SAMCL: Empowering SAM to Continually Learn from Dynamic Domains with Extreme Storage Efficiency

Zeqing Wang[1,2], **Kangye Ji**[1,3], **Di Wang**[1], **Haibin Zhang**[4], **Fei Cheng**[1*]
[1]School of Computer Science and Technology, Xidian University
[2]College of Design and Engineering, National University of Singapore
[3]Tsinghua Shenzhen International Graduate School, Tsinghua University
[4]School of Cyber Engineering, Xidian University
zeqing.wang@u.nus.edu, jky25@mails.tsinghua.edu.cn, wangdi@xidian.edu.cn,
hbzhang@mail.xidian.edu.cn, chengfei@xidian.edu.cn

Abstract

Segment Anything Model (SAM) struggles in open-world scenarios with diverse domains. In such settings, naive fine-tuning with a well-designed learning module is inadequate and often causes *catastrophic forgetting* issue when learning incrementally. To address this issue, we propose a novel continual learning (CL) method for SAM, termed SAMCL. Rather than relying on a fixed learning module, our method decomposes incremental knowledge into separate modules and trains a selector to choose the appropriate one during inference. However, this intuitive design introduces two key challenges: ensuring effective module learning and selection, and managing storage as tasks accumulate. To tackle these, we introduce two components: *AugModule* and *Module Selector*. *AugModule* reduces the storage of the popular LoRA learning module by sharing parameters across layers while maintaining accuracy. It also employs heatmaps—generated from point prompts—to further enhance domain adaptation with minimal additional cost. *Module Selector* leverages the observation that SAM's embeddings can effectively distinguish domains, enabling high selection accuracy by training on low-consumed embeddings instead of raw images. Experiments show that SAMCL outperforms state-of-the-art methods, achieving only 0.19% forgetting and at least 2.5% gain on unseen domains. Each *AugModule* requires just 0.233 MB, reducing storage by at least 24.3% over other fine-tuning approaches. The buffer storage for *Module Selector* is further reduced by up to $256\times$.

Code — https://github.com/INV-WZQ/SAMCL
Extended version — https://arxiv.org/abs/2412.05012

1 Introduction

Segment Anything Model (SAM) has achieved notable success in image segmentation (Kirillov et al. 2023). This success stems from its extensive pre-training on a closed-set dataset and its distinctive architecture that enables effective integration of image and prompt information. However, when deployed in open-world environments (Ji et al. 2024; Tang et al. 2025b), conventional SAM exhibits significant limitations in segmenting instances across diverse domains, including camouflaged, shadowed, and medical domains, thereby restricting its practical applicability.

Recent studies have identified this flaw and utilized fine-tuning strategies using carefully designed learning modules to enhance SAM's performance (Chen et al. 2023; Shaharabany et al. 2023). However, these modules remain fixed throughout the entire learning process, meaning that all additional knowledge is stored in predetermined locations. This design is inherently limited in real scenarios where the model encounters incremental data and needs continual learning (CL) manner (Yang et al. 2024). In such scenarios, fine-tuning on new data often results in significant forgetting of previous knowledge, leading to the issue known as *catastrophic forgetting* (McCloskey and Cohen 1989).

To address this issue, we propose a novel CL method for SAM, termed SAMCL. At a high level, SAMCL allocates different knowledge into distinct modules for learning, and trains an accurate selector that dynamically chooses the most appropriate module during inference. This design mitigates inter-task interference and effectively reduces forgetting.

While conceptually straightforward (Rusu et al. 2016; Wang et al. 2023a; Aljundi, Chakravarty, and Tuytelaars 2017), this approach faces two main challenges in the CL setting for SAM: (1) ensuring effective knowledge learning within each module and maintaining high selection accuracy, and (2) managing the increasing storage overhead from accumulating modules and the rising cost of training an accurate selector as tasks grow.

To overcome these challenges, we introduce an effective and efficient learning module, *AugModule*, and selector, *Module Selector*, for SAMCL.

Firstly, *AugModule*—comprising *SLoRA* and *Prompt Augmentation*—efficiently adapts SAM to a new domain. Unlike vanilla LoRA (Hu et al. 2022) which utilizes independent matrices A and B for each layer, *SLoRA* shares a single matrix A in LoRAs across all layers without compromising accuracy. *Prompt Augmentation*, which transforms point-type prompts into heatmaps, augments prompt utilization without retraining the mask decoder in SAM. Compared with learning modules in other fine-tuning methods, *AugModule* achieves high accuracy across all tested domains, with parameter storage costs of only 0.233 MB, resulting in at least 24.3% reduction.

*Corresponding Author.

Secondly, we propose a lightweight *Module Selector* consisting of only four linear layers to select the appropriate module during inference. We exploit a novel observation that embeddings in the image encoder can accurately differentiate between domains. This enables us to store only a few low-consumed embeddings in the buffer, which are $256\times$ lower than storing raw images (Yang et al. 2024; Buzzega et al. 2020; Chaudhry et al. 2019), for training *Module Selector* while maintaining high accuracy in module selection. This approach not only mitigates the forgetting problem but also enhances knowledge transfer to unseen domains.

Together, SAMCL empowers SAM with CL ability to learn from incremental and dynamic domains. Experimental results show that SAMCL limits average accuracy loss to just 0.19%—significantly lower than state-of-the-art methods—while achieving at least a 2.5% improvement in accuracy when transferring to unseen environments.

In summary, our main contributions are as follows:

- We design *AugModule* for SAM to effectively learn from any new domain with ultra-low storage cost.
- We store a small set of low-consumed embeddings in the buffer to train a lightweight *Module Selector*, leveraging the inherent ability of SAM to effectively distinguish between domains for module selection during inference.
- We design a new experimental setup that can comprehensively evaluate the performance of SAM during CL. Experiments show the excellent performance of SAMCL in learning, maintaining, and transferring knowledge.

2 Related Works

Segment Anything Model. SAM (Kirillov et al. 2023) demonstrates strong performance in image segmentation using interactive prompts. SAM2 (Ravi et al. 2024) extends this capability to video. In this paper, we focus on image segmentation. Due to their architectural similarity and SAM's popularity in fine-tuning, we adopt SAM as the base model to present our method. A detailed discussion of applying our method to SAM2 is included in the Appendix B.4.

Fine-tuning SAM. To address SAM's limitations in challenging domains such as camouflaged, shadow, and medical domains, a common strategy is to freeze parts of SAM and add extra modules for fine-tuning. OCRT (Tang et al. 2025b) extracts informative low-level and high-level image representations to facilitate adaptation to unseen domains. SAM-LST (Chai et al. 2023) appends a ResNet (He et al. 2016) to image encoder and trains mask decoder. AutoSAM (Shaharabany et al. 2023) replaces the prompt encoder with a new network while freezing the rest of SAM. SAM-Adapter (Chen et al. 2023) fine-tunes the mask decoder and uses task-specific knowledge to guide performance within the frozen image encoder. However, most of them overlook the need to handle dynamic environments rather than a single domain. Moreover, fine-tuning the mask decoder or adding large networks incurs significant storage overhead for each domain.

LoRA-type Fine-tuning. As a foundation model, the widely used LoRA technique (Hu et al. 2022) is applicable to SAM (Zhang and Liu 2023). LoRA employs two low-rank matrices, A and B, to fine-tune transformer blocks. This process can be articulated as follows:

$$Y_i = (W_i + B_i A_i) X_i, \tag{1}$$

where Y, X, W, A and B represent output, input, pre-trained weight, and matrices A and B in LoRA. Note that i represents i-th block in the pre-trained model.

While LoRA achieves strong performance with few parameters, AsymmLoRA (Zhu et al. 2024) further reduces redundancy by freezing matrix A across all LoRAs, lowering trainable parameters. However, it still incurs storage cost for matrix A. HydraLoRA (Tian et al. 2024) reduces storage by sharing a single matrix A across multiple LoRAs within a layer. In contrast, our method, *SLoRA*, uses one LoRA for each layer and shares matrix A across all layers, minimizing storage further. Although some similar works, such as ShareLoRA (Song et al. 2025), employ a comparable paradigm in the field of natural language processing, we broaden the application of this paradigm and confirm its efficacy in visual segmentation models, including SAM and SAM2. More importantly, we show that *Prompt Augmentation* offers a simple yet effective way to enhance the performance of *SLoRA*, without additional training on the mask decoder in SAM.

Continual Learning. CL methods are categorized into regularization, replay, and architecture-based approaches (Wang et al. 2023b). Regularization-based methods (Kirkpatrick et al. 2017; Li and Hoiem 2018; Wang et al. 2025) preserve prior knowledge by constraining important historical parameters. Most continual segmentation approaches (Lin, Wang, and Zhang 2022; Yuan, Zhao, and Shi 2024; Tang et al. 2025a, 2024) follow this paradigm, leveraging representation learning or knowledge distillation to reduce segmentation confusion (Yuan and Zhao 2024). Replay-based methods (Buzzega et al. 2020; Chaudhry et al. 2019) retain and replay past samples when training on new tasks. While effective, they incur substantial storage overhead due to the large buffer. Both of them typically rely on a fixed network, leading to entanglement between old and new knowledge.

To mitigate inter-task interference from shared parameters, architecture-based methods (Rusu et al. 2016; Mallya and Lazebnik 2018; Pietron et al. 2023, 2025) allocate separate modules per task. Expert Gate (Aljundi, Chakravarty, and Tuytelaars 2017) uses a pre-trained gate to select experts for task-agnostic inference but introduces high overhead. O-LoRA (Wang et al. 2023a) assigns tasks to orthogonal low-rank subspaces to reduce interference but fails to ensure strict orthogonality.

MoDA (Yang et al. 2024) uses extra tokens as selectors to choose the proper module for SAM. However, it requires storing raw samples to train selectors, incurring high storage costs. Additionally, when paired with encoder-finetuning methods, MoDA must run the image encoder twice during inference, increasing latency. In contrast, SAMCL avoids both issues by training on lightweight embeddings instead of raw images and eliminating redundant forward passes.

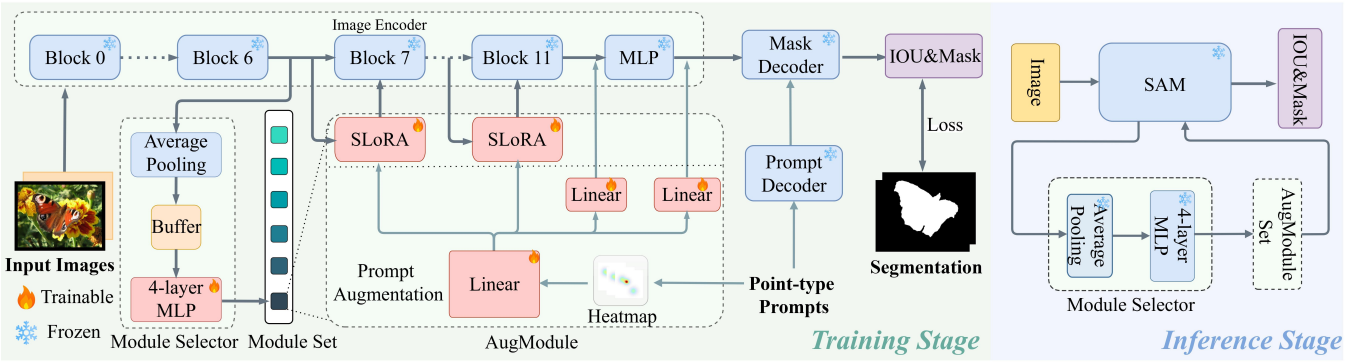

Figure 1: Overview of SAMCL with SAM. During training, SAMCL uses a new *AugModule* to learn from a new domain. All modules are stored in a module set. Meanwhile, *Module Selector* is trained on a few stored embeddings from the image encoder. During inference, SAMCL extracts latent embeddings from the image encoder to select an appropriate module by *Module Selector* and continual inference by SAM with the selected module. A detailed illustration of the inference process is provided in the Appendix B.1.

3 Preliminaries

Problem Formulation. In the CL setting, tasks $\{1, 2, .., T\}$ are executed sequentially with no access to previous data when training on a new task, except for replay data in replay-based methods (Buzzega et al. 2020). The t-th task is described as $\mathcal{D}_t = \{(x_t^i, y_t^i)\}_{i=1}^{n_t}$, where n_t represents the number of samples for the t-th task, and (x_t^i, y_t^i) denotes the input-label pair. For using SAM to segment an instance with the help of prompts, $x_t^i \in \mathbb{R}^{C\times H\times W}$ represents an image, and $y_t^i \in \{0,1\}^{H\times W}$ is the corresponding mask based on the prompt. θ_t denotes the model after training on the t-th task. The objective of the CL is to obtain the model θ_T that performs well on all seen domains from D_1 to D_T, while also exhibiting strong generalization to unseen domains.

Evaluation Metrics. To evaluate segmentation performance, we employ three common metrics: mean absolute error (mMAE), mean F1 score (mF1), and mean intersection over union (mIoU). We denote the accuracy of these metrics for the j-th task, after training on the i-th task, as $a_{i,j}$. To evaluate the CL performance in terms of learning, maintaining, and transferring knowledge, we follow previous works (Lopez-Paz and Ranzato 2017; Wang et al. 2023b) and define three key metrics as follows: (1) **Average Accuracy (AA)**, defined as $\text{AA} = \frac{1}{T}\sum_{i=1}^{T} a_{T,i}$, to evaluate the segmentation performance of all tasks after CL; (2) **Forgetting Measure (FM)**, defined as $\text{FM} = \frac{1}{T}\sum_{i=1}^{T} a_{i,i} - a_{T,i}$, to measure the forgetting degree after CL; (3) **Forward Transfer (FT)**, defined as $\text{FT} = \frac{1}{T-1}\sum_{i=1}^{T-1} a_{i,i+1}$, to evaluate the transferring ability to unseen domains during CL.

4 Method

SAMCL allocates different knowledge into distinct modules for learning, and trains an accurate selector to dynamically choose the most appropriate module during inference. The overview is shown in Figure 1. To ensure both effectiveness and storage efficiency for the learning module and the selector, we introduce two key components in this section: *AugModule* (Section 4.1) and *Module Selector* (Section 4.2).

4.1 AugModule

As shown in Figure 2, *AugModule* enhances SAM's segmentation with an efficient design. It refines LoRA for better storage efficiency without accuracy loss and incorporates prompt information to further improve performance. We first introduce *SLoRA*, then *Prompt Augmentation*.

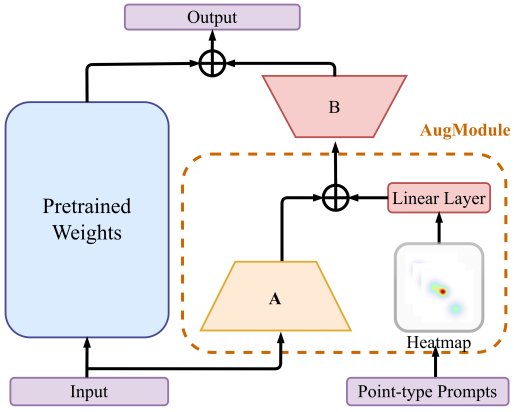

Figure 2: Illustration of *AugModule*, integrating *SLoRA* and *Prompt Augmentation*. *SLoRA* shares matrix A across all LoRAs for efficient adaptation. *Prompt Augmentation* converts point prompts into heatmaps, injected via linear transformation for dimensional alignment.

SLoRA LoRA achieves strong performance with few tunable parameters. AsymmLoRA (Zhu et al. 2024) further reduces them by freezing matrix A and fine-tuning only matrix B. This is based on the observation that A is more generalizable, extracting original features into low-rank representations, while B leverages these features to produce the desired output (Zhu et al. 2024).

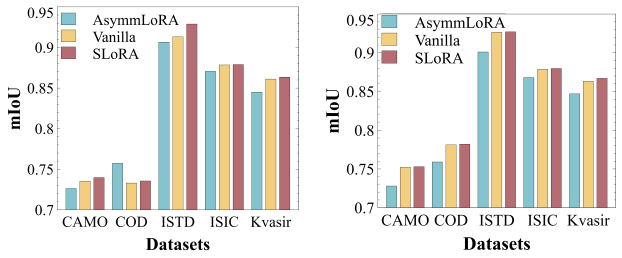

Figure 3: Comparison among AsymmLoRA, vanilla LoRA, and *SLoRA*. Each method fine-tunes SAM's image encoder on the CAMO, COD, ISTD, ISIC, and Kvasir datasets for 20 epochs. (a) shows results without *Prompt Augmentation*, and (b) shows results with *Prompt Augmentation*.

However, this design presents two challenges. *First*, AsymmLoRA still underperforms vanilla LoRA. As shown in Figure 3a, we fine-tune the full QKV projection using a single LoRA-type module per attention block. The results show that vanilla LoRA outperforms AsymmLoRA on most datasets and highlight the (albeit small) benefit of training matrix A. *Second*, as the number of tasks increases, per-task storage becomes a major bottleneck. Although matrix A is frozen during training, it still adds to the storage. Therefore, we aim to retain A's functionality while reducing its storage. To address these challenges, we inevitably raise an idea: **Given matrix A's generalization capacity, can all matrices Bs share a single A while fully leveraging its role?** This insight gives rise to *SLoRA*, defined as follows:

$$Y_i = (W_i + B_i A)X_i, \tag{2}$$

where A is shared across all B_i, and i denotes the index of fine-tuned blocks. As shown in Figure 3a, *SLoRA* outperforms both frozen and vanilla LoRA on most datasets. Since *SLoRA* uses only a single matrix A, its total parameter count is nearly half that of vanilla LoRA. Thus, the empirical results demonstrate that *SLoRA* is an effective and efficient learning module for SAM. Notably, AsymmLoRA performs relatively well on the COD dataset but struggles to integrate prompt information, which we discuss further later.

Prompt Augmentation Given SAM's architecture, prompts provide strong guidance for segmentation. Many fine-tuning methods (Chen et al. 2023; Chai et al. 2023) adapt to new tasks by training the mask decoder, but this introduces substantial tunable parameters for each task during CL. To avoid this, we shift adaptation to the image encoder, bypassing mask decoder training. We use point-type prompts, converting them into heatmaps that integrate with image features. Each heatmap is generated by mapping points to its image-space location, assigning intensity values, and applying smoothing for continuity. We use linear layers for dimensional alignment and inject the resulting *Prompt Augmentation* into all *SLoRA*. This brings us to the whole *AugModule*, as describes below:

$$Y_i = W_i X_i + B_i(AX_i + P), \tag{3}$$

where P represents *Prompt Augmentation*. We also inject *Prompt Augmentation* into the MLP (also known as the neck module) in SAM's image encoder to further enhance its effectiveness, as shown in Figure 1. Details of the generating process are provided in the Appendix B.3.

Comparing Figure 3a and Figure 3b, most methods improve with *Prompt Augmentation*, demonstrating its effectiveness. Both vanilla LoRA and *SLoRA* outperform AsymmLoRA in Figure 3b, highlighting the importance of matrix A in extracting low-rank features and integrating information. The results also show that *Prompt Augmentation* does not degrade low-rank representations, confirming compatibility LoRA-type methods. Note that *Prompt Augmentation* operates in parallel with SAM's native prompts: box, mask, or point prompts go to the mask decoder, while point prompts (which can be sampled from box or mask prompts) are used for *Prompt Augmentation*. For simplicity, we use only point prompts throughout this paper.

4.2 Module Selector

Decoupling incremental knowledge into distinct modules shows strong potential in mitigating forgetting (Rusu et al. 2016). The main challenge is selecting the appropriate module during testing without access to task identities. This requires a "selector" that can automatically identify the input domain (Yang et al. 2024; Aljundi, Chakravarty, and Tuytelaars 2017). However, training such a selector on large buffers of raw images incurs substantial storage costs (Yang et al. 2024), limiting scalability. To achieve low storage overhead and high performance, the key question is: **can we leverage SAM's pre-trained knowledge to aid module selection during testing?** Since SAM's image encoder extracts rich representations essential for mask decoding, we hypothesize that while it may not perform pixel-wise classification, it can effectively distinguish input domains through class-wise classification.

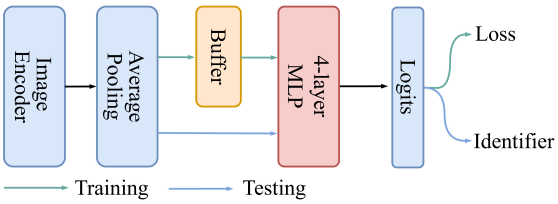

Figure 4: Overview of the *Module Selector*. We extract a fixed number (N_e) of embeddings from a specific block in the image encoder and reduce their dimensions from (N_e, H, W, D) to (N_e, D) by averaging over H and W dimensions. These embeddings ($e_i \in \mathbb{R}^D$, where $i = 1, \ldots, N_e$) are stored in a buffer aggregating embeddings from all learned domains. The *Module Selector*, a lightweight MLP with four linear layers, is trained using cross-entropy loss for 25 epochs on this embedding buffer. The layout of the four layers is detailed in the Appendix B.2.

Inspired by ViT (Dosovitskiy et al. 2021), which uses final embeddings for classification, we leverage latent embeddings from SAM's image encoder to test our hypothesis. The

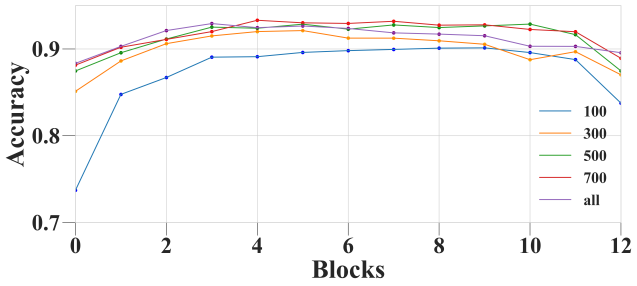

Figure 5: Selection accuracy across blocks in the image encoder of SAM (ViT-b). Each line denotes a different number of stored embeddings per dataset, all showing a similar trend. Results indicate high domain classification accuracy, with middle blocks performing best.

experimental setup is shown in Figure 4. Specifically, we divide the entire SAM model θ into two consecutive parts, $\theta = [\theta_1, \theta_2]$, beginning at a designated intermediate block within image encoder. The embedding extraction is formulated as:

$$e = f_{\theta_1}(x, \text{Prompt}), \tag{4}$$

where f is the forward function and x is the input image. As shown in Figure 5, the model achieves high domain classification accuracy, validating our hypothesis. This motivates the design of the *Module Selector*, trained on low-consumed embeddings from the buffer rather than raw images, while maintaining high selection accuracy.

At inference time, the appropriate module is selected by:

$$\text{id} = S(e), \tag{5}$$

where S denotes the *Module Selector* and id is the index of the selected module. The final SAM inference with the selected module is given by:

$$y = f_{\theta_2^*}(x, \text{Prompt}), \tag{6}$$

where θ_2^* represents θ_2 combined with the selected learning module, and y is the output.

To sum up, let us take an example of the whole process of SAMCL. During training, we obtain embeddings with 300 number per domain from the 6^{th} block in the image encoder to train *Module Selector* and fine-tune only blocks after the 6^{th} block by *AugModule*. During inference, we first extract the embedding from the 6^{th} block and identify the appropriate module by *Module Selector*. Then we extract the corresponding *AugModule* and inject it in the subsequent blocks for inference. Note that SAMCL may appear unable to fully recover the original SAM. However, this can be addressed by training the *Module Selector* on a small COCO subset, effectively representing a virtual module corresponding to the original SAM. In other words, the COCO subset serves as a task in the CL process. The above process and setup are used for SAM, with the SAM2 configuration detailed in the Appendix B.4. The pseudo-code for SAMCL in both training and inference process is provided in Algorithm 1.

Algorithm 1: SAMCL (Section 4)

Input: SAM, Epoch, Task Number T, Training Dataset D, Testing Dataset $Test$, Function of Getting Embedding $F()$, Training Function $Train()$, Inference Function $Infer()$, Select Function $Select()$
Module Set, Buffer $\leftarrow \emptyset, \emptyset$
for $t = 1$ **to** T **do**
  // Training process
  Model $\leftarrow$ Initialize SAM with a new AugModule
  Buffer $\leftarrow F(\text{Model}, \text{sample}(D_t))$
  $Train(\text{Module\_Selector}, \text{Buffer})$
  for epoch=1 **to** Epoch **do**
    $Train(\text{Model}, D_t)$
  end for
  Module Set $\leftarrow$ AugModule in Model
  // Inference process
  for data **in** $Test_t$ **do**
    embedding $\leftarrow F(\text{SAM}, \text{data})$
    ID $\leftarrow$ Module_Selector(embedding)
    $\text{module}_{test} \leftarrow Select(\text{Module Set}, \text{ID})$
    Model $\leftarrow$ Initialize SAM with module_{test}
    outputs $\leftarrow Infer(\text{Model}, \text{embedding})$
  end for
end for

5 Experiment

In this section, we evaluate the performance of SAMCL. We begin by outlining the experimental setup, followed by a comparative analysis against existing CL methods and an assessment of robustness. We then conduct ablation studies to validate the effectiveness of *AugModule* and *Module Selector*. Additional implementation details and experiments are provided in the Appendix C and Appendix D, respectively.

5.1 Experimental setting

For datasets, given limitations noted in prior work (Ji et al. 2024), we choose three challenging domains: camouflaged, shadow, and medical domains. For camouflaged, we use COD (Fan et al. 2020) and CAMO (Yan et al. 2021); for medical, ISIC (Codella et al. 2018) and Kvasir (Pogorelov et al. 2017); and for shadow, ISTD (Wang, Li, and Yang 2018). The CL dataset order used in the main paper is Kvasir, CAMO, ISTD, ISIC, and COD, interleaving different domains. All datasets are trained for 20 epochs and evaluated by mIoU, mF1, and mMAE. We adopt AA, FM, and FT for CL evaluation. We measure storage efficiency by accounting for all extra storage consumption incurred during CL, excluding the base SAM model. To support reverting SAMCL to the original SAM, we train *Module Selector* on selected 100 samples from COCO (Lin et al. 2015) after CL.

We compare the following representative methods: (1) SAM fine-tuning (SAM-Adapter (Chen et al. 2023), SAM-LST (Chai et al. 2023), AutoSAM (Shaharabany et al. 2023)); (2) General CL methods (EWC (Kirkpatrick et al. 2017), ER (Chaudhry et al. 2019), DER (Buzzega et al. 2020)); (3) Architecture-based method for foundation models (O-LoRA (Wang et al. 2023a)); (4) Continual segmen-

Method	Storage (MB)	AA			FM			FT		
		mIoU ↑	mF1 ↑	mMAE ↓	mIoU ↓	mF1 ↓	mMAE ↑	mIoU ↑	mF1 ↑	mMAE ↓
Kvasir → CAMO → ISTD → ISIC → COD										
SAM										
SAM-Adapter	15.72	0.428	0.565	0.125	0.191	0.190	-0.046	0.318	0.457	0.196
SAM-LST	44.87	0.421	0.533	0.170	0.239	0.222	-0.180	0.291	0.421	0.221
AutoSAM	158.53	0.184	0.273	0.258	0.330	0.330	-0.103	0.124	0.199	0.423
LoRA	0.30	0.703	0.804	0.064	0.123	0.089	-0.034	0.513	0.521	0.158
EWC	0.61	0.716	0.816	0.058	0.111	0.078	-0.028	0.549	0.663	0.160
ER	1125.30	0.808	0.881	0.035	0.010	0.007	-0.003	0.630	0.748	0.087
DER	1125.30	0.804	0.879	0.035	0.022	0.015	-0.005	0.643	0.760	**0.082**
SPPA	5.24	0.282	0.407	0.149	0.337	0.315	-0.072	0.417	0.550	0.197
LAG	0.61	0.703	0.810	0.063	0.099	0.066	-0.025	0.452	0.576	0.205
O-LoRA	1.54	0.704	0.806	0.059	0.091	0.066	-0.023	0.519	0.642	0.160
MoDA	135.50	0.756	0.835	0.052	0.020	0.014	-0.002	0.637	0.757	0.094
SAMCL (Ours)	9.11	**0.836**	**0.900**	**0.029**	**0.0019**	**0.0016**	**-0.0003**	**0.668**	**0.773**	0.089
SAM2										
LoRA	0.12	0.664	0.768	0.074	0.169	0.132	-0.046	0.479	0.600	0.225
EWC	0.24	0.697	0.797	0.066	0.139	0.104	-0.038	0.502	0.618	0.241
ER	1125.12	0.801	0.876	0.036	0.028	0.020	-0.008	0.582	0.699	**0.148**
LAG	0.24	0.646	0.753	0.074	0.182	0.142	-0.044	0.477	0.605	0.196
O-LoRA	0.61	0.655	0.767	0.073	0.146	0.108	-0.039	0.499	0.622	0.205
SAMCL (Ours)	6.12	**0.843**	**0.905**	**0.028**	**0.0019**	**0.0017**	**-0.0009**	**0.613**	**0.722**	0.149

Table 1: Comparison results during the CL setting. Considering the compatibility and performance, we select a few representative methods to compare SAMCL with SAM2. The results show the excellent performance of SAMCL. The final testing results for all datasets are provided in the Appendix D.3.

tation (SPPA (Lin, Wang, and Zhang 2022), LAG (Yuan, Zhao, and Shi 2024)); (5) SAM-specific continual learning (MoDA (Yang et al. 2024) + HQ-SAM (Ke et al. 2023)). For fair comparison, all general CL and segmentation methods are integrated with LoRA. We use the ViT-b version of SAM and the tiny version of SAM2 (v2.1), selecting three random object points as prompts during training and inference. The batch size for SAM is 8, whereas for SAM2 it is 16.

For SAMCL, we use AdamW optimizer with an initial learning rate of 0.005 and cosine decay. Each task stores 300 embeddings, same as ER and DER. For MoDA, we follow its original setup, storing 10 raw samples per task. The low-rank dimension of *SLoRA* is 10, consistent across all LoRA-based baselines. *Module Selector* is trained with cross-entropy loss $\mathscr{L}_c$. For fine-tuning SAM with *AugModule*, we follow the official SAM implementation (Kirillov et al. 2023), modifying only loss coefficients. The IoU head is trained with MSE loss to measure the gap between the predicted and true IoU. The mask prediction is supervised by a weighted sum of focal (Lin et al. 2017) and dice loss (Milletari, Navab, and Ahmadi 2016). The full loss function is:

$$\mathscr{L}_s = \mathcal{L}_{Focal} + 10 * \mathcal{L}_{Dice} + \mathcal{L}_{Mse}. \tag{7}$$

5.2 Comprehensive Results

As shown in Table 1, SAMCL with SAM and SAM2 demonstrates exceptional performance during CL. The AA and FM metrics demonstrate that SAMCL achieves strong average accuracy while effectively mitigating forgetting. Notably, the FT metric reveals that SAMCL exhibits excellent transferability without training on unseen tasks.

In contrast, fine-tuning methods (SAM-Adapter, SAM-LST, and AutoSAM) perform poorly in the CL setting, underscoring the need to enhance SAM with CL capabilities across different domains. The FT value also suggests overfitting in fine-tuning methods, indicating a lack of generalization to unseen domains. Additionally, continual segmentation methods (SPPA and LAG) struggle to delay forgetting when integrated with SAM.

General CL methods (EWC, ER, and DER) can delay forgetting to some extent, particularly the replay-based methods ER and DER, as indicated by the FM metric. However, SAMCL outperforms these approaches by effectively preventing interference between different knowledge domains. Moreover, ER and DER exhibit low efficiency, storing 300 raw image samples ($x^i \in \mathbb{R}^{3\times256\times256}$) per domain in buffer, totaling approximately 225MB. In contrast, SAMCL stores an equal number of embedding samples per domain ($e^i \in \mathbb{R}^{768}$ for SAM and $e^i \in \mathbb{R}^{384}$ for SAM2), resulting in a storage cost of only 0.878 MB for SAM and 0.439 MB for SAM2—representing a remarkable reduction of up to $256\times$ and $512\times$, respectively. The total storage cost is reduced by $123.52\times$ and $183.84\times$ for SAM and SAM2 compared with ER and DER, respectively, as shown in Table 1. Notably, while ER and DER require training on both previous and new samples during CL, SAMCL only trains on stored embeddings once using a simple MLP within 25 epochs before addressing a new task, resulting in a negligible training burden. Although some methods, such as EWC and O-LoRA, incur lower storage costs, their performance is significantly inferior to that of SAMCL and ER.

Method	TP (MB)	Kvasir			CAMO			ISTD			ISIC			COD		
		mIoU ↑	mFI↑	MAE↓	mIoU↑	mFI↑	MAE ↓	mIoU↑	mFI↑	MAE ↓	mIoU ↑	mFI↑	MAE↓	mIoU ↑	mFI↑	MAE↓
SAM																
SAM	0	0.736	0.824	0.082	0.579	0.701	0.111	0.612	0.723	0.091	0.650	0.761	0.160	0.655	0.763	0.042
SAM-Adapter	15.724	0.561	0.720	0.098	0.532	0.709	0.114	0.824	0.897	0.038	0.718	0.841	0.050	0.513	0.673	0.062
SAM-LST	44.874	0.709	0.817	0.050	0.471	0.624	0.137	0.854	0.904	0.028	0.826	0.898	0.051	0.510	0.652	0.057
AutoSAM	158.532	0.634	0.726	0.080	0.261	0.379	0.334	0.697	0.756	0.087	0.727	0.805	0.097	0.255	0.355	0.173
LoRA	0.308	0.861	0.919	**0.022**	0.735	0.832	0.055	0.913	0.947	0.013	0.878	0.933	0.035	0.733	0.829	0.026
AsymmLoRA	0.308	0.847	0.909	0.028	0.728	0.829	0.059	0.901	0.940	0.016	0.868	0.927	0.037	0.759	0.847	0.023
HydraLoRA	0.674	0.863	0.922	**0.022**	0.743	0.837	0.052	0.923	0.955	0.010	0.875	0.931	0.036	0.698	0.802	0.030
SLoRA	0.190	0.859	0.916	0.025	0.738	0.836	0.052	0.916	0.950	0.012	0.879	0.931	0.036	0.735	0.832	0.025
AugModule	0.233	**0.867**	**0.922**	0.023	**0.751**	**0.845**	**0.051**	**0.927**	**0.957**	**0.009**	**0.879**	**0.933**	**0.034**	**0.782**	**0.864**	**0.019**
SAM2																
SAM2	0	0.767	0.844	0.081	0.656	0.760	0.111	0.522	0.637	0.200	0.615	0.724	0.232	0.690	0.788	0.042
LoRA	0.123	0.860	0.918	0.023	0.743	0.837	0.052	**0.934**	**0.962**	**0.009**	0.874	0.931	0.034	0.752	0.843	0.023
AsymmLoRA	0.123	0.864	0.920	0.023	0.750	0.845	0.055	0.931	0.959	0.011	0.877	0.932	0.034	0.750	0.841	0.024
HydraLoRA	0.273	0.860	0.916	0.029	0.757	0.848	0.049	0.931	0.958	0.011	0.880	0.933	0.034	0.781	0.863	0.020
SLoRA	0.079	0.872	0.927	0.022	0.752	0.843	0.050	0.930	0.953	0.0013	0.873	0.929	0.0335	0.751	0.843	0.024
AugModule	0.079	**0.884**	**0.935**	**0.015**	**0.779**	**0.866**	**0.044**	**0.934**	0.960	**0.009**	**0.881**	**0.935**	**0.033**	**0.786**	**0.868**	**0.019**

Table 2: Comparison with fine-tuning methods over 20 epochs. We selected three fine-tuning methods for SAM: SAM-Adapter, SAM-LST, and AutoSAM. For LoRA-type methods (LoRA, AsymmLoRA, and HydraLoRA), we set the rank to 10 for both SAM and SAM2, consistent with *AugModule* and *SLoRA* (without *Prompt Augmentation*). Additionally, for HydraLoRA, we configured the number of multiple matrices Bs to 4.

Compared to the architecture-based method O-LoRA, SAMCL excels, as O-LoRA fails to achieve true orthogonality during testing. Additionally, although MoDA is an existing CL method for SAM, it performs worse than SAMCL. Following its original design, MoDA requires storing 10 samples per domain in a buffer, resulting in a storage cost that is still $8.53\times$ higher than that of SAMCL. The total storage consumption reaches over $14.87\times$ that of SAMCL in SAM, as shown in Table 1.

Therefore, this comparison underscores the superior storage efficiency and CL performance of SAMCL with SAM, highlighting its significance in the field. The visualized result during CL process is provided in Appendix D.2.

5.3 Robustness Analysis

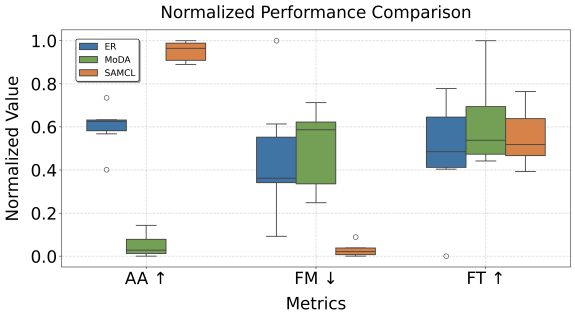

Figure 6: Analysis of the robustness. Following the settings outlined in Table 1, we conduct experiments across six different orders. For clarity, we normalize the mIoU metric values.

CL is challenged not only by catastrophic forgetting but also by the sensitivity to task order (Bell and Lawrence 2022), which can lead to unstable performance. This section evaluates the performance of SAMCL under different task orderings. In Figure 6, we compare the robustness of SAMCL with ER and MoDA. The results on AA and FM show that SAMCL achieves superior robustness in both learning and retention. This is because SAMCL allocates different knowledge to distinct learning modules, and the accurate selector in SAMCL chooses the most appropriate one. For transferring, all three methods exhibit similar levels of dispersion. Detailed values and comparisons for each order of these two methods are presented in Appendix D.3.

5.4 Ablation Study

This section displays ablation studies about *AugModule* and *Module Selector* in SAMCL. We firstly presents the overall ablation study of these two components, as shown in Table 3. Based on this experiment, we will provide a detailed explanation of these two components.

AugModule	Module Selector	AA ↑	FM ↓	FT ↑
×	×	0.703	0.123	0.513
✓	×	0.748	0.086	0.570
✓	✓	**0.836**	**0.0019**	**0.668**

Table 3: Ablation study of all components in SAMCL with SAM. The experiment setting is the same as in Table 1 and only uses mIoU metrics for evaluation. We use LoRA as a baseline reference without SAMCL, shown in the first line.

Discussion on AugModule The *AugModule*, including *SLoRA* and *Prompt Augmentation*, is designed to learn from a new domain, characterized by its strong learning ability and low storage cost, both essential for CL. In this section, we evaluate these two aspects. As shown in Table 2, we compare *AugModule* with other fine-tuning methods in both SAM and SAM2. The results show that *AugModule* achieves great performance in learning all domains over 20

epochs. The cost of storing parameters is significantly lower compared to other methods.

For LoRA-type methods, although AsymmLoRA reduces the number of fine-tuning parameters, its storage cost remains the same as LoRA, and its performance is inferior. *SLoRA*, we proposed, uses almost half of the storage costs with similar performance compared with LoRA and outperforms AsynmmLoRA. HydraLoRA achieves better performance across most datasets than LoRA and *SLoRA*. However, *AugModule*, with the help of *Prompt Augmentation*, outperforms HydraLoRA. Meanwhile, for SAM, *AugModule* uses only 0.233 MB, representing a reduction of 24.3%, compared with LoRA. For SAM2, it utilizes just 0.079 MB, resulting in a 35.7% reduction, compared with LoRA. Therefore, these results show the both effectiveness and storage efficiency. Due to the poor performance of other fine-tuning methods (SAM-Adapter, SAM-LST, and AutoSAM) for SAM, we provide analysis in the Appendix D.6.

Exploration on Module Selector *Module Selector* in SAMCL is responsible for selecting appropriate modules for inference. This function not only mitigates the forgetting problem but also facilitates transfer to unseen domains. As shown in Table 3, there is a minimal difference in the FM metric when only *AugModule* is used. Surprisingly, after applying *Module Selector*, the FM score significantly decreases while the FT score greatly increases, demonstrating its effectiveness in both mitigating forgetting and enhancing transferability. In this section, we verify these two factors of *Module Selector*.

To evaluate the effectiveness of mitigating forgetting, we conduct experiments that gradually add modules, starting with only the *Module Selector*. As shown in Figure 7, we compare SAMCL with MoDA, an existing CL method for SAM that employs a similar selector. Compared to MoDA, SAMCL exhibits greater stability in mitigating forgetting, with minimal changes across each domain during CL. This is primarily due to the *Module Selector*'s exceptional ability to select appropriate modules during the testing process.

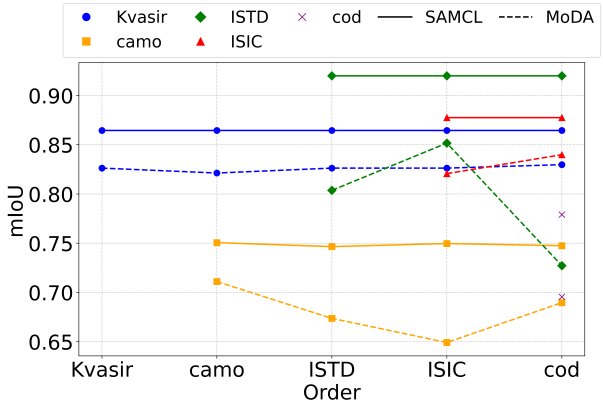

Figure 7: Comparison of performance changes between SAMCL (solid line) and MoDA (dashed line) across each domain during CL. The dataset order and experimental setup are consistent with Table 1.

To verify the effectiveness of transferability, we conduct an experiment to show the potential of using previously seen domains to test unseen domains. To better illustrate the transferability across learning domains, we evaluate only the relationships between learned domains and do not assess the reversion to the original SAM. As illustrated in Figure 8a, the results show that the unseen domain can benefit from multiple previous domains. For example, when testing on the unseen COD dataset, the *Module Selector* automatically selects the module trained on the CAMO dataset. This is primarily because both datasets belong to the camouflage domain. To further verify its effectiveness, we examine the selection similarity during testing on the ISIC dataset, shown in Figure 8b. For example, the first image in the second row is automatically selected as the Kvasir domain, as its texture resembles that of objects in Kvasir, compared to the first image in the first row. The object in the second image appears in a camouflaged environment, so it is automatically identified as belonging to the CAMO domain. The third image, resembling a shadow object, is identified as part of the ISTD domain. Therefore, with the help of the *Module Selector*, SAMCL effectively leverages all previously learned knowledge to address unseen domains, showcasing its excellent transferability.

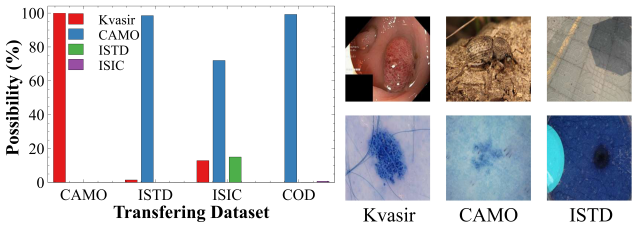

Figure 8: Transferring ability of *Module Selector*. The order of datasets and experimental setup are the same as in Table 1. (a) shows the possibility of selecting modules for testing unseen domains. The x-axis represents testing unseen domains, while the y-axis indicates the possibility of utilizing seen domains for unseen domains. (b) provides details of selecting different learned domains (including Kvasir, CAMO, and ISTD) for testing the ISIC dataset. The first row displays examples from the seen domains, and the second row lists examples from the ISIC dataset that are recognized as belonging to specific seen domains.

6 Conclusion

This paper proposes an efficient CL method, termed SAMCL, to enable SAM to learn continually across dynamic domains by *AugModule* and *Module Selector*. Crucially, by exploring and reducing the redundancy in module design and selection in SAM, SAMCL achieves a more effective approach to delay *catastrophic forgetting* with extreme storage efficiency. In summary, we believe that SAMCL will stimulate significant interest in SAM applications and provide valuable insights for future research.

Acknowledgements

The work was supported by the Fundamental Research Funds for the Central Universities (No. XJSJ25005), Ministry of Education Top-notch Student Training Program in Basic Disciplines 2.0 Research Topics (No. 20252012), the Natural Science Basis Research Plan in Shaanxi Province of China (No. 2025JC-JCQN-089) and the Outstanding Youth Science Foundation of Shaanxi Province under Grant 2025JC-JCQN-083. Thanks to the help provided by the National Experimental Teaching Demonstration Center for Computer Network and Information Security affiliated with Xidian University.

References

Aljundi, R.; Chakravarty, P.; and Tuytelaars, T. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3366–3375.

Bell, S. J.; and Lawrence, N. D. 2022. The Effect of Task Ordering in Continual Learning. *Arxiv e-prints arXiv:2205.13323*.

Buzzega, P.; Boschini, M.; Porrello, A.; Abati, D.; and CALDERARA, S. 2020. Dark Experience for General Continual Learning: a Strong, Simple Baseline. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 15920–15930.

Chai, S.; Jain, R. K.; Teng, S.; Liu, J.; Li, Y.; Tateyama, T.; and wei Chen, Y. 2023. Ladder Fine-tuning approach for SAM integrating complementary network. *Arxiv e-prints arXiv:2306.12737*.

Chaudhry, A.; Rohrbach, M.; Elhoseiny, M.; Ajanthan, T.; Dokania, P. K.; Torr, P. H.; and Ranzato, M. 2019. Continual Learning with Tiny Episodic Memories. *ICML Workshop: Multi-Task and Lifelong Reinforcement Learning*.

Chen, T.; Zhu, L.; Deng, C.; Cao, R.; Wang, Y.; Zhang, S.; Li, Z.; Sun, L.; Zang, Y.; and Mao, P. 2023. Sam-adapter: Adapting segment anything in underperformed scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 3367–3375.

Codella, N. C. F.; Gutman, D.; Celebi, M. E.; Helba, B.; Marchetti, M. A.; Dusza, S. W.; Kalloo, A.; Liopyris, K.; Mishra, N.; Kittler, H.; and Halpern, A. 2018. Skin lesion analysis toward melanoma detection: A challenge at the 2017 International symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 168–172.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*.

Fan, D.-P.; Ji, G.-P.; Sun, G.; Cheng, M.-M.; Shen, J.; and Shao, L. 2020. Camouflaged object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*.

Ji, W.; Li, J.; Bi, Q.; Liu, T.; Li, W.; and Cheng, L. 2024. Segment Anything Is Not Always Perfect: An Investigation of SAM on Different Real-world Applications. *Mach. Intell. Res.*, 21(4): 617–630.

Ke, L.; Ye, M.; Danelljan, M.; Liu, Y.; Tai, Y.-W.; Tang, C.-K.; and Yu, F. 2023. Segment Anything in High Quality. In *NeurIPS*.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; Dollar, P.; and Girshick, R. 2023. Segment Anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 4015–4026.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526.

Li, Z.; and Hoiem, D. 2018. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(12): 2935–2947.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollar, P. 2017. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C. L.; and Dollár, P. 2015. Microsoft COCO: Common Objects in Context. arXiv:1405.0312.

Lin, Z.; Wang, Z.; and Zhang, Y. 2022. Continual semantic segmentation via structure preserving and projected feature alignment. In *European Conference on Computer Vision (ECCV)*, 345–361. Springer.

Lopez-Paz, D.; and Ranzato, M. A. 2017. Gradient Episodic Memory for Continual Learning. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, Inc.

Mallya, A.; and Lazebnik, S. 2018. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

McCloskey, M.; and Cohen, N. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C): 109–165.

Milletari, F.; Navab, N.; and Ahmadi, S.-A. 2016. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, 565–571.

Pietron, M.; Faber, K.; Zurek, D.; and Corizzo, R. 2025. TinySubNets: An Efficient and Low Capacity Continual Learning Strategy. In *AAAI*, 19913–19920.

Pietron, M.; Zurek, D.; Faber, K.; and Corizzo, R. 2023. Ada-QPacknet - Multi-Task Forget-Free Continual Learning with Quantization Driven Adaptive Pruning. In *ECAI*, 1882–1889.

Pogorelov, K.; Randel, K. R.; Griwodz, C.; Eskeland, S. L.; de Lange, T.; Johansen, D.; Spampinato, C.; Dang-Nguyen, D.-T.; Lux, M.; Schmidt, P. T.; Riegler, M.; and Halvorsen, P. 2017. KVASIR: A Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, MMSys'17, 164–169. New York, NY, USA: ACM. ISBN 978-1-4503-5002-0.

Ravi, N.; Gabeur, V.; Hu, Y.-T.; Hu, R.; Ryali, C.; Ma, T.; Khedr, H.; Rädle, R.; Rolland, C.; Gustafson, L.; Mintun, E.; Pan, J.; Alwala, K. V.; Carion, N.; Wu, C.-Y.; Girshick, R.; Dollár, P.; and Feichtenhofer, C. 2024. SAM 2: Segment Anything in Images and Videos. *arXiv preprint arXiv:2408.00714*.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive Neural Networks. *Arxiv e-prints arXiv:1606.04671*.

Shaharabany, T.; Dahan, A.; Giryes, R.; and Wolf, L. 2023. AutoSAM: Adapting SAM to Medical Images by Overloading the Prompt Encoder. In *34th British Machine Vision Conference 2023, BMVC 2023, Aberdeen, UK, November 20-24, 2023*, 530–533. BMVA Press.

Song, Y.; Zhao, J.; Harris, I. G.; and Jyothi, S. A. 2025. ShareLoRA: Parameter Efficient and Robust Large Language Model Fine-tuning via Shared Low-Rank Adaptation. arXiv:2406.10785.

Tang, L.; Huang, K.; Chen, C.; Yuan, Y.; Li, C.; Tu, X.; Ding, X.; and Huang, Y. 2025a. Dissecting Generalized Category Discovery: Multiplex Consensus under Self-Deconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 297–307.

Tang, L.; Yuan, Y.; Chen, C.; Huang, K.; Ding, X.; and Huang, Y. 2024. Bootstrap Segmentation Foundation Model under Distribution Shift via Object-Centric Learning. arXiv:2408.16310.

Tang, L.; Yuan, Y.; Chen, C.; Zhang, Z.; Huang, Y.; and Zhang, K. 2025b. OCRT: Boosting Foundation Models in the Open World with Object-Concept-Relation Triad. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 25422–25433.

Tian, C.; Shi, Z.; Guo, Z.; Li, L.; and Xu, C. 2024. HydraLoRA: An Asymmetric LoRA Architecture for Efficient Fine-Tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wang, J.; Li, X.; and Yang, J. 2018. Stacked Conditional Generative Adversarial Networks for Jointly Learning Shadow Detection and Shadow Removal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, X.; Chen, T.; Ge, Q.; Xia, H.; Bao, R.; Zheng, R.; Zhang, Q.; Gui, T.; and Huang, X. 2023a. Orthogonal Subspace Learning for Language Model Continual Learning. In *The 2023 Conference on Empirical Methods in Natural Language Processing(EMNLP)*.

Wang, Z.; Cheng, F.; Ji, K.; and Huang, B. 2025. LightCL: Compact Continual Learning with Low Memory Footprint For Edge Device. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*.

Wang, Z.; Yang, E.; Shen, L.; and Huang, H. 2023b. A Comprehensive Survey of Forgetting in Deep Learning Beyond Continual Learning. *Arxiv e-prints arXiv:2307.09218*.

Yan, J.; Le, T.-N.; Nguyen, K.-D.; Tran, M.-T.; Do, T.-T.; and Nguyen, T. V. 2021. MirrorNet: Bio-Inspired Camouflaged Object Segmentation. *IEEE Access*, 9: 43290–43300.

Yang, J.; Wu, Y.; Cen, J.; Huang, W.; Wang, H.; and Zhang, J. 2024. Continual Learning for Segment Anything Model Adaptation. arXiv:2412.06418.

Yuan, B.; and Zhao, D. 2024. A Survey on Continual Semantic Segmentation: Theory, Challenge, Method and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1–20.

Yuan, B.; Zhao, D.; and Shi, Z. 2024. Learning At a Glance: Towards Interpretable Data-Limited Continual Semantic Segmentation Via Semantic-Invariance Modelling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1–16.

Zhang, K.; and Liu, D. 2023. Customized Segment Anything Model for Medical Image Segmentation. *arXiv preprint arXiv:2304.13785*.

Zhu, J.; Greenewald, K.; Nadjahi, K.; de Ocáriz Borde, H. S.; Gabrielsson, R. B.; Choshen, L.; Ghassemi, M.; Yurochkin, M.; and Solomon, J. 2024. Asymmetry in Low-Rank Adapters of Foundation Models. In *Forty-first International Conference on Machine Learning (ICML)*.