

Confidence Regularized Self-Training

Yang Zou^{1*} Zhiding Yu^{2*} Xiaofeng Liu¹ B.V.K. Vijaya Kumar¹ Jinsong Wang^{3†}

¹ Carnegie Mellon University ² NVIDIA ³ General Motors R&D

✉ yzou2@andrew.cmu.edu, zhidingy@nvidia.com, liuxiaofeng@cmu.edu

Abstract

Recent advances in domain adaptation show that deep self-training presents a powerful means for unsupervised domain adaptation. These methods often involve an iterative process of predicting on target domain and then taking the confident predictions as pseudo-labels for retraining. However, since pseudo-labels can be noisy, self-training can put overconfident label belief on wrong classes, leading to deviated solutions with propagated errors. To address the problem, we propose a confidence regularized self-training (CRST) framework, formulated as regularized self-training. Our method treats pseudo-labels as continuous latent variables jointly optimized via alternating optimization. We propose two types of confidence regularization: label regularization (LR) and model regularization (MR). CRST-LR generates soft pseudo-labels while CRST-MR encourages the smoothness on network output. Extensive experiments on image classification and semantic segmentation show that CRSTs outperform their non-regularized counterpart with state-of-the-art performance. The code and models of this work are available at <https://github.com/yzou2/CRST>.

1. Introduction

Transferring knowledge learned by deep neural networks from label-rich domains to a new target domain is an important but challenging problem. Such domain change naturally occurs in many applications, such as synthetic data training [42, 46] and simulation for robotics/autonomous driving. The existence of cross-domain differences often leads to considerably decreased model performance, and unsupervised domain adaptation (UDA) aims to address this problem by adapting source model to target domain with the aid of unlabeled target data. To this end, a predominant stream of adversarial learning based UDA methods were proposed to reduce the discrepancy between source and target domain features [9, 10, 24, 26, 34, 38, 44, 50, 53, 60].

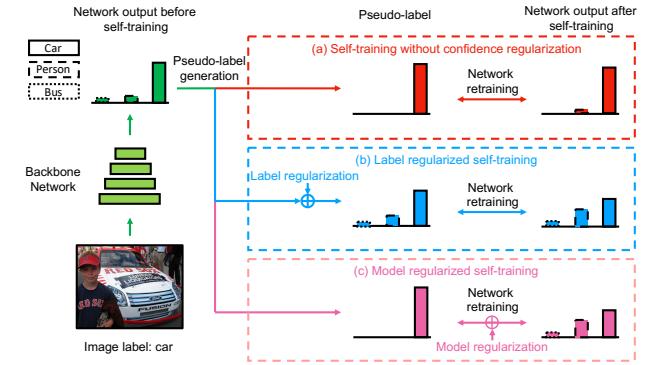


Figure 1: Illustration of proposed confidence regularization. (a) Self-training without confidence regularization generates and retrains with hard pseudo-labels, resulting in sharp network output. (b) Label regularized self-training introduces soft pseudo-labels, therefore enabling outputs to be smooth. (c) Model regularized self-training also retrains with hard pseudo-labels, but incorporates a regularizer to directly promote output smoothness.

More recently, self-training with networks emerged as a promising alternative towards domain adaptation [4, 5, 25, 29, 49, 54, 68]. Self-training iteratively generates a set of one-hot (or hard) pseudo-labels corresponding to large selection scores (i.e., prediction confidence) in target domain, and then retrains network based on these pseudo-labels with target data. Recently, [68] proposes class-balanced self-training (CBST) and formulates self-training as a unified loss minimization with pseudo-labels that can be solved in an end-to-end manner. Instead of reducing domain gap by minimizing both the task loss and domain adversarial loss, the self-training loss implicitly encourages cross-domain feature alignment for each class by learning from both labeled source data and pseudo-labeled target data.

Early work [29] shows that the essence of deep self-training is entropy minimization - pushing network output to be as sharp as hard pseudo-label. However, 100% accuracy cannot always be guaranteed for pseudo-labels. Trusting all selected pseudo-labels as “ground truth” by encoding

*The authors contributed equally.

†Work done during the affiliation with General Motors R&D.

✉Contact emails of corresponding authors.

them as hard labels can lead to overconfident mistakes and propagated errors. In addition, semantic labels of natural images can be highly ambiguous. Taking a sample image from VisDA17 [42] (see Fig. 1) as an example: both person and car dominate significant portions of this image. Enforcing a model to be very confident on only one of the class during training can hurt the learning behavior [2], particularly within the under-determined context of UDA.

The above issues motivate us to prevent infinite entropy minimization in self-training via confidence regularization. A natural idea is to generate soft pseudo-label that redistributes a certain amount of confidence to other classes. Learning with soft pseudo-labels attenuates the misleading effect brought by incorrect or ambiguous supervision. Alternatively, to achieve the same goal, one can also encourage the smoothness of output probabilities and prevent overconfident prediction in network training. Both ideas are illustrated in Fig. 1. At high-level, the major goal of CRST is still aligned with entropy minimization. However, the confidence regularization serves as a safety measure to prevent infinite entropy minimization and degraded performance.

In this work, we choose CBST [68] as a state-of-the-art non-regularized self-training baseline, and propose a variety of specific confidence regularizers to comprehensively validate CRST. Our contributions are listed as follows:

- In Section 3, We generalize CBST to continuous CBST as a necessary preliminary for introducing our CRST, where we relax the feasible space of pseudo-labels from one-hot vectors to a probability simplex,
- In Section 4.1, we introduce label regularized self-training (CRST-LR). CRST-LR generates soft pseudo-labels for self-training. Specifically, we propose a label entropy regularizer (LRENT). In Section 4.2, we introduce model regularized self-training (CRST-MR). CRST-MR introduces an output smoothing regularizer to network training. Specifically, we introduce three model regularizers, including L_2 (MRL2), entropy (MRENT), and KLD (MRKLD).
- In Section 5, we investigate theoretical properties of CRST, and prove that CRST is equivalent to regularized Classification Maximum Likelihood which can be solved via Classification Expectation Maximization (CEM). We also prove the convergence of CRST, and show that LRENT-regularized pseudo-label is equivalent to a generalized softmax with temperature [23].
- In Section 6, we comprehensively evaluate CRST on multiple domain adaptation tasks, including image classification (visDA17/Office-31) and semantic segmentation (GTA5/SYNTHIA → Cityscapes). We demonstrate state-of-the-art or competitive results from the proposed framework, and discuss the comparison between different regularizers in Section 7. We also show that LR+MR may benefit self-training.

2. Related works

Self-training: Self-training has been widely investigated in semi-supervised learning [65, 1, 19]. An overview of different self-training techniques is presented in [59]. Recent interests in self-training were revitalized with deep neural networks [29]. A subtle difference between self-training on fixed features and deep self-training is that the latter involves the learning of embeddings which renders greater flexibility towards domain alignment than classifier-level adaptation. Within this context, [68] proposed class-balanced self-training and achieved state-of-the-art performance in cross-domain semantic segmentation.

Domain adaptation: (Unsupervised) domain adaptation (UDA) has recently gained considerable interests. For UDA with deep networks, a major principle is to let the network learn domain invariant embeddings by minimizing the cross-domain difference of feature distributions with certain criteria. Examples of these methods include maximum mean discrepancy (MMD) [33, 62], deep correlation alignment (CORAL) [56], sliced Wasserstein discrepancy [28], adversarial learning at input-level [24, 17, 13], feature level [8, 15, 24, 31, 50, 61, 64], output space level [60], and a variety of follow up works [10, 34, 44, 53] etc. Open set domain adaptation [40, 52] focuses on the problem where classes are not totally shared between source and target domains. More recently, there have been multiple deep self-training/pseudo-label based methods that are proposed for domain adaptation [4, 21, 25, 49, 54, 68].

Semi-supervised learning (SSL): There exist a natural strong connection between domain adaptation and semi-supervised learning with their problem definitions. A series of teacher-student based approaches have been recently proposed for both SSL [27, 58, 37] and UDA problems[14].

Noisy label learning: Self-training can also be regarded as noisy label learning [39, 45, 55, 66] due to potential mistakes on pseudo-labels. [45] introduced a bootstrapping method for noisy label learning. [55] proposed an extra noise layer into the network adapting the network outputs to match the noisy label distribution.

Network regularization: Regularization is a typical approach in supervised neural network training to avoid overfitting. Besides the standard weight decay, typical regularization techniques include label smoothing [18, 57, 32], network output regularization [43], knowledge distillation [23]. Yet few principled research have considered regularized self-training within the context of SSL/UDA.

3. Continuous class-balanced self-training

In this section, we review the class-balanced self-training algorithm in [68] and reformulate it under a continuous framework. Specifically, for an UDA problem, we have access to the labeled source samples $(\mathbf{x}_s, \mathbf{y}_s)$ from source do-

main $\{\mathbf{X}_S, \mathbf{Y}_S\}$, and target samples \mathbf{x}_t from unlabeled target domain data \mathbf{X}_T . Any target label $\hat{\mathbf{y}}_t = (\hat{y}_t^{(1)}, \dots, \hat{y}_t^{(K)})$ from $\hat{\mathbf{Y}}_T$ is unknown. K is the total number of classes. We define the network weights as \mathbf{w} and $p(k|\mathbf{x}; \mathbf{w})$ as the classifier's softmax probability for class k .

CBST is a self-training framework that performs joint network learning and pseudo-label estimation under a unified loss minimization problem. The pseudo-labels are treated as discrete learnable latent variables being either one-hot or all-zero. Here, we first relax the pseudo-label variables to continuous domain, as shown in Eq. (1):

$$\begin{aligned} \min_{\mathbf{w}, \hat{\mathbf{Y}}_T} \mathcal{L}_{CB}(\mathbf{w}, \hat{\mathbf{Y}}_T) &= - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\ &\quad - \sum_{t \in T} \sum_{k=1}^K \hat{y}_t^{(k)} \log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} \\ \text{s.t. } \hat{\mathbf{y}}_t &\in \Delta^{K-1} \cup \{\mathbf{0}\}, \forall t \end{aligned} \quad (1)$$

The feasible set is the union of $\{\mathbf{0}\}$ and a probability simplex Δ^{K-1} . The continuous CBST is solved by alternating optimization based on the following **a**, **b** steps:

a) Pseudo-label generation Fix \mathbf{w} and solve:

$$\begin{aligned} \min_{\hat{\mathbf{Y}}_T} - \sum_{t \in T} \sum_{k=1}^K \hat{y}_t^{(k)} \log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} \\ \text{s.t. } \hat{\mathbf{y}}_t \in \Delta^{K-1} \cup \{\mathbf{0}\}, \forall t \end{aligned} \quad (2)$$

b) Network retraining Fix $\hat{\mathbf{Y}}_T$ and solve:

$$\begin{aligned} \min_{\mathbf{w}} - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\ - \sum_{t \in T} \sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) \end{aligned} \quad (3)$$

We define going through step **a**) and **b**) once as one “**self-training round**”. For solving step **a**), there is a global optimizer for arbitrary $\hat{\mathbf{y}}_t = (\hat{y}_t^{(1)}, \dots, \hat{y}_t^{(K)})$ as follows.

$$\hat{y}_t^{(k)*} = \begin{cases} 1, & \text{if } k = \arg \max_c \left\{ \frac{p(c|\mathbf{x}_t; \mathbf{w})}{\lambda_c} \right\} \\ & \text{and } p(k|\mathbf{x}_t; \mathbf{w}) > \lambda_k \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

For solving step **b**), one can use typical gradient-based methods such as mini-batch gradient descent. Intuitively, solving **a**) by (4) is actually conducting pseudo-label learning and selection simultaneously. Note that \hat{y}_t^* in (4) not only can be one-hot, but also can be a zero vector $\mathbf{0}$. For each target sample $(\mathbf{x}_t, \hat{\mathbf{y}}_t^*)$, if $\hat{\mathbf{y}}_t^*$ is an one-hot, the sample is selected for model retraining. If $\hat{\mathbf{y}}_t^* = \mathbf{0}$, this sample is

not chosen. Specifically, λ_k is a parameter controlling sample selection. If a sample's predication is relatively confident with $p(k^*|\mathbf{x}_t; \mathbf{w}) > \lambda_{k^*}$, it is selected and labeled as class $k^* = \arg \max_k \left\{ \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} \right\}$. The less confident ones with $p(k^*|\mathbf{x}_t; \mathbf{w}) \leq \lambda_{k^*}$ are not selected.

λ_k are critical parameters to control pseudo-label learning and selection. The same class-balanced λ_k strategy introduced in [68] is adopted for all self-training methods in this work. λ_k for each class k is determined by a single portion parameter p which indicates how many samples we want to select in target domain. Specifically, we define the confidence for a sample as the max of its output softmax probabilities. For each class k , λ_k is determined by the confidence value selecting the most confident p portion of class k predictions in the entire target set. We emphasize that only one parameter p is used to determine all λ_k 's. Practically, we gradually increase p to incorporate more pseudo-labels for each additional round. For detailed algorithm, we recommend to read Algorithm 2 in [68].

Remark: The only difference between CBST and continuous CBST lies in the feasible set where continuous CBST has a probability simplex while CBST has a set of one-hot vectors. Although the feasible set relaxation does not change the solutions of CBST and the pseudo-labels are still one-hot vectors, continuous CBST allows generating soft pseudo-labels if specific regularizers are introduced into pseudo-label generation. Thus it serves as the basis for our proposed label regularized self-training.

4. Confidence regularized self-training

As mentioned in Section 1, we leverage confidence regularization to prevent the over-minimization of entropy that could lead to degraded performance in self-training. Below, we introduce the general definition of CRST:

$$\begin{aligned} \min_{\mathbf{w}, \hat{\mathbf{Y}}_T} \mathcal{L}_{CR}(\mathbf{w}, \hat{\mathbf{Y}}_T) &= \mathcal{L}_{CB}(\mathbf{w}, \hat{\mathbf{Y}}_T) + \alpha \mathcal{R}_C(\mathbf{w}, \hat{\mathbf{Y}}_T) \\ &= - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\ &\quad - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} - \alpha r_c(\mathbf{w}, \hat{\mathbf{y}}_t) \right] \\ \text{s.t. } \hat{\mathbf{y}}_t &\in \Delta^{(K-1)} \cup \{\mathbf{0}\}, \forall t \end{aligned} \quad (5)$$

$\mathcal{R}_C(\mathbf{w}, \hat{\mathbf{Y}}_T) = \sum_{t \in T} r_c(\mathbf{w}, \hat{\mathbf{y}}_t)$ is the confidence regularizer and $\alpha \geq 0$ is the weight coefficient. Similar to CBST, the optimization algorithm of CRST can be formulated as alternatively taking step **a**) pseudo-label generation and step **b**) network retraining. In this paper, we introduce two types of CRST frameworks: label regularized self-training and model regularized self-training.

4.1. Label regularization

The label regularizer has a general form of $\mathcal{R}_C(\hat{\mathbf{Y}}_T) = \sum_{t \in T} r_c(\hat{\mathbf{y}}_t)$ and only depends on pseudo-labels $\{\hat{\mathbf{y}}_t\}$. With fixed \mathbf{w} , the pseudo-label generation in step **a**) of CRST-LR is defined as follows:

$$\begin{aligned} \min_{\hat{\mathbf{Y}}_T} & - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} - \alpha r_c(\hat{\mathbf{y}}_t) \right] \\ \text{s.t. } & \hat{\mathbf{y}}_t \in \Delta^{(K-1)} \cup \{\mathbf{0}\}, \forall t \end{aligned} \quad (6)$$

The global minimizer of (6) can be found via a two-stage optimization given the special structure of the feasible space. The first stage involves minimizing (6) within $\Delta^{(K-1)}$ only, which gives $\hat{\mathbf{y}}_t^\dagger$. The second stage is to select between $\hat{\mathbf{y}}_t^\dagger$ or $\mathbf{0}$ by checking which leads to a lower cost:

$$\hat{\mathbf{y}}_t^* = \begin{cases} \hat{\mathbf{y}}_t^\dagger, & \text{if } \mathcal{C}(\hat{\mathbf{y}}_t^\dagger) < \mathcal{C}(\mathbf{0}) \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (7)$$

where $\mathcal{C}(\hat{\mathbf{y}}_t)$ is the cost of a single sample t in (6):

$$\mathcal{C}(\hat{\mathbf{y}}_t) = -\hat{y}_t^{(k)} \sum_{k=1}^K \log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} + \alpha r_c(\hat{\mathbf{y}}_t) \quad (8)$$

Note that the above regularized term prefers selecting pseudo-labels with certain smoothness rather than sparse ones. In addition, CRST-LR and CBST share the same network retraining strategy in step **b**).

Specifically, we introduce a negative entropy label regularizer (LRENT) in Table 1 with its definition and the corresponding solution of $\hat{\mathbf{y}}_t^\dagger$. For clarity, we write $p(k|\mathbf{x}_t; \mathbf{w})$ as $p(k|\mathbf{x}_t)$ for short. $\hat{\mathbf{y}}_t^\dagger$ can be obtained via solving with a Lagrangian multiplier (KKT conditions) [3]. The detailed derivations are shown in Section A of the Appendix.

4.2. Model regularization

The model regularizer has a general form of $\mathcal{R}_C(\mathbf{w}) = \sum_{t \in T} r_c(p(\mathbf{x}_t; \mathbf{w}))$ where $p(\mathbf{x}_t; \mathbf{w})$ is the network softmax output probabilities. Compared to CBST, CRST-MR has the same hard pseudo-label generation process. But in network retraining of step **b**), CRST-MR uses a cross-entropy loss regularized by an output smoothness encouraging term. We define the optimization problem in step **b**) as follows:

$$\begin{aligned} \min_{\mathbf{w}} & - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\ & - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) - \alpha r_c(p(\mathbf{x}_t; \mathbf{w})) \right] \end{aligned} \quad (9)$$

Specifically, we introduce three model regularizers in Table 1 based on L_2 , negative entropy and KLD between uniform distribution \mathbf{u} and softmax output. The gradients w.r.t. softmax logits z_i are also provided. $H(\mathbf{p})$ is the entropy.

	Regularizer	Pseudo-label solution (LR)/Gradient (MR)
LRENT	$\sum_{k=1}^K \hat{y}_t^{(k)} \log (\hat{y}_t^{(k)})$	$\hat{y}_t^{(i)\dagger} = \frac{(\frac{p(i \mathbf{x}_t)}{\lambda_k})^{\frac{1}{\alpha}}}{\sum_{k=1}^K (\frac{p(k \mathbf{x}_t)}{\lambda_k})^{\frac{1}{\alpha}}}$
MRL2	$\sum_{k=1}^K p(k \mathbf{x}_t)^2$	$2 \sum_{k=1}^K p^2(k \mathbf{x}_t)[\delta_{ki} - p(i \mathbf{x}_t)], \delta_{ki} = \mathbb{1}[k=i]$
MRENT	$\sum_{k=1}^K p(k \mathbf{x}_t) \log p(k \mathbf{x}_t)$	$p(i \mathbf{x}_t)[\log p(i \mathbf{x}_t) + H(p(\mathbf{x}_t))]$
MRKLD	$-\sum_{k=1}^K \frac{1}{K} \log p(k \mathbf{x}_t)$	$p(i \mathbf{x}_t) - \frac{1}{K}$

Table 1: List of proposed regularizers with corresponding pseudo-label solution or gradients w.r.t. softmax logit z_i .

5. Theoretical properties

5.1. A probabilistic view of CRST

There exists an inherent connection between the CRST and some probabilistic models. Specifically, the CRST self-training algorithm can be interpreted as an instance of classification expectation maximization [1]:

Proposition 1. *CRST can be modeled as a regularized classification maximum likelihood (RCML) problem optimized via classification expectation maximization.*

Proof. Please refer to Section B.1 of the Appendix. \square

Proposition 2. *Given pre-determined λ_k , CRST is convergent under certain conditions.*

Proof. Please refer to Section B.2 of the Appendix. \square

5.2. Soft pseudo-label in LRENT

There is an intrinsic connection between the soft pseudo-label of LRENT (given in Table 1) and softmax with temperature. Softmax with temperature [23] is a common approach in neural network for scaling softmax probabilities with applications in knowledge distillation [23], model calibration [20], etc. Typically, networks produce categorical probabilities by a softmax activation layer to convert the logit z_i for each class into a probability $p(i)$. And the softmax with temperature introduces a positive temperature α to scale its smoothness as follows.

$$p(i) = \frac{e^{\frac{z_i}{\alpha}}}{\sum_{k=1, \dots, K} e^{\frac{z_k}{\alpha}}} \quad (10)$$

For high temperature ($\alpha \rightarrow \infty$), the new distribution is softened as a uniform distribution that has the highest entropy and uncertainty. For temperature $\alpha = 1$, we recover the original softmax probabilities. For low temperature ($\alpha \rightarrow 0$), the distribution collapses to a sparse one-hot vector with all probability on the class with the most original softmax probability. Now we draw the connection of soft pseudo-label in LRENT to softmax with temperature:

Proposition 3. If λ_k are equal for all k , the soft pseudo-label of LRENT given in Table 1 is exactly the same as softmax with temperature.

Proof.

$$\begin{aligned}\hat{y}_t^{(i)*} &= \frac{\left(\frac{p(i|x_t)}{\lambda_k}\right)^{\frac{1}{\alpha}}}{\sum_k \left(\frac{p(k|x_t)}{\lambda_k}\right)^{\frac{1}{\alpha}}} = \frac{p(k|x_t)^{\frac{1}{\alpha}}}{\sum_k p(k|x_t)^{\frac{1}{\alpha}}} = \frac{\left[\sum_q e^{z_q}\right]^{\frac{1}{\alpha}}}{\sum_k \left[\sum_q e^{z_q}\right]^{\frac{1}{\alpha}}} \\ &= \frac{(e^{z_i})^{\frac{1}{\alpha}}}{\sum_k (e^{z_k})^{\frac{1}{\alpha}}} = \frac{e^{\frac{z_i}{\alpha}}}{\sum_k e^{\frac{z_k}{\alpha}}}\end{aligned}$$

□

The soft pseudo-label of LRENT can be regarded as a generalized softmax with temperature. In self-training, if selected properly, λ_k can help to generate class-balanced soft pseudo-labels.

Proposition 4. KLD model confidence regularized self-training is equivalent to self-training with pseudo-label uniformly smoothed by $\epsilon = (K\alpha - \alpha)/(K + K\alpha)$, where α is the regularizer weight.

Proof. Please refer to Section B.3 of the Appendix. □

Proposition 5. $D_{KL}(p(\mathbf{x}_t) \parallel \mathbf{u})$ KLD model regularizer (the reverse of the proposed $D_{KL}(\mathbf{u} \parallel p(\mathbf{x}_t))$ KLD regularizer) is equivalent to entropy model regularizer $-H(p(\mathbf{x}_t))$, where \mathbf{u} is the uniform distribution.

Proof. Please refer to Section B.4 of the Appendix. □

6. Experiments

In this section, we conduct comprehensive evaluation on different domain adaptation tasks.

Adaptation for image classification: We consider two adaptation benchmarks: 1) VisDA17 [42] and 2) Office-31 [48]. VisDA17 contains 152, 409 2D synthetic images of 12 classes in the source training set and 55, 400 real images from MS-COCO [30] as the target domain validation set. Office-31 is a small-scale dataset containing images of 31 classes from three domains - Amazon (A), Webcam (W) and DSLR (D). Each domain contains 2, 817, 795 and 498 images respectively. We follow the standard protocol in [48, 53] and evaluate on six transfer tasks $A \rightarrow W$, $D \rightarrow W$, $W \rightarrow D$, $A \rightarrow D$, $D \rightarrow A$, and $W \rightarrow A$.

Adaptation for semantic segmentation: We consider two popular synthetic-to-real adaptation scenarios: 1) GTA5 [46] to Cityscapes [11], and 2) SYNTHIA [47] to Cityscapes. The GTA5 dataset includes 24, 966 images rendered by GTA5 game engine. For SYNTHIA, we choose SYNTHIA-RAND-CITYSCAPES which includes 9, 400 labeled images. Following the standard protocols [24, 60],

we adapt the model to the Cityscapes training set and evaluate the performance on the validation set.

To comprehensively demonstrate the improvement of CRST, we report the performance of CRST with all regularizers and compare with CBST in each task.

6.1. Implementation details

Image classification: For VisDA17/Office-31, we implement CBST/CRSTs using PyTorch [41] and choose ResNet-101/ResNet-50 [22] as backbones. For fair comparison, we compare to other works with the same backbone networks. Both backbones are pre-trained on ImageNet [12], and then fine-tuned on source domain using SGD, with learning rate 1×10^{-3} , weight decay 5×10^{-4} , momentum 0.9 and batch size 32. For self-training, we apply the same training strategy but a different learning rate 1×10^{-4} .

Semantic segmentation: For semantic segmentation, we further consider DeepLabv2 [6] as a backbone besides the ResNet-38 backbone in [68]. For experiments with DeepLabv2, we implement CBST/CRSTs using PyTorch, while following the MXNet [7] implementation of [68] for experiments with ResNet-38. DeepLabv2 is pre-trained on ImageNet and fine-tuned on source domain using SGD, with learning rate 2.5×10^{-4} , weight decay 5×10^{-4} , momentum 0.9, batch size 2, patch size 512×1024 , multi-scale training augmentation (0.5 – 1.5) and horizontal flipping. In self-training, we apply SGD with learning rate of 5×10^{-5} . For fair comparison, we unify the total number of self-training rounds to be 3, each with 2 re-training epochs.

6.2. Domain adaptation for image classification

VisDA17: We present the results on VisDA17 in Table 2 in terms of per-class accuracy and mean accuracy. For each proposed approach, we report the averages and standard deviations of the evaluation results over 5 runs. Note that both MRKLD and LRENT outperform the non-regularized CBST, whereas MRL2 and MRENT show slightly worse results. Among CRSTs with single regularizer, MRKLD achieves the best performance with considerable improvement. The combination of MRKLD and LRENT further outperforms single regularizers and other recently proposed methods. The result even outperforms certain methods with stronger backbones (ResNet-152) [44, 53].

Office-31: We compare the performance of different methods on Office-31 with the same backbone ResNet-50 in Table 3. All CRSTs achieve similar results that outperform the baseline CBST. In addition, MRKLD+LRENT again outperforms single regularizers, achieving comparable or better performance compared with other recent methods.

6.3. Domain adaptation for semantic segmentation

GTA5 → Cityscapes: Table 4 shows the adaptation performance of CRSTs and other comparing methods. On a

Method	Aero	Bike	Bus	Car	Horse	Knife	Motor	Person	Plant	Skateboard	Train	Truck	Mean
Source [50]	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
MMD [33]	87.1	63.0	76.5	42.0	90.3	42.9	85.9	53.1	49.7	36.3	85.8	20.7	61.1
DANN [16]	81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
ENT [19]	80.3	75.5	75.8	48.3	77.9	27.3	69.7	40.2	46.5	46.6	79.3	16.0	57.0
MCD [51]	87.0	60.9	83.7	64.0	88.9	79.6	84.7	76.9	88.6	40.3	83.0	25.8	71.9
ADR [50]	87.8	79.5	83.7	65.3	92.3	61.8	88.9	73.2	87.8	60.0	85.5	32.3	74.8
SimNet-Res152 [44]	94.3	82.3	73.5	47.2	87.9	49.2	75.1	79.7	85.3	68.5	81.1	50.3	72.9
GTA-Res152 [53]	-	-	-	-	-	-	-	-	-	-	-	-	77.1
Source-Res101	68.7	36.7	61.3	70.4	67.9	5.9	82.6	25.5	75.6	29.4	83.8	10.9	51.6
CBST	87.2±2.4	78.8±1.0	56.5±2.2	55.4±3.6	85.1±1.4	79.2±10.3	83.8±0.4	77.7±4.0	82.8±2.8	88.8±3.2	69.0±2.9	72.0±3.8	76.4±0.9
MRL2	87.0±2.9	79.5±1.9	57.1±3.2	54.7±2.9	85.5±1.1	78.1±11.7	83.0±1.5	77.7±3.7	82.4±1.7	88.6±2.7	69.1±2.2	71.8±3.0	76.2±1.0
MRENT	87.1±2.7	78.3±0.7	56.1±4.0	54.4±2.7	84.4±2.3	79.9±10.6	83.7±1.1	77.9±4.4	82.7±2.4	87.4±2.8	70.0±1.4	72.8±3.3	76.2±0.8
MRKLD	87.3±2.5	79.4±1.9	60.5±2.4	59.7±2.5	87.6±1.4	82.4±4.4	86.5±1.1	78.4±2.6	84.6±1.7	86.4±2.8	72.5±2.4	69.8±2.5	77.9±0.5
LRENT	87.7±2.4	78.7±0.8	57.3±3.3	54.5±4.0	84.8±1.7	79.7±10.3	84.2±1.4	77.4±3.7	83.1±1.5	88.3±2.6	70.9±2.1	72.6±2.4	76.6±0.9
MRKLD+LRENT	88.0±0.6	79.2±2.2	61.0±3.1	60.0±1.0	87.5±1.2	81.4±5.6	86.3±1.5	78.8±2.1	85.6±0.9	86.6±2.5	73.9±1.3	68.8±2.3	78.1±0.2

Table 2: Experimental results on VisDA17.

Method	A→W	D→W	W→D	A→D	D→A	W→A	Mean
ResNet-50 [22]	68.4±0.2	96.7±0.1	99.3±0.1	68.9±0.2	62.5±0.3	60.7±0.3	76.1
DAN [33]	80.5±0.4	97.1±0.2	99.6±0.1	78.6±0.2	63.6±0.3	62.8±0.2	80.4
RTN [35]	84.5±0.2	96.8±0.1	99.4±0.1	77.5±0.3	66.2±0.2	64.8±0.3	81.6
DANN [16]	82.0±0.4	96.9±0.2	99.1±0.1	79.7±0.4	68.2±0.4	67.4±0.5	82.2
ADDA [61]	86.2±0.5	96.2±0.3	98.4±0.3	77.8±0.3	69.5±0.4	68.9±0.5	82.9
JAN [36]	85.4±0.3	97.4±0.2	99.8±0.2	84.7±0.3	68.6±0.3	70.0±0.4	84.3
GTA [53]	89.5±0.5	97.9±0.3	99.8±0.4	87.7±0.5	72.8±0.3	71.4±0.4	86.5
CBST	87.8±0.8	98.5±0.1	100±0.0	86.5±1.0	71.2±0.4	70.9±0.7	85.8
MRL2	88.4±0.2	98.6±0.1	100±0.0	87.7±0.9	71.8±0.2	72.1±0.2	86.4
MRENT	88.0±0.4	98.6±0.1	100±0.0	87.4±0.8	72.7±0.2	71.0±0.4	86.4
MRKLD	88.4±0.9	98.7±0.1	100±0.0	88.0±0.9	71.7±0.8	70.9±0.4	86.3
LRENT	88.6±0.4	98.7±0.1	100±0.0	89.0±0.8	72.0±0.6	71.0±0.3	86.6
MRKLD+LRENT	89.4±0.7	98.9±0.4	100±0.0	88.7±0.8	72.6±0.7	70.9±0.5	86.8

Table 3: Experimental results on Office-31.

DeepLabv2 backbone, one could see that MRKLD achieves the best result outperforming previous state-of-the-art. In addition, Fig. 10 visualizes the adapted prediction results obtained by CBST and CRSTs on Cityscapes validation set. Fig. 11 further compares the pseudo-label maps in the second round of self-training. On a wide ResNet-38 backbone, all CRSTs outperform the baseline CBST and we achieve the state-of-the-art system-level performance with the spatial priors (SP) and multi-scale testing (MST) from [68].

SYNTHIA → Cityscapes: Table 5 shows the adaptation results where CRSTs again show the performance on par with or better than the baseline CBST. In particular, MRKLD maintains the best performance among all regularizers and outperforms the previous state-of-the-art [68].

6.4. Parameter analysis

p is an important parameter controlling the pseudo-label generation and selection sensitivity. We adopt the same p policy as [68] where we start p from 20%, and empirically add 5% to p in each additional self-training round. We conduct a sensitivity analysis for portion p similar to [68], where we consider the starting portion p_0 and the incremental portion Δp on a difficult task of Office-31: W → A. Table 6 shows that CRSTs are not sensitive to p_0 and Δp .

In CRST, the coefficient α is an important parameter that balances the weight between self-training loss and confidence regularizer. In all the experiments, we unify α to be 0.025, 0.1, 0.1, 0.25 for MRL2, MRENT, MRKLD and LRENT, respectively. Note that various regularizers have different α due to their intrinsic differences. We also present

the sensitivity analysis of α on W → A in Table 7. We can see all CRSTs are not sensitive to α in certain intervals.

7. Discussion

7.1. How does confidence regularization work?

Confidence regularization smooths the output by lowering the confidence (the max of output softmax) and raising the probability level of other classes. Such smoothing helps to reduce the confidence on false positives (FP), although the confidence of certain true positives (TP) may also decrease. To see the change w/o CR, we compare CBST vs MRKLD/LRENT (DeepLabv2) on GTA5 → Cityscapes, by presenting their per-class mean confidence of TP (C_{TP}), mean confidence of FP (C_{FP}) and the C_{TP}/C_{FP} ratios at the end of first round in Table 8. For both TP and FP, the confidence of MRKLD/LRENT are lower than CBST, but either MRKLD or LRENT outperforms CBST on almost all per-class ratios and mean ratios. This intuitively illustrates how confidence regularization benefits self-training.

7.2. MR versus LR

We analyze MR/LR intuitively and theoretically to give suggestions for practical choice of confidence regularizers.

Complexity analysis: All model regularizers only introduce negligible extra costs for the gradient computation. Label regularizers, however, requires the storage of dataset-level soft pseudo-labels. This does not present an issue in image classification but may introduce extra I/O costs in segmentation, where labels are often too large to be stored in memory and need to be written to disk.

Loss curves: To further illustrate the different properties of regularizers, we visualize how they influence the original loss surfaces by reducing the problem into binary classification with a single sample. We assume a cross-entropy loss $-y \log p - (1-y) \log(1-p)$ plus an MR/LR weighted by α . For MRs, we assume $y = 1$ and illustrate the regularized loss curves versus p in Fig. 4. For all MRs, p^* becomes smoother when α increases. We notice that MRKLD serves as a better barrier to prevent sharp outputs than other MRs

Method	Backbone	Road	SW	Build	Wall	Fence	Pole	TL	TS	Veg.	Terrain	Sky	PR	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU
Source CyCADA [24]	DRN-26	42.7	26.3	51.7	5.5	6.8	13.8	23.6	6.9	75.5	11.5	36.8	49.3	0.9	46.7	3.4	5.0	0.0	5.0	1.4	21.7
		79.1	33.1	77.9	23.4	17.3	32.1	33.3	31.8	81.5	26.7	69.0	62.8	14.7	74.5	20.9	25.6	6.9	18.8	20.4	39.5
Source MCD [51]	DRN-105	36.4	14.2	67.4	16.4	12.0	20.1	8.7	0.7	69.8	13.3	56.9	37.0	0.4	53.6	10.6	3.2	0.2	0.9	0.0	22.2
		90.3	31.0	78.5	19.7	17.3	28.6	30.9	16.1	83.7	30.0	69.1	58.5	19.6	81.5	23.8	30.0	5.7	25.7	14.3	39.7
Source AdaptSegNet [60]	DeepLabv2	75.8	16.8	77.2	12.5	21.0	25.5	30.1	20.1	81.3	24.6	70.3	53.8	26.4	49.9	17.2	25.9	6.5	25.3	36.0	36.6
		86.5	36.0	79.9	23.4	23.3	23.9	35.2	14.8	83.4	33.3	75.6	58.5	27.6	73.7	32.5	35.4	3.9	30.1	28.1	42.4
AdvEnt [63]	DeepLabv2	89.4	33.1	81.0	26.6	26.8	27.2	33.5	24.7	83.9	36.7	78.8	58.7	30.5	84.8	38.5	44.5	1.7	31.6	32.4	45.5
		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	29.2	
Source FCAN [67]	DeepLabv2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	46.6	
		71.3	19.2	69.1	18.4	10.0	35.7	27.3	6.8	79.6	24.8	72.1	57.6	19.5	55.5	15.5	15.1	11.7	21.1	12.0	33.8
Source CBST [68]	DeepLabv2	91.8	53.5	80.5	32.7	21.0	34.0	28.9	20.4	83.9	34.2	80.9	53.1	24.0	82.7	30.3	35.9	16.0	25.9	42.8	45.9
		91.9	55.2	80.9	32.1	21.5	36.7	30.0	19.0	84.8	34.9	80.1	56.1	23.8	83.9	28.0	29.4	20.5	24.0	40.3	46.0
		91.8	53.4	80.6	32.6	20.8	34.3	29.7	21.0	84.0	34.1	80.6	53.9	24.6	82.8	30.8	34.9	16.6	26.4	42.6	46.1
		91.0	55.4	80.0	33.7	21.4	37.3	32.9	24.5	85.0	34.1	80.8	57.7	24.6	84.1	27.8	30.1	26.9	26.0	42.3	47.1
		91.8	53.5	80.5	32.7	21.0	34.0	29.0	20.3	83.9	34.2	80.9	53.1	23.9	82.7	30.2	35.6	16.3	25.9	42.8	45.9
		70.0	23.7	67.8	15.4	18.1	40.2	41.9	25.3	78.8	11.7	31.4	62.9	29.8	60.1	21.5	26.8	7.7	28.1	12.0	35.4
Source CBST [68]	ResNet-38	86.8	46.7	76.9	26.3	24.8	42.0	46.0	38.6	80.7	15.7	48.0	57.3	27.9	78.2	24.5	49.6	17.7	25.5	45.1	45.2
		84.4	52.7	74.7	38.0	32.2	43.7	53.7	38.6	73.9	24.4	64.4	45.6	24.6	63.2	3.2	31.9	45.9	44.2	34.8	46.0
		84.6	49.5	73.9	35.8	25.1	46.2	53.3	43.3	75.2	24.2	63.8	48.2	33.8	65.7	2.9	32.6	50.0	34.7	46.4	
		84.5	47.7	74.1	27.9	22.1	43.8	46.5	37.8	83.7	22.7	56.1	56.8	26.8	81.7	22.5	46.2	27.5	32.3	47.9	46.8
		80.3	40.8	65.8	24.6	30.5	43.1	49.5	40.3	82.1	26.0	54.6	59.4	32.1	68.0	31.9	30.0	21.9	44.8	46.7	45.9
		85.6	55.1	76.9	26.8	23.4	38.9	47.1	46.9	83.4	25.5	68.7	45.6	15.7	79.7	27.7	50.3	38.2	33.4	44.6	48.1
MRKLD-SP	ResNet-38	90.8	46.0	79.9	27.4	23.3	42.3	46.2	40.9	83.5	19.2	59.1	63.5	30.8	83.5	36.8	52.0	28.0	36.8	46.4	49.2
		91.7	45.1	80.9	29.0	23.4	43.8	47.1	40.9	84.0	20.0	60.6	64.0	31.9	85.8	39.5	48.7	25.0	38.0	47.0	49.8

Table 4: Experimental results on GTAS5 → Cityscapes.

Method	Backbone	Road	SW	Build	Wall*	Fence*	Pole*	TL	TS	Veg.	Sky	PR	Rider	Car	Truck	Bus	Motor	Bike	mIoU	mIoU*
Source MCD [51]	DRN-105	14.9	11.4	58.7	1.9	0.0	24.1	1.2	6.0	68.8	76.0	54.3	7.1	34.2	15.0	0.8	0.0	23.4	26.8	
		84.8	43.6	79.0	3.9	0.2	29.1	7.2	5.5	83.8	83.1	51.0	11.7	79.9	27.2	6.2	0.0	37.3	43.5	
Source AdaptSegNet [60]	DeepLabv2	55.6	23.8	74.6	—	—	—	6.1	12.1	74.8	79.0	55.3	19.1	39.6	23.3	13.7	25.0	—	38.6	
		84.3	42.7	77.5	—	—	—	4.7	7.0	77.9	82.5	54.3	21.0	72.3	32.2	18.9	32.3	—	46.7	
AdvEnt [63]	DeepLabv2	85.6	42.2	79.7	8.7	0.4	25.9	5.4	8.1	80.4	84.1	57.9	23.8	73.3	36.4	14.2	33.0	41.2	48.0	
		32.6	21.5	46.5	4.8	0.1	26.5	14.8	13.1	70.8	60.3	56.6	3.5	74.1	20.4	8.9	13.1	29.2	33.6	
Source CBST [68]	ResNet-38	53.6	23.7	75.0	12.5	0.3	36.4	23.5	26.3	84.8	74.7	67.2	17.5	84.5	28.4	15.2	55.8	42.5	48.4	
		64.3	21.3	73.1	2.4	1.1	31.4	7.0	27.7	63.1	67.6	42.2	19.9	73.1	15.3	10.5	38.9	34.9	40.3	
Source CBST [68]	DeepLabv2	68.0	29.9	76.3	10.8	1.4	33.9	22.8	29.5	77.6	78.3	60.6	28.3	81.6	23.5	18.8	39.8	42.6	48.9	
		63.4	27.1	76.4	14.2	1.4	35.2	23.6	29.4	78.5	77.8	61.4	29.5	82.2	22.8	18.9	42.3	42.8	48.7	
		69.6	32.6	75.8	12.2	1.8	35.3	23.3	29.5	77.7	78.9	60.0	28.5	81.5	25.9	19.6	41.8	43.4	49.6	
		67.7	32.2	73.9	10.7	1.6	37.4	22.2	31.2	80.8	80.5	60.8	29.1	82.8	25.0	19.4	45.3	43.8	50.1	
		65.6	30.3	74.6	13.8	1.5	35.8	23.1	29.1	77.0	77.5	60.1	28.5	82.2	22.6	20.1	41.9	42.7	48.7	
		85.6	55.1	76.9	26.8	23.4	38.9	47.1	46.9	83.4	25.5	68.7	45.6	15.7	79.7	27.7	50.3	38.2	33.4	44.6

Table 5: Experimental results on SYNTHIA → Cityscapes.

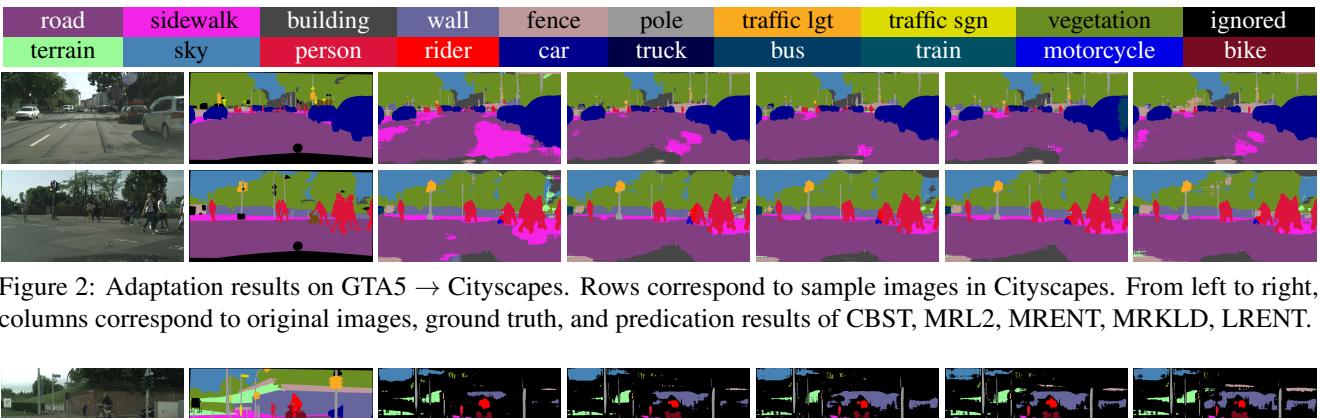


Figure 2: Adaptation results on GTA5 → Cityscapes. Rows correspond to sample images in Cityscapes. From left to right, columns correspond to original images, ground truth, and predication results of CBST, MRL2, MRENT, MRKLD, LRENT.

by having steeper gradient near $p = 1$. This accords with our observation that MRKLD overall works the best. For LRENT, we assume $p = 0.9$ and illustrate the regularized loss curves versus y at different α in Fig. 4. Again, y^* becomes smoother when α increases.

Class ranking: Based on the closed-form solution of LR in Table 1, we can prove that LR preserves the confidence ranking order between classes. On the other hand, given one-hot labels, MRs tend to discard such order information by giving equal confidences to negative classes. Tak-

W → A (Office-31)											
$p_0/\Delta p$ Accuracy	MRL2					MRENT					
	20/5	15/5	25/5	20/2.5	20/7.5	20/5	15/5	25/5	20/2.5	20/7.5	
MRKLD											
$p_0/\Delta p$ Accuracy	20/5	15/5	25/5	20/2.5	20/7.5	20/5	15/5	25/5	20/2.5	20/7.5	
	72.0±0.4	70.8±0.4	70.7±0.2	70.9±0.5	71.0±0.8	71.0±0.3	71.0±0.8	71.2±0.6	71.1±0.5	71.0±0.4	

Table 6: Sensitivity analysis of portion p_0 and portion step Δp .

W → A (Office-31)												
α Accuracy	MRL2			MRENT			MRKLD			LRENT		
	0.01	0.025	0.05	0.075	0.1	0.125	0.075	0.1	0.125	0.1	0.25	0.5
	71.5±0.8	72.1±0.2	71.7±1.1	71.0±0.8	71.0±0.4	70.9±1.0	70.9±0.6	70.9±0.4	70.6±0.7	71.2±1.2	71.0±0.3	70.8±0.6

Table 7: Sensitivity analysis of regularizer weight α .

	Road	SW	Build	Wall	Fence	Pole	TL	TS	Veg.	Terrain	Sky	PR	Rider	Car	Truck	Bus	Train	Motor	Bike	mean	
CBST	C_{TP} (%)	96.2	86.0	94.6	83.8	84.9	84.5	80.4	78.0	93.9	87.9	94.5	90.4	81.4	95.4	88.4	85.9	59.5	78.5	80.6	85.5
	C_{FP} (%)	72.2	74.1	69.8	71.7	76.7	73.7	72.9	76.5	71.9	71.2	68.5	67.2	69.1	66.1	76.9	65.5	76.7	67.2	73.0	71.6
	C_{TP}/C_{FP}	1.33	1.16	1.36	1.17	1.11	1.15	1.10	1.02	1.31	1.23	1.38	1.35	1.18	1.44	1.15	1.31	0.78	1.17	1.10	1.19
MRKLD	C_{TP} (%)	94.7	82.8	92.4	81.7	77.8	84.4	77.0	76.4	93.4	86.5	94.4	88.8	79.7	93.9	87.0	84.9	71.9	77.6	79.2	84.5
	C_{FP} (%)	67.7	70.3	65.4	68.5	69.2	66.7	69.4	71.3	66.7	68.8	66.7	60.0	65.5	63.0	74.6	63.6	70.2	59.3	53.2	66.3
	C_{TP}/C_{FP}	1.40	1.18	1.41	1.19	1.12	1.27	1.11	1.07	1.40	1.26	1.42	1.48	1.22	1.49	1.17	1.34	1.02	1.31	1.49	1.27
LRENT	C_{TP} (%)	95.9	84.4	94.0	80.7	75.3	84.8	77.8	78.3	93.9	86.3	94.5	89.2	79.3	95.3	89.3	80.5	76.4	86.4	78.8	85.3
	C_{FP} (%)	69.5	72.1	68.0	67.8	71.3	69.7	71.5	75.4	69.5	69.9	70.1	64.1	67.6	67.3	77.7	70.3	63.4	58.6	55.2	68.4
	C_{TP}/C_{FP}	1.38	1.17	1.38	1.19	1.06	1.22	1.09	1.04	1.35	1.23	1.35	1.39	1.17	1.42	1.15	1.15	1.2	1.47	1.43	1.25

Table 8: Comparison of C_{TP} , C_{FP} and C_{TP}/C_{FP} on GTA5 → Cityscapes.

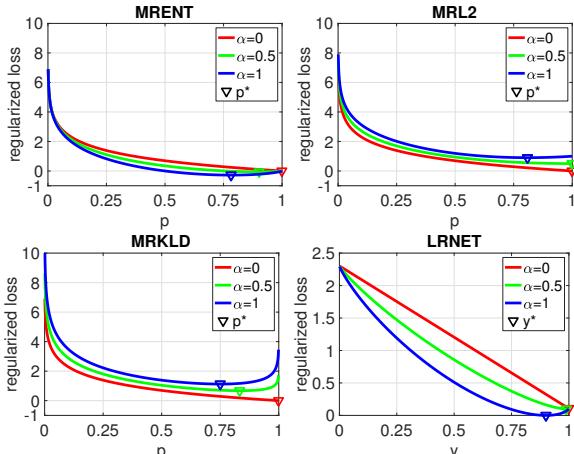


Figure 4: Loss curves regularized by different regularizers.

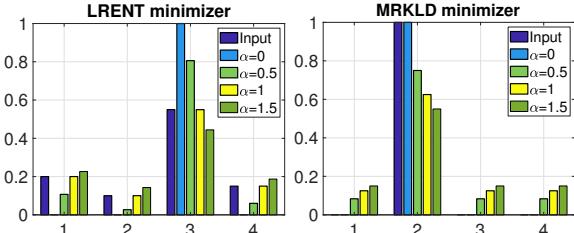


Figure 5: Minimizers of LRENT and MRKLD.

ing MRKLD as example: using Lagrangian multiplier, we can prove the closed-form global minimizer for regularized cross-entropy loss as $p^{*(k)} = (y^{(k)} + \frac{\alpha}{K}) / (1 + \alpha)$, where $k = 1, \dots, K$ is class index. With y being one-hot, the global minimizer is uniformly smoothed on negative classes. Similar property can be also proved for MRENT/MRL2.

We illustrate two examples of LRENT and MRKLD in Fig. 5, where we assume $\mathbf{p} = [0.2, 0.1, 0.55, 0.15]$ for LRENT and $y^{(2)} = 1$ for MRKLD. One can see, LRENT sharpens the input \mathbf{p} when $\alpha \in [0, 1]$ (one-hot when $\alpha = 0$), while smooths \mathbf{p} when $\alpha > 1$. In all cases, the inter-class confidence orders are always preserved, while the same property does not hold for MRKLD.

MR+LR: The combination of MR and LR can take advantages of both regularizers and achieve better performance compared to single regularizer, demonstrated in VisDA17 and Office-31. However, it will also introduce extra cost to validate both hyperparameters for MR and LR.

Practical suggestions: Overall, we recommend CRST-MRKLD most based on the above analysis and its better performance. Moreover, combining MR and LR may also benefit self-training at the cost of slight extra tuning.

8. Conclusions

In this paper, we introduce a confidence regularized self-training framework formulated as regularized self-training loss minimization. Model regularization and label regularization are considered with a family of proposed confidence regularizers. We investigate theoretical properties of CRST, including its probabilistic explanation and connection to softmax with temperature. Comprehensive experiments demonstrate the effectiveness of CRST with state-of-the-art performance. We also systematically discuss the pros and cons of the proposed regularizers and made practical suggestions. We believe this work can inspire future research on novel designs of regularizations as desired inductive biases to benefit many UDA/SSL problems.

References

- [1] Massih-Reza Amini and Patrick Gallinari. Semi-supervised logistic regression. In *ECAI*, 2002. 2, 4, 11
- [2] Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *arXiv:1805.02641*, 2018. 2
- [3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 4, 11
- [4] Pau Panareda Busto, Ahsan Iqbal, and Juergen Gall. Open set domain adaptation for image and action recognition. *IEEE Trans. PAMI*, 2018. 1, 2
- [5] Chaoqi Chen, Weiping Xie, Wenbing Huang, Yu Rong, Xinghao Ding, Yue Huang, Tingyang Xu, and Junzhou Huang. Progressive feature alignment for unsupervised domain adaptation. In *CVPR*, 2019. 1
- [6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. PAMI*, 2018. 5
- [7] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015. 5
- [8] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In *CVPR*, 2019. 2
- [9] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *CVPR*, 2018. 1
- [10] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *ICCV*, 2017. 1, 2
- [11] 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 *CVPR*, 2016. 5
- [12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5
- [13] Aysegul Dundar, Ming-Yu Liu, Zhiding Yu, Ting-Chun Wang, John Zedlewski, and Jan Kautz. Domain stylization: A fast covariance matching framework towards domain adaptation. *IEEE Trans. PAMI*, 2020. 2
- [14] Geoffrey French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. *ICLR*, 2018. 2
- [15] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 2
- [16] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016. 6
- [17] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. DLOW: Domain flow for adaptation and generalization. In *CVPR*, 2019. 2
- [18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>. 2
- [19] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NeurIPS*, 2005. 2, 6
- [20] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 4
- [21] Ligong Han, Yang Zou, Ruijiang Gao, Lezi Wang, and Dimitris Metaxas. Unsupervised domain adaptation via calibrating uncertainties. In *CVPR Workshops*, 2019. 2
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 6
- [23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015. 2, 4
- [24] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018. 1, 2, 5, 7
- [25] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation. In *CVPR*, 2018. 1, 2
- [26] Minyoung Kim, Pritish Sahu, Behnam Gholami, and Vladimir Pavlovic. Unsupervised visual domain adaptation: A deep max-margin Gaussian process approach. In *CVPR*, 2019. 1
- [27] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2016. 2
- [28] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *CVPR*, 2019. 2
- [29] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013. 1, 2
- [30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5
- [31] Xiaofeng Liu, Site Li, Lingsheng Kong, Wanqing Xie, Ping Jia, Jane You, and BVK Kumar. Feature-level frankenstein: Eliminating variations for discriminative recognition. In *CVPR*, 2019. 2
- [32] Xiaofeng Liu, Yang Zou, Tong Che, Peng Ding, Ping Jia, Jane You, and B.V.K. Kumar. Conservative wasserstein training for pose estimation. In *ICCV*, 2017. 2
- [33] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015. 2, 6

- [34] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018. 1, 2
- [35] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *NeurIPS*, 2016. 6
- [36] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017. 6
- [37] Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. In *CVPR*, 2018. 2
- [38] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *CVPR*, 2018. 1
- [39] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *NeurIPS*, 2013. 2
- [40] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *ICCV*, 2017. 2
- [41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5
- [42] Xingchao Peng, Ben Usman, Neela Kaushik, Dequan Wang, Judy Hoffman, Kate Saenko, Xavier Roynard, Jean-Emmanuel Deschaud, Francois Fleuret, Tyler L Hayes, et al. Visda: A synthetic-to-real benchmark for visual domain adaptation. In *CVPR Workshops*, 2018. 1, 2, 5
- [43] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In *ICLR Workshop*, 2017. 2, 13
- [44] Pedro O Pinheiro. Unsupervised domain adaptation with similarity learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8004–8013, 2018. 1, 2, 5, 6
- [45] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR*, 2015. 2
- [46] Stephan Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 1, 5
- [47] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 5
- [48] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, 2010. 5
- [49] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *ICML*, 2017. 1, 2
- [50] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial dropout regularization. In *ICLR*, 2018. 1, 2, 6
- [51] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. 2017. 6, 7
- [52] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *ECCV*, 2018. 2
- [53] Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*, 2018. 1, 2, 5, 6
- [54] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *ICLR*, 2018. 1, 2
- [55] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv:1406.2080*, 2014. 2
- [56] Baichen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016. 2
- [57] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 2, 12
- [58] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 2
- [59] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 2015. 2
- [60] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Ki-hyun Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *CVPR*, 2018. 1, 2, 5, 7
- [61] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017. 2, 6
- [62] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv:1412.3474*, 2014. 2
- [63] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *CVPR*, 2019. 7
- [64] Zuxuan Wu, Xintong Han, Yen-Liang Lin, Mustafa Gokhan Uzunbas, Tom Goldstein, Ser Nam Lim, and Larry S Davis. Dcan: Dual channel-wise alignment networks for unsupervised scene adaptation. In *ECCV*, 2018. 2
- [65] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995. 2
- [66] Zhiding Yu, Weiyang Liu, Yang Zou, Chen Feng, Srikanth Ramalingam, B. V. K. Vijaya Kumar, and Jan Kautz. Simultaneous edge alignment and learning. In *ECCV*, 2018. 2
- [67] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. Fully convolutional adaptation networks for semantic segmentation. In *CVPR*, 2018. 7
- [68] Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018. 1, 2, 3, 5, 6, 7

Appendix

In this appendix, we present the additional details and results that are not covered by the main paper.

A. Derivation of soft pseudo-label in LRENT

For entropy label regularizer, the soft pseudo-label learning problem is defined as follows.

$$\begin{aligned} \min_{\hat{\mathbf{y}}_t} \sum_{k=1}^K -\hat{y}_t^{(k)} \log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} + \alpha \sum_{k=1}^K \hat{y}_t^{(k)} \log(\hat{y}_t^{(k)}) \\ \text{s.t. } \hat{\mathbf{y}}_t \in \Delta^{(K-1)} \end{aligned} \quad (11)$$

where the solution is given as below.

$$\hat{y}_t^{(i)\dagger} = \frac{\left(\frac{p(i|\mathbf{x}_t)}{\lambda_i}\right)^{\frac{1}{\alpha}}}{\sum_{k=1}^K \left(\frac{p(k|\mathbf{x}_t)}{\lambda_k}\right)^{\frac{1}{\alpha}}}$$

It is easy to see that the optimization in (11) is a convex problem. Therefore, the global optimum can be found with a Lagrangian multiplier [3] defined as follows:

$$\begin{aligned} L(\hat{\mathbf{y}}, \beta) = \sum_{k=1}^K -\hat{y}_t^{(k)} \log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} \\ + \alpha \sum_{k=1}^K \hat{y}_t^{(k)} (\log(\hat{y}_t^{(k)}) - 1) + \beta \left(\sum_{k=1}^K \hat{y}_t^{(k)} - 1\right) \end{aligned}$$

Setting the corresponding gradients equals to 0 gives the global optimum ($k = 1, \dots, K$).

$$\begin{aligned} & \left\{ \begin{array}{l} \frac{\partial L}{\partial \hat{y}_t^{(i)\dagger}} = -\log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_i} + \alpha \log \hat{y}_t^{(i)\dagger} + \beta = 0; \\ \sum_{k=1}^K \hat{y}_t^{(k)\dagger} = 1 \end{array} \right. \\ \Leftrightarrow & \left\{ \begin{array}{l} \hat{y}_t^{(i)\dagger} = \exp\left(\frac{-\beta}{\alpha}\right) \left(\frac{p(i|\mathbf{x}_t; \mathbf{w})}{\lambda_i}\right)^{\frac{1}{\alpha}}; \\ \sum_{i=1}^K \hat{y}_t^{(i)\dagger} = 1 \end{array} \right. \\ \Leftrightarrow & \left\{ \begin{array}{l} \hat{y}_t^{(i)\dagger} = \exp\left(\frac{-\beta}{\alpha}\right) \left(\frac{p(i|\mathbf{x}_t; \mathbf{w})}{\lambda_i}\right)^{\frac{1}{\alpha}}; \\ \sum_{i=1}^K \exp\left(\frac{-\beta}{\alpha}\right) \left(\frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k}\right)^{\frac{1}{\alpha}} = 1 \end{array} \right. \\ \Leftrightarrow & \left\{ \begin{array}{l} \hat{y}_t^{(i)\dagger} = \exp\left(\frac{-\beta}{\alpha}\right) \left(\frac{p(i|\mathbf{x}_t; \mathbf{w})}{\lambda_i}\right)^{\frac{1}{\alpha}}; \\ \exp\left(\frac{-\beta}{\alpha}\right) = \frac{1}{\sum_{k=1}^K \left(\frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k}\right)^{\frac{1}{\alpha}}} \end{array} \right. \\ \Leftrightarrow & \left\{ \begin{array}{l} \hat{y}_t^{(i)\dagger} = \frac{\left(\frac{p(i|\mathbf{x}_t; \mathbf{w})}{\lambda_i}\right)^{\frac{1}{\alpha}}}{\sum_{k=1}^K \left(\frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k}\right)^{\frac{1}{\alpha}}}; \\ \beta = \alpha \log \sum_{k=1}^K \left(\frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k}\right)^{\frac{1}{\alpha}} \end{array} \right. \end{aligned}$$

B. Theoretical properties of CRSTs

B.1. Proof of Proposition 1

Classification maximum likelihood (CML) was initially proposed to model clustering tasks, and can be optimized via classification expectation maximization (CEM). Compared with traditional expectation maximization (EM) that has an “expectation” (E) step and a “maximization” (M) step, CEM has an additional “classification” (C) step (between E and M steps) that assigns a sample to the cluster with maximal posterior probability. In [1], CML is generalized to discriminant semi-supervised learning with both labeled and unlabeled data defined as follows:

$$\log \mathcal{L}_C = \log \tilde{\mathcal{L}}_C + \sum_{i \in S, T} \log p(\mathbf{x}_i)$$

where:

$$\log \tilde{\mathcal{L}}_C = \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) + \sum_{t \in T} \sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w})$$

Note that $\hat{y}_t \in \{0, 1\}^K, \forall t$. $p(k|\mathbf{x}_t; \mathbf{w})$ is the posterior probability modeled by classifiers such as logistic classifier and neural network and \mathbf{w} is the learnable weight. [1] uses a discriminant classifier which makes no assumptions about the data distribution $p(\mathbf{x}_t)$. Thus maximizing (B.1) is equal to maximizing (B.1). Below we draw the connection of the CRST self-training algorithm to CEM. We first show that CRST can be rewritten as the following regularized classification maximum likelihood model:

$$\begin{aligned} \max_{\mathbf{w}, \hat{\mathbf{Y}}_T} \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) + \sum_{t \in T} \sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) \\ - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log \lambda_k + \alpha r_c(\mathbf{w}, \hat{\mathbf{y}}_t) \right] \\ = \log \tilde{\mathcal{L}}_C + \mathcal{R}_C \\ \text{s.t. } \hat{\mathbf{y}}_t \in \Delta^{(K-1)} \cup \{\mathbf{0}\}, \forall t \end{aligned}$$

where the above problem contains an additional regularizer term (\mathcal{R}_C) compared with CML, defined as:

$$\mathcal{R}_C = - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log \lambda_k + \alpha r_c(\mathbf{w}, \hat{\mathbf{y}}_t) \right]$$

In addition, the corresponding alternative self-training optimization can be written as the following CEM process:

E-Step: Given the model weight \mathbf{w} , estimate the posterior probability $p(\mathbf{x}_t; \mathbf{w}), \forall t$.

C-Step: Fix \mathbf{w} and solve the following problem for $\hat{\mathbf{Y}}_T$:

$$\begin{aligned} \max_{\hat{\mathbf{Y}}_T} \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) - \alpha r_c(\mathbf{w}, \hat{\mathbf{y}}_t) \right] \\ \text{s.t. } \hat{\mathbf{y}}_t \in \Delta^{(K-1)} \cup \{\mathbf{0}\}, \forall t \end{aligned}$$

M-Step: Fix $\hat{\mathbf{Y}}_T$ and use gradient ascent to solve the following problem for \mathbf{w} .

$$\begin{aligned} \max_{\mathbf{w}} & \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\ & + \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) - \alpha r_c(\mathbf{w}, \hat{\mathbf{y}}_t) \right] \end{aligned}$$

We have thus shown that the CRST self-training algorithm is an instance of CEM.

B.2. Proof of Proposition 2

As a brief recap, the general form of CRST in (5) can be optimized via the following two steps:

a) Pseudo-label learning Fix \mathbf{w} and solve:

$$\begin{aligned} \min_{\hat{\mathbf{Y}}_T} & \sum_{t \in T} \sum_{k=1}^K -\hat{y}_t^{(k)} \log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} + \alpha r_c(\mathbf{w}, \hat{\mathbf{y}}_t) \\ \text{s.t. } & \hat{\mathbf{y}}_t \in \Delta^{(K-1)} \cup \{\mathbf{0}\}, \forall t \end{aligned} \quad (12)$$

which leads to the following solver for each $\hat{\mathbf{y}}_t$:

$$\hat{\mathbf{y}}_t^* = \begin{cases} \hat{\mathbf{y}}_t^\dagger, & \text{if } \mathcal{C}(\hat{\mathbf{y}}_t^\dagger) < \mathcal{C}(\mathbf{0}) \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (13)$$

where \mathbf{y}_t^\dagger is the minimizer of (12) with the feasible set being $\Delta^{(K-1)}$ only, and $\mathcal{C}(\hat{\mathbf{y}}_t)$ is defined as:

$$\mathcal{C}(\hat{\mathbf{y}}_t) = -\hat{y}_t^{(k)} \sum_{k=1}^K \log \frac{p(k|\mathbf{x}_t; \mathbf{w})}{\lambda_k} + \alpha r_c(\mathbf{w}, \hat{\mathbf{y}}_t)$$

b) Network retraining Fix $\hat{\mathbf{Y}}_T$ and solve the following optimization by gradient descent:

$$\begin{aligned} \min_{\mathbf{w}} & - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log(p(k|\mathbf{x}_s; \mathbf{w})) \\ & - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log(p(k|\mathbf{x}_t; \mathbf{w})) - \alpha r_c(\mathbf{w}, \hat{\mathbf{y}}_t) \right] \end{aligned} \quad (14)$$

We assume $\alpha \geq 0$, and $r_c(\mathbf{w}, \hat{\mathbf{y}}_t)$ is convex w.r.t. \mathbf{w} and $\hat{\mathbf{y}}_t$ given the listed regularizers in Table 1. Note that the definition and optimization of continuous CBST is simply a special case of CRST with $\alpha = 0$. Therefore, the convergence of CRST also indicates the convergence of CBST. With the above preliminaries, we have:

Step a) is non-increasing: (13) is obtained by decomposing (12) into two subproblems with feasible sets being $\Delta^{(K-1)}$ and $\mathbf{0}$, respectively. The former is a convex problems which gives a globally optimal solution, while (13) is the result of comparing this solution against $\mathbf{0}$ by taking the

one with a smaller cost. As a result, (13) is also a global minimizer and (12) is guaranteed to be non-increasing.

Step b) is non-increasing: One may use gradient descent to minimize the loss in (14). With a proper learning rate, the loss is guaranteed to decrease monotonically. In practice, network re-training is often done with mini-batch gradient descent instead of gradient descent. This may not strictly guarantee the monotonic decrease of the loss, but will almost certainly converge to a lower one.

One can prove that the self-training loss in (5) is lower bounded. Therefore, the optimization of (5) by alternatively taking step **a**) and **b**) is convergent.

B.3. Proof of Proposition 4

As mentioned in [57], uniformly smoothed pseudo-label $\hat{\mathbf{y}}_t$ with $\epsilon = (K\alpha - \alpha)/(K + K\alpha)$ is

$$\tilde{y}_t^{(k)} = \begin{cases} 1 - \frac{K\alpha - \alpha}{K + K\alpha}, & \text{if } k = \arg \max_k \{\hat{y}_t\} \\ \frac{\alpha}{K + K\alpha}, & \text{otherwise} \end{cases} \quad (15)$$

And the self-training with uniformly smoothed pseudo-labels is defined as follows.

$$\begin{aligned} \min_{\mathbf{w}} & - \frac{1}{1 + \alpha} \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\ & - \sum_{t \in T} \left[\sum_{k=1}^K \tilde{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) \right] \end{aligned} \quad (16)$$

where $\hat{\mathbf{y}}_t$ follows (15).

In KLD model regularized self-training, the model re-training needs to optimize the following problem:

$$\begin{aligned} \min_{\mathbf{w}} & - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\ & - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) + \frac{\alpha}{K} \log p(k|\mathbf{x}_t; \mathbf{w}) \right] \end{aligned} \quad (17)$$

where $\hat{y}_t, \forall t$ are the fixed pseudo-labels and α is the regularizer weight. Here, we show the equivalence of the above

two problems with the following proof:

$$\begin{aligned}
& \min_{\mathbf{w}} - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\
& - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) + \frac{\alpha}{K} \log p(k|\mathbf{x}_t; \mathbf{w}) \right] \\
\Leftrightarrow & \min_{\mathbf{w}} - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\
& - \sum_{t \in T} \left[\sum_{k=1}^K (\hat{y}_t^{(k)} + \frac{\alpha}{K}) \log p(k|\mathbf{x}_t; \mathbf{w}) \right] \\
\Leftrightarrow & \min_{\mathbf{w}} - \frac{1}{1+\alpha} \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log(p(k|\mathbf{x}_s; \mathbf{w})) \\
& - \sum_{t \in T} \left[\sum_{k=1}^K \frac{(K\hat{y}_t^{(k)} + \alpha)}{K+K\alpha} \log p(k|\mathbf{x}_t; \mathbf{w}) \right]
\end{aligned}$$

Replacing \hat{y}_t with a one-hot completes the proof.

B.4. Proof of Proposition 5

In MRENT, the model retraining needs to optimize the following problem:

$$\begin{aligned}
& \min_{\mathbf{w}} - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\
& - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) \right. \\
& \quad \left. - p(k|\mathbf{x}_t; \mathbf{w}) \log p(k|\mathbf{x}_t; \mathbf{w}) \right]
\end{aligned} \tag{18}$$

We will show the above problem is equivalent to the model retraining in the reverse KLD model regularized self-training, which is defined as follows.

$$\begin{aligned}
& \min_{\mathbf{w}} - \sum_{s \in S} \sum_{k=1}^K y_s^{(k)} \log p(k|\mathbf{x}_s; \mathbf{w}) \\
& - \sum_{t \in T} \left[\sum_{k=1}^K \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t; \mathbf{w}) + D_{KL}(p(\mathbf{x}_t) || \mathbf{u}) \right]
\end{aligned} \tag{19}$$

To prove the above equivalence, we have the following.

$$\begin{aligned}
D_{KL}(p(\mathbf{x}_t) || \mathbf{u}) & = - \sum_{k=1}^K p(k|\mathbf{x}_t) \log \frac{1/K}{p(k|\mathbf{x}_t)} \\
& = \log K + \sum_{k=1}^K p(k|\mathbf{x}_t) \log p(k|\mathbf{x}_t)
\end{aligned} \tag{20}$$

In (20), K is a constant. Thus one can prove that the minimization in (18) is equivalent to the minimization in (19).

C. Additional details on experiments

C.1. Accuracy curves

To show the learning behaviors on VisDA17, we plot the curves of mean accuracy (averaged over 5 runs) versus epochs for CBST and CRSTs in Fig. 6. One can see, the proposed self-training methods are generally stable with only slight fluctuations after 10 epochs. Among all comparing methods, MRKLD+LRENT gives the best performance and shows consistent improvement over the CBST baseline.

C.2. Feature visualization

We also visualize the feature embeddings of the source model, CBST and MRKLD+LRENT features on VisDA17, and show them in Fig. 7. Both CBST and MRKLD+LRENT obtain improved class-wise feature alignment than the source model. MRKLD+LRENT shows slightly more accurate feature alignment due to the improved performance from confidence regularization.

C.3. Confusion matrix

In Fig. 8, we illustrate the normalized confusion matrices of the source model, CBST and MRKLD+LRENT on VisDA17. One can see, both CBST and MRKLD+LRENT show more diagonalized confusion matrices than source model, and MRKLD+LRENT shows less mistakes. Specifically, the confusions between pairwise different classes such as ‘‘person vs. horse’’ and ‘‘motor vs. bike’’ have been reduced by confidence regularization.

C.4. Distributions of softmax probability entries

Following the analysis approach in [43], we present the distributions of predicted softmax probability entries in the target domain for different models. Specifically, we consider the ResNet-38 backbone on GTA5 → Cityscapes, with

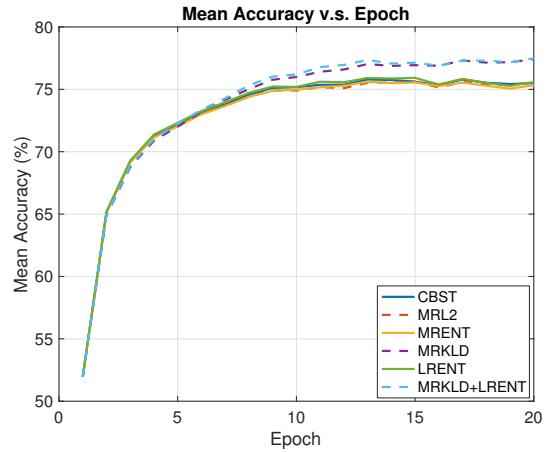


Figure 6: Mean accuracy versus number of epochs.

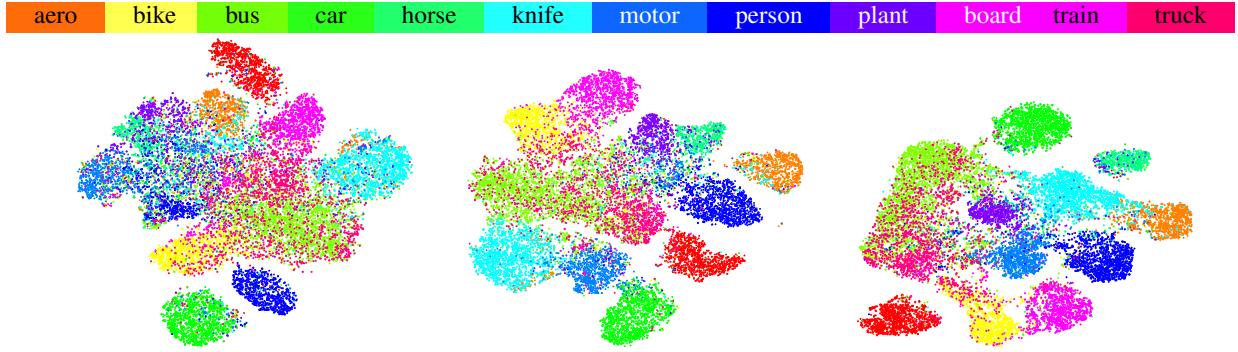


Figure 7: Feature visualization for target domain of VisDA17. From left to right: Source model, CBST, MRKLD+LRENT.

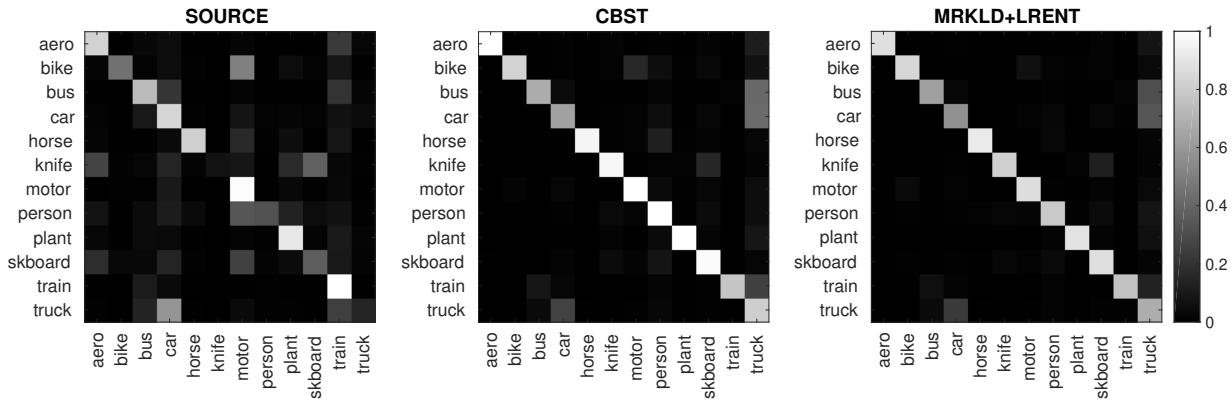


Figure 8: Confusion matrices with normalization for CBST and CRSTs.

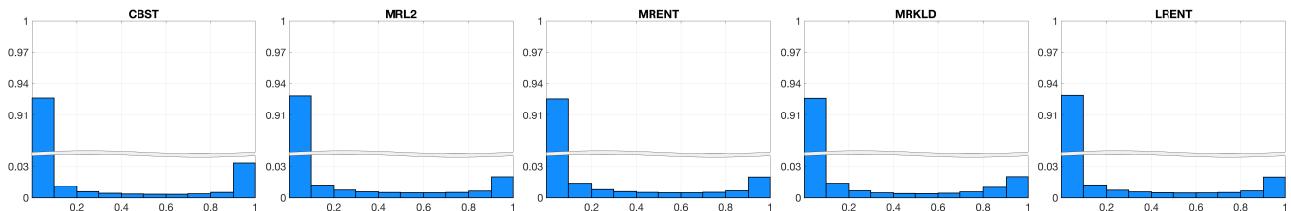


Figure 9: Histograms of softmax probability entries in target domain of GTA5 → Cityscapes.

the distributions shown in Fig. 9. One could see that confidence regularization promote softer distributions by significantly reducing the proportion of highly confident entries.

C.5. Segmentation visualization

For qualitative evaluation, we visualize the segmentation predictions obtained by different models in Fig. 10. Specifically, predictions are made on sampled Cityscapes validation images by GTA5 → Cityscapes models. In Fig. 11, we also visualize the pseudo-labels on sampled Cityscapes training images at the beginning second self-training round.

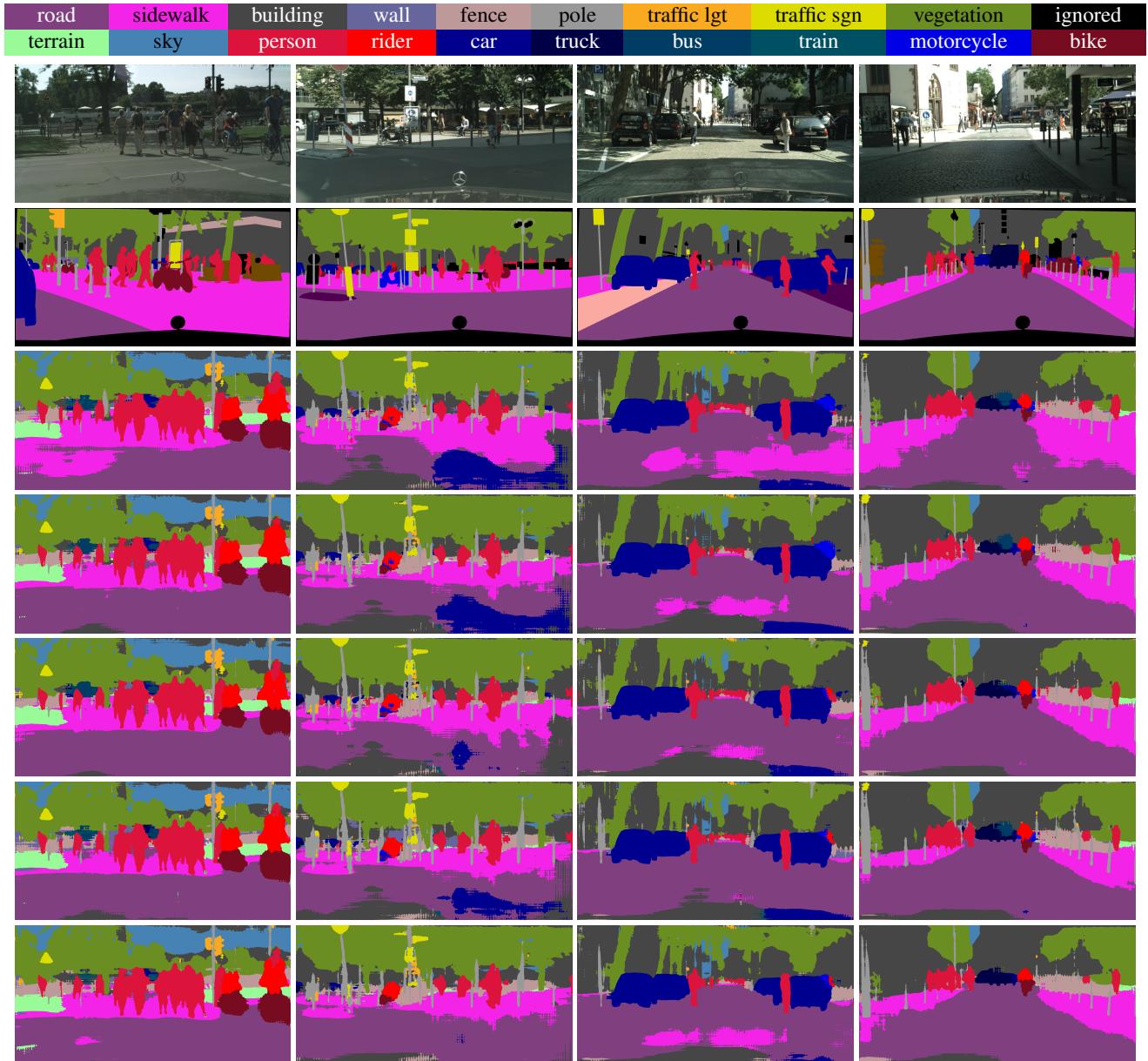


Figure 10: Adaptation results on GTA5 → Cityscapes. Rows correspond to sample images in Cityscapes. From top to bottom, rows correspond to original images, ground truth, and predication results of CBST, MRL2, MRENT, MRKLD, LRENT.

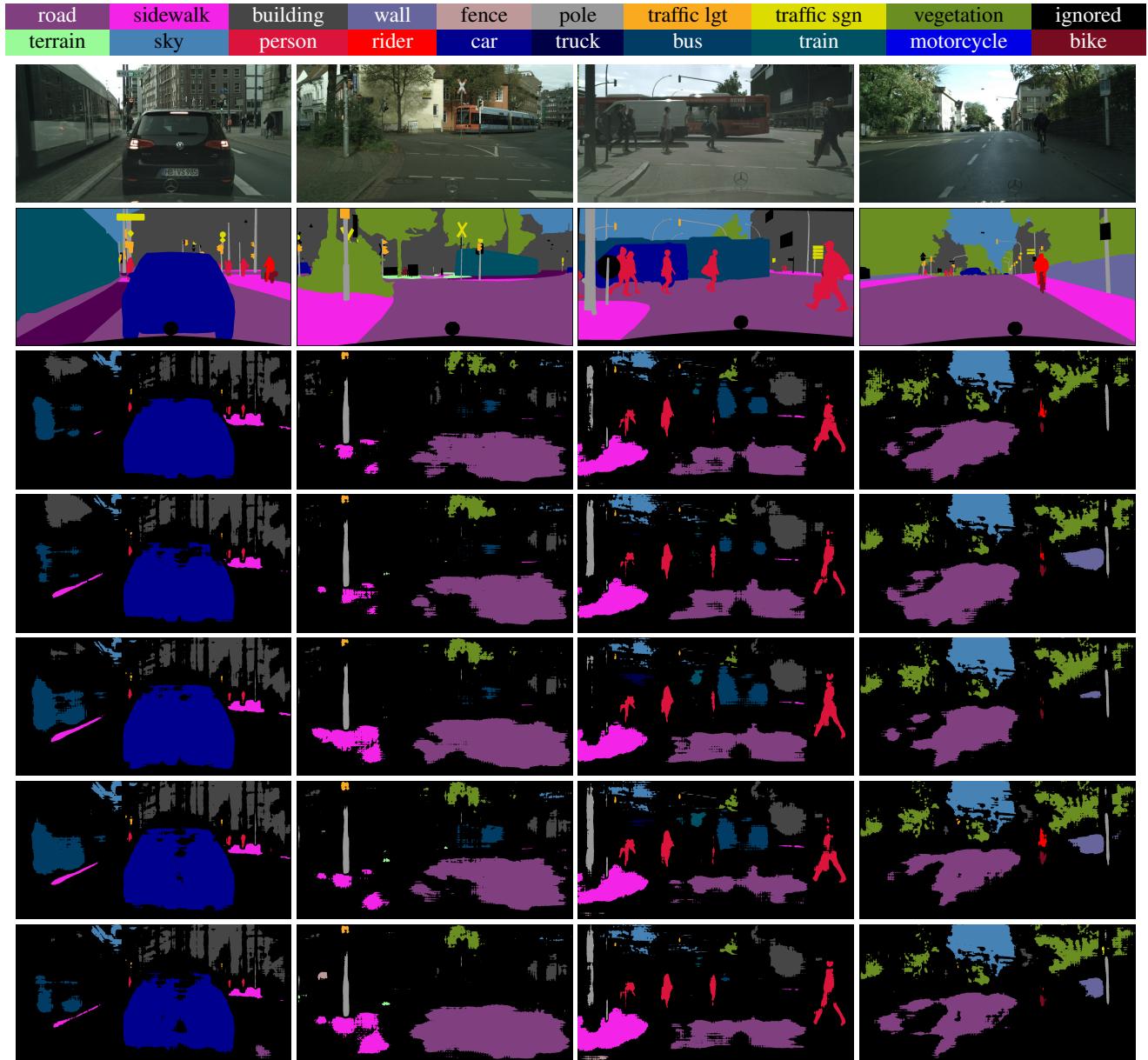


Figure 11: Adaptation results on GTA5 → Cityscapes. Rows correspond to sample images in Cityscapes. From top to bottom, rows correspond to original images, ground truth, and pseudo-label maps of CBST, MRL2, MRENT, MRKLD, LRENT.