Continual Forgetting for Pre-trained Vision Models

Hongbo Zhao[1,3*] Bolin Ni[1,3*] Junsong Fan[2] Haochen Wang[1,3] Yuxi Wang[2]

Fei Zhu[2] Yuntao Chen[2] Gaofeng Meng[1,2,3†] Zhaoxiang Zhang[1,2,3,4†]

[1]State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences

[2]Centre for Artificial Intelligence and Robotics, Hong Kong Institute of Science & Innovation, Chinese Academy of Sciences

[3] University of Chinese Academy of Sciences   [4] Shanghai Artificial Intelligence Laboratory

{zhaohongbo2022, zhaoxiang.zhang}@ia.ac.cn   gfmeng@nlpr.ia.ac.cn

Abstract

For privacy and security concerns, the need to erase unwanted information from pre-trained vision models is becoming evident nowadays. In real-world scenarios, erasure requests originate at any time from both users and model owners. These requests usually form a sequence. Therefore, under such a setting, selective information is expected to be continuously removed from a pre-trained model while maintaining the rest. We define this problem as continual forgetting and identify two key challenges. (i) For unwanted knowledge, efficient and effective deleting is crucial. (ii) For remaining knowledge, the impact brought by the forgetting procedure should be minimal. To address them, we propose Group Sparse LoRA (GS-LoRA). Specifically, towards (i), we use LoRA modules to fine-tune the FFN layers in Transformer blocks for each forgetting task independently, and towards (ii), a simple group sparse regularization is adopted, enabling automatic selection of specific LoRA groups and zeroing out the others. GS-LoRA is effective, parameter-efficient, data-efficient, and easy to implement. We conduct extensive experiments on face recognition, object detection and image classification and demonstrate that GS-LoRA manages to forget specific classes with minimal impact on other classes. Codes will be released on https://github.com/bjzhb666/GS-LoRA.

1. Introduction

As pre-trained models become larger nowadays, more training data are required. These data are usually collected through various ways such as the Internet, books, publicly available datasets, and manual labeling. Within the vast amount of data, there is often erroneous or privacy-sensitive
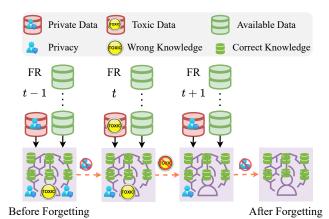

Figure 1. **Illustration of continual forgetting**, which aims to remove specific knowledge in pre-trained models sequentially. "FR" stands for Forgetting Request. The red data (privacy data, toxic data, *etc*.) contains unwanted knowledge which needs to be removed, while the rest should be maintained. The model inherits parameters from the last forgetting task at the beginning of a new forgetting task.

information and pre-trained models may learn from it. For instance, the ImageNet Roulette project [15, 55] shows models tend to be biased toward racist, misogynistic, and cruel *etc*. Furthermore, with increased public awareness of privacy protection and updated privacy regulations [24, 61], individuals are now demanding the removal of any privacy-related information immediately. Therefore, practical model erasing techniques are required upon receiving a deletion request. In real-world scenarios, these requests usually originate at any time from both users and model owners and naturally form a sequence. Under such a setting, selective information is expected to be *continuously* removed from a pre-trained model while maintaining the rest. We identify this novel problem as **continual forgetting** and illustrate it in Fig. 1, where privacy and wrong knowledge need to be removed from the pre-trained model

---

*Equal contribution.

†Corresponding author.

sequentially. This task holds significance in upholding privacy and reducing unwanted model bias, such as gender and racial discrimination, in real-world applications.

A related research topic is machine unlearning, which refers to the process of removing or erasing knowledge or patterns that a machine learning model has learned during training. Prior attempts mostly focused on typical machine learning algorithms [5, 13, 33, 49], *e.g.*, linear/logistic regression [33, 49], and thus have a limited scope of application. Recent studies that explored unlearning techniques for deep learning models are either computationally heavy and only effective on small-scale problems [23, 26, 66], or require specific designs in the pre-training process [5, 82], which are impractical. These approaches lack the ability to proceed with numerous everyday requests and thus are not capable of continual forgetting. We identify two key challenges in the design of continual forgetting algorithms. *(i)* Efficient and effective deleting for unwanted knowledge is crucial. Especially for continual forgetting scenarios, lightweight and fast modifications are more important to achieve deleting information promptly. *(ii)* Should have minimal impact on remaining knowledge, *i.e.*, catastrophic forgetting should be mitigated.

To this end, we propose **G**roup **S**parse **LoRA** (GS-LoRA). Specifically, to achieve efficient forgetting on unwanted knowledge, we utilize LoRA [31] to fine-tune the FFN modules in Transformer blocks inspired by parameter-efficient fine-tuning (PEFT) techniques [30, 31, 42] and Geva *et al.* [21]. To mitigate catastrophic forgetting on remaining knowledge [37], we use a group sparse regularizer to achieve a sparse and accurate modification of FFN modules, as fine-tuning fewer parameters is observed to be effective [50, 51, 79, 87] towards alleviating catastrophic forgetting. This is akin to conducting minimally invasive surgery on a model instead of a major surgery. GS-LoRA is effective, parameter-efficient, data-efficient, easy to implement, and applicable to large models. To verify the effectiveness of our proposed GS-LoRA, we initially conduct experiments on face recognition because it is a fundamental privacy-sensitive task, and then evaluate it on a more general task, *i.e.*, object detection. Empirically, GS-LoRA performs well in both settings, indicating that our method is a general framework with minimal domain knowledge and few inductive biases across various vision tasks.

Our contributions are summarized as follows:

- We are the first to propose the continual forgetting problem, which is essential in practical scenarios for fast model editing and privacy protection.
- To address this problem, we first identify two challenges and propose GS-LoRA to achieve efficient and effective forgetting while maintaining the performance of the rest.
- Extensive experiments on both face recognition and object detection demonstrate that GS-LoRA effectively for-

gets specific classes while maintaining high performance on the remaining categories.

2. Related Work

2.1. Continual Learning

Continual learning aims to enable models to acquire new knowledge without forgetting previously learned information [37]. It is a learning paradigm that is particularly applicable in dynamic and changing scenarios. Researchers have designed three strategies to achieve this goal, including rehearsal-based methods [11, 32, 39, 48, 60, 63, 69, 93, 94], regularization-based methods [2, 37, 43, 65, 92], and structure-based methods [1, 18, 47, 51, 64, 87]. These three strategies for continual learning are frequently combined to improve performance [48, 54, 60, 95].

Our proposed GS-LoRA falls into the category of structure-based methods. However, our problem differs from continual learning as we aim to continuously delete, rather than add new knowledge to the model.

2.2. Machine Unlearning

Machine unlearning involves retraining or modifying machine learning models to diminish or eradicate the influence of previously acquired patterns or biases, aiming to enhance the models' fairness and safety [5, 6, 22, 52, 68, 81]. A lot of studies design unlearning algorithms on simple machine learning algorithms [3, 6, 13, 33, 49, 70]. As a result, the applicability of these algorithms is constrained. Initial work on forgetting in deep learning either slices the data and trains a series of submodels to isolate the effect of specific data points on the model [5, 68, 82] (exact unlearning) or calculates influence functions to approximate the impact of a data item on the parameters of models [23, 26, 34, 66] (approximate unlearning). However, these methods deteriorate when applied to larger datasets and models, and the computational cost is exceedingly high.

Our problem focuses on the continual forgetting of a pre-trained model. One previous work [68] studies the continual exact unlearning by adding a class-specific synthetic signal in the pre-training stage. It should be noted that specific designs cannot be performed in the pre-training process, which is not common in deep learning applications. Cha *et al.* [9] mentions instance-wise forgetting and its continual form, while our setting is at category-level.

2.3. Parameter-Efficient Fine-Tuning

Training large models by self-supervised learning and then fine-tuning them on downstream tasks has become a new paradigm of deep learning [7, 28, 29, 41, 53, 57, 58, 75–77]. Parameter-efficient fine-tuning (PEFT) techniques [12, 30, 31, 35, 42, 89, 91] are proposed to optimize a limited number of parameters, as fully fine-tuning increasing large

models [7, 36, 57, 78] becomes less practical for various downstream tasks.

Recent studies focus on three different types of PEFT methods, categorized based on the origin of trainable parameters. These methods include addition-based approaches [30, 42, 46], freezing-based techniques [40, 59], and parameter-factorization-based methods [10, 31, 71]. All these methods are designed to improve the performance of downstream tasks, while our method modifies pre-trained models with the help of PEFT.

3. Problem Setting

We propose a new problem termed continual forgetting, which involves the selective removal of specific knowledge from a pre-trained model while preserving the performance of the rest. In this section, we first consider the simplest situation where there is only one task that needs to be forgotten, and later extend to a continual form.

Let $\mathcal{M}$ be a model pre-trained on the dataset D, we denote the mapping relationship of the model as $f_M : \mathcal{X}_D \to \mathcal{Y}_D$, where $\mathcal{X}_D$ and $\mathcal{Y}_D$ represent the input set and output set, respectively. Our objective is to selectively discard certain knowledge in the model while retaining the rest. Let D_f and D_r represent datasets containing knowledge to be forgotten and retained. Given that $|D_r|$ is typically large in practical scenarios and the retraining process is time-consuming, we require $|D_r| + |D_f| \ll |D|$. Before forgetting, model $\mathcal{M}$ performs well on both D_f and D_r, *i.e.*,

$$f_M : \mathcal{X}_{D_f} \xrightarrow{f_M} \mathcal{Y}_{D_f}, \mathcal{X}_{D_r} \xrightarrow{f_M} \mathcal{Y}_{D_r}. \qquad (1)$$

The forgetting algorithm $\mathscr{F}$ modifies the model to obtain $\mathcal{M}' = \mathscr{F}(\mathcal{M}, D_f, D_r)$ and a new mapping relationship $f_{M'}$ satisfying

$$f_{M'} : \mathcal{X}_{D_f} \xrightarrow{f_{M'}}\!\!\!\!\!\!\!/\;\; \mathcal{Y}_{D_f}, \mathcal{X}_{D_r} \xrightarrow{f_{M'}} \mathcal{Y}_{D_r}. \qquad (2)$$

Here, $\xrightarrow{f_{M'}}\!\!\!\!\!\!/\;$ means the mapping relationship no longer holds.

Now, we extend the problem to a continual form where the model is required to sequentially forget specific knowledge. Let $D_r = \{D_{r_t}\}$ and $D_f = \{D_{f_t}\}$ for $t = 1, 2, \cdots, T$ represent two sequences of datasets, where T is the number of forgetting tasks, $D_{f_t/r_t} = \{(x^i_{f_t/r_t}, y^i_{f_t/r_t})^{n_t}_{i=1}\}$ is the forgotten or retained dataset of the t-th task, $x^i_{f_t/r_t} \in \mathcal{X}_{f_t/r_t}$ is an input and $y^i_{f_t/r_t} \in \mathcal{Y}_{f_t/r_t}$ is the corresponding label. The forgetting algorithm $\mathscr{F}$ handles erase requests sequentially, starting from $\mathcal{M}$, and generates a sequence of models $\mathcal{M}_{f_1}, \mathcal{M}_{f_2}, \cdots, \mathcal{M}_{f_t}, \cdots, \mathcal{M}_{f_T}$, where $\mathcal{M}_{f_t}$ represents the modified model after the t-th forgetting task. After processing task $\mathcal{T}_t$, model $\mathcal{M}_{f_t}$ performs poorly on D_{f_i} but maintains the original performance in the remaining part,
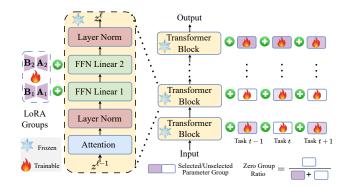

Figure 2. **Overall pipeline of GS-LoRA.** We incorporate a set of LoRA modules in each continual forgetting task and adopt a sparse structure selection strategy to achieve accurate and few modifications. All LoRA modules are added in the Linear layers of FFN in the Transformer blocks and we regard the LoRA modules in a Transformer block as one group. We use group sparse regularization to automatically select LoRA groups. The purple groups are selected to modify and the white groups are neglected. The pre-trained model (including Transformer blocks and other parts) is frozen and only LoRA groups are trainable.

i.e., the corresponding mapping relationship f_{M_t} holds

$$f_{M_t} : \mathcal{X}_{D_{f_i}} \xrightarrow{f_{M_t}}\!\!\!\!\!\!/\;\; \mathcal{Y}_{D_{f_i}}, \mathcal{X}_{D_{r_t}} \xrightarrow{f_{M_t}} \mathcal{Y}_{D_{r_t}}, \qquad (3)$$

where $i = 1, 2, \cdots, t, t = 1, 2, \cdots, T$.

4. Method

Preliminary: LoRA. Hu *et al.* [31] argue that the weight matrix in the pre-trained model has a very low intrinsic rank and utilizes a low-rank decomposition to implement parameter updates. For a weight matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$, it is updated following $\mathbf{W} = \mathbf{W} + \Delta\mathbf{W} = \mathbf{W} + \mathbf{BA}$, where $\mathbf{B} \in \mathbb{R}^{d \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times k}$ are low rank matrices and $r \ll min\{d, k\}$ is the rank of matrix $\mathbf{B}$ and $\mathbf{A}$. Only matrices with low ranks are trainable, while the matrix $\mathbf{W}$ remains frozen during training. LoRA can be added to the linear projection matrices in Multi-Head Attention modules or the Feed-Forward Network (FFN) modules in Transformer blocks.

4.1. Overview

Considering two key challenges in Sec. 1 and the optimization goal in Eq. (3), we propose Group Sparse LoRA (GS-LoRA) with selective forgetting loss and knowledge retention loss to achieve continual forgetting. Fig. 2 shows the overall pipeline of GS-LoRA. To achieve efficient forgetting, we use LoRA to fine-tune the FFN modules in Transformer blocks. To mitigate catastrophic forgetting of the remaining knowledge, smaller network changes are preferred [50, 51, 79, 87]. Therefore, we use group sparse regularization to select and modify fewer blocks. Sec. 4.2 gives a

more detailed description. To achieve the optimization goal in Eq. (3), we use a selective forgetting loss to maximize the original loss for the forgotten classes and a knowledge retention loss to minimize the loss for the remaining classes in Sec. 4.3.

4.2. GS-LoRA

LoRA Based Model Tuning. Following the findings of Geva *et al.* [21], FFN layers in the Transformer blocks store a substantial amount of knowledge, necessitating modification of the FFN modules to achieve knowledge erasure. Although directly modifying these layers is theoretically feasible, it is inefficient due to the large number of parameters in the FFN layers. To reduce the learnable parameters, we incorporate a set of LoRA modules to the FFN in each Transformer block and only make these LoRA modules trainable.

Suppose $\mathbf{x}$ is the input of the ℓ-th FFN module, the mathematical form can be expressed as:

$$\mathcal{FFN}^{(\ell)}(\mathbf{x}) = \max\left(\mathbf{0}, \mathbf{x}\mathbf{W}_1^{(\ell)} + \mathbf{b}_1^{(\ell)}\right)\mathbf{W}_2^{(\ell)} + \mathbf{b}_2^{(\ell)}, \quad (4)$$

where $\mathbf{W}_1^{(\ell)}, \mathbf{W}_2^{(\ell)}, \mathbf{b}_1^{(\ell)}, \mathbf{b}_2^{(\ell)}$ are the weights and biases of two fully connected layers from the pre-trained model, respectively. We use LoRA to only fine-tune the weights of FFN modules:

$$
\begin{aligned}
\mathbf{W}_t^{(\ell)} &= \begin{bmatrix} \mathbf{W}_{1t}^{(\ell)} \\ \mathbf{W}_{2t}^{(\ell)} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_1^{(\ell)} \\ \mathbf{W}_2^{(\ell)} \end{bmatrix} + \sum_{i=1}^{t} \mathbf{B}_i^{(\ell)} \mathbf{A}_i^{(\ell)}, \\
\mathbf{B}_i^{(\ell)} &= \begin{bmatrix} \mathbf{B}_{1i}^{(\ell)} & \mathbf{O} \\ \mathbf{O} & \mathbf{B}_{2i}^{(\ell)} \end{bmatrix}, \quad \mathbf{A}_i^{(\ell)} = \begin{bmatrix} \mathbf{A}_{1i}^{(\ell)} \\ \mathbf{A}_{2i}^{(\ell)} \end{bmatrix},
\end{aligned}
\quad (5)
$$

where $\mathbf{W}_{1t}^{(\ell)}$ and $\mathbf{W}_{2t}^{(\ell)}$ denote the weights of the ℓ-th FFN modules after task $\mathcal{T}_t$, and $\mathbf{B}_{1i}^{(\ell)}, \mathbf{A}_{1i}^{(\ell)}, \mathbf{B}_{2i}^{(\ell)}, \mathbf{A}_{2i}^{(\ell)}$ for $i = 1, 2, \cdots, t$ refer to the corresponding LoRA matrices in task $\mathcal{T}_i$. $\mathbf{O}$ is the zero matrix. Note that the output FFN layers are frozen to ensure forgetting occurs in the backbone and is difficult to recover. A detailed discussion can be found in Sec. 6.1.

Group Sparsity Selection. To mitigate catastrophic forgetting and achieve precise modifications automatically, we introduce a group sparsity selection strategy that enables the selection of fewer Transformer blocks. Although there are many ways to conduct a selection like routers [67, 86], meta learning [73], neural architecture search [14, 62], we utilize *group Lasso*, known for its simplicity and effectiveness in selecting parameters for specific groups [20, 45, 80, 85] while setting others to zero. Suppose LoRA matrices added to the ℓ-th Transformer block in task $\mathcal{T}_t$ are $\mathbf{B}_{1t}^{(\ell)}, \mathbf{A}_{1t}^{(\ell)}, \mathbf{B}_{2t}^{(\ell)}, \mathbf{A}_{2t}^{(\ell)}$. Then the optimization goal with group sparse regularization can be expressed as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{data} + \alpha \mathcal{L}_{strcuture}. \quad (6)$$

Here, $\mathcal{L}_{data}$ denotes the loss on data, which will be elaborated in Sec. 4.3, $\mathcal{L}_{structure}$ is the group sparse loss, and α serves as a hyperparameter to regulate the sparse intensity.

The group sparse loss on a set of weights can be represented as:

$$\mathcal{L}_{structure} = \sum_{\ell=1}^{G} \mathcal{L}_{gs}^{(\ell)}, \quad (7)$$

where G is the number of groups, $\mathcal{L}_{gs}^{(l)}$ is the group sparse loss of the ℓ-th group. We regard the LoRA weights in one Transformer block as a group. Therefore, the group sparse loss in the ℓ-th group can be written as:

$$\mathcal{L}_{gs}^{(l)} = \|\mathbf{B}_t^{(\ell)}\|_F + \|\mathbf{A}_t^{(\ell)}\|_F. \quad (8)$$

Here, $\|\cdot\|_F$ is the Frobenius norm of the LoRA matrices and t denotes task $\mathcal{T}_t$.

Sparsity Warmup. Deep learning models tend to converge to local minima in the landscape [56]. When a high sparsity constraint is imposed, the model's ability to escape local minima is hindered, thus preventing the realization of forgetting. However, achieving a sparse update necessitates a relatively large α. We adopt a warm-up strategy [27] to address this conflict. We utilize a stepwise α to achieve effective forgetting while ensuring a sparse modification. The mathematical expression can be written as:

$$\alpha_k = \begin{cases} 0, & k < K, \\ \alpha_K, & k \geq K. \end{cases} \quad (9)$$

Here, K is a hyperparameter. The model escapes the local minima in k epochs without structure loss and then performs group sparsification to obtain a sparse modification.

4.3. Loss Function

In this section, we will discuss the data loss in Eq. (6) and introduce selective forgetting loss and knowledge retention loss to handle our continual forgetting problem.

Selective Forgetting Loss. In each task $\mathcal{T}_t$ for $t = 1, 2, \cdots, T$, the model needs to forget the knowledge stored in data $D_{f_t} = (\mathcal{X}_{f_t}, \mathcal{Y}_{f_t})$. To achieve forgetting, the optimization goal is $\arg\max_{\mathbf{W}} \mathcal{L}\left(f_{M_{t-1}}(\mathcal{X}_{f_t}), \mathcal{Y}_{f_t}\right)$, where $\mathbf{W}$ is the parameter; $\mathcal{L}$ is the original loss function; $f_{M_{t-1}}$ is the mapping function obtained at the end of task $t - 1$. An intuitive idea is to perform a negative loss, *i.e.*, $\mathcal{L}_{forget} = -\mathcal{L}\left(f_{M_{t-1}}(\mathcal{X}_{f_t}), \mathcal{Y}_{f_t}\right)$. Nevertheless, simply adding a minus sign to the original loss leads to an exploding unbounded loss that is challenging to optimize. Therefore, we employ a ReLU function to introduce a lower bound following Du *et al.* [19], *i.e.*,

$$\mathcal{L}_{forget} = \text{ReLU}\left(\text{BND} - \mathcal{L}\left(f_{M_{t-1}}(\mathcal{X}_{f_t}), \mathcal{Y}_{f_t}\right)\right), \quad (10)$$

where BND is a hyperparameter that determines the bound.

4

Methods	Tunable Ratio ↓	100-5			100-10			100-50			100-90		
		$H \uparrow$	$Acc_r \uparrow$	$Acc_f \downarrow$	$H \uparrow$	$Acc_r \uparrow$	$Acc_f \downarrow$	$H \uparrow$	$Acc_r \uparrow$	$Acc_f \downarrow$	$H \uparrow$	$Acc_r \uparrow$	$Acc_f \downarrow$
Pre-train	-	-	70.2	74.5	-	74.4	73.8	-	74.8	74.0	-	73.8	74.6
L2*	99.73%	67.7	67.8	2.6	67.0	65.0	4.5	63.4	55.3	0.6	53.8	42.2	0.2
EWC* [37]	99.73%	69.0	68.8	1.0	69.2	67.4	2.6	60.9	51.4	0.1	46.3	33.6	0.2
MAS* [2]	99.73%	68.5	69.2	2.4	68.5	66.7	3.4	59.9	50.0	0.1	42.8	30.0	0.1
LwF [20]	99.73%	67.0	67.0	3.1	68.2	65.1	2.1	64.0	56.1	0.3	50.6	38.4	0.2
DER [8]	99.73%	66.4	67.4	4.8	67.9	67.2	5.1	61.3	52.0	0.3	54.0	42.5	0.2
DER++ [8]	99.73%	67.1	66.9	2.9	68.6	67.3	3.9	63.0	54.8	0.6	64.3	57.0	0.6
FDR [4]	99.73%	67.2	69.5	5.0	68.5	67.4	4.2	65.9	59.3	0.5	55.8	44.9	0.5
SCRUB [38]	99.73%	67.0	65.5	1.7	69.2	66.5	1.7	0.0	0.0	0.0	18.2	10.4	**0.0**
SCRUB-S [38]	99.73%	68.6	**71.8**	4.5	68.9	**71.9**	7.7	54.8	63.1	26.4	19.0	10.9	**0.0**
LIRF* [83]	50.66%	28.7	62.6	51.6	26.3	63.3	57.2	46.1	54.2	34.7	46.9	34.9	2.7
Retrain	100.00%	13.2	7.3	**0.0**	16.2	9.1	**0.7**	13.2	7.3	**0.0**	9.5	5.1	**0.0**
GS-LoRA	**1.28%**	**69.3**	70.5	1.9	**71.4**	71.1	2.0	**71.9**	69.9	0.8	**72.2**	**70.5**	0.5

Table 1. **Single-step forgetting results for face recognition.** Acc_r and Acc_f are the accuracies of remaining and forgotten classes. * denotes the original methods with a rehearsal buffer. Note that "retrain" represents retraining the model using replay data and *the training epoch is the same as other methods to ensure a fair comparison.* Pre-train denotes the results before forgetting. All setting is in the form of 100-Y, which means all experiments start from a pre-trained model (100 classes originally) and forget Y classes.

Methods	Tunable Ratio ↓	80-1			80-5			80-40			80-70		
		$H \uparrow$	$AP_r \uparrow$	$AP_f \downarrow$	$H \uparrow$	$AP_r \uparrow$	$AP_f \downarrow$	$H \uparrow$	$AP_r \uparrow$	$AP_f \downarrow$	$H \uparrow$	$AP_r \uparrow$	$AP_f \downarrow$
Pre-train	-	-	44.3	57.1	-	44.8	41.3	-	44.8	44.6	-	45.0	44.6
L2*	99.61%	25.6	35.6	37.1	27.7	34.9	18.4	27.9	32.3	20.0	29.9	34.9	18.4
EWC* [37]	99.61%	37.6	32.6	12.5	31.7	33.4	11.2	33.0	31.7	10.2	33.6	29.5	5.7
MAS* [2]	99.61%	39.4	32.5	6.9	27.9	30.4	15.4	31.1	30.4	12.8	31.6	28.6	9.3
Retrain	100.00%	46.6	39.3	**0.0**	40.3	39.6	**0.2**	40.5	39.2	**2.6**	37.8	39.7	8.6
GS-LoRA	**0.62%**	**49.9**	**44.5**	0.4	**42.4**	**45.0**	1.2	**41.6**	**42.8**	4.1	**43.7**	**43.6**	**0.9**

Table 2. **Single-step forgetting results for object detection** on the COCO dataset. AP_r and AP_f denotes the AP of remaining classes and forgotten classes. All setting is in the form of 80-Y, which means all experiments start from a pre-trained model and forget Y classes.

Knowledge Retention Loss. Besides forgetting selected knowledge, it is crucial for the model to maintain performance on the rest. Catastrophic forgetting on remaining classes [37] still exists. To mitigate this issue, we employ a small rehearsal buffer $D_{r_t} = (\mathcal{X}_{r_t}, \mathcal{Y}_{r_t})$ which satisfies $|D_{r_t}| + |D_{f_t}| \ll |D|$ to alleviate this undesirable forgetting and maintain efficient training. The knowledge retention loss can be written as:

$$\mathcal{L}_{retain} = \mathcal{L}\left(f_{M_{t-1}}\left(\mathcal{X}_{r_t}\right), \mathcal{Y}_{r_t}\right). \tag{11}$$

Combining Eqs. (10) and (11), we get the data loss

$$\mathcal{L}_{data} = \mathcal{L}_{retain} + \beta\mathcal{L}_{forget}, \tag{12}$$

where β is a hyperparameter.

5. Experiments

5.1. Experimental Setup

Datasets and Pre-trained Models. We evaluate the effectiveness and efficiency of GS-LoRA using published Transformer-based models in face recognition tasks and object detection tasks. More experiments on image classification can be found in the *Supplementary Material.* For the face recognition task, we constructed a subdataset called CASIA-Face100 which collects 100 face IDs from the CASIA-WebFace [84] dataset. We use a Face Transformer [90] pre-trained on the CASIA-Face100 dataset. For the object detection task, we use a deformable DETR [96] pre-trained on the COCO 2017 [44] dataset.

Metrics. We need to evaluate the performance of the forgotten classes and the retained classes. We use the average accuracy (Acc) for classification tasks and the mean average precision (AP) for object detection tasks. Ideally, the forgotten classes' performance should approach zero, and the remaining classes' performance should align with the original model's. Similar to Shibata *et al.* [68], we define H-Mean to evaluate the overall performance after learning task $\mathcal{T}_t$, which is computed by:

$$H\text{-}Mean^{(t)} = \frac{2Acc_r^{(t)} \cdot Drop^{(t)}}{Acc_r^{(t)} + Drop^{(t)}}. \tag{13}$$

5

Methods	100-20			80-20				60-20				40-20			
	$H\uparrow$	$Acc_r\uparrow$	$Acc_f\downarrow$	$H\uparrow$	$Acc_r\uparrow$	$Acc_f\downarrow$	$Acc_o\downarrow$	$H\uparrow$	$Acc_r\uparrow$	$Acc_f\downarrow$	$Acc_o\downarrow$	$H\uparrow$	$Acc_r\uparrow$	$Acc_f\downarrow$	$Acc_o\downarrow$
Pre-train	-	74.6	74.6	-	72.9	70.9	-	-	71.9	69.7	-	-	72.7	71.3	-
L2*	66.7	61.9	2.3	63.2	60.9	5.1	9.4	63.8	60.3	2.2	10.1	62.3	56.7	2.2	6.8
EWC* [37]	66.9	61.0	0.4	66.0	62.9	1.5	**0.0**	66.2	63.9	1.2	**0.0**	64.8	59.7	0.5	**0.0**
MAS* [2]	66.6	60.7	0.7	65.4	61.8	1.6	**0.0**	66.1	63.5	0.8	**0.0**	64.2	58.6	**0.3**	**0.0**
LwF [20]	66.2	60.9	2.1	64.6	60.8	2.1	0.5	64.9	61.4	1.4	**0.0**	65.0	60.7	1.4	**0.0**
DER [8]	66.7	62.7	3.4	63.3	59.8	3.7	**0.0**	63.8	60.2	2.0	**0.0**	62.7	57.2	2.0	**0.0**
DER++ [8]	66.1	62.8	4.8	63.8	61.7	5.0	**0.0**	64.4	61.6	2.4	**0.0**	65.0	61.8	2.7	**0.0**
FDR [4]	64.4	59.3	4.1	62.2	58.0	3.9	**0.0**	65.0	62.8	2.4	**0.0**	65.7	62.6	2.3	**0.0**
SCRUB [38]	67.8	63.1	1.3	66.3	64.4	2.6	**0.0**	66.7	64.5	0.8	**0.0**	68.4	66.9	1.5	**0.0**
SCRUB-S [38]	71.2	69.6	1.7	**69.1**	70.4	3.0	9.0	**70.4**	71.9	0.8	6.4	70.1	70.1	1.1	2.3
LIRF* [83]	28.6	60.1	55.8	28.3	58.5	52.2	43.4	35.8	56.1	43.5	33.5	36.8	59.8	44.7	22.1
Retrain	18.4	10.5	**0.3**	16.0	9.1	0.8	**0.0**	16.9	9.6	**0.0**	**0.0**	23.4	14.0	0.5	**0.0**
GS-LoRA	**71.6**	**72.1**	3.5	68.4	**71.1**	4.9	**0.0**	69.7	**72.0**	2.2	**0.0**	**70.2**	**71.0**	1.8	**0.0**

Table 3. **Continual forgetting results for face recognition.** Acc_o is the accuracy of old tasks, *i.e.*, the accuracy on all previously forgotten classes in task $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{t-1}$. There are 4 tasks in total and 20 classes are forgotten in each task.
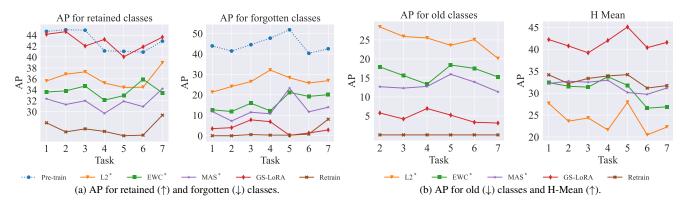

(a) AP for retained ($\uparrow$) and forgotten ($\downarrow$) classes.
(b) AP for old ($\downarrow$) classes and H-Mean ($\uparrow$).

Figure 3. **Comparative results on object detection for continual forgetting.** Pre-train (blue lines) means the performance before forgetting; methods with a * indicate the original methods with rehearsal buffer. "Retrain" (brown lines) refers to the process of retraining the model using replay data and *the training epoch is the same as other methods for a fair comparison*. The red line is our method. There are 7 tasks in total and 10 classes are forgotten in each task.

Here $Acc_r^{(t)}$ is calculated on the retained dataset after task $\mathcal{T}_t$ and $Drop^{(t)} = Acc_f^{(t-1)} - Acc_f^{(t)}$ is the performance drop on forgotten classes before and after training the task.[1] After learning task $\mathcal{T}_t$, we evaluate the performance on all previously forgotten classes in task $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{t-1}$.

5.2. Results and Comparisons

We compared GS-LoRA with *continual learning* methods including L2 regularization, EWC [37], MAS [2], LwF [43], DER [8], FDR [4], *machine unlearning* methods including LIRF [83], SCRUB [38] and retraining. Similar to GS-LoRA, we freeze the final FFN layer to ensure backbone forgetting. For the retraining method, we use replay data to train a randomly initialized model and *the training epoch is the same as other methods*.

Tabs. 1 and 2 show the performance comparisons with the aforementioned baselines for single-task forgetting, the degraded scenario in continual forgetting. The proposed GS-LoRA performs poorly in forgotten classes while retaining approximately the original performance in preserved classes. It is effective whether forgetting a small number of classes (*e.g.*, 1 class), or a large number of classes (*e.g.*, 90% of all the classes). Fig. 3 and Tab. 3 show the results for continual forgetting. For the object detection tasks, 10 classes are forgotten per task (7 tasks in total), while for the classification task, 20 classes are forgotten per task (4 tasks in total). GS-LoRA works the best among the listed methods, especially on object detection tasks. Besides, we can observe that in such a fast modification setting, severe underfitting occurs when using the retraining method.

---

[1] We take the classification problem as an example to define $H\text{-}Mean$. Replace Acc with AP for object detection tasks.
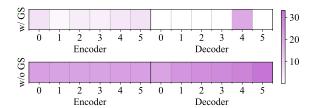
Figure 4. **Comparison of ℓ_2 norm of each LoRA group with or without group sparse loss.** Lighter colors mean smaller ℓ_2 norms which indicate less model modification. The first row shows the result with group sparse loss and the second row is the result of not using it (*i.e.* $\alpha = 0$).

5.3. Ablation Study

In this part, we conduct comprehensive ablation studies to analyze the effectiveness and efficiency of GS-LoRA. If not specified, the default configuration for deformable DETR includes six encoders and six decoders, and the Face Transformer utilizes six Transformer blocks. The rank we use in GS-LoRA is 8, and 10 classes are forgotten in face recognition and object detection tasks.

Group Sparsity Loss. We use group sparse regularization to achieve a sparser and more accurate modification. In Fig. 4, we illustrate each parameter group's ℓ_2 norm in the deformable DETR when forgetting one class. We can find that our GS-LoRA achieves comparable performance with directly using LoRA to fine-tune all FFNs (forget AP: 0.40 *vs.* 0, remain AP: 44.49 *vs.* 44.51) while requiring to modify significantly fewer parameters. Meanwhile, we can easily locate the knowledge more precisely with the help of the group sparsity selection strategy. The upper layers in the decoder contain more class-specific knowledge and need more modifications.

If the data of some remaining classes cannot be replayed, GS-LoRA can effectively reduce catastrophic forgetting in these classes. We conduct the following experiments on Face Transformer. Before forgetting, the model can identify 100 people and we want the model to forget 30 people. Fig. 5 shows the results when data of certain remaining classes are not available for replay. It is clear that GS-LoRA mitigates catastrophic forgetting on remaining classes.

Warm-up Sparsity. Illustrated by Tab. 4, we evaluate the efficacy of our warm-up sparsity strategy on Face Transformer. Without the warm-up sparsity, the model becomes trapped in the original local minima when $\alpha = 0.01$ or larger and fails to forget. Using a warm-up strategy, we can both achieve forgetting and easily control network sparsity. Moreover, adopting a larger α can dramatically increase the network sparsity.

Parameter Efficiency. By adjusting the rank of LoRA, we can easily control the learnable parameters. Here, we study how the rank of LoRA affects the performance when 10
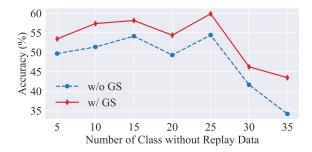

Figure 5. **Ablation study on group sparse (GS) regularization.** In this experiment, 30 classes are forgotten. Among the remaining 70 classes, only some classes can be replayed. The x-axis represents the number of classes without replay data, while the y-axis denotes the accuracy of these classes.

α	Warm-up	$Acc_f \downarrow$	$Acc_r \uparrow$	$H \uparrow$	Zero Group Ratio
Pre-train		73.78	74.63	-	-
0.01		73.78	74.63	0.00	1.00
0.01	✓	1.97	71.06	71.43	0.17
0.02		73.78	74.63	0.00	1.00
0.02	✓	0.70	69.99	71.50	0.50

Table 4. **Ablation study of warm-up sparsity** on Face Transformer. The zero group ratio is 1 means that all LoRA modules are not selected, *i.e.*, the parameters of the pre-trained model do not change. *Without sparsity warmup, forgetting failed when* $\alpha = 0.01$ *and* 0.02. Zero Group Ratio is defined as the number of zero groups divided by the number of all groups.

classes are forgotten in the deformable DETR model. The results in Tab. 5 reveal that a larger rank tends to achieve better performance, but it will also introduce more tunable parameters. The performance plateaus after rank goes beyond 8. Remarkably, forgetting can be achieved using only less than 1% of the parameters with a rank of 8. However, most continual learning methods need to modify nearly all parameters, posing inefficiency for large models.

Data Efficiency. One benefit of our efficient forgetting paradigm is that we only utilize a small amount of data. In practical scenarios, using too much data will dramatically increase training costs. We compare the performance with different data ratios in Tab. 6. Our approach demonstrates satisfactory performance even with minimal training data, which speeds up the forgetting process. Although performance has a marginal improvement with increased training data, the training time rises dramatically.

6. Discussion

6.1. Real Forgetting or Deceptive Forgetting?

When we want to forget some specific classes, the naive solution is to mask their output FFN weights directly, which

Rank	Tunable Ratio	$AP_f \downarrow$	$AP_r \uparrow$	$H \uparrow$
Pre-train	-	43.92	44.73	-
2	0.15%	7.60	44.45	37.98
4	0.31%	6.76	44.33	40.42
8	0.62%	2.89	43.63	42.28
16	1.23%	3.00	44.31	42.53
32	2.47%	3.26	44.32	42.40

Table 5. **Ablation study of the rank of LoRA modules.** Effective forgetting can be achieved by modifying less than 1% parameters.

Data Ratio	Speed	Acc_f	Acc_r	H	Zero Group Ratio
Pre-train	-	73.78	74.63	-	-
0.5	2×	0.93	71.34	72.09	0.33
0.2	5×	1.39	71.29	71.83	0.50
0.1	10×	1.97	71.06	71.43	0.17

Table 6. **Data efficiency comparison.** Data ratio means the ratio of data used for forgetting to data used for pre-training.

we refer to as "head forgetting" for simplicity. However, this trivial solution is *deceptive forgetting* and easy to be recovered. It's like a kid who knows the answer and deliberately does not say it. *Real forgetting* should occur at backbone and is difficult to be recovered.

We design the following experiment to demonstrate the significance of backbone forgetting. We load a model in which forgetting has occurred, freeze its backbone, and fine-tune the output FFN layer using data containing all classes. Then, we evaluate the model's performance on the forgotten and retained classes. Fig. 6 shows the classification accuracy curve with epoch when recovering. Compared to head forgetting, we can find that the model after forgetting via GS-LoRA can only be recovered to approximately 17% on forgotten classes, significantly lower than 70% achieved in head forgetting. Although it is possible to recover the accuracy of forgotten classes to a very low level in backbone forgetting, such recovery adversely impacts the accuracy of the remaining classes. Additionally, the recovery process for GS-LoRA needs more epochs while head forgetting can be recovered within 20 training epochs.

6.2. Scalability

We demonstrate the scalability of GS-LoRA in pre-trained models of different sizes. We first pre-train three Face Transformer models comprising 6 blocks, 12 blocks and 18 blocks. It should be noted that the size of our dataset is limited and slight overfitting occurs when there are 18 blocks. Then we use GS-LoRA to forget selective classes. As depicted in Tab. 7, GS-LoRA exhibits remarkable scalability, demonstrating effective performance across both large
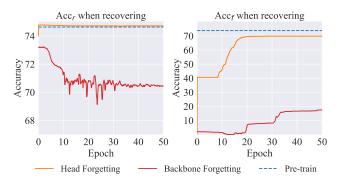

Figure 6. **Accuracy on forgotten classes and retained classes when recovering.** The blue line (Pre-train) is the result before forgetting. The orange line (Head Forgetting) is the trivial masking method. The red line (Backbone Forgetting) is the GS-LoRA.

# Blocks	# Param	$Acc_f \downarrow$		$Acc_r \uparrow$		$H \uparrow$	# Zero Groups
		Pre-train	Forget	Pre-train	Forget		
6	19M	73.78	1.97	74.63	71.06	71.43	1
12	38M	75.99	2.90	76.44	74.31	73.69	4
18	57M	73.43	2.55	73.50	71.20	71.03	6

Table 7. **The scalability of GS-LoRA** for three Face Transformer with different sizes. # means the number of and bf stands for before forgetting, which is the result of the pre-trained model.

and small models. Please refer to the *Supplementary Material* for visualization of group sparsity in each setting. Combined with small tunable parameters and high data efficiency, GS-LoRA can be a useful tool for privacy erasure in large models in practice.

7. Conclusion

This paper presents a new and practical problem called continual forgetting and proposes an efficient and effective method to solve it. For each continual forgetting task, we add a series of LoRA modules and only fine-tune them to achieve knowledge erasure. Additionally, we adopt a group sparse selection strategy to select specific LoRA groups, which can make the modification more accurate and sparser. Thorough experiments demonstrate that our method can achieve effective forgetting under various settings.

In the future, we aim to expand the applicability of our method to diverse domains, including large language models. We believe this paper will introduce an innovative and practical direction of continual learning to the community.

Supplementary Material

This document provides the supplementary materials that cannot fit into the main manuscript due to the page limit. Specifically, we first visualize the results on COCO dataset and the group sparsity in Sec. A. Next, we provide more implementation details for reproducibility in Sec. B. Finally, we provide more experiments in Sec. C.

A. Visualization

Visualization of detection results. We provide visualization results on COCO validation set before and after forgetting. Fig. S2 is the result of single-step forgetting and Fig. S3 is the result of continual forgetting. It is observed that GS-LoRA can achieve selective removal without affecting the remaining classes.

Visualization of parameter groups. To show the scalability of GS-LoRA, we evaluate it on Face Transformers with 6 layers, 12 layers and 18 layers in Tab. 7. We visualize the ℓ_2 norm of each LoRA group in these models in Fig. S1. It is observed that GS-LoRA achieves a sparse modification on models of different sizes and shows excellent scalability. Meanwhile, we can find that deeper layers in the Face Transformer contain more class-specific information.
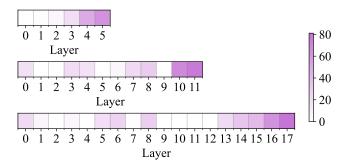

Figure S1. ℓ_2 **norm of each LoRA group in different Face Transformers.** The first row shows a Face Transformer with 6 layers. The second row shows a Face Transformer with 12 layers. The last row shows a Face Transformer with 18 layers. Lighter colors mean smaller ℓ_2 norms which indicate less modification.

B. Implementation Details

B.1. Face Recognition

Network Architecture. Face Transformer is proposed by Zhong *et al*. [90] who first uses Transformer architecture to solve face recognition tasks. A Face Transformer is a stack of Transformer blocks with a CosFace [74] classifier.

Config	Value
optimizer	AdamW
base learning rate	3e-4
learning rate schedule	cosine decay
minimal learning rate	1e-5
weight decay	0.05
momentum	$\beta_1, \beta_2 = 0.9, 0.999$
batch size	480
warm-up epochs	10
warm-up learning rate	1e-6
training epochs	1200
dropout rate	0.1

Table S1. Pre-training settings for Face Transformer.

Config	Value
optimizer	AdamW
base learning rate	1e-2
learning rate schedule	cosine decay
minimal learning rate	1e-5
weight decay	0.05
momentum	$\beta_1, \beta_2 = 0.9, 0.999$
batch size	48
warm-up epochs	0
training epochs	100
dropout rate	0.1
BND	110
K	20
β	0.15
α_K	0.01
LoRA rank	8
data ratio	0.1

Table S2. Forgetting settings for Face Transformer.

Pre-training. We pre-train a Face Transformer on CASIA-Face100 dataset for 1200 epochs. Implementation details can be found in Tab. S1.

Forgetting. For the forgetting process, we use 1337 as the random seed to generate a forgetting order. Implementation details can be found in Tab. S2 for all experimental settings, *which shows the robustness of GS-LoRA.* Here, BND is the bound in Eq. (10), K and α_K is the hyperparameter in Eq. (9), and β is the hyperparameter in Eq. (12). Note that "data ratio" is the ratio of data used for forgetting to data used for pre-training. To achieve fast model editing, the forgetting epoch is set to 100 and $0.1\times$ pre-training data is used. This is equivalent to fine-tuning the model using all pre-training data with only 10 epochs, which is *less than 1%* compared with 1200 epochs in the pre-training process.

B.2. Object Detection

Network Architecture. We use a Deformable DETR in object detection tasks. Deformable DETR has 3 parts: backbone, encoder and decoder. The backbone is an Im-
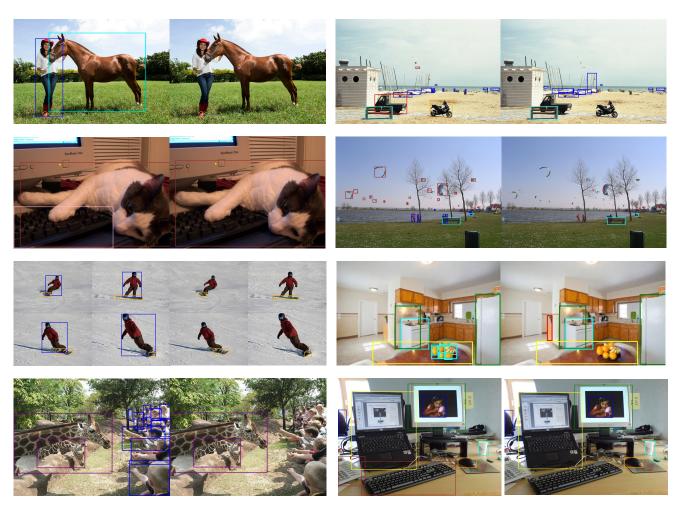
Figure S2. Visualization for **single-step forgetting**.

ageNet [16] pre-trained ResNet-50 [27]. The encoder and decoder both have 6 Transformer blocks using multi-head deformable attention.

Config	Value
optimizer	SGD
momentum	0.9
base learning rate	2e-4
weight decay	1e-4
batch size	16
training epochs	30
dropout rate	0.1
BND	15
β	0.2
α_K	3e-4
gradient clipping	0.1
LoRA rank	8
data ratio	0.1

Table S3. Forgetting settings for Deformable DETR.

Pre-training. We use the pre-trained model released by Zhu *et al*. [96], where Deformable DETR is trained on COCO 2017 training set for 50 epochs and reaches 43.8 AP on COCO 2017 validation set.

Forgetting. For the forgetting process, we first generate a random list using seed 123 following Liu *et al*. [48] to determine the forgetting order. Implementation details can be found in Tab. S3 when 40 classes are forgotten. For other experimental settings, hyperparameters are slightly different on β and the learning rate.

B.3. Baselines Implementation Details

B.3.1 Continual Learning Methods

We implement six continual learning methods to realize continual forgetting. Taking EWC as an example, we conduct it for forgetting as follows. First, we give randomly wrong labels to the forget set. Then, we use the remaining set to calculate the weight importance of EWC. Finally, we regard learning on the modified forget dataset as a new task
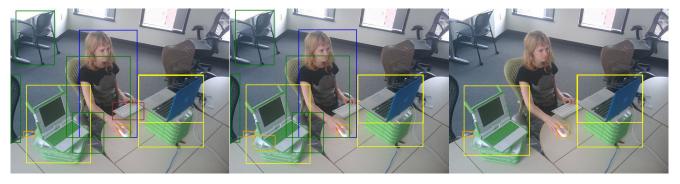
Figure S3. Visualization for **continual forgetting**. The left column shows the results from the pre-trained model. The middle column shows the results when "keyboard" (red bounding boxes in the left column) is erased. The left column shows the results when more objects (*e.g.*, person, book, chair) are erased.

and perform EWC algorithm. Note that we freeze the final FFN layer to ensure backbone forgetting. Additionally, for a fair comparison, we use the remaining set as a replay buffer to enhance the performance, which is denoted as EWC*.

B.3.2 Machine Unlearning Methods

Existing machine unlearning methods can be categorized into exact unlearning and approximate unlearning. Exacting unlearning needs to conduct specific designs in the pre-training process, however, we cannot modify the pre-training process in a continual forgetting setting. Initial studies on approximate unlearning are computationally heavy, *e.g.*, [23, 26, 66] need to calculate the Hessian matrix. These methods cannot be applied to large-scale problems, and we do not compare with them. We compare our

GS-LoRA with state-of-the-art LIRF [83], SCRUB [38] and SCRUB-S (a variant of SCRUB). LIRF [83] also uses a distillation strategy to realize the deposit and withdrawal of knowledge in a model. For a fair comparison, we add an additional replay buffer for LIRF.

SCRUB [38] uses distillation to realize efficient approximate unlearning. The training objective is:

$$
\begin{aligned}
\min_{w^u} \frac{\alpha}{N_r} & \sum_{x_r \in \mathcal{D}_r} d\left(x_r; w^u\right) \\
& + \frac{\gamma}{N_r} \sum_{(x_r, y_r) \in \mathcal{D}_r} \ell\left(f\left(x_r; w^u\right), y_r\right) \\
& - \frac{1}{N_f} \sum_{x_f \in \mathcal{D}_f} d\left(x_f; w^u\right),
\end{aligned}
\tag{S1}
$$

where $d\left(x; w^u\right) = D_{\mathrm{KL}}\left(p\left(f\left(x; w^o\right)\right) \| p\left(f\left(x; w^u\right)\right)\right)$ is

Methods	100-20			80-20				60-20				40-20			
	$H\uparrow$	$Acc_r\uparrow$	$Acc_f\downarrow$	$H\uparrow$	$Acc_r\uparrow$	$Acc_f\downarrow$	$Acc_o\downarrow$	$H\uparrow$	$Acc_r\uparrow$	$Acc_f\downarrow$	$Acc_o\downarrow$	$H\uparrow$	$Acc_r\uparrow$	$Acc_f\downarrow$	$Acc_o\downarrow$
Pre-train	-	68.0	60.5	-	64.1	60.9	-	-	69.2	63.3	-	-	76.5	70.7	-
L2*	47.6	**64.9**	22.9	47.6	68.0	24.3	30.0	40.0	70.1	35.3	24.5	52.7	74.7	29.9	23.5
EWC* [37]	54.1	**64.9**	14.2	55.6	64.9	12.2	13.9	53.8	72.7	20.7	7.8	62.0	80.0	20.1	9.1
MAS* [2]	54.0	**64.9**	14.2	57.1	**69.0**	12.2	13.7	54.0	72.7	20.4	7.7	62.2	80.2	19.8	8.8
Retrain	22.1	13.6	**1.0**	33.0	22.7	**0.7**	**0.0**	41.1	31.0	**2.2**	**0.0**	54.7	46.1	**3.4**	**0.0**
GS-LoRA	**60.3**	63.0	2.6	**61.7**	66.9	3.6	0.8	**65.9**	**74.1**	4.0	0.5	**73.8**	**82.7**	4.1	**0.0**

Table S4. **Continual forgetting results for image classification.** Acc_o is the accuracy of old tasks, *i.e.*, the accuracy on all previously forgotten classes in task $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{t-1}$. There are 4 tasks in total and 20 classes are forgotten in each task.

the KL-divergence between the student (w_u) and teacher (w_o) output distributions for example x, w_o is the weight of a pre-trained model (teacher) and w_u is the student. x_r and x_f are the retained set and forgotten set which contain N_r and N_f samples, respectively. ℓ is the cross-entropy loss and α and γ are hyperparameters.

However, Kurmanji *et al.* [38] find that directly optimizing Eq. (S1) is challenging and utilize a min-max optimization method following GAN [25]. To further improve the performance, we adopt a smoothing optimization method [88] as a variant of SCRUB and name it SCRUB-S.

Tabs. 1 and 3 show the results in single-step forgetting and continual forgetting settings. It is observed that LIRF cannot realize effective forgetting under our fast model erasure setting, which is data-inefficient. SCRUB and SCRUB-S can achieve forgetting when a small number of classes need to be deleted, but GS-LoRA achieves better overall performance (H-Mean). When we want to delete a large number of classes, *only GS-LoRA can achieve complete forgetting while maintaining the performance of the rest.* In a continual forgetting setting, SCRUB-S can achieve comparable performance with GS-LoRA, but the accuracy on previously forgotten classes (Acc_o) is a little bit high in SCRUB-S, which is undesirable. In summary, the data efficiency, parameter efficiency and effectiveness of GS-LoRA make it the most applicable in real-world scenarios.

C. More Experiments

In this section, we conduct more experiments to verify the effectiveness and efficiency of GS-LoRA. In Sec. C.1, we perform GS-LoRA in image classification tasks. In Sec. C.2, we conduct ablation studies on β in the loss function. In Sec. C.4, we perform more experiments when the replay buffer is incomplete and compare GS-LoRA with continual learning baselines.

C.1. Experiments on Image Classification

To further demonstrate the universality of our method, we use GS-LoRA to realize continual forgetting on image classification tasks. We choose ImageNet100 [16] dataset and

a pre-trained ViT [17] model in S4, where *GS-LoRA still outperforms other baselines significantly.*

C.2. Ablations on β in Loss Function

Our data loss function is Eq. (12), where β controls the level of forgetting. We conduct ablation studies in S5 on face recognition tasks. It is amazing that we find GS-LoRA demonstrates excellent performance *across a wide range of* β, which shows the robustness of our method.

β	0.01	0.05	0.1	0.15	0.2	0.25	0.5	0.75	1	5	10	20
$H\uparrow$	60.9	71.5	71.6	**72.2**	71.8	72.0	71.5	72.0	71.8	70.1	70.5	66.3

Table S5. Ablation studies on β.

C.3. Different Grouping Strategies

By default, we regard two LoRA modules in a Transformer block as a group (see in Fig. 2). In this section, we explore the effect of using GS-LoRA with different grouping strategies. In the FFN module [72], there are two linear layers, each of which can add a LoRA module. And in a LoRA module [31], there are two low-rank matrices.

We consider three grouping strategies: *"Block"*, *"Module"* and *"Matrix"*. *"Block"* is the default setting. *"Module"* denotes each LoRA **module** is a group, resulting in twice the number of groups compared to the Transformer blocks. *"Matrix"* means each **matrix** in LoRA modules is a group and the number of groups is four times the number of Transformer blocks.

Grouping Strategy	$Acc_f\downarrow$	$Acc_r\uparrow$	$H\uparrow$	Zero Group Ratio
Block	1.97	71.06	71.43	0.17
Module	0.93	70.58	71.70	0.50
Matrix	1.51	70.36	71.30	0.58

Table S6. **Effect of grouping strategies.** The zero-group ratio goes up when a more detailed grouping strategy is used.

	$Acc_f \downarrow$	$Acc_r \uparrow$	$H \uparrow$	$Acc_r^{\dagger} \uparrow$
Pre-train	73.67	75.00	-	73.97
L2*	0.04	56.81	64.13	21.06
EWC*	0.04	55.42	63.24	14.56
MAS*	**0.00**	55.20	63.11	16.78
Retrain	**0.00**	14.84	24.70	5.65
LoRA	0.04	**70.07**	**71.81**	49.66
GS-LoRA	0.04	**70.07**	**71.81**	**55.43**

(a) No replay data in 5 of the 70 remaining categories.

	$Acc_f \downarrow$	$Acc_r \uparrow$	$H \uparrow$	$Acc_r^{\dagger} \uparrow$
Pre-train	73.67	75.11	-	75.99
L2*	0.04	55.61	63.36	31.12
EWC*	**0.00**	51.41	60.56	16.19
MAS*	**0.00**	51.71	60.77	17.63
Retrain	**0.00**	13.48	22.79	4.68
LoRA	**0.00**	67.57	70.49	51.35
GS-LoRA	0.04	**68.49**	**70.97**	**57.37**

(b) No replay data in 10 of the 70 remaining categories.

	$Acc_f \downarrow$	$Acc_r \uparrow$	$H \uparrow$	$Acc_r^{\dagger} \uparrow$
Pre-train	73.67	74.83	-	76.44
L2*	0.04	51.65	60.71	27.32
EWC*	**0.00**	45.86	56.53	8.31
MAS*	0.04	47.07	57.43	12.07
Retrain	**0.00**	9.12	16.23	0.13
LoRA	**0.00**	66.64	69.98	54.12
GS-LoRA	**0.00**	**66.85**	**70.09**	**58.14**

(c) No replay data in 15 of the 70 remaining categories.

	$Acc_f \downarrow$	$Acc_r \uparrow$	$H \uparrow$	$Acc_r^{\dagger} \uparrow$
Pre-train	73.67	74.79	-	76.66
L2*	0.00	48.67	58.62	24.32
EWC*	0.00	41.32	52.95	6.55
MAS*	0.00	41.56	53.14	7.64
Retrain	0.00	8.14	14.66	0.10
LoRA	0.00	63.83	68.40	49.27
GS-LoRA	0.00	**64.45**	**68.75**	**54.37**

(d) No replay data in 20 of the 70 remaining categories.

	$Acc_f \downarrow$	$Acc_r \uparrow$	$H \uparrow$	$Acc_r^{\dagger} \uparrow$
Pre-train	73.67	74.81	-	75.88
L2*	**0.00**	44.53	55.50	22.14
EWC*	**0.00**	38.28	50.38	7.73
MAS*	**0.00**	37.81	49.97	8.24
Retrain	**0.00**	8.03	14.48	0.17
LoRA	**0.00**	64.40	68.72	54.41
GS-LoRA	0.04	**66.12**	**69.68**	**59.87**

(e) No replay data in 25 of the 70 remaining categories.

	$Acc_f \downarrow$	$Acc_r \uparrow$	$H \uparrow$	$Acc_r^{\dagger} \uparrow$
Pre-train	73.67	74.89	-	74.70
L2*	0.00	40.86	52.56	18.77
EWC*	0.00	33.56	46.11	5.07
MAS*	0.00	35.82	48.20	8.21
Retrain	0.00	7.13	13.00	0.50
LoRA	0.00	58.10	64.96	41.61
GS-LoRA	0.00	**59.52**	**65.84**	**46.25**

(f) No replay data in 30 of the 70 remaining categories.

	$Acc_f \downarrow$	$Acc_r \uparrow$	$H \uparrow$	$Acc_r^{\dagger} \uparrow$
Pre-train	73.67	74.82	-	75.02
L2*	0.00	37.32	49.55	18.23
EWC*	0.00	29.66	42.30	4.24
MAS*	0.00	30.04	42.68	5.66
Retrain	0.00	5.67	10.53	0.09
LoRA	0.00	52.27	61.15	34.16
GS-LoRA	0.00	**56.47**	**63.93**	**43.46**

(g) No replay data in 35 of the 70 remaining categories.

	$Acc_f \downarrow$	$Acc_r \uparrow$	$H \uparrow$	$Acc_r^{\dagger} \uparrow$
Pre-train	73.67	75.04	-	74.70
L2*	0.00	33.29	45.86	16.06
EWC*	0.00	25.08	37.41	4.15
MAS*	0.00	25.22	37.57	4.67
Retrain	0.00	6.02	11.13	0.30
LoRA	0.00	58.24	65.05	46.48
GS-LoRA	0.00	**62.84**	**67.83**	**54.29**

(h) No replay data in 40 of the 70 remaining categories.

Table S7. **Experiment results with incomplete replay.** Thirty classes are forgotten in all experiments. In the remaining 70 classes, only some classes can be replayed. Each subtable shows the results with a different number of replay classes. Pre-train denotes the results before forgetting. L2*, MAS* and EWC* denote the original methods with a rehearsal buffer. LoRA denotes using LoRA to fine-tune FFN modules in Transformer blocks without group sparse. Our method (GS-LoRA) is highlighted in color. We specifically evaluate the accuracy of the classes without replay samples and report it as $Acc_r^{\dagger}$.

We conduct our experiments on a Face Transformer and all the experiments are performed at the same sparse intensity, *i.e.* α is the same. Tab. S6 shows the results of different grouping strategies. Notably, all grouping strategies yield exceptional performance. It is observed that with a more detailed grouping strategy, the zero-group ratio increases,

which makes sense because each group has fewer parameters and more flexibility.

C.4. More Experiments with Incomplete Replay

In Sec. 5.3, we consider the situation where we cannot obtain the replay data of some classes. In this section, we

conduct more experiments with incomplete replay data on other continual learning baselines. We keep our settings the same as Fig. 5, where 30 classes need to be forgotten. In the remaining 70 classes, some classes cannot be replayed. Tab. S7 shows the results when 5, 10, 15, 20, 25, 30, 35 and 40 classes cannot be replayed. We can find that GS-LoRA mitigates catastrophic forgetting to some extent and achieves the best performance among all listed baselines.

References

[1] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375, 2017. 2

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018. 2, 5, 6, 12

[3] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine unlearning: Linear filtration for logit-based classifiers. *Machine Learning*, 111(9):3203–3226, 2022. 2

[4] Ari S Benjamin, David Rolnick, and Konrad Kording. Measuring and regularizing networks in function space. *arXiv preprint arXiv:1805.08289*, 2018. 5, 6

[5] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021. 2

[6] Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *International Conference on Machine Learning*, pages 1092–1104. PMLR, 2021. 2

[7] TomB. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Thomas Henighan, Rewon Child, Aditya Ramesh, DanielM. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Samuel McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv: Computation and Language,arXiv: Computation and Language*, 2020. 2, 3

[8] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020. 5, 6

[9] Sungmin Cha, Sungjun Cho, Dasol Hwang, Honglak Lee, Taesup Moon, and Moontae Lee. Learning to unlearn: Instance-wise unlearning for pre-trained classifiers. *arXiv preprint arXiv:2301.11578*, 2023. 2

[10] Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. One-for-all: Generalized lora for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967*, 2023. 3

[11] Jingfan Chen, Yuxi Wang, Pengfei Wang, Xiao Chen, Zhaoxiang Zhang, Zhen Lei, and Qing Li. Diffusepast: Diffusion-based generative replay for class incremental semantic segmentation. *arXiv preprint arXiv:2308.01127*, 2023. 2

[12] Tianrun Chen, Lanyun Zhu, Chaotao Ding, Runlong Cao, Yan Wang, Zejian Li, Lingyun Sun, Papa Mao, and Ying Zang. Sam fails to segment anything? – sam-adapter: Adapting sam in underperformed scenes: Camouflage, shadow, medical image segmentation, and more. 2023. 2

[13] Yuantao Chen, Jie Xiong, Weihong Xu, and Jingwen Zuo. A novel online incremental and decremental learning algorithm based on variable support vector machine. *Cluster Computing*, 22:7435–7445, 2019. 2

[14] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. *Advances in neural information processing systems*, 32, 2019. 4

[15] Kate Crawford and Trevor Paglen. Excavating ai: The politics of images in machine learning training sets. *Ai & Society*, 36(4):1105–1116, 2021. 1

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 10, 12

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 12

[18] Arthur Douillard, Alexandre Rame, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[19] Min Du, Zhi Chen, Chang Liu, Rajvardhan Oak, and Dawn Song. Lifelong anomaly detection through unlearning. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1283–1297, 2019. 4

[20] Jiashi Feng and Trevor Darrell. Learning the structure of deep convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2749–2757, 2015. 4, 5, 6

[21] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020. 2, 4

[22] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019. 2

[23] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020. 2, 11

[24] Eric Goldman. An introduction to the california consumer privacy act (ccpa). *Santa Clara Univ. Legal Studies Research Paper*, 2020. 1

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 12

[26] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019. 2, 11

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 10

[28] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 2

[29] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[30] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019. 2, 3

[31] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 2, 3, 12

[32] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2

[33] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR, 2021. 2

[34] Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. *arXiv preprint arXiv:2210.01504*, 2022. 2

[35] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, pages 709–727. Springer, 2022. 2

[36] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 3

[37] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 2, 5, 6, 12

[38] Meghdad Kurmanji, Peter Triantafillou, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in neural information processing systems*, 2023. 5, 6, 11, 12

[39] Frantzeska Lavda, Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. Continual classification learning using generative models. *arXiv preprint arXiv:1810.10612*, 2018. 2

[40] Jaejun Lee, Raphael Tang, and Jimmy Lin. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*, 2019. 3

[41] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023. 2

[42] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 2, 3

[43] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 2, 6

[44] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 5

[45] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 806–814, 2015. 4

[46] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021. 3

[47] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 2544–2553, 2021. 2

[48] Yaoyao Liu, Bernt Schiele, Andrea Vedaldi, and Christian Rupprecht. Continual detection transformer for incremental object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23799–23808, 2023. 2, 10

[49] Ananth Mahadevan and Michael Mathioudakis. Certifiable machine unlearning for linear models. *arXiv preprint arXiv:2106.15093*, 2021. 2

[50] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2, 3

[51] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pages 67–82, 2018. 2, 3

[52] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022. 2

[53] Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. Expanding language-image pretrained models for general video recognition. In *European Conference on Computer Vision*, pages 1–18. Springer, 2022. 2

[54] Bolin Ni, Hongbo Zhao, Chenghao Zhang, Ke Hu, Gaofeng Meng, Zhaoxiang Zhang, and Shiming Xiang. Enhancing visual continual learning with language-guided supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24068–24077, 2024. 2

[55] Trevor Paglen. Imagenet roulette – trevor paglen. https://paglen.studio/2020/04/29/imagenet-roulette/, 2020. 1

[56] Tomaso Poggio and Qianli Liao. *Theory II: Landscape of the empirical risk in deep learning*. PhD thesis, Center for Brains, Minds and Machines (CBMM), arXiv, 2017. 4

[57] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2, 3

[58] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2

[59] Shauli Ravfogel, Elad Ben-Zaken, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked languagemodels. *arXiv preprint arXiv:2106.10199*, 2021. 3

[60] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 2

[61] General Data Protection Regulation. General data protection regulation (gdpr). *Intersoft Consulting, Accessed in October*, 24(1), 2018. 1

[62] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021. 4

[63] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[64] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[65] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pages 4528–4537. PMLR, 2018. 2

[66] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021. 2, 11

[67] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 4

[68] Takashi Shibata, Go Irie, Daiki Ikami, and Yu Mitsuzumi. Learning with selective forgetting. In *IJCAI*, page 4, 2021. 2, 5

[69] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017. 2

[70] Nan Sun, Ning Wang, Zhigang Wang, Jie Nie, Zhiqiang Wei, Peishun Liu, Xiaodong Wang, and Haipeng Qu. Lazy machine unlearning strategy for random forests. In *International Conference on Web Information Systems and Applications*, pages 383–390. Springer, 2023. 2

[71] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter efficient tuning of pretrained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*, 2022. 3

[72] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 12

[73] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18: 77–95, 2002. 4

[74] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274, 2018. 9

[75] Haochen Wang, Junsong Fan, Yuxi Wang, Kaiyou Song, Tiancai Wang, Xiangyu Zhang, and Zhaoxiang Zhang. Bootstrap masked visual modeling via hard patches mining. *arXiv preprint arXiv:2312.13714*, 2023. 2

[76] Haochen Wang, Junsong Fan, Yuxi Wang, Kaiyou Song, Tong Wang, and Zhaoxiang Zhang. Droppos: Pre-training vision transformers by reconstructing dropped positions. *Advances in Neural Information Processing Systems*, 2023.

[77] Haochen Wang, Kaiyou Song, Junsong Fan, Yuxi Wang, Jin Xie, and Zhaoxiang Zhang. Hard patches mining for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10375–10385, 2023. 2

[78] Xiao Wang, Guangyao Chen, Guangwu Qian, Pengcheng Gao, Xiao-Yong Wei, Yaowei Wang, Yonghong Tian, and Wen Gao. Large-scale multi-modal pre-trained models: A comprehensive survey. *Machine Intelligence Research*, 20 (4):447–482, 2023. 3

[79] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR22*, 2022. 2, 3

[80] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016. 4

[81] Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. Machine unlearning: Solutions and challenges. *arXiv preprint arXiv:2308.07061*, 2023. 2

[82] Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. Arcane: An efficient architecture for exact machine unlearning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4006–4013, 2022. 2

[83] Jingwen Ye, Yifang Fu, Jie Song, Xingyi Yang, Songhua Liu, Xin Jin, Mingli Song, and Xinchao Wang. Learning with recoverable forgetting. In *ECCV*, 2022. 5, 6, 11

[84] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014. 5

[85] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(1):49–67, 2006. 4

[86] Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012. 4

[87] Chenghao Zhang, Kun Tian, Bin Fan, Gaofeng Meng, Zhaoxiang Zhang, and Chunhong Pan. Continual stereo matching of continuous driving scenes with growing architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18901–18910, 2022. 2, 3

[88] Jiawei Zhang, Peijun Xiao, Ruoyu Sun, and Zhiquan Luo. A single-loop smoothed gradient descent-ascent algorithm for nonconvex-concave min-max problems. *Advances in neural information processing systems*, 33:7377–7389, 2020. 12

[89] Renrui Zhang, Rongyao Fang, Peng Gao, Wei Zhang, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. 2

[90] Yaoyao Zhong and Weihong Deng. Face transformer for recognition. *arXiv preprint arXiv:2103.14803*, 2021. 5, 9

[91] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16825, 2022. 2

[92] Fei Zhu, Zhen Cheng, Xu-yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, 34:14306–14318, 2021. 2

[93] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880, 2021. 2

[94] Fei Zhu, Xu-Yao Zhang, Rui-Qi Wang, and Cheng-Lin Liu. Learning by seeing more classes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7477–7493, 2022. 2

[95] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-Lin Liu. Imitating the oracle: Towards calibrated model for class incremental learning. *Neural Networks*, 164:38–48, 2023. 2

[96] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 5, 10