}^t(\hat{\theta}^t)}{2\Omega_i}, \end{aligned}\quad (19)$$

where C$'$ is the another constant value independent of θ_i.■

APPENDIX B
PROOF OF THEOREM 1

The loss function for task t can be expressed by Taylor expansion as follows:

$$\mathcal{L}^t(\theta) = \mathcal{L}^t(\hat{\theta}^t) + \nabla_\theta\mathcal{L}^t(\hat{\theta}^t)^T(\theta - \hat{\theta}^t) + \frac{1}{2}(\theta - \hat{\theta}^t)^T H(\eta)(\theta - \hat{\theta}^t), \quad (20)$$

where $H(\cdot)$ is a hessian and η is a point on the line segment joining θ to $\hat{\theta}^t$. Then, we can derive the loss value on $\tilde{\theta}^t$ as following as:

$$\begin{aligned}&\mathcal{L}^t(\tilde{\theta}^t)\\ &= \mathcal{L}^t(\hat{\theta}^t) - \frac{\alpha}{2}\nabla\mathcal{L}^T G^{-1}\nabla\mathcal{L} + \frac{\alpha^2}{8}\nabla\mathcal{L}^T G^{-T}H^*G\nabla\mathcal{L}\\ &= \mathcal{L}^t(\hat{\theta}^t) - \frac{\alpha}{2}(\nabla\mathcal{L}^T G^{-T}G^T G^{-1}\nabla\mathcal{L}\\ &\quad -\frac{\alpha}{4}\nabla\mathcal{L}^T G^{-T}H^*G^{-1}\nabla\mathcal{L})\\ &= \mathcal{L}^t(\hat{\theta}^t) - \frac{\alpha}{2}v^t(G^T - \frac{\alpha}{4}H^*)v, \end{aligned}\quad (21)$$

where $\nabla\mathcal{L} = \nabla_\theta\mathcal{L}^t(\hat{\theta}^t)$, $H^* = H(\eta)$, G is a diagonal matrix whose i-th element is Ω_i, and $v = G^{-1}\nabla\mathcal{L}$. Since H^* is bounded and $G > 0$, $G^T - \alpha H^*/4 > 0$ for sufficiently small $\alpha > 0$. Then, $\mathcal{L}(\tilde{\theta}^t) < \mathcal{L}^t(\hat{\theta}^t)$ for sufficiently small α.■

APPENDIX C
DERIVATION OF EQUATION (13)

For θ, minimizing (12) is equal to minimizing the following objective function:

$$\begin{aligned}&\arg\min_\theta \mathbb{E}_{(x,y)\in\mathcal{D}^T, z\sim f(\phi;x)}\left[\mathcal{L}^T_{tot}(\theta; x, y, z)\right]\\ &\quad\approx \arg\min_\theta \mathbb{E}_{(x,y)\in\mathcal{D}^T}\left[\sum_{z\in Z}\mathcal{L}^T_{tot}(\theta; x, y, z)\right]. \end{aligned}\quad (22)$$

Since (12) is not differentiable with respect to ϕ, we introduce a log-trick method [34], inspired by [30]:

$$\begin{aligned}&\nabla_\phi\mathbb{E}_{(x,y)\in\mathcal{D}^T, z\sim f(\phi;x)}\left[\mathcal{L}^T_{tot}(\theta; x, y, z)\right]\\ &= \mathbb{E}_{(x,y)\in\mathcal{D}^T}\left[\sum_{\forall z}\nabla_\phi p(z,\phi)\mathcal{L}^T_{tot}(\theta; x, y, z)\right]\\ &= \mathbb{E}_{(x,y)\in\mathcal{D}^T}\left[\sum_{\forall z}\frac{\nabla_\phi p(z,\phi)}{p(z,\phi)}\mathcal{L}^T_{tot}(\theta; x, y, z)p(z,\phi)\right]\\ &= \mathbb{E}_{(x,y)\in\mathcal{D}^T}\left[\sum_{\forall z}\nabla_\phi \log p(z,\phi)\mathcal{L}^T_{tot}(\theta; x, y, z)p(z,\phi)\right]\\ &= \mathbb{E}_{(x,y)\in\mathcal{D}^T, z\sim f(\phi;x)}\left[\nabla_\phi \log p(z,\phi)\mathcal{L}^T_{tot}(\theta; x, y, z)\right]\\ &\approx \mathbb{E}_{(x,y)\in\mathcal{D}^T}\left[\sum_{z\in Z}\nabla_\phi \log p(z,\phi)\mathcal{L}^T_{tot}(\theta; x, y, z)\right], \end{aligned}\quad (23)$$

where $p(z,\phi) = p(z; f(\phi; x))$. ■

REFERENCES

[1] J. S. Rojas, A. Pekar, A. Rendon, and J. C. Corrales, "Smart user consumption profiling: Incremental learning-based OTT service degradation," *IEEE Access*, vol. 8, pp. 207426–207442, 2020.

[2] F. Feng, R. H. M. Chan, X. Shi, Y. Zhang, and Q. She, "Challenges in task incremental learning for assistive robotics," *IEEE Access*, vol. 8, pp. 3434–3441, 2020.

[3] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, vol. 24. Amsterdam, The Netherlands: Elsevier, 1989, pp. 109–165.

[4] R. Kemker and C. Kanan, "FearNet: Brain-inspired model for incremental learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[5] O. Ostapenko, M. Puscas, 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.

[6] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 2990–2999.

[7] Y.-X. Wang, D. Ramanan, and M. Hebert, "Growing a brain: Fine-tuning by increasing model capacity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2471–2480.

[8] J. Yoon, E. Yang, J. Lee, and S. Ju Hwang, "Lifelong learning with dynamically expandable networks," 2017, *arXiv:1708.01547*.

[9] S. S. Sarwar, A. Ankit, and K. Roy, "Incremental learning in deep convolutional neural networks using partial network sharing," *IEEE Access*, vol. 8, pp. 4615–4628, 2020.

[10] H. Jung, J. Ju, M. Jung, and J. Kim, "Less-forgetting learning in deep neural networks," 2016, *arXiv:1607.00122*.

[11] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
[12] 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. (ECCV)*, Sep. 2018, pp. 139–154.
[13] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassablis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.
[14] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3987–3995.
[15] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 4548–4557.
[16] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2001–2010.
[17] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle, "PodNet: Pooled outputs distillation for small-tasks incremental learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Glasgow, U.K.: Springer, 2020, pp. 86–102.
[18] A. Prabhu, P. H. Torr, and P. K. Dokania, "Gdumb: A simple approach that questions our progress in continual learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Glasgow, U.K.: Springer, 2020, pp. 524–540.
[19] 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. 3931–3940.
[20] 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.
[21] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback Control Theory*. Chelmsford, MA, USA: Courier Corporation, 2013.
[22] D. Park, S. Hong, B. Han, and K. M. Lee, "Continual learning by asymmetric loss approximation with single-side overestimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3335–3344.
[23] B. Camp, J. K. Mandivarapu, and R. Estrada, "Self-Net: Lifelong learning via continual self-modeling," *Frontiers Artif. Intell.*, vol. 3, p. 19, 2020.
[24] 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*.
[25] H. Jung, J. Ju, M. Jung, and J. Kim, "Less-forgetful learning for domain expansion in deep neural networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[26] D. Yin, M. Farajtabar, A. Li, N. Levine, and A. Mott, "Optimization and generalization of regularization-based continual learning: A loss approximation viewpoint," 2020, *arXiv:2006.10974*.
[27] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.
[28] K. Ahmed and L. Torresani, "MaskConnect: Connectivity learning by gradient descent," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 349–365.
[29] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani, "N2N learning: Network to network compression via policy gradient reinforcement learning," 2017, *arXiv:1709.06030*.
[30] 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.
[31] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 2017, pp. 2181–2191.
[32] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 3–18.
[33] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 2074–2082.
[34] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2000, pp. 1–7.
[35] Y. L. Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 1989, pp. 396–404.
[36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf
[37] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.

CHANHO AHN (Graduate Student Member, IEEE) received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His research interests include machine learning, multi-task learning, incremental learning, and computer vision.

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, subspace representation, and computer vision.

SONGHWAI OH (Member, IEEE) received the B.S. (Hons.), M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, in 1995, 2003, and 2006, respectively. Before his Ph.D. studies, he was a Senior Software Engineer at Synopsys, Inc., and a Microprocessor Design Engineer at Intel Corporation. In 2007, he was a Postdoctoral Researcher with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley. From 2007 to 2009, he was an Assistant Professor in electrical engineering and computer science with the School of Engineering, University of California at Merced, Merced. He is currently a Professor with the Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea. His research interests include robotics, computer vision, cyber-physical systems, and machine learning. He is an Associate Editor for the IEEE Transactions on Robotics and IEEE Robotics and Automation Letters.