DKT: Diverse Knowledge Transfer Transformer for Class Incremental Learning

Xinyuan Gao[1], Yuhang He[2]*, Songlin Dong[2], Jie Cheng[3], Xing Wei[1]*, Yihong Gong[1]
[1]School of Software Engineering, Xi'an Jiaotong University
[2]Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
[3]ACS Lab, Huawei Technologies, Shenzhen, China
{gxy010317,hyh1379478,dsl972731417}@stu.xjtu.edu.cn
chengjie8@huawei.com, {weixing,ygong}@mail.xjtu.edu.cn

Abstract

In the context of incremental class learning, deep neural networks are prone to catastrophic forgetting, where the accuracy of old classes declines substantially as new knowledge is learned. While recent studies have sought to address this issue, most approaches suffer from either the stability-plasticity dilemma or excessive computational and parameter requirements. To tackle these challenges, we propose a novel framework, the Diverse Knowledge Transfer Transformer (DKT), which incorporates two knowledge transfer mechanisms that use attention mechanisms to transfer both task-specific and task-general knowledge to the current task, along with a duplex classifier to address the stability-plasticity dilemma. Additionally, we design a loss function that clusters similar categories and discriminates between old and new tasks in the feature space. The proposed method requires only a small number of extra parameters, which are negligible in comparison to the increasing number of tasks. We perform extensive experiments on CIFAR100, ImageNet100, and ImageNet1000 datasets, which demonstrate that our method outperforms other competitive methods and achieves state-of-the-art performance. Our source code is available at https://github.com/MIV-XJTU/DKT.

1. Introduction

Deep neural networks have demonstrated notable success in the domain of class-fixed classification problems, wherein the object classes are predetermined and constant throughout the training and testing phases [12, 13]. Nevertheless, in practical scenarios, these models are frequently employed in an ever-changing and dynamic environment, necessitating the incorporation of capabilities to enable them to learn and identify new classes that arise contin-

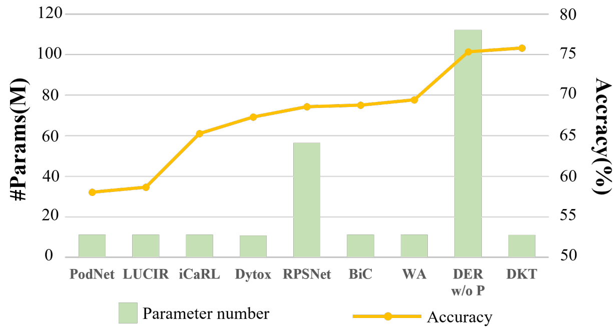

Figure 1. **The accuracy and parameters number comparisons of different methods on CIFAR100 10-step**. We report the final parameter number and accuracy. The height of the green pillars represents the number of final parameters, the polyline represents the average accuracy of different methods. We can see that DKT surpasses state-of-the-art methods with much fewer parameters.

uously. This challenge, commonly known as the Class-Incremental Learning (CIL) problem [4, 27], is of utmost significance.

Numerous recent studies [9, 10, 15, 27, 36, 41, 44, 45] have endeavored to address the challenges associated with Class-Incremental Learning (CIL). Among these methods, some [15,27,41,45] have adopted the concept of knowledge distillation [14], which entails transferring prior knowledge from a teacher model to a student model, to retain the previous knowledge encoded in the output logits of the network. Meanwhile, other methods [10, 36, 44] have employed dynamic expandable networks to overcome the CIL issues. These techniques involve dynamically augmenting the network architectures, such as feature extractors, by utilizing supplementary parameters and memory. Despite recent advancements in addressing the CIL problem, several challenges persist. Firstly, knowledge distillation techniques, as demonstrated in the literature [14], exhibit significant feature degradation [44], leading to reduced performance when transferring knowledge from prior to new tasks. Sec-

* Corresponding authors

ondly, networks must be stable enough to preserve existing knowledge [23,26] while simultaneously exhibiting plasticity to acquire new information, creating a stability-plasticity dilemma [3]. Lastly, dynamic expandable networks demand significant additional parameters, memory storage, and computational resources, hindering their practical application in real-world scenarios.

We introduce a new approach, named the Diverse Knowledge Transfer Transformer (DKT), to address the aforementioned challenges. DKT comprises two innovative attention blocks and a duplex classifier. Firstly, to mitigate feature degradation and catastrophic forgetting, we propose two novel attention blocks: the General Knowledge Transfer Attention Block (GKAB) and the Specific Knowledge Transfer Attention Block (SKAB). These attention blocks can transfer previous knowledge by utilizing a unified task-general token and a set of task-specific tokens from a token pool. For each task, the token pool initializes a new task-specific token to accumulate task-specific knowledge and update the unified task-general token storing the general knowledge of previous tasks. Unlike other dynamic expandable networks that use a feature extractor network, we initialize a task-specific token for each task, which is a 1×384 trainable vector. This design reduces the number of extra parameters and computational requirements significantly. We demonstrate the relationship between parameters and performance in Figure 1. Notably, DKT achieves state-of-the-art performance with only 1/10 of the parameters of the competitive DER w/o P method [44] and significantly outperforms other representative methods. Secondly, to address the stability-plasticity dilemma, we propose a *duplex classifier* comprising a stability classifier to maintain the model's stability on old categories and a plasticity classifier to learn the knowledge of new categories. Additionally, we propose a cluster-separation loss to pull features belonging to the same categories together and push features between old and new tasks apart. This encourages the model to learn diverse task-specific knowledge in different tasks.

We conduct extensive experiments on three widely-used image classification benchmarks, namely CIFAR100, ImageNet100, and ImageNet1000, to showcase the effectiveness of our proposed method. We compare our DKT with other state-of-the-art methods, including Dytox [10] and DER [44]. Dytox is the first attempt to utilize the transformer architecture for CIL, while DER uses extra parameters to achieve state-of-the-art performance. Our proposed approach outperforms Dytox [10] and sets a new state-of-the-art performance surpassing DER [44]. Our ablation study confirms the efficacy of our proposed method. In summary, our key contributions include:

- We propose a novel framework, DKT, that comprises the GKAB and SKAB attention blocks to facilitate diverse knowledge transfer and mitigate catastrophic forgetting in continual learning scenarios.
- We introduce a *duplex classifier* that enables a model to maintain stability in recognizing old categories while retaining plasticity in learning new categories.
- We develop a *cluster-separation loss* that clusters features belonging to the same categories and discriminates features between old and new tasks to encourage the model to learn diverse task-specific knowledge.
- We conduct extensive experiments on three benchmarks, and our approach achieves a new state-of-the-art performance with fewer parameters on the CIL benchmarks.

2. Related Work

First we briefly review how existing incremental learning solves the catastrophic forgetting problem and then introduce the current situation of the vision transformer.

2.1. Incremental Learning

According to recent studies, the most significant research interest of incremental learning is solving catastrophic forgetting. These methods can be divided into rehearsal, architectural, and regularization methods.

Regularization methods can be divided into two main sides, designing loss functions to limit the changes in critical parameters during the learning of new tasks like EWC [16] and using distillation to prevent the modification of models like LwF [18].

Architectural methods aimed at addressing catastrophic forgetting focus on assigning various model substructures. Traditional methods are fixing some parameters of the previous model, such as PackNet [21] and CCGN [1].

Rehearsal methods allow the model to store a subset of old data to replay. iCaRL [27] selects samples using herding and trains a nearest-class-mean classifier. BiC [41] adds a bias correction layer to alleviate the unbalanced outputs between old classes and new classes. WA [45] focuses on Weight Aligning to balance the outputs. TOPIC [32] and ERL [7] utilize topology preservation to maintain old-class knowledge. Recent work [2] proposes cross-space clustering and controlled transfer to cluster the feature. Our cluster-separation loss is inspired by this work. Usually, rehearsal methods can get a better performance in continual learning so recent works [10, 36, 43, 44] try to combine the rehearsal methods with architectural methods.

Dynamic Expandable Networks mean that the model creates a new network architecture when facing a new task. Recently, DER [44] designs a new model which can dynamically expand the representation by adding an extra feature extractor for every task. Then the features are concatenated

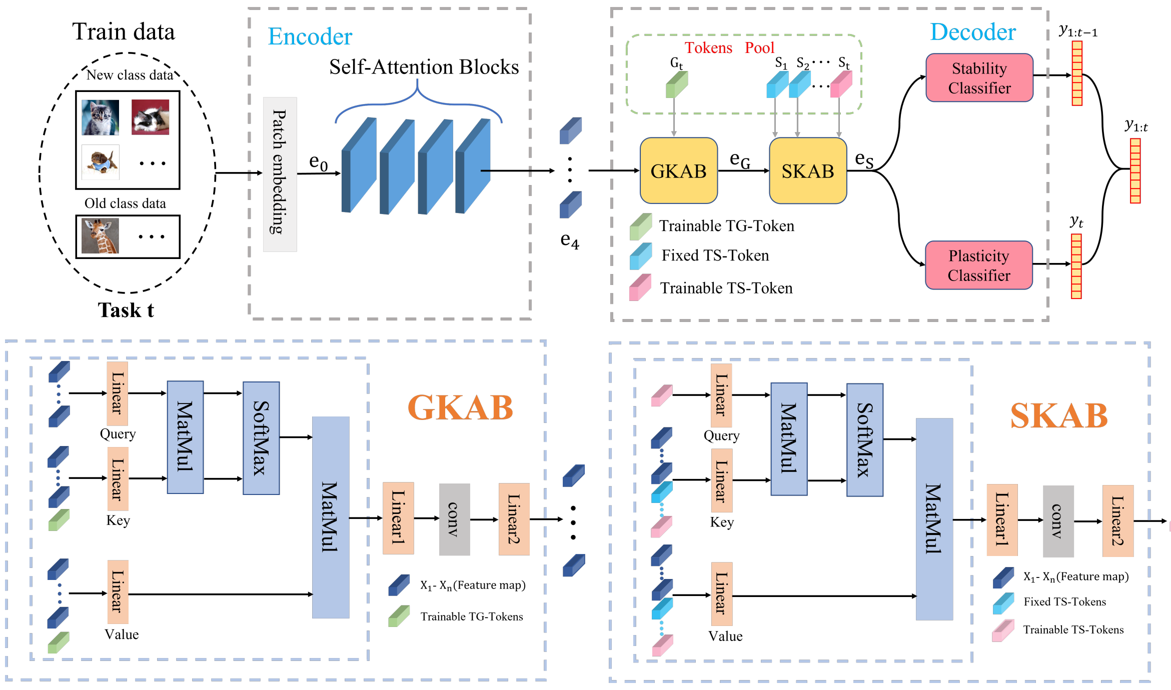

Figure 2. In task t, we first initialize two tokens (task general token and task specific token) in token pool. Subsequently, the image is processed through patch embedding and partitioned into multiple patches, which are then input into four SABs to obtain feature map e_4. e_4 is concatenated with G_t as the key and value for GKAB. The resulting tensor e_G is concatenated with S_1 to S_t as the key and value, and S_t is used as query. The output tensor e_S is fed into both the stability classifier and plasticity classifier independently. Subsequently, the outputs from the two classifiers are concatenated into a single output, which serves as the final output. In addition, we present the GKAB and SAKB structures in the figure, with green, blue, and pink rectangles indicating that they are selected from the token pool. Blue rectangles denote fixed tokens, while green and pink rectangles indicate trainable tokens during model training.

and fed into a single and expandable classifier. DER [44] obtains state-of-the-art performance and shows the advantages over the classical methods based on knowledge distillation [15, 27, 41, 45]. However, this method has three inevitable defects: (i) In the real situation, the continual expansion of the feature extractor makes the overhead unaffordable. (ii) It uses the HAT [29] procedure to prune the booming parameters, which needs a lot of computing resources. Although dynamic networks can reach a wonderful performance, these defects hamper the application in the real world and further improvement. Our work designs a novel architecture to meet the challenge.

2.2. Vision Transformers

Transformer [35], firstly is designed for machine translation tasks. Original transformers have encoder and decoder layers and quickly become state-of-the-art models for most natural language processing tasks [6, 24]. After NLPs, researchers demonstrated that transformers can perform great in computer vision by splitting the image into small tokens [8]. Based on its potential ability in classification, a lot of works developing from ViT [8] have been designed, including Deit [33], Swin [20], RVT [22] and so on. These improvements focus on various aspects, including designing a better architecture [20, 40], getting more knowledge from tokens [28,30], modifying the training procedures [31] and so on. Although ViTs [8,20,22,33,42] get great success in classification, using ViT [8] for CIL is not deep and successful as other computer vision fields. Dytox [10] uses task tokens, a divided classifier to alleviate catastrophic forgetting. Method [38, 39] depend on a large pre-train model (like ViT-B) with a prompt pool instead of training from the beginning like our methods.

Different from previous works [10,37,38], we define two novel tokens instead of a prompt pool to get general knowledge in previous tasks and specific knowledge in every task. We transfer the knowledge by GKAB and SKAB instead of classical cross-attention. Besides our model was trained from scratch, without the need for a large pre-trained model or a large-scale pre-training dataset. These differences show our knowledge transfer is novel.

3. Method

3.1. Problem Setting

In a class-incremental learning paradigm, we aim to incrementally adapt a unified model to new obtained classes. For a dataset, we define a set of tasks T_1, T_2,...,T_t, where T_t

= $\{({x_i}^t, {y_i}^t)\}_{i=1}^N$ is the t-th task with N samples, and ${x_i}^t$ and ${y_i}^t$ is the i-th images and its label. C_t is a set of categories of the training task T_t. There is no overlap between the categories of different tasks so that $\forall i, j,\ C_i \cap C_j = \varnothing$. Only a few samples stored in a memory buffer from previous classes and C_t are available for the current task. The model is evaluated on a combination of test sets $\mathbf{Z}^{1\sim t} = \mathbf{Z}^1 \cup \cdots \cup \mathbf{Z}^t$ and is expected to recognize all the classes $\mathbf{C}_{1\sim t} = \mathbf{C}_1 \cup \cdots \cup \mathbf{C}_t$.

3.2. Overall Framework

Our model consists of two primary components: an encoder for feature extraction and a decoder for alleviating catastrophic forgetting. In task t, given an input ${x_i}^t$, patch embedding outputs the tensor $e_0 \in \mathrm{R}^{M\times D}$, where M denotes the number of patches and D represents the dimension. e_0 is fed into Self-Attention Blocks (SABs) to extract the feature.

$$\begin{aligned} e_l^{'} &= e_l + SA_l(LN(e_l)), \\ e_{l+1} &= e_l^{'} + MLP_l(LN(e_l^{'})), \end{aligned} \tag{1}$$

where SA is the Self-Attention layer and MLP is the Multi-Layer Perceptron, which are both in SAB. We repeat the operation from $l = 0$ to $l = 3$ and the final SAB outputs e_4. e_4 is fed into GKAB to mix the task general knowledge and current knowledge from the task-general token. The output of the GKAB module, $e_G \in \mathrm{R}^{M\times D}$, is then passed to the SKAB module to acquire task-specific knowledge through training with task-specific tokens. The SKAB module outputs $e_S \in \mathrm{R}^{1\times D}$, which is then used as input to both the stability classifier and plasticity classifier, respectively.

$$\begin{aligned} y_{1:t-1} &= \text{s-clf}(e_S), \\ y_t &= \text{p-clf}(e_S). \end{aligned} \tag{2}$$

$y_{1:t-1}$ and y_t are merged as a single one $y_{1:t}$ to be the final output. Initially, we train the model on the base class training set with the binary-cross-entropy loss. Then we incrementally expand the task-specific token in the *Token Pool* for each task and update the task-general token. When a task is completed, we merge the stability classifier and plasticity classifier as a new stability classifier and create a new plasticity classifier to adapt to the new task. Moreover, we utilize a novel loss function, named cluster-separation loss, to cluster the features belonging to the same classes and discriminate the features between old and new tasks. We elaborate on our method in the following subsections.

3.3. Diverse Knowledge Transfer

To mitigate catastrophic forgetting by leveraging knowledge from previous tasks, we introduce two Attention Blocks to transfer task-general and task-specific knowledge to the current task.

Token Pool. We introduce a Token Pool to store both task-specific and task-general tokens. The task-general token $\in R^{1\times 1\times D}$ is employed to capture the general knowledge and is initialized randomly before the first task. In each task, it is updated to adapt to changes in the task's general knowledge. We establish a new task-specific token $\in R^{1\times 1\times D}$ for every task to store the specific knowledge in each task. Assuming training has 10 steps, the pool only has 4224 parameters in the end and the average parameter is 2304. Compared with the overall parameters (about 11 million), extra parameters almost can be ignored unlike the DER [44]. So our method has more practical value than other dynamic expandable networks [36, 44].

General Knowledge Transfer. To transfer the general knowledge to the current task, we define a novel attention block, called GKAB in Figure 2. We initialize a task-general token G_0 in the first task to gain general knowledge, which is then updated in the new task ($G_0 \to G_1 \to \cdots$). GKAB focuses on transferring the general knowledge in the task-general token to the current task to alleviate catastrophic forgetting. For a new task t, we concatenate the feature map e_4 from the encoder with task-general token G_t:

$$z_t = [e_4, G_t]. \tag{3}$$

After concatenation, token z_t is used as key and value in GKAB:

$$\begin{aligned} Q_i &= W_q e_4, \\ K_i &= W_k z_t, \\ V_i &= W_v z_t, \\ A_i &= Softmax(Q_i \cdot K_i^T / \sqrt{D/h}), \\ O_i &= W_o A_i V_i + b_o, \end{aligned} \tag{4}$$

where D means the embedding dimension. The h is the number of heads. We utilize the feature map e_4 to calculate self-attention in order to acquire current knowledge. Since there is a task-general token in z_t, e_i needs to compute the cross-attention with G_t to transfer the general knowledge in the previous tasks to the current task, which forces the attention map to mix the task general knowledge and current task knowledge. Extra task general knowledge can help the model to reduce forgetting. The task-general token is trainable in every task to update the general knowledge. Moreover, the task-general token is only added in key and value, the knowledge transfer does not affect the self-attention of e_i, which ensures GKAB's ability to extract the feature.

Specific Knowledge Transfer. SKAB focuses on transferring the specific knowledge in the previous tasks to reduce forgetting. Different from the GKAB, SKAB only use a new task-specific token S_t instead of a feature map as the query. The output e_S collects the knowledge from other task-specific tokens and the feature map in the current data stream. For a new task t, we combine the e_G with task-specific tokens = $S_1 ... S_t$:

$$z_t^{'} = [S_1, ..., S_t, e_G]. \quad (5)$$

After concatenating, tokens $z_t^{'}$ is fed into SKAB:

$$\begin{aligned} Q_i &= W_q S_t, \\ K_i &= W_k z_t^{'}, \\ V_i &= W_v z_t^{'}, \\ A_i &= Softmax(Q_i \cdot K_i^T / \sqrt{d/h}), \\ O_i &= W_o A_i V_i + b_o. \end{aligned} \quad (6)$$

Similar to GKAB, d means the embedding dimension and h is the number of heads. Unlike GKAB, SKAB does not employ the e_G as a query but a new task specific token S_t. Query computes the cross-attention with all task-specific tokens and feature map e_G. Thus the output $e_S \in R^{1 \times 1 \times D}$ mixes all previous task-specific knowledge and current task knowledge. Building upon this design, SKAB emphasizes on computing cross-attention to collect current knowledge and transfer task-specific knowledge. This approach aids the current model in recollecting previous tasks and mitigating the occurrence of catastrophic forgetting. Only one query guarantees that the e_S can be used to classify in classifiers without extra computing. Furthermore, SKAB has a linear complexity as opposed to quadratic due to the absence of self-attention computation for the feature map.

3.4. Duplex Classifier

In order to achieve a balance between model plasticity and stability, we propose a straightforward yet effective classifier, named the Duplex Classifier, which comprises both a stability classifier and a plasticity classifier. The stability classifier is made of a linear projection $\{W_{t-1}, b_{t-1}\}$, with $W_{t-1} \in R^{C_{1:t-1} \times D}$ and $b_{t-1} \in R^{C_{1:t-1}}$. and the plasticity classifier is made of a linear projection $\{W_t, b_t\}$, with $W_t \in R^{C_t \times D}$ and $b_t \in R^{C_t}$.

$$\begin{aligned} y_{1:t-1} &= \sigma(W_{t-1} LN(e_S) + b_{t-1}), \\ y_t &= \sigma(W_t LN(e_S) + b_t). \end{aligned} \quad (7)$$

For a feature e_S generated from the SKAB in task $t > 0$, it is passed through the plasticity classifier and the stability classifier, respectively. During training, the stability classifier is fixed by freezing the $\{W_{t-1}, b_{t-1}\}$. Since the old samples in a memory buffer can be accessed in the current task, the stability classifier maintains the performance in the old categories by limiting the change of feature e_S of the old samples. Besides, the plasticity classifier is trainable to learn new categories to maintain the plasticity of the model. Merge the two outputs $y_{1:t-1}$ and y_t as a single one to be the final output of the model.

To address the bias caused by the unequal number of images from old and new categories, both the stability classifier and plasticity classifier are trained during fine-tuning. Upon completion of a task, the plasticity classifier and stability classifier are combined to create a new stability classifier. Subsequently, the model generates a new plasticity classifier for the upcoming task's classes.

3.5. Loss Function

The loss function consists of three different parts: (1): the classification loss L_{clf}, a binary-cross entropy, used to classify the new data, (2): the distillation loss includes two portions, the knowledge distillation L_{kd} [14], (3): the cluster-separation loss L_{cs} that we propose. The total loss is in the following:

$$L = (1 - \alpha) L_{clf} + \alpha L_{kd} + \mu L_{cs}. \quad (8)$$

where α corresponds to the fraction of the number of old classes over the number of new classes $\frac{|C_{1:t-1}|}{|C_{1:t}|}$. μ is hyper-parameters, which is 0.003 for all tasks.

Cluster-Separation loss is inspired by [2]. We extended the method and emphasized the differences between old and new tasks to promote task-specific tokens to preserve diverse knowledge. The cluster-separation loss aims to promote feature discrimination between old and new tasks in order to diversify task-specific knowledge, and cluster features that belong to the same classes.

For pairs of samples x_i and x_j which are randomly selected from the current data, x_i is fed into the previous model and x_j is fed into the current model. We define the feature extractor of the previous model as F^{t-1} and the current model as F^t. A_{ij} represents the relationship of x_i and x_j.

$$L_{cs} = \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{B} (1 - cos(F^t(x_i), F^{t-1}(x_j))) * A_{ij}. \quad (9)$$

The relationships between x_i and x_j can be divided into two sides. (1): When x_i and x_j belong to the same classes, A_{ij} is 4. The model is forced to minimize the cosine distance between $F^t(x_i)$ and $F^{t-1}(x_j)$ to cluster the features belonging to the same classes. (2): If x_i and x_j belong to different classes, there are two situations: one is that when x_i and x_j belong to old tasks and new task respectively, A_{ij} is -4 to force the model to maximize the cosine distance between $F^t(x_i)$ and $F^{t-1}(x_j)$ to increase the difference between old tasks and new task to force the task-specific knowledge to be more diverse; the other is that when x_i and x_j both belong to the new task or old tasks, A_{ij} is -1 to

Methods	5 step			10 step			20 step		
	#Paras	**Avg**	**Last**	**Paras**	**Avg**	**Last**	**#Paras**	**Avg**	**Last**
LUCIR [15]	11.22	62.77±0.82	46.85	11.22	58.66±0.71	43.39	11.22	58.17±0.30	40.63
iCaRL [27]	11.22	71.14±0.34	59.62	11.22	65.27±1.02	50.74	11.22	61.20±0.83	43.75
BiC [41]	11.22	73.10±0.82	61.53	11.22	68.80±1.20	53.54	11.22	66.48±0.32	47.02
WA [45]	11.22	72.81±0.28	60.18	11.22	69.46±0.29	53.78	11.22	67.33±0.15	47.31
PodNet [9]	11.22	66.70±0.64	51.52	11.22	58.03±1.27	41.05	11.22	53.97±0.85	35.02
Dytox global [10]	10.73	70.98	57.92	10.73	67.33	51.68	10.73	67.30	48.45
DKT	11.03	**76.88**±0.18	**66.46**	11.03	**75.83**±0.38	**63.04**	11.03	**74.08**±0.31	**58.56**

Table 1. Results on **CIFAR100** over three different steps setting. We use the global memory accuracy reported by Dytox in erratum-distributed.md. We report the last parameter number (*#Paras*), the average accuracy (*Avg*), the Last accuracy (*Last*).

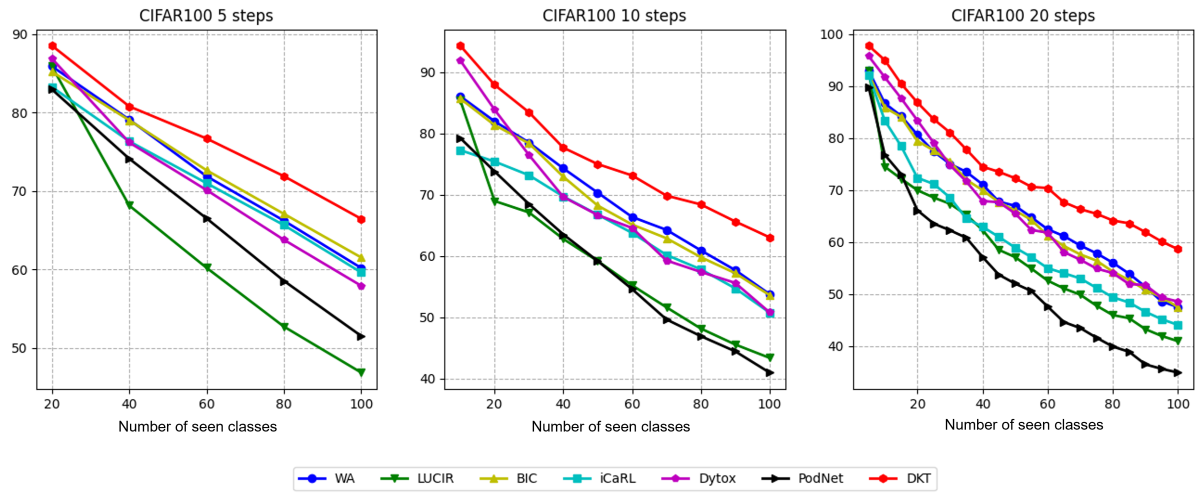

Figure 3. **Performance evolution on CIFAR100**. We report the top-1 accuracy after the learning task. Left is 5 steps on CIFAR100. The middle is 10 steps in CIFAR100. Right is 20 steps in CIFAR100.

separate features from different categories to improve the performance of clustering.

4. Experiments

4.1. Experiment Benchmark and Implementation

Datasets. Our experiment is based on three common incremental learning datasets: CIFAR100 [17], ImageNet100/1000 [5]. **CIFAR100**: CIFAR100 is a small dataset, which consists of 100 classes with 500 images per class for training and 100 images per class for evaluation. **ImageNet100**: ImageNet100 is composed of 100 classes chosen from the ImageNet1000 dataset randomly like previous methods [10, 36, 44]. **ImageNet1000**: ImageNet1000 has more than 1.2 million RGB images for training and 50000 RGB images for validation.

Benchmark. To verify the effectiveness of our methods, we test our methods on CIFAR100, ImageNet100, and ImageNet1000 without initialization classes. We design different settings like previous works [10, 44, 45]. The standard continual setting in ImageNet has 10 steps: we add 10 new classes per step in ImageNet100 and 100 new classes per step in ImageNet1000. In CIFAR100, we train the 100 classes with 5, 10, 20 classes per step. We compare our model with other methods by top-1 accuracy in CIFAR100 and ImageNet, and top-5 accuracy in ImageNet. We validate our method in three standards. We report the number of parameters after the final step (*#Paras*), the last accuracy after the final step (*Last*), and the average accuracy (*Avg*) that we average the accuracy of each step.

Implementation Details. We have 4 Self-Attention Blocks, 1 GKAB, 1 SKAB, and 1 Duplex Classifier. For the model, the embedding dimension is set to 384 and Self-Attention has 12 attention heads. To compare with other classical methods which are using ResNet18 [11] as the backbone, we use the SABs modified from RVT [22] to satisfy the requirement of parameters. This architecture allows the model to train on a small dataset (like CIFAR100). The model in CIFAR100 has 11.03M parameters, slightly less than ResNet18 (11.22M) [11]. Also, the patch size

	ImageNet100 10 steps					ImageNet1000 10 steps				
		top-1		top-5			top-1		top-5	
Methods	**#Paras**	**Avg**	**Last**	**Avg**	**Last**	**#Paras**	**Avg**	**Last**	**Avg**	**Last**
iCaRL [27]	11.22	67.11	50.81	83.60	63.80	11.68	38.40	22.70	63.70	44.00
BiC [41]	11.22	65.13	42.33	90.60	84.40	11.68	-	-	84.00	73.20
WA [45]	11.22	68.60	54.84	91.00	84.10	11.68	65.67	55.60	86.60	81.10
PodNet [9] [44]	11.22	64.03	45.43	84.06	68.18	11.22	–	–	–	–
DyTox global [10]	11.01	71.85	57.94	90.72	87.98	11.36	68.14	**59.75**	87.03	**82.93**
DKT	11.78	**78.20**	**68.66**	**93.72**	**88.72**	11.81	**70.44**	58.26	**88.46**	81.98

Table 2. **Results on ImageNet-100 and ImageNet-1000 datasets**, learned with 10 steps of respectively 10 and 100 new classes. We show the top-1 accuracy and top-5 accuracy in Table. We use the global memory accuracy reported by Dytox in erratum-distributed.md. Similar to CIFAR100, We report the last parameter number (*#Paras*), the average accuracy (*Avg*), the Last accuracy (*Last*).

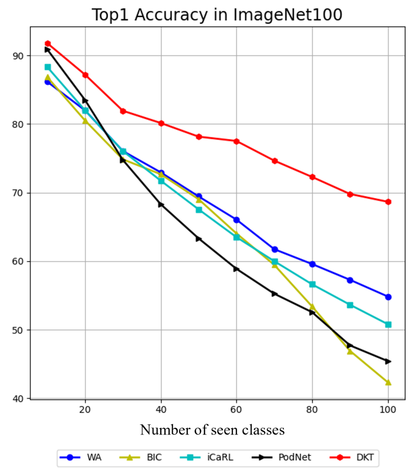

Figure 4. We report the performance evolution on ImageNet100, where DKT significantly outperforms other methods. Note that at the initial step, our model has performance comparable to other baselines. The performance in the last task shows that our method achieves a better performance in reducing catastrophic forgetting.

of ImageNet is larger so the parameters are 11.78M and 11.81M, slightly larger than ResNet18 (11.22M). Besides, we utilize rehearsal methods to store 2000 images as a fixed memory buffer for CIFAR100 and ImageNet100, and 20000 images for ImageNet1000, to compare with other methods fairly [9, 10, 15, 27, 41, 45]. More information will be provided in the Appendix.

4.2. Quantitative Results

CIFAR100. Table 1 summarizes the results of the CIFAR100-B0 benchmark. Experiments show that our method far outperforms other classical methods with similar parameters in all three settings on CIFAR-100. In 'Avg', DKT surpasses the second-best methods by up to **3.78%**, **6.37%** and **6.75%** and Dytox [10] which uses the vision transformer as the backbone by up to **5.90%**, **8.50%** and **6.75%**. In 'Last', DKT surpasses the second-best methods by up to **4.93%**, **9.50%** and **10.11%** and Dytox [10] by up to **8.54%**, **11.36%** and **10.11%**. These Experiments prove that DKT applies the Vision Transformer to the CIL problem effectively and gets state-of-the-art performance compared with previous methods with a similar parameter number. It is worth noting that as the number of steps increases, which means that reducing the forgetting becomes more difficult, the margin between DKT and the second-best methods continuously increases. Moreover, compared with other methods, the last accuracy has significantly improved simultaneously. The above phenomenons show that the improvement of performance is achieved by alleviating catastrophic forgetting, which fully demonstrates the effectiveness of our method on solving CIL problems to alleviate catastrophic forgetting.

ImageNet. Table 2 summarizes the results of the ImageNet100/1000-B0 benchmark. On ImageNet100, DKT achieves an average accuracy of **78.2%** and the last accuracy of **68.66%** in the top-1 setting, which both significantly surpass previous models with similar parameters and obtain the state-of-the-art performance. In comparison, the second-best method Dytox [10] achieves an average accuracy of 71.85% and 57.94%. Our DKT outperforms the Dytox [10] with about **6.35% (71.85%→78.2%)** for the 'Avg' and **10.72% (57.94%→68.66%)** for the 'Last'. In top-5 setting, DKT outperforms the second-best method by **2.72% (91.00%→93.72%)** for the 'Avg' and **0.74% (87.98%→88.72%)** for the 'Last'. Specifically, we report the performance evolution displayed in the top-1 setting in Fig 4. It is worth noting that in the beginning, DKT has a similar performance compared to other baselines. As the tasks increase, DKT still has a satisfactory performance compared with other methods which fall into serious catastrophic forgetting. This phenomenon proves that DKT has a strong ability to prevent forgetting on ImageNet100.

On ImageNet1000, in 'Avg' accuracy, DKT achieves **70.44%** and outperforms the state-of-the-art method Dytox in top-1 setting with about **2.30% (68.14%→70.44%)** for the 'Avg', and performs similar to Dytox [10] for the 'Last'. Critically, DKT is significantly above other baselines.

	CIFAR100				ImageNet100		ImageNet1000	
	5 steps		10 steps		10 steps		10 steps	
Methods	**#Paras**	**Avg**	**#Paras**	**Avg**	**#Paras**	**Avg**	**#Paras**	**Avg**
RPSNet [25]	60.60	70.50	56.50	68.60	–	–	–	–
Simple-DER [19]	–	–	–	–	–	–	28.00	66.63
DER w/o P [44]	56.10	76.80	112.27	75.36	112.27	77.18	116.89	68.84
DER [44]	–	75.55	–	74.64	–	76.12	–	66.73
DKT	**11.03**	**76.88**	**11.03**	**75.83**	**11.78**	**78.20**	**11.81**	**70.44**

Table 3. We compare our model with other methods which contain much more parameters. Results all come from their respective papers. DER w/o P is for the DER without pruning. DER needs to set sensitive hyperparameters and a large amount of computation to prune the booming parameters. Its reported parameters count was an average over all steps so we can not get the number of the final parameters. We can see that DKT surpasses other methods with much fewer parameters to obtain state-of-the-art performance.

4.3. Comparison with large Parameters Methods

We show the comparison results between DKT and other methods which have more Parameters in Table 3. We report the final parameters and the average accuracy in the 5 and 10 steps setup on CIFAR100, 10 steps on ImageNet100 and 10 steps on ImageNet1000. Even though other methods have much more parameters than our model, we obtained state-of-the-art performance on several benchmarks. On ImageNet100, DKT outperforms DER w/o P [44] by **1.02%** (**77.18%**$\rightarrow$**78.20%** in Avg), which contains 10 ResNet18 in parallel and 112.27M parameters. On CIFAR100, DKT both reach the best performance, surpassing DER [44] by **1.33%** and **1.29%** in the 5 and 10 steps setup and obtain state-of-the-art performance. On ImageNet1000, DKT obtains state-of-the-art performance and surpasses DER w/o P [44] and Simple-DER [19] by **1.60%** and **3.81%**.

It is worth noting that the parameter number of DER boosts when tackling a large number of tasks (56.10M in 5 steps vs 112.27M in 10 steps). Thus DER needs a large amount of computation and adjusts the complex hyperparameters to prune the boosting parameters. Since DER reports the average parameter number over all steps instead of the final parameter number (necessarily much higher). Based on this phenomenon, we have not reported the parameter number of DER in the table. Besides, when DER faces complex datasets (like ImageNet1000), the pruning becomes less efficient and more complex. Pruning leads to more severe accuracy loss in ImageNet1000 (68.84%$\rightarrow$66.73%) compared with other simpler datasets (CIFAR100 and ImageNet100). The average parameters number doubles while tackling the same amount of tasks on ImageNet1000 and ImageNet100 (reports 7.67M vs 14.52M). In contrast, DKT can handle a large number of tasks with only a few extra parameters (+0.004% per step). Moreover, DKT has a better ability to handle complex datasets and obtains state-of-the-art performance in ImageNet1000.

Baseline	GKAB	SKAB	CS Loss	D-Clf	**Avg**	**Last**
✓					67.20	43.64
✓	✓				69.43	47.34
✓	✓	✓			71.70	51.57
✓	✓	✓	✓		73.85	57.93
✓	✓	✓	✓	✓	75.83	63.04

Table 4. **Ablations** of the different components in DKT. We report the accuracy in 10 steps on **CIFAR100**. Baseline combines with five basic SABs, one cross-attention block, a single classifier and loss except for cluster-separation loss.

4.4. Ablation Study

To highlight the effectiveness of our contribution points, we design a series of experiments in Table 4. We divide our model into a baseline vision transformer model and several various components. We modify the fifth SAB into GKAB and add SKAB, Duplex Classifiers (D-Clf), and Cluster-Separate loss (CS Loss) in turns.

Baseline combines with five basic SABs, a cross attention block [34], a single classifier, bce loss and kd loss. As can be seen, the baseline obtains 43.64% accuracy in the last, which means that the model falls into serious catastrophic forgetting.

We modify the fifth SAB into GKAB and add SKAB in turns, which significantly improves the performance of reducing forgetting. Then, we add cluster-separation Loss to the model, which significant improvement in performance. Finally, we add the Duplex Classifier to balance the plasticity and stability of the mode. As you can see we succeeded in improving the stability of the model without hampering its plasticity of the model too much.

5. Conclusion

In the paper, we propose DKT, a novel structure consisting of 4 SABs, 1 GKAB, 1 SKAB, and 1 duplex classifier. GKAB and SKAB transfer the general knowledge and the specific knowledge in each task to the current training process to alleviate catastrophic forgetting. To solve the stability-plasticity dilemma, we design a simple but useful classifier to reach a better stability-plasticity trade-off. Besides, we define a novel cluster-separation loss to cluster the features belonging to the same categories and discriminate the feature between old and new tasks to force the task-specific knowledge to be more diverse. Extensive experiments on three incremental learning benchmarks show that our method obtains state-of-the-art performance.

Acknowledgments

This work was funded by the National Key Research and Development Project of China under Grant No. 2020AAA0105600, and by the National Natural Science Foundation of China under Grant No. U21B2048 and No. 62006183. Thanks to Huawei's support.

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.

[2] Arjun Ashok, KJ Joseph, and Vineeth Balasubramanian. Class-incremental learning with cross-space clustering and controlled transfer. *arXiv preprint arXiv:2208.03767*, 2022.

[3] Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.

[4] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, pages 233–248, 2018.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Songlin Dong, Xiaopeng Hong, Xiaoyu Tao, Xinyuan Chang, Xing Wei, and Yihong Gong. Few-shot class-incremental learning via relation knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1255–1263, 2021.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[9] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pages 86–102. Springer, 2020.

[10] 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.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Yuhang He, Xing Wei, Xiaopeng Hong, Wei Ke, and Yihong Gong. Identity-quantity harmonic multi-object tracking. *IEEE Transactions on Image Processing*, 31:2201–2215, 2022.

[13] Yuhang He, Xing Wei, Xiaopeng Hong, Weiwei Shi, and Yihong Gong. Multi-target multi-camera tracking by tracklet-to-target assignment. *IEEE Transactions on Image Processing*, 29:5191–5205, 2020.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *Computer Science*, 14(7):38–39, 2015.

[15] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019.

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

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] Zhizhong Li and Derek Hoiem. Learning without forgetting. *T-PAMI*, 40(12):2935–2947, 2018.

[19] Zhuoyun Li, Changhong Zhong, Sijia Liu, Ruixuan Wang, and Wei-Shi Zheng. Preserving earlier knowledge in continual learning with the help of all previous feature extractors. *arXiv preprint arXiv:2104.13614*, 2021.

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[21] 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.

[22] Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Ranjie Duan, Shaokai Ye, Yuan He, and Hui Xue. Towards robust vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12042–12051, 2022.

[23] 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.

[24] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[25] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[27] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017.

[28] Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021.

[29] Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 2018.

[30] Wenqi Shao, Yixiao Ge, Zhaoyang Zhang, Xuyuan Xu, Xiaogang Wang, Ying Shan, and Ping Luo. Dynamic token normalization improves vision transformer. *arXiv preprint arXiv:2112.02624*, 2021.

[31] Lucas Stoffl, Maxime Vidal, and Alexander Mathis. End-to-end trainable multi-instance pose estimation with transformers. *arXiv preprint arXiv:2103.12115*, 2021.

[32] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12183–12192, 2020.

[33] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers amp; distillation through attention. In *International Conference on Machine Learning*, volume 139, pages 10347–10357, July 2021.

[34] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021.

[35] 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.

[36] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. *arXiv preprint arXiv:2204.04662*, 2022.

[37] Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 171–181, 2022.

[38] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *arXiv preprint arXiv:2204.04799*, 2022.

[39] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022.

[40] Xing Wei, Yuanrui Kang, Jihao Yang, Yunfeng Qiu, Dahu Shi, Wenming Tan, and Yihong Gong. Scene-adaptive attention network for crowd counting. *arXiv preprint arXiv:2112.15509*, 2021.

[41] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. *arXiv preprint arXiv:1905.13260*, 2019.

[42] Jiangtao Xie, Ruiren Zeng, Qilong Wang, Ziqi Zhou, and Peihua Li. Sot: Delving deeper into classification head for transformer. *arXiv e-prints*, pages arXiv–2104, 2021.

[43] Haofei Xu and Juyong Zhang. Aanet: Adaptive aggregation network for efficient stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1959–1968, 2020.

[44] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021.

[45] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13208–13217, 2020.