Growing a Brain with Sparsity-Inducing Generation for Continual Learning

Hyundong Jin[1] Gyeong-hyeon Kim[1] Chanho Ahn[2*] Eunwoo Kim[1]
[1]School of Computer Science and Engineering, Chung-Ang University
[2]Samsung Advanced Institute of Technology (SAIT)
{jude0316, leonardkkh, eunwoo}@cau.ac.kr chanho.ahn@samsung.com

Abstract

Deep neural networks suffer from catastrophic forgetting in continual learning, where they tend to lose information about previously learned tasks when optimizing a new incoming task. Recent strategies isolate the important parameters for previous tasks to retain old knowledge while learning the new task. However, using the fixed old knowledge might act as an obstacle to capturing novel representations. To overcome this limitation, we propose a framework that evolves the previously allocated parameters by absorbing the knowledge of the new task. The approach performs under two different networks. The base network learns knowledge of sequential tasks, and the sparsity-inducing hypernetwork generates parameters for each time step for evolving old knowledge. The generated parameters transform old parameters of the base network to reflect the new knowledge. We design the hypernetwork to generate sparse parameters conditional to the task-specific information and the structural information of the base network. We evaluate the proposed approach on class-incremental and task-incremental learning scenarios for image classification and video action recognition tasks. Experimental results show that the proposed method consistently outperforms a large variety of continual learning approaches for those scenarios by evolving old knowledge.

1. Introduction

The ability of deep learning to incrementally accumulate knowledge for sequentially incoming tasks is an important capability required for modern AI algorithms. However, deep neural networks tend to lose previously acquired knowledge while learning new data. This limitation of deep neural networks, called catastrophic forgetting, has been investigated by previous studies [29, 17]. Catastrophic forgetting significantly degrades performance when data from previous tasks (task-incremental) or already-learned class data (class-incremental) cannot be accessed. Under those incremental learning scenarios, continual learning methods have toiled to preserve the previous knowledge.

Various approaches have been proposed for continual learning and are categorized into three types: regularization [2, 4, 20, 17, 48], data rehearsal [5, 26, 32, 35, 38, 47], and parameter isolation [1, 30, 28]. The conventional continual learning methods introduced regularization approaches to suppress changes in important parameters [48, 2] or input-output mappings from a network trained up to previous tasks [22, 7]. The network is trained in a way that contributable parameters to the performance of the previously trained model remain unchanged. On the other hand, data rehearsal methods [25, 35] address catastrophic forgetting by accessing old data. The old data can be obtained by storing some data from old tasks [35, 47] or leveraging a generative network [38]. However, the privacy of data and the difficulty of learning generative models limit their versatility. Recently, parameter isolation approaches [28, 30, 15] construct a disjoint set of parameters for each task, preventing it from mixing old and new knowledge. These methods learn a new set of parameters by associating all [28] or some helpful old knowledge [15].

Unlike other approaches, the parameter isolation approach is designed to prevent forgetting without requiring access to old data by disallowing updates to previous sets of parameters. Despite this advantage, these methods can limit the ability to learn new tasks by relying on parameters that were learned for older tasks and remain unchanged. We observe that this limitation arises when there is a low correlation between the old and new data. This raises the question: *In parameter isolation-based continual learning, is it possible to transform the knowledge from old tasks to fit newer tasks better?*

In this paper, we introduce a novel method for transforming prior knowledge into a form that can aid in learning new tasks. The approach involves reframing the parameter selection problem, which previous works [15] have addressed, as a generation problem. The proposed framework emphasizes a subset of pre-trained knowledge, selecting and highlight-

*This work was done independently, without any support from SAIT.

ing it for use in learning new tasks. Specifically, the framework consists of two networks: base network and hypernetwork. The base network constructs disjoint parameter sets for sequential tasks. The proposed transformer-based [42] hypernetwork generates a small set of influential parameters that significantly impact the loss. The impact of each element of generated parameters is evaluated by the difference in loss computed with and without its inclusion in the learning process. Also, generated parameters are conditioned on the information of the current task (task token) and the layer of the base network (layer embedding). The generated parameters transform the previous set of parameters in the base network to further accommodate new knowledge. In addition, compared to existing hypernetworks that generate a small set of parameters and apply it to all the layers [13, 44], the proposed hypernetwork adaptively generates parameters according to the layer configurations defined in the base network. Through this strategy, the quantity of layer embeddings for ResNet-50 is reduced by a substantial factor of 24. Furthermore, by leveraging a matrix decomposition technique to represent generated parameters compactly, we achieve a noteworthy reduction in the parameters necessary for the hypernetwork.

We evaluate the effectiveness of the proposed method in both class-incremental and task-incremental learning scenarios using classification benchmark datasets. Furthermore, we demonstrate the efficacy of the proposed method [28, 15] in these scenarios by applying it to video action recognition datasets [19]. Experimental results show that the proposed method performs better than its competitors for most learning scenarios while introducing minor memory overhead from the hypernetwork. The contributions of our work are mainly three-fold: (1) We present a novel approach to evolving previous knowledge in continual learning, aimed at reflecting the knowledge of new tasks. (2) We propose a transformer-based hypernetwork that generates customized parameters for new tasks while suppressing the generation of redundant parameters that have little impact on the loss. (3) Experimental results on diverse benchmarks for video action recognition and image classification demonstrate that ours outperforms its competitors with lower network capacity requirements to perform tasks.

2. Related Work

Continual Learning. Generally, continual learning methods can be classified into three types. Regularization methods [23, 7, 17, 48, 4, 2, 20] add a penalty term that restrains the previously learned knowledge from being updated when learning a new task. In order not to hurt previous knowledge, a new network is introduced to mimic the features obtained from the old network [22, 7]. While other regularization techniques, such as those based on the Fisher information matrix [17], change of loss [48], and variation of

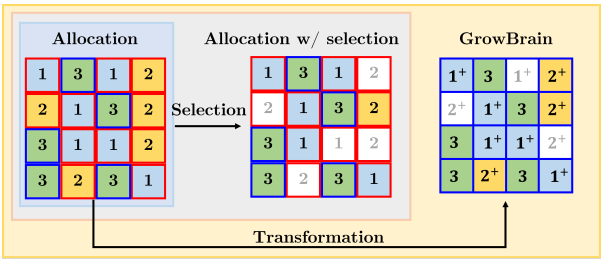

Figure 1. An illustration of the proposed method (right) and baseline methods (middle, left) for the third (new) task. The figure represents task-specific allocated parameters, represented by task numbers. Gray numbers represent the sets of discarded parameters from a selection method. The baseline methods do not update old knowledge (red boxes) while updating a new task (blue boxes). The proposed method transforms the old sets of parameters, represented by $+$, to fit the new task. Best viewed in color.

prediction [2], suppress the update of crucial parameters for each task, they experience significant performance decline as the number of tasks grows [12]. Data rehearsal methods [5, 26, 35, 33, 25, 38] rely on the availability of previous task data, which is achieved through selective storage of data [35, 47, 25]. These methods employ a few previous samples jointly with a new dataset [25, 38] or constraining the gradient based on the angle between that obtained from stored samples and the gradient obtained from the new samples [5, 26]. These methods incur a data imbalance problem due to a small number of stored previous tasks compared to a large amount of new task data. Parameter isolation methods [28, 1, 30, 15, 16] involve constructing disjoint sets of parameters for old tasks, respectively, and updating another set of parameters for a newer task by associating it with old parameters during training. This is usually done by using all previous sets of parameters [28], using a learnable mask to select some previous set of parameters [27], jointly selecting a location to allocate a new set of parameters [16], or searching a subnetwork with coarse-to-fine task association [15].

However, this approach can impede efficient learning because the parameters optimized for old tasks remain the same when learning newer tasks, meaning they need to absorb the information from the newer tasks. In contrast, the proposed method allows the old knowledge to grow by incorporating the knowledge from new tasks. The proposed method and its baselines are illustrated in Figure 1.

Hypernetworks. Hypernetworks generate parameters for a target network [13, 44, 39, 21, 34]. A recent work [34] proposes a method for generating parameters for layers with few parameters, such as activation of ResNet blocks or batch normalization layers. However, generating parameters for convolutional and linear layers that take most of the parameters in the base network can be challenging because it is difficult to represent the entire set of parameters from a

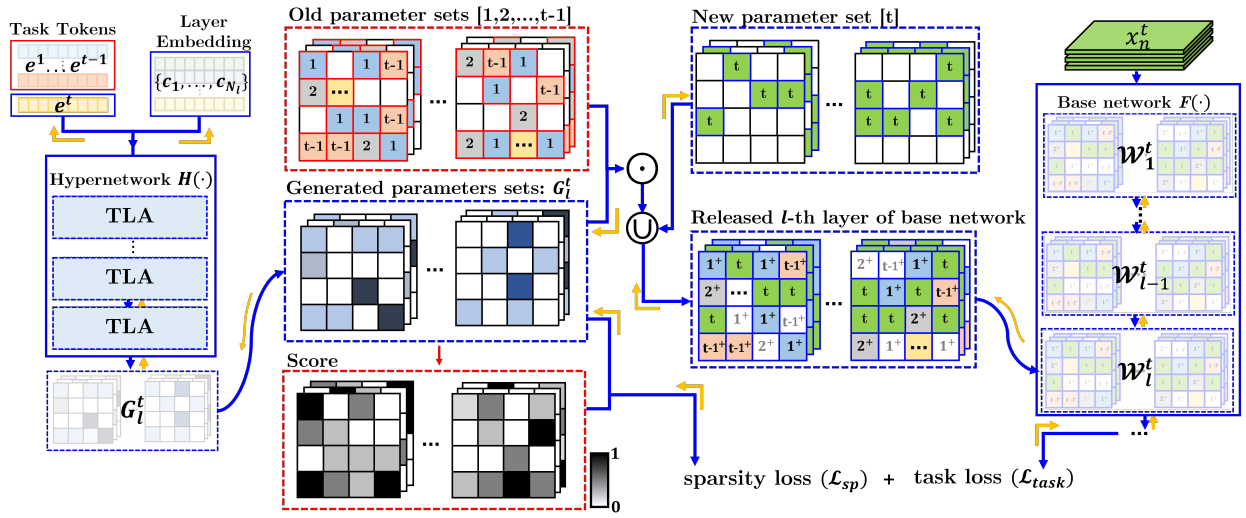

Figure 2. The proposed method for transforming the parameters of the l-th layer of the base network $F(\cdot)$ when learning the t-th task. The numbers in the grid indicate the indices of tasks. The hypernetwork $H(\cdot)$ generates parameters using the task-layer attention block (TLA), which discovers the relationship between the task token and layer embedding. The sparse parameters are generated while suppressing the generation of the redundant ones by evaluating the impact of parameters on the loss. The previous sets of parameters of the l-th layer are transformed by element-wise multiplication with G_l^t. The transformed parameters are combined with the current parameter set through the union operation ($\cup$) and released as a complete set of parameters to learn new knowledge. Best viewed in color.

hypernetwork. To avoid this, existing works generate small sets of parameters and concatenates them [13, 21]. Consequently, to generate a base network of large scale, allowing a large number of input embeddings corresponding to the network is inevitable due to the repetitive generation mechanism. In contrast to the approach, the parameters are generated according to the dimensions defined in the layers of the base network in our method. Note that as these dimensions expand, the corresponding demands on hypernetwork parameters also escalate. To address this, we employ a matrix decomposition technique for a compact representation of the required parameters in the hypernetwork.

Recently, in the continual learning literature, [44] and [10] learn a hypernetwork to generate parameters with task-specific embeddings for new tasks while preventing changes to previously generated parameters for previous tasks. The approaches produce dense parameters for the base network through a task-conditional hypernetwork. In contrast, the proposed transformer-based hypernetwork generates sparse parameters by measuring the effect of individual parameters on the loss. Furthermore, our method generates parameters to evolve previous knowledge to reflect new knowledge, whereas the aforementioned approaches generate parameters directly for the base network.

3. GrowBrain

3.1. Motivation

The base network $F(\cdot)$ constructs a disjoint set of parameters for each task under an iterative train-prune-retrain framework [28]. At a time step t, the base network is learned for the t-th task with the dataset $T^t = \{(x_n^t, y_n^t)\}_{n=1}^{N^t}$, where x_i^t is the i-th sample and y_i^t is corresponding target, respectively. Before training the task, we allocate parameters for the new task to the pruned positions in the base network trained for $t-1$ tasks. As a typical strategy, the allocated parameters W^t are learned from the help of all previous parameter sets [28]. After training, redundant parameters in W^t are pruned, and the unpruned parameters, $\bar{W}^t$, are retrained. Finally, after learning for t tasks, $F(\cdot; \mathcal{W}^t)$ is parameterized with $\mathcal{W}^t = \{\bar{W}^1, \cdots, \bar{W}^t\}$.

Although using previous knowledge can be beneficial in learning new tasks, there is also the possibility of negative knowledge transfer. If earlier tasks are irrelevant to a new task, leveraging their knowledge leads to suboptimal performance of the new task. A recent approach involving the selective use of helpful parameters [15] has been proposed. However, it is based on fixed old knowledge, which limits the model's ability to evolve in response to new information. Consequently, the expressiveness of the model remains constrained by its dependence on immutable prior knowledge. To resolve the issue, we propose a novel framework that transforms old parameters into those absorbing the knowledge of newer tasks. To selectively transform the parameter sets of old tasks $\{\bar{W}^1, \ldots, \bar{W}^{t-1}\}$ for the time t, we generate parameters by presenting a sparsity-inducing hypernetwork. The generated ones enhance the previous parameters to fit the new task better. The entire procedure of the proposed method is illustrated in Figure 2.

3.2. Evolving Old Knowledge

Our approach involves transforming previously learned parameters into more adaptable ones to incorporate knowledge from a new task while retaining the old knowledge. This allows for the evolution of existing knowledge to accommodate new information better. Let us denote the hypernetwork as $H(\cdot;\mathcal{H}^t)$ parameterized by $\mathcal{H}^t$ based on the transformer architecture [42]. Note that $\mathcal{H}^t$ is initialized with the parameters of the hypernetwork after training the previous task, $t-1$. Specifically, the hypernetwork receives the task token $e^t \in \mathbb{R}^D$ and the set of layer embeddings $\mathcal{C}_l = \{c_n \in \mathbb{R}^D | 1 \leq n \leq N_l\}$ and generates the parameter set G_l^t, where N_l and $l \in \{1 \dots L\}$ denote the number of embeddings for layer l and the layer index of the base network, respectively.

To generate G_l^t the task token e^t and layer embedding c_n from the set C_l are concatenated to construct the input of the hypernetwork, i.e., $z_{n,0}^t = \text{Concat}(e^t, c_n) \in \mathbb{R}^{2\times D}$. The input $z_{n,m-1}^t$ is fed into the m-th task-layer attention block (TLA) as

$$\begin{aligned} \bar{z}_{n,m-1}^t &= z_{n,m-1}^t + \text{MSA}(\text{LN}(z_{n,m-1}^t)), \\ z_{n,m}^t &= \bar{z}_{n,m-1}^t + \text{MLP}(\text{LN}(\bar{z}_{n,m-1}^t)), \end{aligned} \tag{1}$$

where $m \in \{1, ..., N_b\}$. N_b, LN, MSA, and MLP denote the number of TLA blocks, layer normalization, multi-head self-attention, and multi-layer perceptron, respectively. The structure of TLA is illustrated in Figure 3. MSA is defined for the input $\hat{z}_{n,m-1}^t \triangleq \text{LN}(z_{n,m-1}^t)$ as

$$\text{MSA}(\hat{z}_{n,m-1}^t) = W^O \text{Concat}(\text{head}_1, \dots, \text{head}_{N_h}). \tag{2}$$

$W^O \in \mathbb{R}^{2\times D}$ is a projection matrix and N_h denotes the number of heads in the MSA layer, where $\text{head}_i \triangleq \text{softmax}(Q_i K_i^T / \sqrt{D/N_h}) V_i$ and

$$Q_i = W_{Q_i} \hat{z}_{n,m-1}^t, \quad K_i = W_{K_i} \hat{z}_{n,m-1}^t, \quad V_i = W_{V_i} \hat{z}_{n,m-1}^t,$$

where W_{Q_i}, W_{K_i}, and W_{V_i} are projection matrices in the i-th head. The parameters for the l-th layer are generated as

$$\begin{aligned} G_l^t &= \text{Concat}\left(g(\theta_1^t), ..., g(\theta_{N_l}^t)\right) \\ &= \text{Concat}\left(H(z_{1,0}^t; \mathcal{H}^t), ..., H(z_{N_l,0}^t; \mathcal{H}^t)\right), \end{aligned} \tag{3}$$

where $g(\cdot)$ denotes the fully connected layer followed by a sigmoid function. $\theta_n^t \in \mathbb{R}^D$ is the first row of z_{n,N_b}^t, which acts as the final embedding in a transformer architecture [8].

To refine old knowledge to align with the knowledge of the t-th task, let us denote the set of old parameters in the l-th layer as $\bar{W}_l^{1:t-1}$. We obtain the evolved parameters $\hat{W}_l^{1:t-1}$ by fusing $\bar{W}_l^{1:t-1}$ and G_l^t as

$$\hat{W}_l^{1:t-1} = \bar{W}_l^{1:t-1} \cdot G_l^t, \tag{4}$$

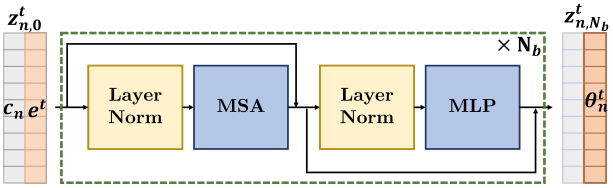

Figure 3. The task-layer attention block (TLA) consists of a layer normalization, multi-head self-attention (MSA), and MLP with a single hidden layer. z_{n,N_b}^t is generated by N_b stacked TLAs.

where we define $\cdot$ as the element-wise multiplication. The base network accommodates the evolved sets of parameters for $t-1$ tasks together with the t-th parameter set as $\hat{\mathcal{W}}^t = \{\hat{W}^1, \dots, \hat{W}^{t-1}, W^t\}$. Finally, the loss function to learn the t-th task using the evolved parameter sets becomes

$$\mathcal{L}_{task} = -\sum_{n=1}^{N^t} y_n^t \ \log(F(x_n^t; \hat{\mathcal{W}}^t))). \tag{5}$$

Another loss function to generate parameters from $H(\cdot;\mathcal{H}^t)$ similar to those learned in the previous task is defined as

$$\mathcal{L}_{reg} = \sum_{k=1}^{t-1} \sum_{l=1}^{L} \parallel \bar{G}_l^k - G_l^k \parallel_2^2, \tag{6}$$

where $\bar{G}_l^k$ represents the set parameters generated from the hypernetwork after training the task $t-1$.

3.3. Sparsity-Inducing Generation

To renew the network without redundant parameters, we propose to generate sparse parameters in the hypernetwork. As G_l^t is generated through the sigmoid function, they can be interpreted as the likelihood that each element of the old set of parameters is selected. Following the practice in [41], we minimize $\sum_{l=1}^{L} -\log(1-G_l^t)$ to achieve sparsity in G_l^t. However, to prevent G_l^t from becoming overly sparse, we ensure that parameters significantly impacting the loss remain survived. Inspired by the work in [24], we evaluate the impact of individual elements in G_l^t[1] to the loss. We compute the set of all entries of the score S_l^t by comparing the network losses computed with and without the inclusion of parameters in the learning process. To estimate the effect of removing a parameter on the loss function, we apply a Taylor expansion approximation of the network loss $\mathcal{L}$ for N samples. We obtain the k-th element of the score, s_k, as

$$\begin{aligned} s_k &= \mathcal{L}(\vec{G}_l^t - M_{\setminus(k)} \cdot \vec{G}_l^t) - \mathcal{L}(\vec{G}_l^t) \\ &\simeq -M_{\setminus(k)} \cdot \vec{G}_l^t \cdot \nabla_{\vec{G}_l^t} \mathcal{L} + \frac{1}{N} \sum_{n=1}^{N} \left(\frac{\partial \mathcal{L}_n}{\partial M_{\setminus(k)} \cdot \vec{G}_l^t} \right)^2, \end{aligned} \tag{7}$$

[1]We denote the vectorized form of G_l^t as $\vec{G}_l^t$ to represent individual entries.

where $M_{\setminus(k)}$ is an all-zero vector except that the k-th element is 1, and $\mathcal{L}_n$ is the loss of the n-th example. We introduce an additional term minimizing the discrepancy between the scores of the complete set of parameters (i.e., the original scores) and the scores of the sparse set of parameters produced as follows:

$$\mathcal{L}_{sp} = \sum_{l=1}^{L} -\log(1-G_l^t) + \left\| S_l^t \cdot \left(1 - \mathbb{1}(\vec{G}_l^t \neq 0)\right) \right\|_1, \quad (8)$$

where $\|\cdot\|_1$ and $\mathbb{1}(\cdot)$ denote the l_1-norm and the indicator function, respectively.

Total Loss. The proposed framework learns the t-th task with the three loss functions. To learn the t-th task, we minimize the cross-entropy loss $\mathcal{L}_{task}$ by updating W^t in the base network. Also, $\mathcal{H}^t$, e^t, and $\{C_l\}_{l=1}^L$ are updated from two losses, $\mathcal{L}_{task}$ and $\mathcal{L}_{sp}$. Last, we update $\mathcal{H}^t$ and $\{C_l\}_{l=1}^L$ by minimizing $\mathcal{L}_{reg}$ to preserve the previously generated parameters. The total loss is

$$\min_{W^t, \mathcal{H}^t, e^t, \{C_l\}_{l=1}^L} \mathcal{L}_{task} + \lambda_{sp}\mathcal{L}_{sp} + \lambda_{reg}\mathcal{L}_{reg}, \quad (9)$$

where λ_{sp} and λ_{reg} are parameters to balance the losses.

Scalability. A hypernetwork is designed to generate parameters for every layer of the base network, and the parameters were composed of compact segments of a consistent size [13, 21]. While it can generate parameters for all layers, this approach requires a substantial quantity of layer embeddings, particularly for deeper layers. To mitigate this, we adjust the scale of the segments for each layer depending on the number of channels. However, it incurs a memory overhead stemming from the substantial dimensions of the heads. To address this, we apply a matrix decomposition technique [39] to generate two reduced sets of low-rank r. By doing so, we achieve a reduction in memory requirements for layer embeddings, accompanied by a marginal increase in hypernetwork parameters. For detailed results, please refer to Section 4.5.1.

4. Experiments

4.1. Scenarios

We applied the proposed method, *GrowBrain*, to image classification and video action recognition problems under class-incremental and task-incremental learning scenarios. **Image classification.** We conducted experiments using ImageNet [6] and five fine-grained datasets including CUBS [45], Stanford Cars [18], Flowers [31], Wikiart [36], and Sketch [11], where each dataset is considered a task. We call the datasets ImageNet+FG for short. In addition, we demonstrated the proposed method with the CIFAR-100. We divided the dataset into subsets of classes based on the random order provided by [35, 9] and considered each subset a task. **Video action recognition.** We used a trimmed version of ActivityNet [3] and UCF-101 [40] video datasets. To conduct experiments, we divided each dataset into subsets containing randomly selected classes provided by [43] and treated each subset as a task. We summarize the datasets as mentioned above in Table 1.

Table 1. Datasets used in this work.

Image Dataset	# Train Images	# Test Images	# Classes
ImageNet [6]	1,287,167	50,000	1,000
CUBS [45]	5,994	5,794	200
Stanford Cars [18]	8,144	8,041	196
Flowers [31]	2,040	6,149	102
Wikiart [36]	42,129	10,628	195
Sketch [11]	16,000	4,000	250
CIFAR-100 [19]	50,000	10,000	100
Video Dataset	**# Train Videos**	**# Test Videos**	**# Classes**
ActivityNet [3]	10,024	4,926	200
UCF-101 [40]	9,280	2,720	101

Table 2. The specification of the hypernetwork.

Base network	Hypernetwork			
	Embed dim (D)	Hidden dim in MLP	# Heads (N_h)	# Blocks (N_b)
ResNet-50 ResNet-34	64	128	8	7
ResNet-18	32	64	8	5

4.2. Implementation details

Image classification. ImageNet+FG and CIFAR-100 were resized to 224×224 and 32×32 pixels, respectively. We applied random cropping and horizontal flip as augmentation. For those datasets, we used the ResNet-50 [14] and ResNet-18 as the base networks, respectively.

Video action recognition. We used TSN [46] with the ResNet-34 backbone network for video action recognition. Each video was split into three segments of equal duration, and a frame was randomly selected from each segment. The action class prediction was performed by aggregating the predictions of the selected frames with a consensus function. We used an average of predictions as a consensus function [46]. The video datasets were augmented with random cropping and horizontal flipping following [46]. We resized each frame of ActivityNet and UCF-101 to 224×224 pixels.

Table 2 reports the base networks and corresponding hypernetworks, which include the embedding dimension, the hidden dimension of the MLP, the number of heads, and the number of TLA blocks. We set the rank r for matrix decomposition as 7. We trained the base and hypernetwork using stochastic gradient descent. In all our experiments, we set λ_{reg} to 1 and λ_{sp} to 0.1, except ImageNet+FG where we set λ_{sp} to 1. We report the average results over five independent runs for all the experiments. To enable class-incremental learning in scenarios where the task identity is unknown, the proposed method first predicts the task index and then conducts classification. Following [44], we predict the task

Table 3. Performance of the class-incremental learning scenario using ImageNet and five fine-grained datasets (ImageNet+FG), where the average accuracy was measured after training each task. # P represents the average number of parameters ($\times 10^6$) to perform the tasks.

Method	ImageNet		CUBS		Stanford Cars		Flowers		Wikiart		Sketch	
	Acc	# P	Acc	# P	Acc	# P	Acc	# P	Acc	# P	Acc	# P
EWC-On	**76.1**	23.4	31.6	23.4	28.6	23.4	29.2	23.4	23.1	23.4	11.3	23.4
LwF	**76.1**	23.4	68.4	23.4	58.2	23.4	55.8	23.4	49.2	23.4	28.4	23.4
iCaRL	**76.1**	23.4	**70.2**	23.4	61.5	23.4	52.7	23.4	48.3	23.4	42.3	23.4
PackNet	75.7	12.5	62.8	13.3	65.2	14.0	66.5	14.6	54.3	15.2	45.3	15.7
Piggyback	**76.1**	23.4	60.1	22.7	54.8	22.2	56.4	22.0	49.7	21.7	47.4	21.4
WSN	74.7	12.5	59.6	12.5	42.2	12.5	29.2	12.5	31.0	12.5	25.0	12.5
H^2	75.7	12.5	69.5	13.0	67.3	12.9	68.4	13.0	61.8	12.9	48.7	12.6
GrowBrain	75.7	12.5	69.2	11.0	**69.3**	10.3	**71.3**	9.8	**64.3**	9.6	**60.1**	9.4

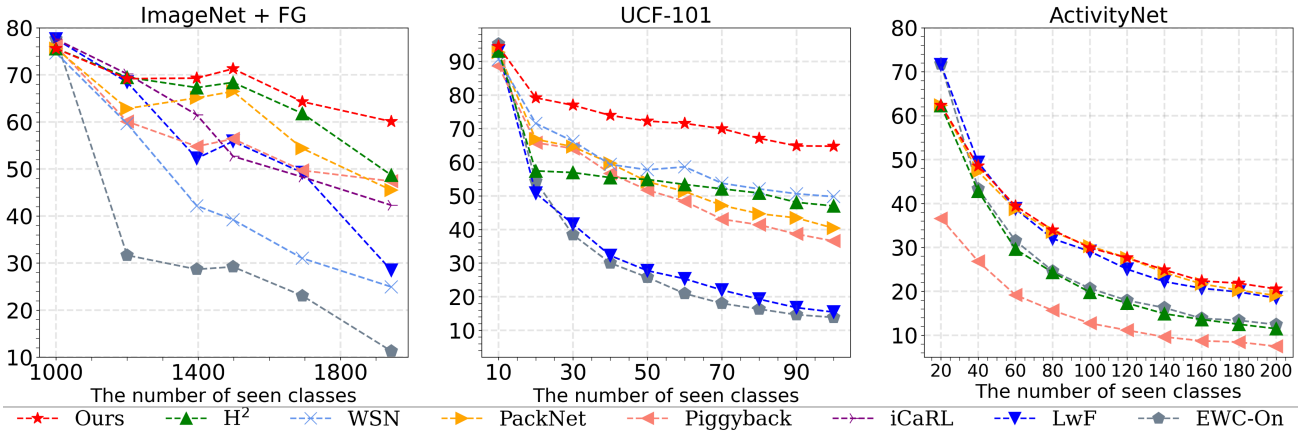

Figure 4. Accuracy of class-incremental learning for image classification on ImageNet+FG (left), video action recognition on UCF-101 (middle) and ActivityNet (right).

index using the network with the least entropy that has been learned thus far.

4.3. Class-incremental learning

4.3.1 Image classification

We evaluated the proposed method of evolving the previous knowledge to the class-incremental learning scenario using ImageNet+FG. We compared with regularization methods, EWC-On [37] and LwF [22], and data rehearsal methods, iCaRL [35], and parameter isolation methods, PackNet [28], Piggyback [27], WSN [16], and H^2 [15]. The data rehearsal methods stored 1,000 raw instances (taking 150 MB[2]). We applied the same task prediction mechanism as ours to the comparing methods, excluding iCaRL.

Table 3 shows the class-incremental learning results for ImageNet and five fine-grained datasets. Overall, the proposed method outperforms competitors (more detailed results are shown on the left of Figure 4). The regularization approaches, EwC-On and LwF, give unsatisfactory results with performance gaps of 48.8% and 31.7% compared to ours, respectively. LwF shows a 20.8% accuracy drop before and after learning Sketch, which is the most significant among the comparing methods. The data rehearsal method, iCaRL, shows a 17.1% performance drop from GrowBrain despite storing 1,000 old samples. PackNet shows large performance drops of 12.2% and 9.0% after training on WikiArt and Sketch, respectively. This is because negative interference occurs when using all previous sets of parameters optimized for different task domains. The methods of associating prior knowledge, Piggyback, WSN, and H^2, show notable performance drop (12.7%, 35.1%, and 11.4%, respectively) compared to the proposed method that improves the previous knowledge from the new task. The results demonstrate that the proposed approach surpasses a wide range of continual learning methods, confirming its superiority.

[2] We calculated the rehearsal memory assuming that each image (or frame of video) has $224 \times 224 \times 3$ pixels.

4.3.2 Video action recognition

We applied our method and the competitors to the video action recognition tasks using the benchmark sets proposed for continual learning [43]. For this experiment, iCaRL was excluded from the comparison since it demands approximately 2TB and 52GB of memory for replay [43] on ActivityNet and UCF-101, respectively. Since Piggyback requires a pre-trained backbone, we pre-trained the backbone model [14] using the validation set of UCF-101 for video action

Table 4. Results of the class-incremental learning scenario using the trimmed version of UCF-101 and AvtivityNet, where the *Avg* and *Last* accuracies for the time steps 5 and 10.

Method	UCF-101						ActivityNet					
	T^5			T^{10}			T^5			T^{10}		
	Avg	*Last*	# P	*Avg*	*Last*	# P	*Avg*	*Last*	# P	*Avg*	*Last*	# P
EWC-On	48.7	25.8	21.2	32.7	13.9	21.2	38.4	20.7	21.2	26.6	12.4	21.2
LwF	47.0	27.7	21.2	33.4	15.4	21.2	**44.1**	29.1	21.2	32.7	18.4	21.2
PackNet	59.3	54.2	11.5	52.5	40.4	15.2	42.4	30.2	11.5	32.4	19.0	15.2
Piggyback	65.4	51.7	20.4	53.5	36.5	20.4	22.2	12.7	20.5	15.6	7.5	20.4
WSN	69.1	58.8	10.1	61.1	49.7	10.5	-	-	-	-	-	-
H^2	63.6	54.8	8.3	56.9	47.0	11.4	37.4	22.5	3.4	26.8	13.3	2.1
GrowBrain	**79.3**	**72.2**	7.2	**73.5**	**64.7**	8.9	42.8	**29.9**	11.5	**33.2**	**20.6**	15.2

recognition experiments. We report the *Avg* and *Last* accuracies [9] for the time steps 5 and 10. Figure 4 shows the *Last* accuracy measured on the UCF-101 and ActivityNet datasets.

UCF-101. The results of the class-incremental learning scenario using the UCF-101 dataset are reported in Table 4 (left). The aspects of the experimental results on UCF-101 of the compared approaches are similar to those of image classification. Despite utilizing a large number of parameters, regularization approaches, EWC-On and LwF, give unsatisfactory performance compared to parameter isolation approaches. PackNet, which uses all parameters from previous tasks, performs less than other parameter isolation approaches, WSN and H^2, using selectively chosen parameters associated with previous knowledge. The proposed method shows the highest performance compared to other comparison methods. Ours outperforms WSN, H^2, PackNet, and Piggyback with the significant accuracy gap on the *Last* measure from 15.0% to 28.2%.

ActivityNet. Table 4 (right) reports the results on ActivityNet. WSN was excluded from the experiment because it did not give promising results due to the lack of data. Overall, the results of ActivityNet show a different pattern compared to those of UCF-101. The regularization method, LwF, outperforms the parameter isolation method, PackNet, with 1.7% and 0.3% *Avg* accuracy gap measured after training the 5-th and 10-th tasks. The proposed approach performs better than Piggyback, PackNet, and H^2 by large margins of 13.1%, 1.6%, and 7.3%, respectively, in terms of the *Last* accuracy measured after training 10-th task. The proposed method outperforms other competitors for image classification and video action recognition problems. In contrast to the UCF-101 dataset, where the proposal can acquire new knowledge by using a small number of parameters from previous tasks and achieve excellent results, the ActivityNet dataset requires previous knowledge to effectively train the current task due to its inadequate data (the number of videos per class is only half compared to UCF-101).

Table 5. Task-incremental learning scenarios using video action recognition and image classification datasets, where average accuracy and parameter consumption are reported.

Method	Image classification				Video action recognition			
	ImageNet+FG		CIFAR-100		UCF-101		ActivityNet	
	Acc	# P	Acc	# P	Acc	# P	Acc	# P
EWC-On	55.3	23.4	44.5	11.1	65.8	21.2	61.5	21.2
LwF	53.6	23.4	39.7	11.1	71.2	21.2	65.8	21.2
PackNet	80.2	15.7	82.1	10.4	92.2	15.2	65.3	15.2
Piggyback	82.2	21.4	75.6	10.5	86.6	20.4	42.1	20.4
WSN	61.7	12.5	84.6	5.6	92.7	10.5	-	-
H^2	82.8	12.6	85.9	1.1	94.5	11.4	44.0	2.1
GrowBrain	**83.0**	9.4	**86.8**	2.5	**96.7**	8.9	**66.9**	15.2

4.4. Task-incremental learning

Additionally, we conducted experiments on the task-incremental learning scenario where task-id is given during the inference step. In this scenario, we used ImageNet+FG and CIFAR-100 as the image classification datasets. Also, we used ActivityNet and UCF-101 for video action recognition. We divided CIFAR-100 into 10 tasks based on [35]. We followed the same experimental setting as in Tables 3 and 4. For the experiments of Piggyback on CIFAR-100, the backbone network was pre-trained on a randomly selected subset (40%) of the CIFAR-100 training set.

Image classification. Table 5 reports the results of task-incremental learning conducted on ImageNet+FG and CIFAR-100. Even if the regularization methods, EwC-On and LwF, try to maintain previous knowledge, they perform significantly less than the parameter isolation methods. GrowBrain consistently achieves high performance compared to other parameter isolation methods, PackNet, Piggyback, WSN, and H^2, in both image classification experiments (e.g., 0.2% and 0.9% performance gap on ImageNet+FG and CIFAR-100 with the most competitive method, H^2).

Video action recognition. Table 5 reports the results of task-incremental learning conducted on UCF-101 and ActivityNet. Overall, the results are similar to those of class-incremental learning scenarios. The regularization method, LwF, achieves higher performance than PackNet on Activ-

Table 6. Ablation study of the proposed method for image classification and video action recognition tasks, where average accuracy, the number of parameters for the base network and hypernetwork (in millions), and the number of layer embeddings are reported.

Method	ImageNet+FG			UCF-101		
	Acc	# $(F+H)$	# c_n	Acc	# $(F+H)$	# c_n
Ours	**60.1**	9.4 + 2.8	133	**64.7**	8.9 + 2.8	58
w/o transform [28]	45.3	15.7 + 0	0	40.4	15.2 + 0	0
w/o sparsity	58.1	15.7 + 2.8	133	64.0	14.1 + 2.8	58
w/o scalability	59.8	15.7 + 2.9	3262	**64.7**	11.0 + 2.9	614

ityNet. The proposed method outperforms both parameter isolation and regularization methods. Notably, our method outperforms WSN, PackNet, and H^2 with a performance gap of 4%, 4.5%, and 2.2%, respectively, on UCF-101.

4.5. Analysis

4.5.1 Ablation study

To demonstrate the effectiveness of our method, we conducted an ablation study. We compared ours by removing transformation of previous knowledge [28], sparsity-inducing generation (λ_{sp} = 0), and scalability of hypernetwork, i.e., channel adaptive segments and matrix decomposition. We applied these methods to class-incremental learning scenarios using the ImageNet+FG and the UCF-101 datasets. We report the *Last* accuracy measured after training all tasks, the average parameter consumption of the base network (F) and hypernetwork (H) to perform each task, and the number of layer embeddings c_n.

Table 6 reports the results of the ablation study. Our method outperforms that without transformation by performance gaps of 14.8% and 24.3% for those datasets, emphasizing that the evolution of prior knowledge is critical in enhancing performance. For video action recognition using UCF-101, the method without sparsity requires more parameters and performs less than the proposed method. The proposed method for ImageNet+FG achieves higher performance while taking fewer parameters than the non-sparse alternative. The reason is that sparsity-inducing hypernetwork prevents the generation of redundant parameters and regularizes the generated parameters to achieve generalization performance. The method without scalability requires 24 and 10.5 times larger layer embeddings on ImageNet+FG and UCF-101 than the proposed method, respectively, without accompanying performance improvement.

4.5.2 Impact of λ_{sp}

We further conducted a study investigating the impact of λ_{sp}, which regulates the sparsity of generated parameters. We experimented with varying values of λ_{sp} in the ImageNet+FG experiment under the class-incremental scenario using the same setup presented in Table 3. Figure 5 shows the parameter consumption (left) and average accuracy (right) with respect to different λ_{sp} after training all tasks. Ours with $\lambda_{sp} = 0.1$ uses a similar amount of parameters as the baseline [28] but outperforms the competing approach by leveraging enhanced prior knowledge. Notably, after training all tasks, $\lambda_{sp} = 1.0$ takes 9.4M parameters while showing 0.7% performance improvement compared to $\lambda_{sp} = 0.1$ consuming 15.7M parameters.

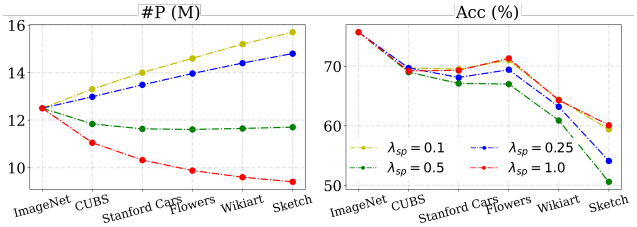

Figure 5. Parameter consumption and average accuracy of the proposed method based on varying λ_{sp} in class-incremental learning for ImageNet+FG.

5. Conclusion

We have proposed a novel approach, GrowBrain, to transform learned old knowledge to absorb the information of newer tasks in parameter isolation-based continual learning. GrowBrain reinforces the previous knowledge with a transformer-based hypernetwork that generates optimized parameters for new tasks while eliminating redundant parameters that show negligible impact on the loss. The proposed method overcomes the limitation of using fixed old knowledge that can constrain the ability to capture novel representations while taking fewer parameters. In the experiments, we have evaluated the proposal compared to a variety of continual learning methods in class- and task-incremental learning scenarios for image classification and video action recognition. Experimental results show that the proposed approach of leveraging transformed prior knowledge generates parameters adapted to the new task without losing the prior knowledge, resulting in significant performance improvements over its strong competitors.

Acknowledgements. This work was supported in part by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT2002-05, Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (2022-0-01341, Artificial Intelligence Graduate School Program(Chung-Ang University)), and BK21 FOUR (Fostering Outstanding Universities for Research) Program funded by Ministry of Education of Korea (No.I22SS7609062).

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3931–3940, 2020. 1, 2

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision*, pages 144–161. Springer, 2018. 1, 2

[3] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970, 2015. 5

[4] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision*, pages 556–572. Springer, 2018. 1, 2

[5] Arslan Chaudhry, Ranzato Marc'Aurelio, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *7th International Conference on Learning Representations, ICLR 2019*. 1, 2

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 5

[7] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5138–5146, 2019. 1, 2

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 4

[9] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9285–9295, 2022. 5, 7

[10] Benjamin Ehret, Christian Henning, Maria Cervera, Alexander Meulemans, Johannes Von Oswald, and Benjamin F Grewe. Continual learning in recurrent neural networks. In *International Conference on Learning Representations*, 2020. 3

[11] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on Graphics*, 31(4):1–10, 2012. 5

[12] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018. 2

[13] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2, 3, 5

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 5, 6

[15] Hyundong Jin and Eunwoo Kim. Helpful or harmful: Inter-task association in continual learning. In *European Conference on Computer Vision*, pages 519–535. Springer, 2022. 1, 2, 3, 6

[16] Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark Hasegawa-Johnson, Sung Ju Hwang, and Chang D Yoo. Forget-free continual learning with winning subnetworks. In *International Conference on Machine Learning*, pages 10734–10750. PMLR, 2022. 2, 6

[17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017. 1, 2

[18] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561. IEEE, 2013. 5

[19] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 2, 5

[20] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *Advances in Neural Information Processing Systems*, 30:4652–4662, 2017. 1, 2

[21] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. DHP: Differentiable meta pruning via hypernetworks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 608–624. Springer, 2020. 2, 3, 5

[22] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016. 1, 2, 6

[23] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 2

[24] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*, pages 7021–7032. PMLR, 2021. 4

[25] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2544–2553, 2021. 1, 2

[26] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6470–6479, 2017. 1, 2

[27] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision*, pages 67–82, 2018. 2, 6

[28] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 1, 2, 3, 6, 8

[29] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[30] Zichen Miao, Ze Wang, Wei Chen, and Qiang Qiu. Continual learning with filter atom swapping. In *International Conference on Learning Representations*, 2021. 1, 2

[31] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 5

[32] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11321–11329, 2019. 1

[33] Jaeyoo Park, Minsoo Kang, and Bohyung Han. Class-incremental learning for action recognition in videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13698–13707, 2021. 2

[34] Dripta S Raychaudhuri, Yumin Suh, Samuel Schulter, Xiang Yu, Masoud Faraki, Amit K Roy-Chowdhury, and Manmohan Chandraker. Controllable dynamic multi-task architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10955–10964, 2022. 2

[35] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 1, 2, 5, 6, 7

[36] Babak Saleh and Ahmed Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *International Journal for Digital Art History*, (2), 2016. 5

[37] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018. 6

[38] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2994–3003, 2017. 1, 2

[39] Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10753–10764, 2021. 2, 5

[40] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 5

[41] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. AdaShare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020. 4

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 4

[43] Andrés Villa, Kumail Alhamoud, Victor Escorcia, Fabian Caba, Juan León Alcázar, and Bernard Ghanem. vCLIMB: A novel video class incremental learning benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19035–19044, 2022. 5, 6

[44] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. In *8th International Conference on Learning Representations*, 2020. 2, 3, 5

[45] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 5

[46] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2740–2755, 2018. 5

[47] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019. 1, 2

[48] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995, 2017. 1, 2