Confidence Regularized Self-Training

Yang Zou\textsuperscript{1}\textsuperscript{*} Zhiding Yu\textsuperscript{2}\textsuperscript{*} Xiaofeng Liu\textsuperscript{1} B.V.K. Vijaya Kumar\textsuperscript{1} Jinsong Wang\textsuperscript{3}\textsuperscript{†}
\textsuperscript{1} Carnegie Mellon University \textsuperscript{2} NVIDIA \textsuperscript{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 \url{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 \cite{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 have been proposed to reduce the discrepancy between source and target domain features \cite{9,10,23,26,34,38,44,50,53,60}.

More recently, self-training with networks emerged as a promising alternative towards domain adaptation \cite{4,5,25,29,49,54,69}. 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 retrain network based on these pseudo-labels with target data. Recently, \cite{69} proposes class-balanced self-training (CBST), formulating self-training as 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 \cite{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 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 [69] 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 [22].
- In section 6, we comprehensively evaluate CRST on multiple domain adaptation tasks, including image classification (visDA17/Office-31) and semantic segmentation (GTAS/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, 18]. 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, [69] proposed class-balanced self-training (CBST) 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 [16, 24, 68], feature level [8, 14, 23, 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, 20, 25, 49, 54, 69].

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[13].

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 [17, 57, 32], network output regularization [43], knowledge distillation [22]. 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 (CBST) [69] and reformulate it as a continuous framework. Specifically, for an UDA problem, we have access to the labeled source samples $(x_s, y_s)$ from source domain
\{X_S, Y_S\} and target samples \(x_t\) from unlabeled target domain data \(X_T\). Any target label \(\hat{y}_t = (\hat{y}_t^{(1)}, ..., \hat{y}_t^{(K)})\) from \(Y_T\) is unknown. \(K\) is the total number of classes. We define the network weights as \(w\) and \(p(k|x_t; w)\) 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 generation problem. The pseudo-labels are selected for model retraining. If a sample’s prediction is relatively confident with \(p(k^*|x_t; w) > \lambda_k\), it is selected and labeled as class \(k^* = \arg\max_k \{p(k|x_t; w)\}\). The less confident ones with \(p(k^*|x_t; w) \leq \lambda_k\) are not selected.

\(\lambda_k\) are critical parameters to control pseudo-label learning and selection. The same class-balanced \(\lambda_k\) strategy introduced in [69] is adopted for all self-training methods in this work. \(\lambda_k\) 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 \(k\), \(\lambda_k\) 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 \(\lambda_k\)’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 [69].

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 (CR) to prevent the over-minimization of entropy in self-training. Here, we introduce the general definition of confidence regularized self-training (CRST):

\[
\min_{w, Y_T} \mathcal{L}_{CR}(w, \hat{Y}_T) = \mathcal{L}_{CB}(w, \hat{Y}_T) + \alpha \mathcal{R}_C(w, \hat{Y}_T)
\]

\[
= - \sum_{s \in S} \sum_{k=1}^{K} y_s^{(k)} \log p(k|x_s; w) - \sum_{t \in T} \sum_{k=1}^{K} \hat{y}_t^{(k)} \log p(k|x_t; w) - \alpha r_c(w, \hat{y}_t)
\]

\[s.t. \\hat{y}_t \in \Delta^{K-1} \cup \{0\}, \forall t\]

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

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

$$\min_{Y_T} - \sum_{t \in T} \left[ \sum_{k=1}^{K} \hat{y}^{(k)}_t \log \frac{p(k|x_t; w)}{\lambda_k} - \alpha r_c(\hat{y}_t) \right]$$

subject to $\hat{y}_t \in \Delta^{(K-1)} \cup \{0\}, \forall t$ \hspace{1cm} (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 $\hat{y}_1^t$. The second stage is to select between $\hat{y}_1^t$ or 0 by checking which leads to a lower cost:

$$\hat{y}_t^t = \begin{cases} \hat{y}_1^t, & \text{if } C(\hat{y}_1^t) < C(0) \\ 0, & \text{otherwise} \end{cases}$$

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

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

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 $\hat{y}_1^t$. For clarity, we write $p(k|x_t; w)$ as $p(k|x_t)$ for short. $\hat{y}_1^t$ can be obtained via solving with a Lagrangian multiplier (KKT conditions) [3]. The detailed derivations are shown in Section B of the Supplementary.

4.2. Model regularization

The model regularizer has a general form of $R_C(w) = \sum_{t \in T} r_c(p(x_t; w))$ where $p(x_t; w)$ 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:

$$\min_w - \sum_{s \in S} \sum_{k=1}^{K} \hat{y}^{(k)}_s \log p(k|x_s; w)$$

$$- \sum_{t \in T} \left[ \sum_{k=1}^{K} \hat{y}^{(k)}_t \log p(k|x_t; w) - \alpha r_c(p(x_t; w)) \right]$$

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

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

<table>
<thead>
<tr>
<th>Regularizer</th>
<th>Pseudo-label solution (LR)</th>
<th>Gradient (MR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRENT</td>
<td>$\sum_{k=1}^{K} \hat{y}^{(k)}_s \log (\hat{y}^{(k)}_s)$</td>
<td>$\sum_{k=1}^{K} \alpha r_c(p(x_s; w))$</td>
</tr>
<tr>
<td>MRL2</td>
<td>$\sum_{k=1}^{K} p(k</td>
<td>x_s)^2$</td>
</tr>
<tr>
<td>MRENT</td>
<td>$\sum_{k=1}^{K} \log p(k</td>
<td>x_s)$</td>
</tr>
<tr>
<td>MRKLD</td>
<td>$- \sum_{k=1}^{K} \log p(k</td>
<td>x_s)$</td>
</tr>
</tbody>
</table>

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 A.1 of Supplementary.

**Proposition 2.** Given pre-determined $\lambda_k$, CRST is convergent under certain conditions.

**Proof.** Please refer to Section A.2 of Supplementary.

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 [22] is a common approach in neural network for scaling softmax probabilities with applications in knowledge distillation [22], model calibration [19], 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 $\alpha$ to scale its smoothness as follows:

$$p(i) = \frac{e^{\frac{z_i}{\alpha}}}{\sum_{k=1}^{K} e^{\frac{z_k}{\alpha}}} \hspace{1cm} (9)$$

For high temperature ($\alpha \to \infty$), 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 ($\alpha \to 0$), 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 $\lambda_k$ are equal for all $k$, the soft pseudo-label of LRENT given in Table 1 is exactly the same as softmax with temperature.

**Proof.**

$$\hat{y}_t^{(i)} = \frac{(p(i|x_t)\lambda_k)^{\frac{1}{\alpha}}}{\sum_k (p(k|x_t)\lambda_k)^{\frac{1}{\alpha}}} = \frac{\sum_k p(k|x_t)^\frac{1}{\alpha} = \frac{\sum_k e^{\frac{1}{\alpha}k\lambda_k}}{\sum_k e^{\frac{1}{\alpha}x_k}}}$$

$$= \frac{(e^{\frac{1}{\alpha}i})^{\frac{1}{\alpha}}}{\sum_k (e^{\frac{1}{\alpha}x_k})^{\frac{1}{\alpha}}} = \frac{e^{\frac{1}{\alpha}i}}{\sum_k e^{\frac{1}{\alpha}x_k}}$$

The soft pseudo-label of LRENT can be regarded as a generalized softmax with temperature. In self-training, if selected properly, $\lambda_k$ 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 $\alpha$ is the regularizer weight.

**Proof.** Please refer to Section A.3 of Supplementary.

**Proposition 5.** $D_{KL}(p(x_t)||u) KLD model regularizer (the reverse of the proposed $D_{KL}(u||p(x_t))$ KLD regularizer) is equivalent to entropy model regularizer $-H(p(x_t))$, where $u$ is the uniform distribution.

**Proof.** Please refer to Section A.4 of Supplementary.

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 $A \rightarrow W, D \rightarrow W, W \rightarrow W, A \rightarrow D, A \rightarrow D, W \rightarrow A$.

**Adaptation for semantic segmentation:** We consider two major 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 [23, 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 [21] 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 $1 \times 10^{-3}$, weight decay $5 \times 10^{-4}$, momentum 0.9 and batch size 32. For self-training, we apply the same training strategy but a different learning rate $1 \times 10^{-4}$.

**Semantic segmentation:** For semantic segmentation, we further consider DeepLabv2 [6] as a backbone besides the ResNet-38 backbone in [69]. For experiments with DeepLabv2, we implement CBST/CRSTs using PyTorch, while following the MXNet [7] implementation of [69] for experiments with ResNet-38. DeepLabv2 is pre-trained on ImageNet and fine-tuned on source domain using SGD, with learning rate $2.5 \times 10^{-4}$, weight decay $5 \times 10^{-4}$, momentum 0.9, batch size 2, patch size $512 \times 1024$, multi-scale training augmentation $(0.5 - 1.5)$ and horizontal flipping. In self-training, we apply SGD with learning rate of $5 \times 10^{-5}$. 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 run 5 times and report the average and standard deviation of the evaluation results. Note that both MRKLD and LRENT outperform the non-regularized CBST, whereas MRL2 and MKENT 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. Interestingly, the result even outperforms certain methods with a stronger backbone 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
DeepLabv2 backbone, one could see that MRKLD achieves the best result outperforming previous state-of-the-art. In addition, Fig. 2 visualizes the adapted prediction results obtained by CBST and CRSTs on Cityscapes validation set. Fig. 3 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 [69].

SYNTAHA → 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 [69].

6.4. Parameter analysis

\(p\) is an important parameter controlling the pseudo-label generation and selection sensitivity. We adopt the same \(p\) policy as [69] where we start \(p\) from 20%, and empirically add 5% in each additional self-training round. We conduct a sensitivity analysis for portion \(p\) similar to [69], where we consider the starting portion \(p_0\) and the incremental portion \(\Delta p\) on a difficult task of Office-31: \(W \rightarrow A\). Table 6 shows that CRSTs are not sensitive to \(p_0\) and \(\Delta p\).

In CRST, the coefficient \(\alpha\) is an important parameter that balances the weight between self-training loss and confidence regularizer. In all the experiments, we unify \(\alpha\) to be 0.025, 0.1, 0.1, 0.25 for MRL2, MRENT, MRKLD and LRENT, respectively. Note that various regularizers have different \(\alpha\) due to their intrinsic differences. We also present the sensitivity analysis of \(\alpha\) on \(W \rightarrow A\) in Table 7. We can see all CRSTs are not sensitive to \(\alpha\) in certain intervals.

7. Discussions

7.1. Why 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/wo CR, we compare CBST vs MRKLD/LRENT (DeepLabv2) on GTA5 → Cityscapes, by presenting their per-class mean confidence of TP (\(C_{TP}\)) and 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 \(\alpha\). For MRs, we assume \(y = 1\) and illustrate the regularized loss curves versus \(p\) in Fig. 4. For all MRs, \(p^*\) becomes smoother when \(\alpha\) increases. We notice that MRKLD serves as a better barrier to prevent sharp outputs than other MRs.

Table 2: Experimental results on VisDA17.

<table>
<thead>
<tr>
<th>Method</th>
<th>A/W</th>
<th>T/W</th>
<th>V/H</th>
<th>A/H</th>
<th>A/L</th>
<th>V/H</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRKLD+LRENT</td>
<td>88.0±0.6</td>
<td>79.2±0.2</td>
<td>61.2±0.3</td>
<td>60.0±1.0</td>
<td>87.5±1.2</td>
<td>81.4±5.6</td>
<td>86.3±1.5</td>
</tr>
</tbody>
</table>

Table 3: Results on Office-31.

<table>
<thead>
<tr>
<th>Method</th>
<th>A/W</th>
<th>T/W</th>
<th>V/H</th>
<th>A/H</th>
<th>A/L</th>
<th>V/H</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRKLD+LRENT</td>
<td>88.0±0.6</td>
<td>79.2±0.2</td>
<td>61.2±0.3</td>
<td>60.0±1.0</td>
<td>87.5±1.2</td>
<td>81.4±5.6</td>
<td>86.3±1.5</td>
</tr>
<tr>
<td>Method</td>
<td>Backbone</td>
<td>Road</td>
<td>SW</td>
<td>Build</td>
<td>Wall*</td>
<td>Fence*</td>
<td>Pole*</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>------</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>Source</td>
<td>CBST [69]</td>
<td>DRN-26</td>
<td>42.7</td>
<td>26.3</td>
<td>51.7</td>
<td>5.5</td>
<td>6.8</td>
</tr>
<tr>
<td>Source</td>
<td>CBST [69]</td>
<td>DRN-105</td>
<td>38.6</td>
<td>14.2</td>
<td>67.4</td>
<td>16.4</td>
<td>12.0</td>
</tr>
<tr>
<td>Source</td>
<td>MCD [51]</td>
<td>DRN-105</td>
<td>71.3</td>
<td>19.2</td>
<td>89.2</td>
<td>35.7</td>
<td>10.0</td>
</tr>
<tr>
<td>Source</td>
<td>AdaptSegNet [60]</td>
<td>DeepLabv2</td>
<td>73.8</td>
<td>16.8</td>
<td>77.2</td>
<td>12.5</td>
<td>21.0</td>
</tr>
<tr>
<td>Source</td>
<td>AdvEnt [63]</td>
<td>DeepLabv2</td>
<td>85.4</td>
<td>33.1</td>
<td>81.8</td>
<td>26.6</td>
<td>26.8</td>
</tr>
</tbody>
</table>

Table 4: Experimental results on GTA5 → Cityscapes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>Road</th>
<th>SW</th>
<th>Build</th>
<th>Wall*</th>
<th>Fence*</th>
<th>Pole*</th>
<th>TL</th>
<th>TS</th>
<th>Veg</th>
<th>Sky</th>
<th>PR</th>
<th>Rider</th>
<th>Car</th>
<th>Truck</th>
<th>Bus</th>
<th>Train</th>
<th>Motor</th>
<th>Bike</th>
<th>mIoU*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>CBST [60]</td>
<td>DRN-26</td>
<td>71.3</td>
<td>53.5</td>
<td>80.4</td>
<td>32.7</td>
<td>31.0</td>
<td>34.0</td>
<td>28.9</td>
<td>14.9</td>
<td>28.0</td>
<td>19.6</td>
<td>30.0</td>
<td>19.7</td>
<td>81.5</td>
<td>11.2</td>
<td>36.0</td>
<td>25.7</td>
<td>14.3</td>
<td>43.7</td>
</tr>
<tr>
<td>Source</td>
<td>CBST [60]</td>
<td>DRN-26</td>
<td>71.3</td>
<td>55.5</td>
<td>80.9</td>
<td>32.1</td>
<td>31.5</td>
<td>21.5</td>
<td>36.7</td>
<td>30.0</td>
<td>19.0</td>
<td>84.8</td>
<td>34.9</td>
<td>80.1</td>
<td>56.1</td>
<td>23.8</td>
<td>83.9</td>
<td>28.0</td>
<td>29.4</td>
<td>20.5</td>
</tr>
<tr>
<td>Source</td>
<td>MRL2</td>
<td>DeepLabv2</td>
<td>91.8</td>
<td>53.4</td>
<td>80.6</td>
<td>32.6</td>
<td>20.8</td>
<td>34.3</td>
<td>19.7</td>
<td>21.0</td>
<td>84.0</td>
<td>34.1</td>
<td>80.6</td>
<td>53.9</td>
<td>24.6</td>
<td>82.8</td>
<td>30.8</td>
<td>34.9</td>
<td>16.6</td>
<td>26.4</td>
</tr>
<tr>
<td>Source</td>
<td>MRL2</td>
<td>DeepLabv2</td>
<td>91.0</td>
<td>55.0</td>
<td>80.9</td>
<td>33.7</td>
<td>21.4</td>
<td>37.3</td>
<td>32.9</td>
<td>24.5</td>
<td>85.0</td>
<td>34.1</td>
<td>80.8</td>
<td>57.7</td>
<td>24.6</td>
<td>84.1</td>
<td>27.8</td>
<td>30.1</td>
<td>26.9</td>
<td>26.0</td>
</tr>
<tr>
<td>Source</td>
<td>MRL2</td>
<td>DeepLabv2</td>
<td>81.3</td>
<td>55.8</td>
<td>80.5</td>
<td>32.7</td>
<td>21.0</td>
<td>34.0</td>
<td>29.0</td>
<td>20.3</td>
<td>83.9</td>
<td>34.2</td>
<td>80.9</td>
<td>53.1</td>
<td>24.0</td>
<td>82.7</td>
<td>30.3</td>
<td>35.9</td>
<td>16.0</td>
<td>25.9</td>
</tr>
</tbody>
</table>

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 pseudo-labels of CBST, MRL2, MRENT, MRKLD, LRENT.

Figure 3: Pseudo-labels in GTA5 → Cityscapes. Rows correspond to sample images in Cityscapes. From left to right, columns correspond to original images, ground truth, and pseudo-labels 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 $\alpha$ in Fig. 4. Again, $y^*$ becomes smoother when $\alpha$ 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. Taking
We illustrate two examples of LRENT and MRKLD in Fig. 5, where we assume $p = [0.2, 0.1, 0.55, 0.15]$ for LRENT and $y^{(2)} = 1$ for MRKLD. One can see, LRENT sharpens the input $p$ when $\alpha \in [0, 1]$ (one-hot when $\alpha = 0$), while smooths $p$ 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.

### Table 6: Sensitivity analysis of portion $p_0$ and portion step $\Delta p$.

<table>
<thead>
<tr>
<th>$p_0/\Delta p$</th>
<th>MRL2</th>
<th>MRENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>20/5</td>
<td>15/5</td>
<td>25/5</td>
</tr>
<tr>
<td>Accuracy</td>
<td>72.1±0.2</td>
<td>71.3±0.2</td>
</tr>
<tr>
<td>20/5</td>
<td>15/5</td>
<td>25/5</td>
</tr>
<tr>
<td>Accuracy</td>
<td>71.0±0.4</td>
<td>71.0±0.6</td>
</tr>
</tbody>
</table>

### Table 7: Sensitivity analysis of regularizer weight $\alpha$.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>MRL2</th>
<th>MRENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>71.5±0.8</td>
<td>72.1±0.2</td>
</tr>
<tr>
<td>0.025</td>
<td>71.7±1.1</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>71.0±0.8</td>
<td>71.0±0.4</td>
</tr>
<tr>
<td>0.075</td>
<td>70.9±0.6</td>
<td>70.9±0.4</td>
</tr>
<tr>
<td>0.1</td>
<td>71.0±0.3</td>
<td>71.0±0.8</td>
</tr>
</tbody>
</table>

### Table 8: Comparison of $C_{TP}$, $C_{FP}$ and $C_{TP}/C_{FP}$ on GTA5 $\rightarrow$ Cityscapes.

<table>
<thead>
<tr>
<th>$C_{TP}$ (%)</th>
<th>$C_{FP}$ (%)</th>
<th>$C_{TP}/C_{FP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>96.2</td>
<td>80.1</td>
<td>96.2</td>
</tr>
<tr>
<td>72.2</td>
<td>74.1</td>
<td>69.8</td>
</tr>
<tr>
<td>1.33</td>
<td>1.16</td>
<td>1.36</td>
</tr>
<tr>
<td>94.7</td>
<td>82.8</td>
<td>92.8</td>
</tr>
<tr>
<td>67.7</td>
<td>70.3</td>
<td>65.4</td>
</tr>
<tr>
<td>1.40</td>
<td>1.18</td>
<td>1.41</td>
</tr>
<tr>
<td>70.9</td>
<td>84.4</td>
<td>94.0</td>
</tr>
<tr>
<td>69.5</td>
<td>72.1</td>
<td>68.0</td>
</tr>
<tr>
<td>1.38</td>
<td>1.17</td>
<td>1.38</td>
</tr>
</tbody>
</table>

### Figure 4: Loss curves regularized by different regularizers.

Figure 5: Minimizers of LRENT and MRKLD.

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

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

[20] Ligong Han, Yang Zou, Ruijiang Gao, Lei Wang, and Dimitris Metaxas. Unsupervised domain adaptation via calibration uncertainties. In CVPR Workshops, 2019. 2
[20] Ligong Han, Yang Zou, Ruijiang Gao, Lei Wang, and Dimitris Metaxas. Unsupervised domain adaptation via calibration uncertainties. In CVPR Workshops, 2019. 2
[41] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In CVPR, 2016. 2
[53] Xuzuan Wu, Xintong Han, Yan-Liang Lin, Mustafa Gokhan Uzunbas, Tom Goldstein, Ser Nam Lim, and Larry S Davis. Dcan: Dual channel-wise alignment networks for supervised scene adaptation. In ECCV, 2018. 2