Multi-Centroid Task Descriptor for Dynamic Class Incremental Inference

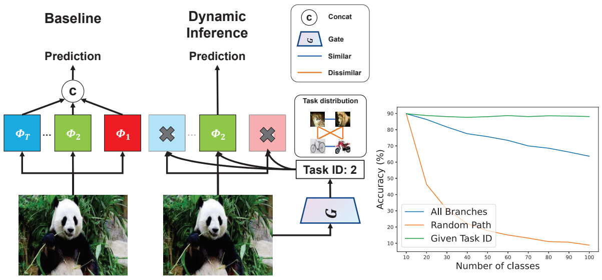
Tenghao Cai[1], Zhizhong Zhang[1,†], Xin Tan[1], Yanyun Qu[2], Guannan Jiang[3], Chengjie Wang[4], Yuan Xie[1]
[1]School of Computer Science and Technology
East China Normal University, Shanghai, China
[2]School of Informatics, Xiamen University, Fujian, China
[3]CATL, China, [4]Tencent Youtu Lab
51215901050@stu.ecnu.edu.cn, {zzzhang,xtan,yxie}@cs.ecnu.edu.cn
yyqu@xmu.edu.cn, jianggn@catl.com, jasoncjwang@tencent.com

Abstract

Incremental learning could be roughly divided into two categories, i.e., class- and task-incremental learning. The main difference is whether the task ID is given during evaluation. In this paper, we show this task information is indeed a strong prior knowledge, which will bring significant improvement over class-incremental learning baseline, e.g., DER [39]. Based on this observation, we propose a gate network to predict the task ID for class incremental inference. This is challenging as there is no explicit semantic relationship between categories in the concept of task. Therefore, we propose a multi-centroid task descriptor by assuming the data within a task can form multiple clusters. The cluster centers are optimized by pulling relevant sample-centroid pairs while pushing others away, which ensures that there is at least one centroid close to a given sample. To select relevant pairs, we use class prototypes as proxies and solve a bipartite matching problem, making the task descriptor representative yet not degenerate to uni-modal. As a result, our dynamic inference network is trained independently of baseline and provides a flexible, efficient solution to distinguish between tasks. Extensive experiments show our approach achieves state-of-the-art results, e.g., we achieve 72.41% average accuracy on CIFAR100-B0S50, outperforming DER by 3.40%.

1. Introduction

As a rapidly developing task in machine learning, incremental learning [7, 27] (IL) aims to continually learn new concepts (classes), where the training data comes as a sequence of tasks with each including a couple of new classes at a time. Such training strategy allows the network to incrementally learn novel knowledge [28] and has become more prevalent in real-world applications.



(a) (b)

Figure 1. (a): Illustration of DER (Left) and our dynamic inference method (Right). (b): Step accuracy on CIFAR100-B0S10 with DER under three different evaluation procedures: i) Given Task ID: use t-th branch to infer the sample in t-th task ii) All branches: concatenate all branches like original DER; iii) Random Path: randomly select one branch for inference.

Generally speaking, incremental learning could be roughly divided into two categories, *i.e.*, **Task-IL vs. Class-IL [35].** It depends on whether the task ID is given during inference. For example, Task-IL could use this task ID information to search prediction in a narrowed label space and is considered to be easier than Class-IL. Consequently, the accuracy of Task-IL is always outperforming Class-IL, and we can consider task ID as strong prior knowledge for incremental learning.

That motivates us to think about a critical problem, *can we utilize task ID to improve the performance of Class-IL approaches?* To answer this question, we select Class-IL SOTA method DER [39] as our baseline shown in the left of Fig. 1(a). Once a new task arrived, DER expands the current network with a new task-specific branch Φ. This multi-branch architecture allows us to conveniently adapt DER to a Task-IL approach. That is, when a couple of categories are trained in a specific task, a test sample belonging to this task would be sent to the corresponding branch for inference. So we compare the results of DER by giving task ID or directly using the original inference strategy over DER

[†] Corresponding authors.

and show the step accuracy in Fig. 1(b). It is amazing that even though the training procedure is the same, the accuracy gap between Task-IL and Class-IL is significantly large.

Based on this observation, we propose to design a gate network by automatically predicting task ID for class incremental inference, which is named dynamic inference in this paper. A straightforward solution to this aim is to treat the task ID prediction problem as a classification task. But it's rather difficult since there is no explicit semantic relationship between categories in the concept of task. So the data in one task has high variance and also lacks criteria to discriminate them.

To address the aforementioned issues, we propose a new multi-centroid task descriptor, by assuming data within a task can form multiple clusters. Their centers, *i.e.,* centroids, and then optimized to representatively describe a task. By pulling every relevant sample-centroid pair while pushing others away, it ensures that there is at least one centroid close to the given sample, which enables our network to distinguish between tasks. To select relevant pairs, we use class prototypes as proxy and solve a linear sum assignment problem [21], *i.e., bipartite matching problem* such that for a given sample, we are able to find its matched centroid based on the prototype-centroid matching results.

During inference, we compare each instance feature and the task descriptor and then find the most relevant branch for inference. The whole framework (dynamic inference network) is trained independently of the baseline, which allows our approach to be flexibly integrated into trained DER or other multi-branch models. We validate our approach on three commonly used benchmarks, including CIFAR-100, ImageNet-100, and ImageNet-1000. The results demonstrate the effectiveness of our approach, which obtains state-of-the-art results.

The main contributions of our work are: 1) Based on the idea of Task-IL, we propose a standalone gate network to automatically predict task ID for class incremental inference. The prediction is efficient, has no restrictions on test data, and is able to distinguish between tasks for large-scale Class-IL at the first time. 2) We propose a trainable multi-centroid task descriptor to describe the complicated task distribution. To make this descriptor representative, we solve a prototype-centroid bipartite matching problem to select relevant sample-centroid pairs for optimization. 3) Extensive experiments on large-scale benchmark CIFAR-100, ImageNet-100 and ImageNet-1000 demonstrate the superiority of our approach compared with the state-of-the-art.

2. Related Work

Class-IL is a long-standing problem and many works have investigated the essence of continual learning. In this section, we will give a brief introduction to these works.

Rehearsal methods [3, 4, 24, 32, 38, 43] explicitly save and retrain on a small part of data from old tasks. Because of its high performance, it has been applied in most Class-IL methods. Regularization methods add an extra regularization term to the loss function to mitigate forgetting, it can be divided into to two categories: 1) Weight regularization approaches [2, 5, 19, 22, 40]. 2) Data regularization approaches [10, 11, 17, 23, 34] with distillation [16].

Dynamic methods [12, 25, 26, 30, 33, 37, 39] dynamically add new task-specific parameters for new tasks and keep the parameters related to old tasks fixed to eliminate forgetting. Recently, DER [39] achieves SOTA performance on Class-IL. It proposes a dynamic expansion of the representation by adding a new task-specific branch to the super feature extractor. Its inference strategy is to concatenate all branches' output together, but, intuitively, as each branch is specialized for a task, it's better to activate only the corresponding branch for inference. However, task ID is unavailable for Class-IL.

Also, some works have proposed to infer task ID during evaluation but most of them depend on specific network architecture, or can not deal with larger scale datasets *e.g., ImageNet.* For instance, iTAML [31] uses the averaged task-wise maximal activation to predict task ID. Yet it assumes all samples inside a test batch share the same task ID, which is a strong and unpractical constraint. CCGN [1] uses a task predictor which concatenates each subnetworks' feature to infer the task ID, which is treated to be a classification task. However, it only works on simple datasets like MNIST. Instead, our dynamic inference network has no restrictions on test data and is able to distinguish between tasks for large-scale Class-IL problems.

3. Methods

In this section, we present our dynamic inference strategy for Class-IL, aiming to achieve a good balance between stability and plasticity.

3.1. Problem Setup and Method Overview

Under the context of Class-IL, it consists of a sequence of T tasks $[D^1, D^2, ..., D^T]$. Each task t includes a training and test data $D^t = (D^t_{train}, D^t_{test})$ with label space $Y^t = [y^t_1, y^t_2, ..., y^t_n]$. Different session has no overlapped categories so when $i \neq j$ we have $Y^i \neq Y^j$. During testing, the model is evaluated on the test data from all previous tasks $D_{test} = D^1_{test} \cup D^2_{test} \cup ... \cup D^t_{test}$ and the learned model is supposed to predict well in the whole label space. Notice that the task ID is not given during inference. Our method adopts the rehearsal strategy, which saves a part of old data in memory M_t at the end of each session and the memory is merged with the data from the new task to form a new dataset $D_t = M_t \cup D^t_{train}$.

We use a similar learning procedure from DER [39] with some modifications. In DER, it trains a supernet with multiple branches for dynamical incremental learning.

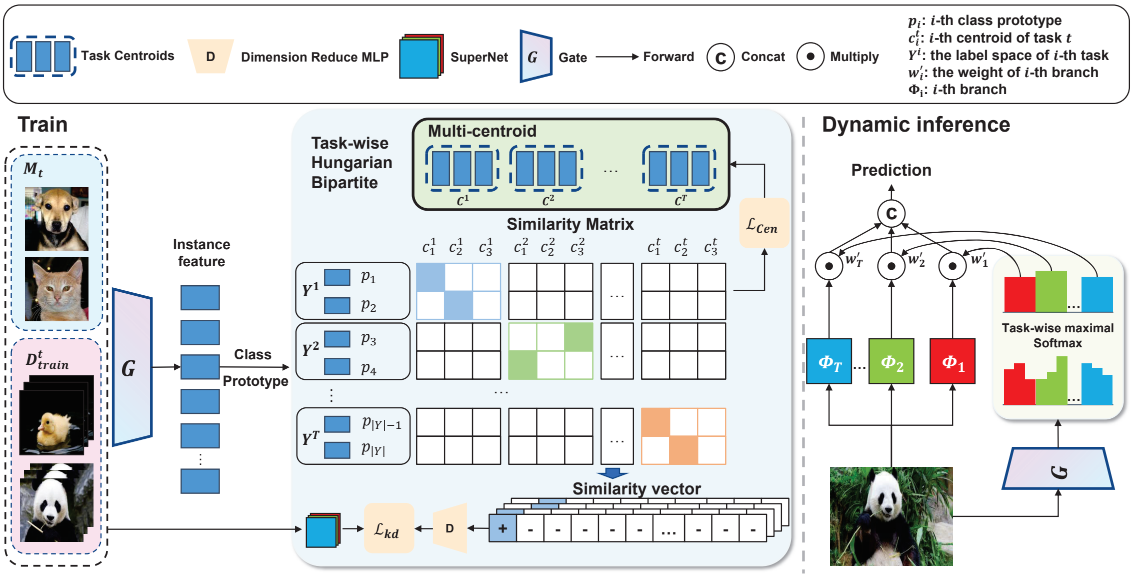
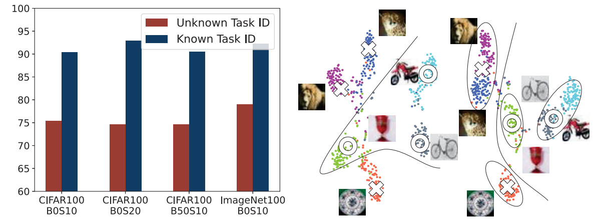


Figure 2. Pipeline of dynamic inference network. (left): During training, we solve a prototype-centroid bipartite problem and get the matched pair (colored square). This matched pair is then used for $\mathcal{L}_{Cen}$ to optimize the centroid and generates similarity vectors for knowledge distillation loss $\mathcal{L}_{\text{kd}}$. (right): During evaluation, we use the gate network G to compute the weight for each branch by selecting the task-wise maximal feature-centroid similarity value.



(a) (b)

Figure 3. (a): Comparison of average accuracy on whether task ID is given during inference, which is consistent with Fig. 1(b). (b): Two tasks [tiger, lion, plate] and [cycle, motorcycle, cup] trained with different descriptors: class prototype (Left), multi-centroid (Right). The bold line indicates the boundary between tasks and the ellipse illustrated the corresponding relationship between class and centroids. It is observed that the tasks can't be discriminated by the class prototype, while our centroid descriptors are representative.

Once a task t arrived, DER expands the current supernet $[\Phi_1, \ldots, \Phi_{t-1}]$ with a new branch Φ_t. Each added branch Φ_t includes a feature extractor F_t and a classifier H_t, where the output channel of H_t is consistent with the class number in task t. The final prediction of DER is obtained by concatenating all features from F_t and sending this feature into a uniform classifier. However, we find this concatenation is heavy and has less improvement. So we directly concatenate the output of H_t instead of the features, with a softmax operator. This allows us to reduce the number of network parameters and yield better performance.

Although DER achieves SOTA results, we notice that DER is not explainable in the test stage as they just simply concatenate all the outputs of Φ_t. In fact, each branch Φ_t is added when t-th task arrived. Hence, intuitively, this branch would be beneficial for this task because Φ_t is specialized in the current classification task by feeding the corresponding data.

As discussed in the introduction, by giving task ID, we can only activate t-th branch Φ_t for inference (the output of H_t). As shown in Fig. 3(a), a large accuracy improvement compared with DER baseline is observed. That motivates us to design a gate network G to infer task ID. To achieve this aim, we meet a critical challenge: the task ID of each test sample is hard to obtain since we randomly select samples to constitute the concept "task", where there is no semantic relationship among tasks.

So we propose a dynamic inference network to overcome this issue and the overview is shown in Fig. 2. During training, we introduce the trainable multi-centroid task descriptor C^t to abstract the t-th task in the upper of the figure. Then, we use a novel task-wise hungarian bipartite matching strategy to select the matched centroid for each sample. This matched pair is then used for a metric loss $\mathcal{L}_{Cen}$ optimized by pulling relevant sample-centroid pairs while pushing others away, which makes the centroid representative. Besides, by comparing the instance feature and our descriptor, we can get a similarity vector for knowledge distillation loss $\mathcal{L}_{\text{kd}}$. During evaluation, the similarity between instance feature and descriptor can be used to find the most relevant branch for inference.

3.2. Multiple Centroid Task Descriptor

Multi-centroid Task Descriptor. We assume that data within a task can form multiple clusters, and the cluster centers, *i.e.,* centroids, are supposed to be discriminative for tasks. A straightforward solution is to leverage the class prototypes as the task descriptor. But as illustrated in the left panel of Fig. 3(b), some visual similar categories *e.g.,* cup and plate, are grouped into different tasks, and thus it is hard to recognize them since the prototype is not trained for describing a task and thus the task gap is bounded by the semantic prototypes.

In this sense, we propose a multi-centroid task descriptor, which is a list of trainable tensor $[C^1, C^2, ..., C^T]$. C^t are the task descriptor of t-th task in the shape of $M \times K^t$, where M denotes the dimension of features and K^t is the number of centroids in the t-th task. Every column c_i^t in C^t represents one centroid of the t-th task. In our settings, K^t is proportional to $|Y^t|$ and usually larger or smaller than $|Y^t|$ (the number of classes in t-th task).

Descriptor Training. To make this descriptor more representative, we assume for any sample, there exists a positive centroid with high relevance while others should be low relevance, such that this descriptor is able to distinguish different tasks. To this end, we establish a metric loss on the centroid descriptor:

$$\mathcal{L}_{\text{Cen}} = -(\frac{exp(f \cdot c^+)}{\sum_j exp(f \cdot c_j)}), \tag{1}$$

where f donotes the feature extracted by gate G, c^+ donotes the matched positive centroid defined later and c_j is the j-th centroid from the whole centroids list $[C^1, C^2, ..., C^T]$.

In our implementation, we use the prototypes as the proxy to do batch-wise prototype-centroid matching, so in each batch, we can find the sample's matched centroid according to its corresponding class. Formally, given a batch of training data, we first compute the mean feature of each class as the prototype of the class:

$$P = [p_1, p_2, ..., p_{|Y|}], \tag{2}$$

where Y is the label space in this batch.

To prevent the selected positive centroid collapsed to the same centroid and thus the multi-centroid behaves like a uni-centroid. We add a constraint that no two classes share the same centroid(To ensure the number of classes is less than the number of centroids, we use a sampling strategy to only select a part of categories of samples.). It transforms the problem of finding a positive sample into solving a linear sum assignment problem, *i.e., bipartite matching problem*. Formally, we search for a permutation of an index for centroids C^t to establish the best bipartite matching for prototype-centroid pairs whose sum of similarity is maximized.

$$\sigma = \arg\max_{\sigma} \sum_{i}^{|Y^t|} p_i^t c_{\sigma(i)}^t, \tag{3}$$

where p_i^t is the prototype of i-th class in t-th task. We use Hungarian algorithm [21] to efficiently compute the optimal permutation. Every query image's positive centroid then is assigned based on which prototype it belongs to. Besides, we treat all the remaining centroids as negative pairs. In this way, the prototype is calculated and sampled batch by batch, and the matching result is dynamically updated, allowing the centroids to be well-trained.

3.3. Adaptive Mixture of Subnetworks

During evaluation, we use the dynamic inference network to select the branch. The weight of a task-specific branch is obtained by first computing the similarity value between the instance feature and all centroids belonging to one task and then choosing the maximal value.

$$w_t = max(d_1^t, d_2^t, ..., d_{|K^t|}^t), \tag{4}$$

where d_i^t denotes the similarity between instance feature and the i-th centroid in t-th task. Then we use softmax with temperature $\zeta = 12$ to normalize the weights and get a smoothed weight vector w'.

$$w'(t|x) = Softmax(w/\zeta). \tag{5}$$

The final prediction is the weighted concatenation of the output of each branch.

$$p = Softmax([w_1'\Phi_1(x)), w_2'\Phi_2((x)), ..., w_t'\Phi_t(x)]). \tag{6}$$

3.4. Optimizing

Instead of training the supernet and dynamic inference network jointly, our training pipeline consists of two stages: 1) firstly, we train the supernet by expanding the new branches for the coming tasks; 2) secondly, we train the multi-centroid dynamic inference network for evaluation.

Supernet Training. Like DER, we train the supernet with cross-entropy loss by using the merged dataset (memory data and new data):

$$\mathcal{L}_{\text{ce}} = -\log(p(y = y_i|x_i))). \tag{7}$$

We also follow DER to use an auxiliary classification loss, which for the new data, encourages the network to predict their labels, and for the memory data, enforces the network to classify them into one extra class by treating all old concepts as one category. Finally, we re-train the classifier to mitigate the bias towards the new task by using a class-balanced subset $\tilde{D}_t$ [18, 42]. Notice that after the first training stage, we fix the parameters of the supernet. This allows our approach to be flexibly integrated into existing trained DER or other multi-branch models.

Dynamic Inference Network Training. To train the dynamic inference network, we find the number of training samples from the old task is not sufficient. Therefore, we use the data augmentation technique in CEC [41], to synthesize a new training set. It augments the samples already used for training the supernet, so we don't introduce any extra training samples. Here RandAugment [6] is selected as it includes many data augmentation techniques. Intuitively, the prediction of supernet on this new training set will be slightly drifted and this additional diversity would also facilitate our centroid descriptor learning.

We also find distillation from the supernet is beneficial in enhancing the accuracy of the dynamic inference network. Hence, we add another supervision by distilling the output of the supernet, where the semantic information contains rich task information. Specifically, in Eq. (4), we compute the similarity between instance features and all task descriptors and collect the result into a similarity vector d. The learned similarities d is then encouraged to recover the semantic labels by using this distillation loss. However, the dimension of d is not aligned with the output of supernet, *i.e.,* the number of classes. Thus, we use an MLP subnetwork to align them. The distillation loss is:

$$\mathcal{L}_{\text{kd}} = -\sum_{i}^{|Y|} \frac{\exp(l_i/\lambda)}{\sum_j \exp(l_j/\lambda)} \log \frac{\exp(d'_i/\lambda)}{\sum_j \exp(d'_j/\lambda)}, \tag{8}$$

where l is the output of the supernet and d' is the aligned similarity vector d by MLP.

Besides, the sample number of old tasks and new tasks is unbalanced. It exhibits a long-tailed label distribution. So we use a variant softmax loss proposed in logit adjustment [29] by assigning a large margin to rare class to improve the rare class accuracy. The distillation loss in Eq.(8) and the centroid metric loss in Eq.(1) becomes:

$$\mathcal{L}_{\text{kd}} = -\sum_{i}^{|Y|} \frac{\exp(l_i/\lambda)}{\sum_j \exp(l_j/\lambda)} \log \frac{\exp((d'_i + \tau \log \pi'_i)/\lambda)}{\sum_j \exp((d'_j + \tau \log \pi'_j)/\lambda)},$$

$$\mathcal{L}_{\text{Cen}} = -log(\frac{\exp(f \cdot c^{+} + \tau \log \pi^{+})}{\sum_j \exp(f \cdot c_j + \tau \log \pi_j)}), \tag{9}$$

where τ is a hyperparameter and the default value is 1. π'_i is the frequency of i-th class and π_i is the frequency of i-th centroid defined as:

$$\pi_i = \frac{1}{K^t} \frac{n_t}{\sum_j n_j}, \tag{10}$$

where n_t is the sample number of task t.

Finally, we optimizie $\mathcal{L}_{\text{Cen}}$ and $\mathcal{L}_{\text{kd}}$ in Eq. (9) to obtain our dynamic inference network.

4. Experiment

4.1. Experimental Setup

In this section, we conduct extensive experiments to verify the effectiveness of the proposed method. In the next, we will introduce our experimental setups, main results, ablation study and show some visualization results.

Datasets. We evaluate our method on widely used Class-IL datasets: CIFAR-100, ImageNet-100, and ImageNet-1000. **CIFAR-100** [20] consists of 60,000 32x32 color images of 100 classes with 500 images for training and 100 images for evaluation per class. **ImageNet-1000** [8] is a large-scale dataset with 1000 classes which consists of 1,281,167 colour images for training and 50,000 images for validation. **ImageNet-100** is a subset of ImageNet-1000 by selecting 100 classes from the whole dataset. Here we use the same class subset in previous work like DER [39] and PODNet [11].

Evaluation Protocols. The general evaluation protocol is followed by DER. Our data are split as 1) **CIFAR100-B0**: we split the 100 classes in CIFAR-100 into 5, 10, 20, and 50 tasks with equal class numbers. We store 2000 images for rehearsal. 2) **CIFAR100-B50**: the model is pretrained using 50 randomly selected classes and the remaining 50 classes are divided into 2, 5, and 10 tasks with equal class numbers. We store 20 images per class in memory. 3) **ImageNet100-B0**: we split 100 classes in ImageNet100 into 10 tasks each consisting of 10 classes and the rehearsal memory size is 2000 images. 4) **ImageNet100-B50**, the model is pretrained using 50 classes and the remaining classes come in 10 steps with equal size. We store 20 images per class for rehearsal. 5) **ImageNet1000-B0**: the 1000 classes are split into 10 tasks and each task contains 100 classes. We store 20000 images in total for rehearsal. For CIFAR-100, we run experiments on three class orders and report the average and standard deviations of top-1 accuracy. For ImageNet, we report the top-1 and top-5 average accuracy and last step accuracy.

Implementation Details. We follow DER [39] to use modified ResNet-18 [15] as feature extractor F_t in supernet. For ImageNet-100 and ImageNet-1000, we use standard ResNet-18 as basic networks. We use the same architecture as F_t to implement the gate network G. The number of centroids is simply set the same as the class number of tasks, where careful tuning would bring performance improvement. For bipartite matching, we sample a few classes randomly such that the class number is less than the centroid number, which ensures that every class prototype has one centroid to match with. The optimizer is SGD with learning rate of 0.1, momentum of 0.9, and weight decay of 5×10^{-4}. The batch size for CIFAR-100 and ImageNet-100 is 256 and 1024 for ImageNet-1000. The epoch number is

The rehearsal setting used in our paper is the same as most rehearsal approaches, such as DER [39], iCaRL [32].

Methods	Venue	5 Steps		10 Steps		20 Steps		50 Steps	
		#P	Avg	#P	Avg	#P	Avg	#P	Avg
Bound		11.2	80.40	11.2	80.41	11.2	81.49	11.2	81.74
iCaRL	CVPR'17	11.2	$71.14_{\pm 0.34}$	11.2	$65.27_{\pm 1.02}$	11.2	$61.20_{\pm 0.83}$	11.2	$56.08_{\pm 0.83}$
UCIR	CVPR'19	11.2	$62.77_{\pm 0.82}$	11.2	$58.66_{\pm 0.71}$	11.2	$58.17_{\pm 0.30}$	11.2	$56.86_{\pm 3.74}$
BiC	CVPR'19	11.2	$73.10_{\pm 0.55}$	11.2	$68.80_{\pm 1.20}$	11.2	$66.48_{\pm 0.32}$	11.2	$62.09_{\pm 0.85}$
WA	CVPR'20	11.2	$72.81_{\pm 0.28}$	11.2	$69.46_{\pm 0.29}$	11.2	$67.33_{\pm 0.15}$	11.2	$64.32_{\pm 0.28}$
PODNet	ECCV'20	11.2	$66.70_{\pm 0.64}$	11.2	$58.03_{\pm 1.27}$	11.2	$53.97_{\pm 0.85}$	11.2	$51.19_{\pm 1.02}$
RPSNet	NeurIPS'19	60.6	70.5	56.5	68.6	-	-	-	-
DyTox	CVPR'22	10.7	–	10.7	67.33	10.7	67.30	10.7	64.39
FOSTER	ECCV'22	11.2	72.56	11.2	72.90	11.2	70.65	–	–
DER	CVPR'21	33.6	$76.80_{\pm 0.79}$	61.6	$75.36_{\pm 0.36}$	117.6	$74.09_{\pm 0.33}$	285.6	$72.41_{\pm 0.36}$
Our	–	+11.2	$\mathbf{78.15}_{\pm 0.58}$	+11.2	$\mathbf{77.40}_{\pm 0.94}$	+11.2	$\mathbf{76.20}_{\pm 1.18}$	+11.2	$\mathbf{75.81}_{\pm 1.38}$

Table 1. Quantitative results on CIFAR100-B0 (average over 3 runs). The subscript denotes the variance. For DyTox, we use the corrected results after the author fixed the rehearsal memory issue in implementation. $\#P$ means the average number of parameters in millions. *Avg* means the average accuracy (%) over steps. $+11.2$ indicates the extra number of parameters brought by the gate network.

Methods	Venue	2 Steps		5 Steps		10 Steps	
		#P	Avg	#P	Avg	#P	Avg
Bound		11.2	77.22	11.2	79.89	11.2	79.91
iCaRL	CVPR'17	11.2	$71.33_{\pm 0.35}$	11.2	$65.06_{\pm 0.53}$	11.2	$58.59_{\pm 0.95}$
UCIR	CVPR'19	11.2	$67.21_{\pm 0.35}$	11.2	$64.28_{\pm 0.19}$	11.2	$59.92_{\pm 2.4}$
BiC	CVPR'19	11.2	$72.47_{\pm 0.99}$	11.2	$66.62_{\pm 0.45}$	11.2	$60.25_{\pm 0.34}$
WA	CVPR'20	11.2	$71.43_{\pm 0.65}$	11.2	$64.01_{\pm 1.62}$	11.2	$57.86_{\pm 0.81}$
PODNet	ECCV'20	11.2	$71.30_{\pm 0.46}$	11.2	$67.25_{\pm 0.27}$	11.2	$64.04_{\pm 0.43}$
FOSTER	ECCV'22	–	–	–	–	11.2	67.95
DER	CVPR'21	22.4	$74.61_{\pm 0.52}$	39.2	$73.21_{\pm 0.78}$	67.2	$72.81_{\pm 0.88}$
Ours	–	+11.2	$\mathbf{76.72}_{\pm 0.62}$	+11.2	$\mathbf{76.19}_{\pm 0.29}$	+11.2	$\mathbf{75.43}_{\pm 0.38}$

Table 2. Quantitative results on CIFAR100-B50 (average over 3 runs). The subscript denotes the variance. $\#P$ means the average number of parameters in millions. *Avg* means the average accuracy (%) over steps. $+11.2$ indicates the extra number of parameters brought by the gate network.

varied due to the scale of datasets and the full hyperparameter details are in the supplementary material.

4.2. Comparison with State-of-the-art

To demonstrate the effectiveness of our method, we compare our model with other SOTA Class-IL methods. The competitors include iCaRL [32], UCIR [17], BiC [38], WA [43], PODNet [11], RPSNet [30], DER [39], DyTox [12], FOSTER [37]. We summarize the results in Tabs. 1 to 3. All results are borrowed from the corresponding papers. The result shows that our approach consistently outperforms other SOTA methods under various datasets and protocols. For example, Table.1 and 2 show that, our model surpasses current SOTA method (DER [39]) by **1.35%**, **2.04%**, **2.11%**, **3.40%** for 5, 10, 20 and 50 steps on CIFAR100-B0 benchmark, respectively, while on CIFAR100-B50, the improvements are **2.11%**, **2.98%**, **2.62%**, respectively. A larger incremental step is even harder for avoiding forgetting, but the improvement by using our dynamic inference is even expanded. Table 3 shows the results on ImageNet100-B0/B50. It appears that our approach also achieves the SOTA performance, *e.g.,* **80.46%** in term of average accuracy with **2.06%** improvement on ImageNet100-B0 10 Steps.

Fig. 4 shows the results where the performance changes as learning steps increase. It is observed that the accuracy of all the approaches drops due to the forgetting phenomenon. However, on all datasets, our approach surpasses other SOTA methods at every step and the relative performance gain even grows as the number of steps increases.

4.3. Ablation Study and Analysis

In this part, we will analyze the characteristic of our dynamic inference strategy, such as the influences of the gate network, and then conduct ablation studies to demonstrate the effectiveness of each component in our approach.

Effectiveness of the Multi-centroid. To verify the effectiveness of the multi-centroid task descriptor, we compare our approach by varying the number of centroids. When the number of centroids degrades to 1, it implies that we directly assign a task ID to each sample and then optimize the gate network as a classification task. It appears that the multi-centroid task descriptor gets an almost 1.5% improvement over this classification method and the accuracy increases as the number of centroids increases. Moreover, compared with the class prototype descriptor, the multi-centroid descriptor achieves superior accuracy even when the centroid number is less than the class number. Here we simply set the centroid number as the class number.

Tab. 4 shows the comparison between our approach and CCCG [1]. CCCG is also designed to infer task ID by concatenating all features from branches and then send it into a task classifier. We can see that CCGN doesn't work well

Methods	Venue	ImageNet100-B0					ImageNet1000-B0				
		#P	top-1		top-5		#P	top-1		top-5	
			Avg	Last	Avg	Last		Avg	Last	Avg	Last
Bound		11.2	-	-	-	95.1	11.2	89.27	-	-	-
iCaRL	CVPR'17	11.2	-	-	83.6	63.8	11.2	38.4	22.7	63.7	44.0
BiC	CVPR'19	11.2	-	-	90.6	84.4	11.2	-	-	84.0	73.2
WA	CVPR'20	11.2	-	-	91.0	84.1	11.2	65.67	55.6	86.6	81.1
RPSNet	NeurIPS'19	-	-	-	87.9	74.0	-	-	-	-	-
DyTox	CVPR'22	11.0	71.85	57.94	90.72	83.52	11.4	68.14	59.75	87.03	82.93
FOSTER	ECCV'22	11.2	78.40	69.91	–	–	11.2	68.34	–	–	–
DER	CVPR'21	61.6	77.18	66.70	93.23	87.52	61.6	68.84	60.16	88.17	82.86
Ours	–	+11.2	**80.46**	**71.52**	**94.76**	**90.90**	+11.2	**70.80**	**62.30**	**88.65**	**84.14**

Methods	Venue	ImageNet100-B50				
		#P	top-1		top-5	
			Avg	Last	Avg	Last
Bound		11.2	81.20	81.5	-	-
UCIR	CVPR'19	11.2	68.09	57.3	-	-
PODNet	ECCV'20	11.2	74.33	-	-	-
TPCIL	ECCV'20	11.2	74.81	66.91	-	-
FOSTER	ECCV'22	11.2	77.54	–	–	–
DER	CVPR'21	67.2	78.20	74.92	94.20	91.30
Ours	–	+11.2	**79.83**	**75.24**	**94.98**	**92.72**

Table 3. Quantitative results on ImageNet100 and ImageNet1000 datasets. Left: ImageNet100-B0 and Imagenet1000-B0. Right: ImageNet100-B50. For DyTox, we use the corrected results after the author fixed the rehearsal memory issue in implementation. *#P* means the average number of parameters in millions. *Avg* means the average accuracy (%) over steps. *Last* denotes the last step accuracy (%). +11.2 indicates the extra number of parameters brought by the gate network.

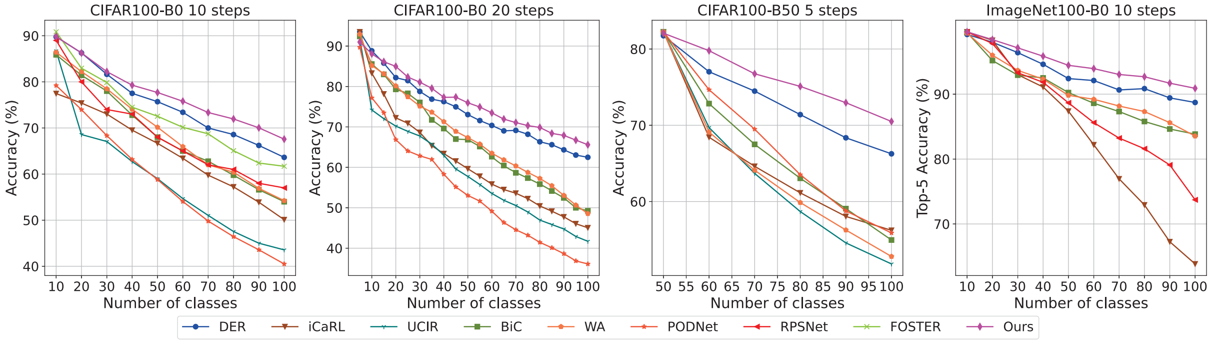


Figure 4. The step performance in term of accuracy on CIFAR100-B0S10/B0S20/B50S5 and ImageNet100-B0S10.

Methods	CIFAR100-B0S10		CIFAR100-B50S10	
	Avg	Last	Avg	Last
CCCG [1]	68.52	52.21	55.62	43.96
Ours	77.40	67.62	75.17	69.72

Table 4. Ablation study on dynamic inference by comparing multi-centroid task descriptor and CCCG. *Avg* means the average accuracy (%) over steps. *Last* denotes the last step accuracy (%).

Gate	DA	KD	LA	Avg	Last
				75.38	65.23
✓				75.64	64.21
✓			✓	76.19	65.50
✓	✓	✓		76.86	65.94
✓	✓		✓	77.23	66.76
✓		✓	✓	76.45	66.09
✓	✓	✓	✓	77.40	67.62

	Avg	Last
DER	75.38	65.24
DER Logit Adjustment	74.84	64.61
DER with RA	76.88	67.06
DER with RA and Distill	76.75	66.53
Ours	**77.40**	**67.62**
Ours on DER Logit Adjustment	**76.85**	**66.92**
Ours on DER with RA	**78.04**	**68.42**

Table 5. Ablation results on different training strategies. Avg means the average accuracy (%) over steps. Last denotes the last step accuracy (%).

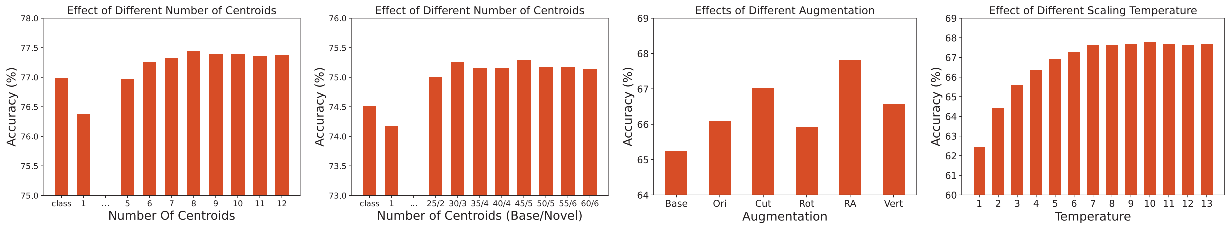
on CIFAR-100 and our method outperforms it by a large margin.

Study on Training Strategy. In section 3.4, we introduce three training strategies to improve the performance of dynamic inference: 1) data augmentation (DA); 2) Knowledge distillation (KD) from supernet; 3) Logit Adjustment (LA). The results on CIFAR100-B0S10 in terms of average accuracy and last step accuracy are shown in Tab. 5. We can see that the dynamic inference would even degrade the last step's accuracy without any additional training strategies. But when combining all of these approaches, the average accuracy is enhanced from 75.38% to 77.40%, while the last step accuracy is from 64.21% to 67.57%. Especially without LA, the dynamic inference would predict a large weight on the branch of new tasks, as the training samples between old tasks and new tasks are highly unbalanced at the end of training.

We also conduct experiments on a stronger DER trained with RandAugment and distillation. DER is a simple baseline that can't be directly implemented with Distillation. So we follow the Born Again Network [13] and conduct a two-stage training, where in the second stage we copy and freeze the model of the first stage as a teacher to distillation. It appears that DER with RandAugment has a small improvement while distillation has a negative impact. With this strong DER, our model can also gain a relatively large improvement.

Study on Different Augmentation. As shown in Fig. 5(c), we study the effects of data augmentation on CIFAR100-B0S10. We choose augmentation from a candidate list including origin (use the original data augmentation in DER), CutOut [9], rotation, RandAugment, and vertical flip. It turns out that all these data augmentations improve the performance compared with the baseline. Among them, RandAugment achieves the highest accuracy.

Effect of Temperature. As illustrated in Fig. 5(d), we investigate the effects of ζ used in Eq. (5) on CIFAR100-B0S10. It appears that standard softmax (temperature $\zeta = 1$) does harm to the performance. Also, as the temperature value increases, the performance first climbs to the peak and then slowly decreases. Intuitively, this phenomenon may



(a) (b) (c) (d)

Figure 5. Ablation of different components. (a), (b): the average accuracy using different numbers of centroid on CIFAR100-B0S10/B50S10 respectively. ”Class” in the x-axis indicates that we use class prototypes as our centroids. (c): the last accuracy under different data augmentation. (d): the last accuracy under different softmax temperatures.

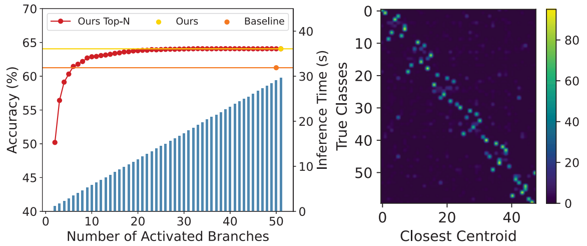
be attributed to the over-confident problem of neural networks [14], while a larger value of temperature results in a smoothing prediction.

Reducing Inference Time With Task Descriptors. As illustrated in Fig. 6(a), the inference time can be reduced by activating the top-N similar branches by comparing the instance feature and task descriptors. Specifically, at the last step of CIFAR100-B0S50, by selecting the top-5 branches, inference time can be reduced to 10% of the baseline method with only a slight drop in accuracy.

4.4. Visualization

Visualization of Confusion Matrix. As illustrated in Fig. 6(b), we visualize the confusion matrix between the first 60 classes and centroids on CIFAR100-B0S10. Elements in the confusion matrix indicate the number of samples from i-th class closest to the j-th centroid. We can see that multiple classes in the same task share the same centroid, which further demonstrates the ability to group similar features into one single cluster, which further highlights the difference between multi-centroid and class prototype.

Visualization of Task Data. We make visualization experiments by using t-SNE [36] on CIFAR100-B0S20. We project the first two task features from the gate network learned by uni-centroid (centroid number is set to 1, left), multi-centroid (middle), and class prototype use class prototype instead of trainable centroid (right) into a 2D plane. As shown in Fig. 7, the features learned by uni-centroid are indiscriminative and cannot represent the complicated task distribution, while multi-centroid descriptor captures this variance by grouping similar features into clusters and the centroids are very representative for tasks. The features learned by class prototypes are entangled together and are not able to distinguish between tasks, which leads to a complicated boundary while features learned by multi-centroid descriptors retain a simpler boundary.



(a) (b)

Figure 6. (a): Inference time and accuracy on CIFAR-100 test set using different numbers of activated branches. (b) Class-centroid confusion matrix of first 60 classes on CIFAR100-B0S10.

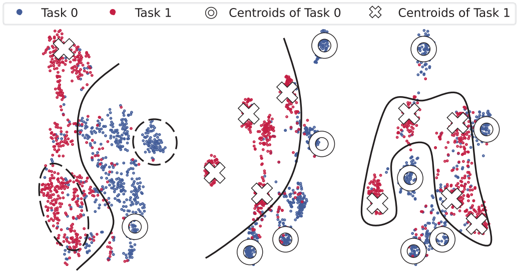


Figure 7. The t-SNE visualization of the first 2 tasks on CIFAR100-B0S20 with task boundary. Left: uni-centroid. Mid: multi-centroid. Right: class prototypes.

5. Conclusion

In this work, we proposed a standalone gate network for class-incremental inference by predicting task ID during evaluation. Considering the high variance of data within a task, a novel multi-centroid task descriptor is introduced to capture the complicated data distribution. Experiments on three major benchmarks show the effectiveness of our dynamic inferencing network even on large-scale datasets *e.g.,* ImageNet-1000, and our method obtains SOTA results. Further study would include using task descriptors for reducing the parameters by merging similar branches.

Acknowledgments. This work is supported by grants from the National Key Research and Development Program of China (2021ZD0111000), National Natural Science Foundation of China (No.62222602, 62176092, 62106075), Natural Science Foundation of Shanghai (23ZR1420400), Shanghai Sailing Program (23YF1410500), Science and Technology Commission (No.21511100700), CAAI-Huawei MindSpore Open Fund, CCF-Lenovo Blue Ocean Research Fund.

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 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3931–3940, 2020. 2, 6, 7

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, pages 139–154, 2018. 2

[3] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 583–592, 2019. 2

[4] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018. 2

[5] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, pages 532–547, 2018. 2

[6] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 5

[7] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. 1

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

[9] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 7

[10] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5138–5146, 2019. 2

[11] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2020. 2, 5, 6

[12] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 6

[13] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR, 2018. 7

[14] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017. 8

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 5

[16] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems (NeurIPS) Workshop*, 2015. 2

[17] 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 (CVPR)*, June 2019. 2, 6

[18] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019. 4

[19] 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. 2

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009. 5

[21] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 2, 4

[22] 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, 2017. 2

[23] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. 2

[24] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12245–12254, 2020. 2

[25] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pages 67–82, 2018. 2

[26] 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 (CVPR)*, pages 7765–7773, 2018. 2

[27] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1

[28] 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

[29] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314*, 2020. 5

[30] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for incremental learning. *Advances in Neural Information Processing Systems*, 2019. 2, 6

[31] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13588–13597, 2020. 2

[32] 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 (CVPR)*, 2017. 2, 5, 6

[33] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018. 2

[34] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, pages 254–270. Springer, 2020. 2

[35] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 1

[36] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 8

[37] 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. 2, 6

[38] 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 (CVPR)*, pages 374–382, 2019. 2, 6

[39] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 5, 6

[40] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017. 2

[41] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 5

[42] Xu Zhang, Felix X. Yu, Svebor Karaman, Wei Zhang, and Shih-Fu Chang. Heated-up softmax embedding. In *arXiv preprint arXiv:1809.04157*, 2018. 4

[43] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13208–13217, 2020. 2, 6