To Adapt or Not to Adapt?
Real-Time Adaptation for Semantic Segmentation

Marc Botet Colomer* [1,2] Pier Luigi Dovesi* [3 †]
Theodoros Panagiotakopoulos[4] Joao Frederico Carvalho[1] Linus Härenstam-Nielsen[5,6]
Hossein Azizpour[2] Hedvig Kjellström[2,3] Daniel Cremers[5,6,7] Matteo Poggi[8]

[1]Univrses [2]KTH [3]Silo AI [4]King [5]Technical University of Munich
[6]Munich Center for Machine Learning [7]University of Oxford [8]University of Bologna

https://marcbotet.github.io/hamlet-web/

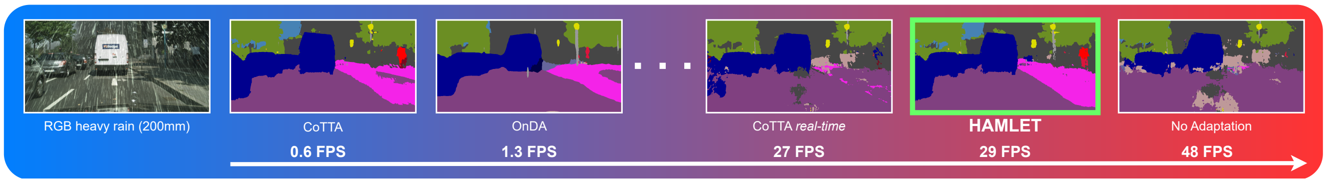


Figure 1. **Real-time adaptation with HAMLET.** Online adaptation to continuous and unforeseeable domain shifts is hard and computationally expensive. HAMLET can deal with it at almost 30FPS outperforming much slower online methods – e.g. OnDA and CoTTA.

Abstract

The goal of Online Domain Adaptation for semantic segmentation is to handle unforeseeable domain changes that occur during deployment, like sudden weather events. However, the high computational costs associated with brute-force adaptation make this paradigm unfeasible for real-world applications. In this paper we propose HAMLET, a Hardware-Aware Modular Least Expensive Training framework for real-time domain adaptation. Our approach includes a hardware-aware back-propagation orchestration agent (HAMT) and a dedicated domain-shift detector that enables active control over when and how the model is adapted (LT). Thanks to these advancements, our approach is capable of performing semantic segmentation while simultaneously adapting at more than 29FPS on a single consumer-grade GPU. Our framework's encouraging accuracy and speed trade-off is demonstrated on OnDA and SHIFT benchmarks through experimental results.

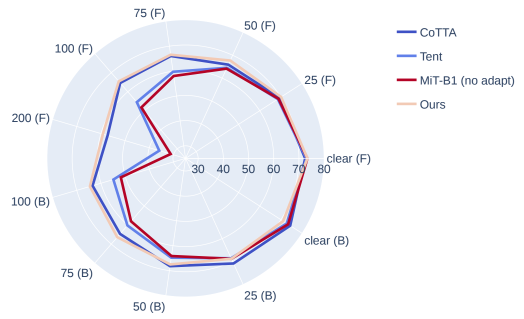


Figure 2: **Online adaptation methods on the Increasing Storm.** We plot mIoUs achieved on single domains. Colors from colder to warmer encode slower to faster methods.

1. Introduction

Semantic segmentation aims at classifying an image at a pixel level, based on the local and global context, to enable a higher level of understanding of the depicted scene. In recent years, deep learning has become the dominant paradigm to tackle this task effectively employing CNNs [5, 69, 4] or, more recently, transformers [65], at the expense of requiring large quantities of annotated images for training. Specifically, annotating for this task needs per-pixel labeling, which is an expensive and time-consuming task, severely limiting the availability of training data.

The use of simulations and graphics engines [42] to generate annotated frames enabled a marked decrease in the time and cost necessary to gather labeled data thanks

* Joint first authorship † Part of the work done while at Univrses

to the availability of the ground truth. However, despite the increasing quality in data realism [47], there is a substantial difference between simulated data generated by graphics engines and real-world images, such that leveraging these data for real-world applications requires adapting over a significant domain shift. The promise of unlocking this cheap and plentiful source of training data has provided a major impulse behind the development of a large body of work on Unsupervised Domain Adaptation (UDA) techniques [74, 61, 18, 15, 55], consisting of training semantic segmentation networks on labelled synthetic frames – the *source* domain – and then adapting the network to operate on real images, representing the *target* domain, without requiring human annotation. However, the synthetic-to-real shift represents only one of many possible domain transitions; specifically, when dealing with real-world deployment, domain shifts can occur from various causes, from different camera placements to different lighting, weather conditions, urban scenario, or any possible combination of the above. Because of the combinatorial nature of the problem, it is simply impossible to evenly represent all possible deployment domains in a dataset. This *curse of dimensionality* prevents having generalized robust perfomances [41, 45]. However, the recent advent of *online* domain adaptation [41] potentially allows us to face continuous and unpredictable domain shifts at deployment time, without requiring data associated with such domain shifts beforehand. Nonetheless, despite its potential, several severe limitations still hamper the online adaptation paradigm. In particular, continuously performing back-propagation on a frame-by-frame schedule [41] incurs a high computational cost, which negatively affects the performance of the network, dropping its overall framerate to accommodate the need for continuous adaptation. Various factors are involved in this matter: first, the severity of this overhead is proportional to the complexity of the network itself – the larger the number of parameters, the heavier the adaptation process becomes; second, we argue that frame-by-frame optimization is an excessive process for the adaptation itself – not only the network might need much fewer optimization steps to effectively counter domain shifts, but also such an intense adaptation definitely increases the likelihood of catastrophic forgetting over previous domains [26, 45]. In summary, a practical solution for online domain adaptation in semantic segmentation that can effectively operate in real-world environments and applications still seems to be a distant goal.

In this paper, we propose a novel framework aimed at overcoming these issues and thus allowing for real-time, online domain adaptation:

- We address the problem of online training by designing an automatic lightweight mechanism capable of significantly reducing back-propagation complexity. We exploit the model modularity to automatically choose to train the network subset which yields the highest improvement for the allocated optimisation time. This approach reduces back-propagation FLOPS by 34% while minimizing the impact on accuracy.
- In an orthogonal fashion to the previous contribution, we introduce a lightweight domain detector. This allows us to design principled strategies to activate training only when it really matters as well as setting hyperparameters to maximize adaptation speed. Overall, these strategies increase our speed by over $5\times$ while sacrificing less than 2.6% in mIoU.
- We evaluate our method on multiple online domain adaptation benchmarks both fully synthetic [45] and semi-synthetic CityScapes domain sequences [41], showing superior accuracy and speed compared to other test-time adaptation strategies.

Fig. 1 demonstrates the superior real-time adaptation performance of HAMLET compared to slower methods such as CoTTA [57], which experience significant drops in performance when forced to maintain a similar framerate by adapting only once every 50 frames. In contrast, HAMLET achieves an impressive 29 FPS while maintaining high accuracy. Additionally, Fig. 2 offers a glimpse of HAMLET's performance on the Increasing Storm benchmark [41], further highlighting its favorable accuracy-speed trade-off.

2. Related Work

We review the literature relevant to our work, about semantic segmentation and UDA, with particular attention to continuous and online methodologies.

Semantic Segmentation. Very much like classification, deep learning plays a fundamental role in semantic segmentation. Fully Convolutional Network (FCN) [36] represents the pivotal step in this field, adapting common networks by means of learned upsample operators (deconvolutions). Several works aimed at improving FCN both in terms of speed [68, 38] and accuracy [5, 6, 7], with a large body of literature focusing on the latter. Major improvements have been achieved by enlarging the receptive field [72, 66, 5, 6, 7], introducing refinement modules [14, 73, 17], exploiting boundary cues [3, 10, 46] or using attention mechanisms in different flavors [13, 31, 58, 64]. The recent spread of Transformers in computer vision [11] reached semantic segmentation as well [64, 69, 65], with SegFormer [65] representing the state-of-the-art in the field and being the object of studies in the domain adaptation literature as well [20].

Unsupervised Domain Adaptation (UDA). This body of research aims at adapting a network trained on a *source*, labeled domain to a *target*, unlabeled one. Early approaches rely on the notion of "style" and learn how to transfer it across domains [74, 61, 18, 32, 12, 67]. Common strategies consist of learning domain-invariant features [15, 25], often using adversarial learning in the process [15, 55, 8, 19, 51]. A popular trend in UDA is *Self-Training*. These methods rely on self-supervision to learn from unlabelled data. In UDA, a successful strategy consists of leveraging target-curated pseudo-labels. Popular approaches for this purpose make use of confidence [77, 37, 76], try to balance the class predictions [75, 20], or use prototypes [2, 71, 70] to improve the quality of the pseudo-labels. Among many domain shifts, the synthetic-to-real one is the most studied, since the earliest works [74, 61, 18] to the latest [60, 30, 21, 28, 16, 40, 24]. However, this shift is one of a kind since it occurs only once after training, and without the requirement of avoiding forgetting the source domain.

Continuous/Test-Time UDA. This family of approaches marries UDA with continuous learning, thus dealing with the *catastrophic forgetting* issue ignored in the synthetic-to-real case. Most *continuous UDA* approaches deal with it by introducing a Replay Buffer [1, 29, 27], while additional strategies make use of style transfer [62], contrastive [44, 53] or adversarial learning [63]. Despite the definition, continuous UDA often deals with *offline* adaptation, with well-defined target domains over which to adapt. Conceptually similar to it, is the branch of *test-time adaptation*, or *source-free* UDA, although tackling the problem in deployment rather than offline – *i.e.* with no access to the data from the source domain [43]. Popular strategies to deal with it consist of generating pseudo-source data to avoid forgetting [35], freezing the final layers in the model [33], aligning features [34], batch norm retraining through entropy minimization [54] or prototypes adaptation [22].

Online UDA. Although similar in principle to test-time adaptation, online UDA [45, 41, 52] aims to tackle multiple domain shifts, occurring unpredictably during deployment in real applications and without clear boundaries between them. On this track, the SHIFT dataset [45] provides a synthetic benchmark specifically thought for this scenario, while OASIS [52] proposes a novel protocol to evaluate UDA approaches, considering an online setting and constraining the evaluated methods to deal with frame-by-frame sequences. As for methods, OnDA [41] implements self-training as the orchestration of a static and a dynamic teacher to achieve effective online adaptation while avoiding forgetting, yet introducing massive overhead.

Real-time performance is an essential aspect of online adaptation, particularly in applications such as autonomous driving where slow models are impractical. A slow adaptation process not only limits the practicality of real-world applications but also fails to provide high accuracy until the adaptation is complete, thereby defeating the original purpose. Therefore, accelerating the adaptation process is crucial for achieving high accuracy in real-time scenarios.

3. Methods

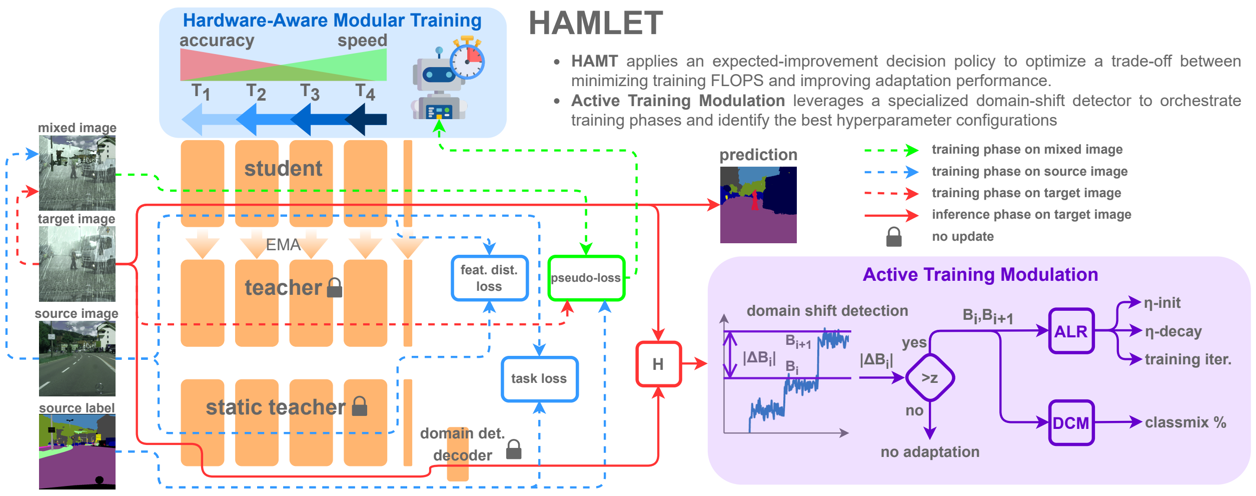
This section introduces HAMLET, a framework for **H**ardware-**A**ware **M**odular **L**east **E**xpensive **T**raining. The framework aims to solve the problem of online domain adaptation with real-time performance through several synergistic strategies. First, we introduce a Hardware-Aware Modular Training (HAMT) agent able to optimize online a trade-off between model accuracy and adaptation time. HAMT allows us to significantly reduce online training time and GFLOPS. Nevertheless, the cheapest training consists of no training at all. Therefore, as the second strategy, we introduce a formal geometric model for online domain shifts that enable reliable domain shift detection and domain estimator signals (Adaptive Domain Detection, Sec. 3.3.1). These can be easily integrated to activate the adaptation process only at specific times, *as least as possible*. Moreover, we can further leverage these signals by designing adaptive training policies that dynamically adapt domain-sensitive hyperparameters. We refer to these as Active Training Modulations. We present an overview of HAMLET in Fig. 3.

3.1. Model Setup

Our approach builds on the recent progress in unsupervised domain adaptation and segmentation networks. We start with DAFormer [20], a state-of-the-art UDA method, and adopt SegFormer [65] as our segmentation backbone due to its strong generalization capacity. We use three instances of the backbone, all pre-trained on the source domain: a student, a teacher, and a static (*i.e.* frozen) teacher. During training, the student receives a mix of target and source images [49] and is supervised with a "mixed-sample" cross-entropy loss, $\mathcal{L}_T$ (represented by green, blue and red dashed lines, in Fig. 3). This loss is computed by mixing the teacher's pseudo-labels and source annotations. To improve training stability, the teacher is updated as the exponential moving average (EMA) of the student. To further regularize the student, we use source samples stored in a replay buffer and apply two additional losses (blue lines in Fig. 3). First, we minimize the feature distance (Euclidean) between the student and the static teacher's encoder, $\mathcal{L}_{FD}$. Then, we employ a supervised cross-entropy task loss $\mathcal{L}_S$. Our complete objective is $\mathcal{L} = \mathcal{L}_S + \mathcal{L}_T + \lambda_{FD}\mathcal{L}_{FD}$, with λ_{FD} being a weight factor. During inference on the target domain, only the student is used (red lines in Fig. 3).

3.2. Hardware-Aware Modular Training (HAMT)

Online adaptation requires updating the parameters during deployment time. However, back-propagation is computationally expensive and hence too slow to be continuously applied on a deployed agent. Opting for a partial weight update, for example by finetuning the last module of the network, would enable much more efficient training time. However, domain shifts can manifest as changes



Hardware-Aware Modular Training
accuracy
speed
T1
T2
T3
T4
HAMLET
HAMT applies an expected-improvement decision policy to optimize a trade-off between minimizing training FLOPS and improving adaptation performance.
Active Training Modulation leverages a specialized domain-shift detector to orchestrate training phases and identify the best hyperparameter configurations
mixed image
target image
source image
source label
student
EMA
teacher
static teacher
feat. dist. loss
pseudo-loss
task loss
domain det. decoder
prediction
training phase on mixed image
training phase on source image
training phase on target image
inference phase on target image
no update
H
Active Training Modulation
domain shift detection
Bi+1
|ΔBi|
Bi
Bi,Bi+1
yes
>z
no
no adaptation
ALR
η-init
η-decay
training iter.
DCM
classmix %

Figure 3: **HAMLET framework.** We employ a student-teacher model with an EMA and a static teacher. HAMT orchestrates the back-propagation over the student restricting it to a network subsection. The Active Training Modulation instead controls the adaptation process by selectively enabling it only when necessary as well as tweaking sensitive training parameters.

in both the data input distribution (such as attributes of the images, *e.g.* day/night) and the output distribution (*e.g.* class priors). This information could be encoded in different parts of the network, therefore just updating the very last segment might not suffice. This motivates the need for orchestrating the training process, to ensure sufficient training while minimizing the computational overhead. Inspired by reward-punishment [48] and reinforcement learning [56] policies, we introduce an orchestration agent in charge of deciding how deeply the network shall be fine-tuned through a trade-off between the pseudo-loss minimization rate and the computational time. In contrast to previous efficient back-propagation approaches [59, 23, 9], our model is pre-trained on the task and thus requires smaller updates to adapt. Let us start by modeling the problem. Our model backbone, f, is composed of four different modules: $f = m_4 \circ m_3 \circ m_2 \circ m_1$. This defines our action space $\mathcal{A} = \{\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3, \mathrm{T}_4\}$ where T_4 corresponds to training just the last module of the network, m_4, while T_3 the last two modules, *i.e.* $m_4 \circ m_3$, T_2 the last three, *i.e.* $m_4 \circ m_3 \circ m_2$, and T_1 the whole network f. We also define a continuous state space $\mathcal{S} = \{\mathrm{R}, \mathrm{V}\}$ where R is the second derivative of the EMA teacher pseudo-loss, l_t, over time, hence $\mathrm{R_t} = -\frac{\Delta^2 l}{(\Delta t)^2}$, computed in discrete form as $R_t = -(l_t - 2l_{t-1} + l_{t-2})$. V represents a cumulative vector with the same dimension as the action space $\mathcal{A}$, initialized at zero. Now we have everything in place to employ an expected-improvement based decision model. At each time-step t, action T_j is selected for $j = \operatorname{argmax} \mathrm{V}_t$. During training step t, $\mathrm{V}[j]$ is updated as:

$$\mathrm{V}[j]_{t+1} = \alpha R_t + (1-\alpha)\mathrm{V}[j]_t \qquad (1)$$

where α is a smoothing factor, e.g. 0.1. *i.e.* $\mathrm{V_t}$ hold a discrete exponential moving average of R_t. Therefore, our policy can be seen as a greedy module selection based on the highest expected loss improvement over its linear approximation. A notable drawback of this policy is that we will inevitably converge towards picking more rewarding, yet expensive, actions *i.e.* $\mathrm{T}_1, \mathrm{T}_2$ compared to more efficient but potentially less effective actions *i.e.* $\mathrm{T}_3, \mathrm{T}_4$. However, our goal is not to maximize $-\frac{\Delta^2 l}{(\Delta t)^2}$ where Δt is the number of updates, our goal is instead to maximize $-\frac{\Delta^2 l}{(\Delta \tau)^2}$ where $\Delta\tau$ is a real-time interval. Therefore, we have to introduce in the optimization policy some notion of the actual training cost of each action in $\mathcal{A}$ on the target device. To start with, we measure the training time associated with each action, obtaining $\omega_T = \{\omega_{T_1}, \omega_{T_2}, \omega_{T_3}, \omega_{T_4}\}$. With this we can compute the time-conditioning vector γ as

$$\gamma_j = \frac{e^{\frac{1}{\beta\omega_{T_j}}}}{\sum_{k=1}^{K} e^{\frac{1}{\beta\omega_{T_k}}}} \quad \text{for } j = 1, \dots, K \qquad (2)$$

where β is the softmax temperature, and K the number of actions, *i.e.* 4 in our model. We modify our update policy to favor less computationally expensive modules by scaling the updates with γ, replacing Eq. 1 with:

$$\mathrm{V}[j]_{t+1} = \begin{cases} \gamma_j \alpha R_t + (1-\alpha)\mathrm{V}[j]_t & \text{if } R_t \geq 0 \\ (1-\gamma_j)\alpha R_t + (1-\alpha)\mathrm{V}[j]_t & \text{if } R_t < 0 \end{cases} \qquad (3)$$

This policy makes it so that more expensive actions receive smaller rewards and larger punishments. Despite its simplicity, this leads to a significant reduction in FLOPS for an average back-propagation β, *i.e.* -30% with $\beta = 2.75$

or -43% with $\beta = 1$. We finally choose $\beta = 1.75$ to obtain a FLOPS reduction of -34%. Exhaustive ablations on HAMT are presented in the supplementary material.

3.3. Active Training Modulation

Continuous and test-time adaptation methods tackle online learning as a continuous and constant process carried out on the data stream. Nevertheless, this approach presents several shortcomings when it comes to real-world deployments. Performing adaptation when the deployment domain is unchanged does not lead to further performance improvements on the current domain; instead, it might cause significant forgetting on previous domains, hence hindering model generalization (we present evidence of this in the supplementary material). Even if mitigated by HAMT, online training remains a computationally expensive procedure, also due to several teachers' necessary forward passes. However, knowing when and what kind of adaptation is needed is not a trivial task. We tackle this by introducing an Adaptive Domain Detection mechanism, in Sec. 3.3.1, and then a set of strategies to reduce the training time while optimizing the learning rate accordingly, in Sec. 3.3.2.

3.3.1 Adaptive Domain Detection

A key element of an online adaptation system consists of acquiring awareness of the trajectory in the data distribution space, *i.e.* domains, traveled by the student model during deployment. We can model the problem by setting the trajectory origin in the source domain. With high dimensional data, the data distribution is not tractable, therefore the trajectory cannot be described in closed form. Recent work [41] introduced the notion of distance between the current deployed domain and source by approximating it with the confidence drop of a source pre-trained model. This approach heavily relies on the assumption that the pre-trained model is well-calibrated. While this might hold for domains close to source, the calibration quickly degrades in farther domains [45, 41]. This myopic behavior dampen the simple use of confidence for domain detection. Furthermore, the additional forward pass increases the computational cost during deployment. We tackle these limitations with an equivalently simple, yet more robust, approach. We modify the backbone of the static teacher f^{st} used for the feature distance loss $\mathcal{L}_{FD}$ by connecting a lightweight segmentation head, d_1^{st}, after the first encoder module m_1^{st}: $h_1^{\text{st}} = d_1^{\text{st}} \circ m_1^{\text{st}}$. This additional decoder, h_1^{st}, is trained offline, on source data, without propagating gradients in the backbone (m_1^{st} is frozen). Given a target sample x_T, we propose to compute the cross-entropy between the one-hot encoded student prediction $p(x_T) = 1_{\text{argmax}(f(x_T))}$ and the lightweight decoder prediction $g(x_T) = h_1^{\text{st}}(x_T)$ as

$$H_T^{(i)} = -\sum_{p=1}^{H\times W}\sum_{c=1}^{C} p\left(x_T^{(i)}\right)\log g\left(x_T^{(i)}\right)\Big|_{p,c} \quad (4)$$

Thanks to the student model's higher generalization capability (both due to a larger number of parameters and the unsupervised adaptation process), it will always outperform the lightweight decoder head. Nevertheless, since now the distance is measured in the prediction space, we are not subjected to model miscalibration. Furthermore, since the student model is in constant adaptation, the domain distance accuracy actually improves over time, leading to better results. We present evidence of these claims in the supplementary material. We now define a denoised signal by using bin-averaging $A_T^{(i)} = \sum_{j=mi}^{m(i+1)-1} \frac{H_T^{(j)}}{m}$ where m is the bin size. Domains are modeled as discrete steps of $A_T^{(i)}$

$$B_0 = A_0 \qquad B_i = \begin{cases} A_i & \text{if } |B_{i-1} - A_i| > z \\ B_{i-1} & \text{otherwise} \end{cases} \quad (5)$$

where B is the discretized signal and z is the minimum distance used to identify new domains. Finally, we refer to the signed amplitude of domain shifts as $\Delta B_i = B_i - B_{i-1}$, and a domain change is detected whenever $|\Delta B_i| > z$.

3.3.2 Least Training and Adaptive Learning Rate

The definitions of B allow us to customize the training process. To this end, we adopt a *Least Training* (LT) strategy and trigger adaptation only when facing a new domain, which occurs when $|\Delta B_i| > z$. Effective online learning performance depends heavily on the choice of hyperparameters such as the learning rate η and learning rate decay rate. Therefore, we can adjust these parameters to facilitate adaptation according to the nature and intensity of domain shifts we encounter, we refer to this orchestration as Adaptive Learning Rate (ALR). For example, the larger the domain shift (*i.e.* $|\Delta B_i|$), the more we need to adapt to counteract its effect. This can be achieved by either running more optimization steps or using a higher learning rate. Whenever a domain shift is detected, we compute the number of adaptation iterations $L = K_l \frac{|\Delta B_i|}{z}$, hence proportionally to the amplitude of the shift $|\Delta B_i|$ relative to the threshold z. K_l is a multiplicative factor representing the minimum adaptation iterations. If a new domain shift takes place before the adaptation process completes, we accumulate the required optimization steps. Then, we can play on two further parameters: K_l and the learning rate schedule. We argue that proper scheduling is crucial for attaining a smoother adaptation. The learning rate, η, is linearly decayed until the adaptation is concluded – the smaller the domain shift, the faster the decay. While the initial learning rate, K_η, should

	HAMT	LT	ALR	DCM	RCS	200mm (mIoU)	All-domains (mIoU)	FPS	Average GFLOPS Total	Average GFLOPS Fwd.	Average GFLOPS Bwd.	Adaptation GFLOPS Fwd.	Adaptation GFLOPS Bwd.
(A)	–	–	–	–	–	62.2 ± 0.9	69.5 ± 0.3	5.9 ± 0.0	125.2 ± 0.0	94.4 ± 0.0	30.8 ± 0.0	56.6 ± 0.0	30.8 ± 0.0
(B)	✓	–	–	–	–	60.2 ± 0.5	68.7 ± 0.3	7.0 ± 0.1	114.7 ± 0.0	94.4 ± 0.0	20.3 ± 0.0	56.6 ± 0.0	20.3 ± 0.0
(C)	✓	✓	–	–	–	51.8 ± 0.5	65.7 ± 0.2	29.5 ± 0.6	44.4 ± 0.5	42.6 ± 0.4	1.8 ± 0.2	56.6 ± 0.0	20.2 ± 0.2
(D)	✓	✓	✓	–	–	54.1 ± 1.2	65.9 ± 0.2	29.5 ± 0.5	44.4 ± 0.3	42.7 ± 0.2	1.8 ± 0.1	56.6 ± 0.0	20.3 ± 0.1
(E)	✓	✓	✓	✓	–	56.6 ± 0.8	66.3 ± 0.1	28.9 ± 0.3	44.7 ± 0.2	42.9 ± 0.2	1.8 ± 0.1	56.6 ± 0.0	20.2 ± 0.0
(F)	✓	✓	✓	–	✓	55.8 ± 1.0	66.3 ± 0.2	29.1 ± 1.1	45.2 ± 0.1	43.2 ± 0.1	2.0 ± 0.0	56.6 ± 0.0	20.3 ± 0.0
(G)	✓	✓	✓	✓	✓	58.2 ± 0.8	66.9 ± 0.3	29.7 ± 0.6	45.7 ± 0.3	43.6 ± 0.2	2.1 ± 0.1	56.6 ± 0.0	20.2 ± 0.1

(a)

	clear 1	200mm	clear 2	100mm	clear 3	75mm	clear 4	clear h-mean	target h-mean	total h-mean	FPS	GFLOPS
(A)	72.9	52.2	73.6	64.2	73.0	67.6	73.4	73.2	60.6	67.2	5.6	125.2
(B)	73.0	50.4	73.4	62.1	73.0	67.3	73.2	73.1	59.1	66.4	6.8	114.7
(C)	73.4	46.0	73.5	61.5	73.6	66.1	73.8	73.6	56.5	65.1	7.2	100.0
(G)	73.4	53.6	73.1	65.2	73.5	68.2	73.2	73.3	61.6	67.8	9.1	82.2

(b)

Table 1: **Ablation studies – HAMLET components.** Top: Increasing Storm (8925 frames per domain) [41], bottom: Fast Storm C [41] (2975 frames per domain). For each configuration, we report mIoU, framerate, and GFLOPS.

be higher when the domain shift is triggered in domains farther from the source

$$K_\eta = K_{\eta,\text{min}} + \frac{(B_\text{i} - B_\text{source})(K_{\eta,\text{max}} - K_{\eta,\text{min}})}{B_\text{hard} - B_\text{source}} \tag{6}$$

where B_source (resp. B_hard) is an estimate of B when the network is close to (resp. far from) the source domain; and $K_{\eta,\text{min}}$ (resp. $K_{\eta,\text{max}}$) is the value of K_η assigned when the network is close to (resp. far away from) the source. Concerning K_l, we posit that moving towards the source requires less adaptation than going towards harder domains: the model shows good recalling of previously explored domains and thanks to the employed regularization strategies

$$K_l = \begin{cases} K_{l,\text{max}} & \text{if } \Delta B_i \geq 0 \\ K_{\text{l,min}} + \frac{(B_\text{i} - B_\text{source})(K_{\text{l,max}} - K_{\text{l,min}})}{B_\text{hard} - B_\text{source}} & \text{otherwise} \end{cases} \tag{7}$$

where $K_{l,\text{min}}$ (resp. $K_{l,\text{max}}$) is the value of K_l assigned when the model is close to (resp. far away from) the source domain. Extensive ablations in the supplementary material will highlight how the orchestration of the adaptation hyperparameters improves the accuracy-speed trade-off.

3.3.3 Dynamic ClassMix (DCM)

ClassMix [39] provides a simple mechanism for data augmentation by mixing classes from the source dataset into target images. Usually 50% of the classes in the source dataset are selected, however we notice that this percentage is a highly sensitive hyperparameter in online domain adaptation. Injecting a significant portion of source classes has a beneficial impact when adapting to domains closer to the source domain, whereas when adapting to domains further from the source the opposite effect can be observed, as it effectively slows down the adaptation process. We therefore exploit once more the deployment domain awareness to control the mixing augmentation:

$$K_\text{CM} = K_\text{CM,min} + \frac{(B_\text{i} - B_\text{source})(K_\text{CM,max} - K_\text{CM,min})}{B_\text{hard} - B_\text{source}}. \tag{8}$$

where K_CM is the percentage of source classes used during adaptation; and $K_\text{CM, min}$ (resp. $K_\text{CM, max}$) is the value of K_CM assigned when the network is close to (resp. far away from) the source domain.

3.3.4 Buffer Sampling

Following [41], to simulate real deployment, we limit our access to the source domain by using a replay buffer. Additionally, instead of initializing at random (with a uniform prior), we apply Rare Class Sampling (RCS) (skewed priors) as in [20]. This incentives a more balanced class distribution over the buffer, ultimately leading to better accuracy.

4. Experimental Results

The experiments are carried out on (a) the OnDA benchmarks [41] and (b) the SHIFT dataset [45]. (a) is a semi-syntehtic benchmark, as it applies synthetic rain and fog [50] over 4 different intensities profiles. The main benchmark, Increasing Storm, presents a storm with a pyramidal intensity profile; see Fig. 4. In contrast, (b) is a purely synthetic dataset, where both the underlying image and the weather are synthetically generated and thus domain change is fully controllable. All models are evaluated using mIoU: following [41], we report the harmonic mean over domains to present the overall adaptation performance. All experiments were carried out using an Nvidia™ RTX 3090 GPU. We refer to supplementary material for further details.

4.1. Ablation Studies

In Tab. 1 we study the impact of each contribution to adaptation performance, both in terms of accuracy and efficiency. For each configuration, we report mIoU over different portions of the sequence, the framerate and the amount of GFLOPS – respectively averages of: total, forward and backward passes, and dedicated adaptation only, also divided in forward (Fwd) and backward (Bwd). Tab. 1 (a) shows results on the Increasing Storm scenario [41]. Here, we show mIoU over the 200mm domain, *i.e.* the hardest in the sequence, as well as the mIoU averaged over forward and backward adaptation, *i.e.*, from *clear* to 200mm rain

		clear		25mm		50mm		75mm		100mm		200mm	h-mean			FPS	GFLOPS
		F	B	F	B	F	B	F	B	F	B	F	F	B	T		
(A)	DeepLabV2 (no adaptation)	64.5	–	57.1	–	48.7	–	41.5	–	34.4	–	18.5	37.3	–	–	39.4	–
(B)	DeepLabV2 fully supervised (oracle)	64.5	–	64.1	–	63.7	–	63.0	–	62.4	–	58.2	62.6	–	–	39.4	–
(C)	OnDA	64.5	64.8	60.4	57.1	57.3	54.5	54.8	52.2	52.0	49.1	42.2	54.2	55.1	–	1.3	–
(D)	SegFormer MiT-B1 (no adaptation)	73.4	–	68.8	–	64.2	–	58.0	–	51.8	–	31.2	57.8	–	–	48.4	34.9
(E)	SegFormer MiT-B5 (no adaptation)	77.6	–	73.9	–	71.0	–	67.2	–	62.6	–	46.7	64.7	–	–	11.5	240.4
(F)	SegFormer MiT-B1 fully supervised (oracle)	72.9	–	72.4	–	72.1	–	71.5	–	70.7	–	68.6	71.3	–	–	48.4	34.9
(G)	TENT	73.0	72.8	68.5	68.6	64.5	64.8	59.7	60.2	54.5	54.8	35.9	56.2	63.6	59.9	10.0	–
(H)	TENT + Replay Buffer	73.0	72.8	68.5	68.6	64.5	64.8	59.7	60.2	54.4	54.7	35.8	56.1	63.6	59.9	7.8	–
(I)	CoTTA	72.5	74.4	69.5	**70.9**	65.9	**68.2**	66.1	64.7	64.6	63.5	57.2	65.6	**68.1**	66.8	0.6	593.8
(J)	CoTTA *real-time*	73.3	**75.4**	**70.3**	70.6	66.9	66.4	62.5	61.4	57.6	56.9	39.7	59.2	65.5	62.3	27.0	**41.7**
(K)	HAMLET (ours)	**73.4**	71.0	70.1	68.8	**67.7**	67.5	**66.6**	**66.4**	**65.5**	**64.6**	**59.2**	**66.8**	67.6	**67.2**	**29.1**	45.7

Table 2: **Comparison against other models – Increasing storm scenario.** (A-C) methods built over DeepLabv2, (D-E) SegFormer variants trained on source, (F) oracle, (G-K) models adapted online. We report mIoU, framerate, and GFLOPS.

and backward. Results are averaged over 3 runs with different seeds, with standard deviation being reported. (A) reports the results achieved by naïvely performing full adaptation of the model. HAMT can increase the framerate by roughly 15% by reducing the Bwd GFLOPS of 34%, at the expense of as few as 0.7 mIoU on average, *i.e.*, about 2 points on the 200mm domain. The main boost in terms of speed is obviously given by LT (C), which inhibits the training in absence of detected domain shifts. LT increases the framerate by approximately $4\times$ by decimating the total GFLOPS, yet not affecting the adaptation Bwd GFLOPS. This comes with a price in terms of mIoU, dropping by about 4 points on average and more than 10 points on 200mm – not a moderate drop anymore. LT impact highly depends on the domain sequence experienced during deployment: frequent domain changes could prevent training inhibition, thus neglecting LT gains in terms of efficiency, as we will appreciate later. The loss in accuracy is progressively regained by adding ALR (D), with further improvements yielded by one between DCM (E) and RCS (F), or both together (G) leading to the full HAMLET configuration. The three together allow for reducing the gap to 2.5 points mIoU – 4 over the 200mm domain – without sacrificing any efficiency. Tab. 1 (b) shows further results, on a faster version of Storm C [41]. This represents a much more challenging scenario, with harsher and $3\times$ more frequent domain shifts. Here we show the single domains mIoU, as well as harmonic mean on source and target domains, and all frames. As expected, in this benchmark, LT alone (C) results much less effective than before, with a much lower gain in FPS and GFLOPS. Here, the synergy between the HAMT, LT, and the other components (G) allows for the best accuracy and speedup – even outperforming the full training variant (A) – highlighting their complementarity. Further ablations are in the supplementary material.

4.2. Results on Increasing Storm

Tab. 2 shows a direct comparison between HAMLET and relevant approaches. The presented test-time adaptation

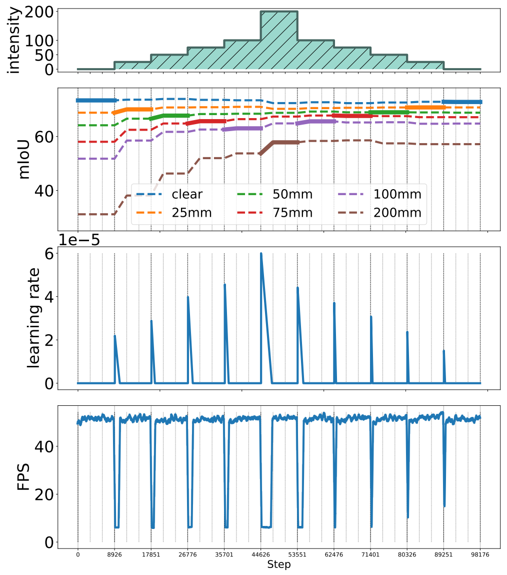


Figure 4: **HAMLET on the Increasing Storm.** We show rain intensity (in millimetres), mIoU over active (bold) and inactive (dashed) domains, learning rate and FPS.

strategies namely – TENT and CoTTA – were revised to handle the online setting and be fairly compared with HAMLET. All methods start with the same exact initial weights – with HAMLET requiring the additional lightweight decoder, not needed by TENT and CoTTA – using SegFormer MiT-B1 as the backbone, since it is $4\times$ faster than SegFormer MiT-B5 and thus better suited to keep real-time performance even during adaptation. We report results achieved by DeepLabv2 trained on source data only (A), an *oracle* model trained with full supervision (B), as well as OnDA [41] (C) as a reference. Then, we report SegFormer models trained on the source domain only (D) and (E). In (F) we show the performance achieved by an oracle SegFormer, trained on all domains fully supervised. Following [41], columns "F" concern forward adaptation from *clear* to 200mm, while columns "B" show backward adap-

	clear		750m		375m		150m		75m	h-mean			FPS	GFLOPS
	F	B	F	B	F	B	F	B	F	F	B	T		
OnDA	64.9	65.8	63.3	62.3	60.7	58.8	51.6	49.1	42.1	55.1	54.1	–	1.3	–
SegFormer MiT-B1 (no adaptation)	71.1	–	70.0	–	67.5	–	58.8	–	46.9	61.3	–	–	48.4	34.9
Full training	71.5	72.1	72.9	74.7	71.9	73.1	67.6	68.1	61.3	68.7	71.9	70.3	5.6	125.2
HAMLET (ours)	71.1	71.6	70.3	70.8	68.8	69.2	64.3	64.3	57.0	65.9	68.9	67.4	24.8	50.7

Table 3: **Results on foggy domains.** Comparison between OnDA, Source SegFormer, full training adaptation, and HAMLET.

	Clear		Cloudy		Overcast		Small rain		Mid rain		Heavy rain	h-mean			FPS	GFLOPS
	F	B	F	B	F	B	F	B	F	B	F	F	B	T		
SegFormer MiT-B1 fully supervised (oracle)	80.1	–	79.9	–	79.8	–	78.9	–	78.7	–	77.1	79.1	–	–	48.4	34.93
SegFormer MiT-B1 (no adaptation)	79.6	–	77.1	–	75.4	–	73.4	–	71.4	–	66.7	73.7	–	–	48.4	34.93
Full training	78.9	79.3	76.7	76.8	76.8	77.9	74.8	74.8	76.3	76.5	74.0	76.2	77.0	76.6	5.0	125.1
HAMLET (ours)	79.6	78.9	76.9	76.6	76.1	77.4	73.3	74.3	74.2	76.0	74.2	75.7	76.6	76.1	26.8	43.9

Table 4: **Results on SHIFT dataset [45].** Comparison between Source SegFormer, full training adaptation, and HAMLET.

tation from 200mm to *clear*, while the h-mean T refers to the overall harmonic mean. We can notice how SegFomer results are much more robust to domain changes with respect to DeepLabv2. Indeed, SegFormer MiT-B5 (E), without any adaptation, results more accurate than DeepLabv2 oracle (B), as well as better and faster than OnDA (C). The faster variant (D) outperforms OnDA both in speed and accuracy, reaching 48 FPS. Nevertheless, domain changes still dampen the full potential of SegFormer. Indeed, the oracle (F) outperforms (D) by about +14 mIoU. However, this is not meaningful for real deployment experiencing unpredictable domain shifts, as it assumes to have data available in advance. Concerning test-time models, TENT starts adapting properly only beyond 50mm, both with (G) and without (H) frame buffer, while it loses some accuracy on 25mm. This makes its overall forward adaptation performance slightly worse compared to the pre-trained model (D), while being better at backward adaptation. Despite outperforming SegFormer MiT-B1, TENT is both slower and less accurate than SegFormer MiT-B5 running without any adaptation, further suggesting the robustness of the latter and making TENT not suitable for real-world deployment. On the contrary, CoTTA (I) outperforms both SegFormer models trained on source only, at the expense of dropping the framerate below 1FPS. It is worth mentioning that these metrics were collected after each domain was completed by each model individually. In an evaluation setup imposing a shared time frame, slower models would present much lower metrics, since their adaptation process would result constantly lagged. In fact, forcing CoTTA to run in real-time, at nearly 30FPS – *i.e.* by training once every 50 frames – dramatically reduces the effectiveness of the adaptation process (J), with drastic drops in the hardest domains. Finally, HAMLET (K) succeeds on any fronts, improving the baseline (D) by about 10 points with only a cost of 25% in terms of speed, while outperforming SegFormer MiT-B5 (E) both on accuracy (+2.5 mIoU) and speed (3× faster) – being the only method achieving this, and thus the only suitable choice for real-time applications. Fig. 4 shows the overall behavior of HAMLET while adapting over the Increasing Storm. In addition to the rain intensity and the

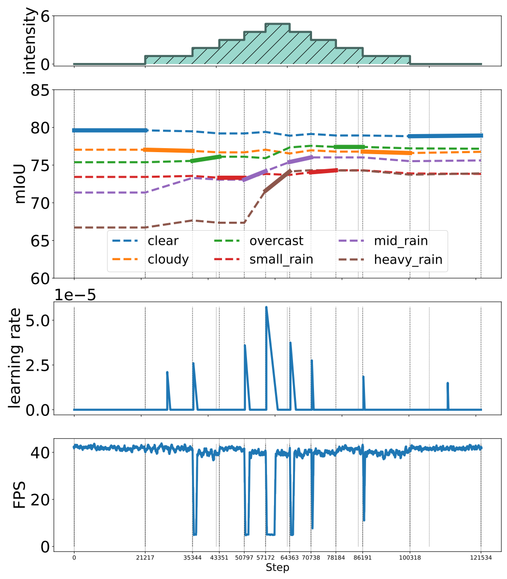


Figure 5: **HAMLET on the SHIFT benchmark.** We show mIoU over active (bold) and inactive (dashed) domains, learning rate and FPS.

mIoU achieved on each domain – active (bold) or inactive (dashed), *i.e.* respectively the mIoU on the domain being currently faced during deployment, and how the current adaptation affects the performance on the other domains to highlight the robustness to forgetting – we also report how the learning rate is modulated in correspondence of detected domain shifts, with a consequent drop in FPS due to the short training process taking place. For further experiments on harsher and sudden adaptation cycles, we include results of Storms A, B, C [41] in the supplementary material.

4.3. Additional Results: Fog and SHIFT

Fog. In Tab. 3, we investigate adaptation on the Increasing Fog scenario in the OnDA benchmark [41]. Crucially, for this experiment, we keep the same hyperparameters used for the Increasing Storm, since in both cases the starting SegFormer model is trained on the same source domain. This allows for validating how the proposed setting general-

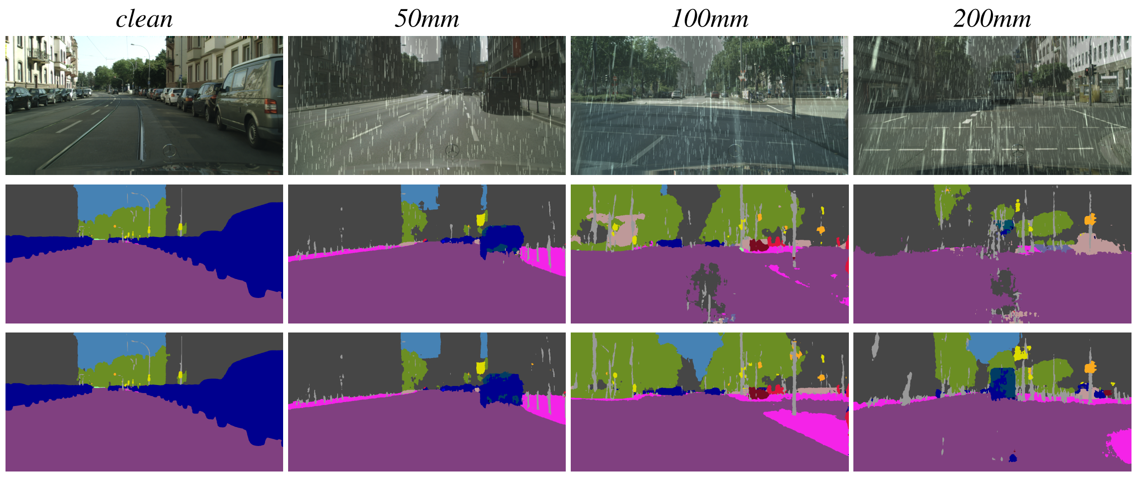


Figure 6: **Qualitative results – HAMLET in action.** From left to right, we show frames from *clean*, *50mm*, *100mm*, and *200m* domains. From top to bottom: input image, prediction by SegFormer trained on source domain and HAMLET.

izes at dealing with different kind of domain shifts, beyond those considered in the main experiments. We effectively use Increasing Fog as test set, and compare against SegFormer trained on source (no adaptation) and a model that has been adapted by means of full online training optimization (configuration (A) of Table 1). HAMLET is able to adapt almost as well as the full online training model, with less than a 3 mIoU gap, while enjoying real-time adaptation at nearly $5\times$ the speed using just 40% of the FLOPS.

SHIFT. We further test HAMLET on the SHIFT dataset [45]. Tab. 4 collects the results achieved by SegFormer trained on source, full online training and HAMLET respectively, both at forward and backward adaptation across *Clear*, *Cloudy*, *Overcast*, *Small rain*, *Mid rain* and *Heavy rain* domains. Here HAMLET results highly competitive with the full training regime, with only 0.5 drop in average mIoU, while being more than $5\times$ faster. Fig. 5 depicts, from top to bottom, the rain intensity characterizing any domain encountered on SHIFT, the mIoU achieved both on current (bold) and inactive (dashed) domains, the learning rate changes based on the domain shift detection, and the framerate achieved at any step. We refer to the supplementary material for a deeper analysis.

Qualitative results. To conclude, Fig. 6 shows some qualitative examples from CityScapes. We can notice how SegFormer accuracy (second tow) drops with severe rain, whereas HAMLET (third row) is capable of keeping the same segmentation quality across the storm.

5. Discussion

Orthogonality. HAMT and LT act independently. Indeed, by strongly constraining the adaptation periods through LT, HAMT has a limited margin of action. The impact of HAMT also depends on the backbone and by carefully crafting modular architectures, one can achieve further optimization. Nevertheless, in a deployment environment where domain shifts occur at high frequencies (e.g., Storm C), LT is ineffective, while HAMT thrives.

Measuring forgetting. An interesting topic we have not investigated consists of introducing an explicit awareness of which domains have been explored and how well we can recall them, expanding the distance B to multiple dimensions.

Safety. We believe dynamic adaptation has the potential to enhance safety, but we acknowledge the necessity for rigorous testing and verification to safeguard against drift or catastrophic forgetting. This mandates a comprehensive effort from academia, industry, and certification authorities for ensuring the integrity of dynamically adapting models.

6. Summary & Conclusion

We have presented HAMLET, a framework for real-time adaptation for semantic segmentation that achieves state-of-the-art performance on established benchmarks with continuous domain changes. Our approach combines a hardware-aware backpropagation orchestrator and a specialized domain-shift detector to enable active control over the model's adaptation, resulting in high framerates on a consumer-grade GPU. These advancements enable HAMLET to be a promising solution for in-the-wild deployment, making it a valuable tool for applications that require robust performance in the face of unforeseen domain changes.

Acknowledgement. The authors thank Gianluca Villani for the insightful discussion on reward-punishment policies, Leonardo Ravaglia for his expertise on hardware-aware training, and Lorenzo Andraghetti for exceptional technical support throughout the project. Their assistance was invaluable in the completion of this work.

References

[1] Andreea Bobu, Judy Hoffman, Eric Tzeng, and Trevor Darrell. Adapting to continuously shifting domains. In *ICLR 2018 Workshop Program Chairs*, 2018. 00000.

[2] Chaoqi Chen, Weiping Xie, Wenbing Huang, Yu Rong, Xinghao Ding, Yue Huang, Tingyang Xu, and Junzhou Huang. Progressive feature alignment for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 627–636, 2019.

[3] Liang-Chieh Chen, Jonathan T Barron, George Papandreou, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4545–4554, 2016.

[4] Liang-Chieh Chen, Raphael Gontijo Lopes, Bowen Cheng, Maxwell D Collins, Ekin D Cubuk, Barret Zoph, Hartwig Adam, and Jonathon Shlens. Naive-student: Leveraging semi-supervised learning in video sequences for urban scene segmentation. In *European Conference on Computer Vision*, pages 695–714. Springer, 2020.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, Apr 2018.

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[8] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2011–2020. IEEE, 2017. 00000.

[9] Feng Cheng, Mingze Xu, Yuanjun Xiong, Hao Chen, Xinyu Li, Wei Li, and Wei Xia. Stochastic backpropagation: A memory efficient strategy for training video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8301–8310, 2022.

[10] Henghui Ding, Xudong Jiang, Ai Qun Liu, Nadia Magnenat Thalmann, and Gang Wang. Boundary-aware feature propagation for scene segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6819–6829, 2019.

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

[12] A. Dundar, M. Y. Liu, Z. Yu, T. C. Wang, J. Zedlewski, and J. Kautz. Domain stylization: A fast covariance matching framework towards domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[13] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3146–3154, 2019.

[14] Jun Fu, Jing Liu, Yuhang Wang, Yong Li, Yongjun Bao, Jinhui Tang, and Hanqing Lu. Adaptive context network for scene parsing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6748–6757, 2019.

[15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, Jan. 2016.

[16] Rui Gong, Martin Danelljan, Dengxin Dai, Danda Pani Paudel, Ajad Chhatkuli, Fisher Yu, and Luc Van Gool. Tacs: Taxonomy adaptive cross-domain semantic segmentation. In *European Conference on Computer Vision*, pages 19–35. Springer, 2022.

[17] Junjun He, Zhongying Deng, Lei Zhou, Yali Wang, and Yu Qiao. Adaptive pyramid context network for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7519–7528, 2019.

[18] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[19] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. FCNs in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR*, 2016. 00000.

[20] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. *arXiv preprint arXiv:2111.14887*, 2021.

[21] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Hrda: Context-aware high-resolution domain-adaptive semantic segmentation. In *European Conference on Computer Vision (ECCV)*, 2022.

[22] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[23] Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi,

Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

[24] Zhengkai Jiang, Yuxi Li, Ceyuan Yang, Peng Gao, Yabiao Wang, Ying Tai, and Chengjie Wang. Prototypical contrast adaptation for domain adaptive semantic segmentation. In *European Conference on Computer Vision*, pages 36–54. Springer, 2022.

[25] Myeongjin Kim and Hyeran Byun. Learning texture invariant representation for domain adaptation of semantic segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020.

[26] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei 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. *Proceedings of the National Academy of Sciences*, 114, 12 2016.

[27] Yevhen Kuznietsov, Marc Proesmans, and Luc Van Gool. Towards unsupervised online domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 261–271, 2022.

[28] Xin Lai, Zhuotao Tian, Xiaogang Xu, Yingcong Chen, Shu Liu, Hengshuang Zhao, Liwei Wang, and Jiaya Jia. Decouplenet: Decoupled network for domain adaptive semantic segmentation. In *European Conference on Computer Vision*, pages 369–387. Springer, 2022.

[29] Qicheng Lao, Xiang Jiang, Mohammad Havaei, and Yoshua Bengio. Continuous domain adaptation with variational domain-agnostic feature replay. *arXiv preprint arXiv:2003.04382*, 2020.

[30] Geon Lee, Chanho Eom, Wonkyung Lee, Hyekang Park, and Bumsub Ham. Bi-directional contrastive learning for domain adaptive semantic segmentation. In *European Conference on Computer Vision*, pages 38–55. Springer, 2022.

[31] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9167–9176, 2019.

[32] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[33] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. *CoRR*, 2020.

[34] Yuejiang Liu, Parth Kothari, Bastien Germain van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When does self-supervised test-time training fail or thrive? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[35] Yuang Liu, Wei Zhang, and Jun Wang. Source-free domain adaptation for semantic segmentation. 2021.

[36] Jonathan Long, Evan Shelhamer, and Trevor Darrel. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[37] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. *Lecture Notes in Computer Science*, page 415–430, 2020.

[38] Vladimir Nekrasov, Chunhua Shen, and Ian Reid. Lightweight refinenet for real-time semantic segmentation. In *British Conference on Computer Vision (BMVC)*, 2018.

[39] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. Classmix: Segmentation-based data augmentation for semi-supervised learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1369–1378, 2021.

[40] Fei Pan, Sungsu Hur, Seokju Lee, Junsik Kim, and In So Kweon. Ml-bpm: Multi-teacher learning with bidirectional photometric mixing for open compound domain adaptation in semantic segmentation. In *European Conference on Computer Vision*, pages 236–251. Springer, 2022.

[41] Theodoros Panagiotakopoulos, Pier Luigi Dovesi, Linus Härenstam-Nielsen, and Matteo Poggi. Online domain adaptation for semantic segmentation in ever-changing conditions. In *European Conference on Computer Vision (ECCV)*, 2022.

[42] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.

[43] Serban Stan and Mohammad Rostami. Unsupervised model adaptation for continual semantic segmentation. In *AAAI*, 2021.

[44] Peng Su, Shixiang Tang, Peng Gao, Di Qiu, Ni Zhao, and Xiaogang Wang. Gradient regularized contrastive learning for continual domain adaptation. 2020. 00000.

[45] 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 *Computer Vision and Pattern Recognition*, 2022.

[46] Towaki Takikawa, David Acuna, Varun Jampani, and Sanja Fidler. Gated-scnn: Gated shape cnns for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5229–5238, 2019.

[47] Phillip Thomas, Lars Pandikow, Alex Kim, Michael Stanley, and James Grieve. Open synthetic dataset for improving cyclist detection, Nov 2021.

[48] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Real-time self-adaptive deep stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 195–204, 2019.

[49] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. Dacs: Domain adaptation via cross-domain mixed sampling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1379–1389, January 2021.

[50] Maxime Tremblay, Shirsendu S. Halder, Raoul de Charette, and Jean-François Lalonde. Rain rendering for evaluating and improving robustness to bad weather. *International Journal of Computer Vision*, 2020.

[51] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[52] Riccardo Volpi, Pau de Jorge, Diane Larlus, and Gabriela Csurka. On the road to online adaptation for semantic image segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[53] Vibashan VS, Poojan Oza, and Vishal M. Patel. Towards online domain adaptive object detection. *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan 2023.

[54] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[55] Haoran Wang, Tong Shen, Wei Zhang, Lingyu Duan, and Tao Mei. Classes matter: A fine-grained adversarial approach to cross-domain semantic segmentation. In *The European Conference on Computer Vision (ECCV)*, August 2020.

[56] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision, 2018.

[57] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation, 2022.

[58] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[59] Bingzhen Wei, Xu Sun, Xuancheng Ren, and Jingjing Xu. Minimal effort back propagation for convolutional neural networks. *arXiv preprint arXiv:1709.05804*, 2017.

[60] Tsung-Han Wu, Yi-Syuan Liou, Shao-Ji Yuan, Hsin-Ying Lee, Tung-I Chen, Kuan-Chih Huang, and Winston H Hsu. D2ada: Dynamic density-aware active domain adaptation for semantic segmentation. In *European Conference on Computer Vision (ECCV)*, 2022.

[61] Zuxuan Wu, Xintong Han, Yen-Liang Lin, Mustafa Gökhan Uzunbas, Tom Goldstein, Ser Nam Lim, and Larry S. Davis. Dcan: Dual channel-wise alignment networks for unsupervised scene adaptation. *Lecture Notes in Computer Science*, page 535–552, 2018.

[62] Zuxuan Wu, Xin Wang, Joseph Gonzalez, Tom Goldstein, and Larry Davis. ACE: Adapting to changing environments for semantic segmentation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2121–2130. IEEE, 2019.

[63] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Incremental adversarial domain adaptation for continually changing environments. 2018. 00000.

[64] Enze Xie, Wenjia Wang, Wenhai Wang, Peize Sun, Hang Xu, Ding Liang, and Ping Luo. Segmenting transparent objects in the wild with transformer. In *IJCAI*, 2021.

[65] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021.

[66] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3684–3692, 2018.

[67] Yanchao Yang and Stefano Soatto. FDA: Fourier domain adaptation for semantic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4084–4094. IEEE, 2020.

[68] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.

[69] Yuhui Yuan, Xiaokang Chen, Xilin Chen, and Jingdong Wang. Segmentation transformer: Object-contextual representations for semantic segmentation. In *European Conference on Computer Vision (ECCV)*, 2020.

[70] Pan Zhang, Bo Zhang, Ting Zhang, Dong Chen, Yong Wang, and Fang Wen. Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. 2021.

[71] Qiming Zhang, Jing Zhang, Wei Liu, and Dacheng Tao. Category anchor-guided unsupervised domain adaptation for semantic segmentation. *Advances in Neural Information Processing Systems*, 2019.

[72] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[73] Yizhou Zhou, Xiaoyan Sun, Zheng-Jun Zha, and Wenjun Zeng. Context-reinforced semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4046–4055, 2019.

[74] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[75] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018.

[76] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5982–5991, 2019.

[77] Yang Zou, Zhiding Yu, Xiaofeng Liu, B. V. K. Vijaya Kumar, and Jinsong Wang. Confidence regularized self-training. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.