SimCS: Simulation for Domain Incremental Online Continual Segmentation

Motasem Alfarra[1,2], **Zhipeng Cai**[1], **Adel Bibi**[3], **Bernard Ghanem**[2], **Matthias Müller**[1]
[1]Intel Labs
[2]King Abdullah University of Science and Technology (KAUST)
[3]University of Oxford
motasem.alfarra@kaust.edu.sa

Abstract

Continual Learning is a step towards lifelong intelligence where models continuously learn from recently collected data without forgetting previous knowledge. Existing continual learning approaches mostly focus on image classification in the class-incremental setup with clear task boundaries and unlimited computational budget. This work explores the problem of Online Domain-Incremental Continual Segmentation (ODICS), where the model is continually trained over batches of densely labeled images from different domains, with limited computation and no information about the task boundaries. ODICS arises in many practical applications. In autonomous driving, this may correspond to the realistic scenario of training a segmentation model over time on a sequence of cities. We analyze several existing continual learning methods and show that they perform poorly in this setting despite working well in class-incremental segmentation. We propose SimCS, a parameter-free method complementary to existing ones that uses simulated data to regularize continual learning. Experiments show that SimCS provides consistent improvements when combined with different CL methods.

1 Introduction

Supervised learning has been the go-to solution for many computer vision problems (He et al. 2016; Ren et al. 2015). The large scale of available labeled data has been the key factor for its success (Radford et al. 2021). However, in many settings the training data is not available all at once but generated sequentially over time. Moreover, the distribution of the training data may vary gradually over time (Cai, Sener, and Koltun 2021; Lin et al. 2021), *e.g.* , images taken in winter with rain and snow versus images with clear skies taken in the summer. Naively applying supervised learning in such a setting suffers from *catastrophic forgetting* (Kirkpatrick et al. 2017), *i.e.* , training a model on new data of a different distribution worsens its performance on old data. Continual learning (CL) attempts to address these issues by designing algorithms that operate on continuous data streams and efficiently adapt to new data while retaining previous knowledge. However, in the existing CL literature (Li and Hoiem 2017; Chaudhry et al. 2018), methods are usually evaluated only on restricted problems such as image classification with carefully crafted data streams that assume non-overlapping tasks, *e.g.* , the class-incremental setting where each task corresponds to a fixed set of classes.

This work studies a more realistic problem of **O**nline **D**omain-**I**ncremental **C**ontinual Learning for Semantic **S**egmentation (ODICS). ODICS is essential for many applications where the perception system needs to be updated over time. In this setting, the model is trained with a *limited computation and memory budget* at each time step on data sampled from a varying distribution (Ghunaim et al. 2023). The variation comes from domain shifts, *e.g.* , data coming from a different environment; the model has *no information about the domain boundaries*. This setup mimics the practical scenario where labeled data from new scenes (weather conditions, cities, *etc.*) are generated continually over time, *e.g.* , when developing a segmentation system for autonomous driving, last-mile delivery, or other robotics applications. The goal is to continually train the model (on the data center) with limited budget to enable frequent updates (*e.g.* , once a day for self-driving cars) of deployed models.

Despite the importance of this problem, it has received little attention in recent years. The few prior arts (Douillard et al. 2021; Maracani et al. 2021) for continual semantic segmentation study the problem under two unrealistic assumptions. First, it is assumed that the deployed model is aware of the *domain boundaries* (Garg et al. 2022), *i.e.* , the domain change, during both training and testing. While this simplifies the problem, domain boundaries are often not available in real-world applications as the transition between different domains is usually smooth or unknown. Second, the model is permitted to make any number of training iterations over current domain data, *i.e.* , learning with unlimited computational budget (Douillard et al. 2021; Garg et al. 2022; Maracani et al. 2021). This means that the model can pause the stream from revealing new data during training, while in realistic setups, streams continuously and uninterruptedly reveal new data and remain agnostic to the training status of the model (Cai, Sener, and Koltun 2021; Alfarra et al. 2023).

Moving closer to practical scenarios, we study the problem of online, *i.e.* limited computational budget, domain-incremental continual learning for semantic segmentation. We propose a new benchmark using public datasets captured from different cities and different weather conditions and order them based on acquisition time. We find that the domain

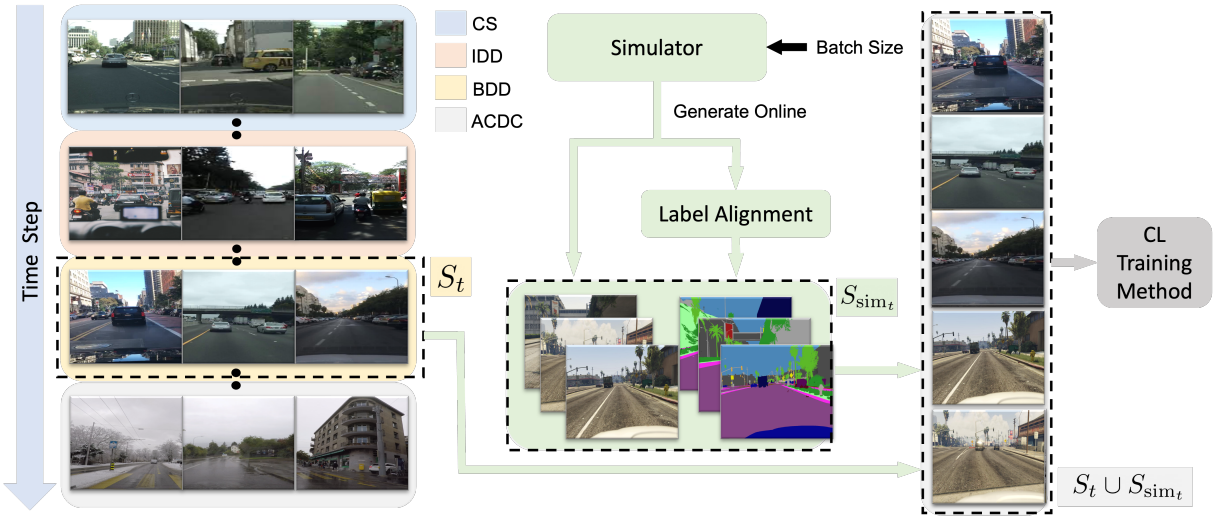

Figure 1: Online Domain Incremental Continual Segmentation (ODICS) with Simulated Data (SimCS). At each time step t, ODICS reveals a batch of labeled images S_t with size B_t from a certain domain, where different domains are presented sequentially to the model. SimCS generates a batch with size B_t of simulated data S_{sim_t} on the fly and aligns its label space to the real stream. The concatenated batch of real and simulated data is presented to the model to aid continual training and to mitigate forgetting previously learnt domains.

shift in this benchmark is severe enough to cause forgetting on previously learned domains even without introducing new classes during training. We benchmark regularization-based methods (Kirkpatrick et al. 2017) that are effective in mitigating forgetting in class-incremental continual segmentation (Douillard et al. 2021) and show that they fail in ODICS. Meanwhile, although replay-based methods (Chaudhry et al. 2019) can effectively mitigate forgetting, they may not be feasible due to privacy concerns, *e.g.* , GDPR (Commission 2021). This is particularly a concern when data is associated with different countries in which storing it for replay is not permissible. Thus, we propose SimCS that uses photo-realistic simulated data with "free" dense labels which can be generated on the fly during ODICS (see Figure 1) to mitigate forgetting without violating privacy constraints.

Our contributions are three-fold. **(1)** We propose online domain-incremental continual learning for semantic segmentation (ODICS) and construct a corresponding new benchmark evaluating several baselines from the literature. **(2)** We propose SimCS, a method that uses simulation as a continual learning regularizer. SimCS is parameter-free and *orthogonal* to existing continual learning frameworks. We demonstrate its effectiveness by combining it with five different continual learning strategies showing performance improvements across the board. **(3)** We conduct a comprehensive analysis, showing that SimCS is robust to the choice of simulator, hyperparameters, and budget constraints in CL.

2 Related Work

Continual Learning. The main challenge in learning a sequence of tasks, classes, or domains from a continuous stream of data is catastrophic forgetting (Wu et al. 2019; Rebuffi et al. 2017). Existing methods can be broadly categorized into two groups, using either regularization or memory. *Regularization-based methods* add regularization terms to the training objective without using previous data. The goal is to maintain parameters that are important to remember previous knowledge (Kirkpatrick et al. 2017; Aljundi et al. 2018; Li and Hoiem 2017). Besides explicit regularization on model parameters (Kirkpatrick et al. 2017; Aljundi et al. 2018), distilling the predictions of older models is also widely used (Li and Hoiem 2017). *Memory-based methods* store historical data it in a replay buffer (Lopez-Paz and Ranzato 2017; Aljundi et al. 2019). This data is then used to regularize the gradient of the optimizer (Chaudhry et al. 2018) or directly mixed with new training data (Chaudhry et al. 2019). Most existing literature focuses on the class-incremental setup. In this work, we evaluate the current progress in continual learning for semantic segmentation in the online domain-incremental setup.

Continual Semantic Segmentation. Recently, class-incremental learning was extended from image classification to semantic segmentation. (Michieli and Zanuttigh 2019; Cermelli et al. 2020). This was done by presenting segmentation masks of the classes belonging to a given task while treating the remaining classes as the background (Douillard et al. 2021; Maracani et al. 2021). More closely related to our work, multi-domain incremental learning was analyzed in the semantic segmentation task (Garg et al. 2022). Despite the current progress, previous methods assume knowledge of task boundaries at test time and unlimited computational budget for training on each task (Garg et al. 2022). There are many practical applications like self-driving cars, where new data is generated constantly at a high data rate (Cai, Sener, and Koltun 2021) and without clear distinction between different tasks (Bang et al. 2022). To better study these scenarios, we propose the setup of online domain incremental con-

tinual learning for semantic segmentation (ODICS), where a new batch of data arrives at each time step and the model is only allowed limited computation on each batch.

Simulators for Semantic Segmentation. Recent works have proposed several simulators that generate fully annotated data for "free" such as CARLA (Dosovitskiy et al. 2017) and VIPER (Richter, Hayder, and Koltun 2017; Richter et al. 2016). Such simulators play a key role for applications like autonomous driving (Blaga and Nedevschi 2019) and visual navigation (Li et al. 2020), where collecting and annotating data is expensive and time consuming. Nonetheless, the use of simulated data in continual learning remains unexplored. In this work, we leverage simulated data to reduce forgetting in continual learning.

3 Online Domain-Incremental Continual Segmentation (ODICS)

In ODICS, a parametrized model $f(\cdot|\boldsymbol{\theta})$ that maps an image $\mathbf{X} \in \mathcal{X}$ to a per-pixel class prediction $\mathbf{Y} \in \mathcal{Y}$ is trained. At each time step $t \in \{1, 2, 3, ..., \infty\}$, a batch of densely labeled images $S_t = \{\mathbf{X}_{i_t}, \mathbf{Y}_{i_t}\}_{i_t=1}^{B_t} \sim \mathcal{D}_t$ is revealed. Then, the model parameters $\boldsymbol{\theta}_t$ are updated using S_t and a limited computation budget before $t+1$. Unlike in supervised learning where the domain $\mathcal{D}_t$ does not change, $\mathcal{D}_t$ may change drastically in ODICS during training. The goal of ODICS is to obtain the model parameters $\boldsymbol{\theta}_t$ that perform well on all previously seen domains, *i.e.* , $\mathcal{D}_1$ to $\mathcal{D}_t$. There are two key concepts in ODICS: online and domain-incremental. *Online* refers to the limited computation budget, *i.e.* , we cannot train a model from scratch within each t. This is important for applications like autonomous driving where new data is continuously revealed over time. *Domain-incremental* refers to the fact that the label space $\mathcal{Y}$ remains constant throughout training, *i.e.* , only the distribution of $\mathbf{X}_{i_t}$ and the ratio of different classes in $\mathbf{Y}_{i_t}$ change over time. While ODICS is relevant for several applications, no benchmarks exist for this setup. To this end, we propose a first attempt in constructing such a benchmark. We focus on outdoor semantic segmentation in the context of self-driving cars where the domain shift can come from different weather conditions, cities, or cameras. To mimic practical scenarios, we construct the stream of data $\{S_t\}_{t=1}^{\infty}$ from multiple domains by composing four different standard benchmarks from the literature: CityScapes (CS) (Cordts et al. 2016), Indian Driving Dataset (IDD) (Varma et al. 2019), Berkeley Driving Dataset (BDD) (Yu et al. 2018), and Adverse Weather Condition Dataset (ACDC) (Sakaridis, Dai, and Van Gool 2021); we treat each dataset as a different domain. Note that each dataset was collected in a different country. CS was collected in Germany, IDD in India, BDD in the United States, and ACDC in Switzerland mimicking the realistic scenario of deploying models in different locations. This diversity introduces a notion of domains based on geographical location and weather conditions. For example, CS contains images with clear weather conditions while ACDC has a variety of adverse weather conditions, *e.g.* , fog and rain. This adds another realistic aspect to our setup, since deployed models experience such adverse conditions when deployed throughout the year.

CS is used as reference for a consistent label space across domains, *i.e.* , an identical set of classes. We construct the stream by concatenating all domains based on the year the dataset was published, resulting in the following order: CS (2016) - IDD (2019) - BDD (2020) - ACDC (2021), which mimics the nature of continual learning where data generated earlier will be seen by the model first. For completeness, we analyze the use of different domain orders. As typical in CL, we evaluate the model trained on the stream on a held out test set from each domain.

4 Methodology

4.1 Continual Learning Strategies

We start with the scenario where at any time step t the model cannot store, hence rehearse, any data from previous time steps (1 to $t-1$). This captures the realistic constraint where data is subject to privacy restrictions (*e.g.* GDPR).

The simplest baseline in this case is applying the same optimization strategy as in supervised learning at each time step, which we call *naive training* in this paper. Specifically, given the training data $S_t = \{\mathbf{X}_{i_t}, \mathbf{Y}_{i_t}\}$ at time step t, we update the model by optimizing the following objective:

$$\min_{\boldsymbol{\theta}_t} \sum_{i_t} \mathcal{L}(f(\mathbf{X}_{i_t}|\boldsymbol{\theta}_t), \mathbf{Y}_{i_t}), \tag{1}$$

where $\mathcal{L}(\cdot)$ is the standard loss for semantic segmentation, *e.g.* , cross entropy. In the online setting, we apply a limited number of (stochastic) gradient descent steps on $\boldsymbol{\theta}_t$.

Regularization-based methods (Kirkpatrick et al. 2017; Li and Hoiem 2017) are a family of continual learning methods extensively studied in image-classification. Instead of optimizing (1), these methods update the model by optimizing:

$$\min_{\boldsymbol{\theta}_t} \sum_{i_t} \mathcal{L}(f(\mathbf{X}_{i_t}|\boldsymbol{\theta}_t), \mathbf{Y}_{i_t}) + \lambda \mathcal{L}_{\text{reg}}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-k}, S_t), \tag{2}$$

where $\mathcal{L}_{\text{reg}}(\cdot)$ is the regularization term used to mitigate forgetting, which is algorithm-specific, and λ is a coefficient that controls the regularization strength. Note that only data from the current step S_t and possibly cached historical models $\boldsymbol{\theta}_{t-k}$ are used for regularization; no historical data is used for optimization. We also consider relaxing the constraint on storing old data and complement our benchmark by comparing against *replay-based methods* (Chaudhry et al. 2019). In particular, we allow the model to store a few historical training samples in a small replay buffer. During training, we optimize:

$$\min_{\boldsymbol{\theta}_t} \sum_{i_t} \mathcal{L}(f(\mathbf{X}_{i_t}|\boldsymbol{\theta}_t), \mathbf{Y}_{i_t}) + \mathcal{L}_{\text{rep}}(\boldsymbol{\theta}_t, S_{\text{rep}_t}), \tag{3}$$

where $\mathcal{L}_{\text{rep}}(\boldsymbol{\theta}_t, S_{\text{rep}_t})$ computes the loss on a batch of data S_{rep_t} sampled from the replay buffer at step t. In the simplest form (Chaudhry et al. 2019), $\mathcal{L}_{\text{rep}}(\cdot)$ is the same as $\mathcal{L}(\cdot)$ but computed on S_{rep_t}. Despite its simplicity, this approach is effective in mitigating forgetting for image classification (Prabhu, Torr, and Dokania 2020).

4.2 Continual Learning with Simulation

In practice, naive training or regularization-based methods are often not effective enough for continual learning due to the strongly biased training data. Although replay-based methods are more effective, they are less practical under privacy or memory constraints. To address this problem, we take an orthogonal and unexplored path, which is using simulation data for continual learning.

Simulation techniques have achieved impressive advancements recently, especially for computer vision. For autonomous driving, state-of-the-art simulators (Dosovitskiy et al. 2017; Richter, Hayder, and Koltun 2017) can generate densely labeled images of simulated driving scenes on the fly. Using simulation for continual learning has several advantages. First, we can obtain an infinite amount of high-quality, diverse, and densely labeled images on the fly by running the simulator; we do not need to store a large amount of data in memory. Second, since all data are synthetic, privacy constraints will not be violated. Inspired by replay-based methods, we propose to use the loss on simulation data as a regularization for continual learning. As shown in Figure 1, at each time step of ODICS, we first generate a batch of labeled simulation data S_{sim_t} on the fly. Then, we update the model by optimizing the following objective:

$$\min_{\boldsymbol{\theta}_t} \sum_{i_t} \mathcal{L}(f(\mathbf{X}_{i_t}|\boldsymbol{\theta}_t), \mathbf{Y}_{i_t}) + \mathcal{L}_{\text{sim}}(\boldsymbol{\theta}_t, S_{\text{sim}_t}). \quad (4)$$

We set $\mathcal{L}_{\text{sim}}(\boldsymbol{\theta}_t, S_{\text{sim}_t}) = \sum_{\mathbf{X}_j, \mathbf{Y}_j \in S_{\text{sim}_t}} \mathcal{L}(f(\mathbf{X}_j|\boldsymbol{\theta}_t), \mathbf{Y}_j)$, *i.e.* , we compute the same loss on both real and simulated data, and sum them together. While more complex strategies can be applied, we found that this simple approach is effective as later shown in the experiments. We call this method Simulation for Continual Segmentation (*SimCS*). Note that the formulation in Eq. (4) does not make any assumptions on $\mathcal{L}$. That is, one can combine SimCS with regularization methods by replacing $\mathcal{L}$ with the combined loss in Eq. (2), or combine it with the replay approach in Eq. (3). This positions SimCS as an orthogonal approach to existing methods in the CL literature. A few challenges need to be addressed to make SimCS general and effective.

Robustness to simulators. It is unclear whether our method can be robust to different simulators with different rendering quality, scene scale, and objects. To address this question, we use two different simulators, *i.e.* , CARLA (Dosovitskiy et al. 2017) and VIPER (Richter, Hayder, and Koltun 2017).

Label Space Alignment. There are many options for defining class labels leading to misalignment between different simulators and real-world datasets. For example, CARLA has a single *vehicle* class but separate classes for *road line* and *road*. Meanwhile, the real-world datasets have separate classes for *cars* and *trucks* but only a single class for *road*. Hence, we merge or relabel the segmentation masks generated by the simulator of choice to achieve the maximal overlap with the label space of the real data. Then, we drop all other labels as opposed to merging them into the background class. The full details of relabeling the simulated data can be found in the appendix. As shown in our experiments, though some of the real-world classes are missing in the simulated data due to non-overlapping label spaces, our approach is still effective. We expect that SimCS has further potential when applied to more advanced simulators.

Data Quantity. It is also not clear how much simulation data is needed for continual learning. Intuitively, using a large amount of simulated data could bias the model to only perform well on the simulated data, while using a small amount of data may only improve performance marginally. At each training iteration of SimCS, the batch of simulated data is generated by randomly setting simulator parameters, *e.g.* camera position, weather, time and traffic conditions. To study the impact of the amount of simulated data on performance, we explore varying the ratio between simulated and real data during training. In the main experiments, we set the sim-real ratio to 1, which provides a good trade-off between computation and performance improvement.

5 Experiments

Experimental Setup. We construct our benchmark by concatenating four different datasets as domains, namely CS, IDD, BDD, and ACDC, as mentioned in Section 3. Throughout, we use the term "domain" and "dataset" interchangeably. Following common practice in semantic segmentation (Douillard et al. 2021), we use 80% of the publicly available data from each dataset for training and evaluate on the 20% held out test set from each domain. We follow standard practice (Douillard et al. 2021; Maracani et al. 2021) in reporting the mean Intersection over Union (mIOU) on the held out test set from each domain. During our experiments, at each time step t of ODICS, the model is presented with a batch of real images S_t of size 8, *i.e.* $B_t = 8\ \forall t$. Before the next time step $t+1$, the model is allowed to train on the batch using a fixed computational budget, measured by the number N of forward and backward passes. Unless stated otherwise, we set $N = 4$ throughout our experiments[1]. Once data from S_{t+1} is revealed, the older batch S_t becomes unavailable to the model unless replay is used. We evaluate all methods using the benchmark introduced in Section 3. In our experiments, we use the DeepLabV3 architecture (Chen et al. 2017) pre-trained on ImageNet (Deng et al. 2009) (unless otherwise stated in the pre-training experiments in Section 5.2), following (Douillard et al. 2021). We utilized 2 NVIDIA V100 for each of our experiments. Further details are provided in the appendix.

We analyze five different types of training strategies. The baseline is *Naive Training (NT)*, *i.e.* , optimizing Eq. (1). We also consider regularization-based (Eq. (2)) and replay-based (Eq. (3)) CL algorithms. For regularization-based methods, we consider *Elastic Weight Consolidation (EWC)* (Kirkpatrick et al. 2017), *Memory Aware Synapses (MAS)* (Aljundi et al. 2018), and *Learning without Forgetting (LwF)* (Li and Hoiem 2017). We do not provide boundaries of dataset transitions during training; we make

[1]We found empirically that setting $N = 4$ provides a good trade-off between preventing the model from under-fitting and significantly increasing the computation. In the appendix we provide results for different choices of N with similar conclusions.

Method \ Domain	CS	IDD	BDD	ACDC	mIOU
NT	40.1	37.9	35.1	48.9	40.5
+ CARLA	44.6	39.6	38.5	51.0	43.4
+ VIPER	45.4	43.8	40.0	50.4	44.9
EWC	41.5	38.8	35.9	47.9	41.0
+ CARLA	45.3	40.5	38.8	51.3	44.0
+ VIPER	45.1	43.4	40.9	50.9	45.1
MAS	41.4	37.1	34.6	48.2	40.3
+ CARLA	46.7	41.3	38.3	50.2	44.1
+ VIPER	45.8	43.5	38.8	49.1	44.3
LwF	44.5	41.9	34.6	46.3	41.8
+ CARLA	47.1	44.3	39.0	48.5	44.7
+ VIPER	46.7	46.7	38.5	47.9	45.0
ER	47.4	47.8	40.9	48.8	46.2
+ CARLA	48.4	48.5	43.2	50.8	47.7
+ VIPER	48.5	50.0	42.5	52.0	48.3
Supervised	62.7	63.6	49.6	62.0	59.5
+ CARLA	62.8	63.7	49.7	62.9	59.7
+ VIPER	63.3	63.9	49.2	63.1	59.8

Table 1: Performance Comparison under ODICS. We report the mIOU (%) of a model trained on our benchmark and evaluated on each domain in the benchmark. We also report the performance of SimCS-enhanced baselines by leveraging either CARLA or VIPER. All methods are trained with $N = 4$ iterations for each received batch. The last row "Supervised" represents the performance of a model trained on the entire stream for 30 epochs as a surrogate to upper bound performance. *SimCS consistently improved the performance of all baselines on all observed domains.*

an exception for regularization-based methods, since this information is crucial to achieve reasonable performance according to our empirical results. For each considered regularizer, we set $\lambda = 0$ in Eq. (2) when training on data from the first domain and $\lambda > 0$ for the other domains. We report the best results for each regularizer cross-validated on different values of λ leaving the result for all values of λ to the appendix. For replay-based methods, we apply Experience Replay (ER) (Chaudhry et al. 2019) with a replay buffer size of 800 images (along with their dense labels), throughout this section and leave the ablations to the appendix.

We explore simulated data generated from CARLA and VIPER (Dosovitskiy et al. 2017; Richter, Hayder, and Koltun 2017) with our SimCS approach. We generate simulated data on the fly in the most realistic town 10 of CARLA by randomly setting the location and camera parameters. On the other hand, with VIPER, we sample (without replacement) from a large pool of the publicly available pregenerated simulated data, since the code to generate data on the fly is not available. We relabel the segmentation masks of the simulated data to align with the labels of the real world following the procedure described in Sec. 4.2. This results in 13 and 15 out of 19 overlapping classes between the simulated and real data for CARLA and VIPER, respectively.

Method \ Domain	CS	IDD	BDD	ACDC	mIOU
NT	40.1	37.9	35.1	48.9	40.5
+ VIPER Pretrain	40.2	40.4	36.5	51.9	42.3
+ VIPER SimCS	47.9	43.0	41.8	54.2	46.7

Table 2: Performance Comparison under VIPER Pretraining. We compare the performance of NT when pretrained with VIPER (on top of ImageNet). We further boost NT + VIPER pretraining with SimCS (with VIPER) during continual learning. *VIPER pretraining boosted the performance of both NT and NT+SimCS.*

5.1 Main Results

We start by analyzing the performance of different CL training strategies in ODICS. Table 1 summarizes the results of a model after being trained on our benchmark and evaluated on each observed domain, where the last column reports the mIOU across all domains. The last row (Supervised) reports the performance of a model trained for 30 epochs using standard supervised learning on all data of the stream, representing a surrogate upper-bound performance.

Unlike the class-incremental setup (Douillard et al. 2021), the simple NT in ODICS enjoys an on-par performance to all considered regularization-based methods. For example, while MAS outperforms NT on earlier domains, *e.g.* , CS, the overall performance degrades to 40.3% compared to 40.5% mIOU for NT. The most effective regularization-based method is LwF, which only outperforms NT by 1.3%. This suggests that further work is needed to develop regularization techniques for this more realistic domain-incremental setup. Meanwhile, rehearsing previously seen examples through ER consistently outperforms other baseline methods in all domains. This conclusion is consistent with previous results in image classification (Lin et al. 2021; Lopez-Paz and Ranzato 2017; Chaudhry et al. 2019), as storing real examples in a replay buffer provides a simple but effective regularization for Continual Learning (CL).

Next, we analyze the effectiveness of including simulated data to all considered training strategies. To apply our approach on methods other than NT, we simply add $l_{\text{sim}}(\boldsymbol{\theta}_t, S_{\text{sim}_t})$ in Eq. 4 to the objective of each method. As shown in Table 1, across *all* domains and *all* considered training schemes, SimCS provides consistent and significant performance improvements. For example, adding VIPER to the CL schemes reduces forgetting of NT and LwF on CS and IDD, respectively, by $\sim 5\%$ (from 40.1 to 45.4 and from 41.9 to 46.7). Further, leveraging simulated data improves the strongest baseline (ER) by a notable 2%. This result shows that simulation data can be leveraged as an effective regularizer for mitigating forgetting in CL. Moreover, different simulators provide different margins of improvements. For instance, while using either simulator (CARLA or VIPER) improves performance, simulated data generated from VIPER often produces larger gains. This observation can be attributed to several factors. For example, different simulators vary in photo-realism; in addition, their labels may be more or less aligned with real-word data labels.

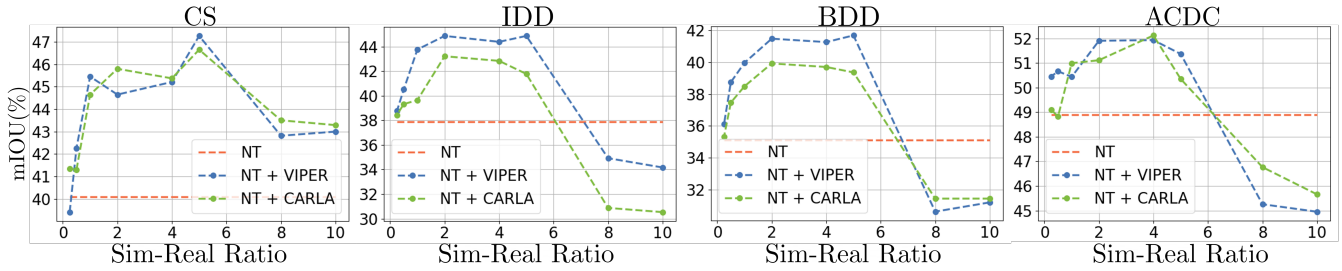

Figure 2: Effect of Varying Sim-Real Ratio on the Performance Gain. We analyze the effect of varying the ratio between simulation and real data from $\{1/4, 1/2, 1, 2, 4, 5, 8, 10\}$ on the performance gain for each observed domain. We find that SimCS provides notable performance improvement on a wide range of ratios (≤ 5).

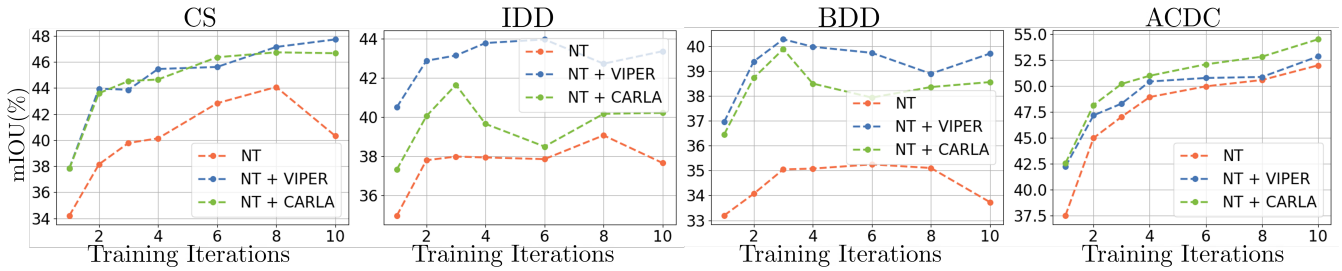

Figure 3: Comparison under different computational budgets. We allow NT, NT+CARLA, and NT+VIPER different computational budgets for training on each received batch from the stream, measured by the number of training iterations. We measure the performance on each observed domain when varying the budget to $\{1, 2, 3, 4,6,8,10\}$ training iterations.

5.2 Pre-training on Simulated Data

In addition to using simulated data as a regularizer within the ODICS setup, we try incorporating it in pre-training, since it can be made available even before the start of the continual learning process. To that end, and before commencing with ODICS, we first fine-tune ImageNet pretrained models with data generated from VIPER. We generate 17K synthetic images, which is equal to the total number of real images presented in the continual setup, and train our model for 30 epochs on this simulated data. We then perform ODICS and compare against NT and NT + VIPER, where in the latter VIPER is used as a regularizer during ODICS.

We report the results in Table 2. We observe that pre-training on simulated data further improves the performance of a continual learner on all observed domains. We report an improvement of 1.7% on average across all observed domains when compared to ImageNet pre-training. Although our model observed data generated from VIPER in the pre-training phase, including simulated data in the continual learning process further enhances the performance by 4% on average. It is worth mentioning that this boosted version of NT surpasses the performance of the best baseline ER by 0.5% on average, without the need to store any new additional data from previous domains during continual learning.

5.3 Impact of the Amount of Simulation Data

In Sections 5.1 and 5.2, we used a 1:1 ratio between simulated and real data to form a mini-batch during continual learning. We analyze the effect of varying this sim-real ratio on performance in Figure 2, where we report the performance of NT, NT+CARLA, and NT+VIPER with ratios of $\{1/4, 1/2, 1, 2, 4, 5, 10\}$. The results show that leveraging simulated data provides consistent performance improvements for a wide range of sim-real ratios. As long as this ratio was smaller than 5, *i.e.* , we generate 5 batches from the simulator for each received batch from the real-world stream, our approach provides significant gains across all observed domains. However, for larger sim-real ratios, *e.g.* 10, the training is biased towards simulated data and thus harms the performance on the real-world stream. This is exemplified in Figure 2, where SimCS with sim-real ratios ≥ 8 degrades the mIOU of NT on 3 out of 4 domains. Further, VIPER outperforms CARLA across most sim-real ratios; this is consistent with previous observations in Table 1.

5.4 Impact of the Computational Budget

In Section 5.1, all methods are given a fixed computational budget of $N = 4$ forward and backward passes for each received batch. In this section, we analyze the performance with different computational budgets. We conduct experiments with $N \in \{1, 2, 3, 4, 6, 8, 10\}$ for NT, NT+CARLA, and NT+VIPER, and report results on each observed domain in Figure 3. We observe that small computational budgets might result in an under-fitting model while larger budgets ($N = 10$) cause the model to over-fit to the last domain, thus, increasing forgetting previous domains. Nonetheless, SimCS provides a stable performance gain across all considered budgets irrespective of the choice of the simulator.

Moreover, and in contrast to prior CL literature, we perform comparisons for when the computational budget is nor-

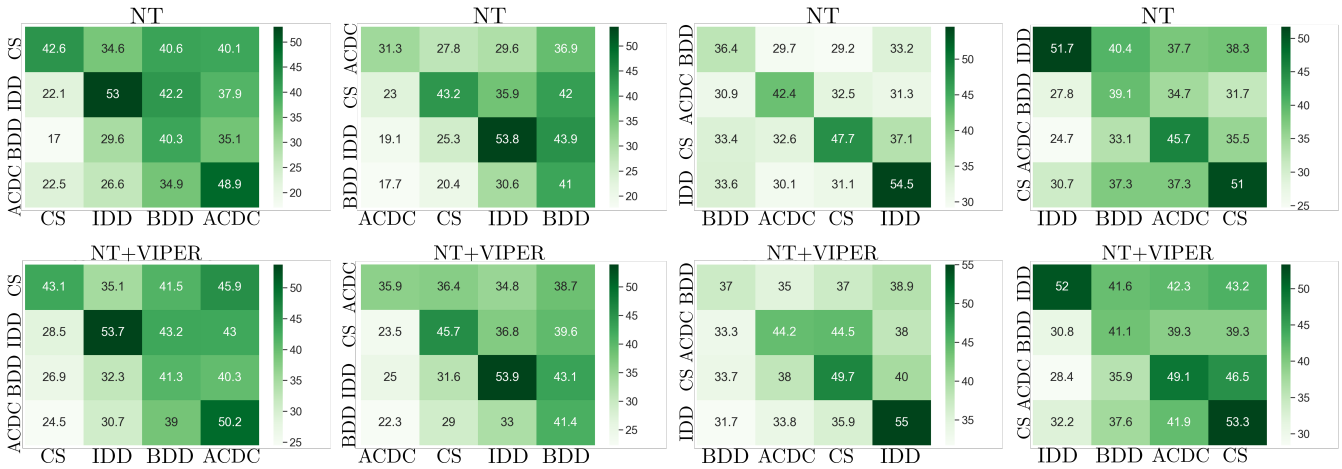

Figure 4: Forward and backward transfer under different domain orders. We analyze the forward and backward transfer during ODICS of both NT and NT+VIPER under different domain orders. The x-axis represents the observed domain within the stream while the y-axis shows the domain, on which we are evaluating the model. SimCS with VIPER improves both the forward (lower triangular) and backward (upper triangular) transfer in ODICS under different domain orders.

malized for all methods, particularly when comparing NT against NT+VIPER. Since NT+VIPER uses a $1:1$ sim-real ratio in the batch, comparing it with NT using the same computational budget ($N=4$) might not be fair for NT. Effectively, NT+VIPER with $N=4$ is equivalent to $N=8$ due to the additional simulated data. Our results in Figure 3 show that even when normalizing the computational budget, SimCS still outperforms the baseline in ODICS. For example, when NT+VIPER is allowed $N=4$ steps of computation, it achieves an mIOU of 43.8% on IDD while NT with $N=8$ achieves 39.1% on the same dataset.

The interested reader may now wonder if the performance gains of SimCS simply come from allowing the model to train on more data. This is not the case. First, when allowing the model to train for a larger number of iterations, SimCS consistently improves performance as shown in Figure 3, unlike the baseline. Second, only increasing the amount of data is not guaranteed to improve performance. The last row of Table 1 shows that simulated data does not help in the fully supervised setting. Combining both observations shows that indeed SimCS improves performance by reducing forgetting. Finally, the cost of generating simulated data is negligible compared to the training cost. For instance, the training time for a batch of 4 images for 4 iterations is 1.8 seconds, while generating a batch of 4 images from CARLA takes 0.18 seconds using the same hardware. This adds a realistic advantage for SimCS, where simulated data can be generated on-the-fly during online learning.

5.5 SimCS Improves Forward Transfer

Finally, we conduct a fine-grained analysis and study the performance on all domains after training on every domain. Figure 4 summarizes this analysis, where the horizontal axis represents the last observed domain within the stream, while the vertical axis represents the domain we evaluate the model on. The last column corresponds to the results in Table 1, where we only report the performance of the final model after the last domain. At last, we extend our analysis to different domain orders to include (ACDC-CS-IDD-BDD), (BDD-ACDC-CS-IDD), and (IDD-BDD-ACDC-CS). Note that in each matrix, the performance difference ($r-d$) between a diagonal element d and an element r to the right of it in the same row reflects the forgetting (smaller is better). On the other hand, the performance difference $d-l$ between a diagonal element d and an element l to the left of it in the same row reflects the forward transfer.

First of all, including simulated data in the ODICS setup not only reduces forgetting, but also improves the forward transfer. For example, including simulated data from VIPER boosts the forward transfer to ACDC in the (CS-IDD-BDD-ACDC) setup from 34.9% to 39% when trained on (CS-IDD-BDD). We note that this result is not specific to the order at which the considered domains are presented. For instance, our approach improves the forward transfer from 20.4% to 29% on BDD when trained on (ACDC-CS) in the (ACDC-CS-IDD-BDD) setup. Meanwhile, differently ordered streams result in larger variations in both forgetting and forward transfer. For example, the performance on CS drops from 40.1% to 36.9% when changing the setup from (CS-IDD-BDD-ACDC) to (ACDC-CS-IDD-BDD). Furthermore, the performance on all domains (except IDD) drops significantly when IDD is the last domain. Specifically, in the (BDD-ACDC-CS-IDD) setup, the forgetting on ACDC is a significant 10.6% mIOU. This can be attributed to the distribution shift that IDD has compared to other domains.

6 Conclusions

In this work, we investigated domain-incremental online continual learning for semantic segmentation. We identified the limitations of existing continual learning strategies and introduced SimCS, an orthogonal approach that utilizes simulated data generated on-the-fly to mitigate forgetting.

Acknowledgements

This work was done during a research internship of the first author at Intel Labs. This work was partially supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-CRG2021-4648. We would like to thank Alejandro Pardo, Botos Csaba, and Shariq Bhat for the help and discussion. Adel Bibi acknowledges the Amazon Research Awards funding.

References

Alfarra, M.; Itani, H.; Pardo, A.; Alhuwaider, S.; Ramazanova, M.; Pérez, J. C.; Cai, Z.; Müller, M.; and Ghanem, B. 2023. Revisiting Test Time Adaptation under Online Evaluation. arXiv:2304.04795.

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 139–154.

Aljundi, R.; Lin, M.; Goujaud, B.; and Bengio, Y. 2019. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32.

Bang, J.; Koh, H.; Park, S.; Song, H.; Ha, J.-W.; and Choi, J. 2022. Online Continual Learning on a Contaminated Data Stream with Blurry Task Boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9275–9284.

Blaga, B.-C.-Z.; and Nedevschi, S. 2019. Semantic Segmentation Learning for Autonomous UAVs using Simulators and Real Data. In *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 303–310.

Cai, Z.; Sener, O.; and Koltun, V. 2021. Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8281–8290.

Cermelli, F.; Mancini, M.; Bulo, S. R.; Ricci, E.; and Caputo, B. 2020. Modeling the background for incremental learning in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9233–9242.

Chaudhry, A.; Ranzato, M.; Rohrbach, M.; and Elhoseiny, M. 2018. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*.

Chaudhry, A.; Rohrbach, M.; Elhoseiny, M.; Ajanthan, T.; Dokania, P. K.; Torr, P. H.; and Ranzato, M. 2019. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

Chen, L.-C.; Papandreou, G.; Schroff, F.; and Adam, H. 2017. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

Commission, E. 2021. Data Protection: Rules for the protection of personal data inside and outside the EU. https://ec.europa.eu/info/law/law-topic/data-protection_en. Accessed: 2023-12-21.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3223.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Li, F.-F. 2009. ImageNet: A large-scale hierarchical image database. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*, 1–16. PMLR.

Douillard, A.; Chen, Y.; Dapogny, A.; and Cord, M. 2021. Plop: Learning without forgetting for continual semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4040–4050.

Garg, P.; Saluja, R.; Balasubramanian, V. N.; Arora, C.; Subramanian, A.; and Jawahar, C. 2022. Multi-Domain Incremental Learning for Semantic Segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 761–771.

Ghunaim, Y.; Bibi, A.; Alhamoud, K.; Alfarra, M.; Al Kader Hammoud, H. A.; Prabhu, A.; Torr, P. H.; and Ghanem, B. 2023. Real-time evaluation in online continual learning: A new hope. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11888–11897.

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*, 770–778.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526.

Li, S.; Chaplot, D. S.; Tsai, Y.-H. H.; Wu, Y.; Morency, L.-P.; and Salakhutdinov, R. 2020. Unsupervised domain adaptation for visual navigation. *arXiv preprint arXiv:2010.14543*.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12): 2935–2947.

Lin, Z.; Shi, J.; Pathak, D.; and Ramanan, D. 2021. The CLEAR Benchmark: Continual LEArning on Real-World Imagery. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Maracani, A.; Michieli, U.; Toldo, M.; and Zanuttigh, P. 2021. Recall: Replay-based continual learning in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7026–7035.

Michieli, U.; and Zanuttigh, P. 2019. Incremental learning techniques for semantic segmentation. In *Proceedings of*

the IEEE/CVF International Conference on Computer Vision Workshops, 0–0.

Prabhu, A.; Torr, P. H.; and Dokania, P. K. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, 524–540. Springer.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 8748–8763. PMLR.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2001–2010.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Richter, S. R.; Hayder, Z.; and Koltun, V. 2017. Playing for Benchmarks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2232–2241.

Richter, S. R.; Vineet, V.; Roth, S.; and Koltun, V. 2016. Playing for data: Ground truth from computer games. In *European conference on computer vision*, 102–118. Springer.

Sakaridis, C.; Dai, D.; and Van Gool, L. 2021. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10765–10775.

Varma, G.; Subramanian, A.; Namboodiri, A.; Chandraker, M.; and Jawahar, C. 2019. IDD: A dataset for exploring problems of autonomous navigation in unconstrained environments. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1743–1751. IEEE.

Wu, Y.; Chen, Y.; Wang, L.; Ye, Y.; Liu, Z.; Guo, Y.; and Fu, Y. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 374–382.

Yu, F.; Xian, W.; Chen, Y.; Liu, F.; Liao, M.; Madhavan, V.; and Darrell, T. 2018. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2(5): 6.