Towards Better Plasticity-Stability Trade-off in Incremental Learning: A Simple Linear Connector

Guoliang Lin
Sun Yat-sen University
Guangdong, China
lingliang@mail2.sysu.edu.cn

Hanlu Chu
South China Normal University
Guangdong, China
hlchu@m.scnu.edu.cn

Hanjiang Lai *
Sun Yat-sen University
Guangdong, China
laihanj3@mail.sysu.edu.cn

Abstract

Plasticity-stability dilemma is a main problem for incremental learning, where plasticity is referring to the ability to learn new knowledge, and stability retains the knowledge of previous tasks. Many methods tackle this problem by storing previous samples, while in some applications, training data from previous tasks cannot be legally stored. In this work, we propose to employ mode connectivity in loss landscapes to achieve better plasticity-stability trade-off without any previous samples. We give an analysis of why and how to connect two independently optimized optima of networks, null-space projection for previous tasks and simple SGD for the current task, can attain a meaningful balance between preserving already learned knowledge and granting sufficient flexibility for learning a new task. This analysis of mode connectivity also provides us a new perspective and technology to control the trade-off between plasticity and stability. We evaluate the proposed method on several benchmark datasets. The results indicate our simple method can achieve notable improvement, and perform well on both the past and current tasks. On 10-split-CIFAR-100 task, our method achieves 79.79% accuracy, which is 6.02% higher. Our method also achieves 6.33% higher accuracy on TinyImageNet. Code is available at https://github.com/lingl1024/Connector.

1. Introduction

In recent years, deep neural networks have been reported promising performance on various tasks. In the dynamic world, the deep model also needs to be updated as new data becomes available. Hence, Incremental Learning (IL) [7, 31] has received much attention, which studies the problem of continually learning from sequential tasks.

In this paper, we consider the data-free incremental learning [36], where the training samples from previous tasks do not exist. Hence, the main criterion of data-free IL [9, 33] is that no data from the previous tasks is stored when continually refining the model as new data becomes available. It is a direct cause of catastrophic forgetting problem [24], and the plasticity-stability dilemma [3, 28] is a more general problem: (1) *plasticity*: the deep model should learn the new knowledge of the current task, and (2) *stability*: it should also preserve the knowledge of previous tasks.

Many algorithms have been proposed to strike a balance between plasticity and stability. An intuitive solution is to generate samples from previous tasks, e.g., ILCAN [40] generates samples to preserve the old knowledge. The regularization-based methods use the extra regularization term in the loss function to consolidate previous knowledge, such as EWC [21] using the Fisher information to calculate each parameter's importance. The architectural methods [24] learn the dynamic architecture of the deep network, e.g., DER [41] freezes the previously learned representation and dynamically expands the network for new task. The algorithm-based methods learn parameter updating rules to preserve the performance of previous tasks. For example, GEM [27] constrains new task updates which do not interfere with the previous knowledge. Adam-NSCL [37] updates network parameters in the null space of all previous tasks and achieves a promising performance on remembering previous knowledge. Although Adam-NSCL can preserve previous knowledge very well, the strong null-space projection also hurts the performance of the current task.

On the other side, many researches have focused on the connectivity in neural network loss landscapes [12, 13]. The previous works [14] found that two minima of independently trained deep networks can be connected in weight space, where the loss along the path remains low. Further, the recent works [13] and [38] showed that there exists a linear path of high accuracy to connect two minima when the networks share only a few epochs of the initialized SGD trajectory. Mode Connectivity SGD (MC-SGD) [29] is designed for incremental learning, which enforces the final weight linearly connected to all tasks' minima. Although

*Hanjiang Lai is the corresponding author.

MC-SGD achieves excellent performance, it needs to store previous samples, which contradicts with our problem setting. Hence, an interesting yet challenging question arises: how to build a high-accuracy pathway between previous and current models without previous samples?

In this paper, we provide a new insight to understand, analyze and build a high-accuracy connector without any previous training samples. To understand why we can linearly connect two minima of previous and current tasks, we first give a simple analysis that provides an upper bound of empirical loss for all tasks. Then, according to the upper bound, we view plasticity and stability as two independent optimization problems of deep neural networks. The two networks are trained to minimize the empirical loss and move towards each other. Finally, we propose a simple linear connector to attain a better balance between these two networks in the light of linear connectivity [13]. Central to our method is that we uncover a simple way to achieve a better plasticity-stability trade-off, i.e., a simple averaging of two carefully designed networks, which leads to higher accuracy neural network.

2. Preliminaries and Related Methods

2.1. Incremental Learning Methods

We review several categories of the existing deep incremental learning methods for plasticity-stability trade-off.

Regularization-based methods: This line of approaches introduce an extra regularization term to balance the trade-off. According to where the regularization term was explicitly applied to, these methods can be further divided into *structural* and *functional* regularization methods [31]. Structural regularization methods constrain changes on model's parameters. For example, EWC [21], SI [44], MAS [2] and UCL [1] explicitly added a regularization term to networks' parameters. The functional regularization methods, also known as the distillation-based methods, use the distillation loss between predictions from the previous model and the current model as the regularization term. The representative works include LwF [25], EBLL [32], GD-WILD [23], etc.

Rehearsal methods: This line of works preserve existing information by replaying data from previous tasks. Some algorithms store a subset of previous data, e.g., iCaRL [33] and GeppNet [15]. When the storage space is limited, it is important to find a suitable subset of data that can approximate the entire data distribution, e.g., SER [17] focuses on exemplar selection techniques. Another way to solve this limitation is using the generative modelling approaches [40] to generate a lot of samples of previous tasks. For example, DGR [35] is a framework with a deep generative model and a task solving model.

Architectural methods: These methods modify the underlying architecture to alleviate catastrophic forgetting, e.g., HAT [34] proposes a task-based binary masks that preserve previous tasks' information. UCB [11] uses uncertainty to identify what to remember and what to change. The dynamic growth approaches [41] are also proposed, e.g., DEN [42] dynamically expands network capacity when arrival of new task. Learn-to-Grow [24] proposes modifying the architecture via explicit neural structure learning.

Algorithm-based methods: These methods carefully design network parameter updating rule, which constrain new task updates that do not interfere the previous tasks. GEM [27] and A-GEM [4] are two representative works. OWM [43] is orthogonal weight modification method to overcome catastrophic forgetting. Adam-NSCL [37], which uses the null space of all previous data to remember existing knowledge, achieves an impressive performance on IL task.

Here, we give a brief review of Adam-NSCL. We have a W_{old} model trained on the previous data X_{old}, and X_{old} is not available when training the new task. To overcome this problem, Adam-NSCL stores the uncentered feature covariance $\mathcal{X}_{old} = \frac{1}{n_{old}} X_{old}^{\top} X_{old}$ for guaranteeing stability, where n_{old} is the number of data points in X_{old}. Then, it uses the SVD result of feature covariance $\mathcal{X}_{old}$ to find the null space of X_{old}, denoted as U_{old}. We have $\mathcal{X}_{old} U_{old} = 0$ in this way. The projection matrix is obtained as $P_{old} = U_{old} U_{old}^{\top}$.

Now when new data X_{new} is available, the W_{old} can be updated to learn the new task as:

$$W_{t+1} = W_t - \alpha P_{old} \cdot g_t, \tag{1}$$

where $W_0 = W_{old}$ and g_t is the gradient only calculated on new data. With the null-space projection, we can update the model that can remember the knowledge of W_{old}.

Adam-NSCL can preserve very well the previous knowledge, while the the updates of new task are limited because of the strong null-space projection. Our method can be viewed as an extension of Adam-NSCL, which achieves a better balance model for both previous and current tasks.

2.2. Linear Mode Connectivity

Optimizing a neural network involves finding a minimum in a high-dimensional non-convex objective landscape, where some forms of stochastic gradient descent (SGD) are used as optimization methods for learning the parameters of deep network. Since the deep neural network are non-convex, there are many local minima. Given a deep network $\mathcal{F}$ with an initial weight W_0, the weight is iteratively updated and the learnt weight at epoch k is denoted as $W_k = Train(\mathcal{F}, W_0)$. Two copies of the deep networks are trained (e.g., using different data augmentations or projections), producing two optimized weights $W_k^1 = Train_1(\mathcal{F}, W_0)$ and $W_k^2 = Train_2(\mathcal{F}, W_0)$.

Recently, a lot of work [6, 8, 38] has been developed to study the neural network optimization landscape. Many intriguing phenomena have been found. For example, one of the interesting observations [10, 14] is that there exists a connector between two optima. The loss minima are not isolated.

Observation 1 (Connectivity) *[10, 14] There exists a continuous path between minima of neural network architectures, where each point along this path has a low loss.*

To find the continuous path, e.g., from W_k^1 to W_k^2, Draxler [10] proposed a method based on Nudged Elastic Band (NEB) [19] to find the smooth and low-loss nonlinear path. Further, [13] showed that two minima can be connected by a low-loss linear path in some cases.

Observation 2 (Linear Connectivity) *[13, 38] There exists a linear connector from W_k^1 to W_k^2 when W_0 is not randomly initialization but is trained to a certain spawn epoch.*

This condition is easy to satisfy. When the optimization trajectory of W_0 is shared, the two optima can be connected in a linear path. Inspired by this, MC-SGD [29] enforces the final weight linearly connected to all tasks' minima. However, MC-SGD stores a small set of previous samples to learn the linear connector. It can not be used in our data-free setting. In this paper, we don't rely on experience replay to force connectivity. Instead, we move the previous and current models closer to ensure connectivity.

3. Method

In this section, we first give the problem formulation of incremental learning problem. Let sequential incremental learning tasks be denoted as $T_1, T_2, \ldots, T_t, \ldots$, and each task includes a set of disjoint classes. In the t-th task, we are only given the t-th training dataset $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$, where N_t is the number of training samples, and the previous model $W_{1:(t-1)}$. We need to update the previous model $W_{1:(t-1)}$ to a new model $W_{1:t}$ such that two inherent properties should be considered: 1) *stability*: the new model should retain the knowledge of previous $t-1$ tasks, and 2) *plasticity*: the ability to learn the new knowledge of the t-th task.

We first give an analysis of how to find high-accuracy pathway. Then, according to the upper bound and linear connectivity, we design a simple linear connector.

3.1. How to Build High-accuracy Pathway for All Tasks?

Suppose that there are K disjoint sequential incremental learning tasks, e.g., two tasks on Figure 1 where the ellipse on the left is the set of optimal weights for task 1 and the right ellipse is the set of optimal models for task 2, can we simply connect any two optimal models of two tasks? For example, the linear connection of $\hat{W}_1$ and $\hat{W}_2$? It may fail since the pathway from $\hat{W}_1$ to $\hat{W}_2$ may result in poor performance. How to find high-accuracy pathway for all tasks is not a trivial problem. Observed on Figure 1, if we can move $\hat{W}_1$ and $\hat{W}_2$ towards the overlapping region, the high-accuracy path between them will be easier to find. In fact, by moving the optimal weights closest, the upper bound of the empirical loss for all tasks will get smallest. Here comes the theoretical explanation.

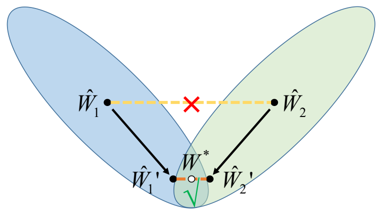

Figure 1. Illustration of how to find high-accuracy pathway, where $\hat{W}_1$ and $\hat{W}_2$ are one of optimal weights for task 1 and task 2, respectively. If they are moved closest to each other (e.g., towards the overlapped region), we can find a good linear path.

Let the optimal or convergent weight for task i be $\hat{W}_i$, which is only trained on the i-th task. The empirical loss for task i is denoted as $L_i(W)$. We aim to find a final weight for all tasks W^* that minimizes the empirical loss for all tasks, e.g., $W^* = arg\min_W \sum_{i=1}^{K} L_i(W)$.

First, we consider one task. For the first task, we can use Taylor expansion to approximate the loss. Following the way in [30], it can be formulated as:

$$\begin{aligned} L_1(W^*) \approx\, & L_1(\hat{W}_1) + (W^* - \hat{W}_1)^\top \nabla L_1(\hat{W}_1) \\ & + \frac{1}{2}(W^* - \hat{W}_1)^\top \nabla^2 L_1(\hat{W}_1)(W^* - \hat{W}_1) \\ \leq & L_1(\hat{W}_1) + \frac{1}{2}\lambda_1^{max} \parallel W^* - \hat{W}_1 \parallel^2, \end{aligned} \tag{2}$$

where $\nabla L_1(\hat{W}_1) \approx 0$ since $\hat{W}_1$ is the optimal weight and the gradient's norm vanishes, and λ_1^{max} is the maximum eigenvalue of $\nabla^2 L_1(\hat{W}_1)$. In the same way, for other tasks, we have:

$$\begin{aligned} L_2(W^*) &\leq L_2(\hat{W}_2) + \frac{1}{2}\lambda_2^{max} \parallel W^* - \hat{W}_2 \parallel^2, \\ &\vdots \\ L_K(W^*) &\leq L_K(\hat{W}_K) + \frac{1}{2}\lambda_K^{max} \parallel W^* - \hat{W}_K \parallel^2. \end{aligned} \tag{3}$$

By summing them up, we have:

$$\sum_{i=1}^{K} L_i(W^*) \leq \sum_{i=1}^{K} L_i(\hat{W}_i) + \frac{1}{2}\lambda^{max} \sum_{i=1}^{K} \parallel W^* - \hat{W}_i \parallel^2, \tag{4}$$

where $\lambda^{max} = \max(\lambda_1^{max}, \lambda_2^{max}, ..., \lambda_K^{max})$. Since $\hat{W}_i$ is the optimal weight for task i ($i = 1, \cdots, K$), $L_i(\hat{W}_i)$ is the minimum. Thus the first term $\sum_{i=1}^{K} L_i(\hat{W}_i)$ is also the minimum. Hence, the empirical loss for all tasks, i.e., $\sum_{i=1}^{K} L_i(W^*)$, can be bounded via minimizing the second term $\sum_{i=1}^{K} \parallel W^* - \hat{W}_i \parallel^2$. It is easy to verify that the optimal weight W^*_{opt} for the second term is:

$$W^*_{opt} = \frac{1}{K}\sum_{i=1}^{K} \hat{W}_i. \tag{5}$$

We can see that W^*_{opt} is the centroid or geometric center of $\{\hat{W}_1, \hat{W}_2, \cdots, \hat{W}_K\}$. Putting W^*_{opt} into $\frac{1}{2}\lambda^{max}\sum_{i=1}^{K} \parallel W^* - \hat{W}_i \parallel^2$, we have $\frac{1}{2}\lambda^{max}\sum\limits_{i=1}^{K} \parallel \frac{1}{K}\sum_{j=1}^{K}\hat{W}_j - \hat{W}_i \parallel^2 \leq \lambda^{max}\frac{1}{2K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} \parallel \hat{W}_i - \hat{W}_j \parallel^2$.

Combining the above inequation with Eq.(4), we have

$$\sum_{i=1}^{K} L_i(W^*_{opt}) \leq \sum_{i=1}^{K} L_i(\hat{W}_i) + \lambda^{max}\frac{1}{2K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} \parallel \hat{W}_i - \hat{W}_j \parallel^2 . \tag{6}$$

The upper bound in Eq. (6) can give us an interesting perspective of the incremental learning: the empirical loss of all tasks can be bounded by minimizing the sum of empirical loss of each individual task and the sum of squared Euclidean distances between each pair of optimal weights.

As indicated by the upper bound of Eq. (6), if we have 1) the previous model $\hat{W}_{old}$ and current model $\hat{W}_{new}$ achieve optimal solutions for previous $K-1$ tasks and current K-th task, respectively and 2) these two models are moved closest to each other, then we can simply use the linear connection of the two models: $W^*_{opt} = \frac{1}{K}\hat{W}_{new} + \frac{1}{K}\sum_{i=1}^{K-1}\hat{W}_{old} = \frac{1}{K}\hat{W}_{new} + \frac{K-1}{K}\hat{W}_{old}$ (see Eq. (5)). So that the upper bound of empirical loss for all tasks would be lowest.

3.2. Linear Connector for Plasticity-Stability Trade-off

According to the upper bound in Eq. (6), we train two independent neural networks, which separately consider plasticity and stability, and the two models are moved towards each other. Finally, we design a simple linear connector according to linear connectivity and Eq. (5).

3.2.1 Remembering Knowledge of Previous Tasks

As discussed in **Section 3.1**, the deep network considering stability should preserve the knowledge of past tasks and move towards the optimal set of the current task.

Algorithm 1: Linear connector for plasticity-stability trade-off

```
Input: A set of sequential learning tasks $T_1, T_2, \cdots$,
       and their training datasets $\mathcal{D}_1, \mathcal{D}_2, \cdots$; A
       neural network $W$ and learning rate $\alpha$
Train the first task to get $W_{1:1} = Train(\mathcal{D}_1)$
# compute the null space
Use the model $W_{1:1}$ and $\mathcal{D}_1$ to obtain feature
  covariance $\mathcal{X}_{1:1}$ and the null-space projection
  matrix $P_{1:1}$
for task $T_t \in \{T_2, T_3, \cdots\}$ do
    # init the two networks
    Let $\overleftarrow{W}^0_{1:(t-1)} = W_{1:(t-1)}$, $\overrightarrow{W}^0_t = W_{1:(t-1)}$ and
      $s = 0$
    while not converged do
        Sample a mini-batch $\{X, Y\}$ from $\mathcal{D}_t$
        $s = s + 1$
        Compute the gradient $\overleftarrow{g}$ and $\overrightarrow{g}$
        # preserve previous knowledge
        $\overleftarrow{W}^s_{1:(t-1)} = \overleftarrow{W}^{s-1}_{1:(t-1)} - \alpha \cdot P_{1:(t-1)} \cdot \overleftarrow{g}$
        # learn new knowledge
        $\overrightarrow{W}^s_t = \overrightarrow{W}^{s-1}_t - \alpha \cdot \overrightarrow{g}$
    # linear connector
    $W_{1:t} = \frac{t-1}{t}\overleftarrow{W}^s_{1:(t-1)} + \frac{1}{t}\overrightarrow{W}^s_t$
    # compute the null space
    Use the model $W_{1:t}$, $\mathcal{D}_t$ and $\mathcal{X}_{1:t-1}$ to obtain
      feature covariance $\mathcal{X}_{1:t}$ and the null-space
      projection matrix $P_{1:t}$
Output: $W_{1:t}$
```

Specially, we use Adam-NSCL to achieve the above goals. The previous model $W_{1:(t-1)}$ is used as the initialization of the deep network $\overleftarrow{W}_{1:(t-1)}$. At iteration s, we randomly sample a mini-batch $\{X, Y\}$ from $\mathcal{D}_t$, and cross-entropy loss function is used to learn the model. The objective function can be formulated as

$$\min_{\overleftarrow{W}_{1:(t-1)}} \mathcal{L}_{CE}(\overleftarrow{W}_{1:(t-1)}). \tag{7}$$

We calculate the gradient as $\overleftarrow{g}$. To preserve the previous knowledge, the gradient is multiplied by the null-space projection matrix. The feature covariance of all $t-1$ tasks is $\mathcal{X}_{1:(t-1)}$ and the projection matrix of all previous data is $P_{1:(t-1)} = UU^\top$, where U is the set of eigenvectors of $\mathcal{X}_{1:(t-1)}$ and their eigenvalues are zero. (Please refer to Algorithm 2 in [37] for more details of obtaining the feature covariance and projection matrix). Then, the weight is updated as

$$\overleftarrow{W}^s_{1:(t-1)} = \overleftarrow{W}^{s-1}_{1:(t-1)} - \alpha \cdot P_{1:(t-1)} \cdot \overleftarrow{g}, \tag{8}$$

where $\overleftarrow{W}^0_{1:(t-1)} = W_{1:(t-1)}$ and α is the stepsize. This updating strategy can ensure that it preserves the knowledge of past tasks, and also the model is moved towards the optimal set of the current task [37] .

3.2.2 Learning New Knowledge of Current Task

Here we update another deep network which considers plasticity. As discussed in **Section 3.1**, the new model $\overrightarrow{W}_t$ should 1) be the optimal model of current task (the first term in the right of Eq. (6)), and 2) be closer to the previous model (the second term in the right of Eq. (6)).

Specially, given the t-th training dataset $\mathcal{D}_t$, the objective function can be formulated as

$$\min_{\overrightarrow{W}_t} \mathcal{L}_{CE}(\overrightarrow{W}_t) + \mathcal{L}_D(\overrightarrow{W}_t), \tag{9}$$

where $\mathcal{L}_{CE}(\overrightarrow{W}_t)$ is the cross-entropy loss function aiming to learn the optimal weight of current task, and $\mathcal{L}_D(\overrightarrow{W}_t)$ aims to move the new model closer to the previous tasks. In general, the previous model $\overleftarrow{W}_{1:(t-1)}$ is not the optimal solution for the current task. Hence, simply using $\mathcal{L}_D(\overrightarrow{W}_t) = ||\overrightarrow{W}_t - \overleftarrow{W}_{1:(t-1)}||^2$ would hurt the performance of $\mathcal{L}_{CE}(\overrightarrow{W}_t)$. Instead, we use the feature distillation loss [45], which is formulated as

$$\mathcal{L}_D(\overrightarrow{W}_t) = \frac{1}{|\mathcal{D}_t|} \sum_{\{X,Y\} \sim \mathcal{D}_t} \parallel F_{new}(X) - F_{old}(X) \parallel^2, \tag{10}$$

where F_{new}/F_{old} are the feature extractors of $\overrightarrow{W}_t/W_{1:(t-1)}$, respectively. Please note that $\overrightarrow{W}_t$ consists of the feature extractor F_{new} followed by the classifier C_{new}. And $F_{new}(X)$/$F_{old}(X)$ are the features of X extracted by F_{new}/F_{old}, respectively. In this way, we can move the current model towards the previous tasks.

Given the previous model $W_{1:(t-1)}$ as the initialization, we can simply use SGD or Adam [20] to learn the knowledge of current task and the gradient is $\overrightarrow{g}$. At iteration s, the neural network is updated as

$$\overrightarrow{W}_t^s = \overrightarrow{W}_t^{s-1} - \alpha \cdot \overrightarrow{g}, \tag{11}$$

where $\overrightarrow{W}_t^0 = W_{1:(t-1)}$.

3.2.3 Plasticity-Stability Trade-off

Now we have two neural networks: $\overleftarrow{W}_{1:(t-1)}$ and $\overrightarrow{W}_t$. The $\overleftarrow{W}_{1:(t-1)}$ preserves the previous knowledge, and the $\overrightarrow{W}_t$ is the optimal weight of the current task. Formally, the linear connector between $\overleftarrow{W}_{1:(t-1)}$ and $\overrightarrow{W}_t$ is formulated as

$$(1-\beta)\overleftarrow{W}_{1:(t-1)} + \beta\overrightarrow{W}_t, \tag{12}$$

for $\beta \in [0, 1]$. According to Eq.(5), we set $\beta = \frac{1}{t}$, which means that we average the weights of all t tasks and get the final network as

$$W_{1:t} = \frac{t-1}{t}\overleftarrow{W}_{1:(t-1)} + \frac{1}{t}\overrightarrow{W}_t. \tag{13}$$

The averaging model $W_{1:t}$ can achieve notable improvement. Stochastic weight averaging [18] also used the averaging model, and they showed that such averaging model can converge to the wider solution with better generalization. Our method is summarized in Algorithm 1.

Linear interpolation: According to Observation 2, to make the two networks linearly connected, we firstly use $W_{1:(t-1)}$ as the initialization to update two models. Then, the two networks are trained in similar manners to arrive the optima. The linear connector provides us a simple method to control the balance between the forgetting and intransigence by changing the value of β : $W_{1:t} = (1 - \beta)\overleftarrow{W}_{1:(t-1)} + \beta\overrightarrow{W}_t$. If $\beta = 0$, our method becomes Adam-NSCL, which mainly focuses on remembering knowledge of previous tasks. When $\beta = 1$, it achieves excellent performance on the new task. Figure 2 shows the performances of the linear combinations with different β.

4. Experimental Results

In this section, we evaluate our model on various incremental learning tasks and compared it with several state-of-the-art baselines. Besides, we have conducted ablation study to see the performances of previous and current tasks with various β in **Eq. (12)**. And we also evaluate our model using the evaluation measures of stability and plasticity.

4.1. Datasets

CIFAR-100 [22] is a dataset including 100 classes of images with size of 32×32 and each class contains 500 images for training and 100 images for testing. **TinyImageNet** [39] contains 120,000 images of 200 classes. The images are downsized to 64×64 and each class contains 500 training images, 50 validation images and 50 test images. In this paper, the validation set of TinyImageNet is used for testing since the labels of test set are unavailable.

We split the dataset into K disjoint subsets of classes such that the training samples of each task are from a disjoint subset of C/K classes, where C is the total number of classes and K is the total number of tasks. When $K = 10$, we get **10-split-CIFAR-100**, and the labels for 10 tasks are $\{\{0-9\}, \{10-19\}, .., \{90-99\}\}$, respectively. When $K = 20$ and $K = 25$, we get **20-split-CIFAR-100** and **25-split-TinyImageNet** respectively in the same way. In task T_t, we only have access to $\mathcal{D}_t$ and no previous data is stored.

4.2. Implementation Detail

To make a fair comparison, we follow the experimental settings of Adam-NSCL [37]. Specifically, we use ResNet-18 as our backbone network and each task has its own single-layer linear classifier. When training in new task, we only update backbone network and classifier of new task, while classifiers of previous tasks remain unchanged. We use Adam optimizer, and the initial learning rate is set to 10^{-4} for the first task T_1 and 5×10^{-5} for both $\overleftarrow{W}^0_{1:(t-1)}$ and $\overrightarrow{W}^0_t$ in other tasks. The total number of epochs is 80 and the learning rate is reduced by half at epoch 30 and epoch 60. The batch size for 20-split-CIFAR-100 is set to 32 and 16 for another two datasets. For parameters that can not be updated by gradient descent method, e.g., $running\_mean$ of batch normalization layer, we also average them as **Eq. (13)**.

4.3. Evaluation Protocol

We use Average Accuracy (ACC) to measure how the model performs on all tasks. Here we denote the number of tasks as K. After finishing training from task T_1 to task T_m, the accuracy of model on test set of task t is denoted as $A_{m,t}$. ACC can be calculated as

$$\text{ACC} = \frac{1}{K}\sum_{t=1}^{K} A_{K,t}, \tag{14}$$

where K is the total number of tasks. The larger ACC is, the better the model performs. Since it's the average accuracy of all tasks, we must take the balance between tasks into account.

We use Backward Transfer (BWT) [27] to measure how much the model forgets in the continual-learning process. BWT is defined as

$$\text{BWT} = \frac{1}{K-1}\sum_{t=1}^{K-1} A_{K,t} - A_{t,t}. \tag{15}$$

It indicates the average accuracy drop of all previous tasks. The larger BWT is, the less model forgets. In this paper, we aim to achieve a more balanced model. Hence, ACC and BWT should be considered together. Given ACC and BWT two measures, we should firstly see the ACC: the larger value of ACC is better. When the two methods have the same ACC values, we can use the BWT to observe how two methods perform on stability and plasticity: the smaller BWT means the method is good at learning new knowledge but it forgets more, larger BWT means it forgets less but learns less new task.

4.4. Results

In this set of experiments, we compare our method with several state-of-the-art baselines. We compare our method with EWC [21], MAS [2] , MUC-MAS [26], SI [44], LwF [25], InstAParam [5], GD-WILD [23], GEM [27], A-GEM [4], MEGA [16], OWM [43] and Adam-NSCL [37]. All methods use ResNet-18 as backbone network for a fair comparison.

Methods	ACC(%)	BWT(%)
EWC	70.77	-2.83
MAS	66.93	-4.03
MUC-MAS	63.73	-3.38
SI	60.57	-5.17
LwF	70.70	-6.27
InstAParam	47.84	-11.92
GD-WILD	71.27	-18.24
GEM	49.48	2.77
A-GEM	49.57	-1.13
MEGA	54.17	-2.19
OWM	68.89	-1.88
Adam-NSCL	73.77	-1.6
Ours	**79.79**	-0.92

Table 1. Results on 10-split-CIFAR-100. Please note that a larger value of ACC is better.

Methods	ACC(%)	BWT(%)
EWC	71.66	-3.72
MAS	63.84	-6.29
MUC-MAS	67.22	-5.72
SI	59.76	-8.62
LwF	74.38	-9.11
InstAParam	51.04	-4.92
GD-WILD	77.16	-14.85
GEM	68.89	-1.2
A-GEM	61.91	-6.88
MEGA	64.98	-5.13
OWM	68.47	-3.37
Adam-NSCL	75.95	-3.66
Ours	**80.80**	-5.00

Table 2. Results on 20-split-CIFAR-100. A larger value of ACC is better and a moderate value of BWT is better for balanced model.

Table 1, Table 2 and Table 3 show the comparison results. The results show that our method achieves significant improvement w.r.t. ACC on three datasets. The results of BWT and ACC indicate that our method can achieve better plasticity-stability trade-off. Detailed analysis is as follows.

10-split-CIFAR-100 The results are shown in Table 1. We can see that our model achieves the best ACC 79.79%, which is 6.02% superior to the second best model Adam-NSCL. The BWT value of our model is -0.92%, which is the second best one compared to baselines. It indicates that our model can obtain a meaningful balance between previous

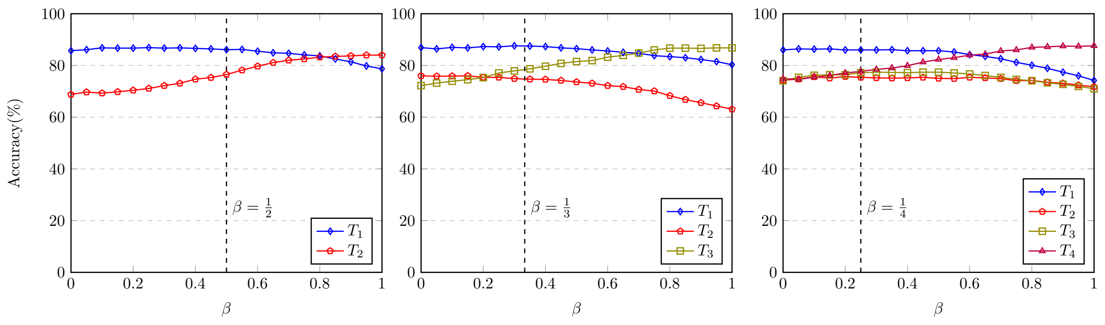

Figure 2. Accuracy of $W_{1:2}$(left), $W_{1:3}$(middle) and $W_{1:4}$(right) with different β on 10-split-CIFAR-100

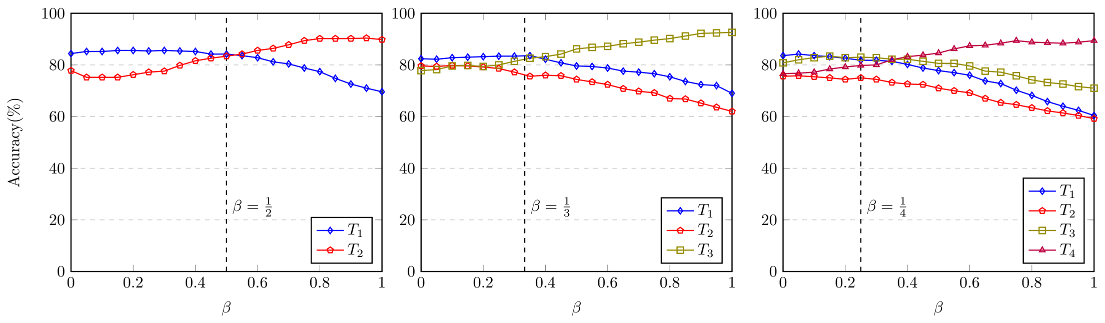

Figure 3. Accuracy of $W_{1:2}$(left), $W_{1:3}$(middle) and $W_{1:4}$(right) with different β on 20-split-CIFAR-100

Methods	ACC(%)	BWT(%)
EWC	52.33	-6.17
MAS	47.96	-7.04
MUC-MAS	41.18	-4.03
SI	45.27	-4.45
LwF	56.57	-11.19
InstAParam	34.64	-10.05
GD-WILD	42.74	-34.58
A-GEM	53.32	-7.68
MEGA	57.12	-5.90
OWM	49.98	-3.64
Adam-NSCL	58.28	-6.05
Our	**64.61**	-6.00

Table 3. Results on 25-split-TinyImageNet.

tasks and new task.

20-split-CIFAR-100 As shown in Table 2, our model still achieves the best ACC 80.80%, which is 3.64% better than the second best model GD-WILD. Note that GD-WILD stores previous data and its BWT value is 9.85% worse than ours. Again, our model achieves a relative balanced BWT value -5.00%.

25-split-TinyImageNet The results of Table 3 show that our method achieves the best ACC 64.61%, and the ACC of second best model Adam-NSCL is 58.28%. The BWT and ACC indicate that the our method not only can achieve better performance, but also obtain a more balanced model. Note that Adam-NSCL achieves an excellent performance, even so, our method performs better than Adam-NSCL.

In summary, two observations can be made from the results: 1) our method yields the best performance on all datasets. 2) Our method achieves a better trade-off between stability and plasticity. Please note that the performance (ACC) of a IL model can be divided into two parts: stability (BWT) and plasticity. Hence, knowing ACC and BWT, we can probably know the performance of plasticity. We will further discuss it in the next subsection.

4.5. Ablation Study

In this set of experiments, we conduct ablation study on three benchmark datasets to see the effects of β. As indicated by **Eq. (12)**, we use β to control the ratio of two independent neural networks.

For demonstration purposes, we only show three sequential learning tasks. The results of other tasks are similar. To be specific, when $t = 2$, T_1 is the previous task and T_2 is the current task. The test accuracies of $W_{1:2}$ using different values of β on tasks T_1 and T_2 are shown in the left of Figure 2, Figure 3 and Figure 4. The results of T_1 indicate the ability to preserve old knowledge, and the accuracies of T_2 indicate the ability to learn new task.

When $t = 3$, test accuracies of tasks T_1, T_2 and T_3 are shown in the middle of Figure 2, Figure 3 and Figure 4.

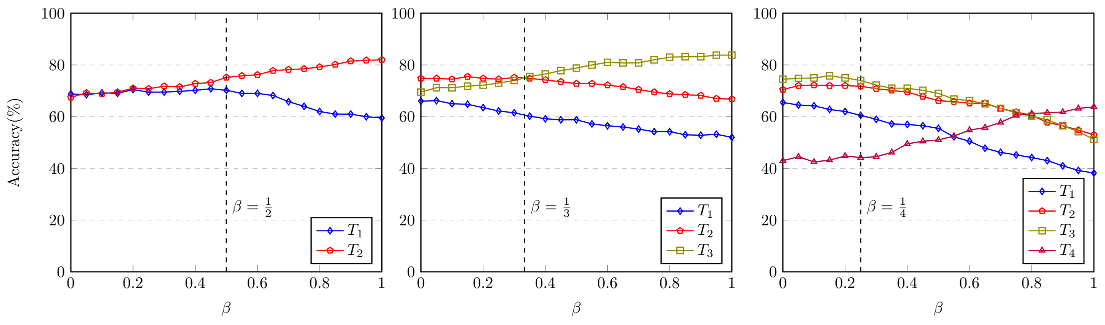

Figure 4. Accuracy of $W_{1:2}$(left), $W_{1:3}$(middle) and $W_{1:4}$(right) with different β on 25-split-TinyImageNet

When $t = 4$, test accucaries of T_1, T_2, T_3 and T_4 are shown in the right of Figure 2, Figure 3 and Figure 4.

As shown in Figure 2, Figure 3 and Figure 4, we can see that: 1) when $\beta = 0$, $W_{1:t} = \overleftarrow{W}_{1:(t-1)}$ can well preserve the previous knowledge. 2) When $\beta = 1$, $W_{1:t} = \overrightarrow{W}_t$ performs well on the new task. 3) The linear paths between $\overleftarrow{W}_{1:(t-1)}$ and $\overrightarrow{W}_t$ are almost smooth, and there are no obvious jumps along the paths. For example, the accuracy of T_2 increases as the value of β gets larger as shown in the left of Figure 2. 4) For 10-split-CIFAR-100 and 20-split-CIFAR-100, the model strikes a balance well on all tasks when β is close to $\frac{1}{t}$. For 25-split-TinyImageNet, though the fused model doesn't perform best on some tasks, $\beta = \frac{1}{t}$ is still the most compromising solution.

4.6. Plasticity-Stability Trade-off Analysis

To better understand our method, we compare it with Adam-NSCL to analyse the plasticity-stability trade-off.

We use BWT as the evaluation measure of stability. Further, we also use Intransigence Measure(IM) [3] to measure plasticity, which indicates how much the model has learnt from new task. The intransigence for the k-th task can be calculated as

$$I_k = A_k^* - A_{k,k}, \tag{16}$$

where A_k^* is the accuracy on the test set of k-th task with dataset $\cup_{i=1}^{k} \mathcal{D}_i$. The smaller the I_k is, the better the model is.

Table 4, Table 5 and Table 6 show the results of BWT and IM. First, the BWT values of our model are bigger than that of Adam-NSCL except for 20-split-CIFAR-100, which means that Adam-NSCL has stronger ability to remember the previous knowledge for 20-split-CIFAR-100. Second, the IM values of our model are much better than Adam-NSCL. Our method considers both the stability and plasticity, and the overall effect makes the ACC higher.

Methods	ACC	BWT(%)	I_{10}(%)
Adam-NSCL	73.77	-1.6	14.50
Ours	**79.79**	**-0.92**	**8.10**

Table 4. BWT and IM on 10-split-CIFAR-100

Methods	ACC	BWT(%)	I_{20}(%)
Adam-NSCL	75.95	**-3.66**	12.60
Ours	**80.80**	-5.00	**7.00**

Table 5. BWT and IM on 20-split-CIFAR-100

Methods	ACC	BWT(%)	I_{25}(%)
Adam-NSCL	58.28	-6.05	10.50
Ours	**64.61**	**-6.00**	**5.75**

Table 6. BWT and IM on 25-split-TinyImageNet

5. Conclusion

In this paper, we proposed a simple linear connector for incremental learning, which is a better plasticity-stability trade-off solution. To explain why we can use a simple linear connector to combine two models, we had given an analysis and showed it can minimize the upper bound of empirical loss for all tasks. Hence, we proposed two independent neural networks. The first network aims to preserve the previous knowledge and the second network is to learn new knowledge. We used the null-space projection to learn the first network and the SGD for the second network. Finally, we simply averaged the two network and achieved a significant improvement. In our future work, we aim to find a better way to combine the two networks and give a better theoretical explanation for non-linear/linear connector.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grants (U1811261,U1811262), Guangdong Basic and Applied Basic Research Foundation (2019B1515130001, 2021A1515012172) and Zhuhai Industry-University-Research Cooperation Project (ZH22017001210010PWC).

References

[1] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems*, volume 32, pages 4392–4402, 2019. 2

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 144–161, 2018. 2, 6

[3] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018. 1, 8

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2018. 2, 6

[5] Hung-Jen Chen, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Mitigating forgetting in online continual learning via instance-aware parameterization. *Advances in Neural Information Processing Systems*, 33:17466–17477, 2020. 6

[6] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38, pages 192–204, 2015. 3

[7] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 1

[8] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *ICML'17 Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pages 1019–1028, 2017. 3

[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, 2020. 1

[10] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning*, pages 1308–1317, 2018. 3

[11] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. In *ICLR 2020 : Eighth International Conference on Learning Representations*, 2020. 2

[12] Stanislav Fort and Stanislaw Jastrzebski. Large scale structure of neural network loss landscapes. In *Advances in Neural Information Processing Systems*, volume 32, pages 6706–6714, 2019. 1

[13] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *ICML 2020: 37th International Conference on Machine Learning*, volume 1, pages 3259–3269, 2020. 1, 2, 3

[14] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P. Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *32nd Conference on Neural Information Processing Systems, NeurIPS 2018*, volume 31, pages 8789–8798, 2018. 1, 3

[15] Alexander Gepperth and Cem Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8(5):924–934, 2016. 2

[16] Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory based lifelong learning algorithm. In *Conference on Neural Information Processing Systems*, 2020. 6

[17] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *AAAI Conference on Artificial Intelligence 2018*, pages 3302–3309, 2018. 2

[18] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885, 2018. 5

[19] Hannes Jonsson, G. Mills, and Karsten Wedel Jacobsen. Nudged elastic band method for finding minimum energy paths of transitions. In *CLassical and Quantum Dynamics in Condensed Phase Simulations*, volume 385, pages 385–404, 1998. 3

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[21] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017. 1, 2, 6

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[23] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 312–321, 2019. 2, 6

[24] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pages 3925–3934, 2019. 1, 2

[25] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. 2, 6

[26] Yu Liu, Sarah Parisot, Gregory Slabaugh, Xu Jia, Ales Leonardis, and Tinne Tuytelaars. More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In *Computer Vision–ECCV 2020: 16th European*

Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16, pages 699–716. Springer, 2020. 6

[27] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 6467–6476, 2017. 1, 2, 6

[28] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4:504–504, 2013. 1

[29] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*, 2021. 1, 3

[30] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. In *NeurIPS*, 2020. 3

[31] Martin Mundt, Yong Won Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *arXiv preprint arXiv:2009.01797*, 2020. 1, 2

[32] Amal Rannen, Rahaf Aljundi, Matthew B. Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1329–1337, 2017. 2

[33] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017, pages 5533–5542, 2017. 1, 2

[34] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *The 35th International Conference on Machine Learning (ICML 2018)*, pages 4548–4557, 2018. 2

[35] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, volume 30, pages 2990–2999, 2017. 2

[36] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *ICCV*, pages 9374–9384, 2021. 1

[37] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021. 1, 2, 4, 5, 6

[38] Mitchell Wortsman, Maxwell Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning neural network subspaces. In *ICML 2021: 38th International Conference on Machine Learning*, 2021. 1, 3

[39] Jiayu Wu, Qixiang Zhang, and Guoxi Xu. Tiny imagenet challenge. *Technical Report*, 2017. 5

[40] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6619–6628, 2019. 1, 2

[41] 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. 1, 2

[42] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung ju Hwang. Lifelong learning with dynamically expandable networks. In *Sixth International Conference on Learning Representations*, 2017. 2

[43] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019. 2, 6

[44] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, volume 70, pages 3987–3995, 2017. 2, 6

[45] Linfeng Zhang, Yukang Shi, Zuoqiang Shi, Kaisheng Ma, and Chenglong Bao. Task-oriented feature distillation. In *Advances in Neural Information Processing Systems*, volume 33, pages 14759–14771, 2020. 5