Balancing Stability and Plasticity through Advanced Null Space in Continual Learning

Yajing Kong[1], Liu Liu[1], Zhen Wang[1], and Dacheng Tao[1,2]

[1] The University of Sydney, Darlington, NSW 2008, Australia
[2] JD Explore Academy, Beijing, China
{ykon9947, liuliu1, zwan4121}@sydney.edu.au, dacheng.tao@gmail.com

Abstract. Continual learning is a learning paradigm that learns tasks sequentially with resources constraints, in which the key challenge is stability-plasticity dilemma, i.e., it is uneasy to simultaneously have the stability to prevent catastrophic forgetting of old tasks and the plasticity to learn new tasks well. In this paper, we propose a new continual learning approach, Advanced Null Space (AdNS), to balance the stability and plasticity without storing any old data of previous tasks. Specifically, to obtain better stability, AdNS makes use of low-rank approximation to obtain a novel null space and projects the gradient onto the null space to prevent the interference on the past tasks. To control the generation of the null space, we introduce a non-uniform constraint strength to further reduce forgetting. Furthermore, we present a simple but effective method, intra-task distillation, to improve the performance of the current task. Finally, we theoretically find that null space plays a key role in plasticity and stability, respectively. Experimental results show that the proposed method can achieve better performance compared to state-of-the-art continual learning approaches.

Keywords: continual learning · catastrophic forgetting · null space

1 Introduction

Humans have excellent abilities in learning new knowledge while maintaining the knowledge learned from past experience through their lifelong time. Continual learning aims at developing algorithms for neural networks with the same capabilities from a stream of data [34, 48]. However, although deep neural networks have made impressive achievements across various domains, they easily suffer performance degradation on the previous tasks when applied to sequential tasks without any access to historical data. The problem, referred to as catastrophic forgetting, is a key challenge in continual learning [16, 21, 30, 34, 38, 48].

This problem is closely related to the stability-plasticity dilemma [31, 32]. Specifically, when learning in a sequential fashion, the network is required to have the plasticity to integrate new knowledge well and the stability to prevent the forgetting of previous tasks. However, the stability-plasticity dilemma indicates that it is hard to simultaneously have high plasticity and high stability. To relieve the

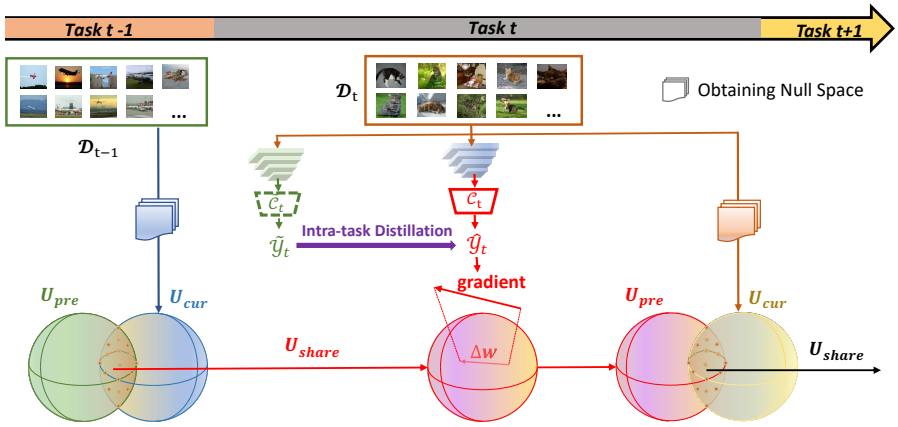

Fig. 1. The pipeline of the proposed method. Left: At the task $\mathcal{T}_{t-1}$, we obtain the shared low-rank null space $\mathbf{U}_{\text{share}}$ based on $\mathbf{U}_{\text{pre}}$ and $\mathbf{U}_{\text{cur}}$. Middle: We project the gradient at each layer onto the shared low-rank null space. Right: $\mathbf{U}_{\text{share}}$ is used for the next task $\mathcal{T}_{t+1}$ as $\mathbf{U}_{\text{pre}}$. Note at the task $\mathcal{T}_t$ (t>1), we conduct Intra-task Distillation between $\tilde{\mathcal{Y}}_t$ and $\hat{\mathcal{Y}}_t$.

dilemma, a growing body of continual learning methods are introduced. These method can be roughly divided into four categories: architecture-based methods expand the network or allocate new neurons for new tasks [13,28,42,60]; replayed-based methods interleave old data with current data by storing historical data in a buffer or generating virtual old data [4,8,12,39,40]; regularization-based methods penalize the update of important parameters of previous tasks [1,17,21,58]; algorithm-based methods modify the update rule of parameters to prevent the interference across tasks [7,26,43,47,49].

For algorithm-based methods, one of the classical approaches is to project the gradient onto the approximation null space of all previous tasks, in which the gradient has little interference on the performance of previous tasks [43,49,57]. However, despite the impressive performance achieved by these methods, there are still some challenges that impede the null space methods to achieve satisfactory stability-plasticity trade-off. First, the null space methods are based on the finding that the model modifies the parameters in the exact null space of previous tasks. However, due to the approximation of null space, the model will occur in interference on the previous tasks. Moreover, the interference would affect the subsequent approximation of null space of past tasks, thus leading to more information deficiency of null space of previous tasks. Therefore, the stability of the model will be unsatisfactory. Second, the model update is based on the gradient projection on the null space of previous task, preventing the model from learning the current task well, i.e., resulting in worse plasticity.

To address the above challenges, we propose a new algorithm-based continual learning approach, Advanced Null Space (AdNS), to achieve a good balance between stability and plasticity. Specifically, to alleviate the impact of approx-

imation on the stability, AdNS makes use of the low-rank approximation to extract the shared null space between the previous null space and the current candidate null space. Unlike existing works that only focus on the current candidate null space [43, 57], AdNS projects the gradient onto the shared null space, which contains the core spaces between null spaces, and thus could mitigate the information deficiency of the previous null spaces and reduce forgetting. Moreover, we present a constraint to control the approximation of null space and propose non-uniform constraint strength, which monotonically decreases with the number of tasks increasing, to further relieve the forgetting. What's more, to improve the performance of the current task, we leverage a simple method, intra-task distillation, to self-distill the knowledge of the current task. The procedure of the proposed method is shown in Fig. 1.

Finally, although various algorithms about null space have been proposed, few efforts were spent on the theoretical foundations. Therefore, in this paper, we theoretically analyze the impact of null space and present two theorems to prove that null space plays a key role in stability and plasticity. The theoretical finding indicates the inherent properties of the stability-plasticity dilemma, in which it is hard to have high plasticity and high stability simultaneously. To summarize, our contributions are threefolds:

- To address the stability-plasticity dilemma, we propose a new algorithm-based continual learning approach, AdNS, which projects the gradient into the shared null space under non-uniform constraint strength to reduce forgetting, and uses intra-task distillation to improve the tasks' performance.
- We present two theorems from the perspective of stability and plasticity, which show that the null space plays a key role in balancing the stability-plasticity dilemma. Specifically, the larger the dimension of the null space, the better the plasticity, the worse the stability.
- We validate the proposed method in several benchmarks, and the empirical results show that the proposed method can outperform related state-of-the-art continual learning methods.

2 Related Work

Algorithms-based methods design the update rule to decrease the interference of parameter update on the performance of old tasks [6,7,24,26,43,47,49,54]. For example, GEM [26] and A-GEM [7] used the historical samples to compute the gradients of old tasks and proposed inequality constraints of gradients to avoid the increase of losses of past tasks. *Arslan et al.* [6] manually divided a random orthonormal space into several subspaces and allocated these subspaces one-to-one to each task. However, these methods require storing data of previous tasks. In contrast, GPM [43] stored the bases of core gradient space and modified the parameters in the direction orthogonal to the core space. OWM [57] modified the parameters in the direction orthogonal to the input space of previous tasks. Adam-NSCL stored uncentered feature covariance and used it to compute the null space [49]. Our work is closely related to Adam-NSCL. However, unlike

Adam-NSCL that only considers the current candidate null space, we project the gradient onto the shared null space between null spaces, which could relieve the information deficiency of null space of previous tasks, resulting in less forgetting. Moreover, our method does not rely on any data of previous tasks and only needs to update the null space in one shot manner at the end of each task.
Regularization-based methods can be divided into two classes: one is to distill knowledge from the previous model which is trained on the previous tasks [14, 15, 23, 36, 52, 59]; another is to explicitly use a regularizer to penalize the update of important parameters of previous tasks, preventing the model from deviating too much from the previous one to avoid forgetting [19, 21, 33, 35, 58]. For example, for the first class, LwF [23] distills the knowledge by using the previous model outputs as soft labels and penalizing the distillation term between the current and recorded output. For the latter one, EWC [17] used the diagonal of the Fisher information matrix as the importance.
Other Approaches. Architecture-based methods allocate different parameters or add new parameters for the new task, while sharing parameters across tasks to reduce the interference of previous tasks [3, 22, 27–29, 41, 42, 44, 53, 56, 60]. However, such methods may lead to a cumbersome and complex network if new tasks continually arrive. Replayed-based methods leverage episodic memory to store representative history data or generate virtual data via a generative model, and replay these samples with current data [2, 7, 8, 12, 26, 37, 39, 40, 45, 50, 51]. However, replayed-based methods may bring some problems since storing old data will result in data imbalance, and the generative model would be large and expensive if it synthesizes the historical data reasonably.

3 Preliminaries

3.1 Settings and Notations

In this part, we present the settings and notations. We consider a sequence of tasks $\mathcal{T}_t$, $t \in \{1, ..., T\}$, where T is the total number of tasks. Let $\mathcal{D}_t = \{\mathcal{X}_t, \mathcal{Y}_t\}$ be the dataset of task $\mathcal{T}_t$, where $\mathcal{X}_t$ and $\mathcal{Y}_t$ are the corresponding inputs set and label set. In continual learning, the model is trained on these datasets sequentially. Let $\mathbf{w} = \{w^1, ..., w^L\}$ be the parameters of L-layer neural network, where w^l is the parameter vector of the l-th layer, $l \in \{1, ..., L\}$. Let $\hat{L}_t(\mathbf{w})$ be the empirical loss of task $\mathcal{T}_t$ with parameters $\mathbf{w}$. Define $\tilde{\mathbf{w}}_t$ as the convergence parameters after the model has been trained on the task $\mathcal{T}_t$. Rank$(\cdot)$ denotes the rank of a matrix, and $[\cdot, \cdot]$ denotes the concatenation of vectors or matrices. $\|\cdot\|_F$ denotes Frobenius norm, $\|\cdot\|_1$ denotes L_1 norm, and $\|\cdot\|_2$ denotes L_2 norm.

3.2 Null Space

Let Δw^l be the parameter update for l-th layer for the current step. If Δw^l lies in the null space of previous tasks at each training step for the l-th layer, $l \in \{1, ..., L\}$, then the stability can be guaranteed, which is illustrated by the following lemma.

Lemma 1. *[49] Define $X_{p,p}^l$ as the input feature of l-th layer when the network is fed with data $\mathcal{X}_p$ after training on the task $\mathcal{T}_p$. Let $\mathcal{N}(\tilde{\mathbf{w}}_t; \mathcal{X}_p)$ be the output of the L-layer network with parameters $\tilde{\mathbf{w}}_t$ when the network is fed with data $\mathcal{X}_p$. If at each training step of task $\mathcal{T}_t$, Δw^l lies in the null space of $X_{t-1}^l = [X_{1,1}^l, ..., X_{t-1,t-1}^l]$, i.e.,*

$$X_{t-1}^l \Delta w^l = \mathbf{0}, \quad l = 1, ..., L, \tag{1}$$

then we have $\mathcal{N}(\tilde{\mathbf{w}}_t; \mathcal{X}_p) = \mathcal{N}(\tilde{\mathbf{w}}_p; \mathcal{X}_p)$ for all $p \in \{1, ..., t-1\}$.

According to Lemma 1, if the parameters are modified in the null space of X_{t-1}^l, then the training loss of previous tasks will be retained and the forgetting can be avoid. Nevertheless, it is unrealistic to expect the existence of the null space, thus previous works [43, 49, 57] use the approximation null space instead. However, the approximation will cause that Eq.(1) no longer holds and result in the occurrence of interference on previous tasks. Moreover, the interference would affect the subsequent approximation of null space of past tasks, leading to more performance degradation on past tasks, i.e., catastrophic forgetting.

4 Methodology

In this section, we propose a new continual learning method, Advanced Null Space (AdNS), involving shared low-rank null space (Section 4.1), non-uniform constraint strength (Section 4.2), and intra-task distillation (Section 4.3), to balance the stability and plasticity. The procedure of AdNS is shown in Fig. 1 and the algorithm is shown in Algorithm 1.

4.1 Shared Low-Rank Null Space

In this part, we introduce a noval null space, the shared low-rank null space, which extracts the shared null spaces between the previous null space and the current candidate null space based on low-rank approximation.

When training on the current task $\mathcal{T}_t$ (t>1), the gradient of l-th layer is projected onto the null space of previous tasks, which is spanned by the columns of $\mathbf{U}_{\text{pre}}^l$[3], $l \in \{1, ..., L\}$ and $\mathcal{U}_{\text{pre}} = \{\mathbf{U}_{\text{pre}}^1, ..., \mathbf{U}_{\text{pre}}^L\}$ is the set containing the previous null spaces of L layers. After training the current task, the current candidate null space $\mathcal{U}_{\text{cur}} = \{\mathbf{U}_{\text{cur}}^1, ..., \mathbf{U}_{\text{cur}}^L\}$ could be obtained based on the input features of tasks seen so far. Specifically, we use the uncentered feature covariance, i.e., $\tilde{X}_t^l = (X_t^l)^\top X_t^l$, to compute the current candidate null space [49]. Such process has moderate memory consumption since the dimension of $\tilde{X}_t^l$ is irrelevant to the size of data. It can be easily proved that the null space of $\tilde{X}_t^l$ is equal to X_t^l. We update the input features of each layer by $\tilde{X}_t^l \leftarrow \tilde{X}_{t-1}^l + (X_{t,t}^l)^\top X_{t,t}^l$, where $X_{t,t}^l$ is the input feature of l-th layer when the network

[3] We use the matrix whose columns are consisted of the orthonormal basis of the null space to represent null space.

Algorithm 1 Advanced Null Space (AdNS)

Input: Network $\mathcal{N}$ with parameters $\mathbf{w}$
Output: Target network $\mathcal{N}$
  for $t = 1, 2, .., T$ **do**
    if $t = 1$ **then**
      while not converged **do**
        Update the gradient according to $\hat{L}_1(\mathbf{w})$
      end while
      $\tilde{X}_1^l \leftarrow (X_{1,1}^l)^\top X_{1,1}^l$
      $\mathbf{U}^l \leftarrow$ Null space of $\tilde{X}_1^l$ based on (3)
      $\mathbf{U}_{\text{pre}}^l \leftarrow \mathbf{U}^l$
      Break ▷ Return to the next task
    end if
    $\tilde{\mathcal{Y}}_t \leftarrow \mathcal{N}(\mathbf{w}; \mathcal{X}_t)$ after updating the classifier $\tilde{\mathcal{C}}_t$
    while not converged **do**
      Update the gradient in the null space at each layer based on $\mathcal{U} = \{\mathbf{U}^1, ..., \mathbf{U}^L\}$ according to (5)
    end while
    $\tilde{X}_t^l \leftarrow \tilde{X}_{t-1}^l + (X_{t,t}^l)^\top X_{t,t}^l$
    $\mathbf{U}_{\text{cur}}^l \leftarrow$ Null space of $\tilde{X}_t^l$ based on (3)
    $\tilde{\mathbf{U}}^l = [\mathbf{U}_{\text{pre}}^l, \mathbf{U}_{\text{cur}}^l]$, $l = 1, ..., L$
    $\mathbf{U}^l \leftarrow$ The shared low-rank null space obtained in (P2)
    $\mathbf{U}_{\text{pre}}^l \leftarrow \mathbf{U}^l$
  end for
  return $\mathcal{N}$

is fed with data $\mathcal{X}_t$ after training on the task $\mathcal{T}_t$, and then obtain the current candidate null space at the end of task.

To alleviate the impact of approximation on the stability, rather than using $\mathcal{U}_{\text{cur}}$, we extract the shared null space between $\mathcal{U}_{\text{pre}}$ and $\mathcal{U}_{\text{cur}}$ by solving the problem of low-rank approximation for the concatenation matrix $\tilde{\mathbf{U}}^l = [\mathbf{U}_{\text{pre}}^l, \mathbf{U}_{\text{cur}}^l]$:

$$\text{minimize}_{\hat{\mathbf{U}}^l} \quad \|\tilde{\mathbf{U}}^l - \hat{\mathbf{U}}^l\|_F \quad s.t. \quad \text{Rank}(\hat{\mathbf{U}}^l) \leq k_l, \quad l = 1, ..., L, \tag{P2}$$

where k_l is the rank of the shared null space of l-th layer. The optimization problem (P2) has analytic solutions in terms of the singular value decomposition. Because we want to project the gradient onto an orthonormal space, according to the properties of singular value decomposition, the objective matrix $\mathbf{U}^l$, i.e., the shared low-rank null space of l-th layer, can be constructed by the singular vectors of $\hat{\mathbf{U}}^l$ rather than using $\hat{\mathbf{U}}^l$ directly. The implementation details can be found in the Appendix.

The shared low-rank null space, which is the range space of $\mathbf{U}^l$, contains the shared information of the previous null space and the current null space. Thus, projecting the gradient onto the shared low-rank space could relieve catastrophic

forgetting. In formal, we project g^l as the following projection operation:

$$\Delta w^l = \mathbf{U}^l(\mathbf{U}^l)^\top g^l, \quad l = 1, ..., L, \tag{2}$$

and the parameter is updated by $w^l \leftarrow w^l - \eta \Delta w^l$, where η is the learning rate.

Regarding the computing complexity, let the dimension of the feature at l-th layer be d^l and the dimension of $\tilde{\mathbf{U}}^l$ be $\tilde{k}_l$, where $\tilde{k}_l < d^l$. Then the complexity of computing the current candidate null space $\mathcal{O}((d^l)^3)$ is larger than the complexity of the low-rank approximation $\mathcal{O}(d^l(\tilde{k}_l)^2)$. Therefore, the time consumption of our proposed method is comparable to previous works [43, 49]. The comparisons of running time can be reffed to Appendix.

4.2 Non-uniform Constraint Strength

In continual learning, it is essential to balance the importance of previous tasks and the current task to achieve satisfying performance [19, 21, 58]. For example, if the model puts too much weight on the current task, it may suffer significant performance degradation on the prior tasks and vice-versa. Therefore, to make a trade-off between the importance of previous tasks and current task, i.e., stability and plasticity, we rewrite the constraint $X_{t-1}^l \Delta w^l = \mathbf{0}$ for $l \in \{1, ..., L\}$ to $\|X_{t-1}^l \Delta w^l\|_1 \leq \epsilon$, where ϵ is a factor controlling the strength of constraint. Such a constraint is reasonable for the following two reasons: (a) In practice, it is unrealistic to expect that there exists a null space satisfying $X_{t-1}^l \Delta w^l = \mathbf{0}$ for $l \in \{1, ..., L\}$. (b) Although the constraint $X_{t-1}^l \Delta w^l = \mathbf{0}$ could guarantee that the model would not suffer from catastrophic forgetting, it would result in poor performance of the current task because the constraint is too strict for the parameter update of the current task. In contrast, introducing the balance factor ϵ allows us to make a trade-off between stability and plasticity flexibly.

Moreover, with new tasks occurring, the number of previous tasks increases, resulting in a greater impact of previous tasks on the final performance than the current task [19, 21, 58]. Hence, with the growth of observed tasks, it is necessary to pay more attention to previous tasks to relieve the catastrophic forgetting. Therefore, we propose non-uniform constraint strength following a common assumption that the importance of previous tasks is related to the number of tasks seen so far. In particular, the non-uniform constraint strength can be represented as

$$\|X_{t-1}^l \Delta w^l\|_1 \leq \epsilon(t), \quad \text{for} \quad l = 1, ..., L, \tag{3}$$

where $\epsilon(t)$ is a function monotonically decreasing with the number of tasks seen so far. With new tasks continually coming, the constraint strength becomes more restrictive, and thus the model pays more attention to preserving the performance of previous tasks, achieving better stability.

4.3 Intra-task Distillation

To further address the plasticity-stability dilemma, we leverage knowledge distillation of the current task, called intra-task distillation, to improve the performance of the current task. In particular, as shown in Fig. 1, before training the

task $\mathcal{T}_t$, we first freeze the backbone, which is learned on the previous $t-1$ tasks sequentially, and only train the classifier $\tilde{\mathcal{C}}_t$ of the task $\mathcal{T}_t$. Then we store the outputs $\tilde{\mathcal{Y}}_t$ for the current task $\mathcal{T}_t$. The frozen backbone and the classifier absorb the information of prior tasks and the current task, respectively. Therefore, by penalizing the difference between the record outputs $\tilde{\mathcal{Y}}_t$ from the classifier $\tilde{\mathcal{C}}_t$ and the current outputs $\hat{\mathcal{Y}}_t$ from the classifier $\mathcal{C}_t$, intra-task distillation could improve the performance of the current task while preserving the acquired knowledge from previous tasks.

Specifically, we use the modified cross-entropy loss as the distillation loss. When training on the task $\mathcal{T}_t$, the distillation loss can be represented as:

$$\hat{L}_d(\tilde{y}_t, \hat{y}_t) = -\sum_{c=1}^{C_t} \tilde{y}_t'^{(c)} \log \hat{y}_t'^{(c)}, \tag{4}$$

where $\tilde{y}_t'^{(c)} = \frac{\exp(\tilde{y}_t^{(c)}/\tau)}{\sum_i \exp(\tilde{y}_t^{(i)}/\tau)}, \hat{y}_t'^{(c)} = \frac{\exp(\hat{y}_t^{(c)}/\tau)}{\sum_i \exp(\hat{y}_t^{(i)}/\tau)}$, C_t is the number of classes of task $\mathcal{T}_t$, τ is the temperature factor; $\tilde{y}_t^{(c)}$ and $\hat{y}_t^{(c)}$ are the recorded and current outputs of a sample x_t in task $\mathcal{T}_t$, respectively. We set $\tau = 2$ by default.

In summary, when training on the task $\mathcal{T}_t$ (t>1), the optimization problem including (P2) and (2)-(4), which is represented as:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \hat{L}_t(\mathbf{w}) + \beta \hat{L}_d(\tilde{\mathcal{Y}}_t, \hat{\mathcal{Y}}_t), \\ s.t. \quad & \text{Rank}(h(X_{t-1}^l)) \leq k_l, \|X_{t-1}^l \Delta w^l\|_1 \leq \epsilon(t), \quad l = 1, ..., L, \end{aligned} \tag{5}$$

where $\hat{L}_t(\mathbf{w})$ is the cross-entropy loss for the current task and β is a coefficient that balances the importance between the cross-entropy loss and the distillation loss $\hat{L}_d(\tilde{\mathcal{Y}}_t, \hat{\mathcal{Y}}_t)$; $\mathbf{w}$ is the network parameters and $\epsilon(t)$ is the constraint strength which impacts the trade-off between the stability and plasticity; $h(X)$ denotes X is mapped to its approximation null space which satisfies the constraint; k_l is used to control the rank of the matrix of l-th layer.

5 Analysis

In this section, we present two theorems to theoretically prove that the null space plays a key role in the stability-plasticity dilemma. Theorem 1 is in terms of plasticity and Theorem 2 is in terms of stability. The proof can be found in Appendix. Before introducing the two theorems, we first present three assumptions. Comparable assumptions are also made in existing studies on continual learning [55].

Assumption 1. *$\hat{L}_t(\mathbf{w})$ is L_f-smooth, i.e., $\hat{L}_t(\mathbf{w}) \leq \hat{L}_t(\mathbf{v}) + \langle \nabla \hat{L}_t(\mathbf{v}), \mathbf{w} - \mathbf{v} \rangle + \frac{L_f}{2}\|\mathbf{w} - \mathbf{v}\|_2^2$, $t \in \{1, ..., T\}$, for any $\mathbf{v}, \mathbf{w} \in \mathbb{R}^d$.*

Assumption 2. *For each task $\mathcal{T}_t$, $t \in \{1, ..., T\}$, the number of iterations is bounded by an integer S.*

Assumption 3. *$\hat{L}_t$ has the σ^2-uniformly bounded gradient variance, i.e.,*

$$\|\nabla \hat{L}_t(\mathbf{w}; x, y) - \nabla \hat{L}_t(\mathbf{w}; \mathcal{X}_t, \mathcal{Y}_t)\|_2^2 \leq \sigma^2, (x, y) \in \mathcal{D}_t, t \in \{1, ..., T\}.$$

Based on the assumptions, we derive the following two Theorems regarding the plasticity and stability. Specifically, we obtain the upper bound of the loss of the current task and forgetting.

Theorem 1. *(Plasticity) Suppose Assumptions 1, 2, and 3 hold. Let $\mathbf{w}_{t,s}$ be the parameters on task $\mathcal{T}_t$ at the s-th step and η be the learning rate. Let the range of space of $\mathbf{U}^l$ be the null space of previous tasks for l-th layer, then the loss of the current task $\mathcal{T}_t$ is upper bound by*

$$\hat{L}_t(\mathbf{w}_{t,S}) \leq \hat{L}_t(\mathbf{w}_{t,0}) + \frac{\eta}{2} \sum_{s=0}^{S-1} \sum_{l=1}^{L} \|(I - \mathbf{U}^l (\mathbf{U}^l)^\top) g_{t,s}^l\|_2^2 - \frac{\eta}{2} \sum_{s=0}^{S-1} \|\nabla \hat{L}_t(\mathbf{w}_{t,s})\|_2^2 + \frac{S L_f \eta^2 \sigma^2}{2},$$

where $g_{t,s}^l$ is l-th layer gradient of $\hat{L}_t(\mathbf{w}_{t,s})$.

Theorem 2. *(Stability) Suppose Assumptions 1, 2, and 3 hold. Let $\mathbf{w}_{t,s}$ be the parameters on task $\mathcal{T}_t$ at the s-th and η be the learning rate. Let $\hat{L}_{1:t-1}$ be the sum of empirical loss function of previous $t-1$ tasks and $g_{1:t-1,s}^l$ is its gradient of l-th layer at $\mathbf{w}_{t,s}$. Let $g_{t,s}^l$ be the gradient of the current task at $\mathbf{w}_{t,s}$ of l-th layer. Let the range of space of $\mathbf{U}^l$ be the null space of previous tasks for l-th layer, then the forgetting of the previous $t-1$ tasks generated by training on the task $\mathcal{T}_t$ is upper bound by*

$$\begin{aligned} \hat{L}_{1:t-1}(\mathbf{w}_{t,S}) - \hat{L}_{1:t-1}(\mathbf{w}_{t,0}) \leq & \eta \sum_{s=0}^{S-1} \sum_{l=1}^{L} \|\mathbf{U}^l (\mathbf{U}^l)^\top\|_2 \|g_{t,s}^l\|_2 \|g_{1:t-1,s}^l\|_2 \\ & + \frac{L_f}{2} \eta^2 \sum_{s=0}^{S-1} \sum_{l=1}^{L} \|\mathbf{U}^l (\mathbf{U}^l)^\top\|_2^2 \|g_{t,s}^l\|_2^2. \end{aligned}$$

Remark 1. From Theorems 1 and 2 , we could conclude that $\mathbf{U}^l$ plays a key role in the stability and plasticity. According to Theorem 1, if the rank of the null space is larger, then the term $\|(I - \mathbf{U}^l (\mathbf{U}^l)^\top) g_{t,s}^l\|_2^2$ will be smaller and the upper bound of $\hat{L}_t(\mathbf{w}_{t,S})$ will be smaller. Therefore, the model could learn the current task better, indicating better plasticity. However, according to Theorem 2, the larger the rank of the null space, the larger the term $\|\mathbf{U}^l (\mathbf{U}^l)^\top\|_2^2$. Therefore, the upper bound of the forgetting would be larger, resulting in poorer stability. The two theorems indicate the inherent properties of the stability-plasticity dilemma that it is hard to have high plasticity and high stability simultaneously.

6 Experiments

6.1 Experimental Setup

Datasets and Architecture. Following [49], we perform experiments on three continual learning benchmarks: 10-Split CIFAR-100 (10-S-CIFAR100), 20-Split-CIFAR-100 (20-S-CIFAR100), and 25-Spilt TinyImageNet (25-S-TinyImageNet).

Specifically, 10-Split CIFAR-100 and 20-Split CIFAR-100 are constructed by splitting CIFAR100 [18] into 10 and 20 sequential tasks, respectively. Each task contains the same classes without replacement out of the total 100 classes. Similarly, 25-Spilt TinyImageNet is constructed by splitting 200 classes of TinyImageNet [46] into 25 sequential tasks, where each task has 8 classes. We use ResNet-18 [11] as the backbone [4, 6, 10, 11]. All tasks share the same backbone, while each task has its separate classifier.

Baselines. We compare the proposed method against competitive and well-established methods[4], including 5 regularization-based methods using importance measure (EWC [17], MAS [1], MUC-MAS [25], SI [58], and CPR [5]), 2 regularization-based methods using knowledge distillation (LwF [23] and GD-WILD [20]), 1 architecture-based method (InstAParam [9]), and 6 algorithm-based methods (GEM [26], A-GEM [7], MEGA [10], OWM [57], GPM [43], and Adam-NSCL [49]). We also provide a lower bound performance of Vanilla which trains tasks sequentially without any countermeasure to forgetting.

Performance Metrics. To evaluate the performance, we use two standard metrics: a) Average accuracy (ACC) [26,32] is the average test accuracy evaluated on all tasks after learning all tasks sequentially; b) Backward Transfer (BWT) [7,26] is the average performance decrease of the network on previous tasks after new learning. In formal, ACC and BWT are defined as: $\text{ACC} = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}$, $\text{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1}(A_{T,i} - A_{i,i})$, where T is the total number of tasks and $A_{j,i}$ is the accuracy of task $\mathcal{T}_i$ after training on the task $\mathcal{T}_j$ sequentially. The larger the two metrics, the better the model. If the performances of ACC are similar, then the method with a larger value of BWT is better [26].

Implementation Details. When obtaining the null space at the end of each task, we approximate the constraint $\|X_{t-1}^{l}\Delta w^{l}\|_1 \leq \epsilon(t)$ like [49]. Specifically, when computing the current candidate null space, we approximate the current candidate null space with the singular values satisfying $\lambda \in \{\tilde{\lambda}|\tilde{\lambda} \leq \alpha(t)\lambda_{\min}^{l}\}$, where $\lambda_{\min}^{l}$ is the smallest singular value of $\tilde{X}_{t-1}^{l}$ and $\alpha(t)$ is a positive value which balances the stability and plasticity. For the non-uniform constraint strength, we use a simple strategy that $\alpha(t)$ linearly decreases with task number t observed so far. We perform experiments on the 10-Split CIFAR-100 and 20-Split CIFAR-100 5 runs, and 25-Spilt TinyImageNet 3 runs. Note that the update of the null space in our method is only performed at the end of task, and no data of old data are stored during training. More implementation details, including the hyperparameters settings, can be found in Appendix.

6.2 Performance Comparison

We show the comparison results of the proposed method and baselines in Table 1. The results execept Vanilla, CPR, and GPM are from [49]. According to Table 1, our method achieves the highest average accuracy (ACC) with comparable forgetting (BWT) on all benchmarks.

[4] We do not compare with replay-based methods because they store the data of previous tasks, which is out of the scope of this paper's setting.

Table 1. Results of ACC (%) and BWT (%) evaluated on the all tasks after finishing learning all tasks. [↑] higher is better.

Method	10-S-CIFAR-100		20-S-CIFAR-100		25-S-TinyImagNet	
	ACC [↑]	BWT [↑]	ACC [↑]	BWT [↑]	ACC [↑]	BWT [↑]
Vanilla	34.91	-60.96	30.48	-65.90	16.96	-66.06
EWC [17]	70.77	-2.83	71.66	-3.72	52.33	-6.71
MAS [1]	66.93	-4.03	63.84	-6.29	47.96	-7.04
MUC-MAS [25]	63.73	-3.38	67.22	-5.72	41.18	-4.03
SI [58]	60.57	-5.17	59.76	-8.62	45.27	-4.45
CPR [5]	74.56	-2.51	72.98	-2.32	58.01	-2.45
LwF [23]	70.70	-6.27	74.38	-9.11	56.57	-11.19
GD-WILD [20]	71.27	-18.24	77.16	-14.85	42.74	-34.58
InstAParam [9]	47.84	-11.92	51.04	-4.92	34.64	-10.05
GEM [26]	49.48	2.77	68.89	-1.2	-	-
A-GEM [7]	49.57	-1.13	61.91	-6.88	53.32	-7.68
MEGA [10]	54.17	-2.19	64.98	-5.13	57.12	-5.90
OWM [57]	68.89	-1.88	68.47	-3.37	49.98	-3.64
GPM [43]	73.66	-2.20	75.20	-7.58	58.96	-6.96
Adam-NSCL [49]	75.03	-2.98	75.59	-3.66	59.10	-7.19
AdNS (Ours)	**77.21**	-2.32	**77.33**	-3.25	**59.77**	-4.58

Compared with regularization-based methods, the proposed method achieves over 5% ACC higher with less forgetting than EWC and MAS on all benchmarks. Although MUC-MAS and SI obtained comparable forgetting on the 25-Spilt TinyImageNet, their ACCs are lower than 50%, largely below AdNS's ACC (59.77%). The forgetting of CPR on the 20-Split CIFAR-100 and 25-Spilt TinyImageNet are less, while its performance of ACC is worse than the proposed method on all benchmarks. For the regularization-based methods using knowledge distillation, the ACCs of LwF and GD-WILD are comparable to the proposed method on the 20-Split CIFAR-100 while their stability is very poor. For the architecture-based method, AdNS is significantly better than InstAParam, e.g., over 25% ACC higher on three benchmarks.

Now we compare AdNS with algorithm-based methods. On the 10-Split CIFAR-100, although the forgetting of GEM, A-GEM, MEGA, OWM, and GPM is slightly better than our method, their ACCs are largely below the proposed method. It is also observed that our method can obtain better performance with less forgetting than Adam-NSCL. As shown in Table 1, the ACCs of the proposed method are 2.18%, 1.74%, and 0.66% higher than Adam-NSCL on the 10-Split CIFAR-100, 20-Split CIFAR-100, and 25-Spilt TinyImageNet, respectively. As for forgetting, the BWTs of the proposed method are 0.66%, 0.41%, and 2.61% better than Adam-NSCL on the three benchmarks, respectively. It is because AdNS considers the shared null space between null spaces and also leverages knowledge distillation to further mitigate the stability-plasticity dilemma.

Table 2. Different methods to obtain the shared null space. "Random" obtains the shared null space with the dimensions randomly from $\mathbf{U}_{\text{pre}}^{l}$ and $\mathbf{U}_{\text{cur}}^{l}$.

Method	10-S-CIFAR-100		20-S-CIFAR-100		25-S-TinyImageNet	
	ACC [↑]	BWT [↑]	ACC [↑]	BWT [↑]	ACC [↑]	BWT [↑]
Random	76.10	-4.13	75.70	-5.81	59.07	-6.78
Low-Rank	**76.45**	**-2.87**	**76.31**	**-3.66**	**59.26**	**-5.77**

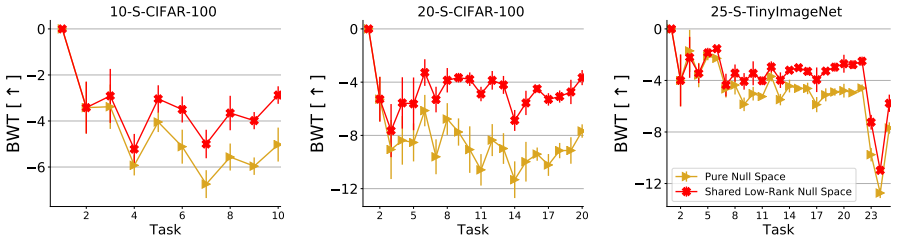

Fig. 2. Comparison of forgetting between the pure null space and the shared low-rank null space. The results are the curves of BWT when the network has been trained on each task ([↑] Higher BWT indicates less forgetting).

6.3 Ablation Studies and Analyses

Effect of Low-Rank. First, we compare the forgetting of using shared low-rank null space and pure null space, in which pure null space projects the gradient onto the current candidate null space instead of the shared null space. According to Fig. 2, with new tasks continually coming, the superiority of the shared low-rank null space on alleviating forgetting becomes more and more obvious than the pure null space, validating that our shared low-rank null space can relieve the catastrophic forgetting due to null space approximation. Next, we validate whether it is effective to use low-rank approximation to obtain the shared space. To realize this, we compare the performance of the shared low-rank null space with the method that obtains the shared null space by extracting dimensions randomly from $\mathbf{U}_{\text{pre}}^{l}$ and $\mathbf{U}_{\text{cur}}^{l}$ [5]. For a fair comparison, intra-task distillation is excluded, and all the settings including the constraints are the same. According to Table 2, using low-rank approximation can achieve higher ACC and better BWT on all benchmarks. Especially for BWT, "Low-Rank" is at least 1% better than the method of "Random", indicating the effectiveness of low-rank approximation.

Effect of Each Component. We now validate the effect of each component to demonstrate that AdNS could achieve better stability-plasticity trade-off. From Table 3, we can find that (1) Both "Low-Rank" and "Non-uniform Constraint

[5] We extract $k_l/2$ dimensions randomly from $\mathbf{U}_{\text{pre}}^{l}$ and another $k_l/2$ dimensions randomly from $\mathbf{U}_{\text{cur}}^{l}$. If the dimension of $\mathbf{U}_{\text{pre}}^{l}$ or $\mathbf{U}_{\text{cur}}^{l}$ is smaller than $k_l/2$, to make up k_l dimensions, we concatenate the whole matrix and the rest dimensions randomly extracted from another matrix.

Table 3. Effect of each component. "NS" denotes "Null Space", "LR" denotes "Low-Rank", "NCS" denotes "Non-uniform Constraint Strength", and "ID" denote Intra-task Distillation.

Module				10-S-CIFAR-100		20-S-CIFAR-100		25-S-TinyImagnet	
NS	LR	NCS	ID	ACC [↑]	BWT [↑]	ACC [↑]	BWT [↑]	ACC [↑]	BWT [↑]
✓				76.11	-5.02	75.53	-7.67	59.20	-7.69
✓	✓			76.45	-2.87	76.31	-3.66	59.26	-5.77
✓		✓		76.15	-4.98	75.66	-7.36	59.42	-5.07
✓	✓	✓		76.45	-2.84	76.34	-3.55	59.27	-5.11
✓			✓	76.99	-4.36	77.04	-6.09	**60.00**	-6.98
✓	✓	✓	✓	**77.21**	**-2.32**	**77.33**	**-3.25**	59.77	**-4.58**

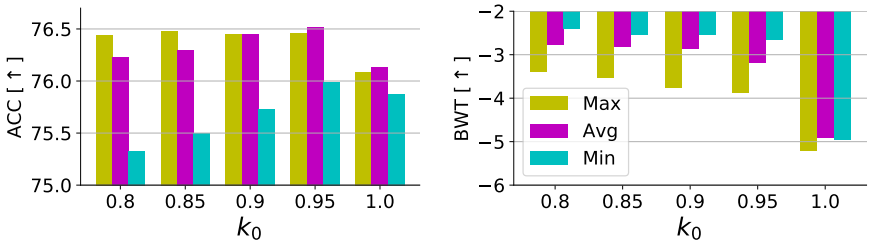

Fig. 3. The effect of k. The dataset is 10-Split CIFAR-100. [↑] higher is better.

Strength" decrease forgetting significantly with better performance. Especially, on the 20-Split CIFAR-100, the shared low-rank null space ("NS" + "LR") is 4.01% BWT better and 0.78% ACC higher than the pure null space ("NS"). On the 25-Spilt TinyImageNet, adding non-uniform constraint strength ("NS" + "NCS") increases BWT 2.62%. (2) Intra-task Distillation improves ACC significantly with better BWT. For example, on the 20-Split CIFAR-100, adding intra-task distillation ("NS" + "ID") increases ACC 1.51 % with less forgetting. (3) Combing all modules can achieve better stability and plasticity simultaneously. For example, on 20-Spilt CIFAR100, combing all modules is 1.8% ACC higher and 4.42% BWT better than the pure null space ("NS").

Effect of k. Now we explore the effect of k. We use k to denote k_l because we apply the same operation for k_l at each layer. Assume that the dimensions of $\mathbf{U}^l_{\text{pre}}$ and $\mathbf{U}^l_{\text{cur}}$ are p and q, respectively. Then "Max", "Avg", and "Min"[6] means that $k = \text{Max}(p,q) \times k_0$, $k = \text{Avg}(p,q) \times k_0$, and $k = \text{Min}(p,q) \times k_0$, respectively, where k_0 is used to adjust the value of k. As shown in Fig. 3, for all strategies, with the decrease of k_0, forgetting continually decreases while ACC first increases and then decreases. It is because with the decrease of k_0, k becomes smaller and the rank of null spaces is smaller. According to Theorems

[6] $\text{Max}(\cdot), \text{Avg}(\cdot)$, and $\text{Min}(\cdot)$ are functions that compute the maximum, average, and minimum values over inputs, respectively.

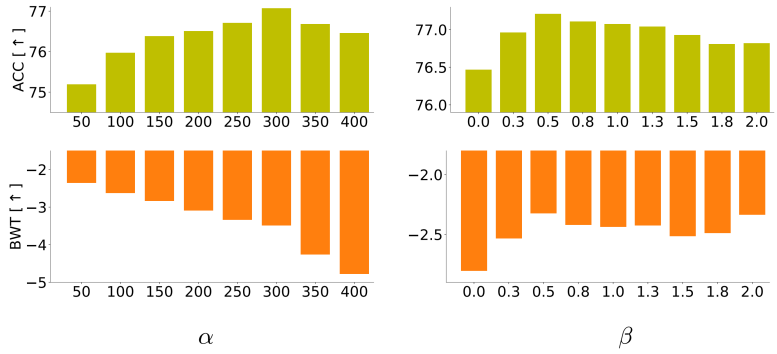

Fig. 4. The effect of α and β. The dataset is 10-Split CIFAR-100. [↑] higher is better.

1 and 2, it would result in better forgetting and worse plasticity, and thus the ACC first increases then decreases as a result of the stability-plasticity dilemma. Considering both stability and plasticity, we choose the strategy of "Avg" for all experiments.

Effect of α and β. Finally, we explore the effect of α in constraint and β in intra-task distillation, respectively. For simplification, we apply constant α for all tasks. Larger α indicates looser constraint in (3). As shown in Fig. 4, with the increase of α, BWT becomes worse, and ACC first increases and then decreases as a result of the stability-plasticity dilemma. The results agree with the theoretical analysis that larger null space (looser constraint) leads to better plasticity and worse stability and vice-versa. For intra-task distillation, as shown in Fig. 4, with proper β, it could mitigate the forgetting and achieve a good balance between stability and plasticity. Too large or too small β will have a negative impact on the performance.

7 Conclusion

To relieve the stability-plasticity dilemma, we propose a new algorithm-based continual learning method, Advanced Null Space (AdNS). Specifically, AdNS extracts the shared low-rank null space based on low-rank approximation and projects the gradient onto the null space to reduce forgetting. Moreover, we introduce non-uniform constraint strength to further alleviate the catastrophic forgetting, and present intra-task distillation to improve performance. Furthermore, we provide theoretical findings of the impact of null space on the stability and plasticity, respectively. Empirical results validate that the proposed algorithm could achieve a better stability-plasticity trade-off.

Acknowledgements. Ms Yajing Kong and Dr Liu Liu are supported by ARC FL-170100117 and DP-180103424.

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 139–154 (2018)
2. Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., Page-Caccia, L.: Online continual learning with maximal interfered retrieval. In: Advances in Neural Information Processing Systems. pp. 11849–11860 (2019)
3. Aljundi, R., Chakravarty, P., Tuytelaars, T.: Expert gate: Lifelong learning with a network of experts. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3366–3375 (2017)
4. Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S.: Dark experience for general continual learning: a strong, simple baseline. arXiv preprint arXiv:2004.07211 (2020)
5. Cha, S., Hsu, H., Hwang, T., Calmon, F.P., Moon, T.: Cpr: Classifier-projection regularization for continual learning. arXiv preprint arXiv:2006.07326 (2020)
6. Chaudhry, A., Khan, N., Dokania, P.K., Torr, P.H.: Continual learning in low-rank orthogonal subspaces. arXiv preprint arXiv:2010.11635 (2020)
7. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. arXiv preprint arXiv:1812.00420 (2018)
8. Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: Continual learning with tiny episodic memories (2019)
9. Chen, H.J., Cheng, A.C., Juan, D.C., Wei, W., Sun, M.: Mitigating forgetting in online continual learning via instance-aware parameterization. Advances in Neural Information Processing Systems **33** (2020)
10. Guo, Y., Liu, M., Yang, T., Rosing, T.: Improved schemes for episodic memory based lifelong learning algorithm. In: Conference on Neural Information Processing Systems (2020)
11. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision. pp. 630–645. Springer (2016)
12. Isele, D., Cosgun, A.: Selective experience replay for lifelong learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
13. Jerfel, G., Grant, E., Griffiths, T., Heller, K.A.: Reconciling meta-learning and continual learning with online mixtures of tasks. In: Advances in Neural Information Processing Systems. pp. 9122–9133 (2019)
14. Jing, Y., Yang, Y., Wang, X., Song, M., Tao, D.: Amalgamating knowledge from heterogeneous graph neural networks. In: CVPR (2021)
15. Jung, H., Ju, J., Jung, M., Kim, J.: Less-forgetting learning in deep neural networks. arXiv preprint arXiv:1607.00122 (2016)
16. Kemker, R., McClure, M., Abitino, A., Hayes, T., Kanan, C.: Measuring catastrophic forgetting in neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
17. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
18. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
19. Lee, J., Hong, H.G., Joo, D., Kim, J.: Continual learning with extended kronecker-factored approximate curvature. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9001–9010 (2020)

20. Lee, K., Lee, K., Shin, J., Lee, H.: Overcoming catastrophic forgetting with unlabeled data in the wild. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 312–321 (2019)
21. Lee, S.W., Kim, J.H., Jun, J., Ha, J.W., Zhang, B.T.: Overcoming catastrophic forgetting by incremental moment matching. In: Advances in neural information processing systems. pp. 4652–4662 (2017)
22. Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. arXiv preprint arXiv:1904.00310 (2019)
23. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
24. Lin, S., Yang, L., Fan, D., Zhang, J.: Trgp: Trust region gradient projection for continual learning (2022)
25. Liu, Y., Parisot, S., Slabaugh, G., Jia, X., Leonardis, A., Tuytelaars, T.: 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. pp. 699–716. Springer (2020)
26. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. In: Advances in neural information processing systems. pp. 6467–6476 (2017)
27. Mallya, A., Davis, D., Lazebnik, S.: Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 67–82 (2018)
28. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7765–7773 (2018)
29. Masse, N.Y., Grant, G.D., Freedman, D.J.: Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. Proceedings of the National Academy of Sciences **115**(44), E10467–E10475 (2018)
30. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24, pp. 109–165. Elsevier (1989)
31. Mirzadeh, S.I., Farajtabar, M., Ghasemzadeh, H.: Dropout as an implicit gating mechanism for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 232–233 (2020)
32. Mirzadeh, S.I., Farajtabar, M., Pascanu, R., Ghasemzadeh, H.: Understanding the role of training regimes in continual learning. Advances in Neural Information Processing Systems **33** (2020)
33. Nguyen, C.V., Li, Y., Bui, T.D., Turner, R.E.: Variational continual learning. arXiv preprint arXiv:1710.10628 (2017)
34. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks **113**, 54–71 (2019)
35. Park, D., Hong, S., Han, B., Lee, K.M.: Continual learning by asymmetric loss approximation with single-side overestimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3335–3344 (2019)
36. Rannen, A., Aljundi, R., Blaschko, M.B., Tuytelaars, T.: Encoder based lifelong learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1320–1328 (2017)
37. Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y.W., Hadsell, R.: Continual unsupervised representation learning. In: Advances in Neural Information Processing Systems. pp. 7647–7657 (2019)

38. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017)
39. Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., Tesauro, G.: Learning to learn without forgetting by maximizing transfer and minimizing interference. arXiv preprint arXiv:1810.11910 (2018)
40. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G.: Experience replay for continual learning. In: Advances in Neural Information Processing Systems. pp. 350–360 (2019)
41. Rosenfeld, A., Tsotsos, J.K.: Incremental learning through deep adaptation. IEEE transactions on pattern analysis and machine intelligence **42**(3), 651–663 (2018)
42. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)
43. Saha, G., Garg, I., Roy, K.: Gradient projection memory for continual learning. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=3AOj0RCNC2
44. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: International Conference on Machine Learning. pp. 4548–4557. PMLR (2018)
45. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: Advances in neural information processing systems. pp. 2990–2999 (2017)
46. Stanford: Tiny ImageNet Challenge (CS231n) (2015), http://tiny-imagenet.herokuapp.com/
47. Tang, S., Chen, D., Zhu, J., Yu, S., Ouyang, W.: Layerwise optimization by gradient decomposition for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9634–9643 (2021)
48. Tani, J.: Exploring robotic minds: actions, symbols, and consciousness as self-organizing dynamic phenomena. Oxford University Press (2016)
49. Wang, S., Li, X., Sun, J., Xu, Z.: Training networks in null space of feature covariance for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 184–193 (June 2021)
50. Wang, Z., Liu, L., Duan, Y., Kong, Y., Tao, D.: Continual learning with lifelong vision transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 171–181 (June 2022)
51. Wang, Z., Liu, L., Duan, Y., Tao, D.: Continual learning through retrieval and imagination. Proceedings of the AAAI Conference on Artificial Intelligence **36**(8), 8594–8602 (Jun 2022). https://doi.org/10.1609/aaai.v36i8.20837, https://ojs.aaai.org/index.php/AAAI/article/view/20837
52. Wang, Z., Liu, L., Tao, D.: Deep streaming label learning. In: International Conference on Machine Learning (ICML). vol. 119, pp. 9963–9972 (2020)
53. Wu, L., Liu, B., Stone, P., Liu, Q.: Firefly neural architecture descent: a general approach for growing neural networks. Advances in Neural Information Processing Systems **33** (2020)
54. Yiduo, G., Wenpeng, H., Dongyan, Z., Bing, L.: Adaptive orthogonal projection for continual learning. AAAI (2022)
55. Yin, D., Farajtabar, M., Li, A., Levine, N., Mott, A.: Optimization and generalization of regularization-based continual learning: a loss approximation viewpoint (2020)

56. Yoon, J., Kim, S., Yang, E., Hwang, S.J.: Scalable and order-robust continual learning with additive parameter decomposition. arXiv preprint arXiv:1902.09432 (2019)
57. Zeng, G., Chen, Y., Cui, B., Yu, S.: Continual learning of context-dependent processing in neural networks. Nature Machine Intelligence (NMI) **1**(8), 364–372 (2019)
58. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International Conference on Machine Learning. pp. 3987–3995. PMLR (2017)
59. Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., Kuo, C.C.J.: Class-incremental learning via deep model consolidation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1131–1140 (2020)
60. Zhou, G., Sohn, K., Lee, H.: Online incremental feature learning with denoising autoencoders. In: Artificial intelligence and statistics. pp. 1453–1461 (2012)