Learning from Future: A Novel Self-Training Framework for Semantic Segmentation

Anonymous Author(s)
Affiliation
Address
email

Abstract

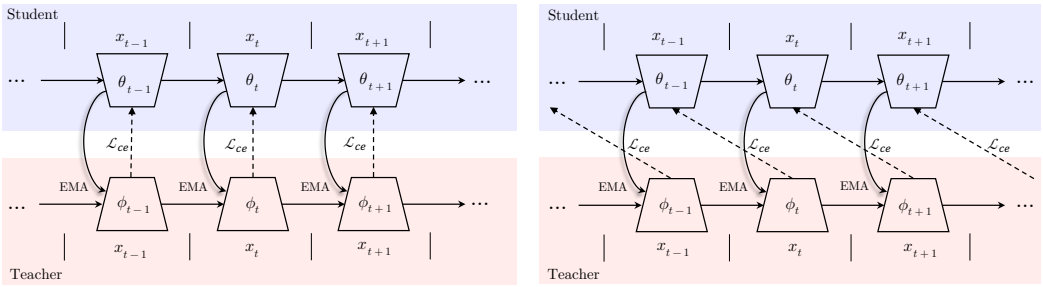
Self-training has shown great potential in semi-supervised learning. Its core idea is to use the model learned on labeled data to generate pseudo-labels for unlabeled samples, and in turn teach itself. To obtain valid supervision, active attempts typically employ a momentum teacher for pseudo-label prediction yet observe the confirmation bias issue, where the incorrect predictions may provide wrong supervision signals and get accumulated in the training process. The primary cause of such a drawback is that the prevailing self-training framework acts as guiding the current state with previous knowledge, because the teacher is updated with the past student only. To alleviate this problem, we propose a novel self-training strategy, which allows the model to *learn from the future*. Concretely, at each training step, we first virtually optimize the student (*i.e.*, caching the gradients without applying them to the model weights), then update the teacher with the virtual future student, and finally ask the teacher to produce pseudo-labels for the current student as the guidance. In this way, we manage to improve the quality of pseudo-labels and thus boost the performance. We also develop two variants of our *future-self-training* (FST) framework through peeping at the future both deeply (FST-D) and widely (FST-W). Taking the tasks of unsupervised domain adaptive semantic segmentation and semi-supervised semantic segmentation as the instances, we experimentally demonstrate the effectiveness and superiority of our approach under a wide range of settings. Code will be made publicly available.

1 Introduction

Improving the labeling efficiency of deep learning algorithms is vital in practice since acquiring high-quality annotations could consume great effort. Self-training (ST) offers a promising solution to alleviate this issue by learning with limited labeled data and large-scale unlabeled data [47, 24]. The key thought is to learn a model on labeled samples and use it to generate pseudo-labels for unlabeled samples to teach the model itself. In general, a teacher network that maintains an exponential moving average (EMA) of the student (*i.e.*, the model to learn) weights is used for pseudo-label prediction, as shown in Fig. 1a. Intuitively, such a training strategy relies on the *previous* student states to supervise the current state, which amounts to using a poor model to guide a good one given the fact that a model tends to perform better along with the training process. As a result, the confirmation bias issue [4, 9] emerges from existing ST approaches, where the wrong supervision signals caused by those incorrect pseudo-labels get accumulated during training.

To break through the predicament of seeking supervision only from the past states, we propose *future-self-training* (FST), which allows the model to learn from its *future self*. Fig. 1b illustrates the concept diagram of our FST. Compared to the conventional ST framework in Fig. 1a, which employs the t-step teacher (*i.e.*, updated with the student at moments $1, 2, \ldots, t-1$) to guide the

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) Self-training (b) Future-self-training

Figure 1: **Concept comparison** between self-training (ST) and our future-self-training (FST). (a) ST employs a teacher, which collects information from the *past* states, to supervise the student. (b) Our FST derives a teacher at the *future* moment and utilizes it to guide the current student.

t-step student, FST presents a new training manner by urging the t-step student to learn from the $(t+1)$-step teacher. However, at the start of the training step t, the $(t+1)$-step teacher is not available yet since it is dependent on the to-be-optimized t-step student. To tackle this obstacle, we come up with a *virtual updating* strategy. Concretely, we first optimize the current student just like that in the traditional ST. Differently, we do *not* actually update the student weights but cache the gradients instead. Such stashed gradients can be treated as the "virtual future" and help derive the $(t+1)$-step teacher. Finally, the training of step t borrows the pseudo-labels predicted by the latest teacher, and this time we apply the gradients to the student weights for real.

Recall that our motivation of encouraging the model to learn from the future is to help it acquire knowledge from an advanced teacher. To this end, we put forward two variants based on our FST framework to make the teacher more capable. On the one hand, we propose FST-D to investigate the future *deeply*. For this case, we ask the teacher to move forward for K steps via virtual updating, thus the t-step student can be better supervised by the $(t+K)$-step teacher. On the other hand, FST-W originates from the idea of model soups [53], which reveals that the averaging weights of multiple fine-tuned models can improve the performance. We hence propose to explore the future *widely* with teachers developed from different training samples and expect the student to learn from all these $(t+1)$-step teachers simultaneously.

We evaluate our proposed FST on the tasks of both unsupervised domain adaptive (UDA) semantic segmentation and semi-supervised semantic segmentation. The superiority of FST over the prevailing ST framework is summarized in Fig. 2, where our teacher model is capable of producing pseudo-labels with much higher quality and hence assists the student with a better performance. This is because, along with the training process, the future states usually outperform the past states and thus can provide more accurate supervision, reducing the damage of confirmation bias. Such a comparison validates our primary motive of learning from the future. Furthermore, we observe consistent performance gain under a broad range of experimental settings (*e.g.*, network architectures and datasets), demonstrating the effectiveness and generalizability of our approach.

Figure 2: **Performance comparison** between self-training (ST) and our future-self-training (FST), including the pseudo-label quality on unlabeled training samples (left) and the evaluation performance (right). The comparison is conducted under the same number of updates of the student, which is the final model used for evaluation.

2 Related work

Domain adaptive semantic segmentation. UDA semantic segmentation aims at transferring the knowledge from a labeled source domain to an unlabeled target domain, which is often viewed as a special semi-supervised learning problem. Early methods for UDA segmentation focus on diminishing the distribution shift between the source and target domain at the input level [23, 40, 18], the feature level [46, 10, 8, 31], or the output level [46, 48, 36]. Over the years, adversarial learning

[19, 17] has been the dominant approach to aligning the distributions. However, the alignment-based methods may destroy the discrimination ability of features and cannot guarantee a small expected error on the target domain [58]. In contrast, self-training [2], which is originated from semi-supervised learning (SSL) [29], is introduced to directly minimize a proxy cross-entropy (CE) loss on the target domain. By leveraging the model itself to generate pseudo-labels on unlabeled data, self-training together with tailored strategies such as consistency regularization [61, 3], cross-domain mixup [45, 62], contrastive learning [26, 34, 63, 31], pseudo-label refine [51, 58, 60], auxiliary tasks [49, 50] and class balanced training [30] achieves excellent performance. Recently, Hoyer et al. [24] empirically proved that the transformer architecture [55] is more robust to domain shift than CNN. They propose a transformer-based framework with three efficient training strategies in pursuit of milestone performance.

Semi-supervised semantic segmentation. Self-training is widely studied in SSL literature [47]. To facilitate the usage of unlabeled samples, Tarvainen et al. [44] propose a mean teacher framework for consistency learning between a *student* and a momentum updating *teacher*. This idea is extended later to semi-supervised semantic segmentation, which trains the student model with high-confident *hard* pseudo-labels predicted by the teacher. On this basis, extensive attempts improve semi-supervised semantic segmentation by CutMix augmentation [16], class-balanced training [64, 25, 21] and contrastive learning [64, 1, 33, 52]. A closely relevant topic to self-training in SSL is consistency regularization, which believes that enforcing semantic or distribution consistency between various perturbations, such as image augmentation [27] and network perturbation [57], can improve the robustness and generalization of the model. In general, consistent regularization methods are used together with a ST framework. We focus on improving the basic ST in this work.

Nesterov's accelerated gradient descent. A related idea to our work is Nesterov's accelerated gradient descent (NAG). Originally proposed in [37] for solving convex programming problem, NAG is a first-order optimization method with a better convergence rate than gradient descent. With the rise of deep learning, NAG is adopted as an alternative to momentum stochastic gradient descent (SGD) to optimize neural networks [43, 14]. It is intuitively considered to perform a look ahead gradient evaluation and then make a correction [7]. Due to its solid theoretical explanations [65, 5] and remarkable performance, many works incorporate NAG with various tasks. In [32], Lin et al. adopt NAG into the area of adversarial attack, where they propose a Nesterov's iterative fast gradient sign method to improve the transfer ability of adversarial examples. In [56], Yang et al. explore the utilization of NAG in federal learning. Different from NAG that pursues accelerated convergence, our work aims at building a stronger pseudo-label generator and improving the performance of traditional self-training.

3 Method

3.1 Background

Consider such a real-world scenario where we have access to a labeled segmentation dataset $\mathcal{D}_L = \{x_l, y_l\}_{l=1}^{n_l}$ from distribution P and an unlabelled one $\mathcal{D}_U = \{x_u\}_{u=1}^{n_u}$ from unknown distribution Q. We are required to build a semantic segmentation model using the combination of $\mathcal{D}_L$ and $\mathcal{D}_U$. A general case is when $P \neq Q$, the problem falls into the category of UDA semantic segmentation. Otherwise, it is usually treated as a regular SSL task.

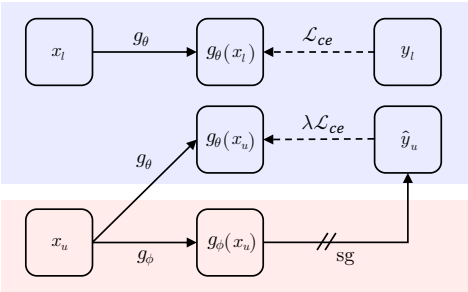


Figure 3: **Illustration** of the ST framework with a teacher g_ϕ. "sg" means stop-gradient.

Self-training provides a unified solution and achieves state-of-the-art performance on both settings [24, 52]. One of the most common and widely used forms of self-training in semantic segmentation is a variant of mean teacher, which is shown in Fig. 3. Denote by g_θ the segmentation model required to be trained, and θ its parameters. The mean teacher framework trains the *student* g_θ on unlabeled data with pseudo-labels predicted by a momentum *teacher* g_ϕ, which has the same architecture to the student but with different parameters ϕ. Specifically, as the training progresses, the teacher evolves with the student by maintaining an EMA of student weights on each training iteration. This ensembling

enables generating high quality predictions on unlabeled samples, and using them as training targets improves performance. Formally, at each training step, the teacher is first updated and then predict pseudo-labels to train the student.

$$\begin{aligned}\phi_{t+1} &= \mu\phi_t + (1-\mu)\theta_t, \\ \theta_{t+1} &= \theta_t - \gamma\nabla_\theta \left[\mathcal{L}(g_{\theta_t}(x_l), y_l) + \lambda\mathcal{L}(g_{\theta_t}(x_u), \hat{y}_u|\phi_{t+1})\right],\end{aligned} \tag{1}$$

where μ is the momentum coefficient, γ is the learning rate, and λ is the dynamic re-weighting parameter to weigh the training of labeled and unlabeled data. $\hat{y}_u$ denotes the pseudo-labels predicted by ϕ_{t+1}, *i.e.*, $\hat{y}_u = \arg\max g_{\phi_{t+1}}(x_u)$. $\mathcal{L}$ is the pixel-wise cross-entropy training objective, which can be written as

$$\mathcal{L}(x, y) = -\sum_{j=1}^{H\times W}\sum_{c=1}^{C} \mathbb{I}_{y^{j,c}=1} \log g_\theta(x)^{j,c}, \tag{2}$$

where $H \times W$ is the input image size and C is the total number of classes.

Limitation of self-training. Despite the remarkable performance, self-training suffers from the problem of confirmation bias. To be specific, the inherent noise in pseudo-labels could undesirably mislead the student training, which in return affects the pseudo-label prediction, and thereby results in noise accumulation. Though a momentum updating strategy in the mean teacher framework improves tolerance with inaccurate pseudo-labels, this issue is still a bottleneck since the student still relies on learning from its own *past* training states.

3.2 Learning from future self

An intuitive observation shown in Fig. 2 is that the performance of the student model generally improves during training, despite the noise in supervision. From this perspective, a reasonable conjecture is, can we use model information from future moments to guide the current training iteration? Motivated by this, we propose *future-self-training* for facilitating the utilization of unlabeled data in semantic segmentation. Concretely, at each training step, we propose to directly update the teacher model by the student weights from the next training moment. To this end, a simple modification to Eq. (1) is made as follows.

$$\begin{aligned}\phi_{t+1} &= \mu\phi_t + (1-\mu)\left(\theta_t - \gamma\nabla_\theta \left[\mathcal{L}(g_{\theta_t}(x_l), y_l) + \lambda\mathcal{L}(g_{\theta_t}(x_u), \hat{y}_u|\phi_t)\right]\right), \\ \theta_{t+1} &= \theta_t - \gamma\nabla_\theta \left[\mathcal{L}(g_{\theta_t}(x_l), y_l) + \lambda\mathcal{L}(g_{\theta_t}(x_u), \hat{y}_u|\phi_{t+1})\right].\end{aligned} \tag{3}$$

Furthermore, it can be seen that Eq. (3) only uses a virtual future state to update the teacher and ignores the current student weights θ_t. Our mission here is to establish a reliable pseudo-label generator (*i.e.* a stronger teacher). In terms of the ensembling effect of EMA, it is not necessary to discard θ_t. Therefore, an improved version of FST is proposed as follows.

$$\begin{aligned}\phi_{t+1} &= \mu'\{\mu\phi_t + (1-\mu)\theta_t\} + (1-\mu')(\theta_t - \gamma\nabla_\theta[\mathcal{L}(g_{\theta_t}(x_l), y_l) + \lambda\mathcal{L}(g_{\theta_t}(x_u), \hat{y}_u|\phi_t)]), \\ \theta_{t+1} &= \theta_t - \gamma\nabla_\theta \left[\mathcal{L}(g_{\theta_t}(x_l), y_l) + \lambda\mathcal{L}(g_{\theta_t}(x_u), \hat{y}_u|\phi_{t+1})\right],\end{aligned} \tag{4}$$

where a new momentum parameter μ' is introduced to distinguish the contribution of current and future model weights to teacher updates. We provide pseudo-codes to further illustrate how we implement Eq. (4) in *Supplementary Material*.

3.3 Exploring a deeper future

We reiterate that the key insight of FST is to look ahead during training, which allows to mine more accurate supervision from future model states. In experiments, we found that Eq. (4) exhibits only a slight improvement in performance (Tab. 2), showing that this one-step future exploration strategy is insufficient.

Therefore, we further propose a looking ahead *deeper* strategy to peek into deeper future student states. To be specific, at each training step, we update the teacher not only with the student weights from the next moment, but also with those from deeper steps. Formally, denote by $\widetilde{\phi}_t = \mu\phi_t + (1-\mu)\theta_t$ and $\widetilde{\theta}_t = \theta_t$ two agent variables firstly. Then, we can use the co-evolving $\widetilde{\phi}_t$ and $\widetilde{\theta}_t$ for *virtual updating* as follows.

$$\begin{aligned}\widetilde{\theta}_{t+k+1} &= \widetilde{\theta}_{t+k} - \gamma\nabla_{\widetilde{\theta}}[\mathcal{L}(g_{\widetilde{\theta}_{t+k}}(x_l), y_l) + \lambda\mathcal{L}(g_{\widetilde{\theta}_{t+k}}(x_u), \hat{y}_u|\widetilde{\phi}_{t+k})], \\ \widetilde{\phi}_{t+k+1} &= \mu'\widetilde{\phi}_{t+k} + (1-\mu')(\widetilde{\theta}_{t+k+1}),\end{aligned} \tag{5}$$

where $k = \{0, ..., K-1\}$ indexes the serial virtual steps for current training and K is the total number of exploration steps. Finally, we use the future information aware teacher $\widetilde{\phi}_{t+K}$ as the pseudo-label generator to supervise the current training. A simple reassignment and gradient descent update are applied to form the deeper version of FST, which is so called FST-D as shown below.

$$\begin{aligned} \phi_{t+1} &= \widetilde{\phi}_{t+K}, \\ \theta_{t+1} &= \theta_t - \gamma \nabla_\theta [\mathcal{L}(g_{\theta_t}(x_l), y_l) + \lambda \mathcal{L}(g_{\theta_t}(x_u), \hat{y}_u | \phi_{t+1})]. \end{aligned} \tag{6}$$

3.4 Exploring a wider future

On the other hand, looking ahead *wider* instead of *deeper* is another intuitive way to enhance future exploration. Inspired by the recent progress [53] that an ensemble of different model weights often shows excellent performance, we propose to first explore the next moment in different optimization directions and then use the average of them to update the teacher. Concretely, we obtain different optimization directions by feeding *different data batches* to the student model at each training moment. Thus, a wider version of FST, *i.e.*, FST-W, is presented as follows.

$$\begin{aligned} \phi_{t+1} &= \mu' \{\mu \phi_t + (1-\mu)\theta_t\} + (1-\mu')(\theta_t - \frac{1}{N} \sum_{i=1}^{N} \gamma \nabla_\theta [\mathcal{L}(g_{\theta_t}(x_l^i), y_l^i) + \lambda \mathcal{L}(g_{\theta_t}(x_u^i), \hat{y}_u^i | \phi_t)]), \\ \theta_{t+1} &= \theta_t - \gamma \nabla_\theta [\mathcal{L}(g_{\theta_t}(x_l), y_l) + \lambda \mathcal{L}(g_{\theta_t}(x_u), \hat{y}_u | \phi_{t+1})], \end{aligned} \tag{7}$$

where i indexes different samples and N is parallel virtual exploration steps.

Eq. (7) holds due to the fact that averaging the model weights is equivalent to averaging the gradients first and then updating the parameters by gradient descent. It is worth noting that FST-D and FST-W are complementary that can be utilized together. However, this is beyond the scope of our work, and we leave this exploration to the future.

4 Experiment

The experiment section is organized as follows. First, we illustrate the experimental setup and implementation details in Sec. 4.1 and Sec. 4.2. Then, we evaluate the proposed FST and analyze the two variants in Sec. 4.3. After that, we conduct extensive ablation studies to dissect our method in Sec. 4.4. Finally, we compare our FST with existing state-of-the-art alternatives on both UDA and semi-supervised benchmarks in Sec. 4.5.

4.1 Setup

Datasets and tasks. We evaluated our method on UDA and semi-supervised semantic segmentation. In UDA segmentation, we use synthetic labeled images from GTAV [38] and SYNTHIA [39] as the source domain and use real images from Cityscapes [13] as the target domain. In addition, PASCAL VOC 2012 [15] is used for standard semi-supervised evaluation. To simulate a semi-supervised setting, we randomly sample a portion *(i.e.*, $1/4$, $1/8$, and $1/16$) of images together with corresponding segmentation masks from the training set as the labeled data and treat the rest as the unlabeled samples.

Evaluation metric. Mean Intersection over Union (mIoU) is reported for evaluation. In SYNTHIA $\rightarrow$ Cityscapes UDA benchmark, 16 and 13 of the 19 classes of Cityscapes are used to calculate mIoU, following the common practice [3, 24].

Baselines. We first build strong baselines of the classical ST framework. For UDA segmentation, we adopt the basic framework from [45], which contains a ClassMix augmentation. Standard cross-entropy loss is calculated on both labeled and unlabeled data. We use the efficient Encoder-Decoder structure for all semantic segmentation models, where the networks various in the structure of encoders and decoders. In the semi-supervised benchmark, we use the classical ST without other tricks as the baseline, because it has been proved to achieve competitive performance while maintaining simplicity [27].

Table 1: **Comparison between ST and our FST**, where we explore the future with either (a) the same data batch as the current or (b) a different data batch from the current. "SourceOnly" means training the model with labeled data only, whose result is borrowed from [24] as the reference. $4\times$ means using quadruple samples per mini-batch. All results are averaged over 3 random seeds.

Method	mIoU	Δ
SourceOnly	34.3 ± 2.2	-
ST	56.3 ± 0.4	-
-	-	-
Naive-FST	56.4 ± 0.4	$\uparrow 0.1$
Improved-FST	57.7 ± 0.6	$\uparrow 1.4$
FST-W	56.8 ± 0.1	$\uparrow 0.5$
FST-D	$\mathbf{59.8 \pm 0.1}$	$\uparrow \mathbf{3.5}$

(a) Future exploration with the <u>same</u> data batch.

Method	Batch	mIoU	Δ
SourceOnly	$1\times$	34.3 ± 2.2	-
ST	$1\times$	56.3 ± 0.4	-
ST	$4\times$	55.5 ± 0.4	$\downarrow 0.8$
Naive-FST	$1\times$	58.7 ± 2.3	$\uparrow 2.3$
Improved-FST	$1\times$	58.7 ± 0.7	$\uparrow 2.4$
FST-W	$1\times$	59.3 ± 0.5	$\uparrow 3.0$
FST-D	$1\times$	$\mathbf{59.6 \pm 1.4}$	$\uparrow \mathbf{3.3}$

(b) Future exploration with a <u>different</u> data batch.

4.2 Implementation details

Image augmentation. The proposed FST and its baselines use the same image augmentation for fair comparison. In UDA semantic segmentation, color jitter, Gaussian blur and ClassMix [45] are used as the strong data augmentation for the unlabeled target domain, which follows the practice in [24]. In semi-supervised semantic segmentation, we use random flip and random crop, and the images are resized to 513×513 for both teacher and student.

Network architecture. We use the DeepLabV2 [11] as the basic segmentation architecture for UDA segmentation, where the ASPP decoder only uses the dilation rates 6 and 12 following [46]. For Transformer-based networks, we adopt from [24] and [54] as the decoders. In semi-supervised segmentation, we evaluate our method on the commonly used DeepLabV2 [22], DeepLabV3+ [12] and PSPNet [59] with ResNet-101 [22] as the backbone.

Optimization. In UDA segmentation, the model is trained with an AdamW [28] optimizer, a learning rate of 6×10^{-5} for the encoder and 6×10^{-4} for the decoder, a weight decay of 0.01, linear learning rate warmup with 1.5k iterations and linear decay afterwards. We train the model on a batch of two 512×512 random crops for a total of 40k iterations. The momentum u is set to 0.999. In semi-supervised segmentation, the model is trained with a SGD optimizer, a learning rate of 0.0001 for the encoder and 0.001 for the decoder, a weight decay of 0.0001. We train the model with 16 labeled and 16 unlabeled images per-batch for a total of 40 epochs.

4.3 Comparison with self-training

We first comprehensively compare our FST with classical ST to evaluate the effectiveness. The results are shown in Tab. 1. To simplify, we use GTAV as the labeled data and Cityscapes as the unlabeled data for evaluation. All methods use the same experimental settings for fairness.

Quantitative analyses. We illustrate the improvements of Naive-FST (Eq. (3)), Improved-FST (Eq. (4)), FST-D (Eqs. (5) and (6)) and FST-W (Eq. (7)) compared with classical ST (Eq. (1)) in Tab. 1a. These methods use the same batch of data for virtual forward at each step of future exploration. As presented, Naive-FST only shows a negligible boost because the current student state is discarded without contributing to the teacher. By revising it, the improved FST in Eq. (6), which is a special case of FST-D when $K = 1$, achieves an improvement of 1.4% mIoU. Further, FST-D (with $K = 3$) clearly outperforms ST by a margin of 3.5% mIoU, which benefits from the higher-quality pseudo-labels generated by a more reliable teacher as shown in Fig. 2. In contrast, FST-W shows a slight improvement of only 0.5% mIoU under the same data batch setting. Thus, we prefer the deeper variant and adopt it as the basic approach in this paper, *i.e.*, FST stands for FST-D unless specified. We also analyse the effect of exploration steps (*i.e.*, K) on the training process. As suggested in Fig. 4a, FST spends only about $1/3$ of the total training time to reach the performance level of ST. Besides, we find that a larger K can achieve higher mIoU at the beginning of the training process. When $K = 4$, however, the performance in later training iterations drops and gets worse than $K = 3$. We speculate that this is because the deeper exploration becomes unnecessary in the later training stage. This interesting phenomenon indicates that an adaptive exploration mechanism may bring better results.

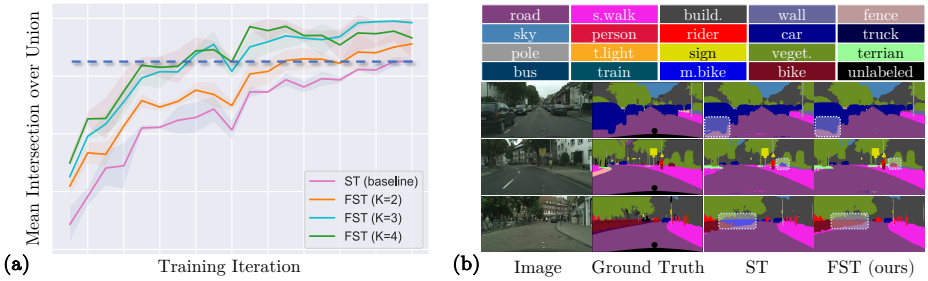


Figure 4: (a) **Performance curves** for ST and FST with various K values. The comparison is conducted under the same number of updates of the student, which is the final model used for evaluation. (b) **Qualitative comparison** on Cityscapes [13], where dashed white boxes highlight the visual improvements.

Table 2: **Generalization** of FST across architectures. All results are averaged over 3 random seeds.

Method	K	mIoU	Δ
ST	-	55.0 ± 0.9	-
FST	2	56.3 ± 1.0	$\uparrow 1.3$
FST	3	$\mathbf{56.9 \pm 0.5}$	$\uparrow \mathbf{1.9}$
FST	4	56.4 ± 0.9	$\uparrow 1.4$

(a) DeepLabV2 [11] *w/* ResNet-50 [22].

Method	K	mIoU	Δ
ST	-	56.3 ± 0.4	-
FST	2	57.8 ± 1.3	$\uparrow 1.5$
FST	3	$\mathbf{59.8 \pm 0.1}$	$\uparrow \mathbf{3.5}$
FST	4	59.7 ± 0.8	$\uparrow 3.4$

(b) DeepLabV2 [11] *w/* ResNet-101 [22].

Method	K	mIoU	Δ
ST	-	56.3 ± 0.8	-
FST	2	58.1 ± 3.1	$\uparrow 1.8$
FST	3	58.5 ± 0.7	$\uparrow 2.2$
FST	4	$\mathbf{58.8 \pm 1.0}$	$\uparrow \mathbf{2.5}$

(c) PSPNet [59] *w/* ResNet-101 [22].

Method	K	mIoU	Δ
ST	-	61.3 ± 0.7	-
FST	2	63.7 ± 2.0	$\uparrow 2.4$
FST	3	64.3 ± 2.3	$\uparrow 3.0$
FST	4	$\mathbf{64.4 \pm 2.0}$	$\uparrow \mathbf{3.1}$

(d) UPerNet [54] *w/* Swin-B [35].

Method	K	mIoU	Δ
ST	-	59.9 ± 2.0	-
FST	2	62.5 ± 1.2	$\uparrow 2.6$
FST	3	62.5 ± 1.9	$\uparrow 2.6$
FST	4	$\mathbf{62.6 \pm 1.8}$	$\uparrow \mathbf{2.7}$

(e) UPerNet [54] *w/* BEiT-B [6].

Method	K	mIoU	Δ
ST	-	68.3 ± 0.5	-
FST	2	69.1 ± 0.3	$\uparrow 0.8$
FST	3	$\mathbf{69.3 \pm 0.3}$	$\uparrow \mathbf{1.0}$
FST	4	68.8 ± 0.9	$\uparrow 0.5$

(f) DAFormer [24] *w/* MiT-B5 [55].

Qualitative analyses. Fig. 4b provides some qualitative comparisons, where our FST can correct some mistakes made by ST. Taking the presented sample in the second row as an instance, ST struggles to distinguish between *bicycle* and *motorcycle*, while our FST successfully predicts it. More visualization results and analyses can be found in *Supplementary Material*.

Data batches for future exploration. In Sec. 3.4, we derive FST-W, which uses different samples for future exploration in parallel. Tab. 1a and Tab. 1b compare the performance of using the same and different data batches. Note that FST-W in Tab. 1a could produce slightly different mixed images for virtual forward, since we use ClassMix augmentation. It is obvious that the parallel exploration with different samples performs better because the differences between models are important for ensembling.

Generalization of popular architectures. To verify the generality under various advanced semantic segmentation models, we evaluate FST (the deeper variant) on two mainstream backbones (*i.e.*, CNN and Transformer) with four commonly used segmentation decoders. As presented in Tab. 2, FST shows consistent performance improvement over classical ST, including DeepLab [11], PSPNet [59] and UPerNet [54]. Besides, FST shows significant improvements not only on supervised pretrained CNN [22] and Transformer backbones [35, 55] but also on unsupervised pretrained BEiT [6]. Note that, the established ST baselines are strong, which even surpass many complex multi-stage methods (*e.g.*, [58]) proposed recently. FST achieves 59.8% mIoU using DeepLabV2 and ResNet-101. More comparisons between FST and existing CNN-based methods are provided in *Supplementary Material*.

4.4 Ablation studies

Deeper or wider. It can be seen from Tab. 1 that FST-D performs better than FST-W no matter using the same or different data batches for future exploration. It is worth noting that under the setting of Tab. 1b, the teacher model in FST sees more data per-iteration compared to the original ST, which has the effect of expanding the batch size in disguise. To make a fair comparison, we also build a ST baseline with a larger batch size. The results show that the performance gain of FST-W comes from the method itself instead of utilizing more data in each iteration. In addition, we conduct ablations to compare FST-D and FST-W with each step (*i.e.*, K and N), which are shown in Tab. 3a and Tab. 3b.

Table 3: **Analyses on the two variants of FST**, including FST-D (Sec. 3.3) and FST-W (Sec. 3.4). All results are averaged over 3 random seeds.

Method	Backbone	K	mIoU	Δ
ST	ResNet-101	-	56.3 ± 0.4	-
FST-D	ResNet-101	2	58.6 ± 0.4	$\uparrow 2.3$
FST-D	ResNet-101	3	59.6 ± 1.4	$\uparrow 3.3$
FST-D	ResNet-101	4	$\mathbf{59.8 \pm 2.0}$	$\uparrow \mathbf{3.5}$

(a) Effect of K in FST-D.

Method	Backbone	N	mIoU	Δ
ST	ResNet-101	-	56.3 ± 0.4	-
FST-W	ResNet-101	2	58.5 ± 1.6	$\uparrow 2.2$
FST-W	ResNet-101	3	$\mathbf{59.3 \pm 0.5}$	$\uparrow \mathbf{3.0}$
FST-W	ResNet-101	4	58.6 ± 2.0	$\uparrow 2.3$

(b) Effect of N in FST-W.

Table 4: (a) **Ablation study** on the hyper-parameter μ' (Sec. 3.2). (b) **Comparison with longer-training baselines.** All results are averaged over 3 random seeds.

Method	μ'	mIoU	Δ
ST	-	56.3 ± 0.4	-
-	-	-	-
FST	0.99	58.8 ± 1.6	$\uparrow 2.5$
FST	0.999	59.8 ± 0.1	$\uparrow 3.5$
FST	0.9999	58.7 ± 0.6	$\uparrow 2.4$
FST	0.99999	$\mathbf{59.9 \pm 0.9}$	$\uparrow \mathbf{3.6}$

(a) Effect of μ' in Eq. (5).

Method	Backbone	Schedule	mIoU	Δ
ST	ResNet-101	$1\times$	56.3 ± 0.4	-
ST	ResNet-101	$4\times$	59.3 ± 0.6	$\uparrow 3.0$
FST	ResNet-101	$1\times$	$\mathbf{59.8 \pm 0.1}$	$\uparrow \mathbf{3.5}$
ST	MiT-B5	$1\times$	68.3 ± 0.5	-
ST	MiT-B5	$3\times$	68.3 ± 1.1	$\uparrow 0.0$
FST	MiT-B5	$1\times$	$\mathbf{69.1 \pm 0.3}$	$\uparrow \mathbf{0.8}$

(b) Comparison with longer training schedules.

Both variants use different data batches for future exploration since FST-W performs well only under this setting. It can be concluded that FST-D performs better than FST-W, which is consistent with the conclusion in the above. Besides, comparing Tab. 2b and Tab. 3a, we observe that using different data batches amplifies the performance jitter of each run. We guess that this may be due to the diversity of data sampled for future exploration.

Effect of the serial exploration steps K. K controls the number of virtual exploration steps in FST-D. In Tab. 2, we ablate K on six settings, each evaluated over 3 runs. As presented, $K = 3$ shows steady improvements, while increasing it further brings negligible impact. Thus, we recommend using $K = 3$ as the basic practice. We also ablate K when using different data batches for exploration, which is presented in Tab. 3a.

Effect of the momentum μ'. It is a common practice to set the momentum μ of EMA to a large value such as 0.999 in self-training. A separate momentum μ' that controls the contribution of future student states to the teacher is set in our FST. We conduct ablation experiments to observe the effect of μ'. As shown in Tab. 4a, FST shows robustness against the change of μ'. In our experiments, we set $\mu' = 0.999$ as the default setting unless specifically stated, which equals the value of μ.

Effect of the parallel exploration steps N. In FST-W, N controls the parallel exploration steps of the next training moment. We conduct ablation experiments in Tab. 3b to verify the influence of N. As can be seen, $N = 3$ performs well among the evaluated values, which is a similar observation to K and implies that it is an acceptable choice in practice.

Longer training schedules. We perform forward and backward propagation to obtain weights as the estimation of future student states. This simple implementation linearly increases the training time over the number of exploration steps, *i.e.*, K. Note that in our method, the student is trained with the same training iterations as classical ST and does *not* see more samples per-iteration, thereby the comparisons in Tabs. 1 and 2 are totally fair. Even though, we establish stronger baselines with longer training schedules and compare them with our method. We find that the performance of the longer scheduled ST baseline decreases heavily in the later training stages as the model is fitting the noise in pseudo-labels. Besides, as shown in Tab. 4b, the performance of longer training baselines still performs worse than our FST, which further proves the effectiveness of our method.

4.5 Comparison with state-of-the-art alternatives

In this subsection, we evaluate our FST with state-of-the-art approaches on the tasks of semi-supervised semantic segmentation and unsupervised domain adaptive semantic segmentation.

Evaluation on semi-supervised segmentation. We first evaluate the proposed FST on traditional semi-supervised semantic segmentation. As shown in Tab. 5, we compare FST (with $K = 3$) with ST

Table 5: **Evaluation on the semi-supervised learning (SSL) setting** on PASCAL VOC 2012 [15], where $1/16$, $1/8$, and $1/4$ stand for using 664, 1323, and 2646 samples as the labeled set, respectively.

	PSPNet [59]			DeepLabV2 [11]			DeepLabV3+ [12]		
Method	1/16	1/8	1/4	1/16	1/8	1/4	1/16	1/8	1/4
ST	65.47	72.24	75.47	68.45	72.54	76.21	73.31	74.20	77.78
FST (ours)	68.35	72.77	75.90	69.43	73.18	76.32	73.88	76.07	78.10
Δ	2.88 ↑	0.53 ↑	0.43 ↑	0.98 ↑	0.64 ↑	0.11 ↑	0.57 ↑	1.87 ↑	0.32 ↑

Table 6: **Evaluation on the unsupervised domain adaptation (UDA) setting** on two benchmarks. Our results are averaged over 3 random seeds.

Method	Road	S.walk	Build.	Wall	Fence	Pole	T.light	Sign	Veget.	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	M.bike	Bike	mIoU
GTAV [38] → Cityscapes [13]																				
SourceOnly	76.1	18.7	84.6	29.8	31.4	34.5	44.8	23.4	87.5	42.6	87.3	63.4	21.2	81.1	39.3	44.6	2.9	33.2	29.7	46.1
ProDA [58]	87.8	56.0	79.7	46.3	44.8	45.6	53.5	53.5	88.6	45.2	82.1	70.7	39.2	88.8	45.5	59.4	1.0	48.9	56.4	57.5
CPSL [30]	92.3	59.9	84.9	45.7	29.7	**52.8**	**61.5**	**59.5**	87.9	41.5	85.0	**73.0**	35.5	90.4	48.7	73.9	26.3	53.8	53.9	60.8
DAFormer [24]	**95.7**	**70.2**	**89.4**	53.5	**48.1**	49.6	55.8	59.4	**89.9**	47.9	92.5	72.2	44.7	92.3	74.5	78.2	65.1	55.9	61.8	68.3
FST (ours)	95.3	67.7	89.3	**55.5**	47.1	50.1	57.2	58.6	**89.9**	**51.0**	**92.9**	72.7	**46.3**	**92.5**	**78.0**	**81.6**	**74.4**	**57.7**	**62.6**	**69.3**
SYNTHIA [39] → Cityscapes [13]																				
SourceOnly	56.5	23.3	81.3	16.0	1.3	41.0	30.0	24.1	82.4	–	82.5	62.3	23.8	77.7	–	38.1	–	15.0	23.7	42.4
ProDA [58]	87.8	45.7	84.6	37.1	0.6	44.0	54.6	37.0	**88.1**	–	84.4	74.2	24.3	88.2	–	51.1	–	40.5	45.6	55.5
CPSL [30]	87.2	43.9	85.5	33.6	0.3	47.7	**57.4**	37.2	87.8	–	88.5	**79.0**	32.0	**90.6**	–	49.4	–	50.8	59.8	57.9
DAFormer [24]	84.5	40.7	**88.4**	41.5	6.5	50.0	55.0	**54.6**	86.0	–	89.8	73.2	**48.2**	87.2	–	53.2	–	53.9	61.7	60.9
FST (ours)	**88.3**	**46.1**	88.0	**41.7**	**7.3**	**50.1**	53.6	52.5	87.4	–	**91.5**	73.9	48.1	85.3	–	**58.6**	–	**55.9**	**63.4**	**61.9**

on three partition protocols. Equipped with three commonly used semantic segmentation networks, *i.e.*, PSPNet [59], DeepLabV2 [11], and DeepLabV3+ [12], FST consistently improves classical ST by considerable margins. For instance, on the $1/8$ partition protocol, FST with DeepLabV3+ outperforms ST by 1.87% mIoU, showing substantial improvement. In short, FST demonstrates remarkable performance on the semi-supervised benchmark. More comprehensive comparisons between our FST against other state-of-the-art alternatives can be found in *Supplementary Material*.

Evaluation on unsupervised domain adaptive semantic segmentation. We then compare our FST with previous self-training based competitors on UDA benchmark. The results are presented in Tab. 6. Our FST is build upon DAFormer [24], which is the state-of-the-art method currently. On GTAV → Cityscapes benchmark, our FST exceeds DAFormer by a considerable margin of 1.0% mIoU and shows dominant performance in most categories. Compared to the source only trained model, FST surpasses it by 23.2% mIoU, indicating a further step toward practical applications. The bottom part of Tab. 6 shows the comparisons on SYNTHIA → Cityscapes benchmark. FST exceeds DAFormer by 1.0% mIoU on 16 classes. Besides, our FST achieves 68.5% mIoU on 13 classes, exceeding DAFormer by 1.1%. In summary, our FST achieves new state-of-the-art performance on the UDA benchmark.

5 Conclusion

In this paper, we present a future-self-training framework for semantic segmentation. As an alternative to classical self-training, our approach mitigates the confirmation bias problem and achieves better performance on both UDA and semi-supervised benchmarks. The key insight of our method is to mine a model's own future states as supervision for current training. To this end, we propose two variants, namely FST-D and FST-W, to explore the future states deeply and widely. Experiments on a wide range of settings demonstrate the effectiveness and generalizability of our methods.

Discussion. The major drawback of this work is that our approach is time-consuming, since we need to forward a temporary model to acquire virtual future model states. Although the number of updates to the student does *not* increase, it is better to optimize the process of obtaining future model parameters. To this end, an acceptable way is to maintain an ahead student model to provide information from future moments, which trades space for time. Besides, our approach is *general* that can be applied to other self-training frameworks such as FixMatch [41] and other tasks such as semi-supervised image recognition [44], object detection [35], few-shot learning [42], and unsupervised representation learning [20]. We will conduct further studies on these issues in the future.

References

[1] I. Alonso, A. Sabater, D. Ferstl, L. Montesano, and A. C. Murillo. Semi-supervised semantic segmentation with pixel-level contrastive learning from a class-wise memory bank. In *Int. Conf. Comput. Vis.*, 2021.

[2] M.-R. Amini, V. Feofanov, L. Pauletto, E. Devijver, and Y. Maximov. Self-training: A survey. *arXiv preprint arXiv:2202.12040*, 2022.

[3] N. Araslanov and S. Roth. Self-supervised augmentation consistency for adapting semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[4] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *Int. Joint Conf. Neural Networks*, 2020.

[5] M. Assran and M. Rabbat. On the convergence of nesterov's accelerated gradient method in stochastic settings. In *Int. Conf. Mach. Learn.*, 2020.

[6] H. Bao, L. Dong, and F. Wei. Beit: Bert pre-training of image transformers. In *Int. Conf. Learn. Represent.*, 2022.

[7] A. Botev, G. Lever, and D. Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In *Int. Joint Conf. Neural Networks*, 2017.

[8] W.-L. Chang, H.-P. Wang, W.-H. Peng, and W.-C. Chiu. All about structure: Adapting structural information across domains for boosting semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[9] B. Chen, J. Jiang, X. Wang, J. Wang, and M. Long. Debiased pseudo labeling in self-training. *arXiv preprint arXiv:2202.07136*, 2022.

[10] C. Chen, W. Xie, W. Huang, Y. Rong, X. Ding, Y. Huang, T. Xu, and J. Huang. Progressive feature alignment for unsupervised domain adaptation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.

[12] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Eur. Conf. Comput. Vis.*, 2018.

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[14] T. Dozat. Incorporating nesterov momentum into adam. In *Int. Conf. Learn. Represent. Worksh.*, 2016.

[15] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 2010.

[16] G. French, S. Laine, T. Aila, M. Mackiewicz, and G. D. Finlayson. Semi-supervised semantic segmentation needs strong, varied perturbations. In *Brit. Mach. Vis. Conf.*, 2020.

[17] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016.

[18] R. Gong, W. Li, Y. Chen, and L. V. Gool. Dlow: Domain flow for adaptation and generalization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Adv. Neural Inform. Process. Syst.*, 2014.

[20] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Adv. Neural Inform. Process. Syst.*, 2020.

[21] D. Guan, J. Huang, A. Xiao, and S. Lu. Unbiased subclass regularization for semi-supervised semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[23] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Int. Conf. Mach. Learn.*, 2018.

[24] L. Hoyer, D. Dai, and L. Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[25] H. Hu, F. Wei, H. Hu, Q. Ye, J. Cui, and L. Wang. Semi-supervised semantic segmentation via adaptive equalization learning. *Adv. Neural Inform. Process. Syst.*, 2021.

[26] G. Kang, Y. Wei, Y. Yang, Y. Zhuang, and A. Hauptmann. Pixel-level cycle association: A new perspective for domain adaptive semantic segmentation. In *Adv. Neural Inform. Process. Syst.*, 2020.

[27] Z. Ke, D. Qiu, K. Li, Q. Yan, and R. W. Lau. Guided collaborative training for pixel-wise semi-supervised learning. In *Eur. Conf. Comput. Vis.*, 2020.

[28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Int. Conf. Learn. Represent.*, 2015.

[29] D.-H. Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Int. Conf. Mach. Learn.*, 2013.

[30] R. Li, S. Li, C. He, Y. Zhang, X. Jia, and L. Zhang. Class-balanced pixel-level self-labeling for domain adaptive semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[31] S. Li, B. Xie, B. Zang, C. H. Liu, X. Cheng, R. Yang, and G. Wang. Semantic distribution-aware contrastive adaptation for semantic segmentation. *arXiv preprint arXiv:2105.05013*, 2021.

[32] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *Int. Conf. Learn. Represent.*, 2020.

[33] S. Liu, S. Zhi, E. Johns, and A. J. Davison. Bootstrapping semantic segmentation with regional contrast. In *Int. Conf. Learn. Represent.*, 2022.

[34] W. Liu, D. Ferstl, S. Schulter, L. Zebedin, P. Fua, and C. Leistner. Domain adaptation for semantic segmentation via patch-wise contrastive learning. *arXiv preprint arXiv:2104.11056*, 2021.

[35] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis.*, 2021.

[36] L. Melas-Kyriazi and A. K. Manrai. Pixmatch: Unsupervised domain adaptation via pixelwise consistency training. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[37] Y. Nesterov. A method of solving a convex programming problem with convergence rate $o\left(\frac{1}{k^2}\right)$. *Soviet Mathematics Doklady*, 1983.

[38] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *Eur. Conf. Comput. Vis.*, 2016.

[39] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[40] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[41] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Adv. Neural Inform. Process. Syst.*, 2020.

[42] J.-C. Su, S. Maji, and B. Hariharan. When does self-supervision improve few-shot learning? In *Eur. Conf. Comput. Vis.*, 2020.

[43] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Int. Conf. Mach. Learn.*, 2013.

[44] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Adv. Neural Inform. Process. Syst.*, 2017.

[45] W. Tranheden, V. Olsson, J. Pinto, and L. Svensson. Dacs: Domain adaptation via cross-domain mixed sampling. In *IEEE Winter Conf. Appl. Comput. Vis.*, 2021.

[46] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. Learning to adapt structured output space for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[47] J. E. Van Engelen and H. H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 2020.

[48] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[49] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez. Dada: Depth-aware domain adaptation in semantic segmentation. In *Int. Conf. Comput. Vis.*, 2019.

[50] Q. Wang, D. Dai, L. Hoyer, L. Van Gool, and O. Fink. Domain adaptive semantic segmentation with self-supervised depth estimation. In *Int. Conf. Comput. Vis.*, 2021.

[51] Y. Wang, J. Peng, and Z. Zhang. Uncertainty-aware pseudo label refinery for domain adaptive semantic segmentation. In *Int. Conf. Comput. Vis.*, 2021.

[52] Y. Wang, H. Wang, Y. Shen, J. Fei, W. Li, G. Jin, L. Wu, R. Zhao, and X. Le. Semi-supervised semantic segmentation using unreliable pseudo-labels. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[53] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*, 2022.

[54] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun. Unified perceptual parsing for scene understanding. In *Eur. Conf. Comput. Vis.*, 2018.

[55] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Adv. Neural Inform. Process. Syst.*, 2021.

[56] Z. Yang, W. Bao, D. Yuan, N. H. Tran, and A. Y. Zomaya. Federated learning with nesterov accelerated gradient momentum method. *arXiv preprint arXiv:2009.08716*, 2020.

[57] L. Zhang and G.-J. Qi. Wcp: Worst-case perturbations for semi-supervised deep learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[58] P. Zhang, B. Zhang, T. Zhang, D. Chen, Y. Wang, and F. Wen. Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[59] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[60] Z. Zheng and Y. Yang. Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation. *Int. J. Comput. Vis.*, 2021.

[61] Q. Zhou, Z. Feng, Q. Gu, G. Cheng, X. Lu, J. Shi, and L. Ma. Uncertainty-aware consistency regularization for cross-domain semantic segmentation. *arXiv preprint arXiv:2004.08878*, 2020.

[62] Q. Zhou, Z. Feng, Q. Gu, J. Pang, G. Cheng, X. Lu, J. Shi, and L. Ma. Context-aware mixup for domain adaptive semantic segmentation. *arXiv preprint arXiv:2108.03557*, 2021.

[63] Q. Zhou, C. Zhuang, X. Lu, and L. Ma. Domain adaptive semantic segmentation with regional contrastive consistency regularization. In *Int. Conf. Multimedia and Expo*, 2022.

[64] Y. Zhou, H. Xu, W. Zhang, B. Gao, and P.-A. Heng. C3-semiseg: Contrastive semi-supervised segmentation via cross-set learning and dynamic class-balancing. In *Int. Conf. Comput. Vis.*, 2021.

[65] Z. A. Zhu and L. Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. In *8th Innovations in Theoretical Computer Science Conference*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See abstract and introduction.
   (b) Did you describe the limitations of your work? [Yes] See Sec. 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sec. G in *Supplementary Material*.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Pseudo-code is provided in Sec. A in *Supplementary Material* and data is publicly available. Our implementation will be made publicly available.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Sec. 4.2 and Sec. B in *Supplementary Material*.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Mean and standard deviation are reported.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sec. B in *Supplementary Material*.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See the References.
   (b) Did you mention the license of the assets? [Yes] See Sec. B in *Supplementary Material*.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]