

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

Marc Botet Colomer*^{1,2} Pier Luigi Dovesi*^{3 †}
 Theodoros Panagiotakopoulos⁴ Joao Frederico Carvalho¹ Linus Härenstam-Nielsen^{5,6}
 Hossein Azizpour² Hedvig Kjellström^{2,3} Daniel Cremers^{5,6,7} Matteo Poggi⁸

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

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

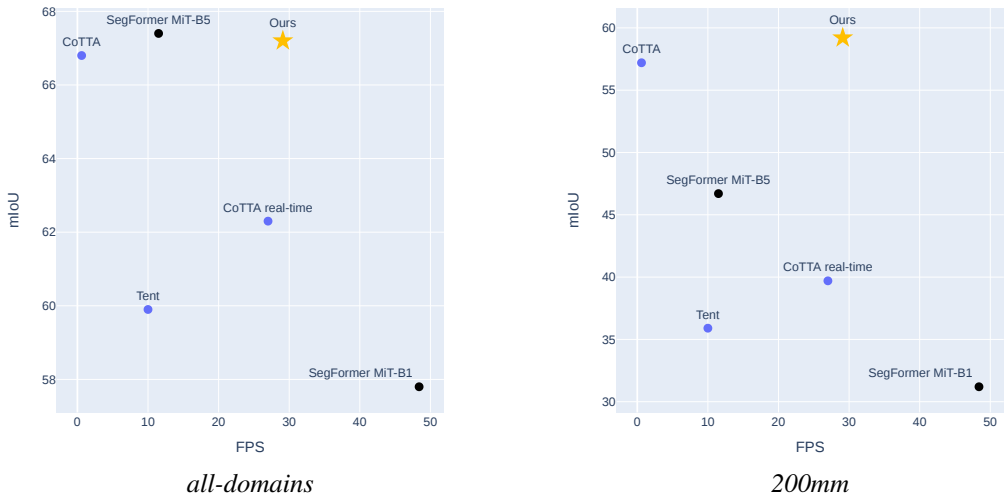


Figure 1: **Model averaged performances over domain.** We can observe how HAMLET reaches state-of-the-art accuracy, while running more than $6\times$ faster. Adaptive models are displayed in blue, while models trained on source, without adaptation, are displayed in black. On the *left* we show averaged performances over all domains, on the *right* we show how metrics drop in the hardest domain (200mm). The metric drop is limited for strong adaptive networks CoTTA [9], and HAMLET, while being a drastic drop for TENT [8] and SegFormer [10] (both MiT-B1 and MiT-B5). Finally, CoTTA *real-time* shows the performance of CoTTA in deployment conditions, hence running once every 50 frames.

This report introduces further details on the ICCV paper - “To Adapt or Not to Adapt? Real-Time Adaptation for Semantic Segmentation”. On the cover of this document, Figure 1, we propose a comparison of HAMLET (Hardware-Aware Modular Least Expensive Training) against state-of-the-art adaptation strategies, hence showing its highly favorable trade-off between speed and accuracy.

Then, starting with Section 1 we present an ablation study on the Hardware-Aware Modular Training (HAMT) method, where we show several speed-accuracy trade-offs. In Section 2 we dive deep in the Least Training (LT) and Adaptive Learning Rate (ALR) methodologies. We illustrate the behavior of different policies by comparing them on the same domain shift, and we also show how the policy copes with noisy domain shift detection. In addition, we present a quantitative analysis focusing on both speed and accuracy. A deeper analysis of the effect of the adaption on every single class is presented in

* Joint first authorship † Part of the work carried out while at Univrsees.

Section 3. Then, in Sections 4 and 5 we detail the model implementation and the chosen hyperparameters. In Section 6 we provide extensive studies on additional storms as presented in the OnDA benchmark [5]. We particularly focus on the behavior of repeated adaptation cycles and how they affect domain shift detection (hence ALR policies). Then, in Section 7 we illustrate additional experiments on the SHIFT dataset [6]. We first present a plot summarizing the model behavior on SHIFT and then we analyze how LT could prevent forgetting in long sequences without relevant domain shifts. We conclude by reporting some qualitative results in Section 8, and by referencing the qualitative videos uploaded on Youtube in Section 9. The first qualitative video (<https://www.youtube.com/watch?v=zjxPbCphPDE&t=139s>) showcases a comparison between HAMLET, CoTTA, and SegFormer MiT-B1 (no adaptation) on a Cityscapes [2] sequence with the Incremental Storm. Finally, we argue that synthetic data could only partially provide evidence of our methods: purposely, we run HAMLET on a real driving video taken in Korea across different rainy domains – <https://www.youtube.com/watch?v=Dwswey-GqQc>, whose author gave us consent to use it – to further support its effectiveness. The second qualitative video (<https://www.youtube.com/watch?v=zjxPbCphPDE>) shows the outcome of this experiment.

1. Ablation study: Hardware-Aware Modular Training

In this section, we present an additional ablation study on the HAMT module to further investigate its performance. Table 1 reports the adaptation results achieved by different configurations that exploit HAMT alone, without Active Training Modulation being enabled. We focus on the average adaptation performance and results on the hardest domain (200mm) and source domain (*clear*) to respectively measure the effectiveness of the adaptation scheme on the hardest domain and the robustness when adapting back on the source domain. Additionally, we report the framerate achieved by each configuration and the GFLOPS performed to backpropagate gradients through SegFormer.

We compare the Full training model (A) to several configurations that use a random sampling policy to pick which module to optimize (B, C, D, E) and HAMT sampling strategy (B', C', D', E'), characterized by uniform sampling – not hardware-aware – (B, B') or by setting β equal to 2.75, 1.75, or 1 for time conditioning, respectively, for entries (C, C'), (D, D'), and (E, E'). We also include a time-conditioned random sampling policy that uses softmax of the measured FPS for each action, with the temperature controlled by the same parameter β introduced in HAMT. The time-conditioned random sampling is achieved by skewing the uniform distribution with a softmax of the measured FPS for each action, effectively making more likely to pick an action, the faster it is. The softmax temperature is controlled by the same parameter β introduced in HAMT. This allows us to compare how HAMT performs compared to a simpler baseline bound to achieve similar FPS, nevertheless, the action choice will not be controlled by HAMT reward-punishment algorithm, but it will be randomly sampled.

As expected, we find that the most aggressive GFLOPS reduction corresponds to lower β values, but this comes with a price in metrics. Our observations show that even the naïve hardware-aware random policy can significantly reduce the GFLOPS dedicated to backpropagation without drastic metric drops. However, given the same β , HAMT policy always results in better performance than the naïve time-conditioned policies, both on hard domains (200mm of rain), the source domain (*clear*, 0mm of rain), and on average (F and T to signal forward and backward adaptation). As reported in the main paper, we set $\beta = 1.75$ as the default choice in any other experiments, allowing for a trade-off between GFLOPS reduction and adaptation effectiveness.

We can notice how the gain lead by HAMT over the naïve the metrics is more prominent for high values of β (*i.e.* less intense time conditioning) since it leaves to the reward-punishment algorithm more freedom of action to pick the best modules to train.

Overall, our ablation study shows that HAMT can effectively reduce the computational cost of adaptation without compromising accuracy. HAMT is especially useful when facing harsh and frequent domain shifts, where adaptation cannot be easily interrupted. Moreover, our study provides insights into the impact of time conditioning on the performance of HAMT and the importance of setting appropriate values of β .

Focus: Why are we using the 2nd derivative and not the 1st? Since every action corresponds to an optimization step, we expect that every action will minimize the loss function. Therefore, on average, all actions would receive positive rewards. This might lead to the model repeatedly taking the same action, moreover, we want to reward only those actions which are leading to a sharper loss reduction compared to the other optimization alternatives. Indeed the 2nd derivative will be positive only if the loss minimization has been greater than the expected linear extrapolation.

2. Ablation study: Active Training Modulation

We now focus on studying variations and single components of the policy we defined in Section 3.3 of the main paper. Specifically, with reference to the notation in Section 3.3, we define 5 different policy variants in an incremental manner, by

		clear		...	200mm		h-mean		Backward	% Backward
		F	B		F	F	B	T	GFLOPS	GFLOPS
(A)	Full training	73.5	73.2		62.6	68.7	71.0	69.8	30.8	100.0
(B)	Random policy (uniform)	73.5	72.9	...	61.6	68.2	70.4	69.3	22.6	73.5
(B')	HAMT (no time conditioning)	73.5	73.1		61.9	68.4	70.4	69.4	22.4	72.9
(C)	Random policy (time-conditioned $\beta = 2.75$)	73.5	73.0	...	59.0	67.5	70.1	68.8	21.4	69.5
(C')	HAMT $\beta = 2.75$	73.4	73.2		61.1	68.0	70.3	69.1	21.3	69.3
(D)	Random policy (time-conditioned $\beta = 1.75$)	73.3	72.8	...	59.9	67.6	70.4	69.0	20.7	67.3
(D')	HAMT $\beta = 1.75$	73.6	72.9		60.9	67.8	70.4	69.1	20.3	65.8
(E)	Random policy (time-conditioned $\beta = 1$)	73.1	72.6	...	60.2	67.6	70.0	68.8	19.4	63.1
(E')	HAMT $\beta = 1$	73.2	72.7		60.0	67.6	70.1	68.9	17.7	57.6

Table 1: **Ablation studies – HAMT module (3.2).** We report adaptation results on the Increasing Storm, achieved by exploiting different HAMT configurations. We also report the framerate, as well as the GFLOPS required to perform the backward pass during optimization.

assuming:

I) Constant learning rate and a number of iterations proportional to $|\Delta B_i|$. In this policy, it is assumed that the length of the adaptation window should grow with the intensity of the observed domain shift with respect to z , the minimum $|\Delta B_i|$ that trigger adaptation. We then compute the number of adaptation iterations as $L = K_l \frac{|\Delta B_i|}{z}$ with the factor K_l and the learning rate η kept constant. If new domain shifts are detected before the end of the adaptation windows, the remaining iterations are accumulated.

II) Constant initial learning rate with decay inversely proportional to $|\Delta B_i|$. In addition to the previous policy, now the learning rate η gradually decays until the adaptation is stopped, the smaller the domain shift, the faster the decay. The initial learning rate K_η is kept constant.

III) Initial learning rate proportional to $|\Delta B_i|$ with constant number of adaptation interations. This policy assumes to always adapt for a fixed amount of steps, hence L is fixed. However, the initial learning rate is proportional to the intensity of the measured domain shift $|\Delta B_i|$ with respect to the minimum detectable shift z . Therefore, the initial learning K_η , is computed as $K_\eta = P_\eta \frac{|\Delta B_i|}{z}$, where P_η is a constant that defines the minimum value of K_η .

IV) Number of iterations proportional to $|\Delta B_i|$, and initial learning rate proportional to the discretized distance B . This policy follows II), yet assuming an initial learning rate K_η that is higher for domains farther from the source.

$$K_\eta = K_{\eta,\min} + \frac{(B_i - B_{\text{source}})(K_{\eta,\max} - K_{\eta,\min})}{B_{\text{hard}} - B_{\text{source}}} \quad (1)$$

where B_{source} (resp. B_{hard}) is an estimate of B when the network is close to (resp. far from) the source domain; and $K_{\eta,\min}$ (resp. $K_{\eta,\max}$) is the value of K_η assigned when the network is close to (resp. far away from) the source.

V) Number of iterations proportional to $|S_i|$, direction sensitive, and initial learning rate proportional to the discretized distance B . This is the policy applied in the main paper, building upon policy IV). Here we use a variable multiplicative factor for the number of iterations K_l which depends both on both the distance from the source domain and the direction of the domain shift. The rationale is that domain shifts moving away from the source domain are likely to require a deeper and longer adaptation window. On the contrary, domain shifts moving closer to the source domain require fewer and fewer adaptation steps as we get closer. This is because the model shows good recalling of previously experienced domains as well as presenting strong performances close to the source thanks to regularization strategies we put in place.

$$\tilde{K}_l = \begin{cases} K_{l,\max} & \text{if } S_i \geq 0 \\ K_{l,\min} + \frac{(B_i - B_{\text{source}})(K_{l,\max} - K_{l,\min})}{B_{\text{hard}} - B_{\text{source}}} & \text{otherwise} \end{cases} \quad (2)$$

Where $K_{l,\min}$ (resp. $K_{l,\max}$) is the value of \tilde{K}_l assigned when the network is close to (resp. far away from) the source domain. We will appreciate how this last policy results in the best trade-off between accuracy and speed.

In Tab. 2 we showcase the results achieved on the Increasing Storm by different instances of SegFormer, according to the policy variants outlined so far. In (A) full training is performed, while in (B) and (C) we propose two baselines where we naïvely optimize the model every 15 and 20 frames respectively, or by implementing our policies (I-V). As for HAMT, we report performance on *clear* and *200mm* domains, as well as the average forward, backwards and total mIoU together with the framerate. As expected, reducing the adaptation steps to one every 15 or 20 frames definitely increases the FPS,

		clear		...	200mm	h-mean			FPS
		F	B		F	F	B	T	
(B)	Train every 15 iterations	73.2	73.3	...	53.3	64.1	68.3	66.2	26.5
(C)	Train every 20 iterations	73.2	73.0		50.2	63.2	67.9	65.5	34.0
(I)	Adapt. iter. (constant η)	73.4	72.8		55.6	65.5	69.3	67.4	25.3
(II)	Adapt. iter.	73.4	73.1	...	58.5	66.5	69.7	68.1	25.2
(III)	Adapt. η	73.4	71.4		55.4	65.3	67.9	66.6	31.0
(IV)	Adapt. iter. and η	73.4	73.2		57.9	66.0	70.0	68.0	23.4
(V)	Adapt. iter. and η , dir. sensitive	73.4	73.2		57.8	66.0	69.0	67.5	29.1

Table 2: **Ablation studies – Active Training Modulation (3.3)**. We report adaptation results on the Increasing Storm, achieved by exploiting different Active Training Modulation policies (I-V), together with framerates.

Domain	Model	Rider	M.bike	Sky	Road	Truck	S.walk	Wall	Veget.	Fence	Tr.Light	Terrain	Bus	Car	Train	Sign	Build.	Person	Pole	Bike	mIoU
clear	(both)	53.5	61.6	94.4	98.0	77.5	83.2	56.4	92.0	53.3	62.8	63.7	78.3	93.6	56.0	72.9	91.5	76.2	57.3	72.1	73.4
50mm	No adapt.	43.8	44.5	83.5	96.3	67.1	73.2	32.5	88.0	43.4	54.0	50.6	70.5	90.6	51.1	67.5	86.3	70.1	40.4	65.7	64.2
50mm	HAMLET	49.0	45.4	92.1	97.2	68.5	78.7	45.2	90.1	48.1	56.3	55.8	73.0	90.5	55.5	68.7	89.1	71.2	45.2	66.7	67.7
100mm	No adapt.	30.0	24.1	37.2	92.6	54.4	57.6	19.3	80.6	30.1	41.9	40.0	58.2	85.1	45.7	60.5	75.6	63.9	28.6	58.4	51.8
100mm	HAMLET	41.6	48.3	90.6	96.8	70.0	76.1	44.5	88.7	46.7	48.3	57.9	70.7	89.5	53.1	64.3	87.5	67.7	38.3	63.5	65.5
200mm	No adapt.	11.7	3.7	1.9	81.9	25.1	25.3	7.8	59.5	9.4	21.8	19.5	33.4	59.2	21.3	45.5	59.2	50.9	14.8	40.1	31.2
200mm	HAMLET	36.2	32.9	85.4	96.1	65.8	71.6	33.1	86.1	40.8	38.5	53.6	69.2	86.7	35.9	57.5	84.4	62.9	29.2	59.1	59.2

Table 3: **Single classes mIoUs**. Results on single classes by the source model on *clear* and *200mm*, and by HAMLET on *50mm*, *100mm*, *200mm* (Incremental storm, forward pass). The improvements achieved with online adaptation are consistent all across the board.

nevertheless it also notably reduces the overall adaptation effectiveness – in particular on the hardest domain of 200mm, with a drop of around 10% compared to full training. To attain a better accuracy-speed trade-off, we employ our policies: we can appreciate how (II) allows for the best overall adaptation performance as well as over 200m of rain while achieving the lowest FPS among the policies. Using (III) we obtain the highest FPS while losing accuracy both on average mIoU and when returning back to the source – specifically, resulting in the worse policy in backward adaptation. Policy (IV) provides for the best backward adaptation results, at the expense of forward adaptation and average performance. Finally, (V) balances all of the aspects considered before, while being the second fastest configuration among those considered. We also highlight how these policies merely represent examples of potential uses of the domain detection signals and how even a simple active training configuration policy could enable very fast and effective adaptation processes.

In Figure 2 we provide insight into the Active Training Modulation mechanism. In the top row, we exemplify two simple domain shift sequences: from clear weather to 50mm to clear (on the left) and a much more sudden change, from clear weather to 200mm to 75mm (on the right). In the second row, we display H , denoised in A (third row) and discretized in B (fourth row). We then display $S = \Delta B$ (fifth row) acting as the first derivative of B over time frames. On the left, the domain shift is correctly detected as a single domain shift. This is visible by having a single spike in S . On the other hand, in the harder scenario (on the right), the domain shift is detected in two consecutive steps. We could define this as a false positive detection. The rows below show how the ALR policies manage the training phases and the learning rate modulation. We remind the reader that when the learning rate is zero, the training phase is inhibited. We notice how the policy formulation can withstand the double-detection of the domain shift by simply recomputing the learning and accumulating the residual iterations, overall presenting a robust behaviour.

3. Single Classes Analysis

In Table 3 we present the per-class mIoU of HAMLET and a model just trained on the source domain (No adapt.). We present results on the source domain, where the two models are equivalent, and in the 50mm, 100mm, and 200mm domains of the forward pass of the Incremental Storm [5]. As expected HAMLET vastly improves the non-adapted baseline on each domain, in every single class. Interestingly, we see that HAMLET improvement is not just on a few classes, but instead, all classes are improved by a consistent amount. As expected some classes are more impacted by the domain change, such as the Sky and rare classes (e.g. M.bike, Rider, Fence), while some others present greater robustness (e.g. Road, Vegetation, Car, Person).

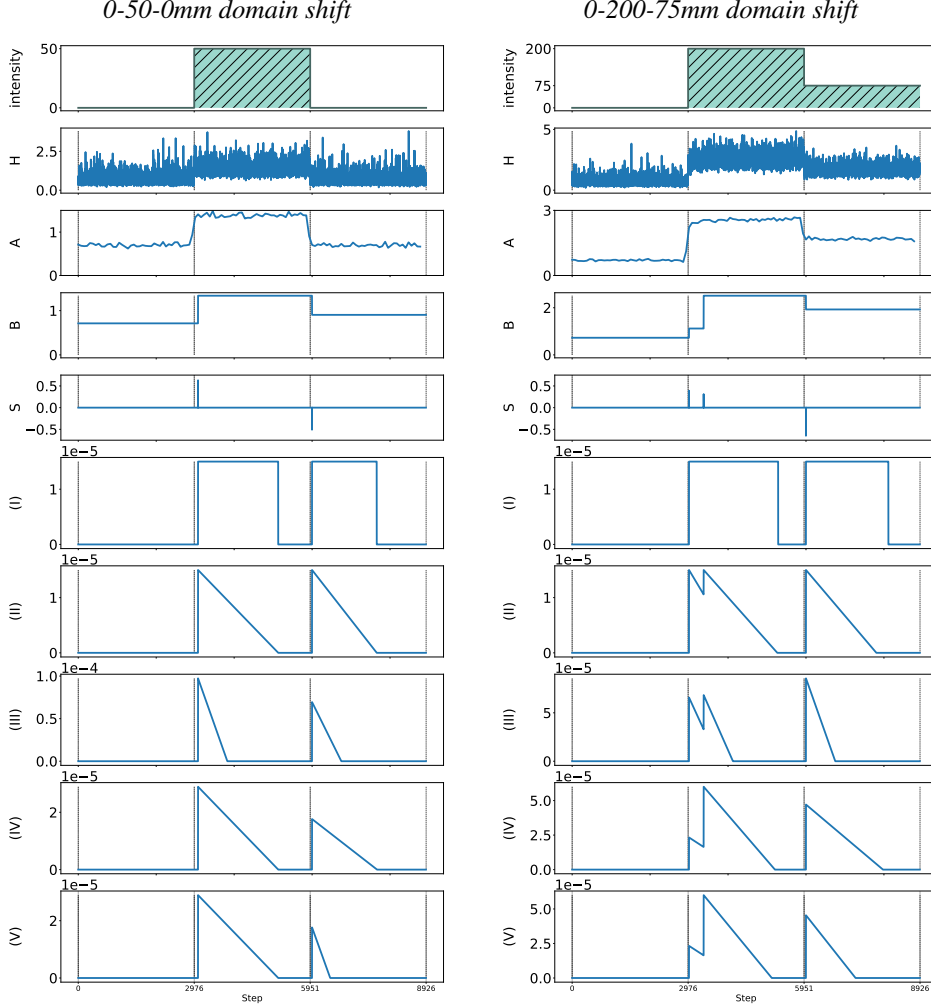


Figure 2: Focus on the mechanism of domain detection signals and relative adaptation process for each applied policy.

4. Model Setup: Additional Information

In this section, we present additional information on the UDA model and backbone used and provide details about the lightweight decoder.

Let $x \in \mathbb{R}^{C_{in} \times H \times W}$ denote an input image and $y \in [0, 1]^{C \times H \times W}$ denote a segmentation label with C number of classes. Let $\mathcal{D}_S = \{(x_S^{(i)}, y_S^{(i)})\}_{i=1}^{n_s}$ be the labeled source dataset and $\mathcal{D}_T = \{x_T^{(i)}\}_{i=1}^{n_t}$ the unlabeled target dataset encounter during deployment, which may contain multiple sequential domains. Our goal is to train a model f_θ that predicts the probability of each class in each pixel of the input image, such that $f_\theta(x) \in \mathbb{R}^{C \times H \times W}$.

To this end, we use a student-teacher scheme with parameters θ for the student model and parameters θ' for the teacher model. During each training iteration i , we optimize the student by minimizing the loss function in Eq. 3. The teacher is updated as an EMA of the student weights f_θ (Eq. 4) where α is the decay rate of the EMA.

$$\mathcal{L}^{(i)} = \mathcal{L}_S^{(i)} + \mathcal{L}_T^{(i)} + \lambda_{FD} \mathcal{L}_{FD}^{(i)} \quad (3)$$

$$\theta'^{(i+1)} \leftarrow \alpha \theta'^{(i)} + (1 - \alpha) \theta^{(i)} \quad (4)$$

The training loss in Equation 3 is a combination of three terms. The first term, \mathcal{L}_S , is a supervised term used to learn the semantic segmentation task using the replay buffer from the labeled dataset and a Cross-Entropy loss. The second term is

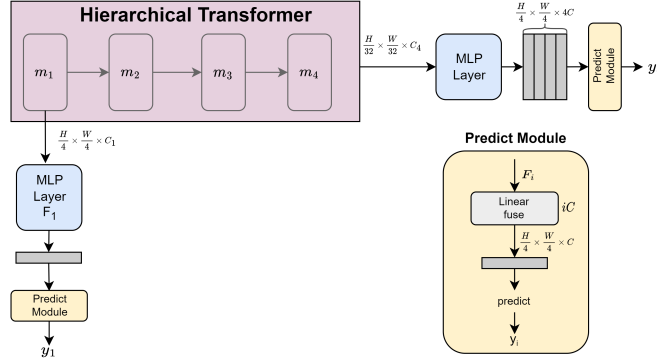


Figure 3: **Adaptive Domain Detector**. We attach a SegFormer all-MLP decoder after the first module m_1^{fd} . This allows us to obtain segmentation maps of any image at a low cost and with very limited speed impact.

a self-training loss that is learned from the target dataset, and the third term is a feature distance loss used as a regularizer. We perform self-training by training the student model f_θ on a strongly augmented version of the target dataset, along with one-hot encoded pseudo-labels generated by the teacher. The augmented images are generated by mixing randomly selected classes from the source image with target images following ClassMix [4]. \mathcal{L}_T is the cross-entropy between the mixed image and the mixed label weighted by factor q_T , as the ratio of the pixels that have a confidence level higher than a certain threshold. To prevent the student network’s weights from deviating significantly from a pre-trained model on the source dataset (static teacher), we incorporate a feature distance loss, denoted as \mathcal{L}_{FD} , in the training process. Specifically, the feature distance loss is computed by taking the features produced by the student network’s encoder f_θ and those produced by a static teacher network with frozen weights on a given input sample, and measuring the Euclidean distance between the feature embeddings generated by these two networks.

For our domain adaptive detector, we utilize SegFormer [10] as a semantic segmentation backbone, incorporating both its encoder and decoder design. We modify the static teacher model f^{fd} by connecting an extremely lightweight segmentation head, denoted as $d^{\text{fd}}1$, after the first encoder module m_1^{fd} , resulting in $h^{\text{fd}}1 = d^{\text{fd}}1 \circ m_1^{\text{fd}}$. This lightweight segmentation head follows the SegFormer decoder architecture, using an all-MLP decoder that takes feature encodings from m_1^{fd} with C_1 channels and produces segmentation maps using only MLP layers (as illustrated in Fig. 3).

5. Implementation details

We report any hyper-parameters used to train the described methods. The supervised models were trained using SegFormer pre-trained weights for 100’000 iterations (selecting the checkpoint with the best validation accuracy) using a learning rate of 6×10^{-5} , warm-up and linear decay scheduling. The online models were trained using AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay 0.01. The hyperparameters values described in the method section are: $\alpha = 0.1$, $K_l = 750$, $K_{\eta, \min} = 1.5 \times 10^{-4}$, $K_{\eta, \max} = 6 \times 10^{-5}$, $K_{l, \min} = 187$, $K_{l, \max} = 562$, $K_{\text{CM}, \min} = 0.5$, $K_{\text{CM}, \max} = 0.75$ and $m = 75$. For the storm and fog scenarios we use: $B_{\text{source}} = 0.8$, $B_{\text{hard}} = 2.55$. For the SHIFT dataset [6], we use $B_{\text{source}} = 0.46$, $B_{\text{hard}} = 1.85$ and $m = 200$. For the video sequences, we use the fog and storm parameters with $m = 350$. We also used 1000 source images as a buffer. All models performed training with a batch size of 1 and images scaled to 512×1024 resolution and random crops of 512×512 where using for training. Both in HAMLET and in the full training baseline we employ SegFormer decoder, without using DAFormer [3] custom head. It’s also worth noting that, to marginalize the impact of different backbones, all tested models in this work are using SegFormer MiT-b1 backbone (*i.e.* HAMLET, TENT, CoTTA) as model backbone, except if specified otherwise (*i.e.* OnDA and Advent [7] are using DeepLabV2 [1]).

During training (evaluation is included), HAMLET consists of the following forward passes:

- Student model using source buffer image
- Static teacher encoder using source buffer image (no decoder)
- EMA teacher using a target image
- Student model using mixed image
- Student model using a target image to provide a prediction
- First module of static teacher in the target image and relative small decoder

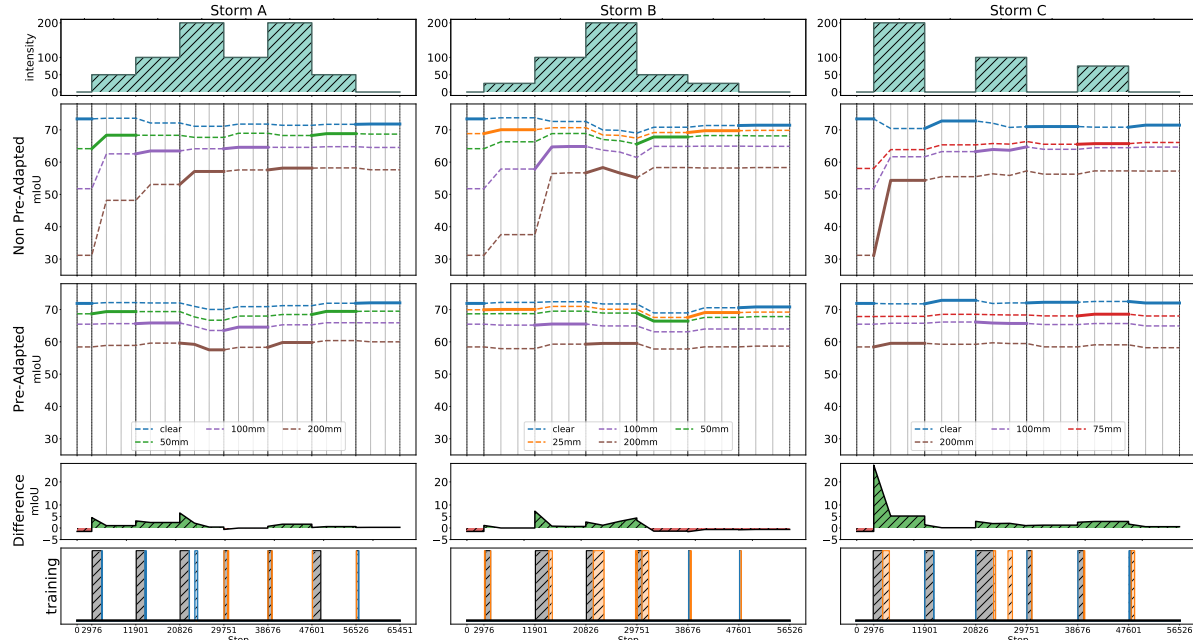


Figure 4: **Experimental results – Storms A, B, C.** Adaptation performance by two HAMLET instances, one trained on source domain (*clear*) and adapted for the first time, and one that has been pre-adapted on the Increasing Storm. In the last two rows, we show the boost in accuracy achieved by the latter model compared to the former, as well as the iterations during which the two are optimized (orange: the former only, blue: the latter only, gray: both).

	clear		50mm		10mm		200mm		h-mean		FPS T	GFLOPS	
	F	B	F	B	F	B	F	B	F	B			
Full training	73.6	73.0	69.3	70.1	66.6	66.4	61.4	62.8	67.4	67.9	67.6	4.6	125.2
HAMLET (non pre-adapted)	73.4	71.8	68.3	68.8	63.5	64.6	57.1	58.2	65.0	65.4	65.2	22.8	58.2
HAMLET (pre-adapted)	71.9	72.1	69.4	69.4	65.8	64.6	58.1	59.8	65.9	66.1	66.0	20.2	59.8

Table 4: **Experimental results – Storm A**

Backpropagation is applied on the student model only. Afterwards, the dynamic teacher is updated as EMA of the student. During simple evaluation, HAMLET consists of the following forward passes:

- Student model using a target image to provide a prediction
- First module of static teacher in the target image and relative small decoder

The full source code used for our experiments is attached to this document (**hamlet.code.zip**).

6. More storms and longer adaptation analysis

We run HAMLET on three additional rainy scenarios, generated as detailed in [5]. We both evaluate the performance of the brand-new adaptation cycle, starting from SegFormer trained on source domain and adapting to the new storms A, B and C. Additionally, we test a model previously online adapted on the Increasing Storm scenario and compare the two (Non Pre-Adapted and Pre-Adapted) in terms of performance and training phases.

Figure 4 collects, from left to right, the results achieved on Storms A, B, and C as defined in [5]. On top, we plot the rain intensity over time faced during the adaptation process, followed by mIoU plots highlighting how the two models introduced before adapt and the difference in terms of mIoU achieved by the pre-adapted model compared to the brand-new one. On the last row, we show the iterations during which the models are optimized, specifically in gray when both run back-propagation, while in orange and blue when only the brand-new or the pre-adapted model are optimized, respectively.

We notice, similarly to OnDA [5], how HAMLET also benefits from previous adaptation on the Increasing Storm. The highest gain is achieved on Storm C, in which the domain rapidly switches from source to the hardest one, *i.e.* 200mm. Moreover, we can appreciate in general how the pre-adapted model witnessed almost no drop in accuracy on the inactive domains. This is caused by the Active Training Modulation strategy, which limits the amount of adaptation steps performed

	clear	25mm	100mm	200mm	50mm	25mm 2	clear 2	total h-mean	FPS	GFLOPS
Full training	73.3	70.6	68.2	64.1	66.1	70.9	72.1	69.2	4.6	125.2
HAMLET (non pre-adapted)	73.4	70.0	67.4	61.6	61.5	68.8	70.6	67.3	20.0	50.3
HAMLET (pre-adapted)	71.9	70.0	67.8	62.5	63.0	67.7	69.9	67.4	25.2	44.9

Table 5: **Experimental results – Storm B**

	clear 1	200mm	clear 2	100mm	clear 3	75mm	clear 4	clear h-mean	target h-mean	total h-mean	FPS	GFLOPS
Full training	73.6	60.1	73.4	65.6	73.0	68.8	72.9	73.2	64.6	69.3	4.5	125.2
HAMLET (non pre-adapted)	73.4	54.4	72.7	64.7	71.0	65.7	71.4	72.1	61.1	67.0	16.4	51.9
HAMLET (pre-adapted)	71.9	59.5	72.9	65.7	72.2	68.6	72.0	72.3	64.4	68.7	22.2	48.3

Table 6: **Experimental results – Storm C**

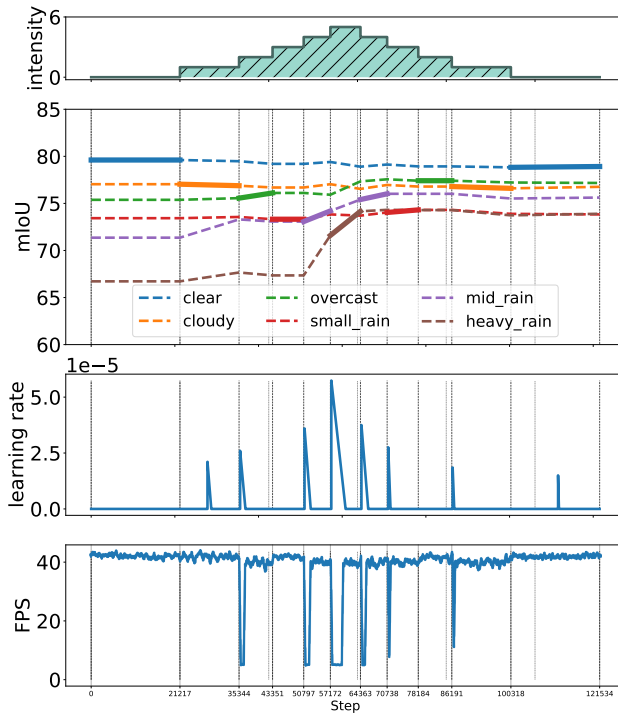


Figure 5: **Experimental results – SHIFT benchmark.** We show mIoU over active (bold) and inactive (dashed) domains, learning rate and FPS.

by HAMLET to be as few as needed to adapt to the new domains while neglecting the occurrence of catastrophic forgetting over inactive domains. Indeed, once adaptation has been performed over the Increasing Storm, it results sufficient to maintain high accuracy when moving to the new storms, as pointed out by the almost horizontal dashed lines in the pre-adapted plots. By focusing on the last row, we point out how, in most times, the non pre-adapted model runs significantly more optimization steps (orange) compared to the pre-adapted one (blue), which can already deal with the domain switches occurring in these storms. This is due to its prior experience on the Increasing Storm, indeed the domain detection relies both on the static lightweight decoder and the student itself. When the student becomes more robust to new domains, also the domain detection becomes more accurate. We also notice how, despite training only a fraction of the iterations, the Non Pre-adapted model can still catch-up with the pre-adapted one, with a delay, even in the hardest transition, *i.e.* storm C. For a quantitative overview of HAMLET performance on the three storms, we collect the results in Tables 4, 5 and 6, respectively for Storms A, B, and C. In particular, we point out how the pre-adapted model is more accurate, as well as faster than the non pre-adapted one on B and C, since it activates adaptation fewer times as previously discussed with reference to Figure 4.

To conclude, HAMLET can benefit from previous adaptation both in terms of accuracy, as well as speed (the fewer the optimization steps, the higher the framerate).

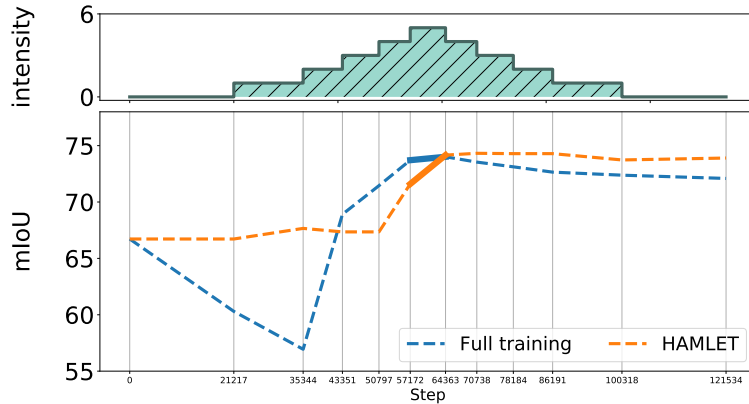


Figure 6: **Experimental results – SHIFT benchmark.** We show the mIoU of the *heavy_rain* domain during the training cycle for Full training and HAMLET. Bold lines represent when the domain is active and dashed lines when is inactive.

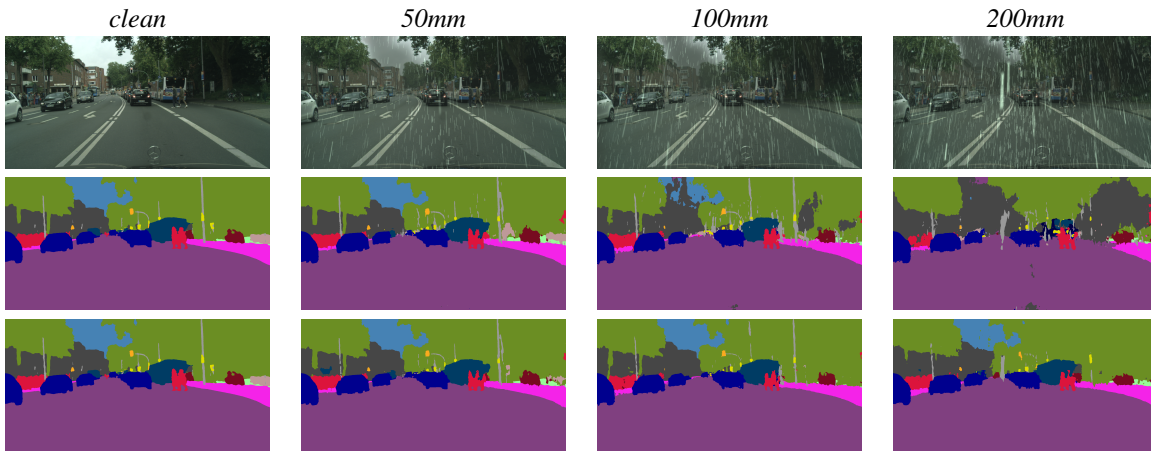


Figure 7: **Qualitative results – HAMLET in action.** From left to right, we show the same source frame, augmented with increasing rain intensity, respectively *clean*, *50mm*, *100mm* and *200m*. From top to bottom: input image, prediction by SegFormer trained on source and HAMLET.

7. SHIFT analysis

We now dive deeper in HAMLET performance on the SHIFT benchmark. Figure 5 depicts, on top, the rain intensity characterizing any domain encountered while running HAMLET on SHIFT. Then, we plot the mIoU achieved both on current (bold) and inactive (dashed) domains, as done for the Increasing Storm in the main paper and Storms A, B, C in the previous section. Then, we show how the learning rate changes based on the domain shift detection, followed by the framerate achieved by HAMLET at any step.

From the mIoU plot, we can notice how the drop in accuracy, even on the hardest domain, is moderate compared to what was observed in OnDA benchmarks [5]. We speculate that this might be caused by the full-synthetic nature of this domain, which makes the task easier. Interestingly, the performance on *small_rain* and *mid_rain* domains continue to improve even after HAMLET moves to further domains. In general, as previously observed on Storms A, B, C, HAMLET do not experience any catastrophic forgetting on the inactive domains that have been faced previously. For what concerns domain shift detection, we can notice how this sometimes occurs with a slight delay – *i.e.* *clear* to *cloudy* and vice-versa – or does not occur at all – *i.e.* *overcast* to *small_rain* and vice-versa. Nevertheless, once again, this confirms that just a few adaptation steps aligned with the domain shifts are enough to achieve an accuracy comparable to the one obtained with full training, as shown in Tab. 4 in the main paper.

This dataset evaluation offers further insights when it comes to observing another problematic behavior of naïve full

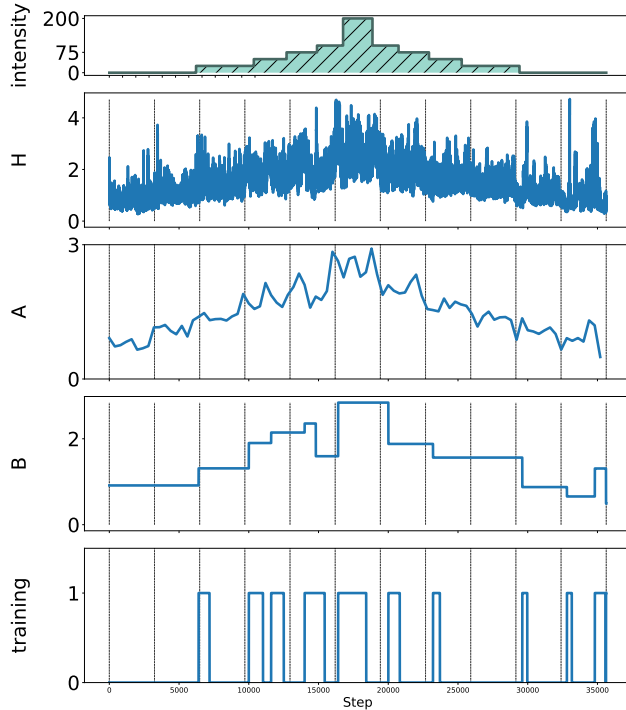


Figure 8: **HAMLET adaptation schedule over the qualitative video sequence.** From first to last row we present: rain intensity, domain detection signals: H , A , B , training phases (1: training + inference, 0: inference)

training. Indeed, besides being vastly more computationally expensive, training when it is not required, contributes to the futile specialization of specific domains, hence leading to worse generalization on other domains. This is clearly visible in Figure 6 when we focus on the performances of our Full Training baseline on the *heavy_rain* domain. During the adaptation to clear weather, we notice how evaluating on *heavy_rain* leads to progressively worse performances without achieving any significant improvement in clear weather either. Despite its ability to eventually adapt, this behavior might raise concerns when it comes to sudden domain shifts and it hints to potential domain forgetting. To support this, we show in Figure 6 the accuracy achieved on the *heavy_rain* domain at any time during adaptation, when being the active (bold) or inactive (dashed) domain, for both SegFormer adapted with full training and HAMLET. We can notice how the full training regime leads to dramatic drops in accuracy on this domain when it is inactive, until it is actually encountered. HAMLET, on the contrary, can preserve its original accuracy on the *heavy_rain*, proving that selective adaptation also avoids catastrophic forgetting, to which full training is prone to.

8. Qualitatives

In Figure 7 we present extensive qualitative examples from the Increasing Storm evaluation set. Figure 7 shows the results achieved by the source model and HAMLET on increasing rain intensity. We can notice how the source model, at first, is robust to moderate rain. When moving towards higher intensity, the model gradually starts failing, whereas HAMLET keeps high accuracy.

9. Videos

To conclude, we attach two qualitative videos to this document. For the first (<https://www.youtube.com/watch?v=zjxPbCphPDE&t=139s>) we emulate a realistic deployment by synthesizing rain over Cityscapes. The domain shift sequence follows the same pattern as the Increasing Storm. In this case, we cap the video framerate at 5.88FPS using the same setup of [5]. On this video, we run SegFormer in three main flavors: 1) trained on source domain, 2) adapting using CoTTA and 3) adapting with HAMLET. We mainly compare against CoTTA: while HAMLET keeps the pace with the considered framerate, CoTTA – which runs at 0.6FPS – is trained over 1 frame every 10, allowing it to keep the pace with the

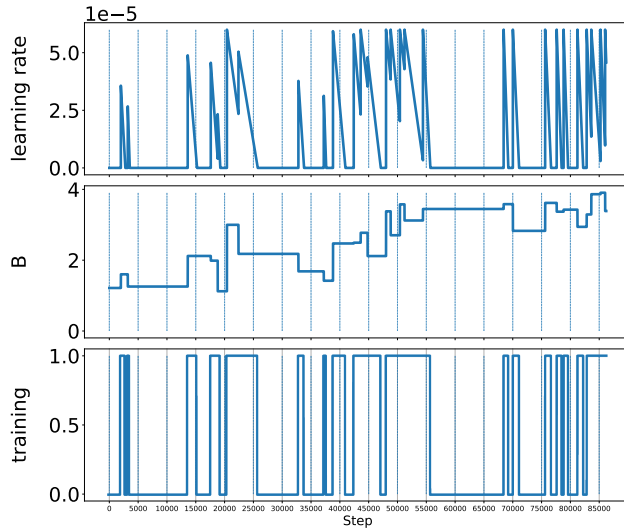


Figure 9: **HAMLET adaptation schedule over the qualitative video sequence.** From first to last row we present: learning rate schedule, domain detection signal B , training phases (1: training + inference, 0: inference)

incoming frames. This emulates a realistic behavior during deployment in which all the frames are processed sequentially, yet favoring CoTTA – since a higher framerate, *e.g.* 30FPS, would require CoTTA to adapt on even fewer frames to keep the pace. The video sequence is unlabelled, so we cannot compute mIoU and thus we can appreciate our results only qualitatively, nevertheless, we can provide an overview of the adaptation process operated by HAMLET. Figure 8 sketches the domain sequence, domain shift detections, and relative training intervals, *i.e.* 1 when training is active and 0 otherwise. The average speed theoretically obtained by HAMLET in this sequence is 20.4FPS, even though the input rate was capped at 5.88FPS. It is also interesting to notice how sequential frames and underlying natural domain shifts taking place over the video are making the adaptation task significantly more challenging than the Increasing Storm benchmark proposed in [5], nevertheless, HAMLET manages to identify and activate short training burst in correspondence to the domain shifts, enough to vouch for effective adaptation to the new domains encountered.

In the second video (<https://www.youtube.com/watch?v=zjxPbCphPDE>) we showcase HAMLET in action in a real environment – *i.e.*, on the road from Seoul to Daegu, Korea. During the trip, we face several different domain transitions, meeting heavy rain, highway environment, dusk, and even nighttime. This latter qualitative result proves that, despite most experiments in the main paper having been conducted in semi-synthetic datasets, HAMLET is effective on real data as well and can be effectively deployed for real applications. The video shows, on top, the input RGB images from the sequence being processed, and at the bottom, the results by SegFormer trained on the source domain (left) and HAMLET being adapted on the sequence itself (right). First and foremost, we point out how the video itself exposes several domain shifts due to the environment itself – *i.e.*, SegFormer has been trained on Cityscapes, featuring cities from Germany in a mostly urban environment, while the whole video features Korea and transits from urban roads to highways. We can appreciate how these domain shifts do not represent a challenge for HAMLET. Then, we observe that rain represents one of the earliest, weather challenges faced in the video, both in the form of small droplets on the glass shield of the car, as well as in actual storms met during driving. While the accuracy of the source SegFormer model dramatically drops in these domains, HAMLET rapidly copes with them and maintains a much higher quality of the results. In the last part of the video, we encounter nighttime domains: despite the much lower brightness in the images and the lower contrast between the different regions (*e.g.*, road vs vegetation or cars), HAMLET can still keep the drop in accuracy moderate, while the source SegFormer model results completely ineffective on such an unseen domain, rarely distinguishing the road from any generic vehicle. Fig. 9 sketches the domain shift detections and relative learning rate schedules, training intervals, *i.e.* 1 when training is active and 0 otherwise. We can observe how HAMLET can identify several domain shifts and tune the adaptation rate accordingly.

References

- [1] 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.
- [2] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. ADVENT: Adversarial entropy minimization for domain adaptation in semantic segmentation. 2019. 00000.
- [8] 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.
- [9] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation, 2022.
- [10] 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.