Memory Efficient Data-Free Distillation for Continual Learning (Appendix)

Xiaorong Li[a,1], Shipeng Wang[a,1], Jian Sun[a,*], Zongben Xu[a]

[a]*School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, Shaanxi, China*

A. Related Work

The existing continual learning methods can be roughly divided into five categories: rehearsal-based, architecture-based, algorithm-based, regularization-based and distillation-based approaches.

Rehearsal-based approach commonly relies on storing data of previous tasks in a memory buffer, which is then replayed when learning a new task to maintain network performance on previous tasks [1, 2]. Some works utilize data from previous tasks to formulate a constrained optimization problem to avoid the increases of training losses on previous tasks [3, 4, 5]. Gu et al. [6] focus on the setting where the training data is accessed only once during the training phase. They propose a sample selection strategy that selects stored samples whose network parameter gradients are most interfered by new incoming samples. However, most of these works suffer from class imbalance [7] between classes from previous and new tasks. Addition-

*Corresponding author.

Email addresses: lixiaorong@stu.xjtu.edu.cn (Xiaorong Li), wangshipeng8128@stu.xjtu.edu.cn (Shipeng Wang), jiansun@xjtu.edu.cn (Jian Sun), zbxu@xjtu.edu.cn (Zongben Xu)

[1]Equal contribution.

ally, storing data of previous tasks introduces privacy and security concerns. An alternative way is generating synthetic data of previous tasks based on generative models [8, 9], whose performance depends on the quality of synthetic data. Wei et al. [10] further apply knowledge distillation to mitigate catastrophic forgetting of the generator for tackling a new problem named incremental zero-shot learning, where the network incrementally learns the knowledge of new classes and will be tested on all previously learned classes and unseen classes. Nonetheless, generating high-quality synthetic data remains a challenge.

Architecture-based approach concentrates on dynamically customizing the network's structure for each task, such as expanding [11], pruning [12] or masking [13] the neural connections, to alleviate catastrophic forgetting. For example, Packnet [12] prunes the network and dedicates a specific subpart of the network for each task. Piggyback [13] learns masks of network parameters to identify the task-specific part of the network. Learn-to-Grow [11] employs architecture search to find the optimal structure for each task. DRT [14] disentangles the latent features into class-disentangled and task-disentangled features by two branches of networks. In contrast to them, our method avoids modifying network architecture for each task, but designs a novel regularizer in network training loss, which is easy to implement.

Algorithm-based approach concentrates on designing a network parameter update rule on the new task which alleviates the performance deterioration on previous tasks. For example, OWM [15] projects the gradient obtained at each training step into an orthogonal space of the space spanned by input features of all network layers, where the projection matrix is up-

dated by recursive least squares. Adam-NSCL [16] considers the network parameter update as the projection of gradient into the approximate null space of uncentered feature covariance based on theoretical analysis. GPM [17] forces the network parameter update to lie in the orthogonal space of the space spanned by input features, which is dependent on storing the basis of the complement to orthogonal space obtained by employing singular value decomposition on partial input features of previous tasks. However, to obtain the network parameter update in each step, these works [15, 16, 17] need to store the projection matrix or basis of subspace, where the memory usages of them are larger than ours.

Regularization-based approach penalizes the variations of network parameters to preserve the performance on previous tasks, where each network parameter is associated with an importance weight. Previous works focus on designing the importance weight in different ways [18, 19, 20, 21]. For example, MAS [19] aims to consider the sensitivity of the output function, and finally implements this by estimating the sensitivity of the norm of predicted output w.r.t. the parameters to measure the importance weight. Inspired by [19], MUC-MAS [21] proposes to integrate an ensemble of auxiliary classifiers to estimate the importance weight, where the auxiliary classifiers are trained on out-of-distribution data irrelevant to the current task. However, these works implicitly assume that the network parameters are independent without considering the impact of the correlation among parameters on the network performance, while our method takes this into consideration.

Distillation-based approach is inspired by knowledge distillation [22]. To preserve the performance on previous tasks in continual learning, a distil-

lation term is utilized to penalize the variations of outputs between teacher and student networks, where the teacher and student networks are respectively set as the network learned from the previous tasks and the network being trained on the current task. Ideally, the data associated with the distillation term should be the original data of previous tasks. However, the full datasets of previous tasks are inaccessible in continual learning, thus existing works mainly focus on how to substitute the datasets of previous tasks [23, 24, 25]. For example, methods in [24, 26] employ the training data of the current task for knowledge distillation. iCaRL [23] stores a few samples of previous tasks as a coreset for knowledge distillation. Besides using coreset, the method in [25] additionally leverages a large stream of unlabeled data in the wild which is assumed to be available at any time for knowledge distillation. Commonly, the dataset they adopted for knowledge distillation may fail to reflect the knowledge of the full original datasets of previous tasks, as a result of which, these methods may not well distill full knowledge of the previous tasks. Moreover, the teacher networks need to be saved in their methods. Different from them, we make use of the whole knowledge of datasets of previous tasks encoded by gradients for knowledge distillation, and we do not need to store teacher networks.

B. Proof of Theorem 1

In this section, we begin by introducing the notation used in the proof, followed by presenting Lemma 1 which forms the basis of proof for Theorem 1. Then we recall Theorem 1 and provide its proof.

Notation. To mitigate catastrophic forgetting, we employ a distillation

loss penalizing the output variations between teacher and student networks on previous task data X_i of task $\mathcal{T}_i$, i.e., $\min_w \|f(X_i, w) - f(X_i, w_i^*)\|_2^2$, where $f(\cdot, w_i^*)$ is the teacher network and $f(\cdot, w)$ is the student network. Since the previous task data X_i are not available when learning task $\mathcal{T}_t$, we approximate $f(X_i, w)$ with its first-order Taylor expansion at $w = w_i^*$, i.e., $f(X_i, w) \approx f(X_i, w_i^*) + G_i^\top (w - w_i^*)$, where $G_i = \frac{\partial f(X_i, w_i^*)}{\partial w_i^*} \in \mathbb{R}^{|w| \times C}$ with $|w|$ and C as the dimension of w and the number of classes respectively. The gradients G_i are further compressed and recovered for memory efficiency, and the recovered gradients are denoted as $\tilde{G}_i$.

For clarity in notation, we omit the subscript i denoting the task index of gradients G_i. The gradient matrix $G \in \mathbb{R}^{|w| \times C}$ on task $\mathcal{T}_i$ is the concatenation of layer-wise gradient matrix $G^l \in \mathbb{R}^{|w^l| \times C}$ such that $|w| = \sum_{l=1}^{L} |w^l|$ with L denoting the number of layers of the network. We additionally introduce a notation $\bar{G} \in \mathbb{R}^{|w| \times C}$ as the concatenation of matrix $EXP(\bar{G}^l) \in \mathbb{R}^{|w^l| \times C}$ $(l = 1, \ldots, L)$ for proving Lemma 1, where $EXP(\cdot)$ denotes the operation that copies each element of the input object to the elements corresponding to the convolutional kernel. For the l-th convolutional layer of the network with n_{out}^l 3D convolutional kernels of size $n_{in}^l \times k^l \times k^l$, the gradients of the l-th layer $G^l \in \mathbb{R}^{|w^l| \times C}$ are gradients of C elements of network output w.r.t. the j-th convolutional kernel, where $G^l = \left[\mathbf{g}_1{}^\top, \ldots, \mathbf{g}_{n_{out}^l}{}^\top \right]^\top$, $\mathbf{g}_j \in \mathbb{R}^{(n_{in}^l \times k^l \times k^l) \times C}$ $(j = 1, \ldots, n_{out}^l)$ and $|w^l| = n_{out}^l \times n_{in}^l \times k^l \times k^l$.

Recall the proposed compression approach, we first average the gradients of each convolutional kernel with size $n_{in}^l \times k^l \times k^l$, which results in a compact matrix $\bar{G}^l \in \mathbb{R}^{n_{out}^l \times C}$. Specifically, we obtain $\bar{G}^l$ by $\bar{G}^l = \left[\bar{\mathbf{g}}_1^\top, \ldots, \bar{\mathbf{g}}_{n_{out}^l}^\top \right]^\top$, where $\bar{\mathbf{g}}_j = \frac{1}{n_{in}^l \times k^l \times k^l} (\mathbf{1}^\top \mathbf{g}_j)^\top \in \mathbb{R}^C$ and $\mathbf{1} \in \mathbb{R}^{n_{in}^l \times k^l \times k^l}$ is a column vector

with its all elements as 1. We then apply SVD to the compact matrix $\bar{G}^l \in \mathbb{R}^{n^l_{out} \times C}$ and obtain its SVD decompositions, i.e., $U^l \in \mathbb{R}^{n^l_{out} \times r}, V^l \in \mathbb{R}^{C \times r}$ and $\Lambda^l \in \mathbb{R}^{r \times r}$ $(r < C)$, which are used to approximate the compact matrix $\bar{G}^l$. The resulting approximate compact matrix is denoted as $\tilde{G}^l \in \mathbb{R}^{n^l_{out} \times C}$, which is obtained by $\tilde{G}^l = U^l \Lambda^l {V^l}^\top$.

We now introduce Lemma 1 which analyzes the approximation error between the gradients G_i and the recovered gradients $\tilde{G}_i$. Lemma 1 forms the basis of the proof of Theorem 1.

Lemma 1. *Given the gradients G of the network output $f(X_i, w_i^*)$ w.r.t. the learned network parameters w_i^*, i.e., $G = \frac{\partial f(X_i, w_i^*)}{\partial w_i^*}$, and the recovered gradients $\tilde{G}$, then we have the following bound on the error between G and $\tilde{G}$:*

$$\|G - \tilde{G}\|_2^2 \leq \sum_{l=1}^{L} \left[\left(\sum_{j=1}^{n^l_{out}} \sum_{c=1}^{C} (n^l_{in} \times k^l \times k^l) \times \sigma^2_{j,c} \right) + (n^l_{in} \times k^l \times k^l)(\sum_{j=r+1}^{C-r} s_j^2) \right], \tag{1}$$

where $\sigma^2_{j,c}$ is the variance of the gradients of the c-th element of the network output $f(X_i, w_i^*)$ w.r.t. the j-th convolutional kernel, r is the number of selected singular values, s_j represents the j-th singular value that is not selected.

Proof.

$$\begin{aligned} \|G - \tilde{G}\|_2^2 &= \|G - \bar{G} + \bar{G} - \tilde{G}\|_2^2 \\ &\leq \|G - \bar{G}\|_2^2 + \|\bar{G} - \tilde{G}\|_2^2 \end{aligned}$$

$$
\begin{aligned}
&= \sum_{l=1}^{L} \left[\|G^l - EXP(\bar{G}^l)\|_2^2 + \|EXP(\bar{G}^l) - EXP(\tilde{G}^l)\|_2^2 \right] \\
&= \sum_{l=1}^{L} \left[\left(\sum_{j=1}^{n_{out}^l} \|\mathbf{g}_j - EXP(\bar{\mathbf{g}}_j)\|_2^2 \right) + (n_{in}^l \times k^l \times k^l) \|\bar{G}^l - \tilde{G}^l\|_2^2 \right] \\
&= \sum_{l=1}^{L} \left[\left(\sum_{j=1}^{n_{out}^l} \sum_{c=1}^{C} (n_{in}^l \times k^l \times k^l) \times \sigma_{j,c}^2 \right) + (n_{in}^l \times k^l \times k^l) \left(\sum_{j=r+1}^{C-r} s_j^2 \right) \right]. \quad (2)
\end{aligned}
$$

□

We now recall Theorem 1 and provide its proof. The proof of Theorem 1 is based on the conclusion of Lemma 1.

Theorem 1 (Bound on Relative Approximation Error)**.** *Given the network output $f(X_i, w)$ on previous task data X_i, we approximate it by $f(X_i, w) \approx f(X_i, w_i^*) + \tilde{G}_i^\top (w - w_i^*)$, where $\tilde{G}_i$ is an approximation of the gradients G_i of $f(X_i, w_i^*)$ w.r.t. w_i^*, then we have the following bound on relative approximation error between $f(X_i, w)$ and $f(X_i, w_i^*) + \tilde{G}_i^\top (w - w_i^*)$:*

$$
\frac{\|f(X_i, w) - f(X_i, w_i^*) - \tilde{G}_i^\top (w - w_i^*)\|_2^2}{\|f(X_i, w_i^*)\|_2^2} \leq e_1 + e_2 \quad (3)
$$

with

$$
e_1 = \frac{\|f(X_i, w) - f(X_i, w_i^*) - G_i^\top (w - w_i^*)\|_2^2}{\|f(X_i, w_i^*)\|_2^2}, \quad (4)
$$

$$
\begin{aligned}
e_2 = &\frac{\|(w - w_i^*)\|_2^2 \alpha^2 \sum_{l=1}^{L} \sum_{j=1}^{n_{out}^l} \sum_{c=1}^{C} (n_{in}^l \times k^l \times k^l) \times \sigma_{j,c}^2}{\|f(X_i, w_i^*)\|_2^2} \\
&+ \frac{\|(w - w_i^*)\|_2^2 \alpha^2 \sum_{l=1}^{L} (n_{in}^l \times k^l \times k^l) (\sum_{j=r+1}^{C-r} s_j^2)}{\|f(X_i, w_i^*)\|_2^2}, \quad (5)
\end{aligned}
$$

where $\alpha = max(\{cos\theta_1, \ldots, cos\theta_C\})$ *is the maximum cosine value with* θ_c $(c = 1, \ldots, C)$ *as the angle between the c-th column of* $G_i - \tilde{G}_i$ *and* $w - w_i^*$, $\sigma_{j,c}^2$ *is the variance of the gradients of the c-th element of the network output* $f(X_i, w_i^*)$ *w.r.t. the j-th convolutional kernel,* r *is the number of selected singular values,* s_j *represents the j-th singular value that is not selected.*

According to Theorem 1, the bound on relative approximation error is determined by e_1 and e_2. e_1 measures the relative truncation error of the Taylor expansion, while e_2 quantifies the approximation error of compressing the gradients. Specifically, the first term and second term of e_2 are influenced by our average operation and SVD respectively.

Proof. Considering the relative approximation error $\frac{\|f(X_i,w)-f(X_i,w_i^*)-\tilde{G}_i^\top(w-w_i^*)\|_2^2}{\|f(X_i,w_i^*)\|_2^2}$, we have

$$
\begin{aligned}
&\frac{\|f(X_i, w) - f(X_i, w_i^*) - \tilde{G}_i^\top (w - w_i^*)\|_2^2}{\|f(X_i, w_i^*)\|_2^2} \\
&= \frac{\|f(X_i, w) - f(X_i, w_i^*) - G_i^\top (w - w_i^*) + G_i^\top (w - w_i^*) - \tilde{G}_i^\top (w - w_i^*)\|_2^2}{\|f(X_i, w_i^*)\|_2^2} \\
&\leq \frac{\|f(X_i, w) - f(X_i, w_i^*) - G_i^\top (w - w_i^*)\|_2^2}{\|f(X_i, w_i^*)\|_2^2} + \frac{\|(G_i - \tilde{G}_i)^\top (w - w_i^*)\|_2^2}{\|f(X_i, w_i^*)\|_2^2} \\
&= e_1 + \frac{\sum_{c=1}^{C} \left[G_i - \tilde{G}_i\right]_c^2 (w - w_i^*)^2 cos^2\theta_c}{\|f(X_i, w_i^*)\|_2^2} \\
&\leq e_1 + \frac{\sum_{c=1}^{C} \left[G_i - \tilde{G}_i\right]_c^2 (w - w_i^*)^2 \alpha^2}{\|f(X_i, w_i^*)\|_2^2}
\end{aligned}
$$

$$= e_1 + \frac{\|(G_i - \tilde{G}_i)\|_2^2 \|(w - w_i^*)\|_2^2 \alpha^2}{\|f(X_i, w_i^*)\|_2^2}, \tag{6}$$

where $[\cdot]_c$ denotes the c-th column of the matrix, $\alpha = max(\{cos\theta_1, \ldots, cos\theta_C\})$ is the maximum cosine value and θ_c $(c = 1, \ldots, C)$ is the angle between the c-th column of $G_i - \tilde{G}_i$ and $w - w_i^*$.

By substituting inequality (2) in the second term of r.h.s. of inequality (6), we have

$$\frac{\|(G_i - \tilde{G}_i)\|_2^2 \|(w - w_i^*)\|_2^2 \alpha^2}{\|f(X_i, w_i^*)\|_2^2} \leq e_2. \tag{7}$$

Combining inequalities (6) and (7), we can conclude that

$$\frac{\|f(X_i, w) - f(X_i, w_i^*) - \tilde{G}_i^\top (w - w_i^*)\|_2^2}{\|f(X_i, w_i^*)\|_2^2} \leq e_1 + e_2.$$

□

C. Details on Compressing Gradients of BN layer

For the batch normalization (BN) layer with n feature channels, each channel is equipped with an affine transformation with two parameters, i.e., scaling weight and bias. Therefore, the gradients of scaling weight and bias at BN layer are both with the size of $n \times C$, where C is the number of classes. We respectively compress the gradients of scaling weight and bias by SVD without using average operation.

D. Algorithm

Algorithm 1 Data-free distillation for continual learning.

Inputs: datasets $\{X_t, Y_t\}$ for task $\mathcal{T}_t \in \{\mathcal{T}_1, \mathcal{T}_2, \dots\}$, network $f(\cdot, w)$ of depth L, hyperparameter λ.

Output: learned network $f(\cdot, w^*)$.

1: Memory buffer $\mathcal{M} = \emptyset$.
2: **for** task $\mathcal{T}_t \in \{\mathcal{T}_1, \mathcal{T}_2, \dots\}$ **do**
3: **if** $t = 1$ **then**
4: Initialize w randomly.
5: $w_t^* \leftarrow \arg\min_w \mathcal{L}(\hat{Y}_t(w), Y_t)$.
6: **else**
7: *# Reconstruction stage as illustrated in Sec.3.2*
8: Take $\{U_i^l, \Lambda_i^l, V_i^l\}_{l=1}^L$ from $\mathcal{M}$ for $i = 1, \dots, t-1$.
9: $G_i = \text{Reconstruct}(U_i^1, \Lambda_i^1, V_i^1, \dots, U_i^L, \Lambda_i^L, V_i^L)$ for $i = 1, \dots, t-1$.
10: Initialize w with w_{t-1}^*.
11: $w_t^* \leftarrow \arg\min_w \mathcal{L}(\hat{Y}_t(w), Y_t) + \lambda \sum_{i=1}^{t-1} (w^\top G_i G_i^\top w - 2w^\top G_i G_i^\top w_i^*)$.
12: **end if**
13: Obtain the gradient $G_t = \frac{\partial f(X_t, w_t^*)}{\partial w_t^*}$.
14: *# Compression stage as illustrated in Sec. 3.2*
15: $U_t^l, \Lambda_t^l, V_t^l = \text{Compress}(G_t^l)$.
16: $\mathcal{M} = \mathcal{M} \cup \{U_t^l, \Lambda_t^l, V_t^l\}_{l=1}^L$.
17: **end for**

In this Appendix, we summarize the algorithm for the proposed DFD in Alg. 1. Given a task sequence $\{\mathcal{T}_1, \mathcal{T}_2, \dots\}$, The basic pipeline of DFD is iteratively optimizing problem (5) of the manuscript to acquire new knowledge while reserving the performance on previous tasks.

Concretely, to train the network acquiring knowledge of the first task $\mathcal{T}_1$, we minimize the cross-entropy loss to obtain the learned network parameter w_t^* as illustrated in line 9. When training the network on task $\mathcal{T}_t$ ($t > 1$), we first recover the gradients as described in lines 13 to 14. Then we optimize problem (5) of the manuscript as described in line 18. After obtaining the learned network parameters for task $\mathcal{T}_t$, we update the memory buffer as

described in line 26 following the compression stage in line 25.

E. Details on the Hyperparameters

The setting of hyperparameters of our method is listed in Table 1, where "lr", "wd" and "bs" denote the initial learning rate, weight decay and batch size respectively. The network is trained for 80 epochs in total for all experiments. The initial learning rate decays at epochs 30 and 60 with a multiplier of 0.5 for all experiments.

Table 1: The settings of hyperparameters of our method on all experiments.

Experiments	lr	wd	bs	λ
10-split CIFAR-100	1×10^{-3}	5×10^{-5}	32	20
20-split CIFAR-100	1×10^{-4}	5×10^{-4}	32	10
25-split TinyImageNet	5×10^{-5}	1×10^{-5}	16	50
20-split miniImageNet	5×10^{-5}	5×10^{-5}	32	20
10-split SubImageNet	1×10^{-4}	5×10^{-5}	16	200

F. Effect of r on the Bound on Relative Approximation Error

We now study the impact of the number of stored top singular values, denoted by r, on the bound of the relative approximation error. According to Theorem 1 of the paper, r is irrelevant to e_1 but affects the value of e_2 which captures the approximation error of compressing the gradients. To investigate the effect of r on the bound of error, we report the values of the bound for different values of r on 10-split CIFAR-100 in Table 2, where we set $w = w_{10}^*$ and $w_i^* = w_1^*$. As shown in Table 2, we observe that the value of e_1 is dominant and the values of e_2 remain stable across different choices of r.

Table 2: The effect of r on the bound of relative approximation error in 10-split CIFAR-100.

The number of stored top singular values	e_1	e_2	α in e_2	$\|(w_{10}^* - w_1^*)\|_2^2\alpha^2$ in e_2
$r = 1$	0.32	8.9×10^{-5}	4.9×10^{-3}	1.3×10^{-3}
$r = 2$	0.32	3.7×10^{-5}	6.2×10^{-3}	7.1×10^{-4}
$r = 3$	0.32	1.2×10^{-5}	4.6×10^{-3}	2.8×10^{-4}
$r = 4$	0.32	1.8×10^{-5}	6.7×10^{-3}	5.3×10^{-4}
$r = 5$	0.32	1.4×10^{-5}	8.0×10^{-3}	5.2×10^{-4}
$r = 6$	0.32	8.0×10^{-6}	6.8×10^{-3}	3.5×10^{-4}
$r = 7$	0.32	8.3×10^{-6}	8.1×10^{-3}	4.5×10^{-4}
$r = 8$	0.32	9.1×10^{-6}	9.5×10^{-3}	6.0×10^{-4}

This finding is consistent with our observation that the network performance is robust to the changes of r, as demonstrated in Figure 4 of the paper.

References

[1] 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.*, pages 524–540, 2020.

[2] Chen Zhuang, Shaoli Huang, Gong Cheng, and Jifeng Ning. Multi-criteria selection of rehearsal samples for continual learning. *Pattern Recognition*, 132:108907, 2022.

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

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *Int. Conf. Learn. Represent.*, 2019.

[5] 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] Yanan Gu, Xu Yang, Kun Wei, and Cheng Deng. Not just selection, but exploration: Online class-incremental continual learning via dual view consistency. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7442–7451, 2022.

[7] 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.

[8] Huaiyu Li, Weiming Dong, and Bao-Gang Hu. Incremental concept learning via online generative memory recall. *IEEE Trans. Neural Networks Learn. Syst.*, 32(7):3206–3216, 2021.

[9] Qicheng Lao, Mehrzad Mortazavi, Marzieh Tahaei, Francis Dutil, Thomas Fevens, and Mohammad Havaei. Focl: Feature-oriented continual learning for generative models. *Pattern Recognition*, 120:108127, 2021.

[10] Kun Wei, Cheng Deng, Xu Yang, and Dacheng Tao. Incremental zero-shot learning. *IEEE Trans. Cyber.*, 52(12):13788–13799, 2022.

[11] 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.

[12] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7765–7773, 2018.

[13] 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.

[14] Kun Wei, Da Chen, Yuhong Li, Xu Yang, Cheng Deng, and Dacheng Tao. Incremental embedding learning with disentangled representation translation. *IEEE Trans. Neural Networks Learn. Syst.*, pages 1–13, 2022.

[15] 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.

[16] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 184–193, 2021.

[17] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *Int. Conf. Learn. Represent.*, 2021.

[18] 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.

[19] 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.

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

[21] 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.*, pages 699–716, 2020.

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

[23] 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.

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

[25] 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.

[26] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5871–5880, 2021.