Training Networks in Null Space of Feature Covariance for Continual Learning

Shipeng Wang[1], Xiaorong Li[1], Jian Sun(✉)[1,2,3], Zongben Xu[1,2,3]

[1] School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, 710049, China
[2] National Engineering Laboratory of Big Data Algorithm and Analysis Technology, Xi'an, 710049, China
[3] Pazhou Lab, Guangzhou, Guangdong, 510335, China

{wangshipeng8128, lixiaorong}@stu.xjtu.edu.cn, {jiansun, zbxu}@xjtu.edu.cn

Abstract

In the setting of continual learning, a network is trained on a sequence of tasks, and suffers from catastrophic forgetting. To balance plasticity and stability of network in continual learning, in this paper, we propose a novel network training algorithm Adam-NSCL which sequentially optimizes network parameters in the null space of all previous tasks. We first propose two mathematical conditions respectively for achieving network stability and plasticity in continual learning. Based on them, the network training for sequential tasks without forgetting can be simply achieved by projecting the candidate parameter update into the approximate null space of all previous tasks in the network training process, where the candidate parameter update can be generated by Adam. The approximate null space can be derived by applying singular value decomposition to the uncentered covariance matrix of all input features of previous tasks for each linear layer. For efficiency, the uncentered covariance matrix can be incrementally computed after learning each task. We also empirically verify the rationality of the approximate null space at each linear layer. We apply our approach to training networks for continual learning on benchmark datasets of CIFAR-100 and TinyImageNet, and the results suggest that the proposed approach outperforms or matches the state-ot-the-art continual learning approaches.

1. Introduction

Deep neural networks have achieved promising performance on various tasks in natural language processing, machine intelligence, etc., [5, 9, 47, 48, 51]. However, the ability of deep neural networks for continual learning is limited, where the network is expected to continually learn knowledge from sequential tasks [15]. The main challenge for continual learning is how to overcome catastrophic forgetting [11, 32, 42], which has drawn much attention recently.

In the context of continual learning, a network is trained on a stream of tasks sequentially. The network is required to have *plasticity* to learn new knowledge from current task, and also *stability* to retain its performance on previous tasks. However, it is challenging to simultaneously achieve plasticity and stability in continual learning for neural networks, and catastrophic forgetting always occurs. This phenomenon is called *plasticity-stability dilemma* [33].

Recently, various strategies for continual learning have been explored, including regularization-based, distillation-based, architecture-based, replay-based and algorithm-based strategies. The regularization-based strategy focuses on penalizing the variations of parameters across tasks, such as EWC [21]. The distillation-based strategy is inspired by knowledge distillation [14], such as LwF [26]. The architecture-based strategy modifies the architecture of network on different tasks, such as [1, 25]. The replay-based strategy utilizes data from previous tasks or pseudo-data to maintain the network performance on previous tasks, such as [4, 36]. The algorithm-based strategy designs network parameter updating rule to alleviate performance degradation on previous tasks, such as GEM [30], A-GEM [7] and OWM [53].

In this paper, we focus on the setting of continual learning where the datasets from previous tasks are inaccessible. We first propose two theoretical conditions respectively for stability and plasticity of neural networks in continual learning. Based on them, we design a novel network training algorithm called Adam-NSCL for continual learning, which forces the network parameter update to lie in the null space of the input features of previous tasks at each network layer, as shown in Fig. 1. The layer-wise null space of input features can be modeled as the null space of the uncentered covariance of these features, which can be incrementally computed after learning each task. Since it is too strict to guarantee the existence of null space, we approximate the null space of each layer by the subspace spanned by singular vectors corresponding to the smallest singular values of the uncentered covariance of input features. We embed this strategy into the Adam optimization algorithm by pro-
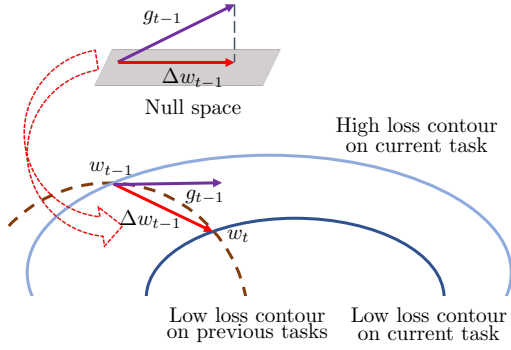
Figure 1. To avoid forgetting, we train network in the layer-wise null space of the corresponding uncentered covariance of all input features of previous tasks.

jecting the candidate parameter update generated by Adam [20] into the approximate null space layer by layer, which is flexible and easy to implement.

We conduct various experiments on continual learning benchmarks in the setting that the datasets of previous tasks are unavailable, and results show that our Adam-NSCL is effective and outperforms the state-of-the-art continual learning methods. We also empirically verify the rationality of the approximate null space.

The paper is organized as follows. We first introduce related works in Sec. 2. In Sec. 3, we present the mathematical conditions and then propose network training algorithm for continual learning in Sec. 4. In Sec. 5, we conduct experiments to verify the efficacy of our approach.

2. Related Work

We next review the related works of continual learning in the following five categories.

Regularization-based strategy. The basic idea of this strategy is penalizing the changes of network parameters when learning current task to prevent catastrophic forgetting. The typical methods include EWC [21], SI [54], MAS [3], RWalk [6] and NPC [38]. They impose regularization on the network parameters, and each network parameter is associated with an importance weight computed by different methods. These importance weights are required to be stored in continual learning. Under the Bayesian framework, VCL [35], CLAW [2] and IMM [24] take the posterior distribution of network parameters learned from previous tasks as the prior distribution of network parameters on current task, which implicitly penalizes the changes of network parameters.

Distillation-based strategy. Inspired by knowledge distillation [14], this strategy takes the network learned from previous tasks as teacher and the network being trained on current task as student, and utilize a distillation term to alleviate performance degradation on previous tasks, such

as LwF [26], GD-WILD [23], lifelong GAN [55], MCIL [29], LwM [10], etc. Due to inaccessibility to the full datasets of previous tasks, they respectively use data of current task [10, 26], external data [23], coreset of previous tasks [29, 43] or synthetic data [55], resulting in distributional shift [27] from the original datasets.

Replay-based strategy. The replay-based strategy trains networks using both data of the current task and "replayed" data of previous tasks. Some existing works focus on selecting a subset of data from previous tasks [4, 40], resulting in imbalance between the scale of datasets from current and previous tasks [50, 56]. An alternative approach is to learn generative model to generate synthetic data to substitute the original data [16, 19, 37, 46]. They do not need to store data of previous tasks, however, the performance is significantly affected by the quality of generated data, especially for complex natural images.

Architecture-based strategy. In this strategy, the network architecture is dynamically modified by expansion or mask operation when encountering new tasks in continual learning. Methods of network expansion modify the network architecture by increasing network width or depth to break its representational limit when facing new tasks [17, 25, 52]. This strategy may result in a powerful but redundant network that is computationally expensive and memory intensive. An alternative approach is assigning different sub-networks to different tasks by masking the neurons [1, 41, 45] or weights [31]. The mask associated with each task needs to be learned and stored in memory.

Algorithm-based strategy. This strategy avoids forgetting from the perspective of network training algorithm. It focuses on designing network parameter updating rule to guarantee that training on current task should not deteriorate performance on previous tasks. GEM [30] computes the parameter update by solving a quadratic optimization problem constraining the angle between the parameter update and the gradient of network parameters on data of previous tasks. A-GEM [7] is an improved GEM without solving quadratic optimization problem. A-GEM constrains that the network parameter update should be well aligned with a reference gradient computed from a random batch of data from the previous tasks. Both of GEM and A-GEM need to store data of previous tasks. Different from GEM and A-GEM, OWM [53] projects the parameter update into the orthogonal space of the space spanned by input features of each linear layer. The computation of the space projection matrix relies on the unstable inversion of matrix.

Our method called Adam-NSCL is a novel network training algorithm for continual learning, which trains networks in the approximate null space of feature covariance matrix of previous tasks to balance the network plasticity and stability. It does not require to design regularizers, revise network architecture and use replayed data in

our method. Compared with OWM, Adam-NSCL relies on the null space of feature covariance for achieving network plasticity and stability with theoretical and empirical analysis, and overcomes the unstable matrix inversion in OWM. More discussions on the differences are in Sec. 5.2.

3. Analysis of Stability and Plasticity

In this section, we first present the preliminaries on the setting of continual learning, then propose mathematical conditions on the network parameter update for the stability and plasticity, as the basis of our algorithm in Sec. 4.

3.1. Preliminaries

In the setting of continual learning, a network f with parameters $\mathbf{w}$ is sequentially trained on a stream of tasks $\{\mathcal{T}_1, \mathcal{T}_2, \dots\}$, where task $\mathcal{T}_t$ is associated with paired dataset $\{X_t, Y_t\}$ of size n_t. The output of network f on data X_t is denoted as $f(X_t, \mathbf{w})$.

The initial parameters of network f with L linear layers on task $\mathcal{T}_t$ are set as $\tilde{\mathbf{w}}_{t-1} = \{\tilde{w}_{t-1}^1, \dots, \tilde{w}_{t-1}^L\}$ which are the optimal parameters after trained on task $\mathcal{T}_{t-1}$. When training f on task $\mathcal{T}_t$ at the s-th training step, we denote the network parameters as $\mathbf{w}_{t,s} = \{w_{t,s}^1, \dots, w_{t,s}^L\}$. Correspondingly, the parameter update at the s-th training step on task $\mathcal{T}_t$ is denoted as $\Delta\mathbf{w}_{t,s} = \{\Delta w_{t,s}^1, \dots, \Delta w_{t,s}^L\}$. When feeding data X_p from task $\mathcal{T}_p$ ($p \leq t$) to f with optimal parameters $\tilde{\mathbf{w}}_t$ on task $\mathcal{T}_t$, the input feature and output feature at the l-th linear layer are denoted as $X_{p,t}^l$ and $O_{p,t}^l$, then

$$O_{p,t}^l = X_{p,t}^l \tilde{w}_t^l, \quad X_{p,t}^{l+1} = \sigma_l(O_{p,t}^l)$$

with σ_l as the nonlinear function and $X_{p,t}^1 = X_p$.

For the convolutional layer, we can reformulate convolution as the above matrix multiplication. Specifically, for each 3-D feature map, we flat each patch as a row vector, where the patch size is same as the corresponding 3-D convolutional kernel size, and the number of patches is the times that the kernel slides on the feature map when convolution. Then these row-wise vectors are concatenated to construct the 2-D feature matrix $X_{p,t}^l$. 3-D kernels at the same layer are flatted as column vectors of the 2-D parameter matrix.

3.2. Conditions for continual learning

When being trained on the current task $\mathcal{T}_t$ without training data from previous tasks, the network f is expected to perform well on the previous tasks, which is challenging since the network suffers from catastrophic forgetting. To alleviate the plasticity-stability dilemma, we propose two conditions for continual learning that guarantee the stability and plasticity respectively as follows.

To derive Condition 1 for network stability, we first present the condition of network parameter update on succeeding training tasks to avoid forgetting in Lemma 1 and

Lemma 2, depending on data of previous tasks. Then, we further propose the equivalent condition in Condition 1, free from storing data of previous tasks. After that, we present Condition 2 for network plasticity.

Lemma 1. *Given the data X_p from task $\mathcal{T}_p$, and the network f with L linear layers is trained on task $\mathcal{T}_t$ ($t > p$). If network parameter update $\Delta w_{t,s}^l$ lies in the null space of $X_{p,t-1}^l$, i.e.,*

$$X_{p,t-1}^l \Delta w_{t,s}^l = 0, \tag{1}$$

at each training step s, for the l-th layer of f ($l = 1, \dots, L$), we have $X_{p,t}^l = X_{p,t-1}^l$ and $f(X_p, \tilde{\mathbf{w}}_{t-1}) = f(X_p, \tilde{\mathbf{w}}_t)$.

Proof. Please refer to the supplemental material. □

Lemma 1 tells us that, when we train network on task $\mathcal{T}_t$, the network retains its training loss on data X_p in the training process, if the network parameter update satisfies Eqn. (1) at each training step. Considering that we initialize parameters $\mathbf{w}_{t,0}$ by $\tilde{\mathbf{w}}_{t-1}$, *i.e.*, the optimal parameters on task $t - 1$ for $t > 1$, we have the following corollary.

Corollary 1. *Assume that network f is sequentially trained on tasks $\{\mathcal{T}_1, \mathcal{T}_2, \cdots\}$. For each task $\mathcal{T}_t$ ($t > 1$) and $p < t$, if Eqn. (1) holds at every training step on task $\mathcal{T}_t$, we have $X_{p,p}^l = X_{p,t}^l$ ($l = 1, \cdots, L$) and $f(X_p, \tilde{\mathbf{w}}_t) = f(X_p, \tilde{\mathbf{w}}_p)$.*

Corollary 1 suggests that the training loss on data X_p is retained if the network trained on the following tasks satisfies Eqn. (1) and network parameters at each task are initialized by the trained network of the last task.

We further denote $\bar{X}_{t-1}^l = [X_{1,1}^{l\top}, \cdots, X_{t-1,t-1}^{l\top}]^\top$, which is the concatenation of input features of l-th network layer on each task data X_p ($p < t$) using trained network parameters on task $\mathcal{T}_p$. Then the following lemma holds.

Lemma 2. *Assume that f is being trained on task $\mathcal{T}_t$ ($t > 1$). If $\Delta w_{t,s}^l$ lies in the null space of $\bar{X}_{t-1}^l$ at each training step s, i.e.,*

$$\bar{X}_{t-1}^l \Delta w_{t,s}^l = 0, \tag{2}$$

for $l = 1, \cdots, L$, we have $f(X_p, \tilde{\mathbf{w}}_t) = f(X_p, \tilde{\mathbf{w}}_p)$ for all $p = 1, \cdots, t - 1$.

Lemma 2 guarantees the stability of f. However, it is inefficient since it requires to store all features $X_{p,p}^l$ of f for all $p < t$, which is memory-prohibited. To overcome this limitation, we propose the following Condition 1 based on uncentered feature covariance $\bar{\mathcal{X}}_{t-1}^l \triangleq \frac{1}{\bar{n}_{t-1}}(\bar{X}_{t-1}^l)^\top \bar{X}_{t-1}^l$ to guarantee stability, where $\bar{n}_{t-1}$ is the total number of seen data, *i.e.*, the number of rows of $\bar{X}_{t-1}^l$.

Condition 1 (stability). *When f is being trained on task $\mathcal{T}_t$, $\Delta w_{t,s}^l$ at each training step s should lie in the null space of the uncentered feature covariance matrix $\bar{\mathcal{X}}_{t-1}^l$ for $l = 1, \cdots, L$, i.e.,*

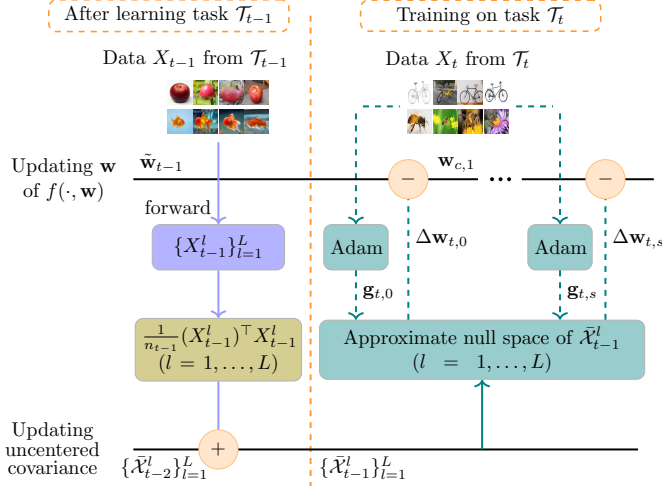$$\bar{\mathcal{X}}_{t-1}^l \Delta w_{t,s}^l = 0. \tag{3}$$

Figure 2. The pipeline of our algorithm.

It is easy to verify that the null space of $\bar{X}_{t-1}^l$ equals to null space of the uncentered feature covariance of $\bar{\mathcal{X}}_{t-1}^l$. Therefore, if Condition 1 holds, we have $f(X_p, \tilde{\mathbf{w}}_t) = f(X_p, \tilde{\mathbf{w}}_p)$ for all $p < t$ according to Lemma 2, *i.e.*, the performance of f on previous task data will not be degraded after learning current task.

Memory analysis. The memory consumption of storing $\bar{\mathcal{X}}_t^l$ is fixed, irrelevant to the number of tasks and data. Specifically, if we denote the dimension of the feature at layer l as h^l, the size of $\bar{X}_t^l$ is $\bar{n}_t \times h^l$ (usually $\bar{n}_t \gg h^l$), while the size of $\bar{\mathcal{X}}_t^l$ is $h^l \times h^l$. Therefore, Condition 1 supplies a more memory efficient way to guarantee the stability of f than Lemma 2.

Condition 1 guarantees the stability of network in continual learning. The other requirement of continual learning is the plasticity of f concerning the ability to obtain new knowledge from current task. Condition 2 will provide the condition that guarantees the plasticity of network f.

Condition 2 (plasticity). *Assume that the network f is being trained on task $\mathcal{T}_t$, and $\mathbf{g}_{t,s} = \{g_{t,s}^1, \ldots, g_{t,s}^L\}$ denotes the parameter update generated by a gradient-descent training algorithm for training f at training step s. $\langle \Delta \mathbf{w}_{t,s}, \mathbf{g}_{t,s} \rangle > 0$ should hold where $\langle \cdot, \cdot \rangle$ represents inner product.*

If parameter update $\Delta \mathbf{w}_{t,s}$ satisfies Condition 2 when training f on task $\mathcal{T}_t$, the training loss after updating parameters using $\Delta \mathbf{w}_{t,s}$ will decrease, *i.e.*, the network can learn the knowledge of this task. Please see supp. for proof.

4. Network Training in Covariance Null Space

In this section, we propose a novel network training algorithm called Adam-NSCL for continual learning based on Adam and the proposed two conditions respectively for

stability and plasticity. In continual learning, we maintain layer-wise feature covariance matrices which are incrementally updated using features of the coming task. Given the current task, starting from the network trained on previous tasks, we update the network parameters on current task using Adam-NSCL where the candidate parameter update generated by Adam is projected into the approximate null space of corresponding feature covariance matrix layer by layer to balance the network stability and plasticity.

Figure 2 illustrates the pipeline of our proposed continual learning algorithm. After learning task $\mathcal{T}_{t-1}$, we feed X_{t-1} to the learned network $f(\cdot, \tilde{\mathbf{w}}_{t-1})$ to get the input feature X_{t-1}^l at each layer. Then we compute the uncentered covariance $\mathcal{X}_{t-1}^l = \frac{1}{n_{t-1}}(X_{t-1}^l)^\top X_{t-1}^l$ with n_{t-1} as the number of data on task $\mathcal{T}_{t-1}$. Subsequently, we update the uncentered feature covariance matrix for each layer by

$$\bar{\mathcal{X}}_{t-1}^l = \frac{\bar{n}_{t-2}}{\bar{n}_{t-1}} \bar{\mathcal{X}}_{t-2}^l + \frac{n_{t-1}}{\bar{n}_{t-1}} \mathcal{X}_{t-1}^l, \tag{4}$$

with $\bar{n}_{t-1} = \bar{n}_{t-2} + n_{t-1}$ as the total number of seen data. Following that, we compute the approximate null space of $\bar{\mathcal{X}}_{t-1}^l$. When training the network with $\tilde{\mathbf{w}}_{t-1}$ as initialization on task $\mathcal{T}_t$, we first utilize Adam to generate a candidate parameter update $\mathbf{g}_{t,s}$ at the s-th step ($s = 1, 2, \ldots$), then project $\mathbf{g}_{t,s}$ into the approximate null space of $\bar{\mathcal{X}}_{t-1}^l$ layer by layer to get the parameter update $\Delta \mathbf{w}_{t,s}$.

In the following, we first introduce derivation of the approximate null space in Sec. 4.1, and discuss the projection satisfying Conditions 1 and 2. Subsequently, we present our proposed continual learning algorithm in Algorithm 1 and the way to find the approximate null space in Algorithm 2.

4.1. Approximate null space

According to Condition 1, for the stability of network in continual learning, we can force the parameter update to lie in the null space of uncentered covariance of all previous input features at each network layer. However, it is too strict to guarantee the existence of null space. Therefore, we propose Algorithm 2 to find the approximate null space based on SVD of the uncentered covariance matrix.

By applying SVD to $\bar{\mathcal{X}}_{t-1}^l$, we have

$$U^l, \Lambda^l, (U^l)^\top = \text{SVD}(\bar{\mathcal{X}}_{t-1}^l), \tag{5}$$

where $U^l = [U_1^l, U_2^l]$ and $\Lambda^l = \begin{bmatrix} \Lambda_1^l & 0 \\ 0 & \Lambda_2^l \end{bmatrix}$. If all singular values of zero are in Λ_2^l, *i.e.*, $\Lambda_2^l = 0$, then $\bar{\mathcal{X}}_{t-1}^l U_2^l = U_1^l \Lambda_1^l (U_1^l)^\top U_2^l = 0$ holds, since U^l is an unitary matrix. It suggests that the range space of U_2^l is the null space of $\bar{\mathcal{X}}_{t-1}^l$. Thus we can get the parameter update $\Delta w_{t,s}^l$ lying in the null space of $\bar{\mathcal{X}}_{t-1}^l$ by

$$\Delta w_{t,s}^l = U_2^l (U_2^l)^\top g_{t,s}^l \tag{6}$$

with $U_2^l(U_2^l)^\top$ as projection operator [34, Eqn. (5.13.4)]. Thus we get $\Delta w_{t,s}^l$ satisfying Condition 1.

However, it is unrealistic to guarantee that there exists zero singular values. Inspired by Principal Component Analysis, if considering U_1^l as principal components, $\bar{\mathcal{X}}_{t-1}^l$ can be approximated by $U_1^l\Lambda_1^l(U_1^l)^\top$, which indicates that $\bar{\mathcal{X}}_{t-1}^l U_2^l \approx U_1^l\Lambda_1^l(U_1^l)^\top U_2^l = 0$, *i.e.*, we can take the range space of U_2^l as the approximate null space of $\bar{\mathcal{X}}_{t-1}^l$ where U_2^l corresponds to the smallest singular values in Λ_2^l. We adaptively select Λ_2^l with diagonal singular values $\lambda \in \{\lambda | \lambda \le a\lambda_{\min}^l\}$ $(a > 0)$, where $\lambda_{\min}^l$ is the smallest singular value. Furthermore, to empirically verify the rationality of the approximation, we utilize the proportion R of $\bar{\mathcal{X}}_{t-1}^l$ explained by U_2^l [18] as

$$R = \frac{\Sigma_{\lambda\in\text{diag}\{\Lambda_2^l\}}\lambda}{\Sigma_{\lambda\in\text{diag}\{\Lambda^l\}}\lambda}, \quad (7)$$

where "diag" denotes the diagonal elements. If R is small, the sum of singular values of Λ_2^l is negligible, suggesting that it is reasonable to approximate null space of uncentered covariance matrix by the range space of U_2^l.

Example. We take the pretrained ResNet-18 on dataset of ImageNet ILSVRC 2012 [44] as example. Figure 3 shows the curves of singular values of uncentered covariance matrix $\bar{\mathcal{X}}_t^l$ of each linear layer indexed by l with $a = 50$. All proportions R of different layers are smaller than 0.05, indicating that the selected U_2^l corresponding to smallest singular values is negligible to explain $\bar{\mathcal{X}}_t^l$. Therefore, it is reasonable to approximate the null space by the range space of U_2^l.
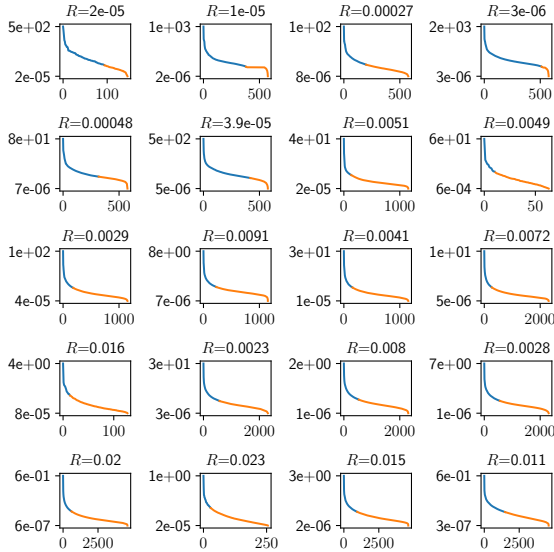

Figure 3. Singular values of uncentered covariance matrix at different layers of pretrained ResNet-18 on ImageNet ILSVRC 2012. Orange curves denote the singular values smaller than $50\lambda_{\min}^l$.

Algorithm 1 Adam-NSCL for continual learning
Inputs: Datasets $\{X_t, Y_t\}$ for task $\mathcal{T}_t \in \{\mathcal{T}_1, \mathcal{T}_2, \dots\}$, network $f(\cdot, \mathbf{w})$ with L linear layers, learning rate α.
Initialization: Initialize $\tilde{\mathbf{w}}_0$ randomly, $\bar{\mathcal{X}}_0^l = 0$, number of seen data $\bar{n}_0 = 0$.
1: # *sequential tasks*
2: **for** task $\mathcal{T}_t \in \{\mathcal{T}_1, \mathcal{T}_2, \dots\}$ **do**
3:    **if** $t > 1$ **then**
4:       # *compute the approximate null space*
5:       Get U_2^l, $\bar{\mathcal{X}}_{t-1}^l$ and $\bar{n}_{t-1}$ $(l = 1, \dots, L)$ by Algorithm 2 with $\{X_{t-1}, Y_{t-1}\}$, $f(\cdot, \tilde{\mathbf{w}}_{t-1})$, $\bar{\mathcal{X}}_{t-2}^l$ and $\bar{n}_{t-2}$ as inputs.
6:    **end if**
7:    # *train $f(\cdot, \mathbf{w})$ on task $\mathcal{T}_t$.*
8:    Set $s = 0$ and $\mathbf{w}_{t,0} = \tilde{\mathbf{w}}_{t-1}$;
9:    **while** not converged **do**
10:      Sample a batch $\{\mathbf{x}, \mathbf{y}\}$ from $\{X_t, Y_t\}$.
11:      Compute $f(\mathbf{x}, \mathbf{w}_{t,s})$, then get candidate parameter update $\mathbf{g}_{t,s} = \{g_{t,s}^1, \dots, g_{t,s}^L\}$ by Adam.
12:      **if** $t = 1$ **then**
13:        $\Delta w_{t,s}^l = g_{t,s}^l, l = 1, \dots, L$
14:      **else**
15:        $\Delta w_{t,s}^l = U_2^l(U_2^l)^\top g_{t,s}^l, l = 1, \dots, L$
16:      **end if**
17:      $w_{t,s+1}^l = w_{t,s}^l - \alpha\Delta w_{t,s}^l, l = 1, \dots, L$
18:      $s = s + 1$
19:    **end while**
20: **end for**

As a summary, for a novel task $\mathcal{T}_t$, our continual learning algorithm (shown in Fig. 2) projects parameter update $\mathbf{g}_{t,s} = \{g_{t,s}^l\}_{l=1}^L$ at s-th training step generated by Adam into the approximate null space layer by layer, and get the parameter update $\Delta\mathbf{w}_{t,s} = \{\Delta w_{t,s}^l\}_{l=1}^L$ following Eqn. (6). We prove that $\langle\Delta\mathbf{w}_{t,s}, \mathbf{g}_{t,s}\rangle \ge 0$ always holds, which can be found in the supplemental material. To guarantee that the network can be trained on task $\mathcal{T}_t$ using the above parameter updating rule, it is supposed that $\langle\Delta\mathbf{w}_{t,s}, \mathbf{g}_{t,s}\rangle > 0$ as discussed in Condition 2. We empirically found that it holds in our experiments and our algorithm can succeed in decreasing training losses on sequential training tasks.

Our Adam-NSCL is summarized in Algorithm 1. The training process is to loop over incoming tasks, and task $\mathcal{T}_t$ $(t > 1)$ is trained by Adam with gradients projected into the null space of accumulated covariance (line 15 in Algorithm 1). The null space is obtained by Algorithm 2 after learning task $\mathcal{T}_{t-1}$. In Algorithm 2, we first feed all training data of task $\mathcal{T}_{t-1}$ to accumulate covariance in lines 3-8, and update the uncentered covariance in line 10. Then we compute the approximate null space in lines 12-14 by SVD.

The hyperparameter a controls the balance of stability and plasticity. Larger a suggests that we use larger approx-

Algorithm 2 Updating the uncentered covariance incrementally and computing the null space.

Inputs: Dateset $\{X_{t-1}, Y_{t-1}\}$ of size n_{t-1} for task $\mathcal{T}_{t-1}$, network $f(\cdot, \tilde{\mathbf{w}}_{t-1})$, $\{\bar{\mathcal{X}}_{t-2}^l\}_{l=1}^L$, hyperparameter $a > 0$, number of seen data $\bar{n}_{t-2}$.
Output: U_2^l, $\{\bar{\mathcal{X}}_{t-1}^l\}_{l=1}^L$, $\bar{n}_{t-1}$

1: *# Compute the uncentered covariance on task $\mathcal{T}_{t-1}$*
2: Initialize uncentered covariance matrices $\mathcal{X}_{t-1}^l$ on task $\mathcal{T}_{t-1}$ as 0 for $l = 1, \ldots, L$.
3: **for** batch $\{\mathbf{x}, \mathbf{y}\}$ from $\{X_{t-1}, Y_{t-1}\}$ **do**
4:     Get the input feature $\mathbf{x}^l$ at the l-th layer ($l = 1, \ldots, L$) by forward propagating $\mathbf{x}$ on $f(\cdot, \tilde{\mathbf{w}}_{t-1})$.
5:     $\mathcal{X}_{t-1}^l = \mathcal{X}_{t-1}^l + (\mathbf{x}^l)^\top \mathbf{x}^l$ for $l = 1, \ldots, L$.
6: **end for**
7: $\mathcal{X}_{t-1}^l = \frac{1}{n_{t-1}} \mathcal{X}_{t-1}^l$ for $l = 1, \ldots, L$.
8: $\bar{n}_{t-1} = \bar{n}_{t-2} + n_{t-1}$.
9: *# Update the uncentered covariance $\bar{\mathcal{X}}_{t-2}^l$.*
10: $\bar{\mathcal{X}}_{t-1}^l = \frac{\bar{n}_{t-2}}{\bar{n}_{t-1}} \bar{\mathcal{X}}_{t-2}^l + \frac{n_{t-1}}{\bar{n}_{t-1}} \mathcal{X}_{t-1}^l$, $l = 1, \ldots, L$.
11: *# Compute the approximate null space for each layer*
12: $U^l, \Lambda^l, (U^l)^\top = \text{SVD}(\bar{\mathcal{X}}_{t-1}^l)$.
13: Get Λ_2^l with diagonal singular values $\lambda \in \{\lambda | \lambda < a\lambda_{\min}^l\}$ where $\lambda_{\min}^l$ is the smallest singular value.
14: Get singular vectors U_2^l that correspond to Λ_2^l.

---

imate null space U_2^l covering more small singular values, then the null space assumption in Eqn. (3) hardly holds, reducing the stability in continual learning. On the other hand, larger a leads to larger approximate null space, enabling us to update network parameters in a larger null space, increasing the plasticity to learn knowledge on current task.

5. Experiments

We apply Adam-NSCL algorithm to different sequential tasks for continual learning[1]. After introducing experimental setting, we show the results compared with SOTA methods, following which we empirically analyze our algorithm.

5.1. Experimental setting

We first describe the experimental settings on datasets, implementation details, metrics and compared methods.

Datasets. We evaluate on continual learning datasets, including 10-split-CIFAR-100, 20-split-CIFAR-100 and 25-split-TinyImageNet. 10-split-CIFAR-100 and 20-split-CIFAR-100 are constructed by splitting CIFAR-100 [22] into 10 and 20 tasks, and the classes in different tasks are disjoint. 25-split-TinyImageNet is constructed by splitting TinyImageNet [49] containing 64×64 RGB images into 25 tasks, which is a harder setting due to longer sequence.

Implementation details. Adam-NSCL is implemented using PyTorch [39]. We take ResNet-18 [13] as the back-

---
[1]https://github.com/ShipengWang/Adam-NSCL

bone network in our experiments. All tasks share the same backbone network but each task has its own classifier. The classifier will be fixed after the model is trained on the corresponding task. For batch normalization layer, we regularize its parameters using EWC [21]. The learning rate starts from 5×10^{-5} and decays at epoch 30, 60 with a multiplier 0.5 (80 epochs in total). The batch size is set to 32 for 10-split-CIFAR-100 and 16 for the other two datasets. The regularization coefficient of EWC for penalizing parameters at batch normalization is set to 100. At each linear layer, to approximate null space of uncentered covariance, we set $a = 30$ for 20-split-CIFAR-100 while $a = 10$ for the other two datasets. We also study the effect of a in Sec. 5.3.

Compared methods. We compare our method with various continual learning methods including *EWC* [21], *MAS* [3], *MUC-MAS* [28], *SI* [37], *LwF* [26], *GD-WILD* [23], *GEM*, [30], *A-GEM* [7], *MEGA* [12], InstAParam [8] and *OWM* [53]. For a fair comparison, the backbone networks employed in these methods are all ResNet-18. *EWC*, *MAS*, *MUC-MAS* and *SI* regularize the changes of parameters across tasks, where each parameter is associated with a weight of importance. *LwF* and *GD-WILD* are based on knowledge distillation using different datasets for distillation to preserve learned knowledge on previous tasks. GEM, A-GEM, *MEGA* and OWM focus on designing network training algorithm to overcome forgetting. InstAParam is based on architecture-based strategy. Among these methods, EWC, MAS, MUC-MAS and SI need to store the importance weight in memory, GD-WILD, GEM A-GEM and MEGA need to store data of previous tasks, and OWM needs to store the projection matrix which is incrementally computed with an approximate inversion of matrix.

Evaluation metrics. We employ the evaluation metrics proposed in [30], including backward transfer (BWT) and average accuracy (ACC). BWT is the average drop of accuracy of the network for test on previous tasks after learning current task. Negative value of BWT indicates the degree of forgetting in continual learning. ACC is the average accuracy of the network on test datasets of all seen tasks. With similar ACC, the one having larger BWT is better.

5.2. Experimental results

We next compare different continual learning algorithms. The details on the comparative results on three datasets are presented as follows.

10-split-CIFAR-100. The comparative results on 10-split-CIFAR-100 are illustrated in Tab. 1, where the proposed Adam-NSCL achieves the highest ACC 73.77% with competitive BWT -1.6%. The BWT values of GEM and A-GEM are better than Adam-NSCL, however, their ACC values are 49.48% and 49.57%, significantly lower than ours. EWC, LwF and GD-WILD achieve marginally worse ACC

Method	10-split-CIFAR-100	
	ACC (%)	BWT(%)
EWC [21]	70.77	-2.83
MAS [3]	66.93	-4.03
MUC-MAS [28]	63.73	-3.38
SI [37]	60.57	-5.17
LwF [26]	70.70	-6.27
InstAParam [8]	47.84	-11.92
*GD-WILD [23]	71.27	-18.24
*GEM [30]	49.48	2.77
*A-GEM [7]	49.57	-1.13
* MEGA [12]	54.17	-2.19
OWM [53]	68.89	-1.88
Adam-NSCL	**73.77**	-1.6

Table 1. Comparisons of ACC and BWT for ResNet-18 sequentially trained on 10-split-CIFAR-100 using different methods. Methods required to store previous data are denoted by *.

Method	20-split-CIFAR-100	
	ACC (%)	BWT(%)
EWC [21]	71.66	-3.72
MAS [3]	63.84	-6.29
MUC-MAS [28]	67.22	-5.72
SI [37]	59.76	-8.62
LwF [26]	74.38	-9.11
InstAParam [8]	51.04	-4.92
*GD-WILD [23]	**77.16**	-14.85
*GEM [30]	68.89	-1.2
*A-GEM [7]	61.91	-6.88
*MEGA [12]	64.98	-5.13
OWM [53]	68.47	-3.37
Adam-NSCL	**75.95**	-3.66

Table 2. Comparisons of ACC and BWT for ResNet-18 sequentially trained on 20-split-CIFAR-100 using different methods.

Method	25-split-TinyImageNet	
	ACC (%)	BWT(%)
EWC [21]	52.33	-6.17
MAS [3]	47.96	-7.04
MUC-MAS [28]	41.18	-4.03
SI [37]	45.27	-4.45
LwF [26]	56.57	-11.19
InstAParam [8]	34.64	-10.05
*GD-WILD [23]	42.74	-34.58
*A-GEM [7]	53.32	-7.68
*MEGA [12]	57.12	-5.90
OWM [53]	49.98	-3.64
Adam-NSCL	**58.28**	-6.05

Table 3. Performance comparisons for ResNet-18 sequentially trained on 25-split-TinyImageNet using different methods.

compared with Adam-NSCL, but the BWT values of LwF and GD-WILD are much lower. Both ACC and BWT values of MAS, MUC-MAS and SI are much lower than ours. OWM has comparable BWT with our Adam-NSCL, but the ACC of OWM is 4.88% lower than ours. Overall, our Adam-NSCL is the most preferable method among all these compared methods for continual learning.

To justify the rationality of approximate null space, we show the curves of singular values in descending order and proportion values of R defined in Eqn. (7) for the 2-th, 7-th, 12-th and 17-th layers of network in Fig. 4 on sequential tasks. As the results indicate, all proportion values of U_2^l
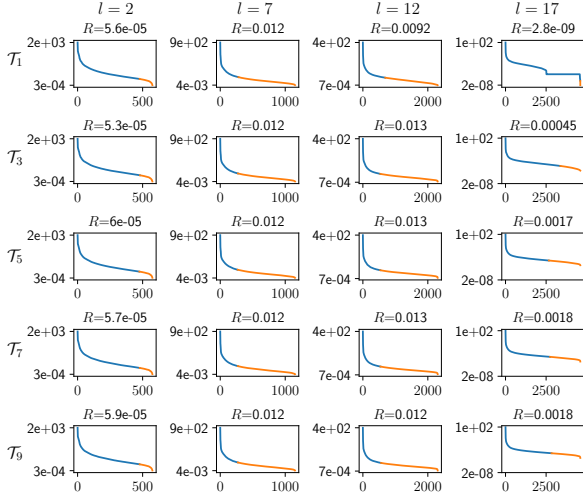

Figure 4. The illustration of proportion values of R and singular values of uncentered covariance matrix at 2-th, 7-th, 12-th and 17-th linear layers of network trained on 10-split-CIFAR-100. Orange curves denote the singular values smaller than $10\lambda_{\min}^l$.

are smaller than 0.05, indicating that it is reasonable to take the range space of insignificant components U_2^l as the approximate null space at the l-th layer ($l = 1, \ldots, L$).

20-split-CIFAR-100. The comparisons on 20-split-CIFAR-100 dataset are shown in Tab. 2. Our method achieves the second best ACC 75.95%. Though GD-WILD achieves 1.21% higher ACC than ours, the BWT of GD-WILD is 11.19% lower than that of our Adam-NSCL. Furthermore, GD-WILD requires to save data of previous tasks and a large amount of external data. EWC, GEM and OWM achieve 4.29%, 7.06% and 7.48% lower ACC values compared with our method. LwF has marginally lower ACC than ours, but its BWT value is significantly worse than ours. Other methods including MAS, MUC-MAS, SI and A-GEM fail to achieve comparable results as ours. Therefore, our Adam-NSCL outperforms the other compared methods for continual learning.
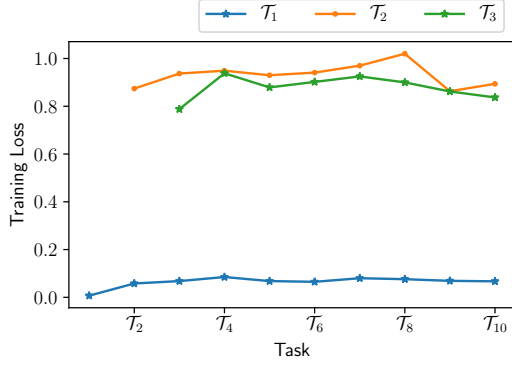
25-split-TinyImageNet. As shown in Tab. 3, on 25-

Figure 5. The curves of training losses of network on tasks $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ when the network is trained on sequential tasks.

split-TinyImageNet dataset, the proposed Adam-NSCL outperforms the other compared methods with the best ACC and competitive BWT values. Specifically, Adam-NSCL achieves the best ACC 58.28% with comparable BWT -6.05%. Though the BWT of Adam-NSCL is marginally lower than MUC-MAS, SI and OWM, these compared methods achieve 16.68%, 13.01% and 8.3% lower ACC than ours. LwF achieves the second best ACC, but with much inferior BWT compared with our Adam-NSCL. With marginally lower BWT, EWC and MAS achieve 5.95% and 10.32% lower ACC than Adam-NSCL.

We now discuss the difference between Adam-NSCL and OWM [53]. The main difference of Adam-NSCL and OWM is the way to find the null space as discussed in Sec. 2. Computing the projection matrix in OWM relies on the approximate generalized inversion of feature matrix, and the approximate error can be accumulated when incrementally update the projection matrix. While in Adam-NSCL, the null space is specified by the uncentered feature covariance which can be incrementally computed without approximate error. Additionally, Adam-NSCL consistently performs better than OWM on 10-split-CIFAR-100 and 20-split-CIFAR-100, as shown in Tabs. 1 and 2, where Adam-NSCL achieves 4.88% and 7.48% larger ACC and similar BWT in comparison with OWM respectively. On 25-split-TinyImageNet, Adam-NSCL has significantly better ACC with comparable BWT than OWM, as shown in Tab. 3.

5.3. Model analysis

Evolution of training loss. To justify the proposed Adam-NSCL indeed guarantees the stability of network training on sequential tasks, we show the curves of training losses on the tasks $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ after learning new tasks in Fig. 5 on 10-split-CIFAR-100. According to Fig. 5, the training losses of the network on previous tasks are retained after learning new tasks, verifying that Adam-NSCL, with Condition 1 as basis, guarantees the stability of network.
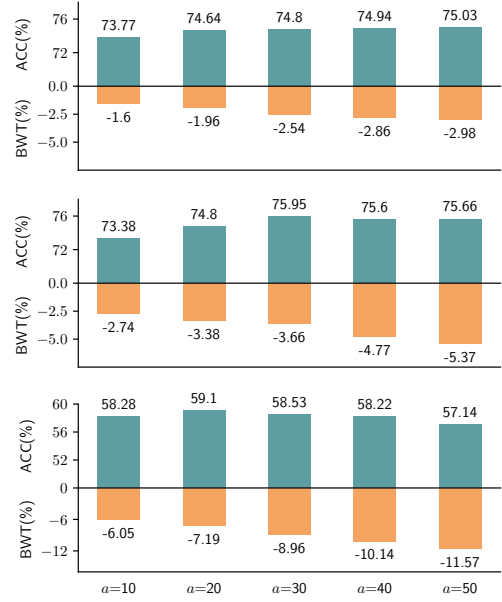

Figure 6. Stability and plasticity analysis. Top: 10-split-CIFAR-100. Middle: 20-split-CIFAR-100. Bottom: 25-split-TinyImageNet.

Stability and plasticity analysis. To study the balance of stability and plasticity, which is controlled by a, we compare the performance of our Adam-NSCL by varying $a = 10, 20, 30, 40, 50$. According to Fig. 6, BWT becomes worse when a is larger, suggesting that the network forgets more learned knowledge of previous tasks with larger a. Since ACC is affected by both of stability and plasticity, it increases first and then decreases with the increase of a at the middle and bottom sub-figures of Fig. 6.

6. Conclusion

In this paper, we address the *plasticity-stability dilemma* for continual learning where the datasets of previous tasks are inaccessible. We propose two theoretical conditions to guarantee stability and plasticity for network parameter update when training networks on sequential tasks. Then we design a novel continual learning algorithm Adam-NSCL, which is based on Adam. The candidate parameter update generated by Adam is projected into the approximate null space of uncentered feature covariance matrix of previous tasks. Extensive experiments show that the proposed algorithm outperforms the compared methods for continual learning. In the future, we consider further improving the approximation of null space and conducting theoretical analysis for our algorithm.

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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3931–3940, 2020. 1, 2

[2] Tameem Adel, Han Zhao, and Richard E Turner. Continual learning with adaptive weights (claw). In *Int. Conf. Learn. Represent.*, 2019. 2

[3] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Eur. Conf. Comput. Vis.*, pages 139–154, 2018. 2, 6, 7

[4] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Adv. Neural Inform. Process. Syst.*, pages 11816–11825, 2019. 1, 2

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Int. Conf. Learn. Represent.*, 2015. 1

[6] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Eur. Conf. Comput. Vis.*, pages 532–547, 2018. 2

[7] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *Int. Conf. Learn. Represent.*, 2018. 1, 2, 6, 7

[8] Hung-Jen Chen, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Mitigating forgetting in online continual learning via instance-aware parameterization. In *Adv. Neural Inform. Process. Syst.*, pages 17466–17477, 2020. 6, 7

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[10] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5138–5146, 2019. 2

[11] Robert M French. Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.*, 3(4):128–135, 1999. 1

[12] Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory-based lifelong learning. In *Adv. Neural Inform. Process. Syst.*, pages 1023–1035, 2020. 6, 7

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Eur. Conf. Comput. Vis.*, pages 630–645, 2016. 6

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *Adv. Neural Inform. Process. Syst. Worksh.*, 2015. 1, 2

[15] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In *Adv. Neural Inform. Process. Syst. Worksh.*, 2018. 1

[16] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *Int. Conf. Learn. Represent.*, 2018. 2

[17] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In *Adv. Neural Inform. Process. Syst.*, pages 13669–13679, 2019. 2

[18] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philos. Trans. R. Soc. A*, 374(2065), 2016. 5

[19] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *Int. Conf. Learn. Represent.*, 2018. 2

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Int. Conf. Learn. Represent.*, 2015. 2

[21] 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. *Proc. Natl. Acad. Sci. USA*, 114(13):3521–3526, 2017. 1, 2, 6, 7

[22] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 6

[23] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Int. Conf. Comput. Vis.*, pages 312–321, 2019. 2, 6, 7

[24] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Adv. Neural Inform. Process. Syst.*, pages 4652–4662, 2017. 2

[25] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Int. Conf. Mach. Learn.*, pages 3925–3934, 2019. 1, 2

[26] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, 2017. 1, 2, 6, 7

[27] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D. Bagdanov, Shangling Jui, and Van De Weijer Joost. Generative feature replay for class-incremental learning. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2020. 2

[28] 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 *Eur. Conf. Comput. Vis.*, 2020. 6, 7

[29] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12245–12254, 2020. 2

[30] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Adv. Neural Inform. Process. Syst.*, pages 6467–6476, 2017. 1, 2, 6, 7

[31] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Eur. Conf. Comput. Vis.*, pages 67–82, 2018. 2

[32] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychol. Learn. Motiv.* Elsevier, 1989. 1

[33] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Front. Psychol.*, 4:504, 2013. 1

[34] Carl D Meyer. *Matrix analysis and applied linear algebra*, volume 71. Siam, 2000. 5

[35] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *Int. Conf. Learn. Represent.*, 2018. 2

[36] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11321–11329, 2019. 1

[37] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11321–11329, 2019. 2, 6, 7

[38] Inyoung Paik, Sangjun Oh, Tae-Yeong Kwak, and Injung Kim. Overcoming catastrophic forgetting by neuron-level plasticity control. In *AAAI*, pages 5339–5346, 2020. 2

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Adv. Neural Inform. Process. Syst.*, pages 8024–8035, 2019. 6

[40] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Eur. Conf. Comput. Vis.*, 2020. 2

[41] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In *Adv. Neural Inform. Process. Syst.*, pages 12669–12679, 2019. 2

[42] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychol. Rev.*, 97(2):285, 1990. 1

[43] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2001–2010, 2017. 2

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015. 5

[45] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *Int. Conf. Mach. Learn.*, pages 4548–4557, 2018. 2

[46] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Adv. Neural Inform. Process. Syst.*, pages 2990–2999, 2017. 2

[47] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 1

[48] Concetto Spampinato, Simone Palazzo, Isaak Kavasidis, Daniela Giordano, Nasim Souly, and Mubarak Shah. Deep learning human mind for automated visual classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6809–6817, 2017. 1

[49] Jiayu Wu, Qixiang Zhang, and Guoxi Xu. Tiny imagenet challenge. Technical report, Stanford University, 2017. 6

[50] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 374–382, 2019. 2

[51] Yang Yan, Jian Sun, Huibin Li, and Zongben Xu. Deep admm-net for compressive sensing mri. In *Adv. Neural Inform. Process. Syst.*, pages 10–18, 2016. 1

[52] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *Int. Conf. Learn. Represent.*, 2018. 2

[53] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nat. Mach. Intell.*, 1(8):364–372, 2019. 1, 2, 6, 7, 8

[54] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Int. Conf. Mach. Learn.*, pages 3987–3995, 2017. 2

[55] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong gan: Continual learning for conditional image generation. In *Int. Conf. Comput. Vis.*, pages 2759–2768, 2019. 2

[56] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 13208–13217, 2020. 2