CLoRA: Parameter-Efficient Continual Learning with Low-Rank Adaptation

Shishir Muralidhara[1] **Didier Stricker**[1,2] **René Schuster**[1,2]
[1]Augmented Vision Group, German Research Center for Artificial Intelligence (DFKI)
[2]RPTU – University of Kaiserslautern-Landau, Kaiserslautern
firstname.lastname@dfki.de

Abstract

In the past, continual learning (CL) was mostly concerned with the problem of catastrophic forgetting in neural networks, that arises when incrementally learning a sequence of tasks. Current CL methods function within the confines of limited data access, without any restrictions imposed on computational resources. However, in real-world scenarios, the latter takes precedence as deployed systems are often computationally constrained. A major drawback of most CL methods is the need to retrain the entire model for each new task. The computational demands of retraining large models can be prohibitive, limiting the applicability of CL in environments with limited resources. Through CLoRA, we explore the applicability of Low-Rank Adaptation (LoRA), a parameter-efficient fine-tuning method for class-incremental semantic segmentation. CLoRA leverages a small set of parameters of the model and uses the same set for learning across all tasks. Results demonstrate the efficacy of CLoRA, achieving performance on par with and exceeding the baseline methods. We further evaluate CLoRA using NetScore, underscoring the need to factor in resource efficiency and evaluate CL methods beyond task performance. CLoRA significantly reduces the hardware requirements for training, making it well-suited for CL in resource-constrained environments after deployment.

1 Introduction

While neural networks have demonstrated remarkable performance in deep learning across various domains, their rigid structure is prohibitive in adapting to new tasks. Typically, neural networks are trained on a fixed dataset, but real-world scenarios are non-stationary (Hadsell et al., 2020) where data drift occurs, which can impact model performance (Ditzler et al., 2015), and objectives may change over time, such as the introduction of new classes or tasks (Hadsell et al., 2020). Fine-tuning or transfer learning on the new data would lead to catastrophic forgetting (Kirkpatrick et al., 2017), causing the network to perform poorly on the previously learned tasks, this stems from the inherent plasticity of neural networks where previously learned information is overwritten during the learning of new tasks. Incrementally learning results in the stability-plasticity dilemma (Mermillod et al., 2013), where networks with higher stability are restrictive in learning new tasks, and higher plasticity tend to overwrite previously learned information. A straightforward solution to prevent forgetting is to retrain the model with all encountered data, but this requires significant computational resources (Trinci et al., 2024), training time, storage, and may not always be feasible due to data unavailability (Lesort et al., 2020). In contrast to traditional machine learning, where models are trained in isolation and fixed, continual learning (CL) is a dynamic learning paradigm that more accurately mirrors the non-stationary nature of the real world. Continual learning involves incrementally learning a sequence of tasks (Kalb et al., 2021), while minimizing catastrophic forgetting on previous tasks under restricted or no access to previously encountered data. Each task can represent a change either in the input or output distribution resulting in two main incremental-learning settings. In domain-incremental learning, a task represents a change in the input distribution, such as different image sources. This can also be extended to learning modalities in the case of modality-incremental learning (Hegde et al., 2025). Class-incremental learning represents a shift in the output distribution by incorporating novel, previously unseen classes, and extends to learning evolving classes in CLEO (Muralidhara et al., 2024). CL therefore focuses on efficiently adapting an existing model to new tasks, circumventing the need for retraining from scratch. However, most CL methods other than those based on transfer-learning are designed under the assumption of having access to offline resources (Prabhu et al., 2023), overlooking computational constraints of deployed systems which could be prohibitive in updating large networks. In this work, we address both storage and computational constraints and present an approach that adheres to the original constraint of restricted data access, while being able to operate under resource constrained environments, without compromising on the choice of networks.

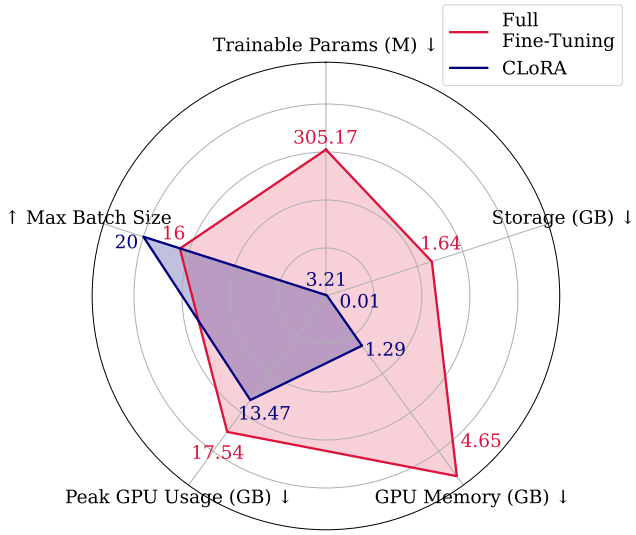

Figure 1: Comparison of resource efficiency of CLoRA against full fine-tuning, under identical conditions.

Parameter-efficient fine-tuning (PEFT) methods were developed for adapting Large Language Models (LLMs) to specific downstream tasks (Houlsby et al., 2019). PEFT methods significantly reduce the number of trainable parameters while still achieving performance on-par with full fine-tuning. PEFT offers several advantages over full fine-tuning, such as reducing hardware requirements, and avoiding overfitting on the fine-tuned dataset (Xu et al., 2023). PEFT with CL (PECL) minimizes the number of trainable parameters updated during adaptation, thus allowing for more efficient usage of computational resources, which is crucial in scenarios where resources are limited. (Yuan & Zhao, 2023) note that within current CL methods for semantic segmentation, those utilizing stronger backbones consistently achieve superior performance across both old and new classes. PECL facilitates leveraging the capabilities of extremely large models, even within constrained environments. In this work, we introduce Continual Learning with Low Rank-Adaptation (CLoRA), the first PECL method that uses LoRA (Hu et al., 2021) for class-incremental semantic segmentation, which forms our primary contribution. A key strength of CLoRA lies in its modular compatibility, since it is agnostic to the regularization strategy used to mitigate forgetting, making it a lightweight, resource-efficient extension to existing CL methods. We validate this versatility through extensive experiments using several baselines such as MiB (Cermelli et al., 2020), RCIL (Zhang et al., 2022), SATS (Qiu et al., 2023), and SSUL (Cha et al., 2021) across different segmentation networks. CLoRA offers resource efficiency advantages over traditional CL, as illustrated in Fig. 1 and detailed under Sec. 4.5, in which we show that CLoRA requires less hardware resources while achieving comparable or superior segmentation results to our baselines.

2 Background

Continual learning encompasses methods for enabling a system to incrementally learn new information, while retaining previously learned knowledge. These methods can be categorized into three main categories: Architecture-, replay-, and regularization-based approaches. Additionally, there are hybrid approaches that integrate a combination of these methods. A more detailed review and survey of these approaches is presented by Wang et al. (2024).

2.1 Continual Semantic Segmentation

MiB (Cermelli et al., 2020) proposes a novel distillation loss to account for background shift in incremental segmentation by comparing the background class prediction by the old model with background and new class predictions by the new model. PLOP (Douillard et al., 2021) addresses the background shift by using pseudo-labels for the background pixels predicted by the previous task model. SATS (Qiu et al., 2023) uses self-attention maps from transformers and class-specific region pooling is used for between and within class knowledge distillation. RCIL (Zhang et al., 2022) maintains two branches during training, one branch is frozen after initial training and preserves the knowledge of old classes, whereas the other branch is trainable and is used for learning new tasks. REMINDER (Phan et al., 2022) uses a class-similarity based distillation, to distill knowledge from a previous model with classes similar to the new classes. AWT (Goswami et al., 2023) addresses background shift through classifier initialization by identifying the

most relevant weights from the previous background for the new classes. EWF (Xiao et al., 2023) merges the trained models of previous and current tasks, weighted by a merging factor. SSUL-M (Cha et al., 2021) addresses background shift by introducing an unknown class that separates the future classes from the background and uses pseudo-labeling for previous classes. ALIFE (Oh et al., 2022) stores feature representations extracted from previous models for replay instead of explicitly storing images from previous tasks. RECALL (Maracani et al., 2021) uses a generic generative model or web-crawler for generating/retrieving images of previous classes and pseudo-labeling. DiffusePast (Chen et al., 2023) addresses issues with GAN-generated images, and uses a stable diffusion model for generating accurate images. In this work, we focus on class-incremental learning with knowledge distillation.

2.2 Transfer-Learning-based Continual Learning

Transfer-learning-based CL methods leverage large pretrained networks for feature extraction and only train the classifier for incrementally learning new classes. Typically, these methods use a pretrained network, that is either frozen directly or after training for the initial task. By forgoing full network retraining, this approach addresses the computational constraints in continual learning. Pelosin (2022) proposes a straightforward approach that extracts features and stores class prototypes for a prototype-based classification. FeTrIL (Petit et al., 2023) uses a pseudo feature generator to represent past classes through geometric translations of new class features and old class prototypes. A linear classifier is then jointly trained on new classes and old classes. Adapt and Merge (APER) (Zhou et al., 2024) adapts the pretrained model to incremental data using PEFT to bridge the domain gap. Subsequently, it aggregates the adapted and pretrained model embeddings and freezes them for incremental tasks. RanPAC (McDonnell et al., 2024) uses random projection layers to map the features extracted from the pretrained network into a higher-dimensional space, increasing class separation. FeCAM (Goswami et al., 2024) highlights the shortcomings of Euclidean distance based prototype classification in incremental learning and proposes using a Bayesian classifier.

2.3 Parameter Efficient Continual Learning

There are several parameter-efficient fine-tuning (PEFT) methods (Xu et al., 2023) like: Additive fine-tuning, which introduces additional parameters through adapters (Rücklé et al., 2021; Liu et al., 2022) or prompts (Lester et al., 2021; Li & Liang, 2021); partial fine-tuning (Zaken et al., 2021; Zhao et al., 2020; Guo et al., 2020) where a subset of pretrained parameters are selected; reparametrized fine-tuning (Hu et al., 2021; Valipour et al., 2022; Dettmers et al., 2024) which uses low-rank transformation to reduce the number of trainable parameters. The utilization of PEFT in continual learning is gaining traction, leading to the development of computationally efficient approaches, we refer to as parameter-efficient continual learning (PECL). Hyder et al. (2022) propose a dynamically growing network with incremental rank updates, where for each task, a new trainable rank-1 matrix is added while the previous low-rank matrices are frozen. During inference, it requires the task-ID for selecting the appropriate weights. Continual learning with low rank adaptation (CoLoR) (Wistuba et al., 2023) uses LoRA for training expert models and k-means clustering for storing k cluster centers for each dataset. During inference, the task-ID is inferred by determining the nearest cluster center and the corresponding expert model is selected. This additional step incurs an additional computational overhead during both training and inference. Chitale et al. (2023) use LoRA to train expert models for each task, and then merges them using task arithmetic (Ilharco et al., 2023). It requires fine-tuning on a small subset of data gathered from each class across all tasks, similar to a rehearsal-based approach. LAE (Gao et al., 2023) framework consists of three stages: Learning new tasks by leveraging pretrained models with an online PEFT module, accumulating task-specific knowledge into an offline PEFT module, and ensembling during inference using the online and offline modules. Orthogonal low-rank adaptation (Wang et al., 2023) incrementally adds LoRA for each task and ensures orthogonality between the current and previous modules to mitigate interference between tasks and minimize forgetting. We present a PECL method, that leverages a single LoRA module for learning across all tasks. We discuss the challenges of using task-specific modules for segmentation in Sec. 3.2.

3 CLoRA: Continual Low-Rank Adaptation

Continual learning involves learning a sequence of tasks $T = \{t_0, t_1, ..., t_n\}$, where each task is associated with task-specific data (X_t, Y_t). Depending upon the incremental setting, either the distribution of X_t or Y_t varies across tasks. In class-incremental learning, the input distribution remains consistent and in each task, subsets $C_t \subset C$ of non-overlapping classes $C_i \cap C_j = \emptyset, i \neq j$ are introduced. These subsets compose the totality of classes $C = C_0 \cup C_1 \cup ... \cup C_t$.

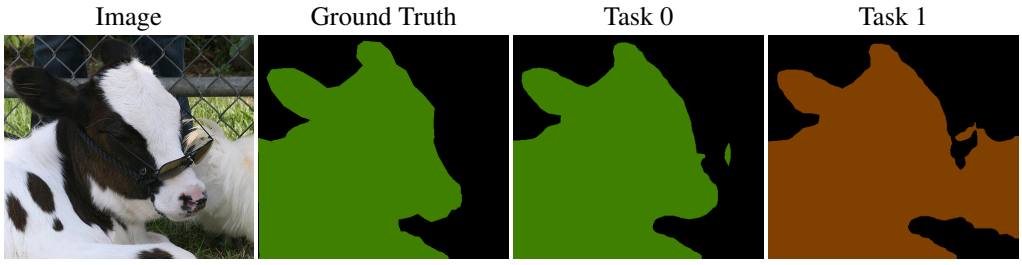

Figure 2: Conflicting predictions from task-specific modules on the PASCAL VOC (Everingham et al., 2010) dataset using the *15-5* setting, in which the two modules have conflicting predictions.

3.1 Low-Rank Adaptation (LoRA)

Low-rank adaptation (LoRA) (Hu et al., 2021) is a parameter-efficient fine-tuning method that uses reparameterization for adapting pretrained models to downstream tasks. LoRA uses low-rank transformation to significantly reduce the number of trainable parameters and the computational requirements while still achieving performance on par with full fine-tuning. For a pretrained network with weights $W \in \mathbb{R}^{d\times k}$, LoRA fine-tunes a very small subset of weights $\triangle W$, represented using two low-rank matrices, $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$, where r is the rank of the matrix and $r \ll min(d, k)$. A is initialized using random Gaussian distribution, and B is initialized as a zero matrix. The rank r is a hyperparameter that determines the number of trainable parameters. The two low-rank matrices are substituted in place of the original weights W in each layer of a transformer, and only these low-rank matrices, which are typically a fraction of the original weights, are trainable. Furthermore, LoRA is applied only to the query and value projection layers, which contributes to further reducing the number of trainable parameters. Additionally, LoRA also drastically reduces the storage footprint, allowing to store modules for each task and switching only the task-specific LoRA modules, while the pretrained weights remain constant. During training, the pretrained weights are frozen and only the LoRA weights are updated. The forward pass is modified and the output h is calculated as

$$h = W(x) + \triangle W(x) = W(x) + BA(x) \tag{1}$$

Unlike other PEFT methods such as adapters, which add inference overhead, LoRA avoids inference latency by merging the LoRA modules with the pretrained weights. The weights are updated as $W^{'} = W + BA$.

3.2 Challenges in Class-Incremental Semantic Segmentation

Current PECL methods for image classification (Hyder et al., 2022; Wistuba et al., 2023) mostly leverage individual, task-specific LoRA modules for incrementally learning new tasks. This approach holds significant appeal, achieving performance on par with full fine-tuning while entirely mitigating catastrophic forgetting as it avoids overwriting of information. Furthermore, it demonstrates storage efficiency compared to methods using dynamically expanding networks (Rusu et al., 2016), with LoRA modules consuming only a fraction of the memory required for storing complete models. While these factors make using task-specific LoRA modules a compelling choice, its application to class-incremental *semantic segmentation* presents several challenges. Unlike class-incremental classification, where images typically have a single label, and the task-ID can be used to select the expert model trained on that specific class for inference, task-ID inference in class-incremental segmentation is not straightforward as the tasks are not mutually exclusive. In segmentation, images are typically annotated with multiple classes, which may span across different tasks in a class-incremental setting, and the final prediction may require combining predictions of classes learned across different tasks, with task-specific LoRA modules. A potential solution is to use all task-specific modules during inference and merge their results (see appendix for details). However, with this approach the inference time increases proportionally with number of tasks, making it ineffective. Furthermore, merging task-wise predictions is particularly challenging due to *background shift* in incremental segmentation, where the background class is a catch-all class that encompasses all previously seen and potential future classes. During incremental learning, only the current task classes are annotated, and the remaining classes are labeled as the background class. This results in the task-specific modules being unaware of past and future classes. Consequently, the definition of the background changes across tasks, leading to inconsistent predictions of the background by the individual modules. Due to the isolated nature of task-specific modules, they are not aware of classes learned in other tasks through different modules. As a consequence, they make predictions based only on the subset of classes they have encountered. This can lead to conflicting predictions for visually similar classes, with different modules predicting different classes for the same pixel. Resolving these discrepancies and determining the correct prediction poses a significant challenge.

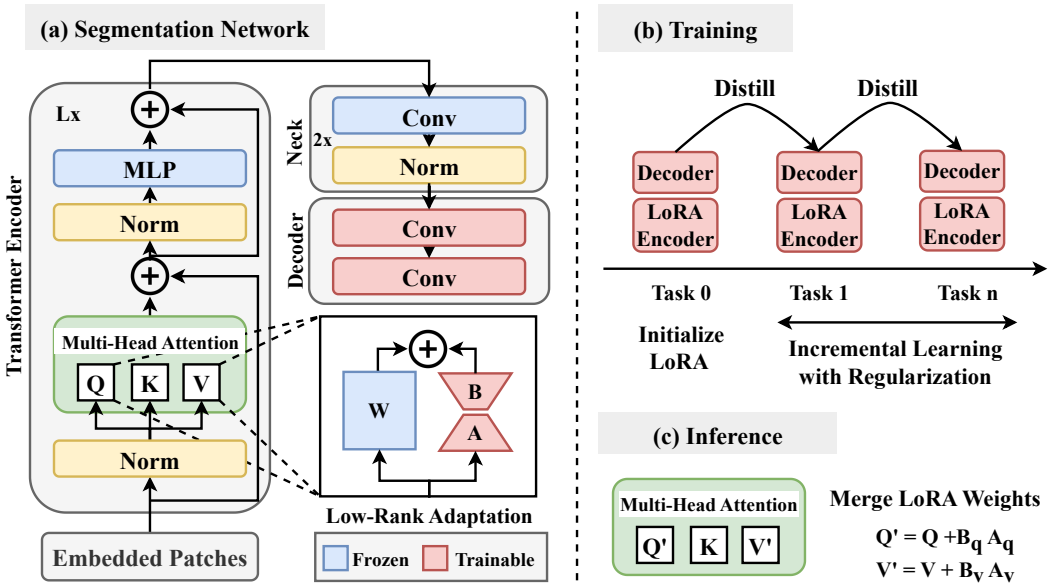

Figure 3: Continual Learning with Low-Rank Adaptation (CLoRA). (a) CLoRA uses low-rank adaptation for parameter-efficient fine-tuning of Vision Transformers in resource-constrained environments. The decoder undergoes fine-tuning while all other weights remain frozen. (b) Training: The encoder using LoRA is initialized for the initial task and trained. For the subsequent tasks, the same LoRA weights and decoder are updated using knowledge distillation. (c) Inference: After learning all tasks, the LoRA weights are merged with the frozen weights. This approach mitigates additional inference time and parameters and facilitates inference without task-ID.

Figure 2 illustrates this challenge using the *15-5* setting in PASCAL VOC. For the class *cow* which was learned in task 0, the task 1 module mistakenly predicts the class as *sheep*. This error arises because the two classes are visually similar, and during task 1, the module has not seen images of *cow* to discriminate between *cow* and *sheep*.

3.3 Proposed Approach

Considering the limitations associated with dynamically growing task-specific modules for incremental learning in semantic segmentation, we introduce CLoRA. CLoRA leverages a single LoRA module to incrementally learn tasks with knowledge distillation. This approach addresses the aforementioned limitations and presents several advantages: Unlike methods that add new LoRA modules for each task, CLoRA maintains a consistent network architecture size throughout the learning process. Utilizing a single LoRA module across all tasks results in a task-agnostic model, circumventing the challenges related to task-ID inference and conflicting predictions typically associated with employing multiple LoRA modules. Additionally, as CLoRA does not necessitate inference and merging of multiple tasks, the inference time remains constant.

An overview of CLoRA is presented in Fig. 3. For the first task, the LoRA module is initialized, and the encoder weights are substituted with LoRA, by freezing the encoder and training only the LoRA weights similar to PEFT. The decoder, which constitutes a very small portion of the network undergoes full fine-tuning. Subsequently, when new tasks are introduced incrementally, the same trained LoRA weights from the previous task are reused and updated. To preserve knowledge from previous tasks and mitigate catastrophic forgetting, distillation is used. Specifically, we utilize the knowledge distillation loss proposed by MiB (Cermelli et al., 2020). However, it is feasible to integrate other regularization approaches with minimal adjustments, as shown in our experiments (Tab. 4 and Sec. 4.4.1).

During incremental training, knowledge distillation loss is used to transfer knowledge from the previous task model, acting as the teacher f_{t-1} to the student model f_t being trained on the current task t. However, in class-incremental semantic segmentation, this approach faces the background shift problem. For the current task t with the set of classes C_t, the teacher model, trained on previous tasks, may have learned the current task classes as background in the earlier class set $C_{0:t-1}$. As a result, during distillation, there is a mismatch: the teacher predicts background class for pixels belonging to new classes, while the student correctly predicts them to the actual

Table 1: Results on PASCAL VOC (Everingham et al., 2010) dataset after learning all tasks.

Method	15-5				15-1				5-3				10-1			
	0-15	16-20	**All**	**FS**	0-15	16-20	**All**	**FS**	0-5	6-20	**All**	**FS**	0-10	11-20	**All**	**FS**
FT	05.89	40.47	14.12	-67.57	04.56	01.52	03.84	-77.85	12.05	05.66	07.49	-74.20	06.45	01.12	03.91	-77.78
CLoRA (FT)	04.69	43.47	13.94	-68.72	04.44	02.11	03.88	-78.78	11.78	07.00	08.37	-74.29	06.38	01.21	03.92	-78.74
JT	82.53	79.02	81.69	0.00	82.53	79.02	81.69	0.00	80.27	82.26	81.69	0.00	81.57	81.83	81.69	0.00
CLoRA (JT)	83.52	79.89	82.66	0.00	83.52	79.89	82.66	0.00	81.13	83.27	82.66	0.00	83.02	82.26	82.26	0.00
MiB	77.52	49.73	**70.91**	-10.78	75.22	19.47	61.95	-19.74	63.23	42.18	48.19	-33.50	26.47	19.70	23.24	-58.45
MiB (TL)	21.03	04.62	17.12	-64.57	17.76	03.15	14.28	-67.41	15.21	04.32	07.43	-74.26	00.00	03.43	01.63	-80.06
CLoRA Reinit	71.14	52.58	66.34	-16.32	80.82	31.47	69.07	-13.59	62.68	45.06	50.09	-32.56	22.66	23.17	22.90	-59.76
CLoRA	74.17	56.57	70.39	-12.27	81.29	34.41	**70.13**	-12.53	69.92	45.50	**52.47**	-30.19	31.38	29.22	**30.35**	-52.31

classes in C_t. This mismatch affects the balance between stability and plasticity, limiting the model's ability to adapt to new classes. Addressing this limitation, MiB (Cermelli et al., 2020) introduces a novel distillation strategy by aggregating the student logits for the new classes with the background logits before applying knowledge distillation.

Through knowledge distillation, we can utilize the same LoRA weights across tasks, and ensure the LoRA module is aware of classes learned across different tasks and avoids conflicting predictions. Finally, after learning all tasks, the LoRA weights can be merged back into the encoder to update the original weights. This approach helps avoid additional inference overhead such as additional computation (Hyder et al., 2022), determining task-ID and selecting task-experts (Wistuba et al., 2023), or merging of multiple task outputs.

4 Experiments and Results

In this section we discuss the datasets used to evaluate CLoRA, implementation details, and comparison baselines. We outline the task settings for each dataset and report their results (visualizations in the appendix). All reported results are evaluated in terms of mean Intersection over Union (mIoU). To assess the extent of forgetting, we include the forget score (FS), by comparing the model's performance after learning all tasks with the corresponding joint training (JT) baseline, which is used to approximate the upper bound. We highlight the efficacy of CLoRA against a frozen encoder, explore the influence of varying ranks, and reinitializing the LoRA module after each task. We demonstrate that CLoRA can be extended to integrate with other networks and approaches. Additionally, we quantify the resource efficiency of CLoRA using NetScore (Wong, 2019).

4.1 Datasets

- **PASCAL VOC** part of the Visual Object Classes Challenge (Everingham et al., 2010) contains images and annotations for 21 classes such as animals, person, vehicles and household items, including the background.
- **ADE20K** (Zhou et al., 2017) consists of over 25k images for training and 2k images for testing. The dataset covers 150 classes, allowing to design CL tasks with sizable number of classes being added in each increment.
- **Cityscapes** (Cordts et al., 2016) comprises urban environment images, with 2975 for training and 500 for validation. It includes dense annotations for 19 classes, posing challenges for segmentation tasks.

4.2 Baselines and Implementation

To evaluate the efficacy of CLoRA, a PECL method, we compare it against full fine-tuning. Full fine-tuning refers to the standard training where the entire model with all the parameters are retrained for each task. These approaches include the continual learning baselines of fine-tuning and joint training. Fine-tuning (FT) involves incrementally learning new tasks with the trained model, without any explicit intervention to mitigate catastrophic forgetting. Joint training (JT) or offline training uses all the data to train the model in a single step. Since there is no incremental learning, it circumvents forgetting and forms the upper bound for comparison. Primarily , we compare against Modeling the Background (MiB) (Cermelli et al., 2020), which uses full fine-tuning and highlight the efficacy of CLoRA, utilizing the same knowledge distillation loss. We further include comparison with SATS (Qiu et al., 2023), SSUL (Cha et al., 2021) and RCIL Zhang et al. (2022) to demonstrate the versatility of CLoRA.

Table 2: Results on ADE20K (Zhou et al., 2017) dataset after learning all tasks.

Method	100-50				50-50				25-25				100-10			
	0-100	101-150	All	FS	0-50	51-150	All	FS	0-25	26-150	All	FS	0-100	101-150	All	FS
FT	00.09	10.55	03.58	-40.68	00.01	06.47	04.31	-39.95	00.00	03.47	02.89	-41.37	00.00	00.20	00.06	-44.20
CLoRA (FT)	00.00	21.69	07.23	-34.12	00.00	09.60	06.40	-34.95	00.00	04.94	04.12	-37.23	00.00	01.75	00.58	-40.77
JT	49.53	33.73	44.26	0.00	57.38	37.71	44.26	0.00	67.40	39.64	44.26	0.00	49.53	33.73	44.26	0.00
CLoRA (JT)	48.85	26.35	41.35	0.00	56.83	33.61	41.35	0.00	66.35	36.35	41.35	0.00	48.85	26.35	41.35	0.00
MiB	46.63	20.80	38.02	-06.24	50.12	21.27	30.89	-13.37	62.33	27.25	33.09	-11.17	42.17	08.22	**30.85**	-13.41
CLoRA	44.43	25.52	**38.13**	-03.22	49.05	25.85	**33.58**	-07.77	61.02	30.06	**35.22**	-06.13	39.27	13.47	30.67	-10.68

Table 3: Results on Cityscapes (Cordts et al., 2016) dataset after learning all tasks.

Method	14-5				14-1				7-3				10-1			
	1-14	15-19	All	FS	1-14	15-19	All	FS	1-7	8-19	All	FS	1-10	11-19	All	FS
FT	00.00	00.13	00.03	-59.35	00.00	00.00	00.00	-59.38	00.00	02.25	01.42	-57.96	00.00	03.18	01.51	-57.87
CLoRA (FT)	00.00	24.78	06.52	-54.25	00.00	03.31	00.87	-59.90	00.00	07.54	04.76	-56.01	00.00	00.01	00.00	-60.77
JT	61.83	52.52	59.38	0.00	61.83	52.52	59.38	0.00	58.07	60.14	59.38	0.00	59.35	59.71	59.38	0.00
CLoRA (JT)	61.41	58.99	60.77	0.00	60.41	58.99	60.77	0.00	56.34	63.35	60.77	0.00	57.87	63.98	60.77	0.00
MiB	59.73	08.93	46.36	-13.02	60.20	07.47	46.32	-13.06	49.26	27.61	35.58	-23.80	55.44	30.41	**43.59**	-15.79
CLoRA	60.78	36.14	**54.30**	-06.47	61.57	13.01	**48.79**	-11.98	55.89	43.73	**48.21**	-12.56	56.36	23.86	40.96	-19.81

The segmentation network consists of a Vision Transformer (ViT) (Dosovitskiy et al., 2021) as the encoder, and we use the corresponding LoRA implementation by Zhang & Liu (2023). The decoder and the classifier consist of a single convolutional layer, and we use the CL framework by Cermelli et al. (2020). For training the full fine-tuning models we use the default hyperparameters defined by Cermelli et al. (2020). For training with CLoRA, we use a batch size of 6 with a higher learning rate of 0.04 for the initial task, and for subsequent tasks we use a learning rate of 0.001 for smaller increments of single classes and 0.005 for all other increments. We use a rank $r = 32$ for LoRA, which amounts to 1.04% of the total trainable parameters of the model. LoRA is applied only to the encoder, while the decoder and the classifiers are fine-tuned. We study the effect of rank r on the performance of the model and determine $r = 32$ is sufficient for almost all experiments. All models are trained for 30 epochs on each task and are evaluated using mean IoU. We present results on both the initial and incremental tasks to analyze the approach's balance between learning and retaining information.

4.3 Task Settings and Evaluation

We present results from various CL tasks across three datasets. In class-incremental learning, the tasks follow the format *init-inc*, where *init* is number of classes learned initially and the *inc* is number of classes learned in each increment. The steps are repeated until all classes are learned.

4.3.1 PASCAL VOC

We present four CL experiments using the 21 classes in PASCAL, with different sequence lengths: *15-5* (2 steps), *15-1* (6 steps), *5-3* (6 steps), and *10-1* (11 steps). The results from these experiments are presented in Tab. 1, and we observe that for most tasks except *15-5*, CLoRA surpasses MiB. CLoRA demonstrates greater effectiveness in longer and more challenging sequences of tasks, as observed in *15-1*, *5-3*, and *10-1*. Notably, CLoRA is more adept in learning new tasks across all experiments, despite MiB achieving slightly better overall results in the *15-5* setting. In the remaining experiments, CLoRA significantly outperforms MiB in retaining previous knowledge.

4.3.2 ADE20K

Leveraging the large number of classes in the ADE, we design four experiments where each step introduces a significant number of new classes. These include *100-50* (2 steps), *50-50* (3 steps), *25-25* (6 steps), *100-10* (6 steps). The results are presented in Tab. 2, in both the *100-50* and *100-10* settings, the results achieved by MiB and CLoRA are relatively similar, with CLoRA achieving slightly better performance in the *100-50* setting and MiB in the *100-10* setting. In the remaining *50-50* and *25-25* settings, CLoRA outperforms MiB by a significant margin. Once again, the effectiveness of CLoRA in learning new classes is evident across all experiments. However, with joint training, CLoRA underperforms compared to full fine-tuning, possibly due to the dataset's large size. This is further illustrated

Table 4: Results on PASCAL VOC (Everingham et al., 2010) dataset after learning all tasks using SegFormer (Xie et al., 2021). $*$ indicates rank $r = 20$.

Method	15-5	15-1	5-3	10-1
MiB (Cermelli et al., 2020)	**69.90**	58.82	**52.05**	40.69
MiB + CLoRA	69.83	**59.89**	51.48	**43.32**
SATS (Qiu et al., 2023)	69.23	**61.49**	**55.00**	39.67
SATS + CLoRA	**69.65**	60.83	52.29*	**40.23***

by examining the effect of rank on learning under Sec. 4.4. With fine-tuning, we can observe complete overwriting of information from the initial task, both with CLoRA and default fine-tuning. However, we can observe CLoRA being able to learn new classes to a greater extent.

4.3.3 CITYSCAPES

Utilizing the Cityscapes dataset, we replicate experiments similar to those conducted with PASCAL, resulting in the following experiments: *14-5* (2 steps), *14-1* (5 steps), *7-3* (5 steps), and *10-1* (10 steps). Unlike PASCAL, where images typically feature atmost few classes, Cityscapes contains multiple recurring classes such as road, sky, buildings, and vehicles, making it a much more challenging dataset. From the results presented in Tab. 3, we observe that almost across all experiments, barring *10-1*, CLoRA surpasses MiB. Notably, even with joint training, CLoRA performs better compared to full fine-tuning. The results from the fine-tuning approach exhibit the lowest performance in Cityscapes compared to the other two datasets. Besides completely overwriting previous task knowledge, it fails to learn new classes. This can be attributed to the fact that the classes learned incrementally are underrepresented, and the model lacks sufficient data to learn effectively.

4.4 ADDITIONAL EXPERIMENTS

4.4.1 EXTENDING CLORA

We demonstrate the model- and approach-agnostic nature of CLoRA, which allows for the use of any distillation method to transfer knowledge between tasks. In this experiment, we employ the SegFormer (Xie et al., 2021) network and MiT-B1 with MiB (Cermelli et al., 2020) and SATS (Qiu et al., 2023) to illustrate CLoRA's adaptability. We use rank $r = 16$ which corresponds to 4.91% trainable parameters. While CLoRA exhibits maximum benefit with larger networks, this experiment highlights its suitability even for smaller networks. All previously discussed advantages of CLoRA regarding efficiency are preserved here, albeit proportionally. We repeat all tasks from the PASCAL VOC dataset (Everingham et al., 2010) and the results are presented in Tab. 4.

4.4.2 CLORA VS FROZEN ENCODER

We highlight the efficacy of CLoRA, by comparing it with a frozen encoder, and only fine-tuning the decoder for CIL, similar to transfer-learning based CL. Typically, encoders use pretrained networks on ImageNet (Deng et al., 2009) or from SAM (Kirillov et al., 2023) in our case, which holds the capability to generalize to other datasets. We repeat the PASCAL experiments with a frozen encoder. The results are presented in Tab. 1 as MiB (TL). The model fails to learn adequately even for the initial task, presumably due to the domain gap between the pretrained data and the task data, resulting in poor performance on subsequent tasks. By fine-tuning only 1% of parameters, CLoRA outperforms significantly, surpassing even full fine-tuning which uses 100% of the parameters.

4.4.3 REINITIALIZING LORA

For the initial task, the LoRA parameters A and B for all query and value matrices in the encoder are initialized to the default values and then updated through training. These updated LoRA weights are used when learning the subsequent tasks, while preserving the original pretrained weights W. In this experiment, we investigate the effects of reinitializing the LoRA weights after each task. This involves merging LoRA with the pretrained weights after learning each task t, resulting in updated weights W_t. For the subsequent task, these updated weights W_t are used instead of the original weights W, and the LoRA weights are recreated with default values. In the seventh row of Tab. 1, we observe that this approach does not significantly influence the performance, and even performs sub-optimally in certain cases.

Table 5: Performance across varying ranks for joint training. Cityscapes shows a linear performance increase with rank, PASCAL VOC results fluctuate, and ADE displays CLoRA underperforming compared to full fine-tuning.

Dataset	Ranks					Baseline
	16	32	64	96	128	
PASCAL VOC (Everingham et al., 2010)	81.50	82.66	81.53	82.85	82.03	81.69
Cityscapes (Cordts et al., 2016)	59.79	60.77	60.87	61.56	61.23	59.38
ADE20K (Zhou et al., 2017)	40.98	41.35	42.53	41.91	41.64	44.26

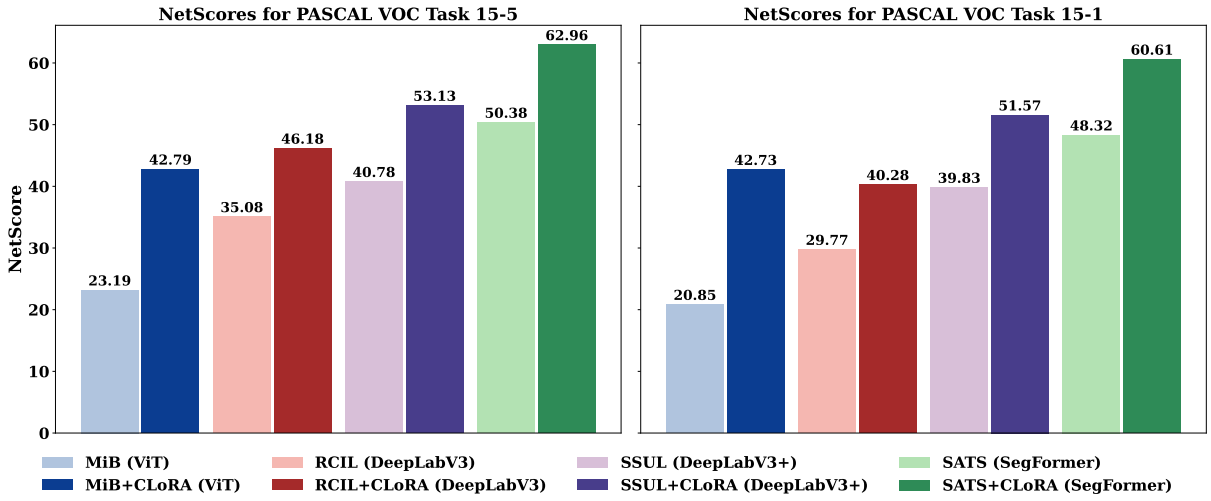

Figure 4: NetScore results across CL scenarios on PASCAL VOC (Everingham et al., 2010) using different baselines and networks. CLoRA improves NetScore substantially across network sizes, enhancing resource efficiency1-10.

4.4.4 Effect of LoRA Rank

The rank r serves as a hyperparameter that determines the number of trainable parameters. Across nearly all experiments, we observe that a rank of $r = 32$, which represents approximately 1% of the total parameters, is adequate for learning all tasks. The performance across varying ranks and for offline joint training is depicted in Tab. 5 across the three datasets. More results for incremental settings are in the appendix. We observe that for Cityscapes, the results increase linearly with an increase in rank. For PASCAL, the results seem to be saturated, with performance fluctuating across different ranks. Notably, in the case of offline training with ADE, CLoRA underperforms compared to full fine-tuning. This discrepancy could be attributed to the larger size of the dataset relative to the other two datasets.

4.5 Efficiency aspects of CLoRA

Currently, CL methods focus on task performance and the ability to retain knowledge across tasks without forgetting. However, there is a growing emphasis on the efficiency aspects of CL methods beyond forgetting for practical applicability of CL in resource constrained environments (Harun et al., 2023; Hayes & Kanan, 2022). NetScore (Wong, 2019) provides a comprehensive metric combining performance a_N, network size p_N, and the computational complexity m_N. Following previous studies (Harun et al., 2023; Hayes & Kanan, 2022; Loo et al., 2023), we also use NetScore for a holistic and effective evaluation of CL models, emphasizing the importance of resource efficiency alongside performance. The modified NetScore Ω_N for CL training is calculated as:

$$\Omega_N = 20 \log \left(\frac{a_N^\alpha}{p_N^\beta \cdot m_N^\gamma} \right) \tag{2}$$

where a_N is the final mIoU after learning all tasks, p_N is the number of parameters in millions, and m_N is the number of multiply–accumulate (MAC) operations. According to (Wong, 2019), m_N is measured during inference; however, we consider it in the training phase, since the focus of CL is updating a model. We use the default values of $\alpha = 2, \beta = \gamma = 0.5$. We evaluate the impact of CLoRA using NetScore across multiple continual learning methods

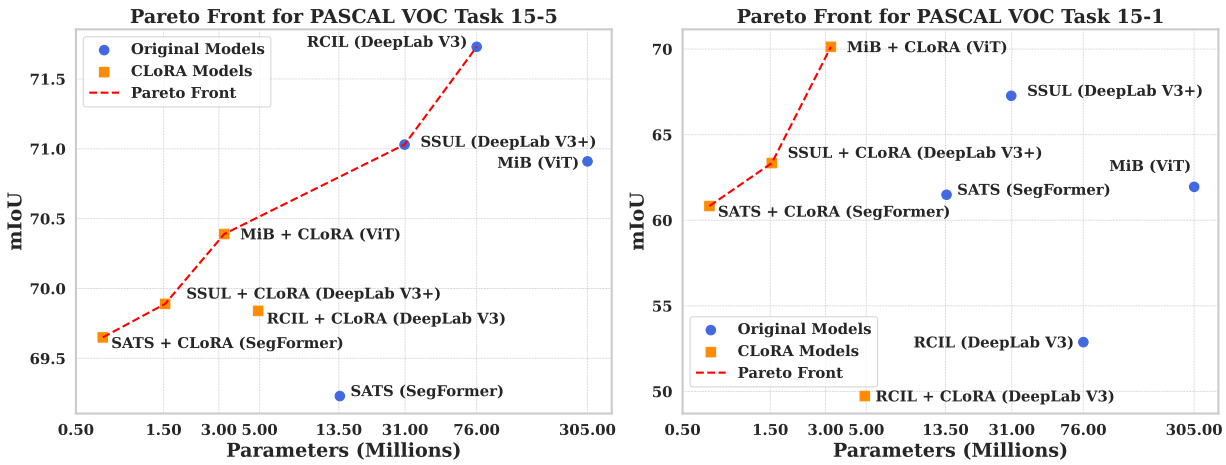

Figure 5: Pareto Front of mIoU *vs*. Trainable Parameters on PASCAL VOC (Everingham et al., 2010). The plots compare the performance-efficiency trade-off for models and the corresponding CLoRA augmented methods.

(MiB (Cermelli et al., 2020), SATS (Qiu et al., 2023), SSUL (Cha et al., 2021) and RCIL (Zhang et al., 2022)) by comparing each baseline with the corresponding CLoRA-augmented method. We demonstrate the effectiveness of CLoRA across a wide range of networks including Vision Transformer (Dosovitskiy et al., 2021), SegFormer (Xie et al., 2021), DeepLabV3 (Chen et al., 2017) and DeepLabV3+ (Chen et al., 2018). While certain continual learning methods such as SSUL (Cha et al., 2021) and RCIL (Zhang et al., 2022) are inherently efficient by updating a subset of parameters during the incremental steps, CLoRA improves the efficiency by further reducing the number of trainable parameters. To compute NetScore, the number of parameters is averaged between the full set used in the initial step and the reduced set used during incremental learning, providing a fair measure of resource efficiency.

Fig. 4 presents the NetScore results across different continual learning baselines and networks for two tasks from PASCAL VOC (Everingham et al., 2010). For the larger ViT-based network, we observe a substantial increase in the NetScore with CLoRA, highlighting its effectiveness in optimizing larger models. For the smaller networks, which already attains a relatively high NetScore, CLoRA further provides an improvement. CLoRA consistently demonstrates advantages across network sizes, enhancing resource efficiency without compromising performance relative to fully trained models. We analyse the trade-off between performance and efficiency by plotting the Pareto front of mIoU *vs*. trainable parameters for 15-5 and 15-1 task settings on PASCAL VOC (Everingham et al., 2010).

The Pareto front in Fig. 5 highlights not just the best-performing methods, but those that offer the most favourable balance between accuracy and efficiency. While RCIL achieves the highest mIoU and appears Pareto-optimal in the 15-5 setting, its CLoRA counterpart is dominated by more efficient alternatives such as SSUL+CLoRA and MiB+CLoRA, rather than by RCIL itself. Notably, across all methods, the CLoRA-augmented variants consistently improve the efficiency–performance trade-off and are never dominated by their corresponding baselines. In the 15-1 setting, both RCIL and RCIL+CLoRA are dominated, suggesting that the underlying method is the limiting factor rather than the use of CLoRA. This reinforces that while CLoRA enhances parameter efficiency, the overall performance is still bounded by the effectiveness of the baseline method.

5 Conclusion

In this work, we present Continual Learning with Low-Rank Adaptation (CLoRA), a parameter-efficient continual learning (PECL) method. CLoRA utilizes LoRA for incrementally learning new tasks, using a small fraction of parameters instead of training all the parameters. In contrast to existing PECL methods focusing on image classification with task-specific modules, we discuss the constraints in extending it to class-incremental segmentation. Addressing these challenges, we design CLoRA as a single module reused across all tasks and updated using knowledge distillation. We demonstrate the effectiveness of CLoRA, achieving results on-par, and surpassing the baselines where all parameters are updated. This introduces a novel and efficient training process for continual learning with limited resources, without sacrificing model performance. One potential limitation of CLoRA could be in handling larger datasets, although this is less of a concern in continual learning, where increments are typically small.

ACKNOWLEDGMENTS

This work was partially funded by the Federal Ministry of Education and Research Germany under the projects DECODE (01IW21001) and COPPER (01IW24009).

REFERENCES

Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Sungmin Cha, YoungJoon Yoo, Taesup Moon, et al. Ssul: Semantic segmentation with unknown label for exemplar-based class-incremental learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

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:2308.01127*, 2023.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision (ECCV)*, 2018.

Rajas Chitale, Ankit Vaidya, Aditya Kane, and Archana Ghotkar. Task arithmetic with lora for continual learning. *arXiv:2311.02428*, 2023.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *Computational Intelligence Magazine*, 2015.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Plop: Learning without forgetting for continual semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Mark Everingham, Luc van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 2010.

Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. In *International Conference on Computer Vision (ICCV)*, 2023.

Dipam Goswami, René Schuster, Joost van de Weijer, and Didier Stricker. Attribution-aware weight transfer: A warm-start initialization for class-incremental semantic segmentation. In *Winter Conference on Applications of Computer Vision (WACV)*, 2023.

Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost van de Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv:2012.0746*, 2020.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 2020.

Md Yousuf Harun, Jhair Gallardo, Tyler L Hayes, and Christopher Kanan. How efficient are today's continual learning algorithms? In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2023.

Tyler L Hayes and Christopher Kanan. Online continual learning for embedded devices. *arXiv:2203.10681*, 2022.

Niharika Hegde, Shishir Muralidhara, René Schuster, and Didier Stricker. Modality-incremental learning with disjoint relevance mapping networks for image-based semantic segmentation. In *Winter Conference on Applications of Computer Vision (WACV)*, 2025.

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 (ICML)*, 2019.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2021.

Rakib Hyder, Ken Shao, Boyu Hou, Panos Markopoulos, Ashley Prater-Bennette, and M. Salman Asif. Incremental task learning with incremental rank updates. In *European Conference on Computer Vision (ECCV)*, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.

Tobias Kalb, Masoud Roschani, Miriam Ruf, and Jürgen Beyerer. Continual learning for class-and domain-incremental semantic segmentation. In *IEEE Intelligent Vehicles Symposium (IV)*, 2021.

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. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

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*, 2017.

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 2020.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv:2101.00190*, 2021.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Chu Kiong Loo, Wei Shiung Liew, and Stefan Wermter. Explainable lifelong stream learning based on" glocal" pairwise fusion. *arXiv:2306.13410*, 2023.

Andrea Maracani, Umberto Michieli, Marco Toldo, and Pietro Zanuttigh. Recall: Replay-based continual learning in semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 2021.

Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

Shishir Muralidhara, Saqib Bukhari, Georg Schneider, Didier Stricker, and René Schuster. Cleo: Continual learning of evolving ontologies. In *European Conference on Computer Vision (ECCV)*, 2024.

Shishir Muralidhara, René Schuster, and Didier Stricker. Domain-incremental semantic segmentation for autonomous driving under adverse driving conditions. In *International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, 2025.

Youngmin Oh, Donghyeon Baek, and Bumsub Ham. Alife: Adaptive logit regularizer and feature replay for incremental semantic segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Francesco Pelosin. Simpler is better: off-the-shelf continual learning through pretrained backbones, 2022.

Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Winter Conference on Applications of Computer Vision (WACV)*, 2023.

Minh Hieu Phan, Son Lam Phung, Long Tran-Thanh, Abdesselam Bouzerdoum, et al. Class similarity weighted knowledge distillation for continual semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet K Dokania, Philip HS Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally budgeted continual learning: What does matter? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Yiqiao Qiu, Yixing Shen, Zhuohao Sun, Yanchong Zheng, Xiaobin Chang, Weishi Zheng, and Ruixuan Wang. Sats: Self-attention transfer for continual semantic segmentation. *Pattern Recognition*, 2023.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

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

Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *International Conference on Computer Vision (ICCV)*, 2021.

Marco Toldo, Umberto Michieli, and Pietro Zanuttigh. Learning with style: Continual semantic segmentation across tasks and domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Tomaso Trinci, Simone Magistri, Roberto Verdecchia, and Andrew D Bagdanov. How green is continual learning, really? analyzing the energy consumption in continual training of vision foundation models. *arXiv:2409.18664*, 2024.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv:2210.07558*, 2022.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. *arXiv:2310.14152*, 2023.

Martin Wistuba, Prabhu Teja Sivaprasad, Lukas Balles, and Giovanni Zappella. Continual learning with low rank adaptation. In *Workshops of Advances in Neural Information Processing (NeurIPS)*, 2023.

Alexander Wong. Netscore: towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge usage. In *International Conference on Image Analysis and Recognition*, pp. 15–26. Springer, 2019.

Jia-Wen Xiao, Chang-Bin Zhang, Jiekang Feng, Xialei Liu, Joost van de Weijer, and Ming-Ming Cheng. Endpoints weight fusion for class incremental semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv:2312.12148*, 2023.

Bo Yuan and Danpei Zhao. A survey on continual semantic segmentation: Theory, challenge, method and application. *arXiv:2310.14277*, 2023.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv:2106.10199*, 2021.

Chang-Bin Zhang, Jia-Wen Xiao, Xialei Liu, Ying-Cong Chen, and Ming-Ming Cheng. Representation compensation networks for continual semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Kaidong Zhang and Dong Liu. Customized segment anything model for medical image segmentation. *arXiv:2304.13785*, 2023.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. Masking as an efficient alternative to finetuning for pretrained language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision (IJCV)*, 2024.

APPENDIX

We first detail the issues with individual, task-specific modules, *i.e*. merging of predictions. We assess the robustness of a single LoRA module to handle both input and output distribution shifts. Then, we provide additional visual results for our experiments. Lastly, we perform additional experiments with varied LoRA ranks.

A MERGING TASK-SPECIFIC MODULES

Individual adapters per task are only aware of the classes they have learned. Naturally it might occur that more than one module predicts multiple non-background classes for the same pixel. In such cases, which occur especially when semantically and visually similar classes are learned in different tasks, the individual predictions need to be merged before a final class is derived. Figure 6 illustrates this challenge using PASCAL VOC (Everingham et al., 2010) tasks. In the *10-1* setting, for the class *sheep* learned in task 7, the previous modules that were trained on other animal classes predict the specific animal associated with their respective tasks. This error arises because the animal classes are visually similar, and the modules trained on earlier tasks have not encountered images of *sheep* to discriminate between *sheep* and the animal class learned in their respective tasks. Resolving the conflict is non-trivial. A naive, inferior solution could be achieved by concatenation of all logits. *I.e*. during inference, the unnormalized logits of all task-specific classifiers are concatenated and soft-maxed. Afterwards, the most probable class is selected. However, this has limitations, *e.g*. the task-wise class distributions are not calibrated (different numbers of classes per task).

B ROBUSTNESS TO DISTRIBUTION SHIFTS

This work focuses on class-incremental learning which involves sequentially learning new classes. This results in a shift in the output distribution, while the input distribution remains constant. However, in real-world scenarios, domain shift may occur due to varying weather, lighting, or geographical locations. Such a novel incremental setting with both semantic shift (new classes) and domain shift (new domains) has been previously studied by Toldo et al. (2024); Muralidhara et al. (2025). While this is not the primary focus of our work, we recognize the importance of assessing whether our approach with CLoRA, can remain effective under such shifts. To this end, we perform an experiment using Cityscapes (Cordts et al., 2016) as the base task (initial step), and ACDC (Sakaridis et al., 2021) which consists of adverse domains for the incremental tasks. Both datasets share the same set of semantic classes, which allows us to design a class-incremental learning setup while the differing visual conditions introduce an input domain shift. We use MiB (Cermelli et al., 2020) as the baseline continual learning method and report results on ACDC with and without CLoRA in Tab. 6. Notably, we use a single LoRA module across all tasks, without any task-specific adaptation for different domains. Despite the additional domain gap, CLoRA exhibits the same performance trends relative to MiB as observed in the class-incremental experiments using only Cityscapes (Cordts et al., 2016).

C ADDITIONAL VISUALIZATIONS

Figure 7 provides results of the *15-5*, *15-1*, *5-3* and *10-1* experiments on PASCAL VOC (Everingham et al., 2010). The four experiments *100-50*, *50-50*, *25-25*, and *100-10* on ADE20K (Zhou et al., 2017) are visualized in Fig. 8. Despite its increased efficiency, CLoRA appears competitive or even more detailed, compared to previous work (Cermelli et al., 2020), in all four experiments.

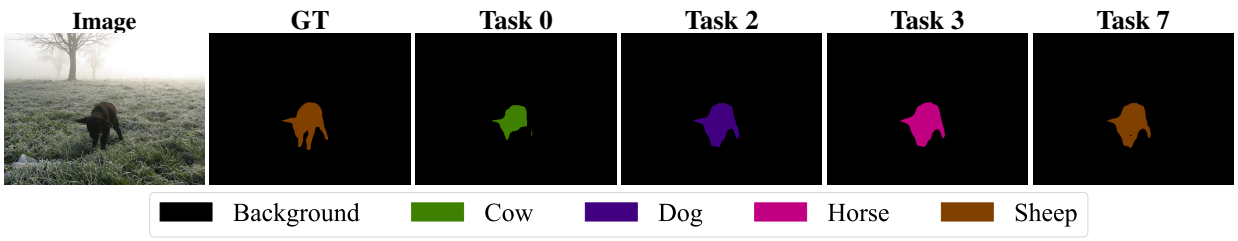

Figure 6: Conflicting predictions from task-specific modules on the PASCAL VOC (Everingham et al., 2010) dataset using the *10-1* setting, in which multiple modules have conflicting predictions.

Table 6: Results on class-incremental learning with varying input domains using Cityscapes (Cordts et al., 2016) and ACDC (Sakaridis et al., 2021).

Method	14-5			14-1			7-3			10-1		
	1-14	15-19	**All**	1-14	15-19	**All**	1-7	8-19	**All**	1-10	11-19	**All**
MiB	48.11	02.08	36.00	48.31	04.05	36.66	42.15	19.89	28.09	44.09	18.46	**31.95**
CLoRA	47.96	16.89	**39.79**	49.06	03.75	**37.14**	47.16	26.05	**33.83**	43.71	16.67	30.90

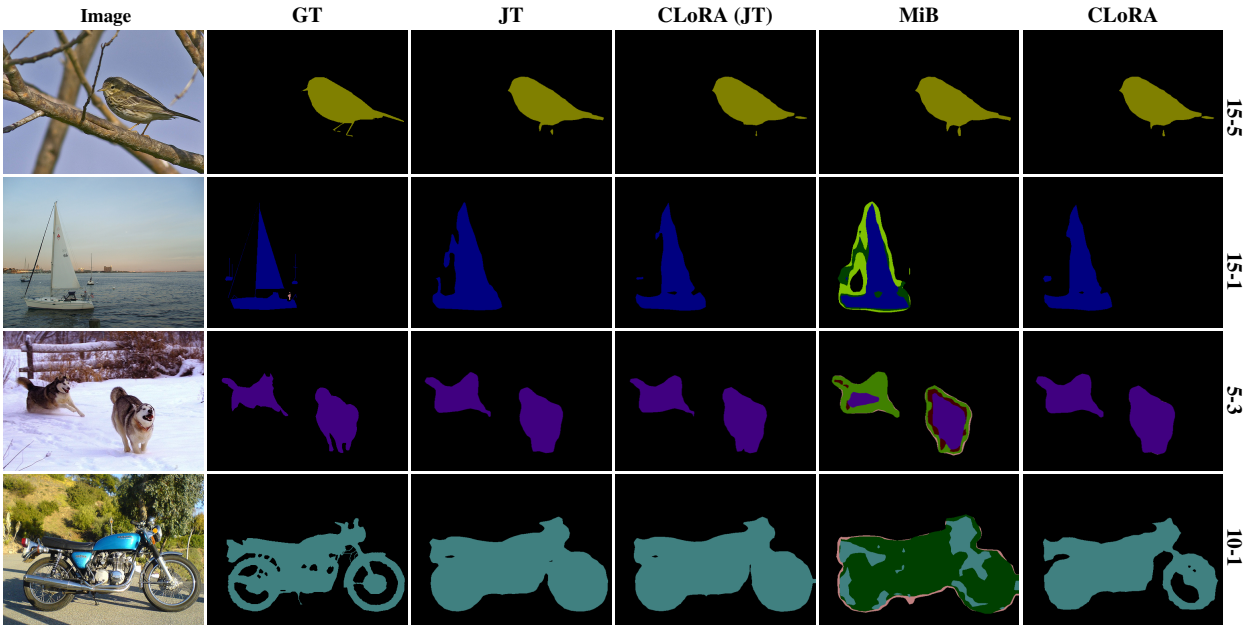

Figure 7: Qualitative visualizations from the PASCAL VOC (Everingham et al., 2010) dataset.

D Effect of LoRA Rank

Table 7 presents the results for the PASCAL VOC (Everingham et al., 2010) tasks for different ranks of LoRA (Hu et al., 2021). The rank is a hyperparameter that influences the number of trainable parameters. In all our main experiments with the ViT-based network, we use a rank $r = 32$ which corresponds to $\sim$1% of trainable parameters. In Sec. 4.4.4 of the main paper, we study the influence of varying ranks in the offline setting (joint training) across the three datasets. Here, we explore the impact of different ranks on performance in more detail in different continual learning settings of the PASCAL VOC dataset (Everingham et al., 2010). We observe that for rank $r = 64$, the results improve consistently across all tasks. For longer task sequences, such as *5-3* and *10-1*, higher ranks yield better results. However, since task sequence lengths are typically unknown and higher ranks entail greater computational costs, we choose $r = 32$ as a balanced configuration for all experiments.

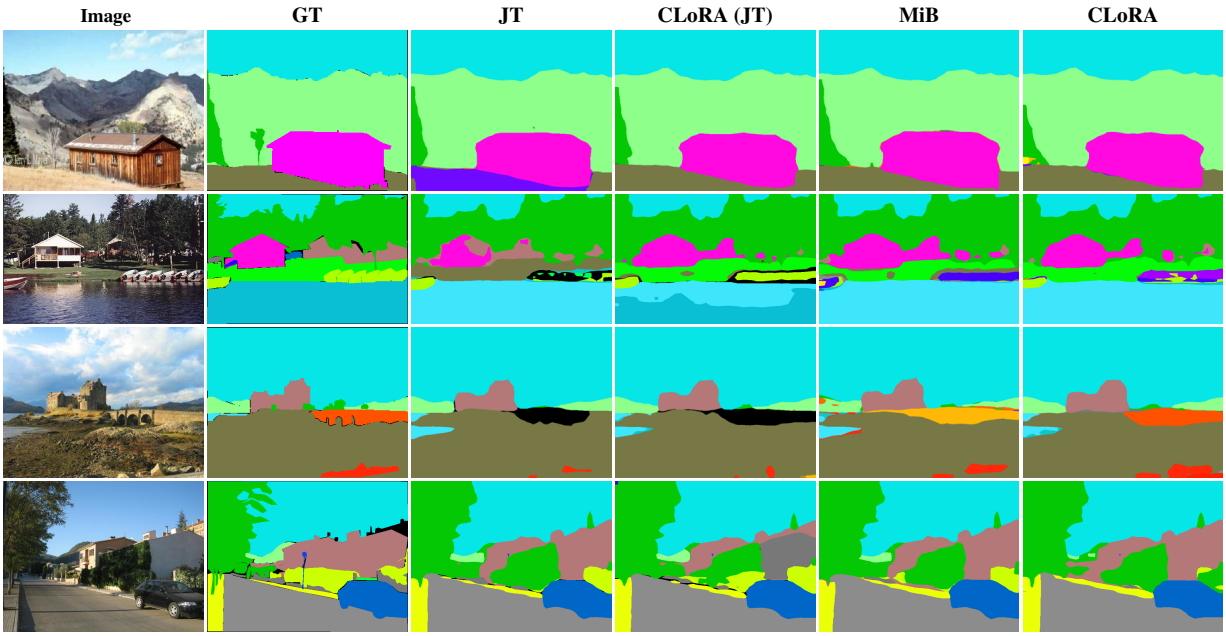

Figure 8: Qualitative visualizations from the ADE20K (Zhou et al., 2017) dataset.

Table 7: Results of CLoRA on PASCAL VOC (Everingham et al., 2010) dataset with varying ranks for LoRA (Hu et al., 2021) after learning all tasks.

Rank	15-5			15-1			5-3			10-1		
	0-15	16-20	**All**	0-15	16-20	**All**	0-5	6-20	**All**	0-10	11-20	**All**
16	74.87	55.82	70.34	80.71	34.37	69.67	67.06	47.32	52.96	28.05	26.23	27.19
32	74.17	56.57	70.39	81.29	34.41	70.13	69.92	45.50	52.47	31.38	29.22	30.35
64	79.62	62.31	**75.50**	82.06	34.08	**70.63**	71.46	45.70	53.06	47.62	33.71	**41.00**
96	74.45	56.30	70.13	77.18	29.43	65.82	69.92	50.08	**55.75**	48.15	30.97	39.97
128	75.39	57.42	71.11	81.01	31.76	69.28	66.21	50.66	55.10	42.57	32.61	37.83