University of Liège
School of Engineering and Computer Science

Online Distillation with Continual Learning for Cyclic Domain Shifts

A dissertation submitted in partial fulfillment of the requirements
for the degree of
Master of Science in Data Science

Author
Joachim Houyon

Advisors
Pr. Marc Van Droogenbroeck

Academic year 2022-2023

Abstract

The technique of online distillation has become increasingly popular in adapting real-time deep neural networks using a slow and accurate teacher model. However, one of the most significant challenges encountered with online distillation is catastrophic forgetting, which happens when the student model is updated with new domain data and loses the previously learned knowledge.

The main contribution in this thesis is to apply continual learning techniques to mitigate the problem fo catastrophic forgetting in online distillation. Indeed, continual learning has shown to be useful in a more general setting, where the model tend to forget knowledge from previous tasks when learning the current one. The study aims to assess the efficacy of various state-of-the-art continual learning methods in reducing catastrophic forgetting when applied to online distillation, particularly in cyclic domain shifts.

The experimental results presented in this study show improved accuracy and robustness in the context of online distillation when leveraging continual learning methods to reduce catastrophic forgetting, with potential applications in fields such as video surveillance or autonomous driving. As such, this work represents a significant contribution to the fields of online distillation and continual learning, providing new insights and avenues for future research.

The content of this thesis is mostly based on the work of Houyon *et al.* [1], of which I am a first author. This paper was peer-reviewed and accepted for publication in the CLVision workshop at CVPR 2023. In addition, this thesis provides more comprehensive and detailed explanations of the continual learning methods used in the study, as well as additional experiments, results, and discussions. This report takes into account the feedbacks and remarks from the CLVision reviewers, making it, overall, more complete than the published paper. Finally, we provide a detailed implementation on a public Github repository: `https://github.com/Houyon/online-distillation-cl`.

Acknowledgements

I would like to thank my family and friends who supported me during my thesis, special thank to my partner who kept on encouraging me when I lacked motivation. Also a special thank to Julien Gustin, my duo for all the projects during my studies.

I would also like to thank my supervisor Marc Van Droogenbroeck for his help and guidance, and also to ge me the opportunity to work on the field of online distillation. The topic was interesting and I have learned a lot from it.

I want to express my gratitude to doctoral students and professors from the University of Liège (Anthony Cioppa, Anaïs Halin, Maxim Henry and Marc Van Droogenbroeck), as well as professors and doctoral students from KAUST University in Saudi Arabia (Yasir Ghunaim, Motasem Alfarra, and Bernard Ghanem) that I have collaborated to publish a scientific article. It was a very enriching experience to work on the domain of continual learning with such talented people.

Finally, I would like to personally thank Anthony Cioppa for his help, guidance and encouragement throughout the year. This thesis would certainly not have come to its successful conclusion without his help, support and trust.

Contents

Chapter 1

Introduction

1.1 Problem statement

Deep Neural Networks (DNNs) have shown outstanding performance on a range of computer vision tasks by assuming that the training and testing data share similar distributions, as mentioned in [2–4]. Nevertheless, DNNs can experience significant loss in performance when tested on out-of-distribution data, where the testing data contains domain shifts that differ from the training data, as stated in [5, 6]. Moreover, when learning a continuous stream of tasks, DNNs tend to forget previously learned distributions, which can lead to a considerable performance loss, as discussed in [7]. The occurrence of domain shifts in real-world applications due to variations in brightness, weather conditions, and sensor perturbations, as cited in [8], highlights the importance of developing algorithms that can enable DNNs to adapt to such shifts while maintaining high performance in real-world scenarios.

Continual learning enables machine learning models to learn from a continuous flow of data without forgetting previously learned knowledge [9, 10]. This research focuses on a specific practical scenario of online continual learning [11], which deals with cyclic domain shifts, where a sequence of data switches between two distributions for a certain period. For instance, an autonomous driving system that travels between cities and countrysides might face domain shift as the instance's distribution changes between the two scenes, potentially leading to online learning failures and real-world deployment issues. Although prior research has explored online continual learning in different settings, such as unsupervised domain adaptation [12], domain incremental learning [13], and test-time adaptation [14], these studies generally consider more general settings where domain shifts are unconditional and possibly less realistic. By focusing on cyclic domain shifts, this research has enabled the development of new algorithms that can adapt to these changes more effectively in real-world scenarios, exploring a pragmatic approach.

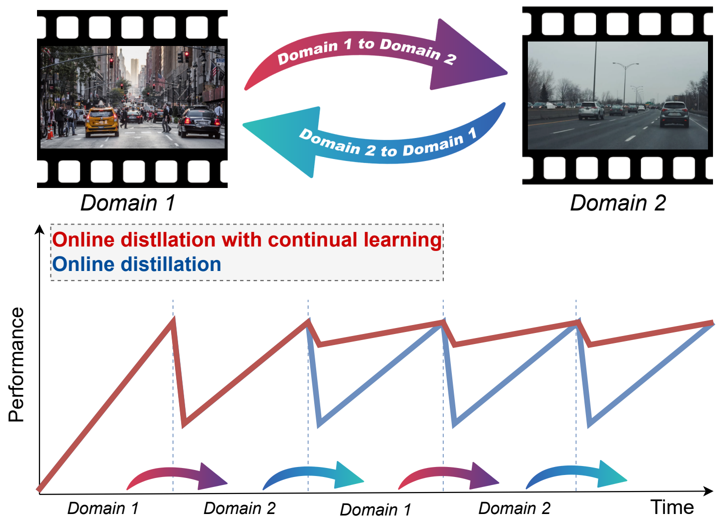


Figure 1.1. (© 2023 IEEE) **Online distillation with continual learning.** According to Houyon *et al.* [1] , when cyclic domain shifts occur in long videos, the online distillation framework proposed by Cioppa *et al.* [15] forgets the previously acquired knowledge as it fine-tunes on the current domain. It is studied, in this work, the inclusion of state-of-the-art continual learning methods inside the online distillation framework to mitigate this catastrophic forgetting around the domain shifts.

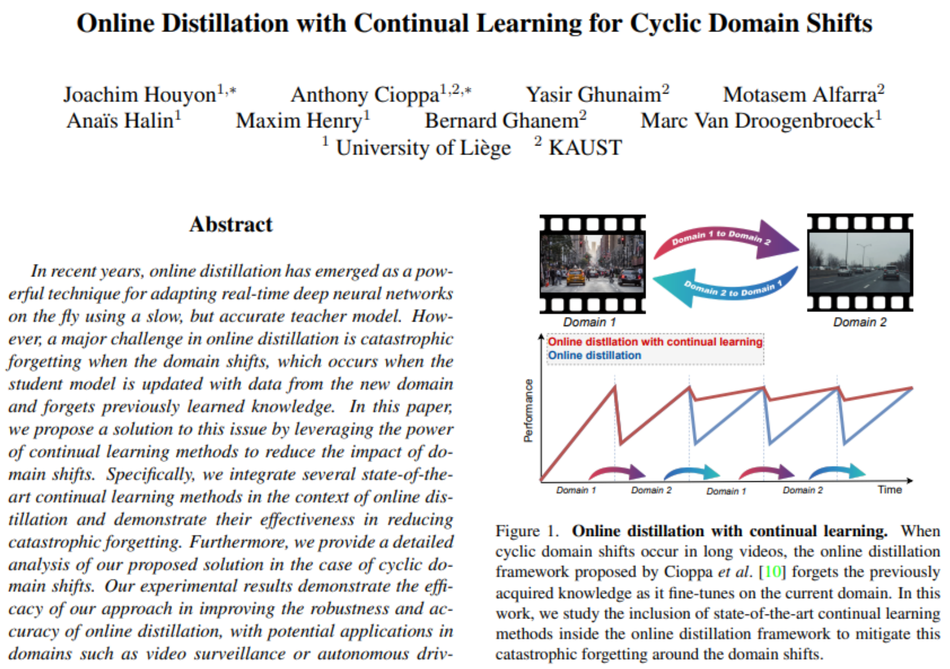
This work proposes a novel approach to address the challenge of adapting to cyclic domain shifts in the context of online domain incremental learning. Specifically, a previously published real-time online distillation technique [15] is employed to learn from the unlabeled cyclic stream of data. The online distillation technique asynchronously updates a student-teacher based approach on the received data, enabling the model to continually learn from new data. However, it was found that the way the student was being trained can cause the student to forget the previously learned domain, leading to a significant loss in performance. To mitigate this undesirable effect, online distillation is combined with state-of-the-art continual learning techniques as shown in Figure 1.1. Both regularization- and replay-based approaches from the continual learning literature are leveraged. The proposed approach effectively enables the student to adapt to cyclic domain shifts and maintain high performance over time, making it suitable for real-world deployment.

1.2 Contributions

Contributions are summarized in three points. The first point is the definition of the cyclic online continual learning setup and a proposal for the corresponding evaluation metrics. Second, new metrics are proposed in order to evaluate the model in the specific scenario of cyclic domain shifts. Third, we combine online distillation with both regularization and replay-based continual learning approaches to better learn from cyclic domains. Then, experiments are conducted on the proposed stream where it is shown that these approaches mitigates forgetting on the original online distillaiton framework.

1.3 Collaboration

The work of this thesis has led to the writing of a scientific article (Houyon *et al.* [1]) that has been peer-review and accepted at CLVision workshop which is part of the CVPR (Computer Vision and Pattern Recognition) 2023 conference. This article is the result of collaboration between doctoral students and professors from the University of Liège (Anthony Cioppa, Anaïs Halin, Maxim Henry, and Marc Van Droogenbroeck), as well as professors and doctoral students from KAUST University in Saudi Arabia (Yasir Ghunaim, Motasem Alfarra, and Bernard Ghanem). To achieve this, bi-weekly meetings were held so that these people could have continuous monitoring of my progress and guide me towards the right path. I personnaly took care of the implementation and my collaborators guided me based on their respective expertise and helped me write the paper.



Online Distillation with Continual Learning for Cyclic Domain Shifts

Joachim Houyon[1,*] Anthony Cioppa[1,2,*] Yasir Ghunaim[2] Motasem Alfarra[2]
Anaïs Halin[1] Maxim Henry[1] Bernard Ghanem[2] Marc Van Droogenbroeck[1]
[1] University of Liège [2] KAUST

Abstract

In recent years, online distillation has emerged as a powerful technique for adapting real-time deep neural networks on the fly using a slow, but accurate teacher model. However, a major challenge in online distillation is catastrophic forgetting when the domain shifts, which occurs when the student model is updated with data from the new domain and forgets previously learned knowledge. In this paper, we propose a solution to this issue by leveraging the power of continual learning methods to reduce the impact of domain shifts. Specifically, we integrate several state-of-the-art continual learning methods in the context of online distillation and demonstrate their effectiveness in reducing catastrophic forgetting. Furthermore, we provide a detailed analysis of our proposed solution in the case of cyclic domain shifts. Our experimental results demonstrate the efficacy of our approach in improving the robustness and accuracy of online distillation, with potential applications in domains such as video surveillance or autonomous driv-

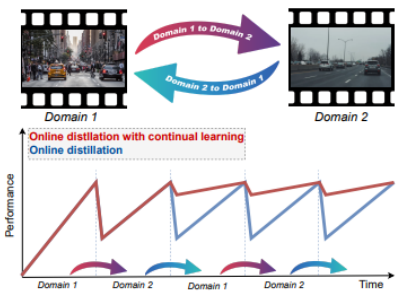


Figure 1. **Online distillation with continual learning.** When cyclic domain shifts occur in long videos, the online distillation framework proposed by Cioppa *et al.* [10] forgets the previously acquired knowledge as it fine-tunes on the current domain. In this work, we study the inclusion of state-of-the-art continual learning methods inside the online distillation framework to mitigate this catastrophic forgetting around the domain shifts.

Figure 1.2. (© 2023 IEEE)Screenshot of our published paper accepted at the CLVision workshop at the CVPR 2023 conference [1]

Chapter 2

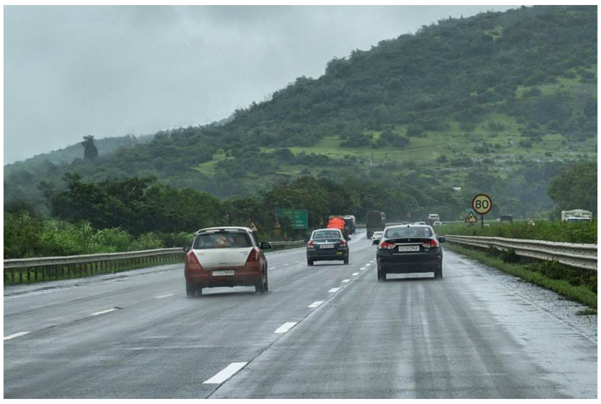
Related Work

2.1 Domain shifts

A domain shift is characterized by a change in the statistical distribution of data between distinct domains, as explained by Farahani [16]. Recent studies in computer vision have shown that this phenomenon is often observed at test time in open-world scenarios [17–19]. In the field of autonomous driving, domain shift can occur due to various factors [20], such as diverse environmental conditions like rural or urban roads, lighting conditions such as day or night, weather conditions like sunny or snowy [8], traffic conditions, or differences in the appearance of roads or traffic signs across different countries [21].

In the context of autonomous driving, it is essential for algorithms to be able to handle dynamic domain shifts to ensure the vehicle can perceive and understand its surroundings, and avoid obstacles. To address this challenge, domain adaptation has become a critical area of research, particularly in open-world scenarios such as autonomous vehicles [20, 22–25], where data is collected in a constantly changing environment.

(a) Downtown environment


(b) Highway environment

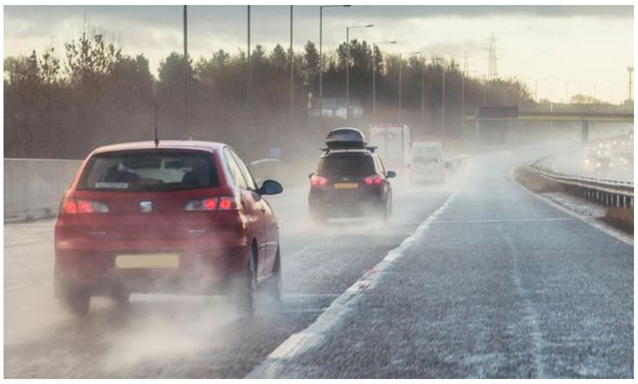

(c) Rainy environment

Figure 2.1. Illustration of different environments. In autonomous driving, domains can differ from where the car drives (highway (Figure b), downtown (Figure a), urban, forest, ...) but also according to weather conditions (sunny, rainy (Figure c), ...). The resulting model should be able to handle all these domains.

The present study focuses on investigating the cyclic domain shift phenomenon specifically in the context of autonomous driving, where the domains are characterized by alternating sequences of *highway* and *downtown* driving conditions.

2.2 Online distillation

Achieving high performance, real-time speed, and generalizability across multiple domains is a challenge in the field of deep neural networks. On one hand, top-performing models tend to exhibit strong performance across diverse domains, they can be memory-intensive for embedded systems or too slow for use in real-time applications [26–28]. On the other hand, lightweight and fast networks show good performance on smaller domains but may not have the same level of generalizability [29].

To address this issue, Cioppa *et al.* [15] proposed an online distillation method for videos, which enables the training of a lightweight student network using a slower, larger teacher model. During testing, the teacher provides pseudo ground truths to the student, allowing it to specialize in the specific domain being analyzed. This approach helps the student model to adapt to changing video conditions and match the performance of the slower teacher. The online distillation technique is applicable to various tasks, including semantic segmentation and multi-modal object detection [15, 30].

The objective of the proposed approach is to produce real-time predictions $\hat{y}_i$ for each frame x_i of a long untrimmed video $\mathcal{V}$ for a given task $\mathcal{T}$, such as object detection or semantic segmentation. The approach uses a student-teacher architecture with a fast and slow route. The student network $\mathbf{S}$ computes $\hat{y}_i$ at a rate of $r_{\mathcal{V}}$ in the fast route (inference). In parallel, the slower but high-performance frozen teacher network $\mathbf{T}$ produces pseudo ground-truths $\tilde{y}_{i'} = \mathbf{T}(x_i)$ at a slower rate of $r_{\mathbf{T}}$ on a subset of $\mathcal{V}$ in the slow route (training).

Each new pair $(x_{i'}, \tilde{y}_{i'})$ is stored into an online dataset $\mathcal{D}$ that has a fixed size N. The stored pairs are used to train a copy $\mathbf{S}_c$ of the student network, and when the online dataset is full, the oldest pairs are replaced with new incoming pairs.

Iteratively, $\mathbf{S}_c$ is trained on $\mathcal{D}$, by minimizing the loss:

$$\mathcal{L} = \sum_{n=1}^{N} L(\mathbf{S}_c(x_n), \tilde{y}_n) \ ,$$

where the distance function L depends on the specific task being performed by the student-teacher architecture. The parameters of $\mathbf{S}$ are updated by transferring the parameters θ of $\mathbf{S}_c$ at a rate equal to the inverse of the training time of $\mathbf{S}_c$ on one epoch of $\mathcal{D}$. The complete process is outlined in Algorithm 1.

Algorithm 1 The proposed online distillation algorithm from [15]

```
1 Choose $\mathbf{T}$, initialize $\mathbf{S}$ and $\mathbf{S}_c$ with $\theta_0$, collect $\mathcal{D}_1$
2 while incoming video stream $\mathcal{V}$ do
3     while $\mathbf{S}_c$ trains with $\mathcal{D}_k$ do
4         Segment all incoming frames with $\mathbf{S}$
5         Compute $\mathbf{T}(X_{i'})$ for some incoming frames $X_{i'}$
6     $\mathbf{S}$ becomes $\mathbf{S}_c$ by copying weights $\theta_k$ of $\mathbf{S}_c$ into $\mathbf{S}$
7     $\mathcal{D}_k$ becomes $\mathcal{D}_{k+1}$ by replacing some frames by $(X_{i'}, \mathbf{T}(X_{i'}))$
8     Increment $k$ by 1
```

The introduced framework allows $\mathbf{S}$ to specialize in the latest portion of the video being analyzed, enabling it to adapt to slow domain changes as long as reliable predictions are produced by $\mathbf{T}$.

Nonetheless, the continuous fine-tuning process, due to the way the online dataset is updated as shown in Figure 2.2, causes the network to forget previously learned knowledge over time. For example, when abrupt domain shifts occur, $\mathbf{S}$ requires multiple updates to regain good performance even if the same domain has previously appeared in the video.

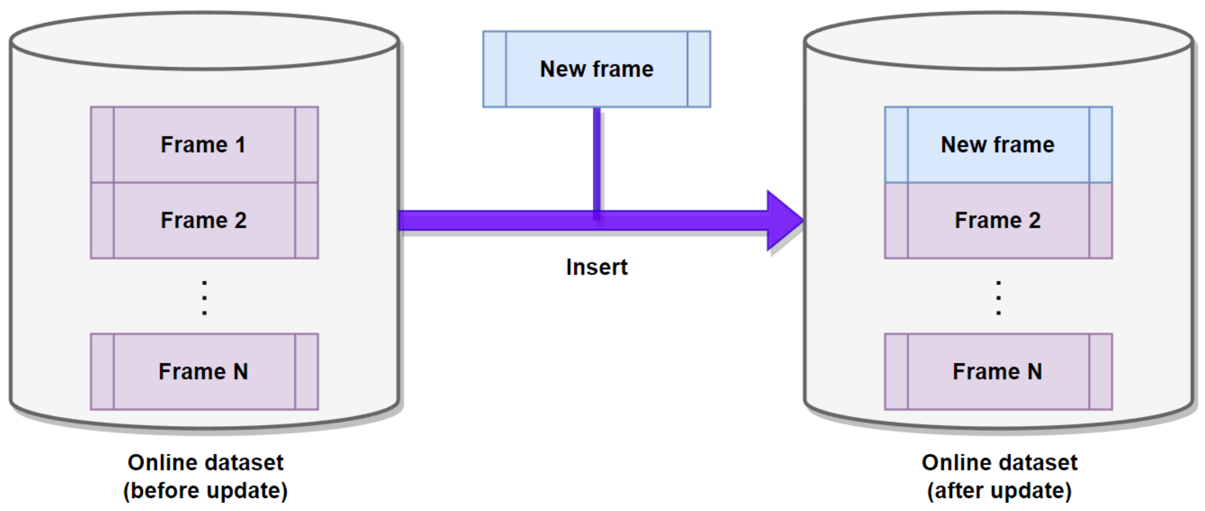


Figure 2.2. **Baseline.** Illustration on how the online dataset works in the original distillation framework. Upon the arrival of a new frame, it gets incorporated into the online dataset. However, if the online dataset has reached its capacity limit, the oldest frame in it is discarded; it works in a **First In First Out (FIFO)** manner. This results in the online dataset being populated with frames that are only from the last h minutes of video. Consequently, the student's parameters get adjusted to align with this particular distribution of frames, without considering any previous contexts. This entire procedure can lead in catastrophic forgetting.

Therefore, this work investigates several continual learning techniques to alleviate the impact of catastrophic forgetting in online distillation, particularly in cases of cyclic domain shifts. This work proposes to combine online distillation with regularization- and replay-based methods for a more effective continual learning approach.

2.3 Continual learning

Continual Learning (CL) is a learning paradigm that focuses on learning from a stream of data that may have a changing distribution over time [31, 32]. However, one of the main challenges of this paradigm is the problem of catastrophic forgetting, where previously learned knowledge is lost when adapting to newly arriving data samples [7, 10].

One approach to mitigating the forgetting effect in continual learning is to regularize the training process by constraining the changes of important network parameters [7, 33, 34], or performing knowledge distillation [9, 35, 36]. Another approach is to use replay-based methods, where previously seen examples are rehearsed by storing a subset of the observed data in a replay buffer [10, 37, 38]. While both approaches were originally proposed for the class-incremental setup and classification tasks, they have recently been extended to the more realistic domain incremental setup and the more challenging semantic segmentation task [13, 39]. However, most prior research assumes fully supervised setups where the stream reveals labeled data for the student learner. In this work, the domain incremental setup for semantic segmentation is analyzed under a semi-supervised setup; the teacher network **T** is trained using supervised learning, which then will provide pseudo-groundtruths to unlabeled data from the incoming stream that are used to train the student network.

Chapter 3

Methodology

In this section, our extension of online distillation including continual learning is presented. Then, the integration of regularization-based and replay-based continual learning methods into the new online distillation framework is detailed. Finally, an explanation is provided on how to evaluate and benchmark online continual learning methods under the cyclic stream.

3.1 Online distillation with continual learning

The existing online distillation framework is extended with continual learning techniques to mitigate catastrophic forgetting of previously learned knowledge when dealing with cyclic domain shifts. Specifically, the benchmark includes two types of techniques: replay-based methods (CL_{Rep}) that operate on the online dataset $\mathcal{D}$ using selection function f_S and update function f_U, and regularization-based methods (CL_{Reg}) that act on the loss function $\mathcal{L}$ The extended framework is illustrated in Figure 3.1.

Reguarding replay-based methods, designing the adequate selection function f_S and update function f_U is important because it will determine which samples are used for replay and which samples will remain in the buffer. They must cover several and crucial aspects such as diversity (keep samples from as many domains as possible), or similarity to incoming batch (we would rather replay samples that are not similar to incoming data as they are likely to come from a past domain).

Reguarding regularization-based methods, they represent an additional term in the loss (a regularizer) such that it constraint the parameters in order to retain past knowledge.

In the original framework, f_S selects all pairs from $\mathcal{D}$, and f_U stores the new samples in a First In First Out (FIFO) manner.

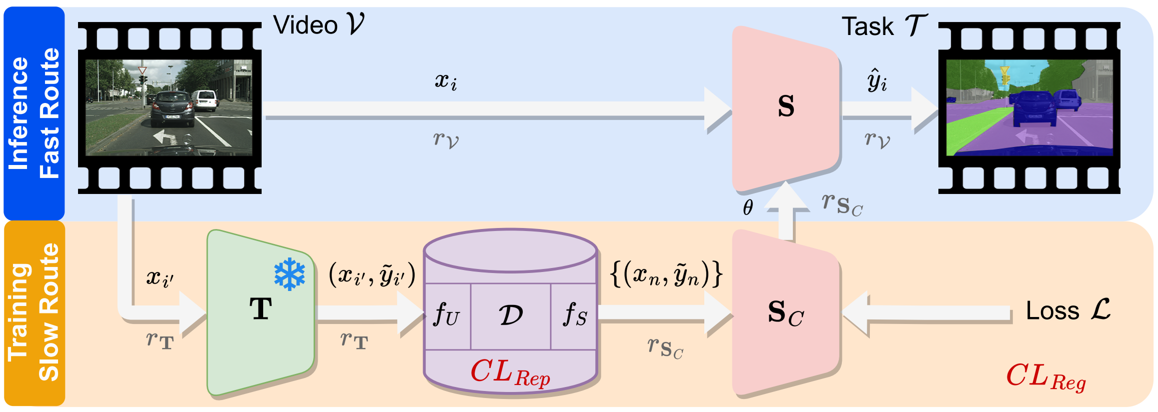


Figure 3.1. (© 2023 IEEE) **Online distillation**. According to Houyon *et al.* [1], the framework is composed of a fast and a slow route. In the fast route (inference), the video stream $\mathcal{V}$ is processed by a student network $\mathbf{S}$ on a task $\mathcal{T}$ (e.g., semantic segmentation for autonomous driving) and produces predictions $\hat{y}_i$ for each frame of the video x_i at the original video rate $r_{\mathcal{V}}$ (*i.e.*, in real time). In parallel in the slow route (training), a frozen teacher $\mathbf{T}$ produces pseudo ground-truths $\tilde{y}_{i'}$ from a subset of frames $x_{i'}$ at a slower rate $r_{\mathbf{T}}$. The pair $(x_{i'}, \tilde{y}_{i'})$ are then stored in an online dataset (or replay buffer) $\mathcal{D}$ through an update function f_U. $\mathcal{D}$ is sampled through a selection function f_S and the selected pairs $(x_n, \tilde{y}_n)$ are used to train a copy of the student network $\mathbf{S}_c$ for one epoch using a loss $\mathcal{L}$. The parameters θ of $\mathbf{S}_c$ are then transferred to $\mathbf{S}$ at a rate $r_{\mathbf{S}_c}$ (corresponding to the inverse of the training time of $\mathbf{S}_c$ on one epoch) so that $\mathbf{S}$ improves on the latest domain of $\mathcal{V}$. One of the contribution of the work consists in including replay-based Continual Learning (CL) methods, CL_{Rep}, inside $\mathcal{D}$ and regularization-based methods, CL_{Reg}, on $\mathcal{L}$.

3.2 Replay-based methods

The methods that leverage a replay buffer involve using a finite-sized collection of data and corresponding ground-truth labels that are accessed by a selection function, denoted as f_S, and updated with new data by an update function, denoted as f_U, at every training epoch. The online distillation framework, which was presented earlier, can be considered as a replay-based method. In this case, the replay buffer corresponds to $\mathcal{D}$, the pseudo ground-truth predictions $\tilde{y}_n$ serve as the labels, f_S chooses all the data in the replay buffer to be used during the training epoch, and f_U determines how the samples in the replay buffer are updated.

In the original online distillation framework, the size of the replay buffer is equal to the number of samples, denoted as N, that are passed to the model during each training step. However, in the extended version of the framework, the replay buffer is expanded to include $M \geq N$ samples, where N samples are sampled without replacement from the buffer at each training step. The selected samples are then augmented with the new incoming data from the stream.

By enlarging the replay buffer to hold M samples and then selecting only N samples from it, the student can benefit from a more diverse set of training samples, without incurring additional computational overhead for a training epoch.

Several strategies to modify f_U and f_S to reduce the catastrophic forgetting are considered: FIFO, Uniform, Prioritized, and MIR.

3.2.1 First In First Out

In the **First In First Out (FIFO)** strategy, f_U inserts the new frames while removing the oldest ones. This strategy is the one implemented in the original framework's update strategy where $M = N$, and f_S selects all M samples from the replay buffer. This strategy is used as a baseline with other methods. Algorithm 2 illustrates f_U in the **FIFO** strategy.

Algorithm 2 f_U in the **FIFO** strategy

```
1 Replay buffer $\mathcal{D}$, set of samples $X$, buffer capacity $M$
2 function FIFO_$f_U$($\mathcal{D}$, $X$, $M$)
3     $s \leftarrow M - |\mathcal{D}| + |X|$                    Available space in $\mathcal{D}$
4     if $s < 0$ then
5         $\mathcal{D} \leftarrow \mathcal{D} \setminus \mathcal{D}[1, ..., s]$     Remove the $s$ oldest samples in $\mathcal{D}$
6     $\mathcal{D} \leftarrow \mathcal{D} \cup X$
7     return $\mathcal{D}$
```

3.2.2 Uniform

The objective of the replay-based methods is to keep in memory a wider variety of samples to reduce catastrophic forgetting. The **Uniform** strategy works such as f_U (Algorithm 3) randomly selects samples to replace, and f_S (Algorithm 4) randomly selects N frames for replay.

Thanks to this, the probability to replace or select a sample decays exponentially. The **Uniform** strategy is a first improvement in order to reduce the forgetting of the student network.

Algorithm 3 f_U in the **Uniform** strategy

```
1 Replay buffer $\mathcal{D}$, set of samples $X$, buffer capacity $M$
2 function UNIFORM_$f_U$($\mathcal{D}$, $X$, $M$)
3     for $x \in X$ do
4         if $|\mathcal{D}| == M$ then
5             $j \leftarrow \text{RandInt}([0, M-1])$          Select an index at random
6             $\mathcal{D} \leftarrow \mathcal{D} \setminus \{\mathcal{D}[j]\}$        Remove the $j-th$ sample in $\mathcal{D}$
7         $\mathcal{D} \leftarrow \mathcal{D} \cup \{x\}$
8     return $\mathcal{D}$
```

In terms of computation overhead, the **Uniform** strategy is (very) slightly bigger than **FIFO** strategy. Indeed, in the **Uniform** strategy, the selection function is $\mathcal{O}(N)$ in-

Algorithm 4 f_S in the **Uniform** strategy

```
1 Replay buffer $\mathcal{D}$, number of samples to select $N$
2 function UNIFORM_$f_S$($\mathcal{D}$, $N$)
3     $s \leftarrow \min(|\mathcal{D}|, N)$
4     $J \leftarrow$ RandomChoice($[0, |\mathcal{D}| - 1]$, $s$)  Select $s$ indexes in the range $[0, |\mathcal{D}|]$ without replacement
5     $\mathcal{Z} \leftarrow \mathcal{D}[J]$
6     return $\mathcal{Z}$
```

stead of $\mathcal{O}(1)$ for the **FIFO** strategy. The complexity of the update function f_U remains unchanged.

3.2.3 Prioritized

The **Prioritized** strategy is an improvement of the **Uniform** strategy and is inspired of a technique that was proposed by Schaul *et al.* [40] in the context of reinforcement learning. We adapt the work of these authors in the context of continual learning. The authors of this paper introduce a new approach to experience replay in deep reinforcement learning. The experience replay, in the context of reinforcement learning, is a method that enables an agent to learn from its past experiences by storing and randomly sampling batches of transitions (state, action, reward, next state) from a replay memory buffer.

The proposed approach, called Prioritized Experience Replay (PER), assigns priorities to each transition in the memory buffer based on its estimated potential for improving the learning process. The priorities are calculated using the temporal-difference (TD) error, which represents the difference between the predicted and actual values of the expected reward. Transitions with higher TD errors are considered more important, and therefore, are sampled more frequently than low-priority transitions.

The paper presents two methods for prioritizing experience replay: Proportional Prioritization, where the priority of each transition is proportional to its TD error, and Rank-Based Prioritization, where transitions are ranked by their TD errors, and the sampling probability is based on their rank.

This strategy for reinforcement learning can be adapted to a strategy; in the **Uniform** strategy, frames are assumed to be equally important. In the **Prioritized** strategy, an importance score is assigned to each frame. More formally, let $\mathcal{D} = \{x_1, ..., x_M\}$ be the M frames in the replay buffer $\mathcal{D}$. Importance scores $\{\mathcal{I}_1, ..., \mathcal{I}_M\}$ are computed for each frame in the replay buffer. $\mathcal{I}_n$ denotes the importance of the frame x_n, and is measured by the loss function:

$$\mathcal{I}_n = L(\mathbf{S}(x_n), \mathbf{T}(x_n)) \ .$$

It is worth noticing that the importance score could be measured according to other metrics, such as the accuracy, the recall, the precision, the F1-score or the Mean Intersection

Over Union (mIoU). The notion of importance can vary on the context and the task that needs to be achieved.

The importance score constitutes the probability of determining actions on the frames. For the selection function f_S (Algorithm 6), it determines the probability to select the sample for replay. For the update function f_U (Algorithm 5), it determines the probability to remove this sample from the replay buffer $\mathcal{D}$. The transformation of the importance score into a probability is straightfoward:

$$p_n = \frac{\mathcal{I}_n^{-1}}{\sum_{n'=1}^{M} \mathcal{I}_{n'}^{-1}} \ .$$

To control the prioritization, an hyper-parameter $\alpha \geq 0$ is added to adjust the conservation of frames. The probability of defining actions on the frame is as follows:

$$P(n) = \frac{p_n^{\alpha}}{\sum_{n'=1}^{M} p_{n'}^{\alpha}} \ .$$

Therefore, a high value of α will be more conservative, while a low value of α will be less conservative, with $\alpha = 0$ being the **Uniform** strategy.

However, due to time constraints, it is not acceptable to re-compute, after each epoch, the importance score for all M frames in the replay buffer as it would represent M forward passes in the student network $\mathbf{S}$. Therefore, it is decided to only re-compute the importance score for the frames that are chosen for the next epoch and this update is done inside the update function f_U.

Overall, the **Prioritized** is slower than the **Uniform** strategy in terms of updates. Reguarding the selection function, the theoretical complexity is $\mathcal{O}(M)$ instead of $\mathcal{O}(N)$, but, practically speaking, it is very negligible.

Algorithm 5 f_U in the **Prioritized** strategy

```
1 Replay buffer $\mathcal{D}$, set of samples $X$, buffer capacity $M$
2 function PRIORITIZED_$f_U$($\mathcal{D}$, $X$, $M$)
3     for $x \in X$ do
4         if $|\mathcal{D}| == M$ then
5             $\mathcal{P} \leftarrow [P(0), ..., P(M-1)]$
6             $j \leftarrow$ RandInt($[0, M-1]$, $\mathcal{P}$)   Select an index at random, according to the
  probability distribution $\mathcal{P}$
7             $\mathcal{D} \leftarrow \mathcal{D} \setminus \{\mathcal{D}[j]\}$   Remove the $j-th$ sample in $\mathcal{D}$
8         $\mathcal{D} \leftarrow \mathcal{D} \cup \{x\}$
9     return $\mathcal{D}$
```

Algorithm 6 f_S in the **Prioritized** strategy

```
1 Replay buffer $\mathcal{D}$, number of samples to select $N$
2 function PRIORITIZED_$f_S$($\mathcal{D}$, $N$)
3     $s \leftarrow \min(|\mathcal{D}|, N)$
4     $\mathcal{P} \leftarrow [P(0), ..., P(|\mathcal{D}| - 1)]$
5     $J \leftarrow$ RandomChoice($[0, |\mathcal{D}| - 1]$, $s$, $\mathcal{P}$)        Select $s$ indexes in the range $[0, |\mathcal{D}|]$
  without replacement according to the probability distribution $\mathcal{P}$
6     $\mathcal{Z} \leftarrow \mathcal{D}[J]$
7     return $\mathcal{Z}$
```

3.2.4 Maximum Interfered Retrieval

Proposed by Rahaf *et al.* [38], Maximum Interfered Retrieval (MIR) states that replay strategies relying in the selection of random samples from a replay memory or a generative model is suboptimal. Instead, MIR suggests a controlled sampling of memories for replay. To do that, MIR selects samples that are most interfered. In other words, it retrieves samples that are the most negatively impacted by the foreseen parameters update.

This method makes the assumption that some already seen samples from memory may be unaffected or even improved, thus retraining on them represents a waste of computation.

In the online distillation framework, MIR represents a selection function f_S (Algorithm 7) since it retrieves samples from memory. Let $\mathbf{S}_c(\cdot;\theta)$ be parameterized with parameters θ. Given a standard objective $\min_{\theta} L(\mathbf{S}_c(X;\theta), \mathbf{T}(X))$, when receiving samples X from the video stream, the would-be parameters θ^v are estimated with X:

$$\theta^v = \theta - \alpha \nabla L(\mathbf{S}_c(X;\theta), \mathbf{T}(X))$$

θ^v can be used to find the samples that would be the most interfered from parameters updated with the incoming batch. Let k be the number of samples that are retrieved from the replay buffer $\mathcal{D}$. The top-k samples from the replay buffer $\mathcal{D}$ are retrieved according to the following criteria:

$$s(x) = L(\mathbf{S}_c(x;\theta^v), \mathbf{T}(x)) - L(\mathbf{S}_c(x;\theta), \mathbf{T}(x)) \ .$$

$s(x)$ denotes the interference on sample x caused by updating the parameters from θ to θ^v. The memory could be augmented in order to keep track of the parameters θ^* that has given the best $L(\mathbf{S}_c(x;\theta), \mathbf{T}(x))$ so far for a given sample x, denoted as $L(\mathbf{S}_c(x;\theta^*), \mathbf{T}(x))$. With this strategy, the top-k samples from the replay buffer $\mathcal{D}$ are retrieved according to the following criteria:

$$s^*(x) = L(\mathbf{S}_c(x;\theta^v), \mathbf{T}(x)) - min(L(\mathbf{S}_c(x;\theta), \mathbf{T}(x)), L(\mathbf{S}_c(x;\theta^*), \mathbf{T}(x))) \ .$$

$s^*(x)$ denotes the interference on sample x caused by updating the parameters from θ^* to θ^v

To further reduce the compute cost, $C > k$ samples from the replay buffer $\mathcal{D}$ are chosen randomly. The search criterion is applied on these C samples. The selection is done ramdomly in order to ensure the diversity of the samples.

Algorithm 7 f_S in the **MIR** strategy

```
1  Replay buffer $\mathcal{D}$, number of samples to select $N$, incoming batch $X$, budget $C$
2  function MIR_$f_S$($\mathcal{D}$, $N$, $X$, $C$)
3      $q \leftarrow \min(|\mathcal{D}|, C)$
4      $J_C \leftarrow \text{RandomChoice}([0, |\mathcal{D}| - 1], q)$
5      $\mathcal{Z} \leftarrow \mathcal{D}[J_C]$                                  Samples to evaluate
6      $\theta^v \leftarrow \theta - \alpha \nabla L(\mathbf{S}_c(X; \theta), \mathbf{T}(X))$
7      $S \leftarrow [L(\mathbf{S}_c(z; \theta^v), \mathbf{T}(z)) - L(\mathbf{S}_c(z; \theta), \mathbf{T}(z))$ for $z \in \mathcal{Z}]$
8      $q \leftarrow \min(|\mathcal{D}|, N)$
9      $J_N \leftarrow \text{ArgMax}(S, q)$   Select the $q$ indexes according to the $q$ biggest scores inside $S$
10     $\mathcal{Z} \leftarrow \mathcal{Z}[J_N]$                                 Selected samples
11     return $\mathcal{Z}$
```

In terms of time complexity, f_S in the **MIR** strategy is quite expensive since it requires to perform a gradient step on a copy of $\mathbf{S}_c$ in order to search for the best samples for replay.

3.3 Regularization-based methods

Regularization-based methods aim to mitigate forgetting by adding a regularization term to the training loss function $\mathcal{L}$. This can be formulated as follows:

$$\mathcal{L} = \sum_{n=1}^{N} L(\mathbf{S}_c(x_n), \tilde{y}_n) + \mathcal{R} \ ,$$

where $\mathcal{R}$ is a method-specific regularization term. In this work, four regularization-based continual learning methods are considered, namely ER-ACE [41], LwF [9], MAS [33], and RWalk [34].

Some methods were designed to work on the hypothesis that task boundaries were known, i.e. in an offline setup. In the online setup, this hypothesis does not hold anymore because task boundaries are unknown. For instance, methods like LwF, MAS and RWalk were designed for an offline setup. An explanation on how to adapt these methods and simulate task boundaries will be given in section 3.3.2.

3.3.1 ER-ACE

Experience Replay with Asymmetric Cross-Entropy (ER-ACE) is a regularization method proposed by Caccia *et al.* [41] that reduces changes in the learned presentation (past tasks' knowledge) when training on samples from a new class. Given a model $f_\theta(x)$ with parameters θ, the goal is to minimize some classification loss $\mathcal{L}$ on the incoming batch of data without negatively interfering with the previously learned classes. To address this issue, the simplest but efficient approach is to use a replay buffer and some strategies like the **Uniform** strategy, as presented in 3.2.2

However, the **Uniform** strategy treats both the incoming batch and the replayed batch in a similar manner, i.e. they are both minimizing the same loss function $\mathcal{L}$. Instead, while the replayed batch is used to minimize $\mathcal{L}$, the incoming batch minimizes some new loss function $\mathcal{L}'$ such that it limits the interference with the previously well learned classes.

ER-ACE uses the **Uniform** strategy to replay samples. In the online distillation framework, this regularization method can be used as follows:

Given an incoming batch of frames X, let C_{old} be the set of previously learned classes and let C_{cur} be the set of classes observed in X (provided by $\mathbf{T}(X)$). If C is the set of classes included in the cross-entropy loss, $\mathcal{L}_{ce}$ is the cross-entropy loss defined as:

$$\mathcal{L}_{ce}(X, C) = -\sum_{x \in X} \sum_{p=0}^{|x|-1} \sum_{c \in C} \mathbf{T}(x)_{p,c} \log(\mathbf{S}_c(x)_{p,c})$$

where p is the pixel index, and c is the class index. This definition of the cross-entropy loss enables to focus on the representation of specific classes while ignoring the others. Finally, let X^{bf} and X^{in} be the framess retrieved from the replay buffer and the incoming batch of frames respectively. The loss applied at each step would be:

$$\mathcal{L}_{ace}(X^{bf} \cup X^{in}) = \mathcal{L}_{ce}(X^{bf}, C_{old} \cup C_{curr}) + \mathcal{L}_{ce}(X^{in}, C_{curr})$$

where C_{curr} is the set of classes in the incoming batch, while C_{old} is the set of already seen classes that do not appear in C_{curr}. The first term in the loss ensures that the representation of past samples are preserved. The second term in the loss makes it so that only the labels present in the batch will serve in the gradient update, thus accelerating the representation of the classes contained in C_{curr}. The algorithm is developped in 8.

Algorithm 8 ER-ACE

```
1 Incoming batch $X^{in}$, Online dataset $\mathcal{D}$, number of samples to select $N$, set of classes
  $C$, $\mathbf{S}_c$'s parameters $\theta$
2 **function** ER-ACE($X^{in}$, $\mathcal{D}$, $N$, $\theta$)
3     $X^{bf} \leftarrow$ Uniform_$f_S(\mathcal{D}, N)$
4     $C_{curr} \leftarrow$ Unique($\mathbf{T}(X^{in})$)            set of classes appearing in the incoming batch
5     $\mathcal{L} \leftarrow \mathcal{L}_{ce}(X^{bf}, C) + \mathcal{L}_{ce}(X^{in}, C_{curr})$
6     Optimizer($\nabla\mathcal{L}$, $\theta$)
7     $\mathcal{D} \leftarrow$ Uniform_$f_S(\mathcal{D}, X^{in}, |\mathcal{D}|)$
```

ER-ACE's time complexity is negligibly higher than the time complexity of the **Uniform** strategy as the additional computation overhead is only for adapting the loss for the incoming batch.

3.3.2 LwF

When learning a new task, Li *et al.* [9] proposes a method called **Learning without Forgetting (LwF)** which trains the network on the new task while preserving its on the previously learned tasks.

In this set up, the goal is to integrate new tasks without using previous tasks' data. To do this, **LwF** keeps in memory two networks: the network being trained : f_θ, and an older version of the same network: $f_{\theta'}$. In the continual online distillation setup, the network would refer to $\mathbf{S}_c$, and the older version of that network would represent an older version of $\mathbf{S}_c$.

For a given batch of data X from the new task, the standard cross-entropy loss is applied:

$$\mathcal{L}_{new}(X) = -\sum_{x \in X} \sum_{p=0}^{|x|-1} (\mathbf{T}(X)_p)^T log(\mathbf{S}_c(X;\theta)_p).$$

In the other hand, old knowledge is preserved by using knowledge distillation loss, which encourages a network to output the same outputs of another network. For the same batch of data X, the knowledge distillation loss serves as regularization term:

$$\mathcal{L}_{old}(X) = -\sum_{x \in X} \sum_{p=0}^{|x|-1} (\mathbf{S}_c(X;\theta')_p)^T log(\mathbf{S}_c(X;\theta)_p).$$

The loss applied at each step would be:

$$\mathcal{L}_{LwF}(X) = \mathcal{L}_{new}(X) + \mathcal{L}_{old}(X)$$

Therefore, $\mathcal{L}_{new}$ trains the student network on a new task while $\mathcal{L}_{old}$ penalizes the student network if its outputs differ a lot from a previous version of itself, thus preventing forgetting.

Algorithm 9 LwF

```
1 batch of frames $X$, $\mathbf{S}_c$'s parameters $\theta$, past version of $\mathbf{S}_c$'s parameters $\theta'$
2 **function** LwF($X$, $\theta$, $\theta'$)
3     $\mathcal{L} \leftarrow \mathcal{L}_{new}(X) + \mathcal{L}_{old}(X)$
4     Optimizer($\nabla\mathcal{L}$, $\theta$)
```

When training on the first task, there is no distillation loss to evaluate as there is no previous version of the network. Therefore, the network is trained on $\mathcal{L}_{new}$.

This regularization method adds computational overhead because the regularization term ($\mathcal{L}_{old}$) requires to perform $|X|$ forward passes into an older version of the student network.

3.3.3 MAS

Memory Aware Synapses (MAS) from *Aljundi et al.* [33] preserves knowledge by penalizing large changes to important parameters. For each parameter of the network, an importance weight is assigned measuring how sensitive the network is to a specific parameter change on a particular set of data.

The sensitivity of a network f with respect to its parameters θ and a data point x_k is measured by adding a small perturbation $\delta = \{\delta_i\}$ in the parameters $\theta = \{\theta_i\}$. In the online distillation framework, the result represents a change in the network that can be approximated by

$$\mathbf{S}_c(x_n, \theta + \delta) - \mathbf{S}_c(x_n, \theta) \approx \sum_i g_i(x_k)\delta_i$$

where

$$g_i(x_k) = \frac{\partial \mathbf{S}_c(x_n, \theta)}{\partial \theta_i}$$

is the gradient of the learned student network with respect its parameter θ_i evaluated for a single data point x_n, and δ_i is the change in parameter θ_i.

If δ_i is assumed to be very small, the estimation can be based on the direct magnitude of the gradient g_i, thus estimating the sensibility of the learned student network for a small perturbation to a specific parameter change. Let X be a batch of frames, importance weights $\Omega = \{\Omega_i\}$ can be obtained by taking the sum of the gradients of each frame $x \in X$ such that the importance weight Ω_i for parameter θ_i is given by

$$\Omega_i = \frac{1}{|X|} \sum_{x \in X} ||g(x)||.$$

Therefore, parameters with low importance weights do not affect the output, meaning that they can be used to learn new tasks, while parameters with high importance weights need to be unchanged as they're the one preserving the knowledge of the current and past tasks.

However, the student network $\mathbf{S}_c$ produces multi-dimensional outputs, up to millions of elements. Therefore, the computation of a single importance weight Ω_i is measured by computing the gradient of each output of $\mathbf{S}_c$, which represents as many backward passes as the output dimensionality of $\mathbf{S}_c$. Practically speaking, this computation is not possible when given time constraints. This issue is adressed by computing the gradient of the squared l_2 norm of $\mathbf{S}_c$ instead of the l_2 norm of the gradient of $\mathbf{S}_c$:

$$g_i(x) = \frac{\partial l_2^2(\mathbf{S}_c(x, \theta))}{\partial \theta_i}.$$

Thanks to this trick, a scalar value is given instead of a vector output, meaning that only one backward pass in the student network is needed to estimate Ω.

Importance weights Ω are estimated from the batch of frames X and it will prevent catastrophic forgetting for the part of the input space that X belongs to. However, parameters that do not affect this region will be given low importance weights, which can affect the knowledge of other regions in the input space. To adress this issue, a replay buffer could be added to the method so that it ensures diversity of the frames that are used to compute Ω. Another technique could be to perform exponential averaging. Let Ω^t be the current importance weights, when updating these importance weights, perform exponential averaging:

$$\Omega^{t+1} = (1-\beta)\Omega^t + \beta\Omega$$

where $\beta \in [0, 1]$ is a hyperparameter.

When training, importance parameters Ω is used as a regularizer in the loss function. For a given set of frames X, the loss at each step would be:

$$\mathcal{L}_{MAS}(X) = \mathcal{L}(X) + \lambda \sum_i \Omega_i^t(\theta_i - \theta_i^{t-1})^2$$

where λ is a hyperparameter for the regularizer, and θ^{t-1} are the parameters of $\mathbf{S}_c$ of the latest update of the importance parameters. Importance parameters update are shown in algorithm 10 and the loss computation shown in algorithm 11.

Algorithm 10 MAS: Update of the importance weights

```
1 batch of frames $X$, $\mathbf{S}_c$'s parameters $\theta$, current importance parameters $\Omega$
2 **function** UPDATE_Ω($X$, $\theta$, $\Omega$)
3     $S \leftarrow \frac{1}{|X|}\sum_{x\in X} l_2^2(\mathbf{S}_c(x,\theta))$
4     $\Omega' \leftarrow$ BACKWARD($S$)
5     $\Omega \leftarrow (1-\beta)\Omega + \beta\Omega'$
6     **return** $\theta$, $\Omega$          Return old parameters $\theta$, and new importance weights $\Omega$
```

Algorithm 11 MAS

```
1 batch of frames $X$, $\mathbf{S}_c$'s parameters $\theta$, importance parameters $\Omega$, $\mathbf{S}_c$'s parameters $\theta'$
  of the latest update of $\Omega$
2 **function** MAS($X$, $\theta$, $\Omega$, $\theta'$)
3     $\mathcal{L} \leftarrow \mathcal{L}(X) + \lambda\sum_i \Omega_i(\theta_i - \theta_i')^2$
4     Optimizer($\nabla\mathcal{L}$, $\theta$)
```

The regularization's computation complexity is linear with respect to the number of parameters. Reguarding the importance weights update, it consists in $|X|$ additional forward passes in $\mathbf{S}_c$ and a backward pass.

3.3.4 RWalk

Riemann Walk (RWalk) by *Chaudrhy et al.* [34] adapts the work of *Aljundi et al.* [33] by defining three components to regularize the loss. The first component is a KL-divergence-based regularization over the conditional likelihood $p_\theta(y|x)$ ($\mathbf{S}_c(x)$ in the online distillation framework). The second component is a parameter importance score that mesures the sensitivity of the loss with respect to movements on the Riemaniann manifold. The third component introduces methods for obtaining a small set of representative samples from past tasks (these methods similar to replay-based methods, such as the **Uniform** strategy).

The KL-divergence-based regularization helps the network learn parameters for the current task by constraining the $p_\theta(y|x)$ to be close to some older conditional likelihood $p_{\theta'}(y|x)$. In the online distillation framework, the loss would be as follows:

$$\mathcal{L}(X) + \lambda D_{KL}(\mathbf{S}_c(X;\theta)||\mathbf{S}_c(X;\theta'))$$

where λ is a hyperparameter and $D_{KL}(\mathbf{S}_c(X;\theta)||\mathbf{S}_c(X;\theta')) = \frac{1}{2}\sum_i F_{\theta'_i}(\theta_i - \theta'_i)^2$ where $F_{\theta'}$ is the empirical Fisher information matrix and it can be approximated by the importance weights computed in **MAS**, assuming that the Fisher information matrix is diagonal (independence of the parameters).

In the same way as in **MAS**, the Fisher information matrix is updated by performing exponential averaging:

$$F_\theta^t = (1-\beta)F_\theta^t + \beta F_\theta^{t-1}$$

where $\beta \in [0,1]$ is a hyperparameter and t denotes the task number. Since the Fisher information matrix is a description of how the model behaves, but it mostly depend on the current task. Therefore, they do not take into account how the parameters affected the model's optimization **over time**. The authors augment the Fisher information matrix with a parameter importance score that is accumulated over the whole training trajectory of the loss. The score is calculated by dividing the change in the loss function $\mathcal{L}$ by the distance between the conditional likelihood distributions as the model progresses through the parameter space.

Let t be the training step. When switching from θ_t to θ_{t+1}, the change in the loss can be estimated by:

$$\mathcal{L}(\theta^{t+\Delta t}) - \mathcal{L}(\theta^t) \approx -\sum_i \Delta\mathcal{L}_t^{t+1}(\theta_i)$$

where $\mathcal{L}_t^{t+1}(\theta_i)$ is the accumulated change in the loss caused by parameter θ_i from t to $t+\Delta t$. Finally, the importance of a parameter θ_i from training step t_1 to t_2 is computed as

$$s_{t_1}^{t_2}(\theta_i) = \sum_{t=t_1}^{t_2} \frac{\Delta \mathcal{L}_t^{t+1}(\theta_i)}{\frac{1}{2} F_{\theta_i}^t \Delta \theta_i^2 + \epsilon}$$

where $\epsilon > 0$. The combination of the Fisher information matrix based importance and the optimization-path based importance scores produces the following loss function:

$$\mathcal{L}_{RWalk}(X) = \mathcal{L}(X) + \lambda \sum_i (F_{\theta'_i} + s_{t_0}^{t'}(\theta_i))(\theta_i - \theta'_i)^2$$

where $s_{t_0}^{t'}(\theta_i)$ denotes the accumulated score from the first training iteration t_0 until the last training iteration t' corresponding to the latest task. To avoid score accumulation to be too big, scores are averaged after each task. Algorithm is illustrated in 12

Algorithm 12 RWalk

```
1 batch of frames $X$, $\mathbf{S}_c$'s parameters $\theta$, $\mathbf{S}_c$'s parameters $\theta'$ of the latest update of the
  Fisher information matrix, Fisher information matrix $F_{\theta'}$
2 function RWalk($X$, $\theta$, $\theta'$, $F_{\theta'}$)
3     $\mathcal{L} \leftarrow \mathcal{L}(X) + \lambda \sum_i (F_{\theta'_i} + s_{t_0}^{t'}(\theta_i))(\theta_i - \theta'_i)^2$
4     Optimizer($\nabla\mathcal{L}$, $\theta$)
```

3.3.5 Adaptation to the online case

LwF, MAS and **RWalk** were designed to work in an offline manner, meaning that task boundaries were known. In this setup, this hypothesis does not hold; task boundaries are unknown. In this thesis, we propose a solution to make these regularization-based methods work for online streams, with unknown task boundaries.

To make these algorithms work for online streams without task boundaries, two properties are used: a warmup, and an update frequency. The training procedure is illustrated in algorithm 13

The warmup lets $\mathbf{S}_c$ to be initialized during the warmup phase, where the regularization term $\mathcal{R}$ is set to 0. In fact, it represents a number of training steps k_0 such that for each step $i \in [0, k_0[$, the student network is trained without taking into consideration the regularizer.

The update frequency is one way to simulate task boundaries. Let k be the interval (in training epochs) between two updates. Then, k can be used to simulate that task boundary after k training epochs. Therefore, k is an hyperparameter for these three methods.

Algorithm 13 Online adaptation of offline algorithms

```
1  Online dataset $\mathcal{D}$, $\mathbf{S}_c$'s parameters $\theta$, warmup epochss $k_0$, update frequency $k$, current
   number of training epochs $t$
2  function Online_training($\mathcal{D}$, $\theta$, $k_0$, $k$, $t$)
3      for batch $X \in \mathcal{D}$ do
4          $\mathcal{L} \leftarrow 0$
5          if $t < k_0$ then
6              $\mathcal{L} \leftarrow \sum_{x \in X} L(\mathbf{S}_c(x;\theta), \mathbf{T}(x))$
7          else
8              $\mathcal{L} \leftarrow \sum_{x \in X} L(\mathbf{S}_c(x;\theta), \mathbf{T}(x)) + \mathcal{R}$
9          Optimizer($\nabla\mathcal{L}$, $\theta$)
10     if $t\%k == 0$ then
11         Simulate end of task. Update $\mathcal{R}$
12     $t \leftarrow t + 1$
```

3.4 Evaluation methodology

To evaluate the student network $\mathbf{S}_c$ (or equivalently $\mathbf{S}$), we design adequate metrics in order to evaluate the performance and the forgetting of the student network through time.

First, a task-specific metric is needed to evaluate the performances of the model over time. This task-specific metric could be accuracy for classification, or the **mean Intersection over Union (mIoU)** for semantic segmentation.

Second, it is also desired to know the performances of the student network on other domains when being trained on the current domain. Evaluating the model on already-seen domains could show the forgetting of the model.

Finally, another way to evaluate forgetting is to evaluate the performances on the model near domain shifts. Indeed, if this metric value is high, this would mean that the model did not suffer from a drop of its performances due to the domain shift, thus confirming that the model did not suffer from catastrophic forgetting.

3.4.1 Performances over time

The performance of the student network $\mathbf{S}_c$ (or equivalently $\mathbf{S}$) over time is evaluated according to its task-specific metric $\mathcal{M}$ (mIoU for semantic segmentation, accuracy for classification, ...). At time i', the student network $\mathbf{S}_c$ is evaluated on a set of size I frames $X_i' = \{x_{i'}, ..., x_{i'+I}\}$ and pseudo ground truths $\tilde{Y}_i' = \{\mathbf{T}(x_{i'}), ..., \mathbf{T}(x_{i'+I})\}$:

$$\mathcal{M}(\mathbf{S}_c(X_{i'}; \theta_{i'}, \tilde{Y}_i)$$

where $\theta_{i'}$ are the parameters of $\mathbf{S}_c$ at time i'. Since $\mathbf{S}_c$ operated at a different rate $r_{\mathbf{S}_c}$ than $\mathbf{S}$, the training of $\mathbf{S}_c$ and the update of $\mathbf{S}$ may be asynchronous.

3.4.2 Backward Transfer

To evaluate the forgetting of the student network over time, the student network is evaluated on other domains when being trained on the current domain. To do this, the student network is evaluated on already-seen samples from the past, belonging to previous domains. **Backward Transfer (BWT)** [42] is a mean to measure the influence that learning on a new domain has on the performance on past domains.

Originally, **BWT** was designed for an offline setting because evaluating on past tasks requires to know these tasks boundaries. In this work, a modified version of **BWT** is proposed to evaluate forgetting of some model for online streams. The student network's forgetting at time i' is evaluated with respect to previous data corresponding to the previous domain:

$$\mathrm{BWT}(i') = \mathcal{M}(\mathbf{S}_c(X_{i'-h};\theta_{i'}), \tilde{Y}_{i'-h}) \ ,$$

where h is the backward time shift.

To illustrate the effect of the metric, let's assume an online stream $\mathcal{V}$ being an ordered set of set of samples $\mathcal{V} = \{\mathcal{V}_1^A, \mathcal{V}_1^B, \mathcal{V}_2^A, \mathcal{V}_2^B, ...\}$ where $\mathcal{V}_i^A$ and $\mathcal{V}_i^B$ are set of frames from domain D^A and domain D^B respectively such that all sequences have the same length: $|\mathcal{V}_1^A| = |\mathcal{V}_1^B| = |\mathcal{V}_2^A| = |\mathcal{V}_2^B| = ...$ Setting $h = |\mathcal{V}_1^A|$ means that, at time i', the set of frames $X_{i'}$ and the set of frames $X_{i'-h}$ do not belong to the same domain. If the performance of the student network increases over time, i.e. from time $i', i'+1, i'+2, ...$ and its performances decreases on the sets of frames from the past, i.e. at time $i'-h, i'-h+1, i'-h+2, ...$, this would be an indication that $\mathbf{S}_c$ is getting better on the current domain but **forgets** the previous domain. In the other hand, if the performances of $\mathbf{S}_c$ on the current domain increases over time and the performance of the $\mathbf{S}_c$ remain unchanged for the previous domain, it indicates that $\mathbf{S}_c$ has been able to retain previous knowledge, thus alleviating catastrophic forgetting. An illustration is given at Figure 3.2

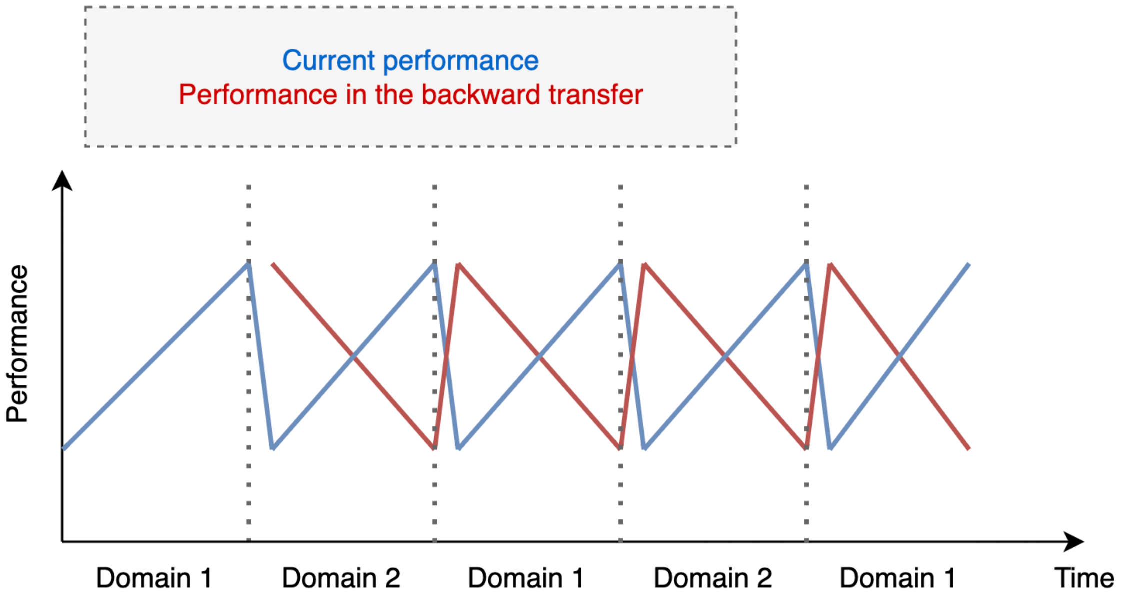


Figure 3.2. Theoretical **Backward Transfer** on the baseline with online stream $\mathcal{V}$. The blue line shows the performance of $\mathbf{S}_c$ at time i'. The red line shows the **BWT** at time i'. In the student network's training, the weights of $\mathbf{S}_c$ are fine-tuned according to the current domain it perceives. As a consequence, $\mathbf{S}_c$ improves its performances on the said domain. However, the **BWT** decreases, which indicates that $\mathbf{S}_c$ forgets the previous domain. The **BWT** is not defined at the beginning because there is no previous domain. At the time of the domain shift, the student network suffers from a drop of performances, but the **BWT** increases by a lot. This is because, at the time of the domain shift, $\mathbf{S}_c$ has trained on a domain that represents the new domain in the backward transfer.

3.4.3 Final Backward Transfer

In addition to the **BWT**, the **Final Backward Transfer (FBWT)** is a special case of **BWT** where the evaluated set of frames $\mathcal{Z}$ is the whole online stream $\mathcal{V}$ and h is set to 0. Formally, if the length of the stream is K, the **FBWT** is defined as follows:

$$\text{FBWT}(\mathcal{V}) = \mathcal{M}(\mathbf{S}_c(\mathcal{Z};\theta_K), \mathbf{T}(\mathcal{Z})$$

where $\mathcal{Z}$ is the set of all frames contained in the online stream $\mathcal{V}$, and θ_K are the parameters obtained when training on $\mathcal{V}$.

This metric gives the overall performance of the student network $\mathbf{S}_c$ on the stream it has trained on. This metric is interesting in the context of catastrophic forgetting because if $\mathbf{S}_c$ suffers from catastrophic forgetting, then $\mathbf{S}_c$ would very likely be performant on only one domain, and suffering from bad performances on the other domains. Taking the definition of $\mathcal{V}$ as earlier, an illustration of the metric with the online strearn $\mathbf{V}$ is given at Figure 3.3.

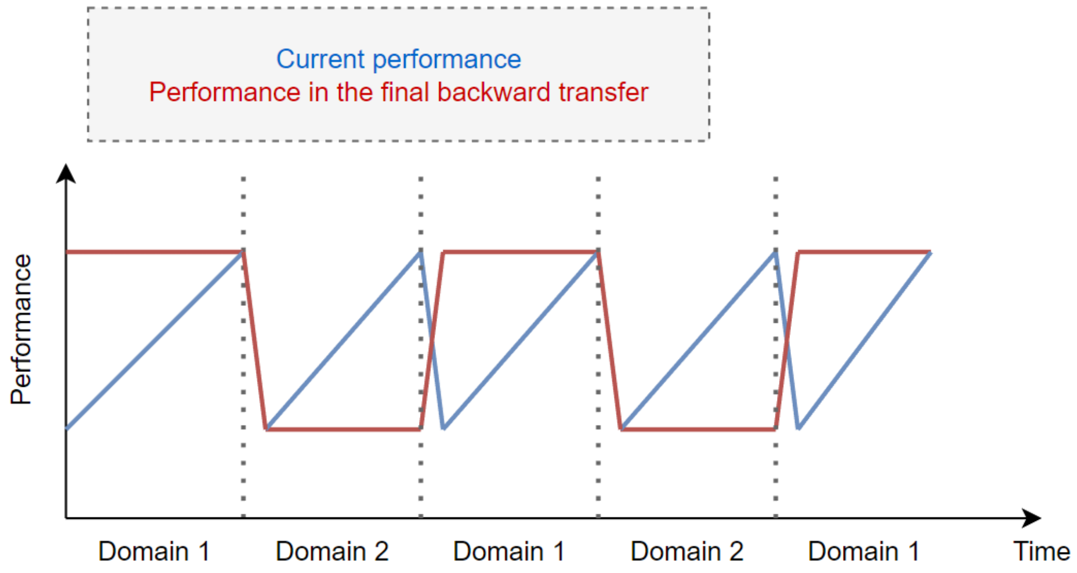


Figure 3.3. Theoretical **Final Backward transfer** on the online distillation baseline of Cioppa et al. [15] with online stream $\mathcal{V}$. The blue line shows the performance of $\mathbf{S}_c$ being trained, and the red line shows the performance of the final version of $\mathbf{S}_c$ when being trained on the entire stream. If the last domain $\mathbf{S}_c$ has trained on is domain 1, the performance over time of the final $\mathbf{S}_c$ would show good performance on domain 1 and bad performance on domain 2. This behavior happens because $\mathbf{S}_c$ has fine-tuned its parameters on the latest domain without taking into consideration the others domains.

3.4.4 Forward Transfer

Similar to **BWT**, **Forward Transfer (FWT)** excepts that it is a mean to measure the influence that learning on a new domain has on the performance on future domains.

Originally, **FWT** was designed to evaluate the zero-shot capabilities of a model. In this work, **FWT** evaluates the student network $\mathbf{S}_c$ on future (therefore, unseen) samples from the online stream belonging to a possibly already seen domain. A motivation to use **BWT** and **FWT** is to check for forgetting for sure, but also overfitting. Indeed, for instance, if $\mathbf{S}_c$ shows a high **BWT** but a low **FWT** on the online stream $\mathcal{V}$, this might indicate that $\mathbf{S}_c$ only overfitted already-seen frames, thus did not alleviate forgetting. An illustration of **FWT** on online stream $\mathcal{V}$ is shown in Figure 3.4.

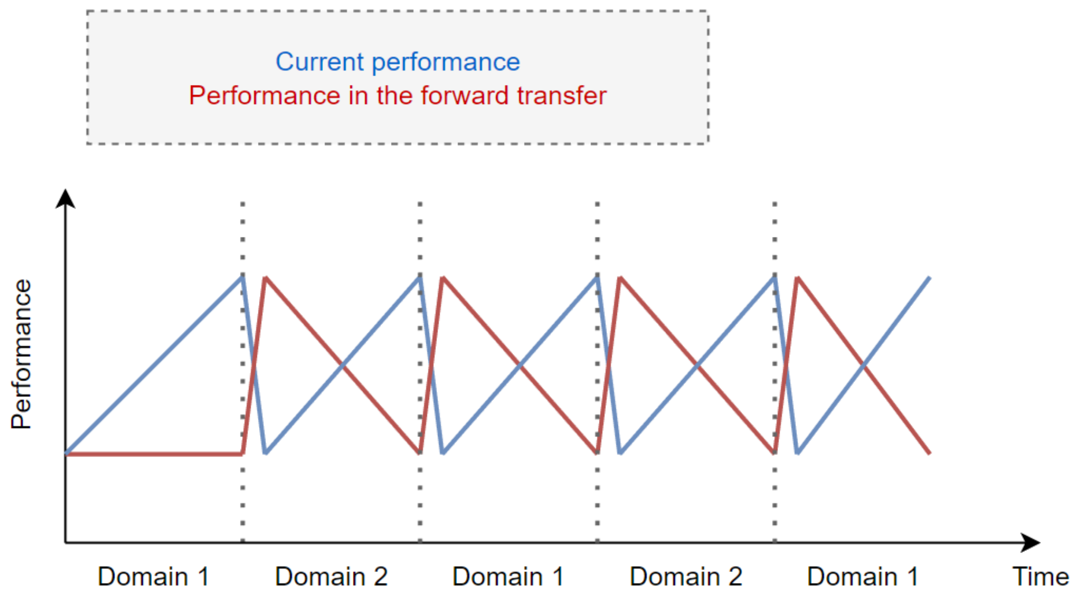


Figure 3.4. **Forward transfer** on the baseline with online stream $\mathcal{V}$. At time i', the performances of the student network (blue line) and the **FWT** (red line) are displayed. At the beginning, $\mathbf{S}_c$'s **FWT** is very low; the student network has never seen the second domain. Then, the behavior of the student network with the baseline is the same as in Figure 3.2.

3.4.5 Task-specific metric near domain shifts

A simple, yet effective metric, is to record the task-specific metric $\mathcal{M}$ near the times a domain shift occurs. Indeed, catastrophic forgetting leads to a huge drop of the performances of the student network $\mathbf{S}_c$ (equivalently $\mathbf{S}$) when the domain changes.

To do this, a time window of size $2h+1$ is defined. If a domain shift occurs at time i', the task-specific metric $\mathcal{M}$ near domain shift is defined as

$$\mathcal{M}_{NDS}(i', k) = \frac{1}{2k+1} \sum_{j=i'-k}^{i'+k} \mathcal{M}(X_j, \tilde{Y}_j).$$

$\mathcal{M}_{NDS}(i', k)$ gives an average of the metric $\mathcal{M}$ of a time window of size $2k+1$ centered at i'. An illustration of the metric is given at Figure 3.5.

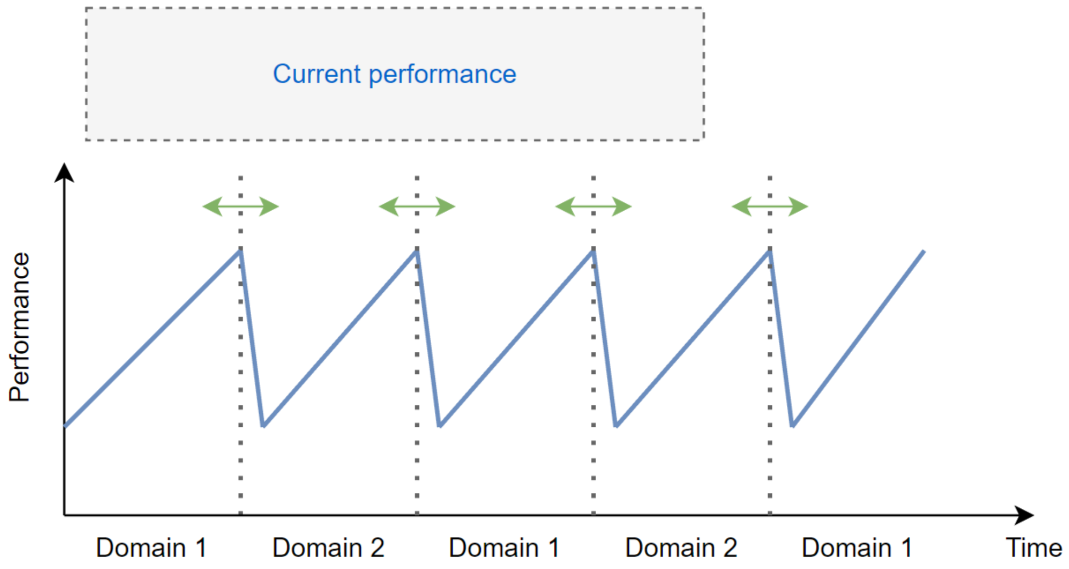


Figure 3.5. Task-specific $\mathcal{M}$ near domain shifts on the baseline with online stream $\mathcal{V}$. Green arrows represent time windows where the task-specific metric $\mathcal{M}$ recorded in each time window is averaged to produce the task-specific metric near domain shift.

Chapter 4

Experiments

In this section, the experimental setup utilized for benchmarking the continual online distillation framework is described. Subsequently, quantitative results are presented through a comparative study using a proposed evaluation methodology. To illustrate the practical impact for autonomous driving applications, some qualitative results are also showcased. Finally, an ablation study is done to demonstrate the importance of some features for alleviating catastrophic forgetting efficiently.

4.1 Experimental setup

The online continual learning framework presented is agnostic to task, metric, and training parameters, allowing it to be adaptable to diverse scenarios. The following section contains technical details of the experiments conducted in various settings, providing a thorough description of the framework.

4.1.1 Task

Benchmark are done on a real scenario wich is semantic segmentation in the context of autonomous driving. Indeed, the task consists of producing segmentation masks from a video taken by a camera located behind the windshield of a car.

Semantic segmentation is a computer vision task that involves assigning a label or category to each pixel in an image. It is a more advanced form of image segmentation, which involves dividing an image into different regions based on similar visual properties such as color, texture, or intensity.

In semantic segmentation, the goal is to not only segment the image but to assign a meaningful label or category to each pixel. This requires a more detailed understanding of the image's contents and context. For example, in an image of a street scene, semantic segmentation would identify each pixel belonging to the road, sidewalk, buildings, cars, pedestrians, and any other objects present in the scene.

In autonomous driving, semantic segmentation is used to localize objects on the road, such as pedestrians, vehicles, and road signs, which is critical for safe and effective navigation.

This task is relevant in the context of catastrophic forgetting. Indeed, it is likely that, for an undetermined amount of time, the camera does not catch any pedestrian during the trip (for instance, when driving on the highway). Due to the FIFO nature of the baseline's online dataset, the online dataset could lack of examples of what is a pedestrian and, therefore, forget its representation. This kind of behavior is not permitted as it would make autonomous driving no longer safe at all.

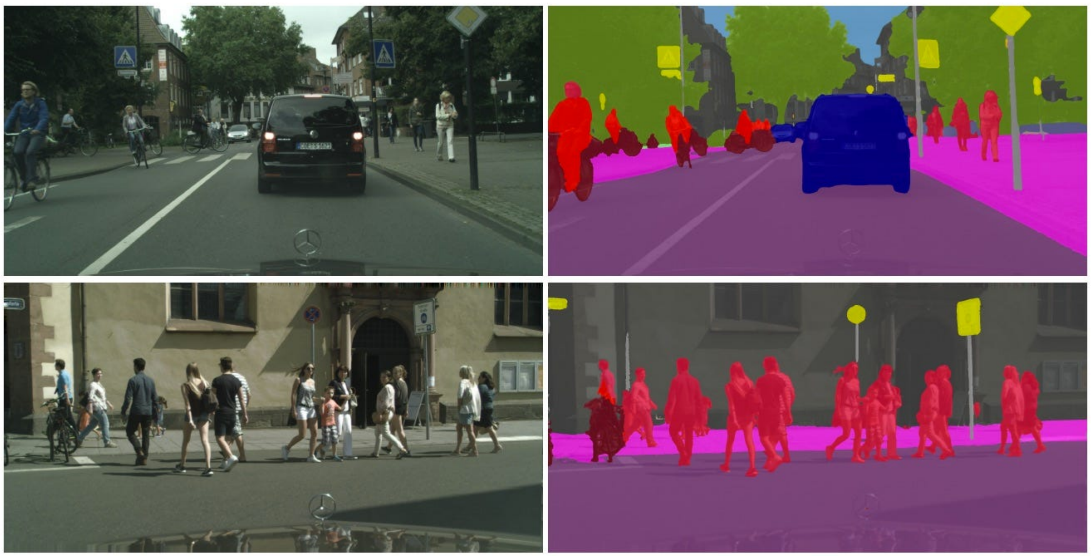


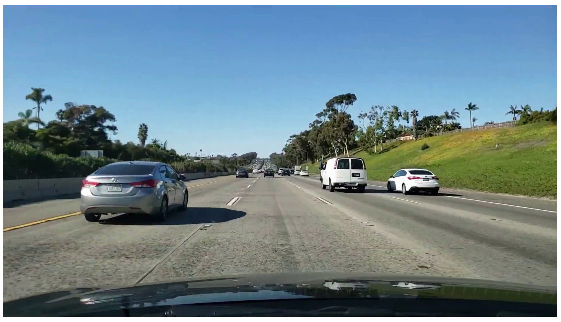
Figure 4.1. Semantic segmentation task example. Left column is the input image, right column is the segmentation mask. The task is to train the student network model **S** to produce high quality segmentation masks in the context of autonomous driving. That is, **S** must be able to detect and localize pedestrians, cars, road signs, roads, etc.

4.1.2 Dataset

The online distillation framework works on long untrimmed videos. These videos need to highlight the task's objectives and thus contain cyclic domain shifts. However, most of the semantic segmentation datasets on the internet are either frames or small video clips (*e.g.*, CityScapes [43], BDD100K [21], etc.). Therefore, it is not possible to use them.

To have a dataset that highlights the task's objectives, it has been chosen to artificially construct videos by concatenating sequences from three domains D^A, D^B and D^C by cycling from one domain to another. In the autonomous driving case, D^A is defined as a highway environment, D^B is defined as a downtown environment and D^C is defined as a forest environment. (see Figure 4.2 for more details) From these domains, two artificially videos are constructed from these environments.

(a) Downtown environment


(b) Highway environment

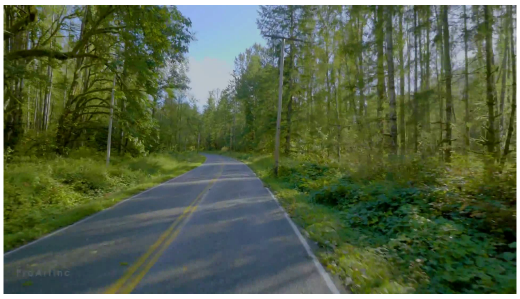

(c) Forest environment

Figure 4.2. Considered environments (or domains). In the downtown environment (Figure a), there are a lot of objects such as cars, pedestrian, traffic lights, buildings, ... In the highway environment (Figure b), there is a clear sky, almost no buildings, no pedestrian. The most common objects are cars and the road. In the forest environment (Figure c), almost no cars are seen, trees are all around. There three environments are relevant towards the task that needs to be achieved; they all have their own properties and the goal of the student network **S** is to be able to be performant on all these domains without suffering from catastrophic forgetting. The student network must not forget what is a pedestrian or a traffic light as it would be a huge problem for autonomous driving applications.

The first video is constructed from two domains: D^A and D^B. The resulting video is an ordered set $\mathcal{V}_1 = \{\mathcal{V}_1^A, \mathcal{V}_1^B, \mathcal{V}_2^A, \mathcal{V}_2^B, ...\}$ where $\mathcal{V}_i^A$ is a sequence belonging to domain D^A and $\mathcal{V}_i^B$ is a sequence belonging to domain D^B. For this video, two versions are proposed: one with 20 minutes sequences, and one with 40 minutes sequences. The duration of the sequences are relevant in this continual online distillation framework because longer sequences have an impact on the forgetting of the model. The experiments on these two versions of that video are presented in the paper Houyon *et al.* [1] and in this document.

The second video is constructed from three domains: D^A, D^B and D^C. The resulting video is an ordered set $\mathcal{V}_2 = \{\mathcal{V}_1^B, \mathcal{V}_1^C, \mathcal{V}_1^A, \mathcal{V}_2^B, \mathcal{V}_2^C, \mathcal{V}_2^A ...\}$ where $\mathcal{V}_i^A$ is a sequence belonging to domain D^A, $\mathcal{V}_i^B$ is a sequence belonging to domain D^B and $\mathcal{V}_i^C$ is a sequence belonging to domain D^C. For this video, each sequence is 20 minutes long. The experiments on that video are not presented on the paper.

4.1.3 Evaluation metric

Mean Intersection over Union

The **mean Intersection over Union (mIoU)** is a commonly used evaluation metric in computer vision tasks, such as image segmentation and object detection. It measures the similarity between two sets of points: the predicted set and the ground truth set. The formal definition of mIoU is:

$$\text{mIoU} = \frac{1}{C}\sum_{i=1}^{C} \text{IoU}_i$$

where C is the total number of objects or regions being evaluated, IoU_i is the intersection over union (IoU) for the i-th object or region.

The IoU_i for each object or region is defined as:

$$\text{IoU}_i = \frac{\text{Area of overlap between the predicted set and the ground truth set}}{\text{Area of union between the predicted set and the ground truth set}}$$

where the "Area of overlap" is the intersection between the predicted set and the ground truth set, and the "Area of union" is the union of the two sets.

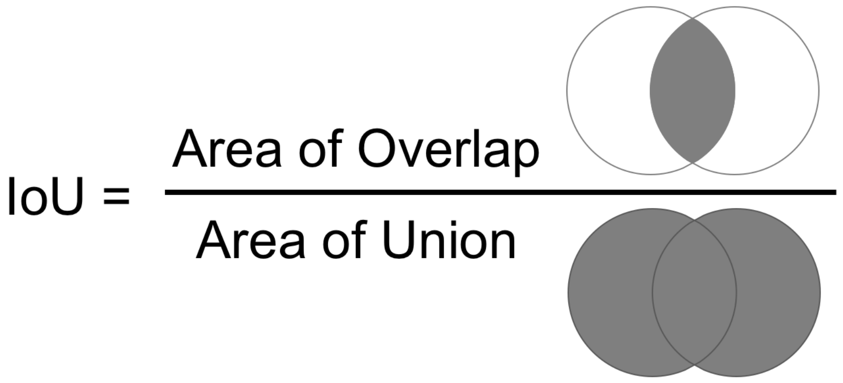


Figure 4.3. Computation of the IoU.

The mIoU ranges from 0 to 1, with higher values indicating better performance. A value of 1 indicates perfect overlap between the predicted and ground truth sets, while a value of 0 indicates no overlap.

What does the metric really evaluate?

In the online distillation framework, the ground truth set is not available. Instead, pseudo-groundtruths are available; outputs from the teacher network **T**. Generally, the teacher network is assumed to generate outputs that are close enough to the real groundtruth.

That being said, the metric evaluates the capacity of the student network **S** to imitate the teacher network **T**. If it is assumed that, for each frame x:

$$\mathbf{T}(x) \approx Y,$$

where Y is the real groundtruth, then

$$\text{mIoU}(\mathbf{S}(x), \mathbf{T}(x)) \approx \text{mIoU}(\mathbf{S}(x), Y)$$

is a good approximation on the capacity of the student network $\mathbf{S}$ to achieve the task.

Metrics

For each video, the mIoU, the BWT, the FBWT, the FWT and the mIoU near domain shifts metrics are provided during the whole video. The average of these metrics over the whole video is also provided. The considered set of frames for evaluation is set to 1 minute and k is set to 2 minutes for the mIoU near domain shifts metric. Finally, depending on the sequence length, the parameter h from the BWT and FWT is set to 20 or 40. (20 if the sequence length is 20 minutes, 40 if the sequence length is 40)

4.1.4 Teacher network

It is important to choose a teacher network $\mathbf{T}$ that is able to produce the best pseudo-groundtruths as possible. The presented network is Segformer [26], a network using transformers (and not Convolutional Neural Networks) to perform semantic segmentation. Furthermore, it is the state of the art performance for semantic segmentation tasks on popular semantic segmentation datasets.

Image as a sequence

Given an image of size $H \times W \times C$, this image can be divided into patches of size $M \times M$ that are fed one after the other to the Transformer architecture.

Architecture

The Segformer architecture (see Figure 4.4) is performant thanks to several components. First, this is a positional-encoding free Transformer. In older versions of Transformer architectures for semantic segmentations, positional encoding was used to introduce the location information. However, if the resolution at test time is different from the resolution at train time, the model suffers from a drop of its performances as the positional code needs to be interpolated. To adress this issue, positional information can be provided by a simple convolution layer infine the feed-forward network (Refered as Mix-FFN in Figure 4.4).

Second, Transformers are known to be computationally demanding, especially due to the self-attention layer. Self-attention is computed as

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_{head}}})V$$

where $Q = K = V$ all have the same dimension $N \times C$ where $N = H \times W$ is the length of the sequence and C is the number of channels. Self-attention has a time complexity

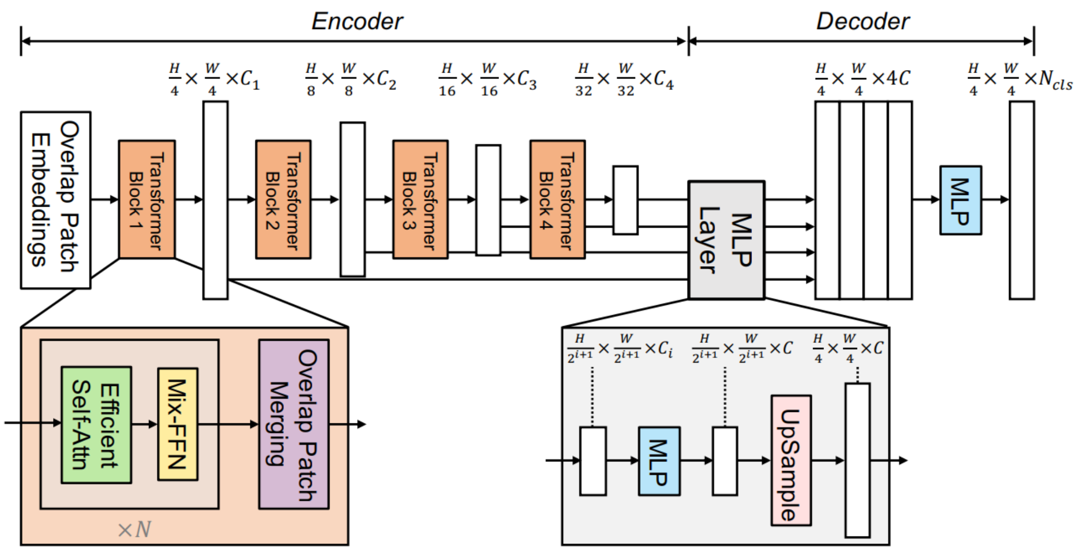

Encoder
Decoder
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
$\frac{H}{4}\times\frac{W}{4}\times 4C$
$\frac{H}{4}\times\frac{W}{4}\times N_{cls}$
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Efficient Self-Attn
Mix-FFN
Overlap Patch Merging
$\times N$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C_i$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C$
$\frac{H}{4}\times\frac{W}{4}\times C$
MLP
UpSample

Figure 4.4. Segformer architecture from [26]. It consists of two parts. First, a hierarchical Transformer encoder to extract features. Then an All-MLP decoder taking the global and local features induced by the Transformer encoder to produce the semantic segmentation mask.

of $O(N^2)$, which is computationally expensive for high resolution images. To adress this issue, the length of the sequence can be reduced by a ratio R as follows:

$$\hat{K} = \text{Reshape}(\frac{N}{R}, C \cdot R)(K)$$
$$K = \text{Linear}(C \cdot R, C)(\hat{K})$$

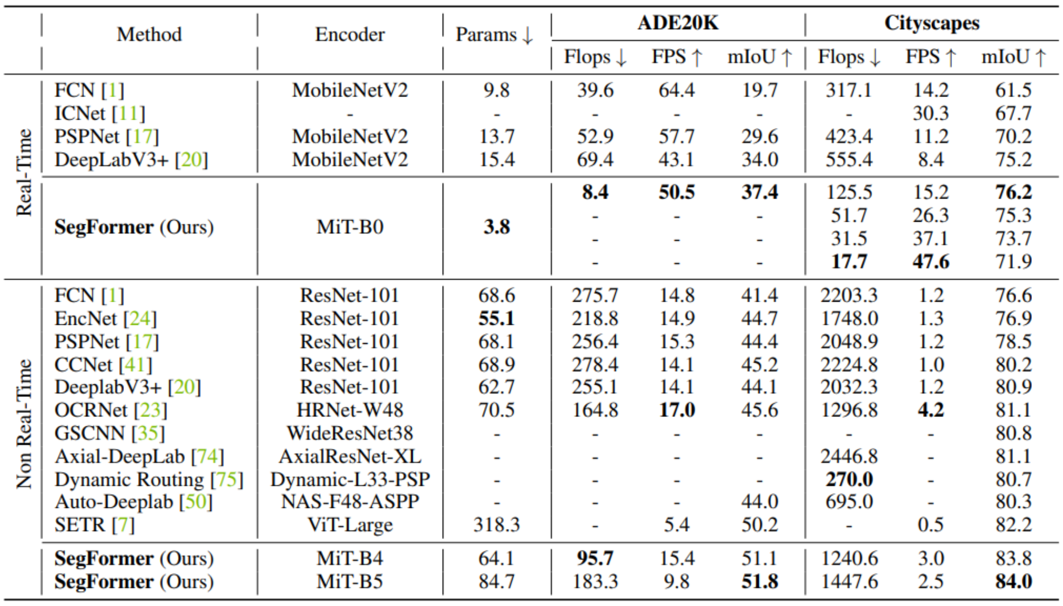
$\hat{K}$ is K reshaped to dimensions $\frac{N}{R} \times C \cdot R$, which then produces a new K when $\hat{K}$ is passed to a linear layer producing $\frac{N}{R} \times C$ outputs. Therefore, the complexity of the self-attention mechanism becomes $O(\frac{N^2}{R})$ instead of $O(N^2)$. This trick enables the training of bigger transformer architectures and thus improve the performance on some tasks.

Finally, the multi-level features (i.e. the features extracted from all Transformer blocks) are fed to the multi-layer perceptron, then upsampled to gather a segmentation mask of dimension $\frac{H}{4} \times \frac{W}{4} \times N_{cls}$, where N_{cls} is the number of categories.

Training

The Segformer architecture is taken from an open source semantic segmentation toolbox which is open source (source: `https://github.com/open-mmlab/mmsegmentation`). It has been trained on the CityScapes dataset [43]. Cityscapes provides high-quality, pixel-level annotations for a diverse set of urban images captured in different cities across Germany. The dataset consists of images taken from the perspective of a moving vehicle and covers a wide range of urban scenes, including streets, intersections, sidewalks, buildings, vehicles, and pedestrians.

Segformer trained on Cityscapes is relevant for the autonomous driving task. Furthermore, Segformer is able to produce quality predictions that are close to the real groundtruth.



	Method	Encoder	Params ↓	ADE20K			Cityscapes		
				Flops ↓	FPS ↑	mIoU ↑	Flops ↓	FPS ↑	mIoU ↑
Real-Time	FCN [1]	MobileNetV2	9.8	39.6	64.4	19.7	317.1	14.2	61.5
	ICNet [11]	-	-	-	-	-	-	30.3	67.7
	PSPNet [17]	MobileNetV2	13.7	52.9	57.7	29.6	423.4	11.2	70.2
	DeepLabV3+ [20]	MobileNetV2	15.4	69.4	43.1	34.0	555.4	8.4	75.2
	SegFormer (Ours)	MiT-B0	**3.8**	**8.4**	**50.5**	**37.4**	125.5	15.2	**76.2**
				-	-	-	51.7	26.3	75.3
				-	-	-	31.5	37.1	73.7
				-	-	-	**17.7**	**47.6**	71.9
Non Real-Time	FCN [1]	ResNet-101	68.6	275.7	14.8	41.4	2203.3	1.2	76.6
	EncNet [24]	ResNet-101	**55.1**	218.8	14.9	44.7	1748.0	1.3	76.9
	PSPNet [17]	ResNet-101	68.1	256.4	15.3	44.4	2048.9	1.2	78.5
	CCNet [41]	ResNet-101	68.9	278.4	14.1	45.2	2224.8	1.0	80.2
	DeeplabV3+ [20]	ResNet-101	62.7	255.1	14.1	44.1	2032.3	1.2	80.9
	OCRNet [23]	HRNet-W48	70.5	164.8	**17.0**	45.6	1296.8	**4.2**	81.1
	GSCNN [35]	WideResNet38	-	-	-	-	-	-	80.8
	Axial-DeepLab [74]	AxialResNet-XL	-	-	-	-	2446.8	-	81.1
	Dynamic Routing [75]	Dynamic-L33-PSP	-	-	-	-	**270.0**	-	80.7
	Auto-Deeplab [50]	NAS-F48-ASPP	-	-	-	44.0	695.0	-	80.3
	SETR [7]	ViT-Large	318.3	-	5.4	50.2	-	0.5	82.2
	SegFormer (Ours)	MiT-B4	64.1	**95.7**	15.4	51.1	1240.6	3.0	83.8
	SegFormer (Ours)	MiT-B5	84.7	183.3	9.8	**51.8**	1447.6	2.5	**84.0**

Figure 4.5. Performances of SegFormer on the ADE20K and Cityscapes datasets from [26]. Segformer shows a significant improvement on accuracy with respect to others known models.

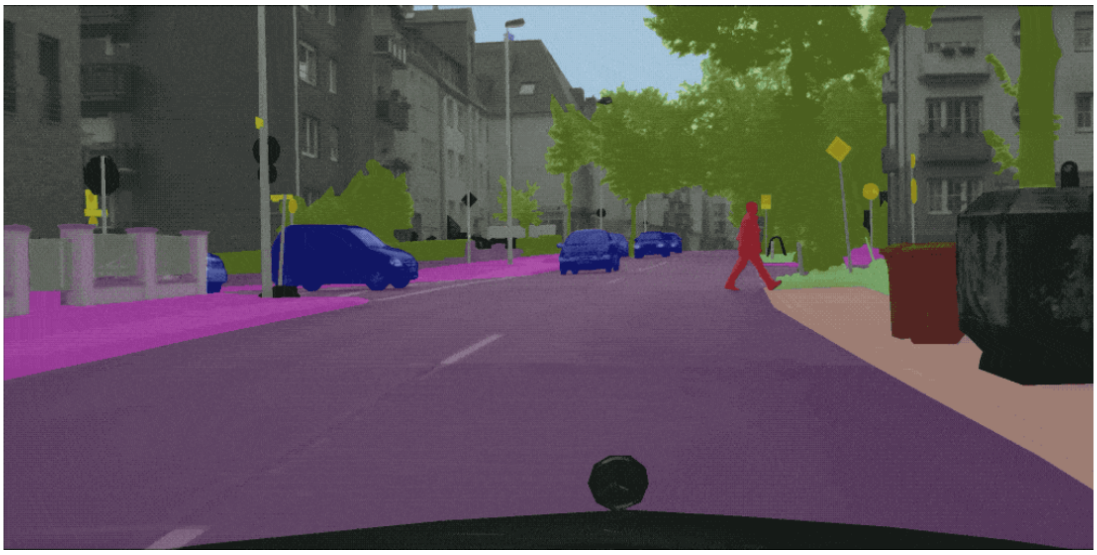


Figure 4.6. Segmentation mask from the Segformer architecture on the Cityscapes dataset. Segformer is able to recognize all the objects and regions in the image.

For the experiments, the teacher network $\mathbf{T}$ is assumed to process a frame at a rate of $r_{\mathbf{T}} = 3$ seconds per frame, which is the assumed framerate of the network teacher $\mathbf{T}$ used in Cioppa et al. [15]. In fact, the framerate of Segformer is higher, but it is decided to make the setup more difficult because the online distillation may be performed on embedded devices that may not process frames faster.

4.1.5 Student network

As for the student network $\mathbf{S}$ and $\mathbf{S}_c$, the TinyNet architecture is chosen [15, 29]. TinyNet is a lightweight network for semantic segmentation which a scaled-down version of the

PSPNet architecture [44].

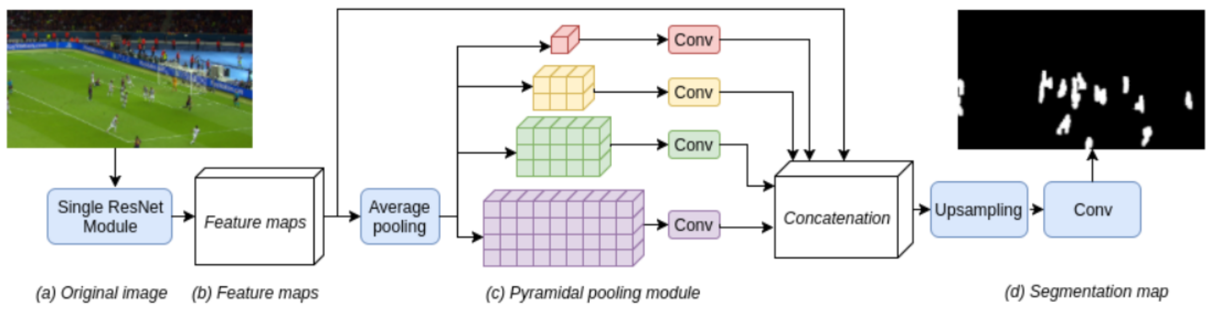


Figure 4.7. Overview of TinyNet taken from [29]. It is composed of four components. First, the original image (a) that will be segmented. Second, a feature map (b) which is a simple ResNet module [2] that is used to extract feature maps. Third, a pyramidal module that helps at gathering context information. Finally, an upsampling block in order to produce the final segmentation map. Initially, TinyNet was proposed for line and player segmentation on sport videos. However, it is completely viable for other fields such as autonomous driving.

PSPNet introduced a pyramidal pooling module which can capture multi-scale contextual information from an input image. Pyramidal pooling module aggrgates global contextual information by using pooling operations at multiple scales as shown in Figure 4.7. The ResNet module outputs high-level features that capture low-level details and high-level semantic information. Then, the pyramid pooling perform pooling operations at multiple scales; the feature map is splited into different regions and each region are independently processed. Finally, all the pooling results are concatenated and upsampled to provide the prediction.

TinyNet requires few sample to train, it can adapt to new scenes quite fast. Furthermore, due to its low size (way less parameters than classical models like PSPNet or Segformer), it makes predictions very fast. For the experiments, the student network is assumed to process at a rate $r_S = 30$ frames per second.

4.1.6 Experiment procedure

All the experiments are done on already recorded videos. In this subsection, explanations on how the annotated frames are produced and how the student is evaluated are given.

Annotated frames

Given a video at 30 FPS, a pseudo-annotated frame is produced every 90 frames, representing a frame every 3 seconds. Once all the annotated frames are gathered, they will be used to perform online distillation. This is done as such because we do not want to re-compute all pseudo-groundtruths whenever an experiment is performed.

Evaluation of the student network

Given the annotated frames from the teacher and the video, the online distillation is performed as follows:

1. Start with an empty replay buffer $\mathcal{D}_0 = \{\}$ and initialize $\mathbf{S}_c$ with θ_0. Set $t = 0$
2. At time t, a new batch of annotated frames X_t arrives, representing 1 minute of video (20 annotated frames).
3. A dataset $\mathcal{D} = X \cup f_S(\mathcal{D}_t)$ is used to perform one epoch on the student network $\mathbf{S}_c(\cdot; \theta_t)$
4. $\mathbf{S}_c(\cdot; \theta_t)$ becomes $\mathbf{S}_c(\cdot; \theta_{t+1})$ after training. $\mathbf{S}_c(\cdot; \theta_{t+1})$ is evaluated on the sets of frames X_{t+1}, X_{t+1-h} and X_{t+1+h} to produce the mIoU, the BWT (if possible) and the FWT (if possible) respectively.
5. Update the buffer $\mathcal{D}_{t+1} = f_U(\mathcal{D}_t, X_{t+1})$. Increment t.
6. if it is not the end of the stream, go back to 2. Otherwise, compute the FBWT and the mIoU NDS.

4.1.7 Hyperparameters

Student network

The student network $\mathbf{S}_c$ is trained from scratch at the beginning of the video. ADAM optimizer is used at a learning rate of 10^{-4}. The batch size if set to 1, which follows the setup from Cioppa et al. [15] that does not use a decreasing learning rate because the data stream may be infinite and it is desirable that the student networks keeps training on it.

Replay-based methods

The buffer size is set to $M = 250$ for the first experiment and it is set to $M = 150$ for the second experiment (this gives an intuition on the impact of the capacity of the replay buffer on the performance of the student network). The number of selected frames is set to $N = 100$. For the Prioritized strategy, the prioritization parameter α is set to 3. For the MIR strategy, the number of samples C to apply the search criterion is set to 150.

Regularization-based methods

To simulate the boundaries, the warmup (in terms of epochs) is set to 10 epochs. For the update frequency, it is set to 1.

MAS and RWalk have an hyperparameter λ to adjust the importance of the regularizer; it is set to 1. Furthermore, they both use a decay parameter β; it is set to 0.1.

4.2 Two domains setup

In this section, performance of the baseline is compared with several continual learning approaches on the video $\mathcal{V}_1$. Memoryless methods are studied as a naïve approach; they represent methods that are not using a replay-buffer. Instead, each annotated frame from the teacher is used once for training, then deleted. Furthermore, combination of replay-based and regularization-based methods are studied to assess their effects (if they are benefical or not). Replay-buffer capacity is set to $M = 250$.

4.2.1 Quantitative results

Methods	**Parameters**			**Metrics (mean %)**				
	f_S	f_U	$\mathcal{R}$	*mIoU*	*mIoU* NDS	FWT	BWT	Final BWT
Memoryless	/	/	/	18.4/19.4	14.9/15.1	6.8/4.8	7.8/7.5	14.9/15.0
	/	/	MAS	14.0/14.0	13.0/13.3	11.1/11.1	12.9/12.9	14.2/14.2
	/	/	LwF	15.7/15.9	12.0/11.0	9.7/6.8	11.3/8.9	14.7/12.9
	/	/	RWalk	18.3/19.3	14.6/14.7	7.5/4.7	8.6/6.5	15.1/14.2
Baseline	All	FIFO	/	23.4/24.2	19.8/18.2	14.5/9.5	17.7/13.9	21.9/19.9
Replay Buffer	Uniform	Uniform	/	25.5/25.0	23.6/21.1	**22.2**/17.3	30.6/28.8	29.4/28.4
	Prioritized	Prioritized	/	25.1/25.1	23.2/20.8	21.3/17.3	29.2/28.4	29.2/28.9
	MIR	Uniform	/	25.2/25.2	23.7/**24.5**	21.9/**17.9**	30.5/28.6	29.5/29.7
	MIR	Uniform	MAS	14.5/14.9	13.4/14.7	12.1/13.6	13.9/15.2	15.1/15.4
	MIR	Uniform	LwF	18.7/18.1	17.6/15.7	17.4/13.9	21.0/20.2	22.4/21.1
	MIR	Uniform	ACE	**25.6**/**25.5**	**24.2**/21.8	22.0/17.5	**30.8**/29.4	28.8/28.5
	MIR	Uniform	RWalk	25.2/25.4	23.4/22.0	21.8/**18.0**	30.0/**30.8**	**30.1**/**30.8**

Table 4.1. (© 2023 IEEE) **Quantitative results.** The baseline by Cioppa *et al.* [15] compared to memoryless and replay-based methods. A benchmark of the different selections functions f_S, update functions f_U, and regularizers $\mathcal{R}$ are performed for each category. It displays the different metrics for the 20/40 concatenated sequences for the video $\mathcal{V}_1$. There is a noticeable difference of performance between the replay-based methods and the memoryless methods. Furthermore, the baseline remains better than memoryless methods but is outperformed by all replay-based methods except for replay-based methods using LwF and MAS regularization methods. Indeed, LwF and MAS tend to decrease the performance, while ACE and RWalk increase the performance. Overall, the Uniform method, the MIR method, MIR+ACE and MIR-RWalk show the best performance. A comparison of the temporal evolution of the baseline with some methods are given at figures 4.8, 4.9 and 4.10.

In table 4.1, we can see that memoryless methods do not perform well compared to the baseline. An illustration of Memoryless RWalk is given at Figure 4.8. This indicates the necessity to use replay-based methods in order to improve the performance of the student network over time. Moreover, replay-based methods work pretty well without any regularizer; they all outperform the baseline, where MIR shows the best results among all the replay-based only methods. However, adding a regularizer to these replay-based methods are not always benefical; LWF and MAS, combined with the MIR method, perform clearly worse than the baseline. An hypothesis could be that the regularizer prevents the student from adapting to new domains, meaning that when trying to learn the second domain, the regularizer completely blocks the training of the student in order to remember the first domain. Furthermore, the size of the model could also play a role because, for MAS, constraining the change of parameters could be a too huge penalty for small models.

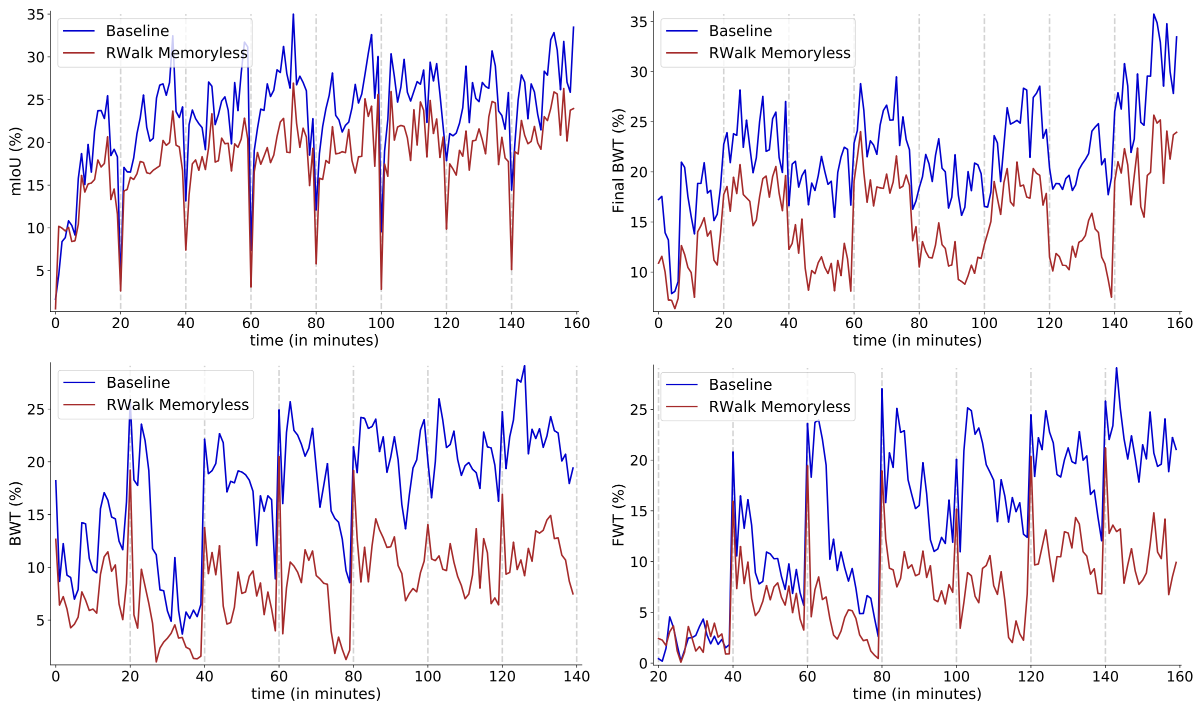


Figure 4.8. (© 2023 IEEE) **Evolution of performance over time.** The baseline is compared with the memoryless RWalk regularization method. (Top-left) mIoU: RWalk Memoryless underperform against the baseline. Furthermore, it does not even avoid catastrophic forgetting as it suffers from a huge drop of its performance at each domain shift. (This drop is even bigger than the baseline's). (Bottom-left) BWT: When evaluated on the previous domain, RWalk Memoryless underpeforms against the baseline. They are both not able to retain knowledge on already-seen training samples. (Top-Right) Final BWT: Again, RWalk Memoryless underperforms compared to the baseline. For both methods, they're only performing at their best on the latest domain they were trained on. (Bottom-right) FWT: When evaluated on the next domain, both methods do not generalize well on unseen samples from already-seen domains.

Overall, the biggest improvement is done with MIR. Nonetheless, other replay-based methods also show similar performance than MIR, such as the Uniform method shown in Figure 4.9. The Uniform method has the benefits of not being computationally expensive, at least compared to the baseline, unlike MIR who requires more computation power to work. Therefore, if the computation power is an important variable to take into account (which is the case for autonomous driving application), a more lightweight method could be used if it nearly reaches the same performance. Reguarding the Prioritized method, it underperforms compared to the Uniform method. More explanation on this will be given in the ablation (see section 4.4.1)

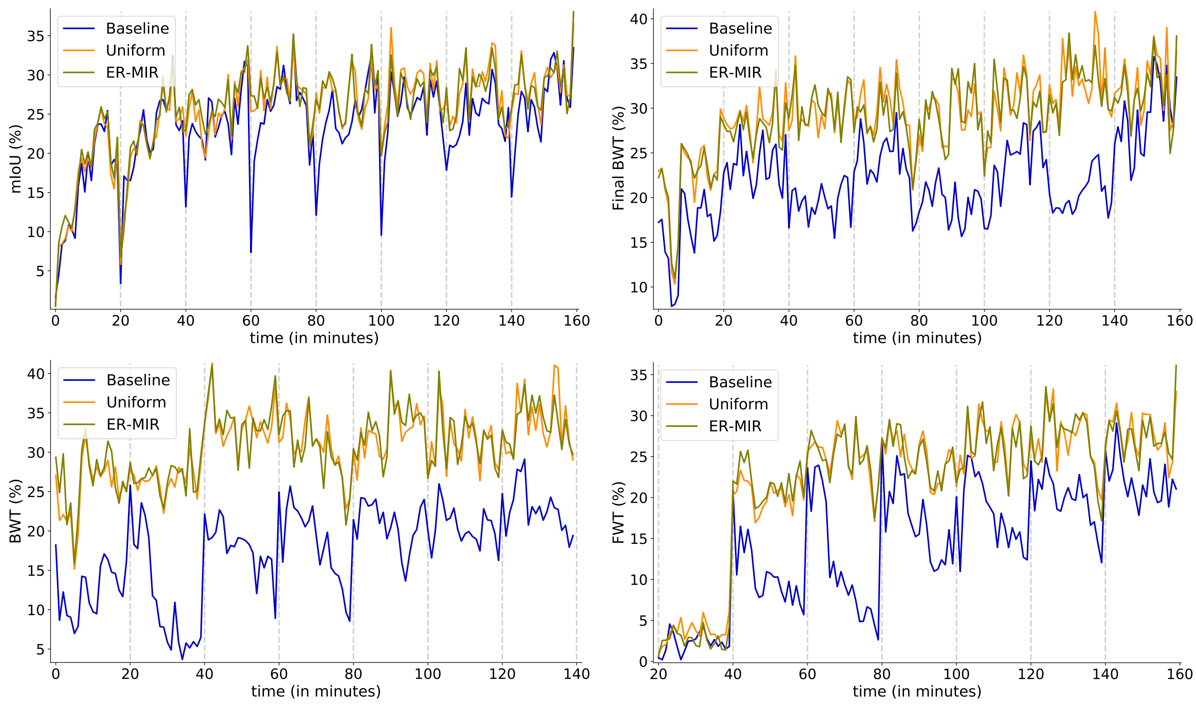


Figure 4.9. **Evolution of performance over time.** Uniform and MIR are compared with the baseline. Both methods clearly outperform the baseline on every aspect. (Top-left) mIoU: All the methods have similar performance within a domain. The difference occurs at each domain shift; Uniform and MIR do not suffer from a drop of their performance, unlike the baseline. (Bottom-left) BWT: When evaluated on previous domains, Uniform and MIR are able to retain information on past training frames. (Top-right) Final BWT: Uniform and MIR's performance remain decent on all domains, unlike the baseline who is only decent on the last domain it has trained on. (Bottom-right) FWT: Uniform and MIR are able to generalize well on new samples from already-seen domains.

The best performing methods are MIR+ACE and MIR+RWalk. For the mIoU, MIR+ACE is slightly better; an explanation could be that MIR+ACE does not only retain knowledge with the replay buffer, but it also accelerate the learning of new classes (due to the asymmetric loss). This assumption is relevant because MIR+ACE has a better mIoU NDS than MIR+RWalk. Reguarding the RWalk regularization method, it is a smarter version of MAS who keeps track of all variation in loss caused by the parameters of the model. This additional feature prevents the model from being completely frozen (not able to learn a new task).

In figure 4.10, the evolution of the performance over time for the baseline and on one of the best method (MIR+RWalk) is shown. Reguarding the mIoU plot, both methods are behaving the same way; this is expected because they both discover the two domains. The second transition shows the difference between the two methods: the baseline suffers from a drop of its performance while MIR+RWalk remains performant. This comment also holds for all the next transitions. This drop of performance by the baseline is caused by the forgetting of the previous domain. Indeed, one can confirm this assumption by checking the other metrics; MIR+RWalk outperforms the baseline in BWT which is an indication that MIR+RWalk is able to retain more information from the previous domain than the baseline. Furthermore, the FWT is also high for MIR+RWalk compared to the baseline who decreases over time on the same domain, this indicates thar MIR+RWalk can better generalize on new frames from a previous domain than the baseline. Finally, the Final-BWT is clearly better for MIR+RWalk than the baseline, indicating that MIR+RWalk does not suffer from forgetting on past domains.

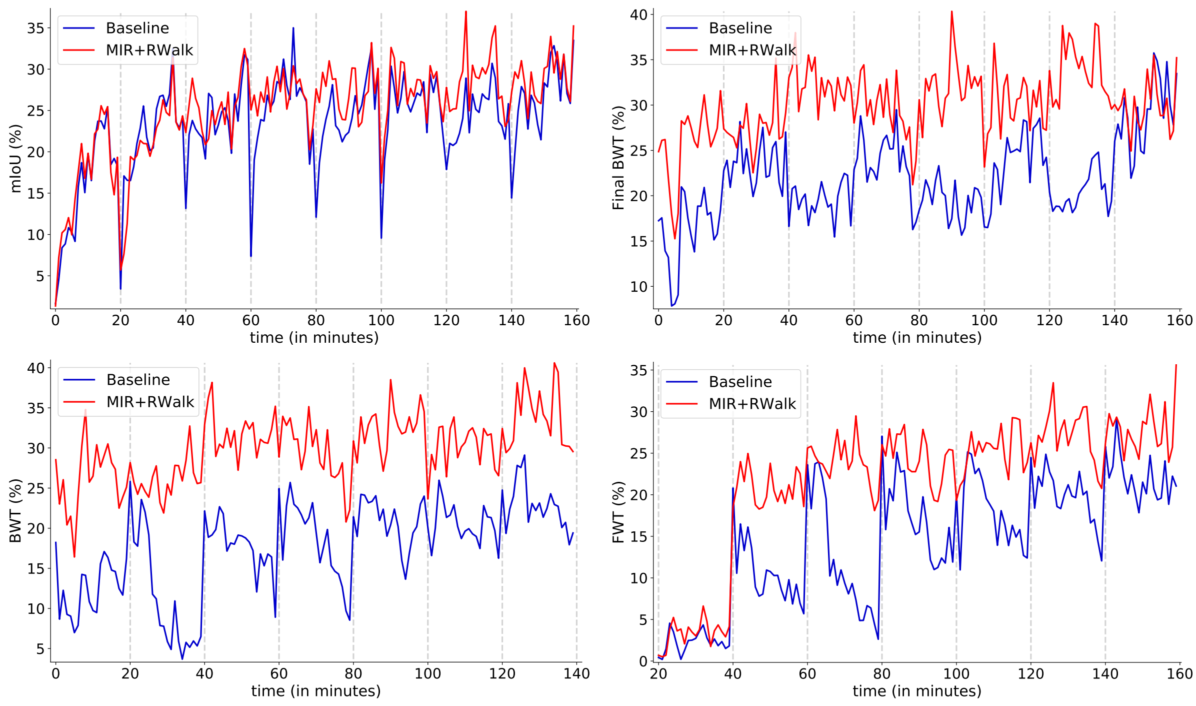


Figure 4.10. **Evolution of performance over time.** MIR-RWalk is compared with the baseline. Comments are similar than Uniform and MIR (4.9). This is the method achieving the best performance.

4.2.2 Qualitative results

The qualitative demonstration of the impact of top-performing continual learning methods on mitigating catastrophic forgetting is presented in this study. Specifically, the accuracy of segmentation masks is examined by following two specific transitions: from the highway to downtown (where the student has only encountered downtown once before) and from downtown to highway (where the student has previously experienced the highway domain six times).

In Figure 4.11, a visual comparison is shown between the segmentation masks produced by the baseline method, MIR, and MIR+RWalk, in relation to the ground-truth mask. Notably, even though the student has been exposed to the domain previously, the segmentation masks generated by the baseline approach immediately after the domain shift are notably inadequate. This deficiency could potentially lead to hazardous situations for both the autonomous vehicle and its occupants. Conversely, the segmentation masks obtained through the application of MIR and MIR+RWalk exhibit a significantly closer alignment with the ground-truth masks.

Furthermore, quantitative results corroborate these observations by highlighting the considerable improvement in prediction quality achieved through the integration of continual learning algorithms into the online distillation framework. These enhancements make the framework more practical and reliable for real-world applications.

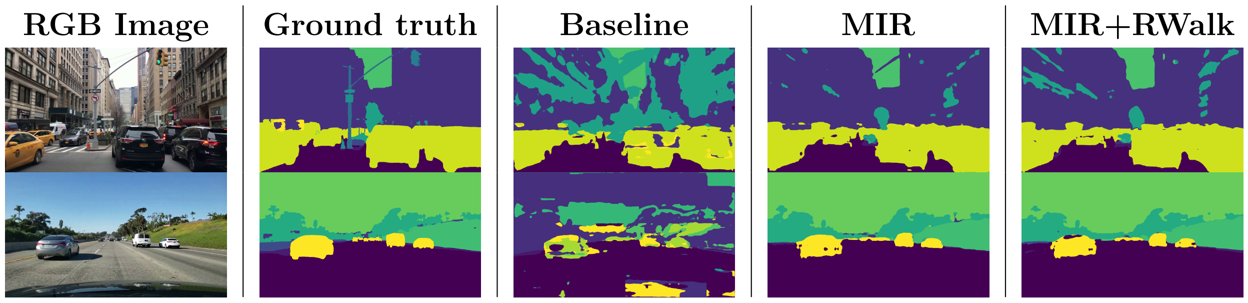


Figure 4.11. (© 2023 IEEE) **Qualitative results.** Segmentation masks from several methods at the time of a domain shift. Top row shows a frame taken right after the second domain shift (shift to the downtown environment). Down row shows a frame taken right after the seventh domain shift (shift to the highway environment). The baseline performs a poor segmentation masks right after domain shifts even though it had already seen them. MIR and MIR+RWalk manage to produce decent segmentation masks.

4.3 Three domains setup

In this section, performance of the baseline is compared with several continual learning approaches on the video $\mathcal{V}_2$. The study is the same than the two domains setup, except that this experiment involves a more difficult setup (an additional domain: the forest enviromnent). This time, the replay-buffer capacity is set to $M = 150$.

4.3.1 Quantitative results

Methods	Parameters			Metrics (mean %)				
	f_S	f_U	$\mathcal{R}$	*mIoU*	*mIoU* NDS	FWT	BWT	Final BWT
Memoryless	/	/	/	19.9	16.4	6.8	7.7	14.4
	/	/	MAS	17.0	13.9	9.3	14.1	15.5
	/	/	LwF	17.1	13.1	7.4	11.4	13.0
	/	/	RWalk	19.8	16.6	6.8	8.1	14.2
Baseline	All	FIFO	/	23.8	20.2	8.9	17.3	19.8
Replay Buffer	Uniform	Uniform	/	24.9	22.3	18.5	28.9	26.4
	Prioritized	Prioritized	/	25.0	**23.3**	**19.8**	28.8	26.9
	MIR	Uniform	/	25.1	22.7	18.9	29.3	27.1
	MIR	Uniform	MAS	16.5	15.4	14.5	19.0	18.4
	MIR	Uniform	LwF	19.8	17.6	16.7	22.3	22.6
	MIR	Uniform	ACE	24.9	22.7	18.9	**29.3**	**27.8**
	MIR	Uniform	RWalk	**25.2**	22.9	19.3	29.0	26.5

Table 4.2. **Quantitative results.** The baseline by Cioppa *et al.* [15] compared to memoryless and replay-based methods. A benchmark of the different selections functions f_S, update functions f_U, and regularizers $\mathcal{R}$ are performed for each category. It displays the different metrics for the 20 concatenated sequences for the video $\mathcal{V}_2$. Memoryless methods decrease the performance compared to the baseline. Replay-based methods outperform the baseline on all aspects. MAS and LwF regularization methods decrease the performance while ACE and RWalk increase the performance. A comparison of the temporal evolution of performance of some mehods are given in figures 4.12, 4.13 and 4.15.

In table 4.2, again, memoryless methods perform poorly compared to the replay-based methods. Indeed, they underperform on all metrics with respect to the baseline, except Memoryless-MAS who outperforms the baseline on the FWT metric. On the hand, all replay-based methods outperform the baseline. Indeed, Uniform, Prioritized and MIR show extremely good results on all metrics compared to the baseline. However, when combining replay-based methods with a regularizer, MAS and LwF decrease the performance, while RWalk and ACE increase the performance.

Figure 4.12 compares the baseline with Memoryless-MAS. Even though Memoryless-MAS underpeforms against the baseline, it shows improvement over other Memoryless methods. Indeed, Memoryless-MAS is method outperforms all other Memoryless methods on the FWT, BWT and Final BWT metrics, indicating that it is able to retain some knowledge. It can even outperform the baseline on the FWT at some time (40-60, 60-80, 100-120, 160-180). This might be due to the way its importance weights are updated. Indeed, after the warmup and at each epoch, its importance weights are being updated using exponential averaging. Therefore, Memoryless-MAS forgets less quickly (as time passes, importance weight will only focus on the current domain but it takes several updates).

Reguarding replay-based only methods, Prioritized and MIR slightly outperform Uniform. This might indicate that if the capacity of the replay buffer is limited, a smarter selection and/or update function can improve the performance. Figure 4.13 illustrates a comparison between the baseline, Prioritized and MIR. Prioritized method shows the best FWT and mIoU NDS among all methods. The reason could be that, since the replay-buffer's capacity is limited, Prioritized is more able to preserve older samples from past domains, unlike a uniform selection function who, on average, will delete these samples quicker.

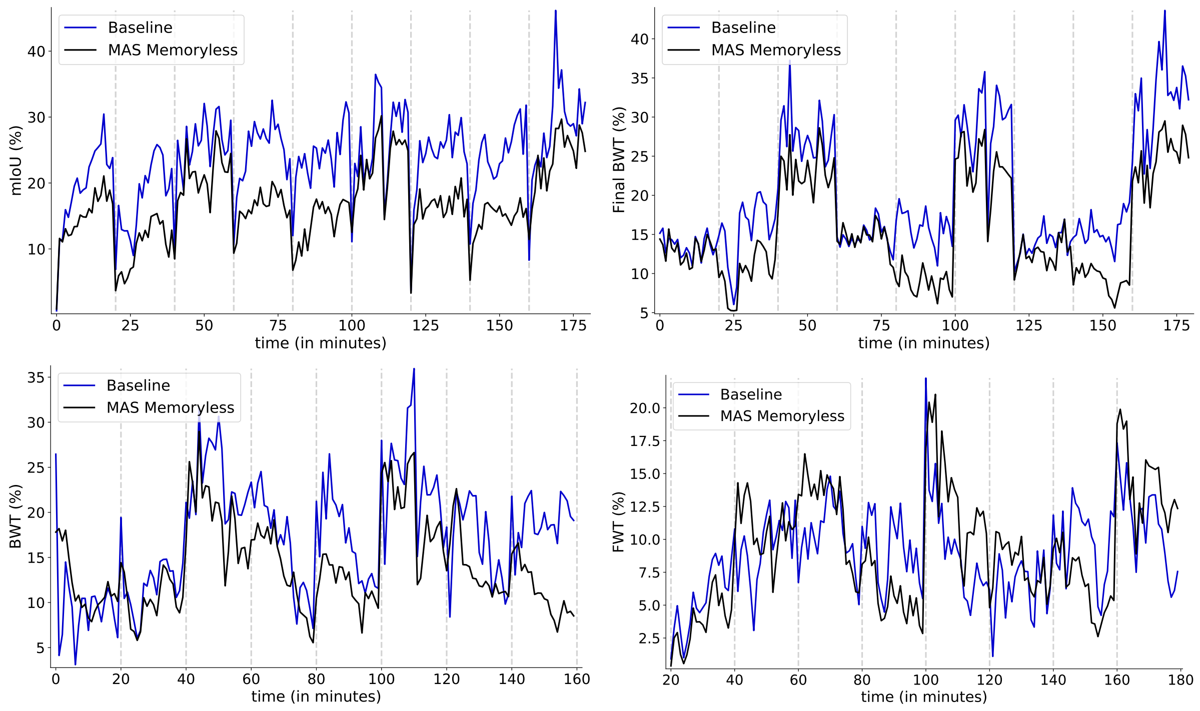


Figure 4.12. **Evaluation of performance over time.** Comparison between the baseline and Memoryless-MAS. (Top-left) mIoU: MAS underperforms against the baseline all the time. Furthermore, it also suffers from a drop of its performance during a domain shift. (Bottom-left) BWT: Memoryless-MAS underperforms against the baseline, meaning that its capacity to retain knowledge from the past is worse than the baseline. (Top-right) Final-BWT: On the first domain (0-20, 60-80, 120-140), Memoryless-MAS almost shows equal performance. Like the baseline, Memoryless-MAS performs the best on the last seen domain, and performs very poorly on the others, indicating that it did not retain knowledge from past domains. (Bottom-right) FWT: Surprisingly, Memory-less MAS is able to outperform the baseline at some time, meaning that it can better generalize on unseen samples from already-seen domains.

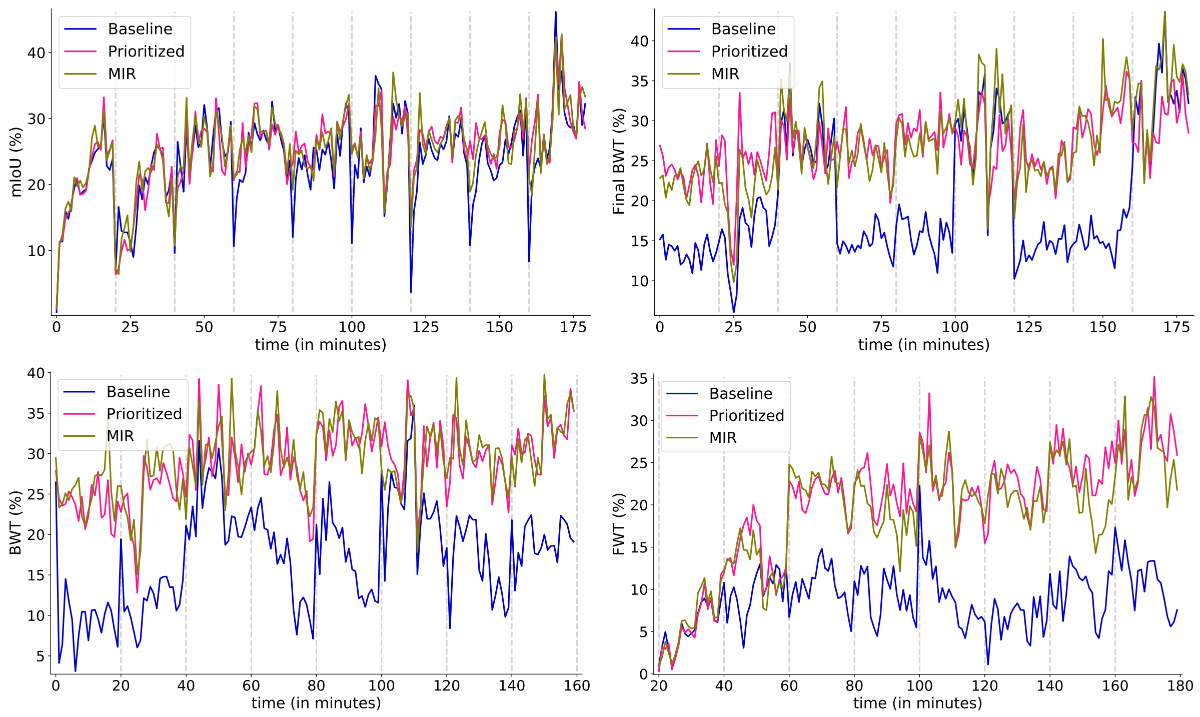


Figure 4.13. **Evolution of the performance over time.** Prioritized and MIR are compared with the baseline. Both methods, again, clearly outperform the baseline on every aspect. (Top-left) mIoU: All the methods have similar performance within a domain. the difference occurs at each domain shift; reguarding MIR and Prioritized, they both do not suffer from a huge drop of performance. (Bottom-left) BWT: MIR and Prioritized are both able to retain information from the past. MIR and Prioritized almost shows the same performance. (Top-right) Final-BWT: The baseline only remembers the latest domain it has trained on (bad performances on the two others domains), while Prioritized and MIR were able to retain information from the past. Again, MIR and Prioritized almost show the same performance. (Bottom-right) FWT: When evaluated on future domains, MIR and Prioritized remain performant on new samples from already seen domains, showing good generalization. Prioritized slightly outperform MIR on the FWT.

Concerning MIR+MAS and MIR+LWF, their mIoU is low compared with the baseline. However, they still perform better than the baseline in terms of FWT and BWT (also Final BWT for MIR+LWF). In the other hand, MIR+MAS and MIR+LWF perform worse than MIR, which may indicates that the regularizer prevents the network from learning a lot from new domains. Constraining the model by a previous version of the model does not seem to be a good idea in the online distillation framework. Fig 4.14 shows the performance of MIR+LWF compared to the baseline. It shows that MIR+LWF (or MIR+MAS) is able outperform the baseline in terms of BWT, FWT and Final-BWT, but they get outperformed by the baseline inside the current domain.

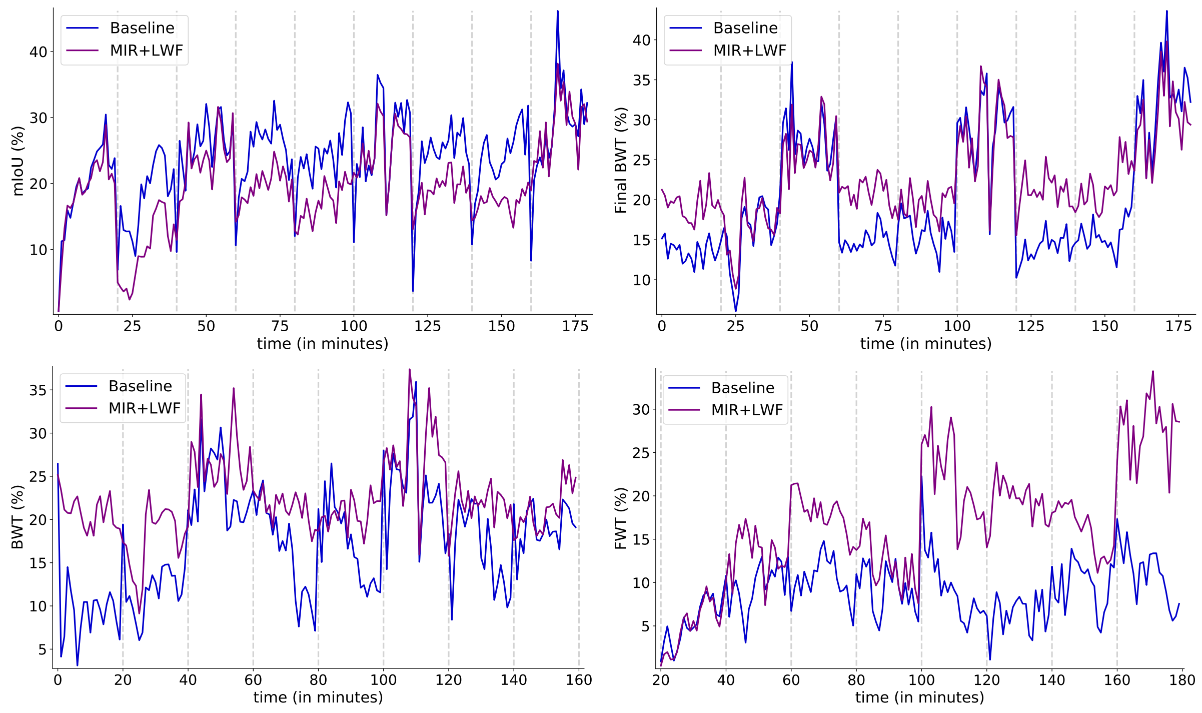


Figure 4.14. **Evolution of performance over time.** Comparison of MIR+LWF with the baseline. (Top-left) mIoU: During a domain shift, MIR+LWF doesn't suffer from catastrophic forgetting. However, it does not manage to outperform the baseline inside the same domain. (Bottom-left) BWT: When evaluating on already-seen samples, MIR+LWF does not seem to suffer from forgetting, but its learning capacity is limited. (Top-right) Final-BWT: MIR+LWF shows good performance on the last seen domain it has trained on. However, its performance are weak on the other domains, but it is still better than the baseline. (Bottom-right) FWT: When evaluated on unseen samples of already seen domains, MIR+LWF shows greater capacity to generalize than the baseline.

Finally, Figure 4.15 illustrates the performance of one of the best method: MIR+ACE. In the mIoU plot, from minute 0 to minute 59, MIR+ACE behaves the same way than the baseline; this is expected since it discovers the three domain. The first difference occurs at minute 60, where the student perceives frames from the first domain. The baseline suffers from a drop of its performance while MIR+ACE keeps showing good performance. For the Final BWT, the final model using with MIR+ACE shows good performance on all domains, meaning that it has been able to retain knowledge from the past. This observation is reinforced by the BWT and the FWT, showing that MIR+ACE mitigates forgetting and generalizes over unseen samples from already seen domains.

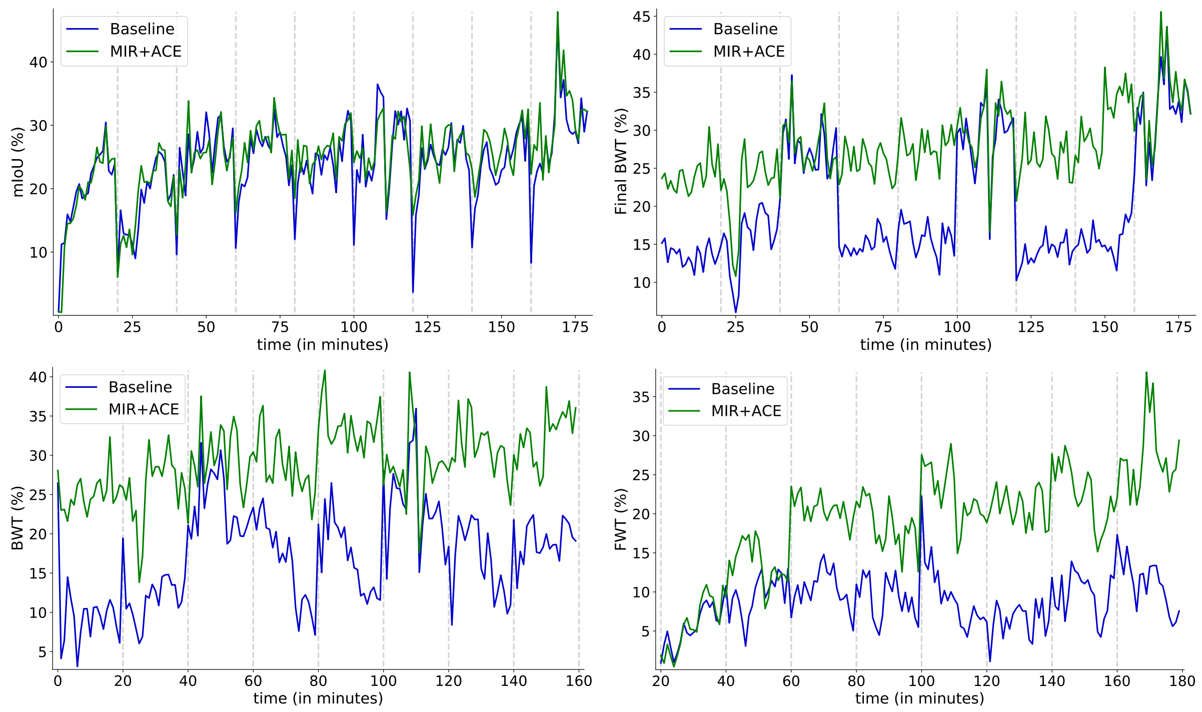


Figure 4.15. **Evolution of performance over time.** MIR-ACE is compared with the baseline. Comments are similar than Prioritized and MIR (4.15). This is one of the methods achieving the best performance.

4.3.2 Qualitative results

A qualitative demonstration of the impact of some continual learning methods on mitigating catastrophic forgetting is presented in this study. As for the first experiment, the accuracy of segmentation masks is examined by following all the possible transitions: from the forest environment to the downtown environment, from the downtown environment to the highway environment and from the highway environment to the forest environment. (In all cases where the student has previously experienced all domains exactly one time).

In figure 4.16, a visual comparison is shown between the segmentation masks produced by the baseline method, Prioritized and MIR+ACE, in relation to the ground-truth mask. Again, the baseline produces poor segmentation masks, even worse than for the first experiment. Indeed, this time, the baseline produce segmentation masks that are not even close to the ground truth, it looks more like a can of paint thrown on a board. This baseline is not suitable for real world application such as autonomous driving.

Reguarding Prioritized, it manages to produce segmentation masks where the regions are well segmented, but it fails at capturing objects (pedestrian, cars, ...) precisely. MIR+ACE manages to produce decent segmentation masks, but fails at capturing minor details (poles, ...).

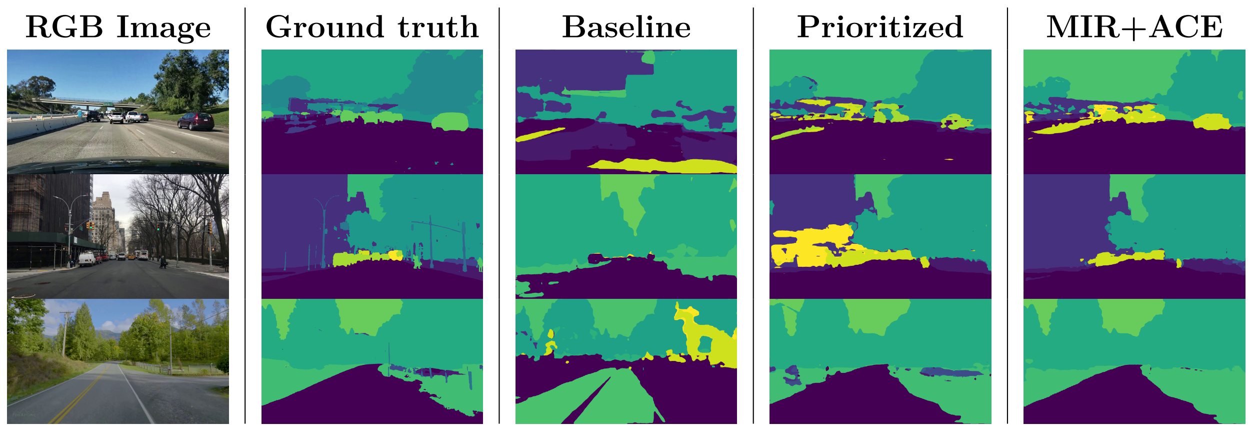


Figure 4.16. **Qualitative results.** Segmentation masks from several methods at the time of a domain shift. Top row shows a frame taken right after the third domain shift (shift from the forest environment to the highway environment). Middle row shows a frame taken right after the fourth domain shift (shift from the highway environment to the downtown environment). Down row shows a frame taken right after the fifth domain shift (shift from the downtown environment to the forest environment). The baseline performs segmentation masks that are very wrong compared to the groundtruth. Prioritized have somme difficulties at representing some classes clearly (cars in this case) and fails at producing decent segmentation masks for them. Prioritized manages to produce good segmentation masks for representing regions (sky, road, trees, ...). MIR+ACE is able to produce good segmentation masks.

4.4 Discussion

4.4.1 Effect of replay-based methods

Experiments have shown that replay-based methods are a good way in order to mitigate forgetting. Furthermore, the way the replay buffer is managed also impacts the performance of the model. When the capacity of the buffer is large enough (wirh respect to the number of domains to be considered), the chosen selection and update functions does not seem to have a huge impact; it has one when either the capacity of the replay-buffer is limited (it can have several causes, such as limited hardware). Or if there are too many domains to remember. If the first case, when the replay-buffer's capacity is too small, frames from past domains are likely to disappear, and thus the student will start forgetting it. In the other hand, if the sequence inside a domain is too long, the same behavior would appear. This behavior has been observed in table 4.1, when the sequence length goes from 20 minutes to 40 minutes, the mIoU NDS, FWT and BWT all decreased. Indeed, as time passes, there is a moment where the number of samples from past domains is too low compared to the samples from the current domain, limiting the network from learning. Furthermore, smarter methods such as Prioritized and MIR show better results than Uniform when the capacity of the replay buffer is low, as seen in table 4.2.

To further demonstrate the utility of using smarter update functions for the replay-buffer in order to mitigate forgetting, one experiment is ran using the Uniform method and the Prioritized method, where the capacity of the buffer is set to $M = 100$, which is the number of samples that are selected for training. Therefore, the selection function selects all samples for training.

Methods	**Parameters**			**Metrics (mean %)**				
	f_S	f_U	$\mathcal{R}$	$mIoU$	$mIoU$ NDS	FWT	BWT	Final BWT
Baseline	All	FIFO	/	23.8	20.2	8.9	17.3	19.8
Replay Buffer	All	Uniform	/	23.7	20.9	14.8	25.0	24.1
	All	Prioritized	/	**23.8**	**21.8**	**15.7**	**25.2**	**25.4**

Table 4.3. **Quantitative results**. The baseline is compared with the Uniform method and the Prioritized method when $M = N = 100$ on the video $\mathcal{V}_2$. In terms of mIoU, all methods perform the same. First difference is for the mIoU NDS, where the Prioritized method decreases less from a drop of performance when a domain shift occurs. In this case, the Prioritized method outperforms all other methods.

Table 4.3 shows the results of these methods on the video $\mathbf{V}_2$, which involves three domains with sequences of 20 minutes long. Overall, the Prioritized method outperforms the baseline on all metrics except for the mIoU. At the time of a domain shift, the Prioritized method shows more robustness compared to the Uniform method. Indeed, due to the limited capacity of the replay-buffer ($M = 100$), samples from past domains are likely to be replaced by new incoming samples from the current domain. In these experiments, the next minute of video the student is trained on is added in the replay-buffer after each epoch. The probability of a sample to be replaced grows quickly with respect to the number of epochs. In the other hand, the Prioritized method takes into account the importance of a frame by computing a score, which makes non-uniform probabilities. Therefore, if the student shows a decrease of its performance on past domains, the probability of replacing samples from past domains by a new sample from the current domain will decreases. Figure 4.17 shows the performance of the methods over time.

One could think that MIR could be an efficient update function as it uses scores to deterministically select samples. Practically, it is not convenient as the considered domains may contain out-of-distribution samples. Therefore, MIR, as an update function, would keep all these samples in the replay-buffer and thus train on out-of-distribution samples, which is something that must be avoided. Using scores to represent probabilities is more convenient to address these issues.

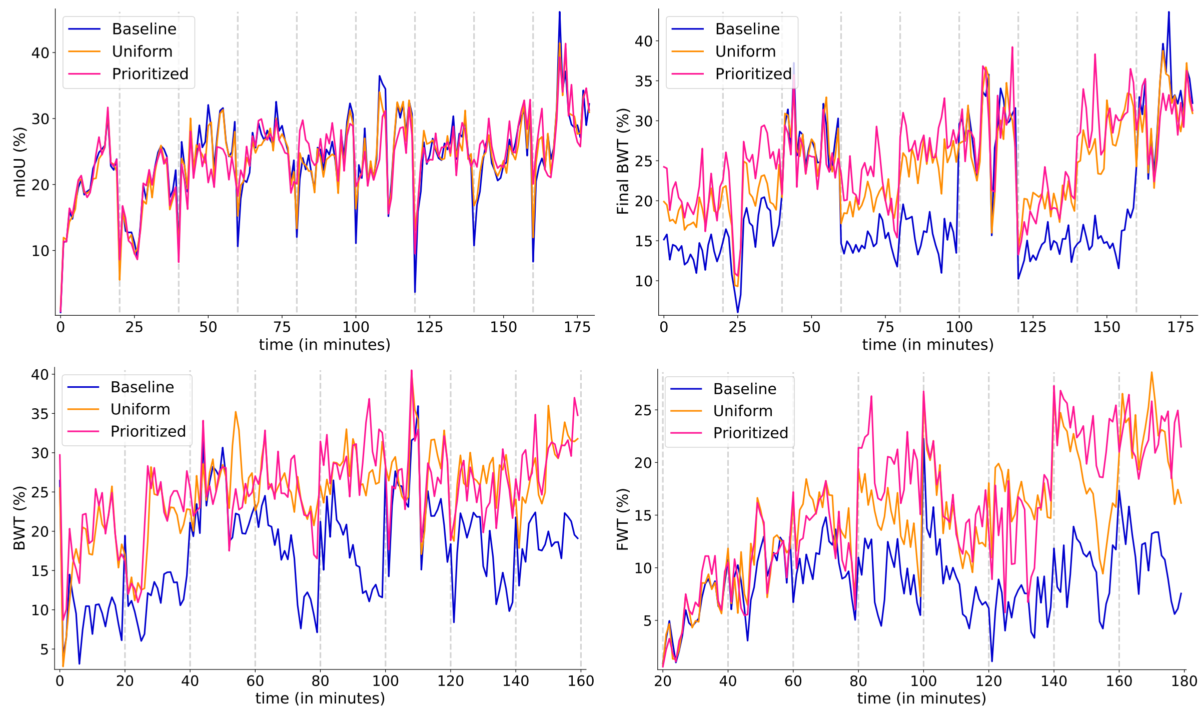


Figure 4.17. **Evolution of performance over time.** The baseline is compared with the Uniform method and the Prioritized method. (Top-left) mIoU: Uniform and Prioritized almost show equal performance than the baseline, except that their suffer less from a drop of their performance at the time of a domain shift. (Bottom-left) BWT: When evaluated on past samples from past domains, Uniform and Prioritized show the same performance. (Top-right) Final-BWT: Prioritized shows better capacity at retaining information from past domains than Uniform. (Bottom-right) FWT: Prioritized shows better generalization than Uniform.

4.4.2 Effect of regularisation-based methods

Experiments have shown that using regularizer methods are not sufficient to mitigate catastrophic forgetting. Indeed, all memoryless methods underperform against the baseline. When regularization-based methods is used with replay-based methods, it can slightly increase the performance.

Nonetheless, regularization-based methods such as MAS and LWF are able to retain knowledge in order to reduce catastrophic forgetting. Indeed, Figure 4.18 compares FIFO-Memoryless (The baseline without the replay-buffer) and MAS-Memoryless. Even though MAS-Memoryless underperforms against FIFO-Memoryless, MAS-Memoryless does not suffer from catastrophic forgetting. This is reinforced by the fact that MAS-Memoryless outperforms FIFO-Memoryless on the BWT and the FWT. Indeed, MAS-Memoryless is able to retain knowledge from past domains but at the price of decreasing its performance on the current domain.

Finally, regularization-based methods may show unsatisfying performance because, initially, they were meant to be used in an offline setup, where task boundaries are known. A better fine-tuning of their hyper-parameters (method-based hyper-parameters, task-boundary simulated hyper-parameters), may increase the performance.

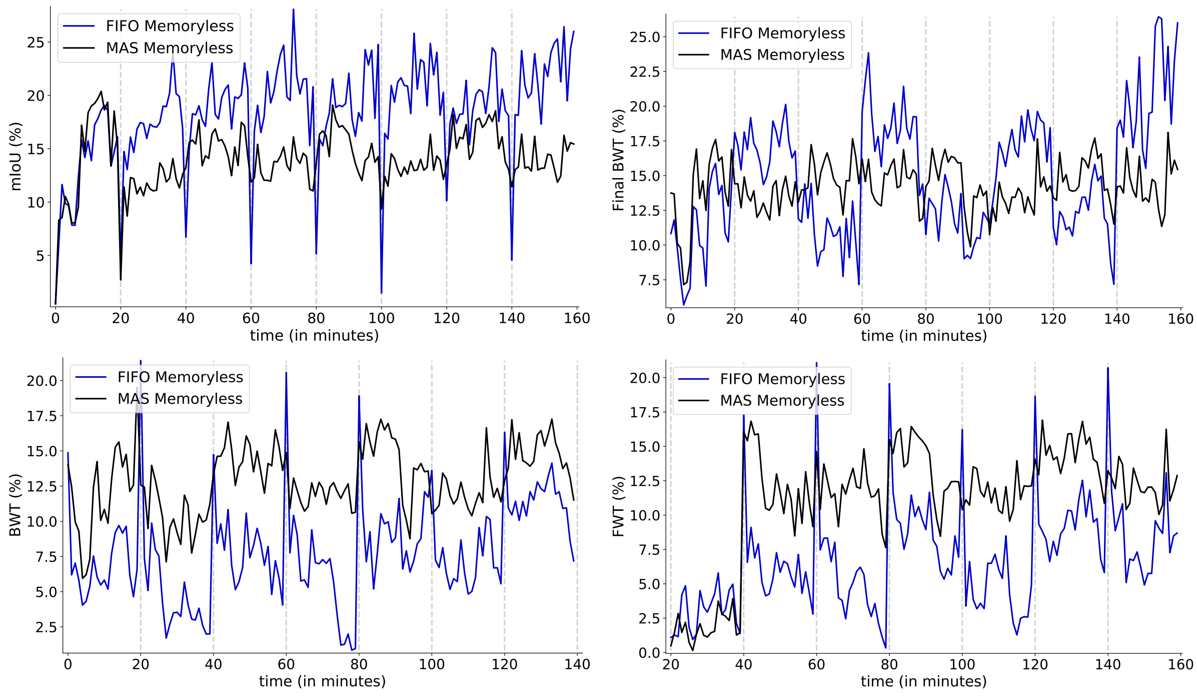


Figure 4.18. **Evolution of performance over time.** FIFO-Memoryless is compared with MAS-Memoryless on the video sequence $\mathcal{V}_1$, where the length of a sequence is 20 minutes. (Top-left) mIoU: FIFO-Memoryless outperforms MAS-Memoryless on the current domain. When a domain shift occurs, MAS-Memoryless does not suffer from a huge drop of its performance unlike FIFO-Memoryless. (Bottom-left) BWT: Memoryless-MAS is able to retain knowledge from frames it has trained on. FIFO-Memoryless quickly forgets the knowledge acquired. (Top-right) Finalt-BWT: FIFO-Memoryless only remembers the last domain it has trained on. MAS-Memoryless shows the same performance on all domains; it underperforms against FIFO-Memoryless on the latest domain they have trained on, but it slightly outperforms it on the other domain. (Bottom-right) FWT: MAS-Memoryless shows more generalization capacity than FIFO-Memoryless

Chapter 5

Conclusion

5.1 Conclusion

In conclusion, the emergence of online distillation has opened up new possibilities for adapting deep neural networks in real-time. Nonetheless, the challenge of catastrophic forgetting during domain shifts has posed a significant obstacle in implementing this technique. This study presents an innovative solution to address this challenge by integrating continual learning methods, enhancing the suitability of online distillation for practical applications like autonomous driving. Through our experiments, we assessed several cutting-edge continual learning approaches and demonstrated their effectiveness in mitigating catastrophic forgetting. Furthermore, we conducted a thorough analysis of our proposed solution, specifically in the context of cyclic domain shifts. The outcomes emphasize that our approach enhances the resilience and accuracy of online distillation, making it a promising technique for real-world applications. This research represents a significant advancement in the field of online distillation and continual learning, offering substantial potential to make a meaningful impact in diverse fields such as autonomous driving.

5.2 To go further

Several doors are opened to improve our online distillation framework. First, other replay-based and regularization-based methods could be used. There exists others more recents regularization- and replay-based methods ([45], [46]) where the problem of catastrophic forgetting is seen from another perspective.

Furthermore, mitigating catastrophic forgetting could also be directly done inside the network itself. One of them are Memory-Augmented Neural Networks (MANNs) [47] where the network is equipped with an external memory to access and retrieve information (refered as Neural Turing Machines [48]). The authors state that their architecture would potentially be suitable to alleviate catastrophic forgetting. However, the core of their study was to achieve one-shot learning.

Bibliography

[1] Joachim Houyon et al. “Online Distillation with Continual Learning for Cyclic Domain Shifts”. In: *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Workshop on Continual Learning in Computer Vision*. Vancouver, Canada: IEEE, June 2023. URL: https://arxiv.org/abs/2304.01239 (pages 1, 6, 7, 14, 34).

[2] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Las Vegas, NV, USA: Inst. Electr. Electron. Eng. (IEEE), June 2016, pp. 770–778. DOI: 10.1109/cvpr.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90 (pages 5, 39).

[3] Ming Liang and Xiaolin Hu. “Recurrent convolutional neural network for object recognition”. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Boston, MA, USA: Inst. Electr. Electron. Eng. (IEEE), June 2015, pp. 3367–3375. DOI: 10.1109/cvpr.2015.7298958. URL: https://doi.org/10.1109/CVPR.2015.7298958 (page 5).

[4] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003. URL: https://doi.org/10.1016/j.neunet.2014.09.003 (page 5).

[5] Dan Hendrycks and Thomas Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations”. In: *Int. Conf. Learn. Represent. (ICLR)*. 2019 (page 5).

[6] Oguzhan Fatih Kar et al. “3D Common Corruptions and Data Augmentation”. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. New Orleans, LA, USA: Inst. Electr. Electron. Eng. (IEEE), June 2022, pp. 18941–18952. DOI: 10.1109/cvpr52688.2022.01839. URL: https://doi.org/10.1109/CVPR52688.2022.01839 (page 5).

[7] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proc. National Acad. Sci. (PNAS)* 114.13 (Mar. 2017), pp. 3521–3526. DOI: 10.1073/pnas.1611835114. URL: https://doi.org/10.1073/pnas.1611835114 (pages 5, 11).

[8] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. “ACDC: The Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding”. In: *IEEE Int. Conf. Comput. Vis. (ICCV)*. Montreal, QC, Canada: Inst. Electr. Electron. Eng. (IEEE), Oct. 2021, pp. 10745–10755. DOI: 10.1109/iccv48922.2021.01059. URL: https://doi.org/10.1109/ICCV48922.2021.01059 (pages 5, 8).

[9] Zhizhong Li and Derek Hoiem. “Learning without Forgetting”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.12 (Dec. 2018), pp. 2935–2947. DOI: 10.1109/tpami.2017.2773081. URL: https://doi.org/10.1109/TPAMI.2017.2773081 (pages 5, 11, 19, 21).

[10] Arslan Chaudhry et al. “Continual Learning with Tiny Episodic Memories”. In: *Int. Conf. Mach. Learn. (ICML)*. 2019 (pages 5, 11).

[11] Zhipeng Cai, Ozan Sener, and Vladlen Koltun. “Online Continual Learning with Natural Distribution Shifts: An Empirical Study with Visual Data”. In: *IEEE Int. Conf. Comput. Vis. (ICCV)*. Montréal, Can.: Inst. Electr. Electron. Eng. (IEEE), Oct. 2021, pp. 8261–8270. DOI: 10.1109/iccv48922.2021.00817. URL: https://doi.org/10.1109/ICCV48922.2021.00817 (page 5).

[12] Baochen Sun and Kate Saenko. “Deep CORAL: Correlation Alignment for Deep Domain Adaptation”. In: *Eur. Conf. Comput. Vis. (ECCV)*. Vol. 9915. Lect. Notes Comput. Sci. Springer Int. Publ., 2016, pp. 443–450. DOI: 10.1007/978-3-319-49409-8_35. URL: https://doi.org/10.1007/978-3-319-49409-8_35 (page 5).

[13] Prachi Garg et al. “Multi-Domain Incremental Learning for Semantic Segmentation”. In: Jan. 2022, pp. 2080–2090. DOI: 10.1109/wacv51458.2022.00214. URL: https://doi.org/10.1109/WACV51458.2022.00214 (pages 5, 11).

[14] Dequan Wang et al. “Tent: Fully test-time adaptation by entropy minimization”. In: *CoRR* abs/2006.10726 (2020). arXiv: 2006.10726 (page 5).

[15] Anthony Cioppa et al. “ARTHuS: Adaptive Real-Time Human Segmentation in Sports Through Online Distillation”. In: *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports*. Long Beach, CA, USA: Inst. Electr. Electron. Eng. (IEEE), June 2019, pp. 2505–2514. DOI: 10.1109/cvprw.2019.00306. URL: https://doi.org/10.1109/CVPRW.2019.00306 (pages 6, 9, 10, 29, 38, 40, 41, 46).

[16] Abolfazl Farahani et al. “A Brief Review of Domain Adaptation”. In: *Trans. Comput. Sci. Comput. Intell.* (2021), pp. 877–894. DOI: 10.1007/978-3-030-71704-9_65. URL: https://doi.org/10.1007/978-3-030-71704-9_65 (page 8).

[17] Hyesu Lim et al. “TTN: A Domain-Shift Aware Batch Normalization in Test-Time Adaptation”. In: *Int. Conf. Learn. Represent. (ICLR)*. 2023, pp. 1–19. URL: https://openreview.net/pdf?id=EQfeudmWLQ (page 8).

[18] Qin Wang et al. “Continual Test-Time Domain Adaptation”. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. New Orleans, LA, USA: Inst. Electr. Electron. Eng. (IEEE), June 2022, pp. 7191–7201. DOI: 10.1109/cvpr52688.2022.00706. URL: https://doi.org/10.1109/CVPR52688.2022.00706 (page 8).

[19] Fatemeh Azimi et al. “Self-supervised Test-time Adaptation on Video Data”. In: *IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*. Waikoloa, HI, USA: Inst. Electr. Electron. Eng. (IEEE), Jan. 2022, pp. 2603–2612. DOI: 10.1109/wacv51458.2022.00266. URL: https://doi.org/10.1109/WACV51458.2022.00266 (page 8).

[20] Tao Sun et al. “SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation”. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. New Orleans, LA, USA: Inst. Electr. Electron. Eng. (IEEE), June 2022, pp. 21339–21350. DOI: 10.1109/cvpr52688.2022.02068. URL: https://doi.org/10.1109/CVPR52688.2022.02068 (page 8).

[21] Fisher Yu et al. “BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning”. In: *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR).* Seattle, WA, USA: Inst. Electr. Electron. Eng. (IEEE), June 2020, pp. 2633–2642. DOI: 10.1109/cvpr42600.2020.00271. URL: https://doi.org/10.1109/CVPR42600.2020.00271 (pages 8, 33).

[22] Jinlong Li et al. “Domain Adaptive Object Detection for Autonomous Driving under Foggy Weather”. In: *IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV).* Waikoloa, HI, USA: Inst. Electr. Electron. Eng. (IEEE), Jan. 2023, pp. 612–622. DOI: 10.1109/wacv56688.2023.00068. URL: https://doi.org/10.1109/WACV56688.2023.00068 (page 8).

[23] Sébastien Piérard et al. “Mixture Domain Adaptation to Improve Semantic Segmentation in Real-World Surveillance”. In: *IEEE/CVF Winter Conf. Appl. Comput. Vis. Work. (WACVW).* Waikoloa, HI, USA: Inst. Electr. Electron. Eng. (IEEE), Jan. 2023, pp. 22–31. DOI: 10.1109/wacvw58289.2023.00007. URL: https://doi.org/10.1109/WACVW58289.2023.00007 (page 8).

[24] Theodoros Panagiotakopoulos et al. “Online Domain Adaptation for Semantic Segmentation in Ever-Changing Conditions”. In: *Eur. Conf. Comput. Vis. (ECCV).* Vol. 13694. Lect. Notes Comput. Sci. Springer Nat. Switz., 2022, pp. 128–146. DOI: 10.1007/978-3-031-19830-4_8. URL: https://doi.org/10.1007/978-3-031-19830-4_8 (page 8).

[25] Yawei Luo et al. “Taking a Closer Look at Domain Shift: Category-Level Adversaries for Semantics Consistent Domain Adaptation”. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR).* Long Beach, CA, USA: Inst. Electr. Electron. Eng. (IEEE), June 2019, pp. 2502–2511. DOI: 10.1109/cvpr.2019.00261. URL: https://doi.org/10.1109/CVPR.2019.00261 (page 8).

[26] Enze Xie et al. “SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers”. In: vol. 34. 2021, pp. 12077–12090 (pages 9, 36–38).

[27] Hengshuang Zhao et al. “Pyramid Scene Parsing Network”. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR).* Honolulu, HI, USA: Inst. Electr. Electron. Eng. (IEEE), July 2017, pp. 6230–6239. DOI: 10.1109/cvpr.2017.660. URL: https://doi.org/10.1109/CVPR.2017.660 (page 9).

[28] Sixiao Zheng et al. “Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers”. In: June 2021, pp. 6877–6886. DOI: 10.1109/cvpr46437.2021.00681. URL: https://doi.org/10.1109/CVPR46437.2021.00681 (page 9).

[29] Anthony Cioppa, Adrien Deliège, and Marc Van Droogenbroeck. “A bottom-up approach based on semantics for the interpretation of the main camera stream in soccer games”. In: *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports.* Salt Lake City, UT, USA, June 2018, pp. 1846–1855. DOI: 10.1109/CVPRW.2018.00229. URL: http://hdl.handle.net/2268/222427 (pages 9, 38, 39).

[30] Anthony Cioppa et al. “Multimodal and multiview distillation for real-time player detection on a football field”. In: *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports.* Seattle, WA, USA, June 2020, pp. 3846–3855. DOI: 10.1109/CVPRW50498.2020.00448. URL: https://doi.org/10.1109/CVPRW50498.2020.00448 (page 10).

[31] Michael McCloskey and Neal J. Cohen. “Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem”. In: *Psychol. Learn. Motiv.* (1989), pp. 109–165. DOI: 10.1016/s0079-7421(08)60536-8. URL: https://doi.org/10.1016/S0079-7421(08)60536-8 (page 11).

[32] Robert M. French. “Catastrophic forgetting in connectionist networks”. In: *Trends Cogn. Sci.* 3.4 (Apr. 1999), pp. 128–135. DOI: 10.1016/s1364-6613(99)01294-2. URL: https://doi.org/10.1016/S1364-6613(99)01294-2 (page 11).

[33] Rahaf Aljundi et al. “Memory Aware Synapses: Learning What (not) to Forget”. In: *Eur. Conf. Comput. Vis. (ECCV)*. Vol. 11207. Lect. Notes Comput. Sci. Springer Int. Publ., 2018, pp. 144–161. DOI: 10.1007/978-3-030-01219-9_9. URL: https://doi.org/10.1007/978-3-030-01219-9_9 (pages 11, 19, 22, 24).

[34] Arslan Chaudhry et al. “Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence”. In: *Eur. Conf. Comput. Vis. (ECCV)*. Vol. 11215. Lect. Notes Comput. Sci. Springer Int. Publ., 2018, pp. 556–572. DOI: 10.1007/978-3-030-01252-6_33. URL: https://doi.org/10.1007/978-3-030-01252-6_33 (pages 11, 19, 24).

[35] James Smith et al. “Always Be Dreaming: A New Approach for Data-Free Class-Incremental Learning”. In: *IEEE Int. Conf. Comput. Vis. (ICCV)*. Montreal, QC, Canada: Inst. Electr. Electron. Eng. (IEEE), Oct. 2021, pp. 9354–9364. DOI: 10.1109/iccv48922.2021.00924. URL: https://doi.org/10.1109/ICCV48922.2021.00924 (page 11).

[36] Qiankun Gao et al. “R-DFCIL: Relation-Guided Representation Learning for Data-Free Class Incremental Learning”. In: *Eur. Conf. Comput. Vis. (ECCV)*. Vol. 13683. Lect. Notes Comput. Sci. Springer Nat. Switz., 2022, pp. 423–439. DOI: 10.1007/978-3-031-20050-2_25. URL: https://doi.org/10.1007/978-3-031-20050-2_25 (page 11).

[37] David Lopez-Paz and Marc’Aurelio Ranzato. “Gradient episodic memory for continual learning”. In: *Adv. Neural Inf. Process. Syst. (NeurIPS)*. 2017 (page 11).

[38] Rahaf Aljundi et al. “Online continual learning with maximally interfered retrieval”. In: *Adv. Neural Inf. Process. Syst. (NeurIPS)*. 2019 (pages 11, 18).

[39] Motasem Alfarra et al. “SimCS: Simulation for Online Domain-Incremental Continual Segmentation”. In: *CoRR* abs/2211.16234 (2022). DOI: 10.48550/arXiv.2211.16234. arXiv: 2211.16234. URL: https://doi.org/10.48550/arXiv.2211.16234 (page 11).

[40] Tom Schaul et al. “Prioritized Experience Replay”. In: *CoRR* abs/1511.05952 (2015). DOI: 10.48550/arXiv.1511.05952. arXiv: 1511.05952. URL: https://doi.org/10.48550/arXiv.1511.05952 (page 16).

[41] Lucas Caccia et al. “New Insights on Reducing Abrupt Representation Change in Online Continual Learning”. In: *Int. Conf. Learn. Represent. (ICLR)*. 2022 (page 19).

[42] Natalia Dıaz-Rodrıguez et al. “Don’t forget, there is more than forgetting: new metrics for Continual Learning”. In: *Continual learning W., Neural Inf. Process. Syst. (NeurIPS)*. 2018 (page 27).

[43] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. Las Ve-

gas, NV, USA: Inst. Electr. Electron. Eng. (IEEE), June 2016, pp. 3213–3223. DOI: 10.1109/cvpr.2016.350. URL: https://doi.org/10.1109/cvpr.2016.350 (pages 33, 37).

[44] Hengshuang Zhao et al. "Pyramid Scene Parsing Network". In: 2017. arXiv: 1612.01105 [cs.CV] (page 39).

[45] Yaoyao Liu et al. "Mnemonics Training: Multi-Class Incremental Learning Without Forgetting". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.01226. URL: https://doi.org/10.11092Fcvpr42600.2020.01226 (page 55).

[46] Bowen Zhao et al. "Maintaining Discrimination and Fairness in Class Incremental Learning". 2019. arXiv: 1911.07053 [cs.CV] (page 55).

[47] Adam Santoro et al. "One-shot Learning with Memory-Augmented Neural Networks". 2016. arXiv: 1605.06065 [cs.LG] (page 55).

[48] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural Turing Machines". 2014. arXiv: 1410.5401 [cs.NE] (page 55).

Appendix A

Online Distillation with Continual Learning for Cyclic Domain Shifts

Online Distillation with Continual Learning for Cyclic Domain Shifts

Joachim Houyon[1,*] Anthony Cioppa[1,2,*] Yasir Ghunaim[2] Motasem Alfarra[2]
Anaïs Halin[1] Maxim Henry[1] Bernard Ghanem[2] Marc Van Droogenbroeck[1]
[1] University of Liège [2] KAUST

Abstract

In recent years, online distillation has emerged as a powerful technique for adapting real-time deep neural networks on the fly using a slow, but accurate teacher model. However, a major challenge in online distillation is catastrophic forgetting when the domain shifts, which occurs when the student model is updated with data from the new domain and forgets previously learned knowledge. In this paper, we propose a solution to this issue by leveraging the power of continual learning methods to reduce the impact of domain shifts. Specifically, we integrate several state-of-the-art continual learning methods in the context of online distillation and demonstrate their effectiveness in reducing catastrophic forgetting. Furthermore, we provide a detailed analysis of our proposed solution in the case of cyclic domain shifts. Our experimental results demonstrate the efficacy of our approach in improving the robustness and accuracy of online distillation, with potential applications in domains such as video surveillance or autonomous driving. Overall, our work represents an important step forward in the field of online distillation and continual learning, with the potential to significantly impact real-world applications.

1. Introduction

Deep Neural Networks (DNNs) have shown remarkable performance on various computer vision tasks thanks in part to the assumption that the training and testing data are identically distributed [21, 27, 37]. However, DNNs' performance degrade significantly when tested on out-of-distribution data, such as testing data that contains domain shifts relative to the training data [22, 23]. Even worse, DNNs tend to forget previously learned distributions when learning continually on a stream of tasks [24]. This performance loss is a major concern because domain shifts are likely to occur in real-world deployments due to changes in brightness between day and night, weather conditions across seasons, and sensor perturbations [35]. Therefore, it is essential to develop algorithms that can enable DNNs to adapt to such domain shifts and maintain high performance in real-world settings.

Continual learning aims at building machine learning models that can learn from a continuous stream of data without forgetting previously learned knowledge [9, 20, 26]. We investigate a practical scenario of online continual learning [7]. Specifically, we consider cyclic domain shifts where a stream of data consistently alternates in revealing new *unlabeled* data from one of two distributions for a period of time. For instance, consider an autonomous driving system that frequently travels between cities and countrysides, where the distribution of instances varies between the two scenes. Such domain variation can cause the online learner to fail in adapting to this distribution shift, raising concerns about the real-world deployment of such systems. While online continual learning has been studied in

(*) Equal contributions
Contacts: joachim.jouyon@student.uliege.be, anthony.cioppa@uliege.be.
Data/code available at github.com/Houyon/online-distillation-cl.

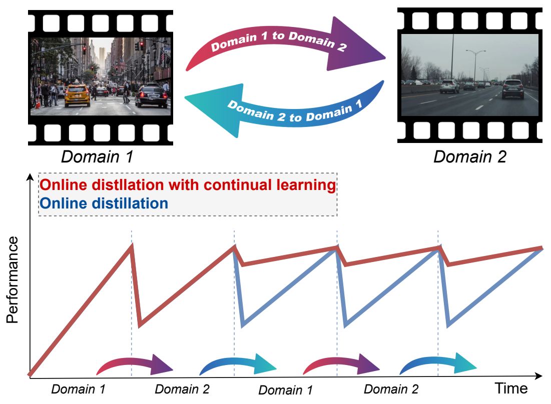


Figure 1. **Online distillation with continual learning.** When cyclic domain shifts occur in long videos, the online distillation framework proposed by Cioppa *et al.* [10] forgets the previously acquired knowledge as it fine-tunes on the current domain. In this work, we study the inclusion of state-of-the-art continual learning methods inside the online distillation framework to mitigate this catastrophic forgetting around the domain shifts.

several contexts, such as domain incremental learning [18], unsupervised domain adaptation [39], and test-time adaptation [41], these works typically analyze the more general, and potentially less realistic, setup where domain variations are unconditional. Our focus on cyclic domain shifts enables us to explore a pragmatic setting and develop novel algorithms that can better adapt to these changes.

In this work, we propose a novel approach to address the challenge of adapting to cyclic domain shifts in the context of online domain incremental learning. Specifically, we employ a previously published real-time online distillation technique [10] to learn from the unlabeled cyclic stream of data. Online distillation asynchronously updates a student-teacher based approach on the received data, which enables the model to continually learn from new data. However, we found that the cyclic domain shift can cause the student to forget the previously learned domain, leading to a significant loss in performance. To mitigate this undesirable effect, we combine online distillation with state-of-the-art continual learning as shown in Figure 1, leveraging both regularization- and replay-based approaches from the continual learning literature. Our proposed approach enables the student to effectively adapt to cyclic domain shifts and maintain high performance over time, making it suitable for real-world deployment.

Contributions. We summarize our contributions in two points: **(i)** We define the cyclic online continual learning problem setup and propose corresponding evaluation metrics. **(ii)** We combine online distillation with both regularization- and replay-based continual learning approaches to better learn on cyclic domains. We conduct experiments on the proposed stream where we show that our approach mitigates the forgetting of the original online distillation framework.

2. Related Work

Domain shifts. A domain shift is a change in the statistical distribution of data between different domains [15]. This phenomenon is commonly observed at test time in open-world scenarios [4, 19, 28, 42]. In autonomous driving, domain shift can be caused by many diverse factors [40], such as different environments (*e.g.*, rural or urban roads), lighting conditions (*e.g.*, day or night), weather conditions (*e.g.*, sunny or snowy) [35], traffic conditions or even differences in the appearance of roads or traffic signs across different countries [44]. However, it is crucial for autonomous vehicles to have algorithms that are robust to these dynamic domain shifts in order to constantly be able to perceive and understand their surrounding environment to avoid obstacles. Domain adaptation is an active area of research that aims at addressing the domain shift problem, especially in open-world applications such as autonomous vehicles [25, 30, 32, 33, 40], where data is collected in a highly dynamic environment. In this work, we study the particular case of cyclic domain shifts in the field of autonomous driving, where the domains can be represented as a succession of *highway* and *downtown* driving conditions.

Online distillation. In the field of deep neural networks, there is a trade-off between speed, performance, and generalizability across multiple domains. While the best-performing models often exhibit high performance across diverse domains, they tend to be memory-greedy for embedded systems or too slow for use in real-time applications [43, 46, 47]. In contrast, lightweight and fast networks show good performance on smaller domains but lack generalizability [12]. To address this issue, Cioppa *et al.* [10] proposed an online distillation approach for videos, that enables the online training of a lightweight student network using a slower, larger teacher model. At test time, the teacher provides pseudo ground truths to the student, allowing it to specialize in the specific domain being analyzed. The student model therefore adapts to changing video conditions, even matching the performance of the slower teacher. This online distillation approach may be used for different tasks such as semantic segmentation [10] or multi-modal object detection [11]. However, this technique experiences a temporary loss of performance during domain shifts. In this paper, we investigate several continual techniques to mitigate the effects of catastrophic forgetting in online distillation, particularly in cases of cyclic domain shifts. We combine online distillation with both regularization- and replay-based approaches for a better continual learning scheme.

Continual learning. Continual Learning (CL) aims at learning from data arriving as a stream with changing distribution [16, 31]. However, this learning paradigm face the catastrophic forgetting challenge, that is, previously learned knowledge is forgotten when adapting to the newly arriving data samples [9, 24]. One approach of mitigating the forgetting effect is regularizing the training process through constraining the changes of important network parameters [2, 8, 24] or performing knowledge distillation [17, 26, 38]. Alternatively, replay-based methods rehearse previously seen examples by storing a subset of the observed data in a replay buffer [3, 9, 29, 34]. While both approaches were originally proposed for class-incremental setup and classification task, they were recently extended to the more realistic domain incremental setup and the more challenging semantic segmentation task [1, 18]. Nevertheless, prior art assumes fully supervised setups where the stream reveals labeled data for the student learner. To that end, we analyze the domain incremental setup for semantic segmentation under an unsupervised setup.

3. Methodology

In this section, we first describe online distillation in a mathematical framework suited for continual learning. Next we detail the regularization-based and replay-based continual learning methods that we integrate into the online distillation framework. Finally, we explain how to evaluate and benchmark online continual leaning methods under our cyclic stream.

3.1. Online distillation framework

The online distillation framework proposed by Cioppa *et al.* [10] allows a real-time network to adapt to domain shifts at test time. Formally, given a long untrimmed video $\mathcal{V}$ composed of a stream of frames x_i produced at a rate $r_{\mathcal{V}}$ and a task $\mathcal{T}$ (*e.g.*, object detection, semantic segmentation, *etc.*), the objective is to produce a stream of predictions $\hat{y}_i$ for each frame x_i in real time (*i.e.*, at a rate $r_{\mathcal{V}}$). To do so, the authors leverage a student-teacher architecture with a fast and slow route. In the fast route (inference), a student network $\mathbf{S}$ computes $\hat{y}_i = \mathbf{S}(x_i)$ at the rate $r_{\mathcal{V}}$. In parallel in the slow route (training), a slower but high-performance frozen teacher network $\mathbf{T}$ produces pseudo ground-truths $\tilde{y}_{i'} = \mathbf{T}(x_{i'})$ at an asynchronous slower rate $r_{\mathbf{T}}$ on a subset of $\mathcal{V}$. Each new pair $(x_{i'}, \tilde{y}_{i'})$ is then stored through an update function f_U into an online dataset $\mathcal{D}$ of size N that is used to train a copy $\mathbf{S}_c$ of the student network. In the original framework, f_U is chosen as a First In First Out (FIFO) algorithm. Iteratively, $\mathbf{S}_c$ is trained on selected samples extracted from $\mathcal{D}$ by a function f_S, by minimizing the loss:

$$\mathcal{L} = \sum_{n=1}^{N} L(\mathbf{S}_c(x_n), \tilde{y}_n) ,$$

where L is a distance function suited to learn task $\mathcal{T}$. In the original framework, f_S selects all pairs in $\mathcal{D}$ one time. The parameters of $\mathbf{S}$ are updated by copying the parameters θ of $\mathbf{S}_c$ at the rate $r_{\mathbf{S}_c}$, corresponding to the inverse of the training time of $\mathbf{S}_c$ on one epoch of $\mathcal{D}$. The complete pipeline may be found in Figure 2.

Thanks to this framework, $\mathbf{S}$ becomes specialized to the last minutes of the particular video it is analyzing. This allows it to adapt to slowly changing domains in $\mathcal{V}$ as long as $\mathbf{T}$ is able to produce reliable predictions. However, this continual fine-tuning makes it forget previously acquired knowledge over time. For instance, when sudden shifts in domain occurs, $\mathbf{S}$ needs several updates to recover good performance even if the same domain already appeared in the video. In the following, we propose to incorporate Continual Learning (CL) techniques in the existing online distillation framework to minimize the catastrophic forgetting of previously acquired knowledge in the case of cyclic domain shifts. In particular, we benchmark several replay-based methods (CL_{Rep}) that act on $\mathcal{D}$ and regularization-based methods (CL_{Reg}) that act on $\mathcal{L}$ as shown in Figure 2.

3.2. Replay-based methods

This set of methods leverage a replay buffer (*i.e.* a collection of data and corresponding ground-truth labels) of finite size that is accessed by the selection function f_S and updated with new data by an update function f_U at each training epoch. The online distillation framework presented above can be formulated as a replay-based method, where the replay buffer corresponds to $\mathcal{D}$, the labels are the pseudo ground-truth predictions $\tilde{y}_n$, f_S selects all data of the replay buffer to be used during the training epoch, and f_U determines the policy to update samples in the replay buffer. In the original online distillation framework, the size of the replay buffer is also the number of samples, N, passed to the model at each training step. We extend the replay buffer to include $M \geq N$ samples where we sample N samples without replacement from the buffer at each training step. We augment the selected samples with the new incoming data from the stream.

We consider several strategies to modify f_U and f_S to reduce the catastrophic forgetting: FIFO, Uniform, Prioritized, and MIR.

FIFO: f_U stores the most recent samples in the replay buffer while removing oldest ones. This is equivalent to the original framework's update strategy that is used as a baseline for comparison with other methods.

Uniform: f_U stores incoming data at randomly selected replay buffer indices. This strategy leads to an expected remaining lifespan of data to decay exponentially [5], which could avoid forgetting. As for memory selection f_S, it performs a random selection from memory for constructing a training batch.

Prioritized: Adapting the work of Schaul *et al.* [36] on reinforcement learning, we set f_U to assign an importance score $\mathcal{I}$ for each sample in the replay buffer following:

$$\mathcal{I}_n = L(\mathbf{S}(x_n), \mathbf{T}(x_n)) .$$

The importance score is then used as a probability of determining which samples to remove from the replay buffer following:

$$p_n = \frac{\mathcal{I}_n^{-1}}{\sum_{n'=1}^{M} \mathcal{I}_{n'}^{-1}} .$$

To perform the memory selection f_S operation, prioritized follows the same strategy described above for the update function f_U.

MIR [3]: is a selection function f_S that selects a subset of the replay buffer samples that are maximally interfered by the incoming data in a stream. In other words, it constructs a set of training samples from memory that are negatively affected the most by the next parameter update.

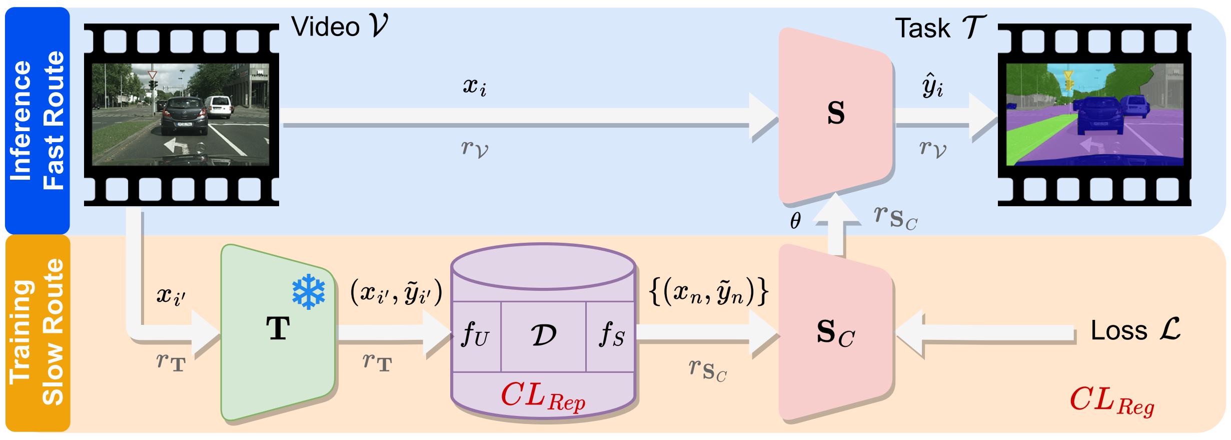


Figure 2. **Online distillation**. The framework is composed of a fast and a slow route. In the fast route (inference), the video stream $\mathcal{V}$ is processed by a student network $\mathbf{S}$ on a task $\mathcal{T}$ (*e.g.*, semantic segmentation for autonomous driving) and produces predictions $\hat{y}_i$ for each frame of the video x_i at the original video rate $r_\mathcal{V}$ (*i.e.*, in real time). In parallel in the slow route (training), a frozen teacher $\mathbf{T}$ produces pseudo ground-truths $\tilde{y}_{i'}$ from a subset of frames $x_{i'}$ at a slower rate $r_\mathbf{T}$. The pair $(x_{i'}, \tilde{y}_{i'})$ are then stored in an online dataset (or replay buffer) $\mathcal{D}$ through an update function f_U. $\mathcal{D}$ is sampled through a selection function f_S and the selected pairs $(x_n, \tilde{y}_n)$ are used to train a copy of the student network $\mathbf{S}_c$ for one epoch using a loss $\mathcal{L}$. The parameters θ of $\mathbf{S}_c$ are then transferred to $\mathbf{S}$ at a rate $r_{\mathbf{S}_c}$ (corresponding to the inverse of the training time of $\mathbf{S}_c$ on one epoch) so that $\mathbf{S}$ improves on the latest domain of $\mathcal{V}$. One of the contribution of our paper consists in including replay-based Continual Learning (CL) methods, CL_{Rep}, inside $\mathcal{D}$ and regularization-based methods, CL_{Reg}, on $\mathcal{L}$.

3.3. Regularization-based methods

Regularization-based methods mitigate forgetting by adding a regularization term to the training loss function $\mathcal{L}$. Generally, this can be formulated as:

$$\mathcal{L} = \sum_{n=1}^{N} L(\mathbf{S}_c(x_n), \tilde{y}_n) + \mathcal{R} ,$$

where $\mathcal{R}$ is a method-specific regularization term. In this paper, we consider four different regularization-based continual learning methods: ER-ACE [6], LwF [26], MAS [2], and RWalk [8]. We summarize these methods hereafter.

ER-ACE [6] aims at reducing the changes in the learned representation when training on samples from a new class. It does so by applying an asymmetric parameter update on the incoming data and the previously seen data that are sampled from a replay buffer. Specifically, ER-ACE restricts the loss computation on classes presented in the incoming data while ignoring remaining classes. We note that ER-ACE only works on incoming data while keeping the original loss on the data sampled from replay buffer.

The following methods were originally proposed for settings with clear task boundaries. We adopt them to work on online streams without task boundaries by using two properties: **(i)** warmup and **(ii)** update frequency. The warmup defines a time period for the network to be initialized during the warmup phase, we set $\mathcal{R} = 0$. The update frequency simulates an artificial task boundary after every k steps, where k is a fixed hyperparameter for all methods.

LwF [26] uses knowledge distillation to encourage the current network's output to resemble that of a network trained on data from previous time steps. In our setup, LwF keeps a previous version of our student network $\mathbf{S}_c$ to guide the future parameter updates of this network. Maintaining a previous network that is potentially more tailored to previous domains could help in preserving learned knowledge.

MAS [2] assigns an importance weight for each network parameter by approximating the sensitivity of the network output to a parameter change. When training on new distributions, it penalizes large changes to important parameters and, thus, preserves previously learned knowledge.

RWalk [8] is a generalized formulation that combines a modified version of the two popular importance-based methods: EWC [24] and PI [45]. RWalk computes importance scores for network parameters, similar to MAS, and regularizes over the network parameters.

3.4. Evaluation methodology

To evaluate the adaption to new domains and the forgetting of past domains, we propose several evaluation metrics. Following the work of Cioppa *et al.* [10], the performance of the student network $\mathbf{S}_c$ (equivalent to $\mathbf{S}$) over time is defined as follows: given a task-specific metric $\mathcal{M}$ (*e.g.*, $mIoU$ for semantic segmentation or $accuracy$ for classification), a set of size I of frames $X_i' = \{x_{i'}, ..., x_{i'+I}\}$ and pseudo ground truths $\tilde{Y}_i' = \{\tilde{y}_{i'}, ..., \tilde{y}_{i'+I}\}$, the performance of the student network at time i' is given by:

$$\mathcal{M}(\mathbf{S}_c(X_{i'};\theta_{i'}),\tilde{Y}_{i'})\ ,$$

where $\theta_{i'}$ are the parameters of $\mathbf{S}_c$ at time i', which may be asynchronous with the training of $\mathbf{S}_c$ and update of $\mathbf{S}$ as it operates at a the different rate $r_{\mathbf{S}_c}$.

Backward Transfer (BWT): Motivated by the discrete implementation of backward transfer [14], we propose a modified version for online streams that measures forgetting of the current student network with respect to previous data, which corresponds to the previous domain in our case:

$$\text{BWT}(i') = \mathcal{M}(\mathbf{S}_c(X_{i'-h};\theta_{i'}),\tilde{Y}_{i'-h})\ ,$$

where h refers to the backward time shift.

In addition, we report the **Final Backward Transfer (Final BWT)**. Given a stream of length K, we evaluate the backward transfer of the final model θ_K on the entire stream, *i.e.* setting $h = 0$ in BWT. This metrics allows to evaluate the final student model on all previous domains, rather than only one specific past domain.

Forward Transfer (FWT): Similar to the backward transfer, we adapt the discrete version [14] of forward transfer for our online setup as follows:

$$\text{FWT}(i') = \mathcal{M}(\mathbf{S}_c(X_{i'+h};\theta_{i'}),\tilde{Y}_{i'+h})\ .$$

Forward transfer measures the model's performance on future unseen data. In our case, this metric is useful in evaluating the current model on the next domain.

4. Experiments

In this section, we first describe the experimental setup on which we benchmark our continual online distillation framework. Next, we provide quantitative results including a comparative study, of our framework using our proposed evaluation methodology. Finally, we display some qualitative results to show the practical impact for autonomous driving applications.

4.1. Experimental setup

Our online continual learning framework is agnostic to the task, metric, and training parameters. In this section, we provide the technical details describing our experiments in various settings.

Task. We benchmark our framework on the outdoor semantic segmentation task, which consists in assigning a class label to each pixel of a frame. We study the particular case of videos taken behind the windshield of vehicles, which is the typical study-case for autonomous driving applications.

Dataset. The online distillation framework requires long untrimmed videos, in our case containing cyclic domain shifts. Additionally, these videos must be relevant to highlight the task's objectives. Since most datasets for semantic segmentation are composed of frames or small video clips (*e.g.*, CityScapes [13], BDD100K [44], *etc.*), they cannot be used in our context of online continual learning. Hence, we follow the same strategy to simulate long videos with domain shifts as in [10] and propose to artificially construct a video $\mathcal{V}$ by concatenating sequences from 2 different domains, D^A and D^B, alternating in cycle from one domain to the other. The resulting video is therefore an ordered set $\mathcal{V} = \{\mathcal{V}_1^A, \mathcal{V}_1^B, \mathcal{V}_2^A, \mathcal{V}_2^B, ...\}$, where the $\mathcal{V}_i^A$ and $\mathcal{V}_i^B$ are sequences from domain D^A and domain D^B, respectively. In our autonomous driving case, we define the two domains D^A and D^B as a highway environment and a downtown environment, which differ from the priors on the semantic classes (*e.g.*, there should be fewer persons in highways than downtown) or the background (*e.g.*, there are more buildings in downtown and more empty spaces in highways). We extract several clips from each domain and alternatively concatenate them to build $\mathcal{V}$. To consider clips of different time lengths, we construct two video $\mathcal{V}$ streams where the extracted clips are 20 minutes and 40 minutes long respectively.

Evaluation metric. Following the standards in semantic segmentation, we use $\mathcal{M} = mIoU$ to evaluate the segmentation masks of each frame as described in Section 3. Following the work of Cioppa *et al.* [10], since ground-truth data is unavailable for our dataset, we evaluate the performance of the student with respect to the pseudo ground truths produced by the teacher. This evaluates the capacity of the student to imitate the teacher. We provide the $mIoU$, FWT, BWT, Final BWT metrics either during the video or averaged over the entire video (referred as mean). We choose $I = 1$ minute and $h = 20$ minutes or $h = 40$ minutes depending on the domain sequences length to evaluate the forgetting on the previous or future domains. Finally, we also compute the average across a time window of $\pm$ 2 minutes of each domain shift occurrence. We call this metric $mIoU$ Near Domain Shifts ($mIoU$ NDS).

Networks and training parameters. For the teacher network $\mathbf{T}$, we chose SegFormer [43] trained on the CityScapes dataset, which is the state of the art in semantic segmentation on this dataset. For the student networks $\mathbf{S}$ and $\mathbf{S}_c$, we chose TinyNet [10, 12], a lightweight segmentation network that only needs a few training samples to specialize on a particular domain, that is fast to train, and operates in real time (at least 30 frames per second for full-HD videos on a Nvidia 1080 GPU). The student network $\mathbf{S}_c$ is trained from scratch at the beginning of the video using a learning rate of 10^{-4} and ADAM optimizer for online learning following [10]. The replay buffer size is set to $M = 250$ and the number of selected frames to $N = 100$ frames. Given the chosen video, networks, and replay buffer size,

Table 1. **Quantitative results.** We compare several memoryless and replay-based methods with the original baseline framework proposed by Cioppa *et al.* [10]. For each category, we benchmark several selection functions f_S, update functions f_U, and regularizers $\mathcal{R}$. The performance is provided for our proposed evaluation metrics for the 20/40 concatenated sequences. The replay-based methods generally outperform the baseline and the memoryless methods. The LwF and MAS regularization methods decrease the performance, while ACE and RWalk increase the performance. The best results are obtained with a uniform replay buffer, MIR, MIR+ACE, and MIR+RWalk. We compare the temporal evolution of the performance of the Baseline with one of the best performing method MIR+RWalk in Figure 3.

Methods	**Parameters**			**Metrics (mean %)**				
	f_S	f_U	$\mathcal{R}$	$mIoU$	$mIoU$ NDS	FWT	BWT	Final BWT
Memoryless	/	/	/	18.4/19.4	14.9/15.1	6.8/4.8	7.8/7.5	14.9/15.0
	/	/	MAS	14.0/14.0	13.0/13.3	11.1/11.1	12.9/12.9	14.2/14.2
	/	/	LwF	15.7/15.9	12.0/11.0	9.7/6.8	11.3/8.9	14.7/12.9
	/	/	RWalk	18.3/19.3	14.6/14.7	7.5/4.7	8.6/6.5	15.1/14.2
Baseline	All	FIFO	/	23.4/24.2	19.8/18.2	14.5/9.5	17.7/13.9	21.9/19.9
Replay Buffer	Uniform	Uniform	/	25.5/25.0	23.6/21.1	**22.2**/17.3	30.6/28.8	29.4/28.4
	Prioritized	Prioritized	/	25.1/25.1	23.2/20.8	21.3/17.3	29.2/28.4	29.2/28.9
	MIR	Uniform	/	25.2/25.2	23.7/**24.5**	21.9/**22.5**	30.5/28.6	29.5/29.7
	MIR	Uniform	MAS	14.5/14.9	13.4/14.7	12.1/13.6	13.9/15.2	15.1/15.4
	MIR	Uniform	LwF	18.7/18.1	17.6/15.7	17.4/13.9	21.0/20.2	22.4/21.1
	MIR	Uniform	ACE	**25.6/25.5**	**24.2**/21.8	22.0/17.5	**30.8**/29.4	28.8/28.5
	MIR	Uniform	RWalk	25.2/25.4	23.4/22.0	21.8/18.0	30.0/**30.8**	**30.1/30.8**

the rates are: $r_{\mathcal{V}} = 30$ frames per second, $r_{\mathbf{T}} = 3$ seconds per frame, and $r_{\mathbf{S}_c} = 60$ seconds per epoch.

4.2. Quantitative results

We compare the performance of the original framework with the proposed continual learning approaches. As a naive approach, we also study a memoryless online distillation framework, in which the online dataset does not store any frame. In this setup, the pairs produced by the teacher are used only once for training and are then deleted. As can be seen from Table 1, the memoryless approaches perform worse than the original framework for all metrics, showing that retaining some information in an online dataset (or replay buffer) improves the performance. Interestingly, all replay-based methods without regularizers improve compared to the baseline, with the best performance obtained by MIR overall. Adding a regularizer is however not always beneficial. For instance, MAS and LwF systematically decrease the performance, while ACE and RWalk slightly increase the performance. We hypothesize that this can be attributed to the fact that MAS and LwF were proposed in the offline setup with the aim of reducing the elasticity of the model towards adapting to new information. While this approach was proven to be useful in several scenarios, it could hinder the student from quickly adapting to new domains in the online setup. The biggest improvement is therefore mainly due to the replay buffer method with MIR.

In Figure 3, we show the evolution of the performance over time for the baseline and on one of the best method (MIR+RWalk) for cycles of 20. As can be seen from the $mIoU$ plot, during the two first cycles, both methods have similar results. This is expected as they both discover the new domains. The first difference can be seen at the second transition, where the first domain is seen once again. The baseline method has a huge drop, while the continual learning method shows good performance. At each other transition, MIR+RWalk does not suffer from the drop in performance caused by the forgetting of the previous domain. We conduct a comparison between the MIR+RWalk method and the original online distillation framework (baseline) by analyzing the performance evolution of the $mIoU$, BWT, Final-BWT, and FWT metrics. When evaluated on the previous domain, MIR+RWalk significantly outperforms the baseline in BWT, indicating its ability to retain information about the previous domain on frames it has been trained on. In the case of Final-BWT, the baseline quickly forgets past knowledge, while MIR+RWalk is able to maintain high performance for both domains across many cycles. Finally, when evaluated on the future domain, MIR+RWalk also shows significant performance improvements compared to the baseline in FWT, indicating its ability to generalize on new frames from a previous domain.

4.3. Qualitative results

We qualitatively demonstrate the effect of the best performing continual learning methods on the catastrophic forgetting. To do so, we investigate the quality of the segmentation masks right after the second transition from high-

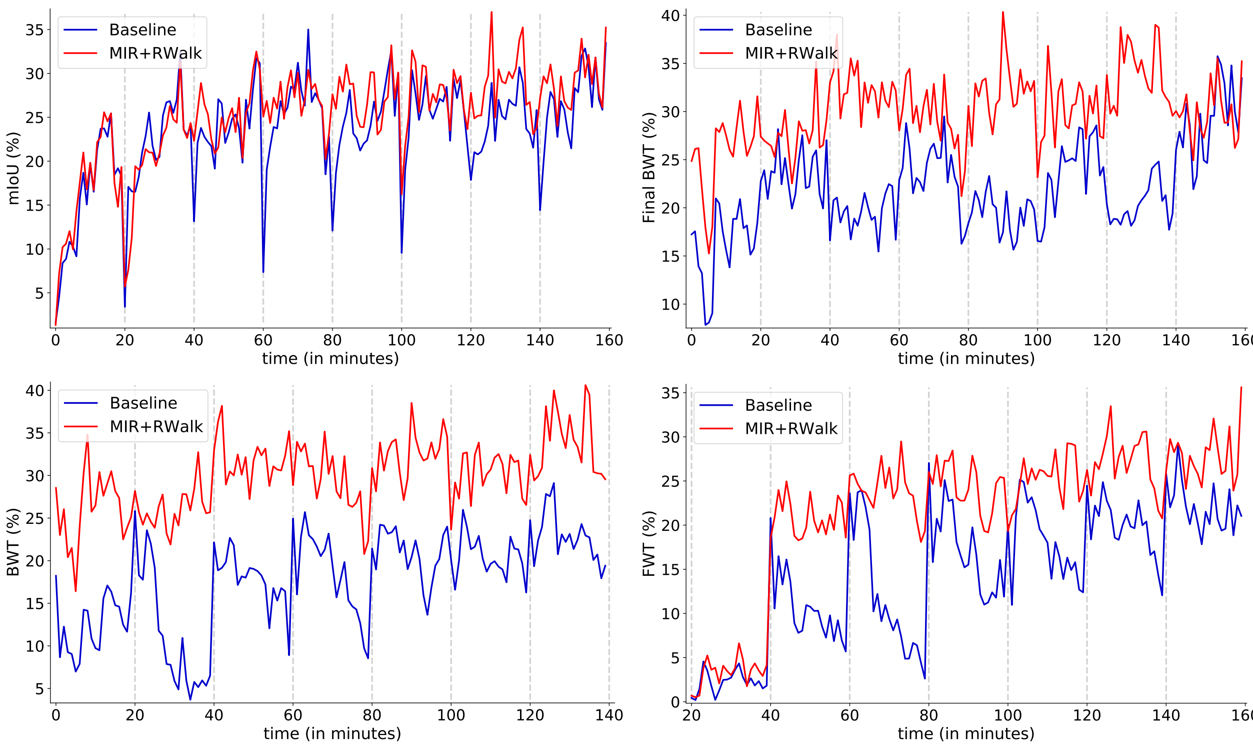


Figure 3. **Evolution of the performance over time.** We compare the evolution with respect to $mIoU$, BWT, Final-BWT, and FWT of the MIR+RWalk method with the original online distillation framework (baseline). (Top-left) $mIoU$: the performances are mostly similar within the domain, but around the domain shifts (from the second cycle), the baseline suffers from forgetting while MIR+RWalk keeps high performance. (Bottom-left) BWT: when evaluating on the previous domain, MIR+RWalk clearly outperforms the baseline, showing that it is able to retain information about the previous domain, on frames it has trained on. (Top-right) Final-BWT: the baseline quickly forgets past knowledge, while MIR+RWalk is able to retain high performance for both domains across many cycles. (Bottom-right) FWT: when evaluating on the future domain, MIR+RWalk also significantly outperforms the baseline, showing that it is able to generalize on new frames of a particular domain using information from a previous domain it has seen before.

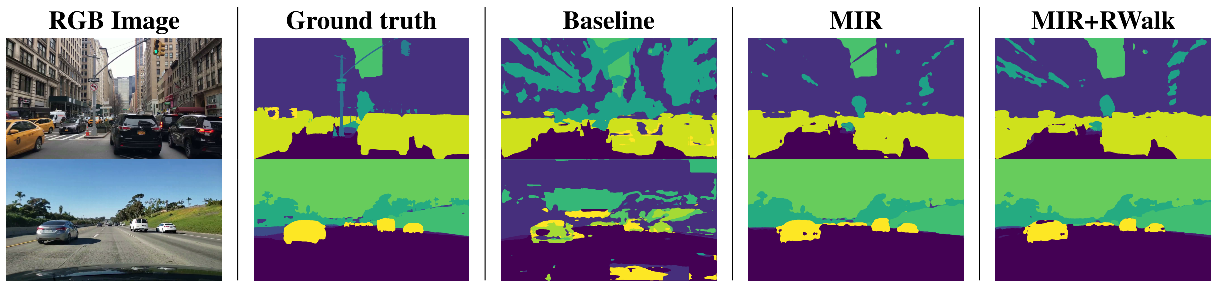


Figure 4. **Qualitative results.** Comparison of the segmentation masks obtained by different online continual learning methods: (top row) a frame taken right after second transition between highway and downtown, and (bottom row) a frame taken right after seventh transition between downtown and highway. The baseline method predicts poor segmentation masks after the domain shift, even though it has already seen this domain before. In contrast, MIR and MIR+RWalk produce better segmentation masks.

way to downtown (the student has seen the downtown only once before), and the seventh's transition from downtown to highway (the student has already seen the highway domain 6 times before). Figure 4 compares the segmentation masks obtained by the baseline method, MIR, and MIR+RWalk with the ground-truth mask. As shown, even though the student has already seen the domain previously, the segmentation masks of the baseline right after the domain shift are very poor. In practice, this could lead to hazardous situations for the autonomous vehicle and its passengers. On the contrary, the segmentation masks obtained with MIR and MIR+RWalk are much closer to the ground-truth masks. The quantitative results demonstrate that incorporating continual learning algorithms into the online distillation framework considerably enhances the quality of the predictions, rendering it more viable for real-world applications.

5. Conclusion

In conclusion, the development of online distillation has brought new opportunities for adapting deep neural networks in real time, making them more suitable for practical applications such as autonomous driving. However, the issue of catastrophic forgetting when the domain shifts has been a major challenge in the implementation of this technique. In this paper, we proposed a novel solution to this issue by incorporating continual learning methods. Through our experimentation, we evaluated several state-of-the-art continual learning methods and demonstrated their effectiveness in reducing catastrophic forgetting. We also conducted a detailed analysis of our proposed solution in the case of cyclic domain shifts. The results highlight that our approach improves the robustness and accuracy of online distillation, making it a promising technique for real-world applications. This work represents a significant step forward in the field of online distillation and continual learning, with the potential to have a meaningful impact on various fields such as autonomous driving.

Acknowledgement. A. Cioppa is funded by the F.R.S.-FNRS. A. Halin is funded by the Walloon region (Service Public de Wallonie Recherche, Belgium) under grant No. 2010235 (ARIAC by DIGITALWALLONIA4.AI). M. Henry is funded by PIT MecaTech, Belgium, under grant No. C8650 (ReconnAIssance). 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.

References

[1] Motasem Alfarra, Zhipeng Cai, Adel Bibi, Bernard Ghanem, and Matthias Müller. SimCS: Simulation for online domain-incremental continual segmentation. *CoRR*, abs/2211.16234, 2022. 2

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 11207 of *Lect. Notes Comput. Sci.*, pages 144–161. Springer Int. Publ., 2018. 2, 4

[3] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019. 2, 3

[4] Fatemeh Azimi, Sebastian Palacio, Federico Raue, Jorn Hees, Luca Bertinetto, and Andreas Dengel. Self-supervised test-time adaptation on video data. In *IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, pages 2603–2612, Waikoloa, HI, USA, Jan. 2022. Inst. Electr. Electron. Eng. (IEEE). 2

[5] Olivier Barnich and Marc Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Trans. Image Process.*, 20(6):1709–1724, Jun. 2011. 3

[6] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *Int. Conf. Learn. Represent. (ICLR)*, 2022. 4

[7] Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 8261–8270, Montréal, Can., Oct. 2021. Inst. Electr. Electron. Eng. (IEEE). 1

[8] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 11215 of *Lect. Notes Comput. Sci.*, pages 556–572. Springer Int. Publ., 2018. 2, 4

[9] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. In *Int. Conf. Mach. Learn. (ICML)*, 2019. 1, 2

[10] Anthony Cioppa, Adrien Deliege, Maxime Istasse, Christophe De Vleeschouwer, and Marc Van Droogenbroeck. ARTHuS: Adaptive real-time human segmentation in sports through online distillation. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports*, pages 2505–2514, Long Beach, CA, USA, Jun. 2019. Inst. Electr. Electron. Eng. (IEEE). 1, 2, 3, 4, 5, 6

[11] Anthony Cioppa, Adrien Deliège, Noor Ul Huda, Rikke Gade, Marc Van Droogenbroeck, and Thomas B. Moeslund. Multimodal and multiview distillation for real-time player detection on a football field. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports*, pages 3846–3855, Seattle, WA, USA, Jun. 2020. 2

[12] Anthony Cioppa, Adrien Deliège, and Marc Van Droogenbroeck. A bottom-up approach based on semantics for the interpretation of the main camera stream in soccer games. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW), CVsports*, pages 1846–1855, Salt Lake City, UT, USA, Jun. 2018. 2, 5

[13] 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 *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3213–3223, Las Vegas, NV, USA, Jun. 2016. Inst. Electr. Electron. Eng. (IEEE). 5

[14] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. In *Continual learning W., Neural Inf. Process. Syst. (NeurIPS)*, 2018. 5

[15] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A brief review of domain adaptation. *Trans. Comput. Sci. Comput. Intell.*, pages 877–894, 2021. 2

[16] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.*, 3(4):128–135, Apr. 1999. 2

[17] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-DFCIL: Relation-guided representation learning for data-free class incremental learning. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 13683 of *Lect. Notes Comput. Sci.*, pages 423–439. Springer Nat. Switz., 2022. 2

[18] Prachi Garg, Rohit Saluja, Vineeth N. Balasubramanian, Chetan Arora, Anbumani Subramanian, and C. V. Jawahar. Multi-domain incremental learning for semantic segmentation. In *IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, pages 2080–2090, Waikoloa, HI, USA, Jan. 2022. Inst. Electr. Electron. Eng. (IEEE). 2

[19] Yasir Ghunaim, Adel Bibi, Kumail Alhamoud, Motasem Alfarra, Hasan Abed Al Kader Hammoud, Ameya Prabhu, Philip H. S. Torr, and Bernard Ghanem. Real-time evaluation in online continual learning: A new hope. *CoRR*, abs/2302.01047, 2023. 2

[20] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 13923–13932, Seattle, WA, USA, Jun. 2020. Inst. Electr. Electron. Eng. (IEEE). 1

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 770–778, Las Vegas, NV, USA, Jun. 2016. Inst. Electr. Electron. Eng. (IEEE). 1

[22] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Int. Conf. Learn. Represent. (ICLR)*, 2019. 1

[23] Oguzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3D common corruptions and data augmentation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 18941–18952, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 1

[24] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proc. National Acad. Sci. (PNAS)*, 114(13):3521–3526, Mar. 2017. 1, 2, 4

[25] Jinlong Li, Runsheng Xu, Jin Ma, Qin Zou, Jiaqi Ma, and Hongkai Yu. Domain adaptive object detection for autonomous driving under foggy weather. In *IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, pages 612–622, Waikoloa, HI, USA, Jan. 2023. Inst. Electr. Electron. Eng. (IEEE). 2

[26] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, Dec. 2018. 1, 2, 4

[27] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3367–3375, Boston, MA, USA, Jun. 2015. Inst. Electr. Electron. Eng. (IEEE). 1

[28] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. TTN: A domain-shift aware batch normalization in test-time adaptation. In *Int. Conf. Learn. Represent. (ICLR)*, pages 1–19, 2023. 2

[29] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017. 2

[30] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2502–2511, Long Beach, CA, USA, Jun. 2019. Inst. Electr. Electron. Eng. (IEEE). 2

[31] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychol. Learn. Motiv.*, pages 109–165, 1989. 2

[32] Theodoros Panagiotakopoulos, Pier Luigi Dovesi, Linus Härenstam-Nielsen, and Matteo Poggi. Online domain adaptation for semantic segmentation in ever-changing conditions. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 13694 of *Lect. Notes Comput. Sci.*, pages 128–146. Springer Nat. Switz., 2022. 2

[33] Sébastien Piérard, Anthony Cioppa, Anaïs Halin, Renaud Vandeghen, Maxime Zanella, Benoît Macq, Saïd Mahmoudi, and Marc Van Droogenbroeck. Mixture domain adaptation to improve semantic segmentation in real-world surveillance. In *IEEE/CVF Winter Conf. Appl. Comput. Vis. Work. (WACVW)*, pages 22–31, Waikoloa, HI, USA, Jan. 2023. Inst. Electr. Electron. Eng. (IEEE). 2

[34] Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. GDumb: A simple approach that questions our progress in continual learning. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 12347 of *Lect. Notes Comput. Sci.*, pages 524–540. Springer Int. Publ., 2020. 2

[35] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 10745–10755, Montreal, QC, Canada, Oct. 2021. Inst. Electr. Electron. Eng. (IEEE). 1, 2

[36] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015. 3

[37] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, Jan. 2015. 1

[38] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 9354–9364, Montreal, QC, Canada, Oct. 2021. Inst. Electr. Electron. Eng. (IEEE). 2

[39] Baochen Sun and Kate Saenko. Deep CORAL: Correlation alignment for deep domain adaptation. In *Eur. Conf. Comput. Vis. (ECCV)*, volume 9915 of *Lect. Notes Comput. Sci.*, pages 443–450. Springer Int. Publ., 2016. 2

[40] Tao Sun, Mattia Segu, Janis Postels, Yuxuan Wang, Luc Van Gool, Bernt Schiele, Federico Tombari, and Fisher Yu. SHIFT: A synthetic driving dataset for continuous multi-task domain adaptation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 21339–21350, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 2

[41] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *CoRR*, abs/2006.10726, 2020. 2

[42] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 7191–7201, New Orleans, LA, USA, Jun. 2022. Inst. Electr. Electron. Eng. (IEEE). 2

[43] 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 *Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 34, pages 12077–12090, 2021. 2, 5

[44] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving dataset for heterogeneous multitask learning. In *IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2633–2642, Seattle, WA, USA, Jun. 2020. Inst. Electr. Electron. Eng. (IEEE). 2, 5

[45] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Int. Conf. Mach. Learn. (ICML)*, 2017. 4

[46] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 6230–6239, Honolulu, HI, USA, Jul. 2017. Inst. Electr. Electron. Eng. (IEEE). 2

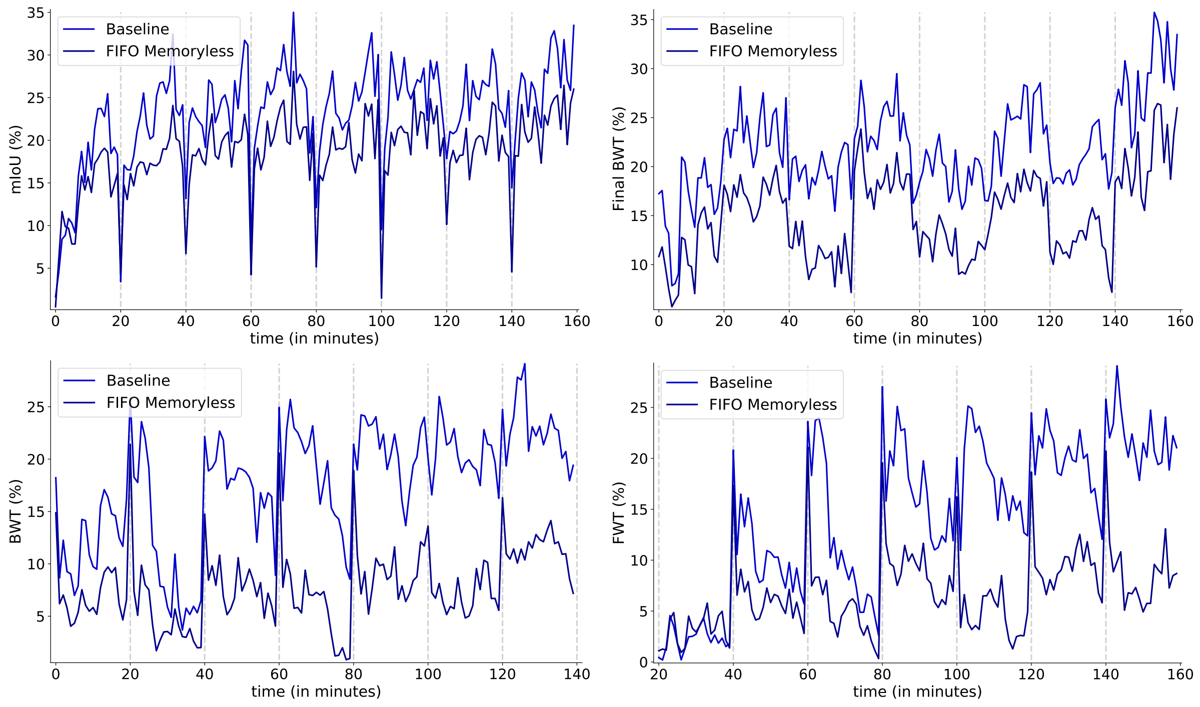
[47] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H. S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 6877–6886, Nashville, TN, USA, Jun. 2021. Inst. Electr. Electron. Eng. (IEEE). 2

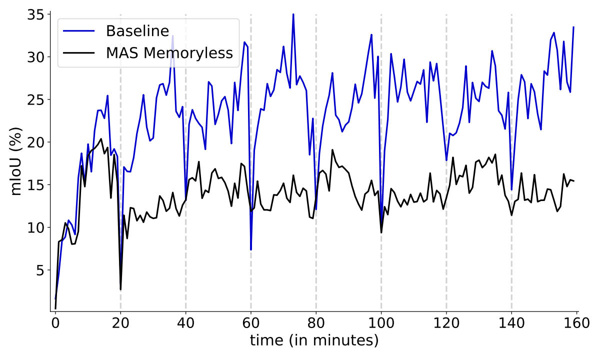
Appendix B

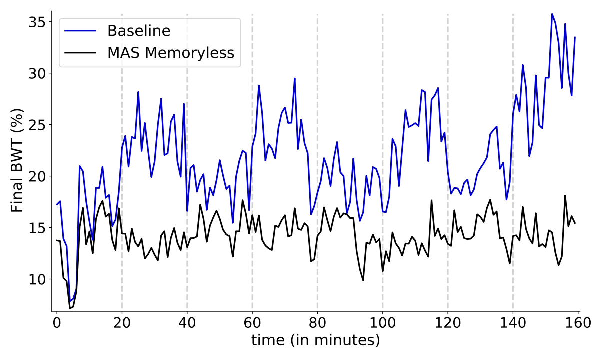
All experiment results

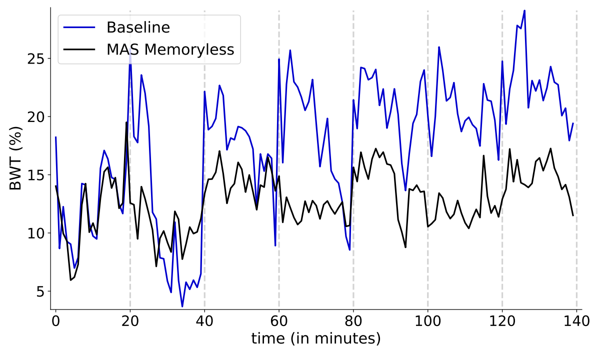
B.1 Two domains setup

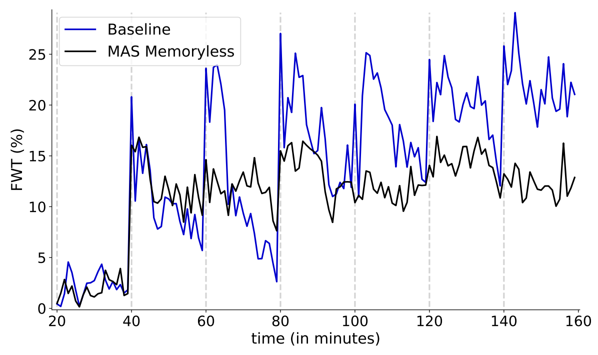
B.1.1 20 minutes sequences

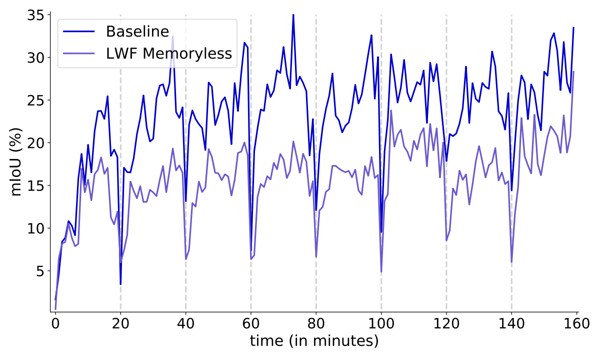


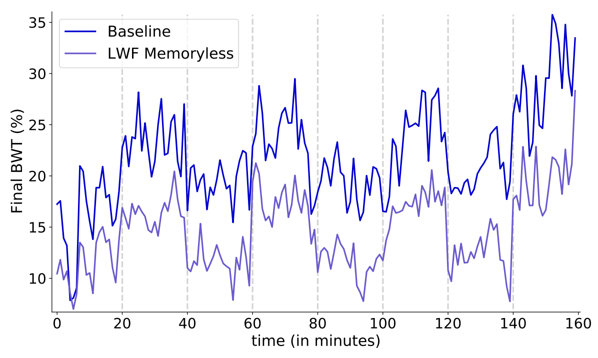

Baseline
MAS Memoryless
mIoU (%)
time (in minutes)

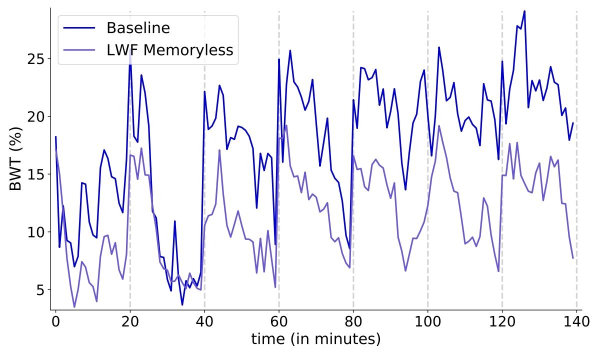

Baseline
MAS Memoryless
Final BWT (%)
time (in minutes)

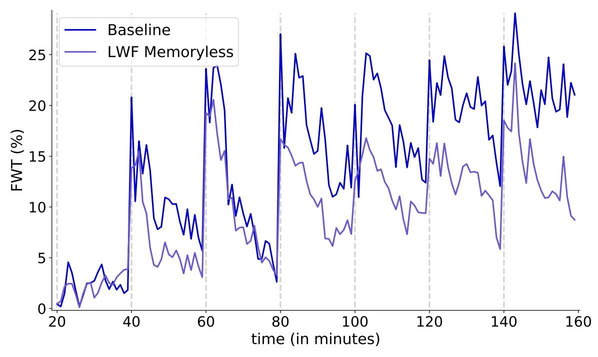

Baseline
MAS Memoryless
BWT (%)
time (in minutes)

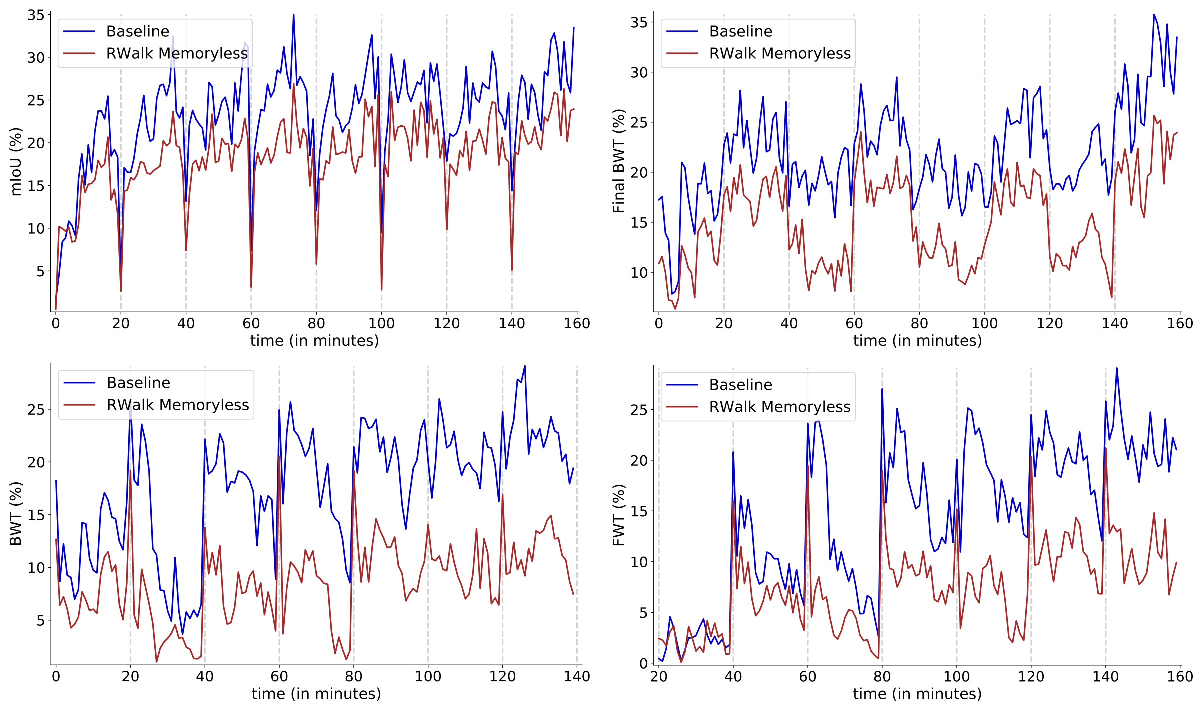

Baseline
MAS Memoryless
FWT (%)
time (in minutes)

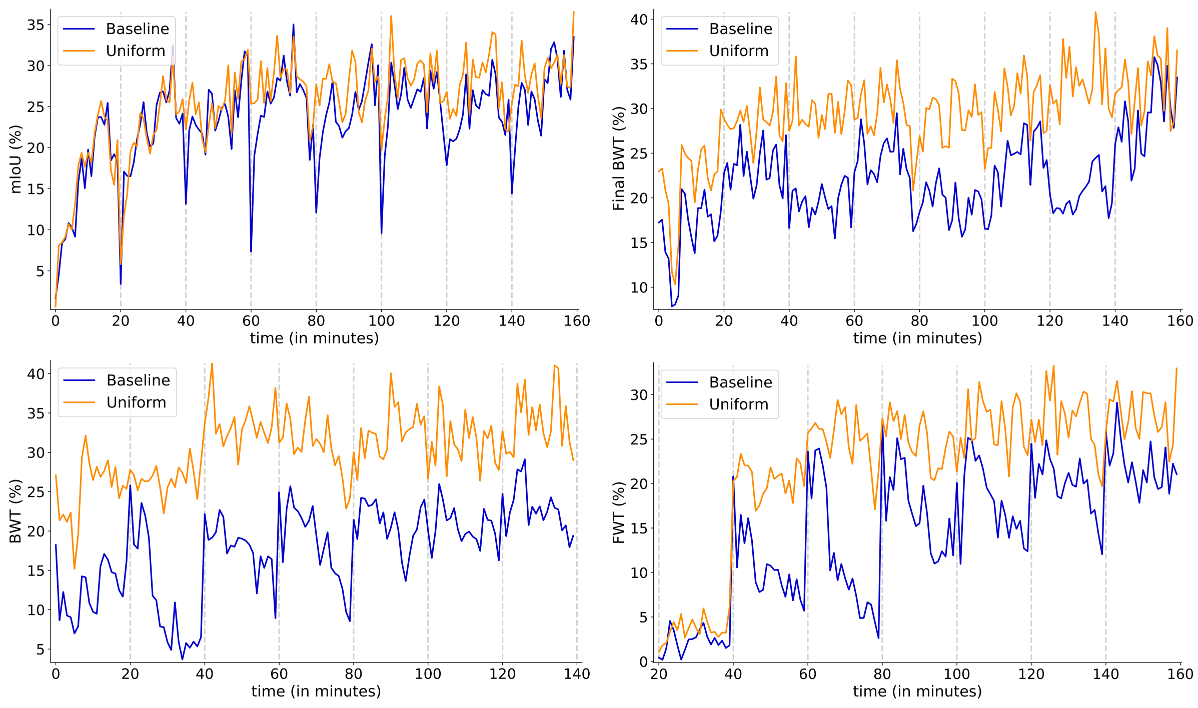

Baseline
LWF Memoryless
mIoU (%)
time (in minutes)

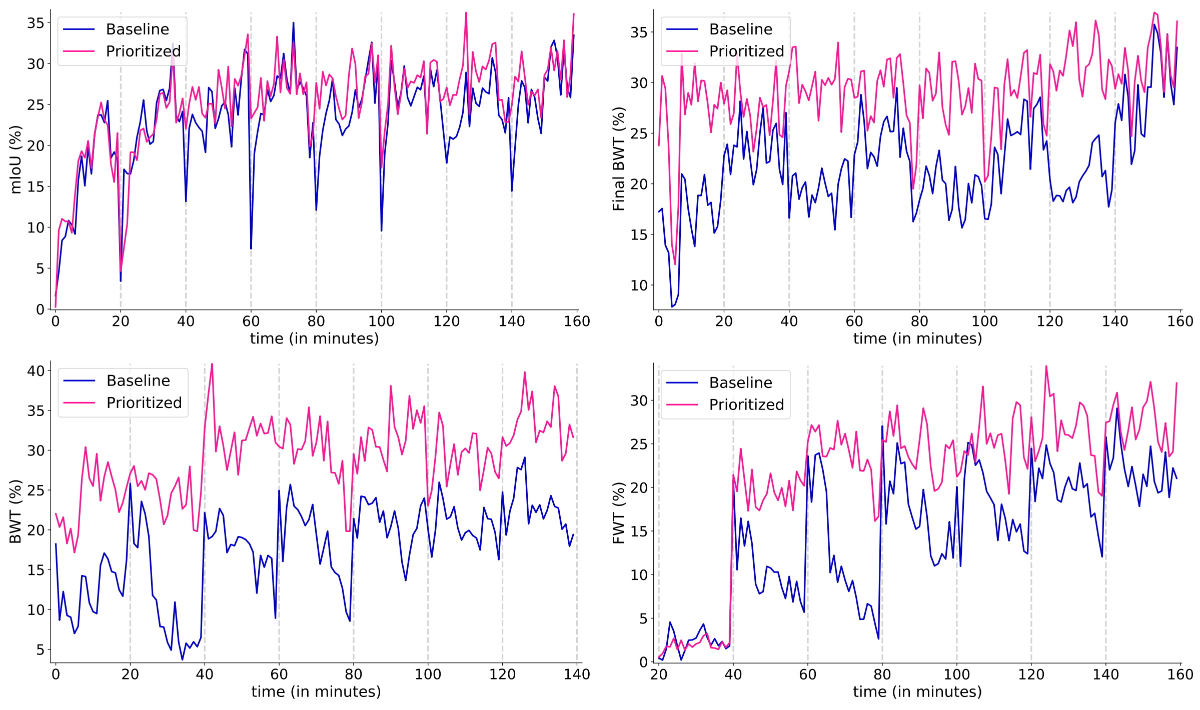

Baseline
LWF Memoryless
Final BWT (%)
time (in minutes)


Baseline
LWF Memoryless
BWT (%)
time (in minutes)

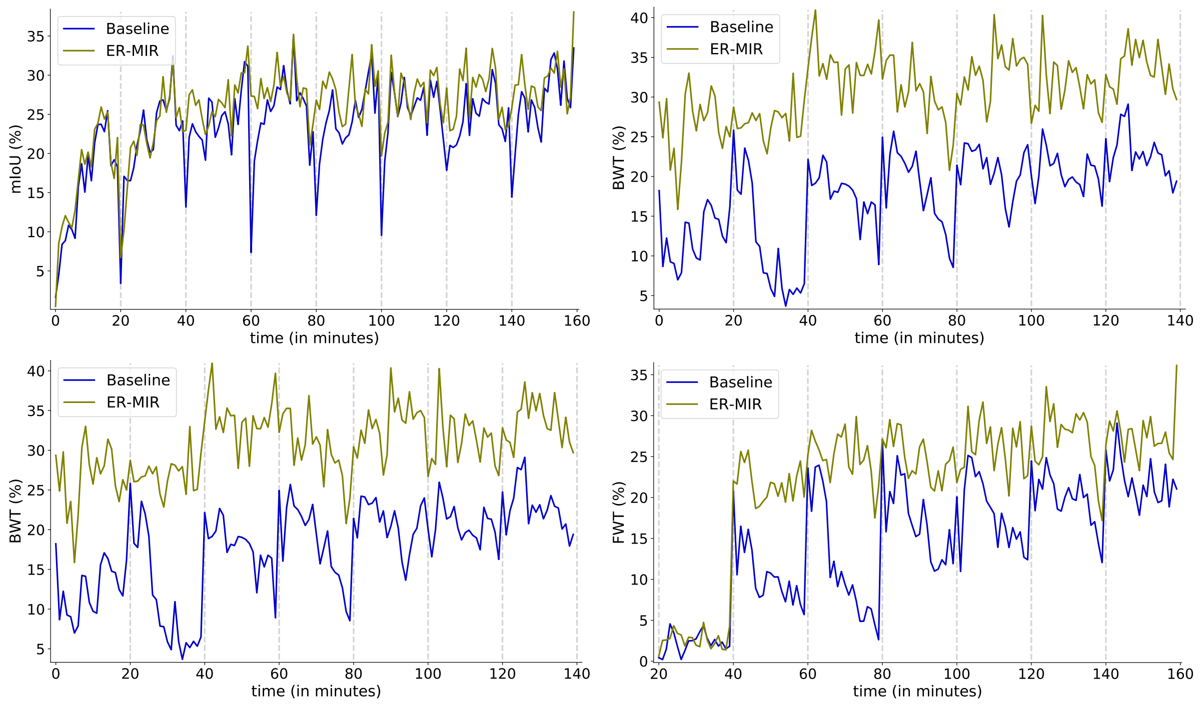

Baseline
LWF Memoryless
FWT (%)
time (in minutes)


Baseline
RWalk Memoryless
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)

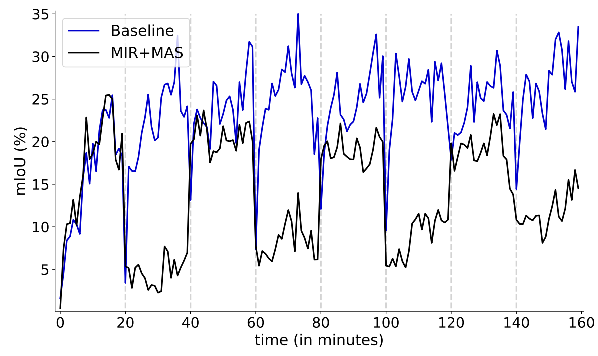

Baseline
Uniform
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)

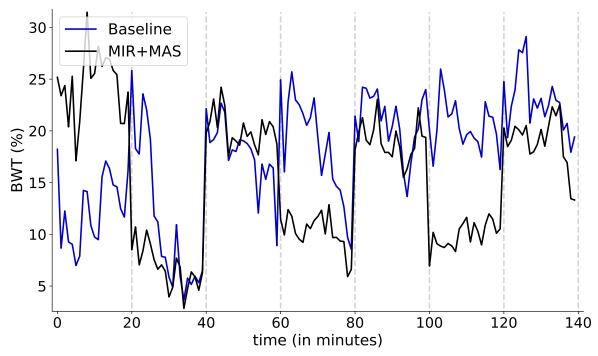


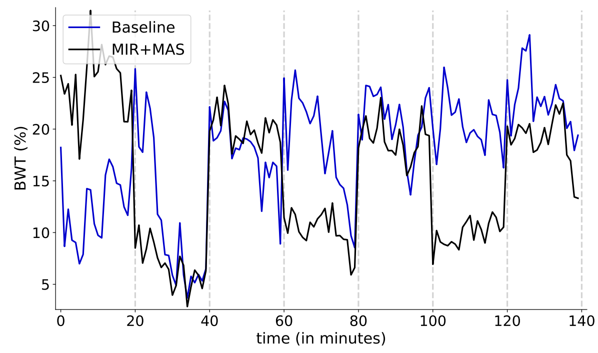
Baseline
Prioritized
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)

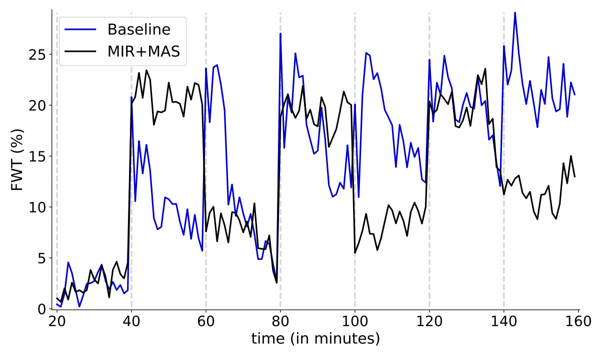


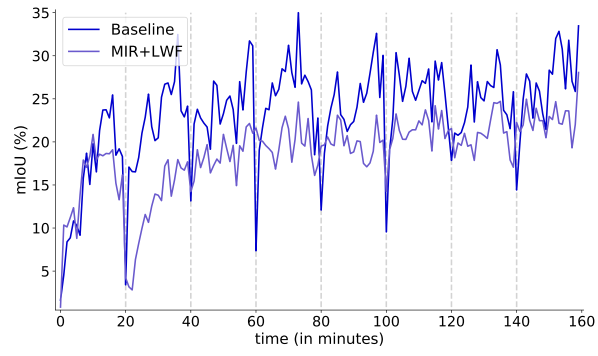
Baseline
ER-MIR
mIoU (%)
BWT (%)
FWT (%)
time (in minutes)

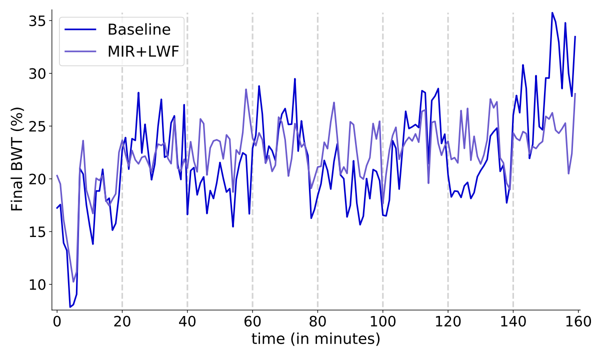

Baseline
MIR+MAS
mIoU (%)
time (in minutes)

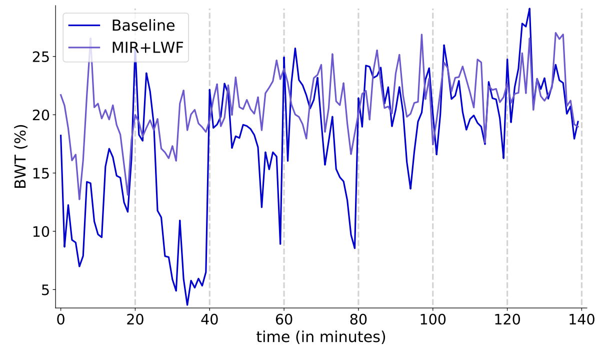

Baseline
MIR+MAS
BWT (%)
time (in minutes)

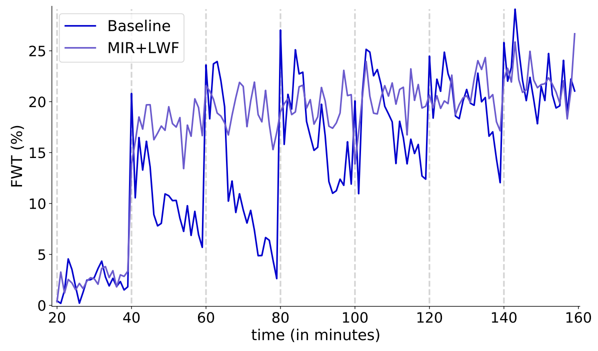

Baseline
MIR+MAS
BWT (%)
time (in minutes)

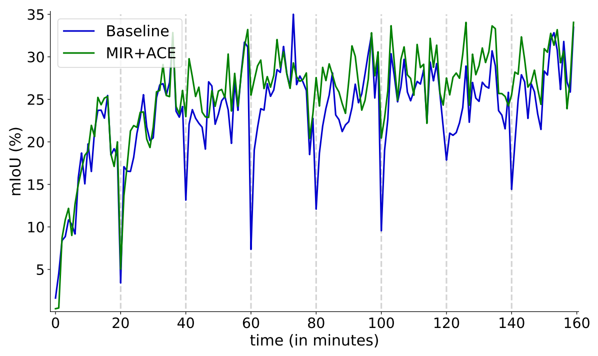

Baseline
MIR+MAS
FWT (%)
time (in minutes)

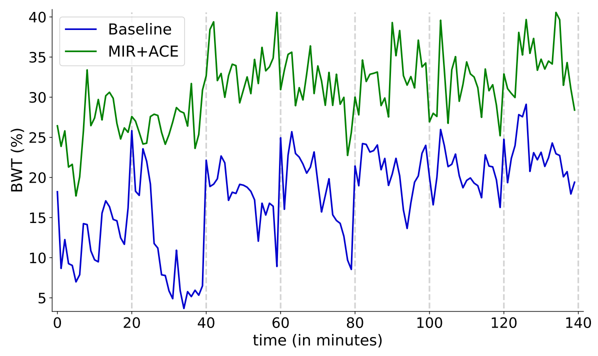

Baseline
MIR+LWF
mIoU (%)
time (in minutes)

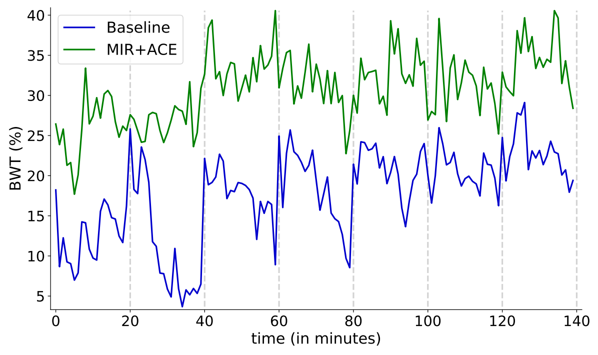

Baseline
MIR+LWF
Final BWT (%)
time (in minutes)

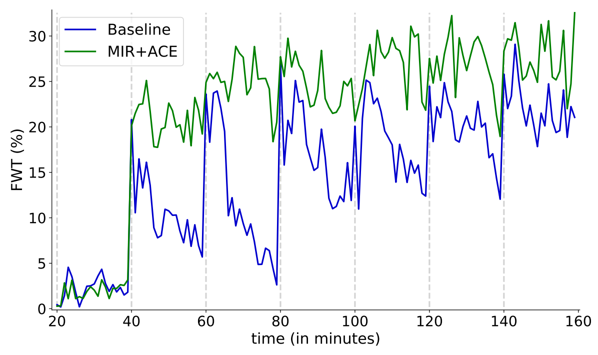

Baseline
MIR+LWF
BWT (%)
time (in minutes)

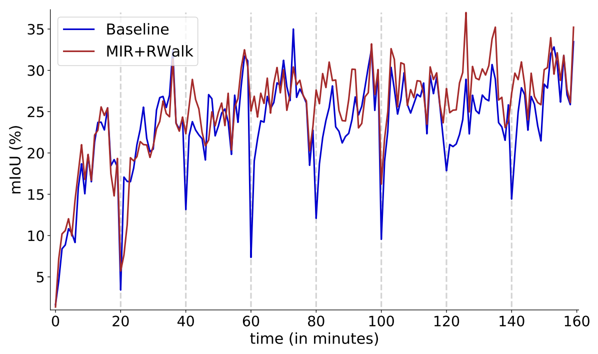

Baseline
MIR+LWF
FWT (%)
time (in minutes)

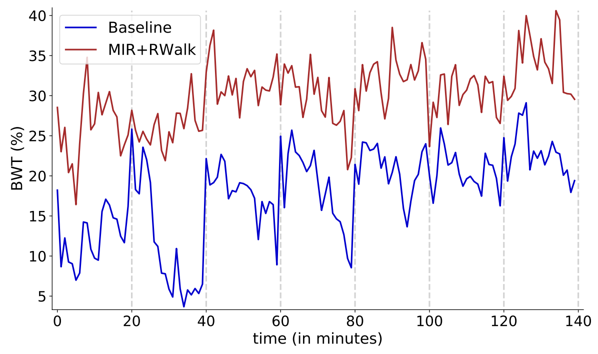

Baseline
MIR+ACE
mIoU (%)
time (in minutes)

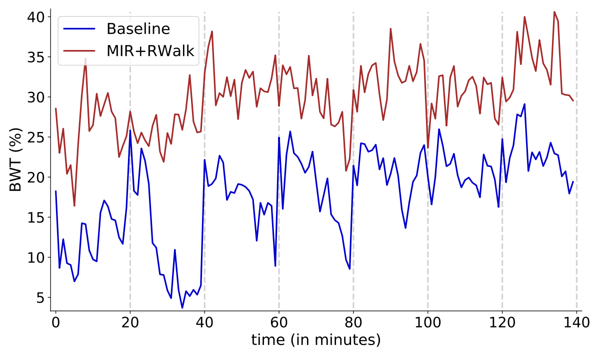

Baseline
MIR+ACE
BWT (%)
time (in minutes)

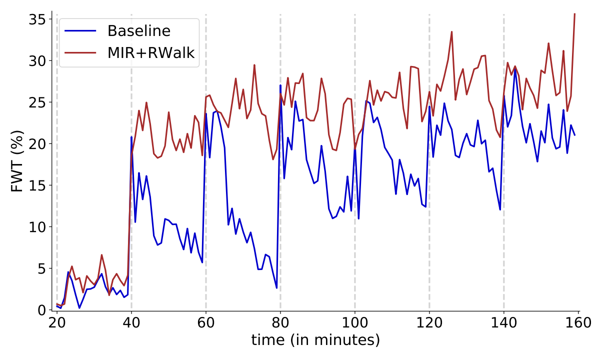

Baseline
MIR+ACE
BWT (%)
time (in minutes)


Baseline
MIR+ACE
FWT (%)
time (in minutes)

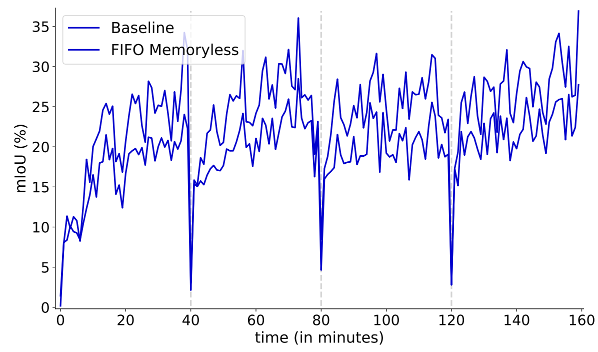

Baseline
MIR+RWalk
mIoU (%)
time (in minutes)

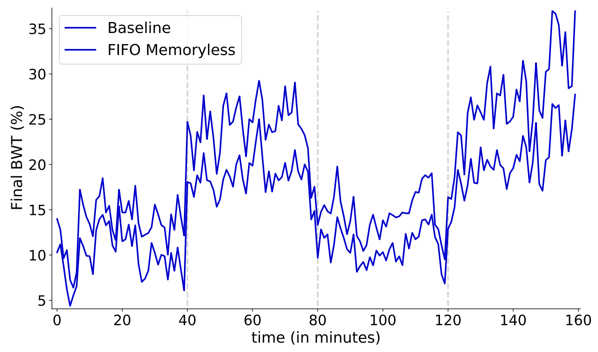

Baseline
MIR+RWalk
BWT (%)
time (in minutes)

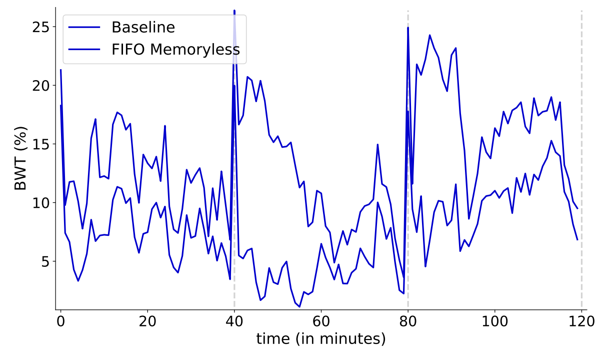

Baseline
MIR+RWalk
BWT (%)
time (in minutes)

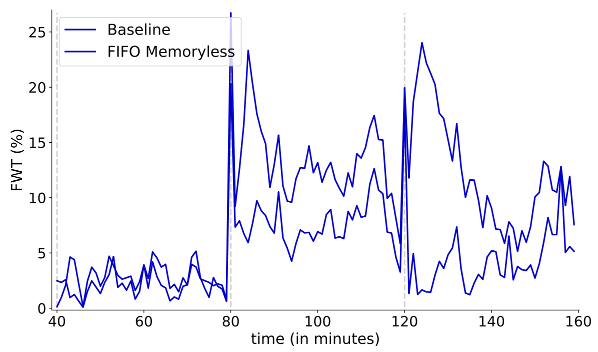

Baseline
MIR+RWalk
FWT (%)
time (in minutes)

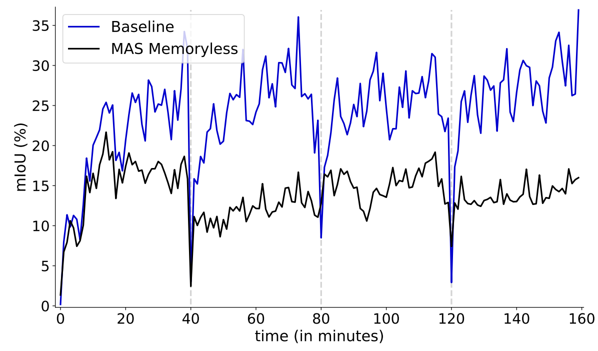
B.1.2 40 minutes sequences

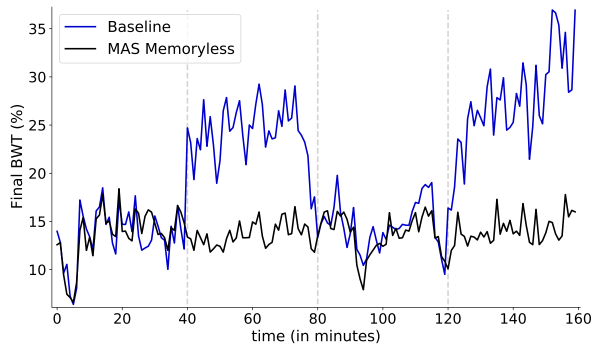


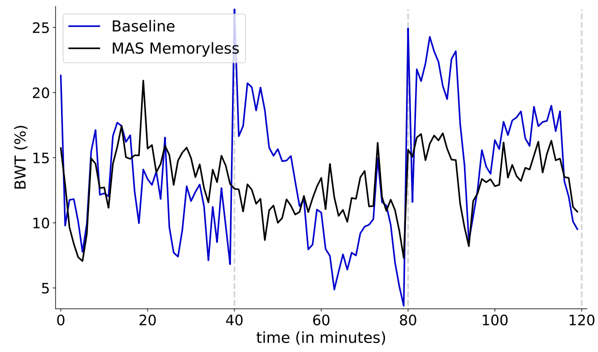


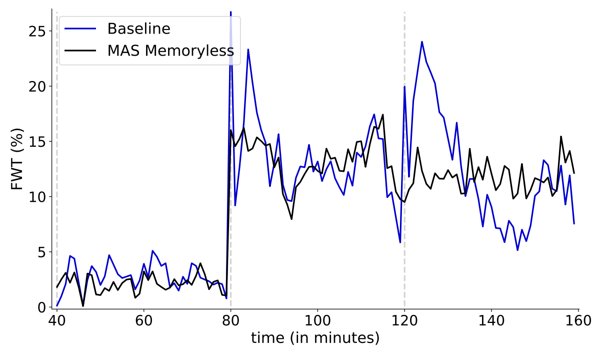


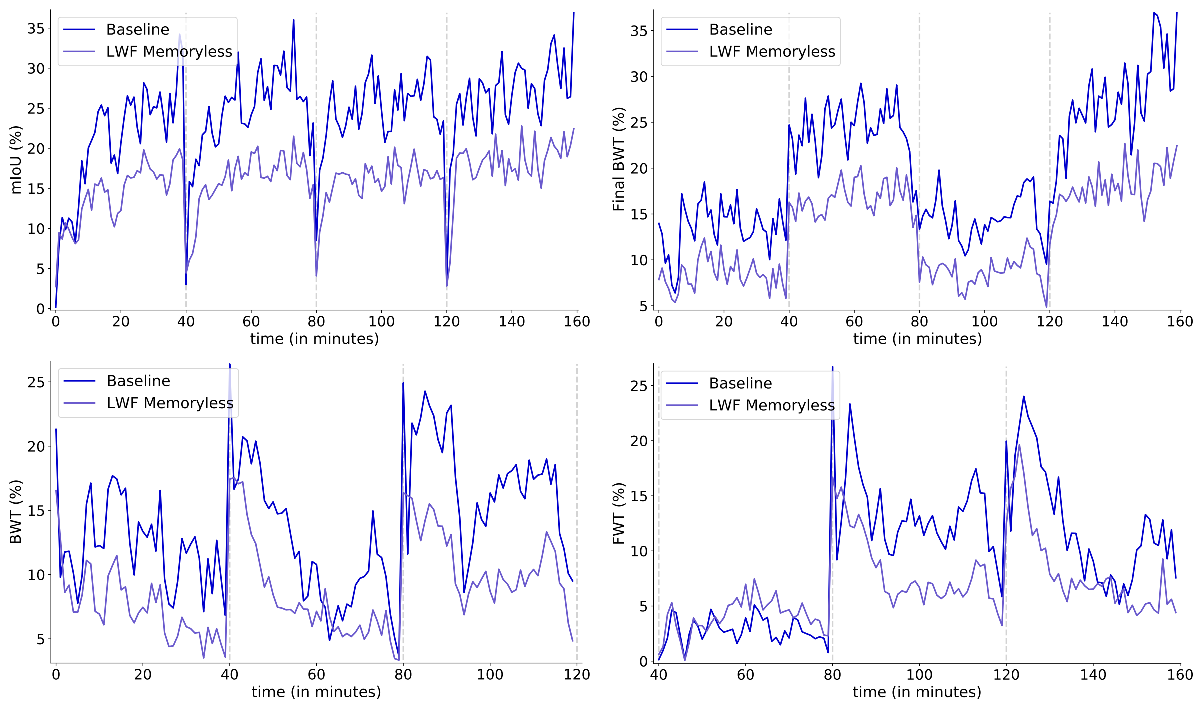


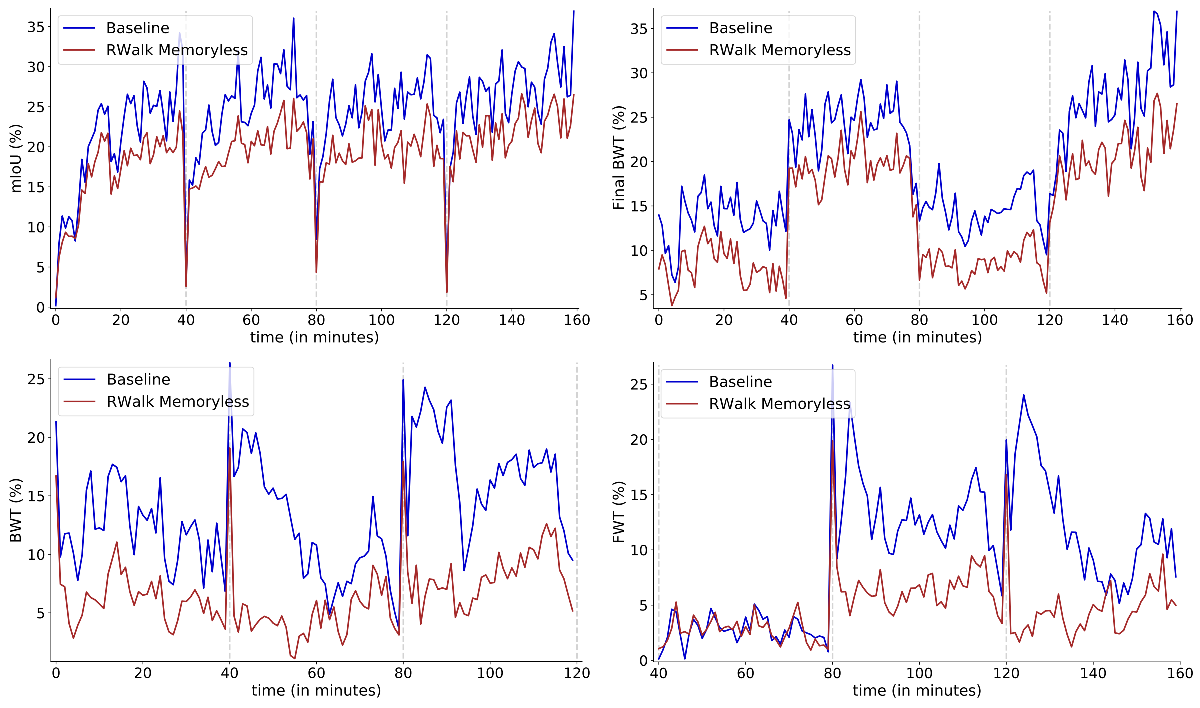


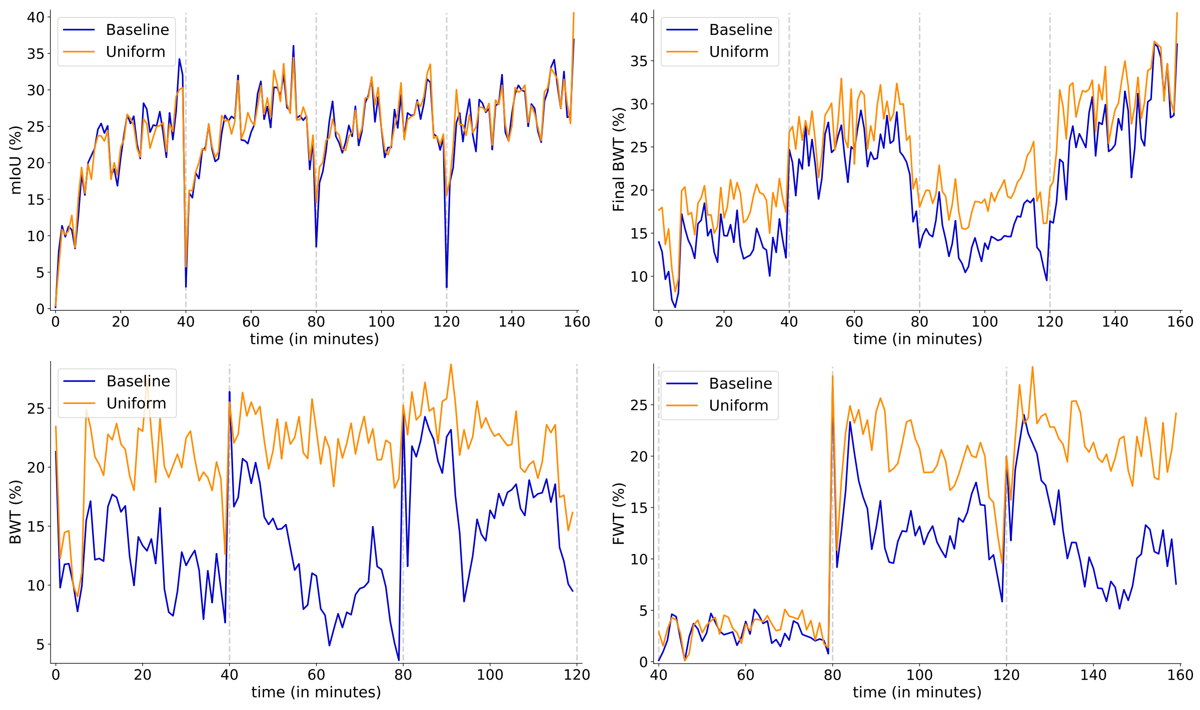


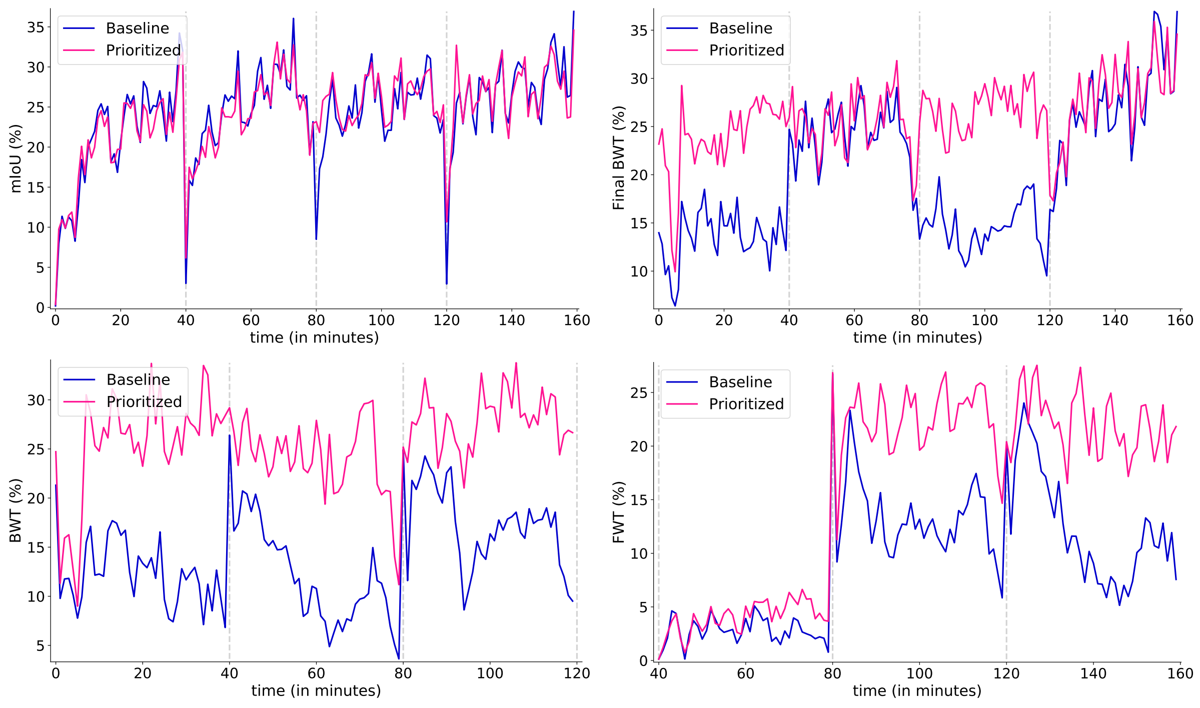


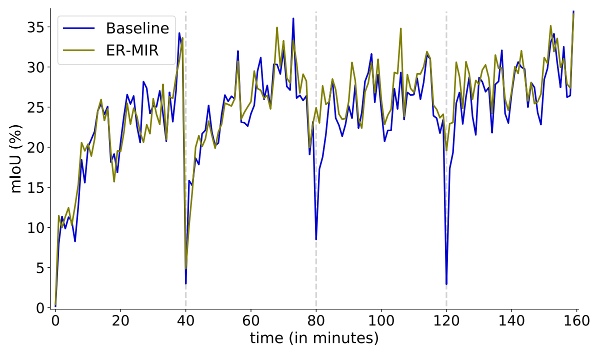
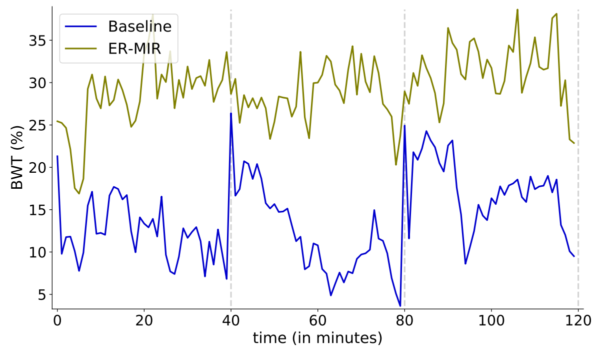
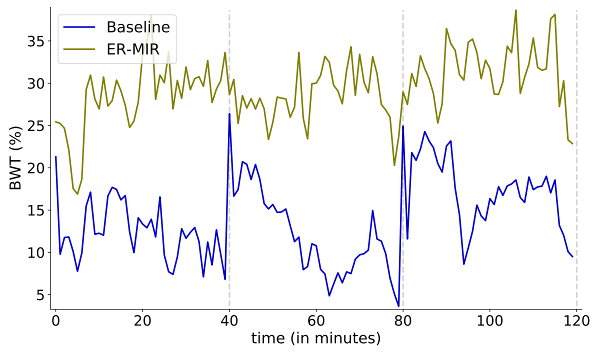
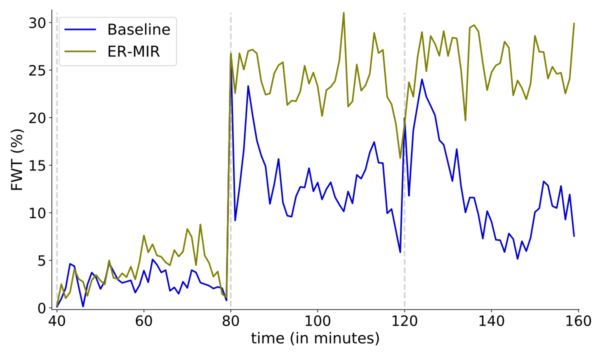
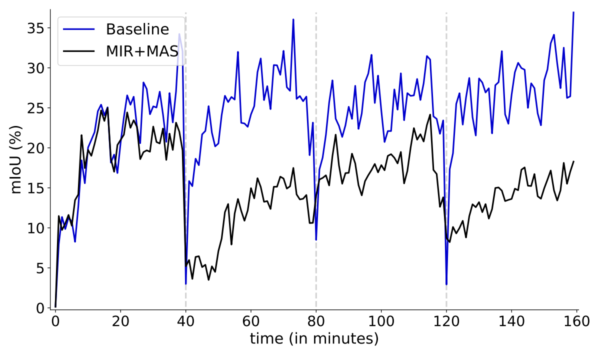
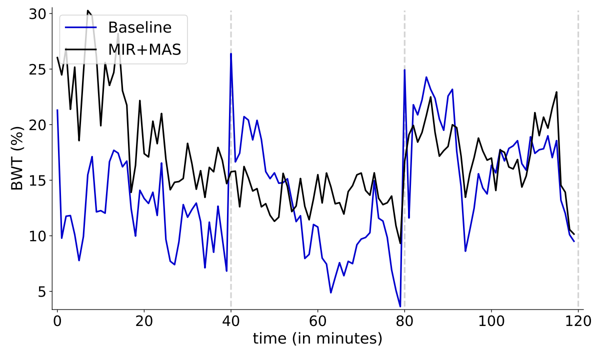
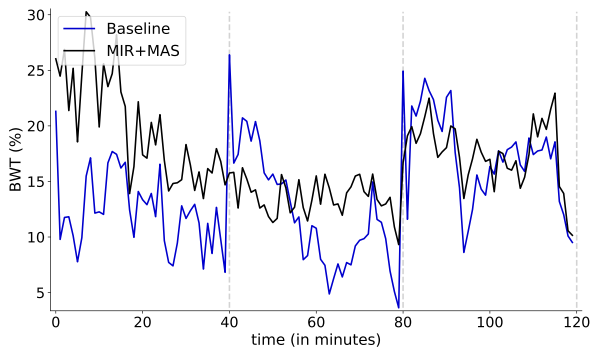
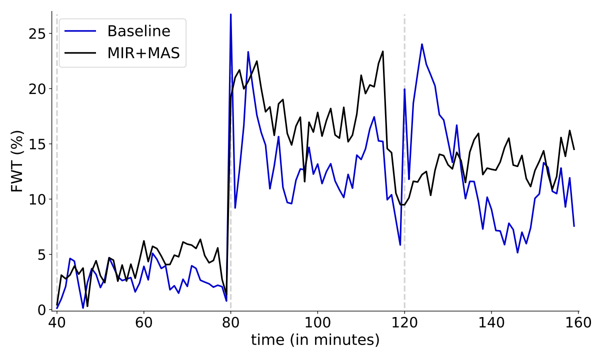


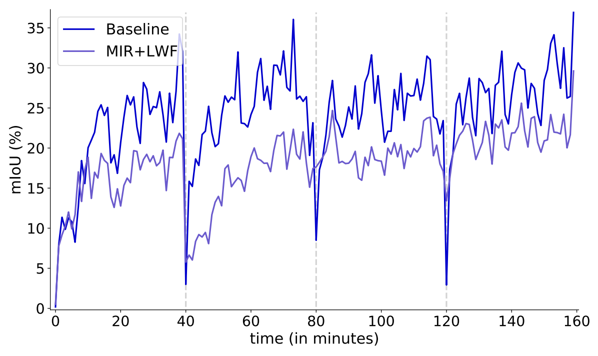

Baseline
LWF Memoryless
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)


Baseline
RWalk Memoryless
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)

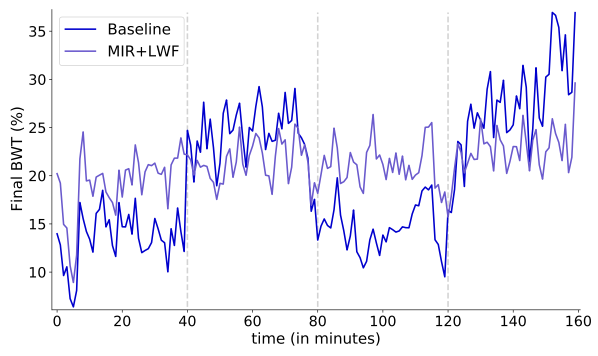

Baseline
Uniform
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)


Baseline
Prioritized
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)

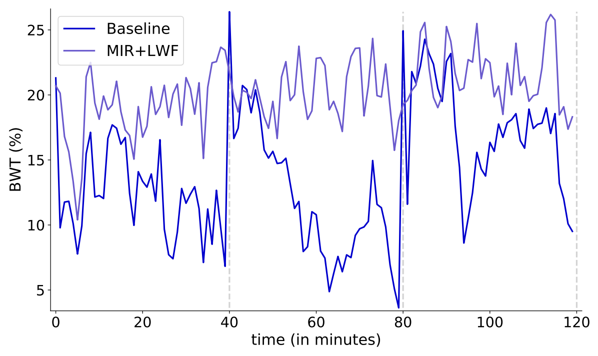

Baseline
ER-MIR
mIoU (%)
time (in minutes)

Baseline
ER-MIR
BWT (%)
time (in minutes)

Baseline
ER-MIR
BWT (%)
time (in minutes)

Baseline
ER-MIR
FWT (%)
time (in minutes)

Baseline
MIR+MAS
mIoU (%)
time (in minutes)

Baseline
MIR+MAS
BWT (%)
time (in minutes)

Baseline
MIR+MAS
BWT (%)
time (in minutes)

Baseline
MIR+MAS
FWT (%)
time (in minutes)



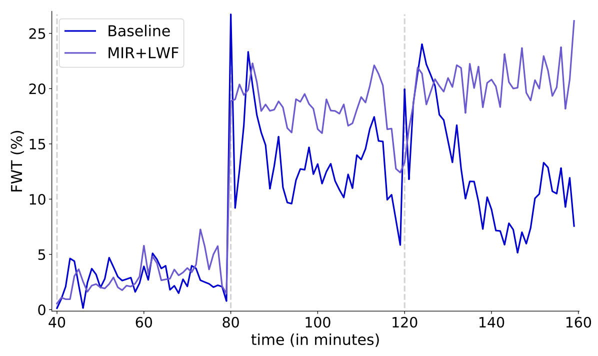
Baseline
MIR+LWF
mIoU (%)
time (in minutes)



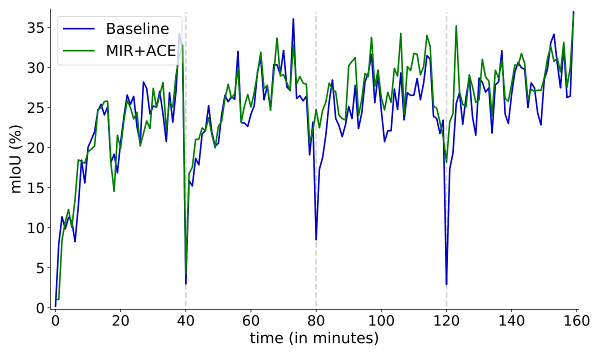
Baseline
MIR+LWF
Final BWT (%)
time (in minutes)



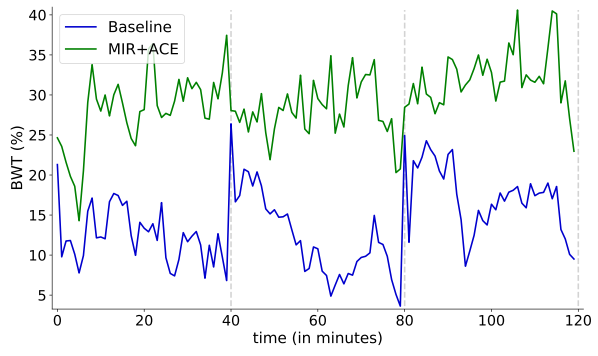
Baseline
MIR+LWF
BWT (%)
time (in minutes)



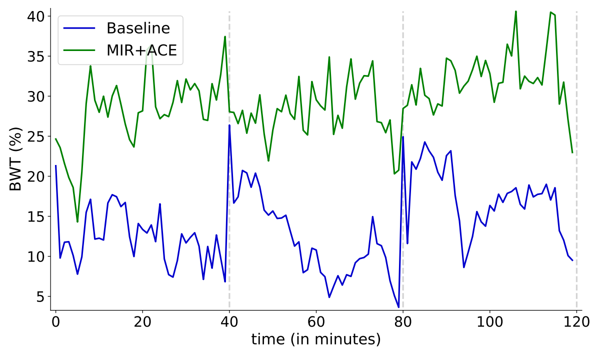
Baseline
MIR+LWF
FWT (%)
time (in minutes)



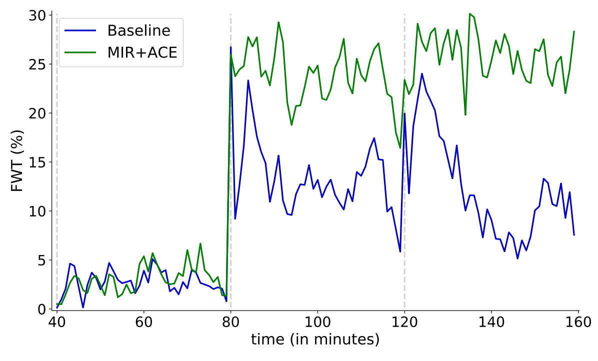
Baseline
MIR+ACE
mIoU (%)
time (in minutes)



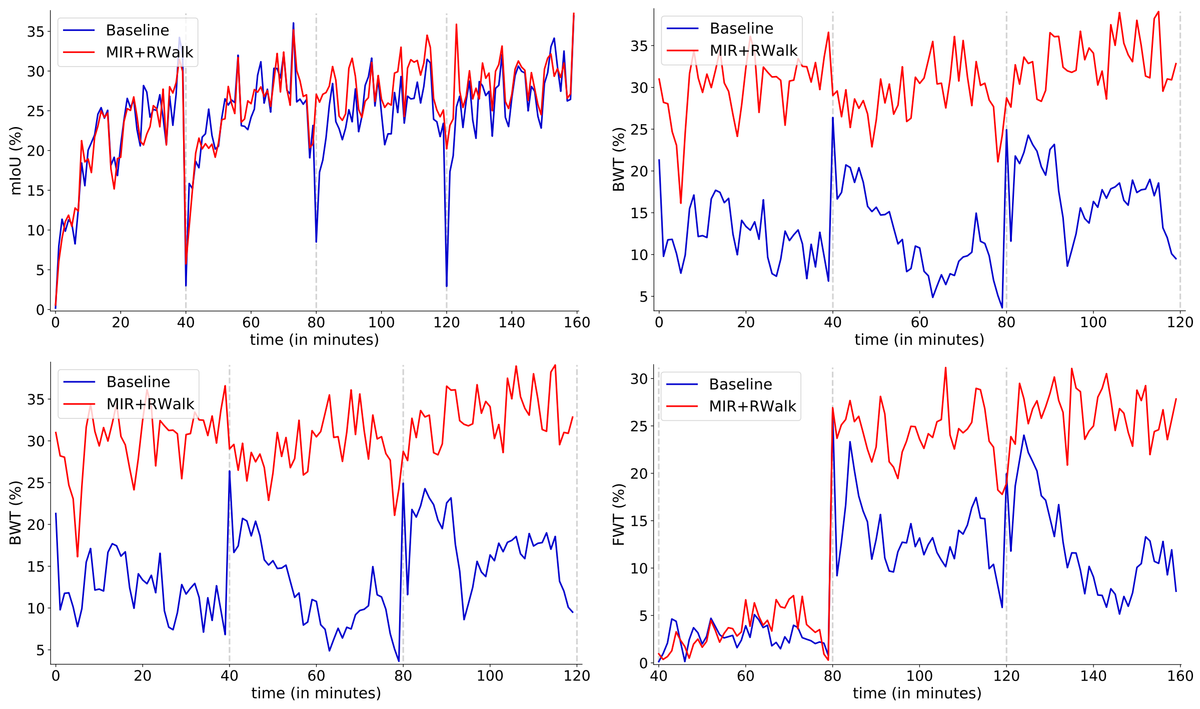
Baseline
MIR+ACE
BWT (%)
time (in minutes)



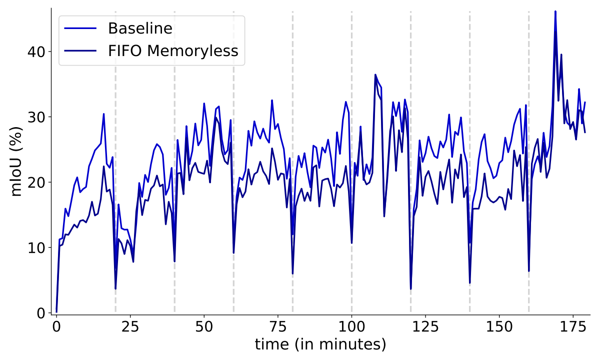
Baseline
MIR+ACE
BWT (%)
time (in minutes)

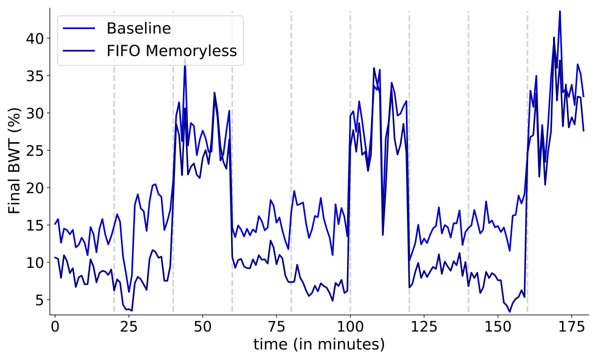


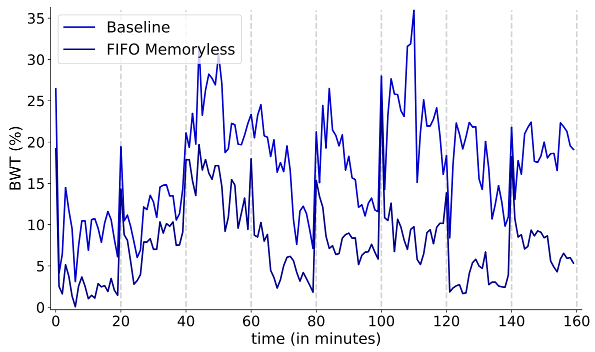
Baseline
MIR+ACE
FWT (%)
time (in minutes)

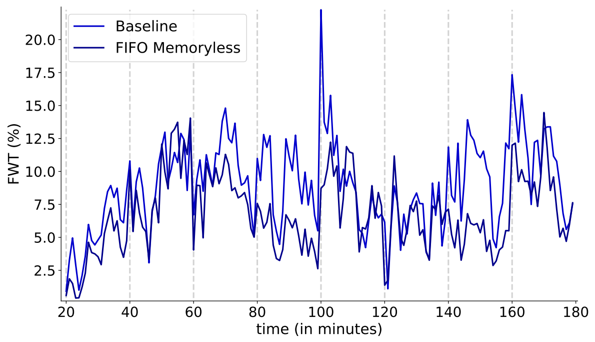

Baseline
MIR+RWalk
mIoU (%)
BWT (%)
FWT (%)
time (in minutes)

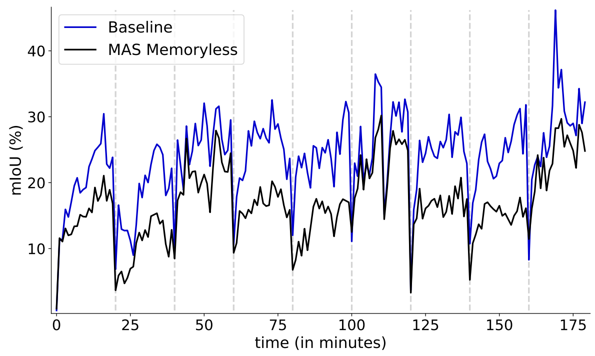
B.2 Three domains setup

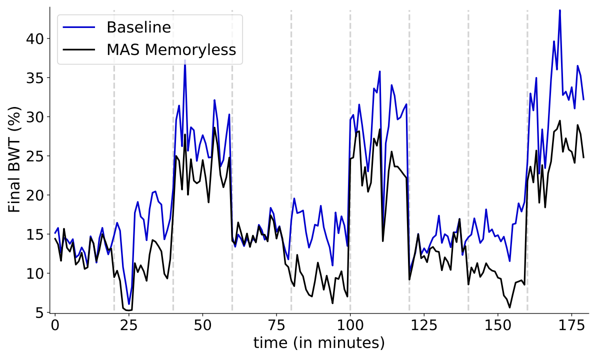


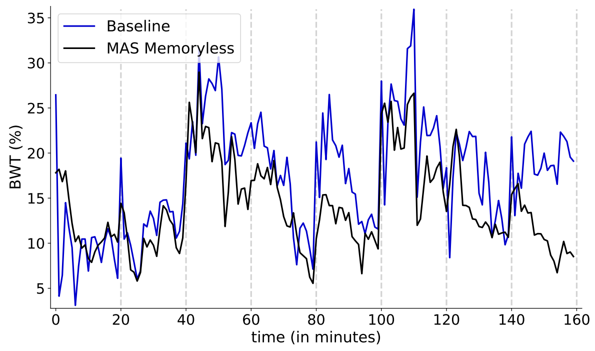


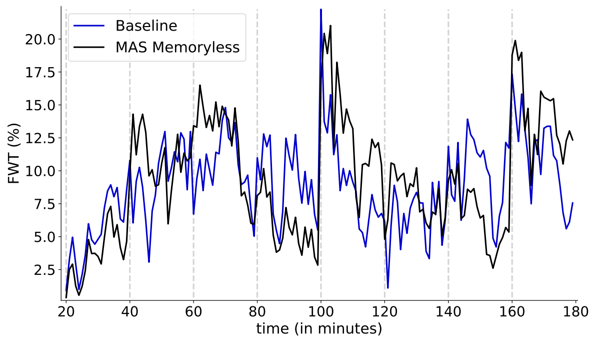


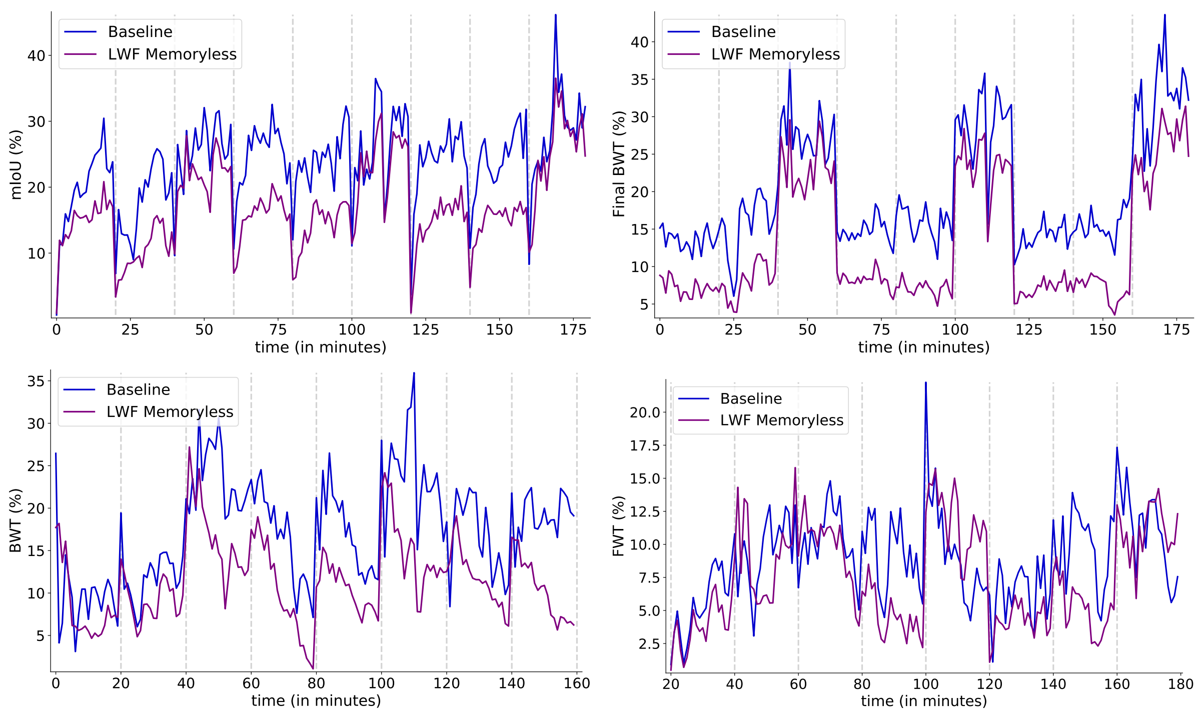


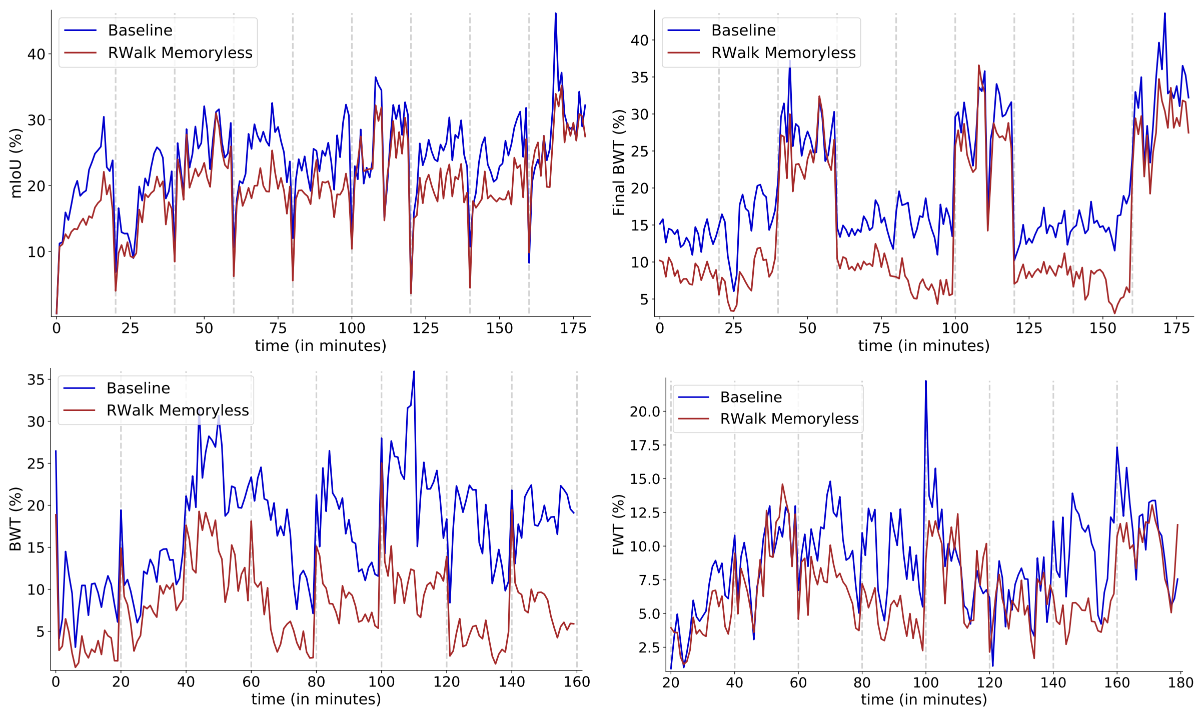


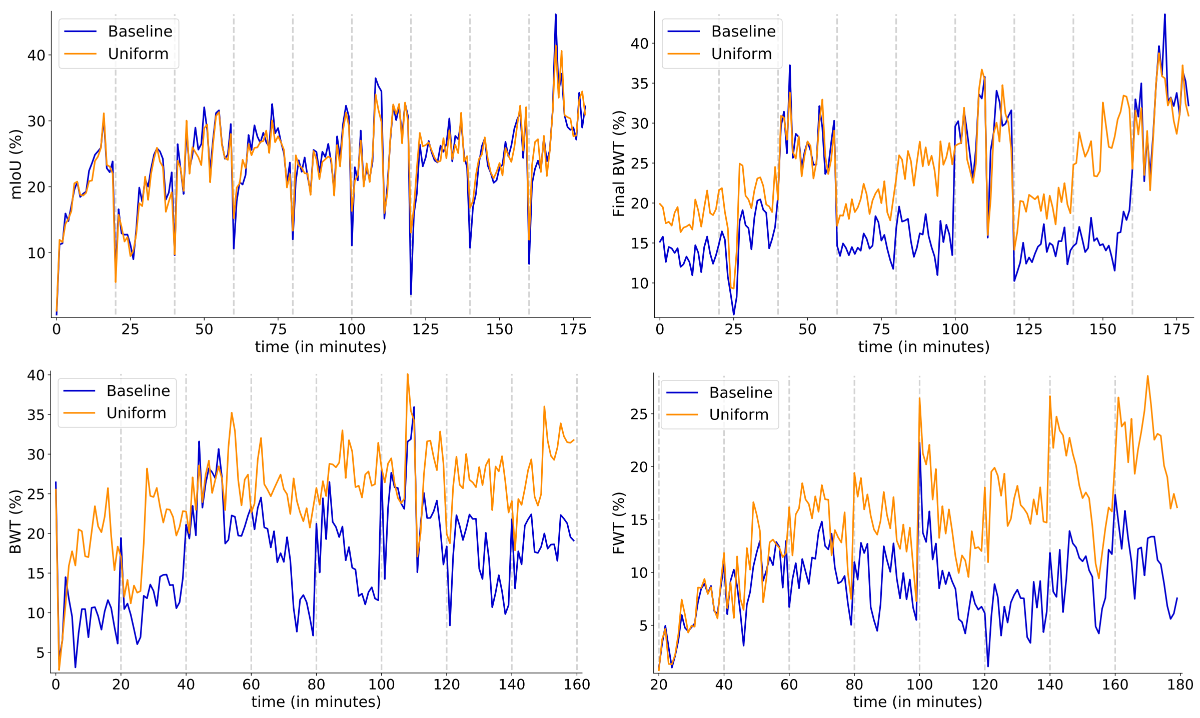


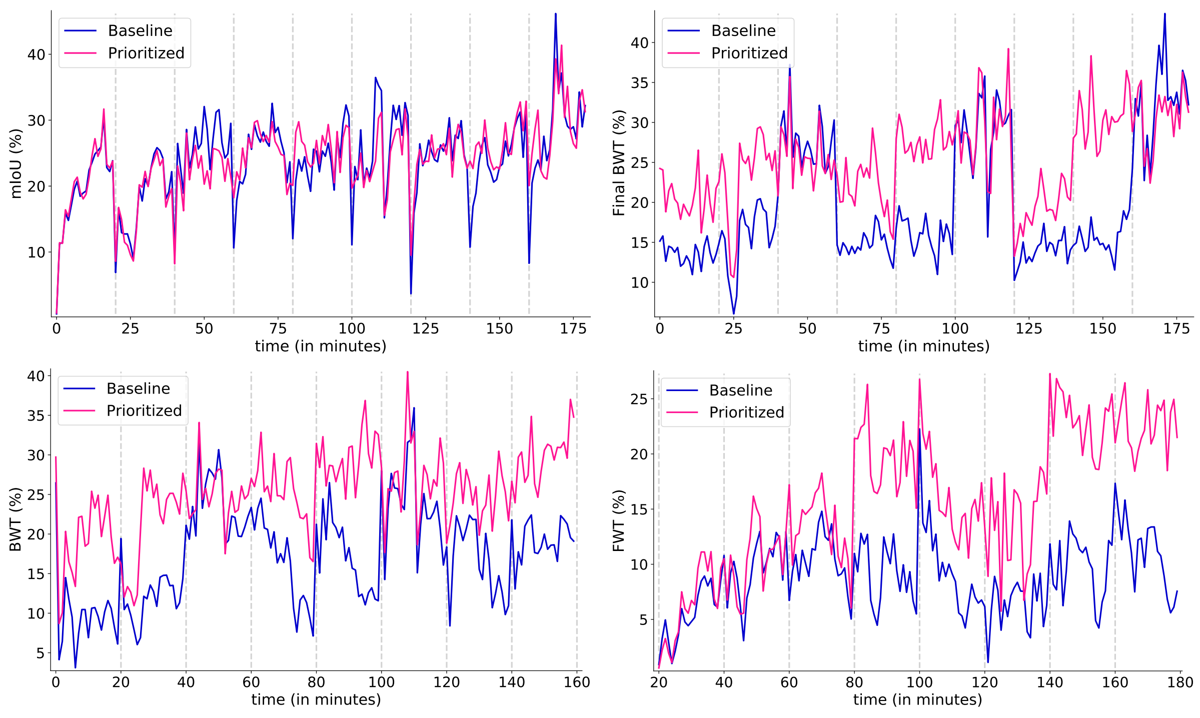


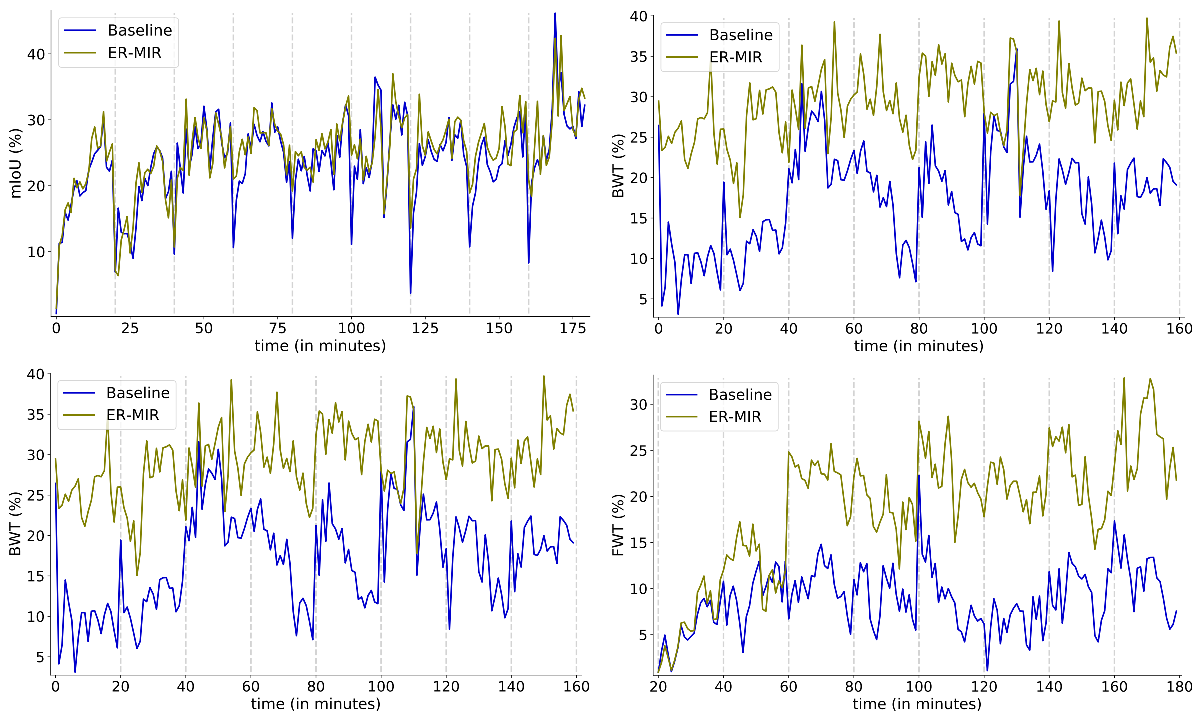


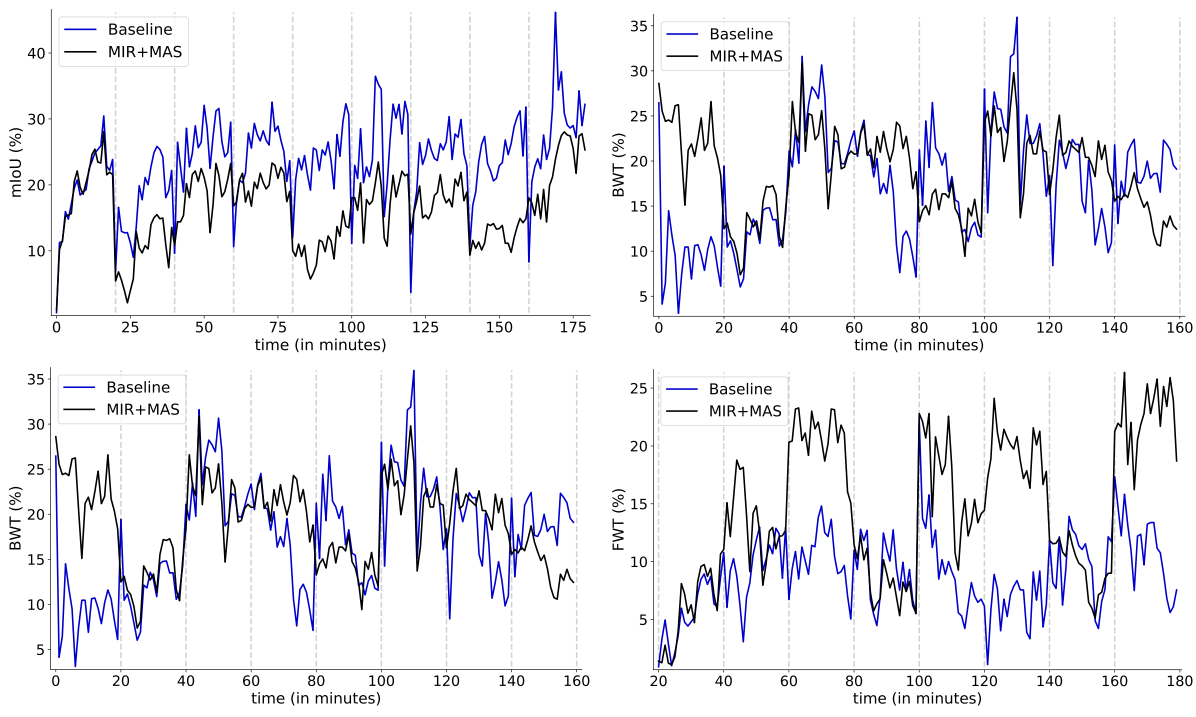

Baseline
LWF Memoryless
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)

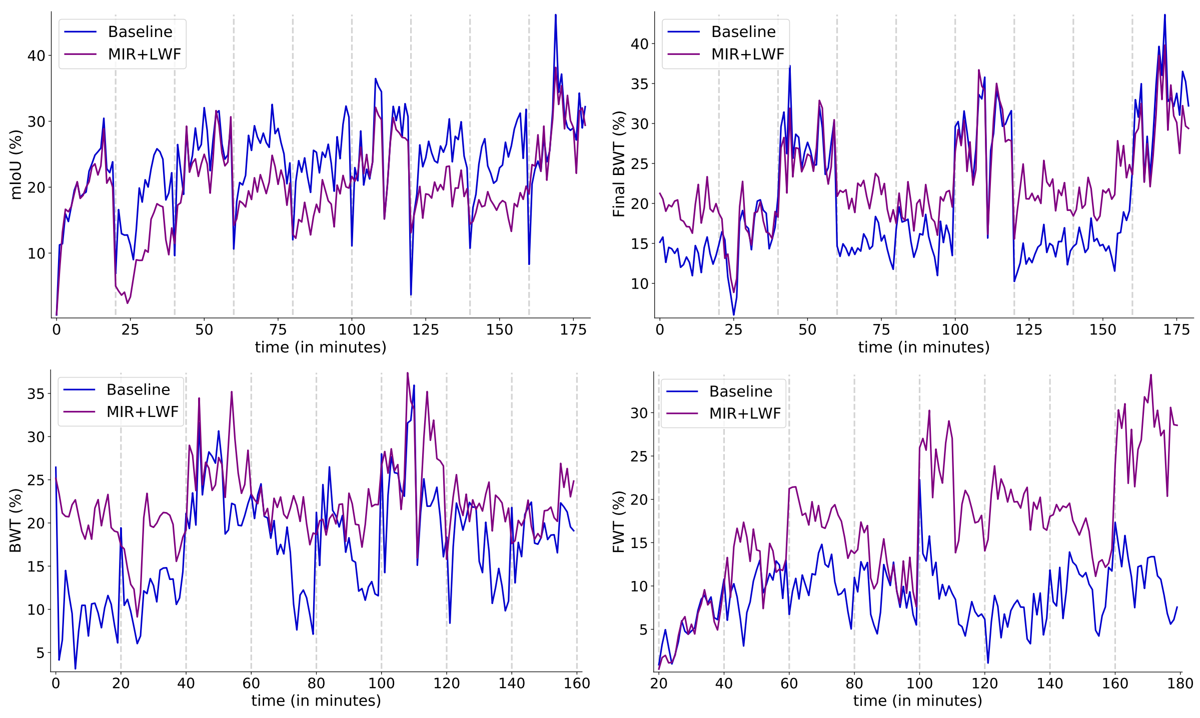

Baseline
RWalk Memoryless
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)

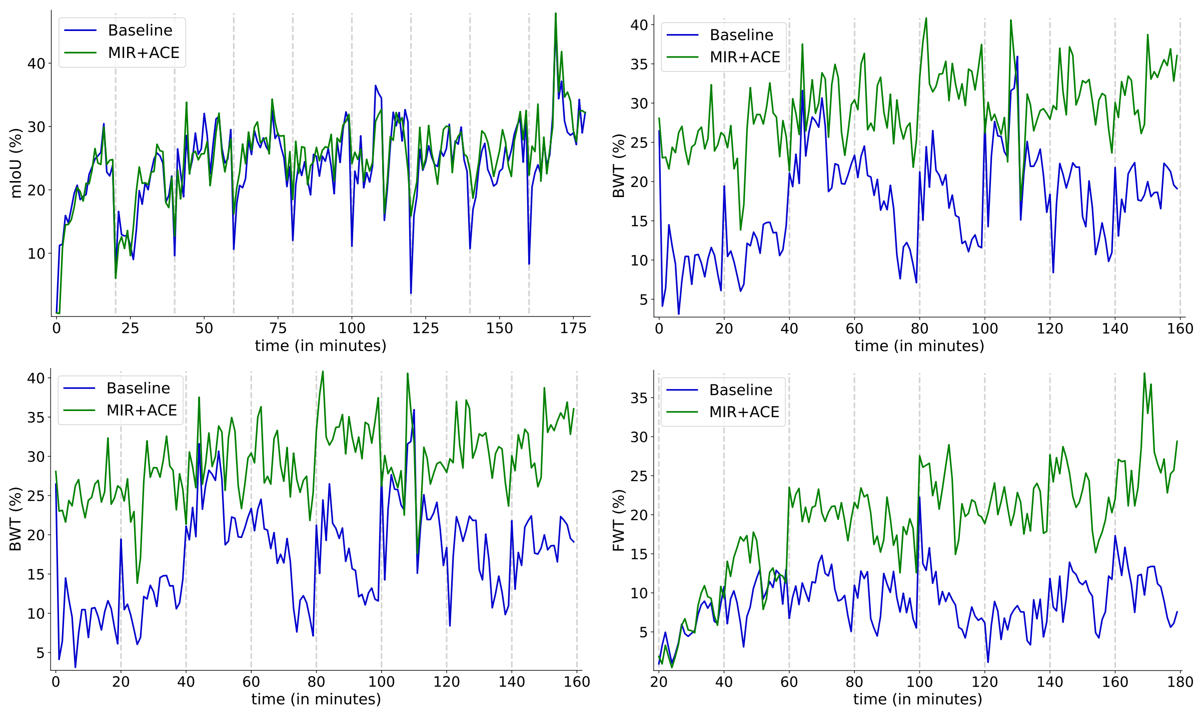

Baseline
Uniform
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)

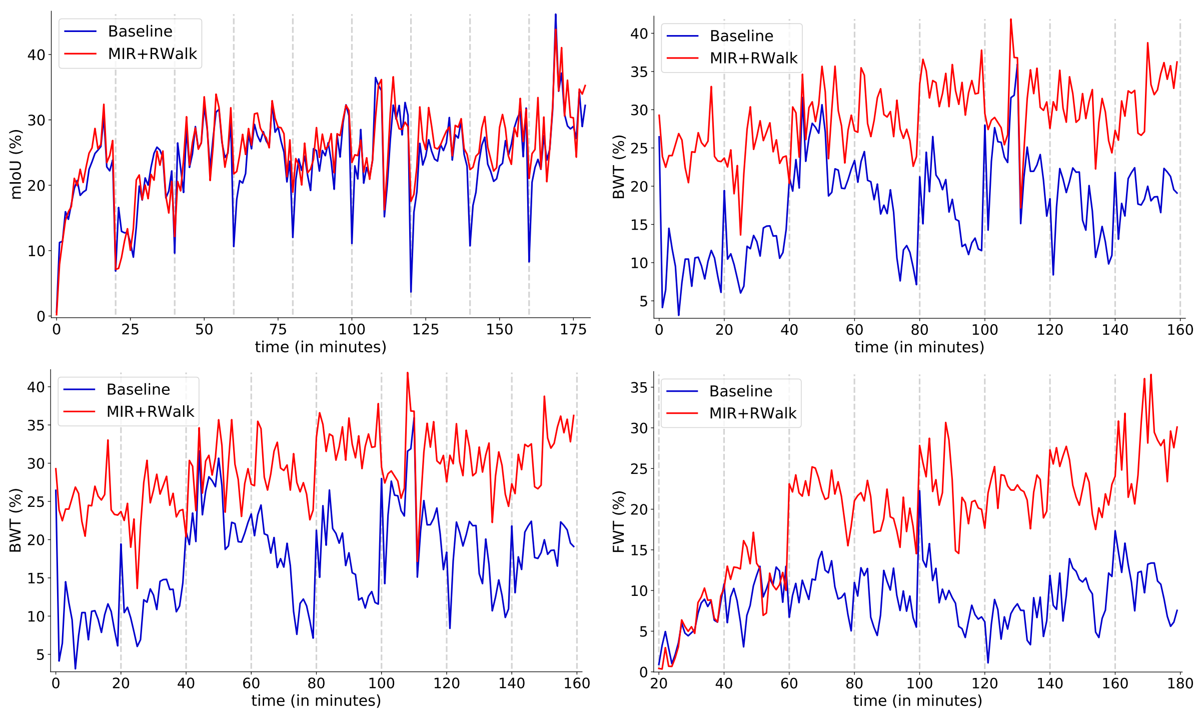

Baseline
Prioritized
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)


Baseline
ER-MIR
mIoU (%)
BWT (%)
FWT (%)
time (in minutes)


Baseline
MIR+MAS
mIoU (%)
BWT (%)
FWT (%)
time (in minutes)


Baseline
MIR+LWF
mIoU (%)
Final BWT (%)
BWT (%)
FWT (%)
time (in minutes)


Baseline
MIR+ACE
mIoU (%)
BWT (%)
FWT (%)
time (in minutes)


Baseline
MIR+RWalk
mIoU (%)
time (in minutes)
BWT (%)
time (in minutes)
BWT (%)
time (in minutes)
FWT (%)
time (in minutes)