Supplementary: Confidence Regularized Self-Training

Yang Zou[1*] Zhiding Yu[2*] Xiaofeng Liu[1] B.V.K. Vijaya Kumar[1] Jinsong Wang[3†]
[1] Carnegie Mellon University [2] NVIDIA [3] General Motors R&D
✉ yzou2@andrew.cmu.edu, zhidingy@nvidia.com, liuxiaofeng@cmu.edu

A. Theoretical properties of CRSTs

A.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 (A.1) is equal to maximizing (A.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:

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

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

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

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 $\hat{\mathbf{Y}}_T$:

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

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

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

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

A.2. Proof of Proposition 2

As a brief recap, the general form of CRST can be optimized via the following two steps:
a) Pseudo-label learning Fix $\mathbf{w}$ and solve:

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

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

$$\hat{\mathbf{y}}_t^* = \begin{cases} \hat{\mathbf{y}}_t^\dagger, & \text{if } \mathcal{C}(\hat{\mathbf{y}}_t^\dagger) < \mathcal{C}(\mathbf{0}) \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{2}$$

*The authors contributed equally.
†Work done during the affiliation with General Motors R&D.
✉Contact emails of corresponding authors.

where $\mathbf{y}_t^\dagger$ is the minimizer of (1) with the feasible set being Δ^{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:

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

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 of the main paper. 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: (2) is obtained by decomposing (1) into two subproblems with feasible sets being Δ^{K-1} and $\mathbf{0}$, respectively. The former is a convex problems which gives a globally optimal solution, while (2) is the result of comparing this solution against $\mathbf{0}$ by taking the one with a smaller cost. As a result, (2) is also a global minimizer and (1) is guaranteed to be non-increasing.

Step b) is non-increasing: One may use gradient descent to minimize the loss in (3). 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 is lower bounded. Therefore, the optimization by alternatively taking step **a)** and **b)** is convergent.

A.3. Proof of Proposition 4

As mentioned in [6], uniformly smoothed pseudo-label $\hat{\mathbf{y}}_t$ with $\epsilon = (K\alpha - \alpha)/(K + K\alpha)$ is

$$\tilde{y}_t^{(k)} = \begin{cases} 1 - \dfrac{K\alpha - \alpha}{K + K\alpha}, \text{ if } k = \arg\max_k \{\hat{\mathbf{y}}_t\} \\ \dfrac{\alpha}{K + K\alpha}, \text{ otherwise} \end{cases} \tag{4}$$

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

$$\begin{aligned} \min_{\mathbf{w}} & -\frac{1}{1+\alpha}\sum_{s\in S}\sum_{k=1}^{K} y_s^{(k)} \log p(k|\mathbf{x}_s;\mathbf{w}) \\ & -\sum_{t\in T}[\sum_{k=1}^{K} \tilde{y}_t^{(k)} \log p(k|\mathbf{x}_t;\mathbf{w})] \end{aligned} \tag{5}$$

where $\hat{\mathbf{y}}_t$ follows (4).

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

$$\begin{aligned} \min_{\mathbf{w}} & -\sum_{s\in S}\sum_{k=1}^{K} y_s^{(k)} \log p(k|\mathbf{x}_s;\mathbf{w}) \\ & -\sum_{t\in T}[\sum_{k=1}^{K} \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t;\mathbf{w}) + \frac{\alpha}{K} \log p(k|\mathbf{x}_t;\mathbf{w})] \end{aligned} \tag{6}$$

where $\hat{\mathbf{y}}_t, \forall t$ are the fixed pseudo-labels and α is the regularizer weight. We will show the above two problems are equivalent.

To prove the above equivalence, we have the following:

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

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

A.4. Proof of Proposition 5

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

$$\begin{aligned} \min_{\mathbf{w}} & -\sum_{s\in S}\sum_{k=1}^{K} y_s^{(k)} \log p(k|\mathbf{x}_s;\mathbf{w}) \\ & -\sum_{t\in T}\Bigg[\sum_{k=1}^{K} \hat{y}_t^{(k)} \log p(k|\mathbf{x}_t;\mathbf{w}) \\ & - p(k|\mathbf{x}_t;\mathbf{w}) \log p(k|\mathbf{x}_t;\mathbf{w})\Bigg] \end{aligned} \tag{7}$$

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.

$$\begin{aligned}\min_{\mathbf{w}} &-\sum_{s\in S}\sum_{k=1}^{K} y_s^{(k)} \log p(k|\mathbf{x}_s;\mathbf{w}) \\ &-\sum_{t\in T}\left[\sum_{k=1}^{K}\hat{y}_t^{(k)}\log p(k|\mathbf{x}_t;\mathbf{w}) + D_{KL}(p(\mathbf{x}_t)||\mathbf{u})\right]\end{aligned} \tag{8}$$

To prove the above equivalence, we have the following.

$$\begin{aligned}D_{KL}(p(\mathbf{x}_t)||\mathbf{u}) &= -\sum_{k=1}^{K} p(k|\mathbf{x}_t)\log\frac{1/K}{p(k|\mathbf{x}_t)} \\ &= \log K + \sum_{k=1}^{K} p(k|\mathbf{x}_t)\log p(k|\mathbf{x}_t)\end{aligned} \tag{9}$$

In (9), K is a constant. Thus it is easy to prove minimization in (7) is equivalent to minimization (8).

B. Derivation of soft pseudo-label in LRENT

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

$$\begin{aligned}&\min_{\hat{\mathbf{y}}_t}\sum_{k=1}^{K} -\hat{y}_t^{(k)}\log\frac{p(k|\mathbf{x}_t;\mathbf{w})}{\lambda_k} + \alpha\sum_{k=1}^{K}\hat{y}_t^{(k)}\log(\hat{y}_t^{(k)}) \\ &\text{s.t. } \hat{\mathbf{y}}_t \in \Delta^{(K-1)}\end{aligned} \tag{10}$$

where the solution is given as below.

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

It is easy to see that the optimization in (10) is a convex problem. Therefore, the global optimum can be found with a Lagrangian multiplier [2] defined as follows:

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

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

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

C. Additional details on experiments

C.1. Accuracy curves

To see the learning behaviors of CRSTs, for VisDA17, we plot the curves of mean per-class-accuracy v.s. epochs for CBST and CRSTs in Fig. 1. As can be seen in this figure, each CRST is stable with a slightly fluctuation after 10 epochs. Almost all domain adaptation methods have the error propagation problem leading to performance drop in later stage of domain adaptation. Due to the lack of ground truth labels in target domain, it's difficult to validate which model is the optimal model. The stable learning behavior of CRST is a great benefit for model choice.

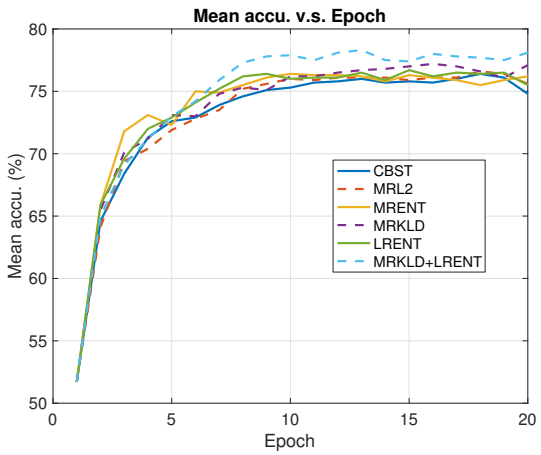

Figure 1: Mean per-class-accuracy v.s. epochs

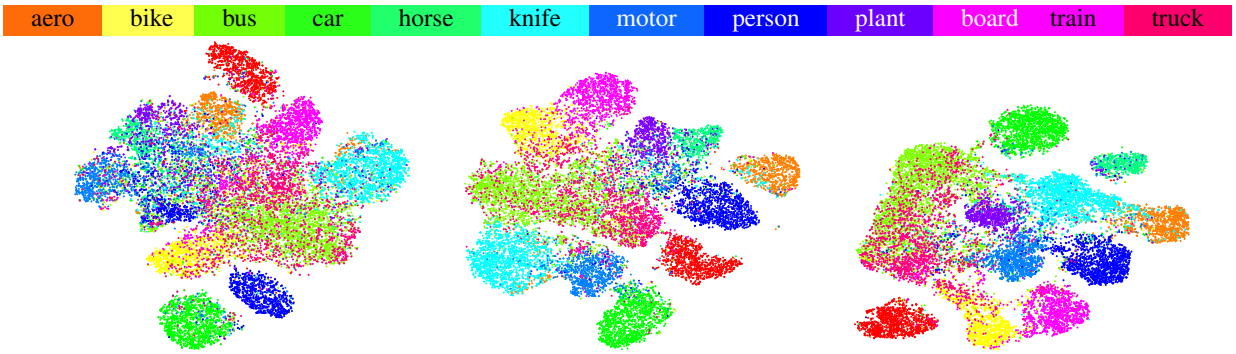

Figure 2: Feature visualization for target domain of VisDA17. From left to right: Source model, CBST, MRKLD+LRENT

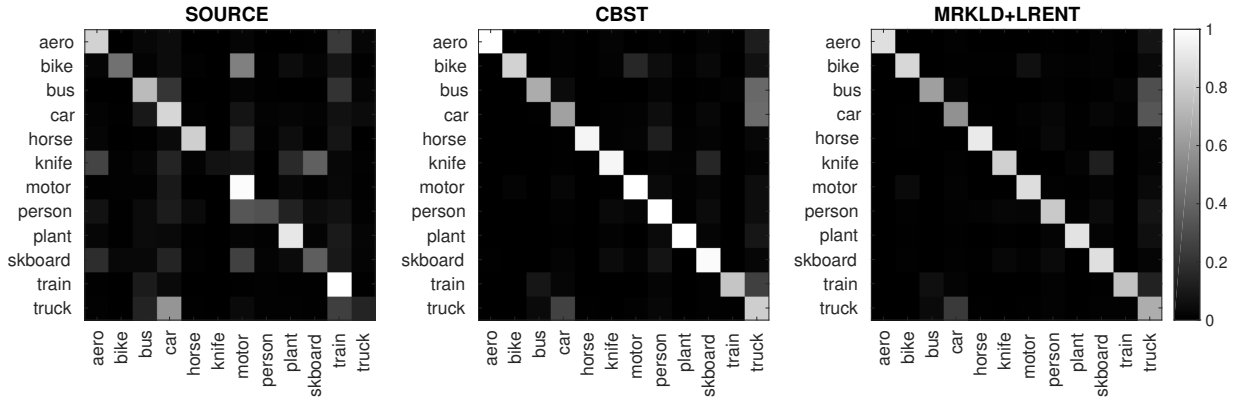

Figure 3: Confusion matrices with normalization for CBST and CRSTs

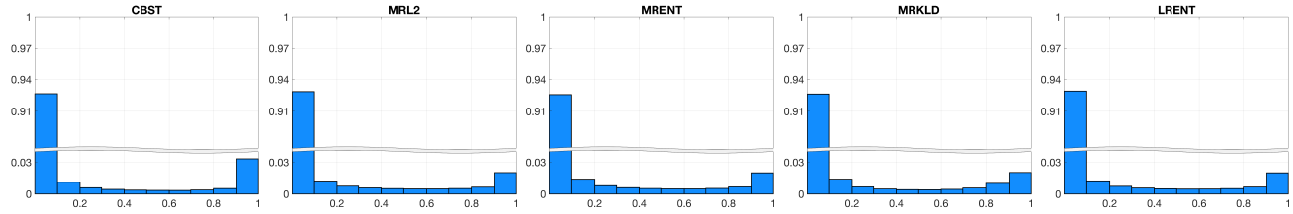

Figure 4: Histograms of softmax probability entries in target domain of GTA5 → Cityscapes

C.2. Feature visualization

In Fig. 2, for VisDA17, we provide the feature visualization by t-SNE [4] for source model, CBST, MRKLD+LRENT. Both CBST and MRKLD+LRENT get better class-wise feature alignment than source model. MRKLD+LRENT learns more accurate features since confidence regularization reduces overconfident mistakes and propagated errors which benefits feature learning.

C.3. Confusion matrix

We give the normalized confusion matrix comparison in Fig. 3 for for VisDA17. As can be seen, both CBST and MRKLD+LRENT have better performances than source model while MRKLD+LRENT is better than CBST. Specifically, the confusion between person and horse, motor and bike, etc., has be reduced by confidence regularization.

C.4. Distributions of softmax probability entries

Following the analysis method from [5], we also present the distributions of softmax probability entries in the target domain of GTA5 → Cityscapes in Fig. 4 with Resnet-38 being the backbone. One could see that confidence regularization promotes softer softmax distributions by significantly reducing the portion number of highly confident entries.

C.5. Segmentation visualization

To intuitively see the improvement of CRSTs compared with CBST, in Fig. 5 we give more prediction samples for GTA5 $\rightarrow$ Cityscapes while in Fig. 6 we give more pseudo-label maps.

References

[1] Massih-Reza Amini and Patrick Gallinari. Semi-supervised logistic regression. In *ECAI*, 2002. 1

[2] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 3

[3] Gilles Celeux and Gérard Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3):315–332, 1992.

[4] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 4

[5] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In *ICLR Workshop*, 2017. 4

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 2

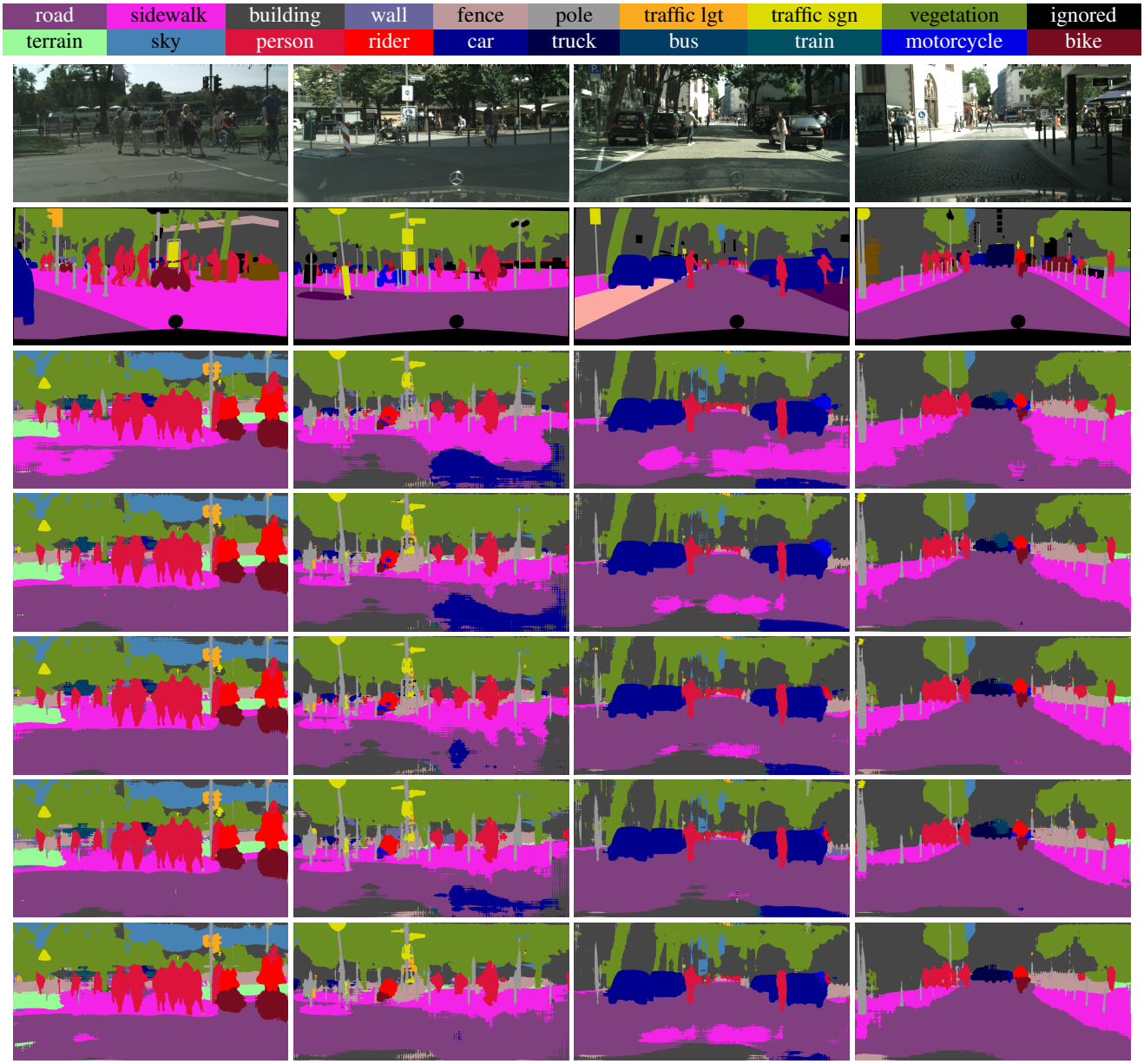

Figure 5: Adaptation results on GTA5 $\rightarrow$ Cityscapes. Rows correspond to sample images in Cityscapes. From top to bottom, rows correspond to original images, ground truth, and predication results of CBST, MRL2, MRENT, MRKLD, LRENT.

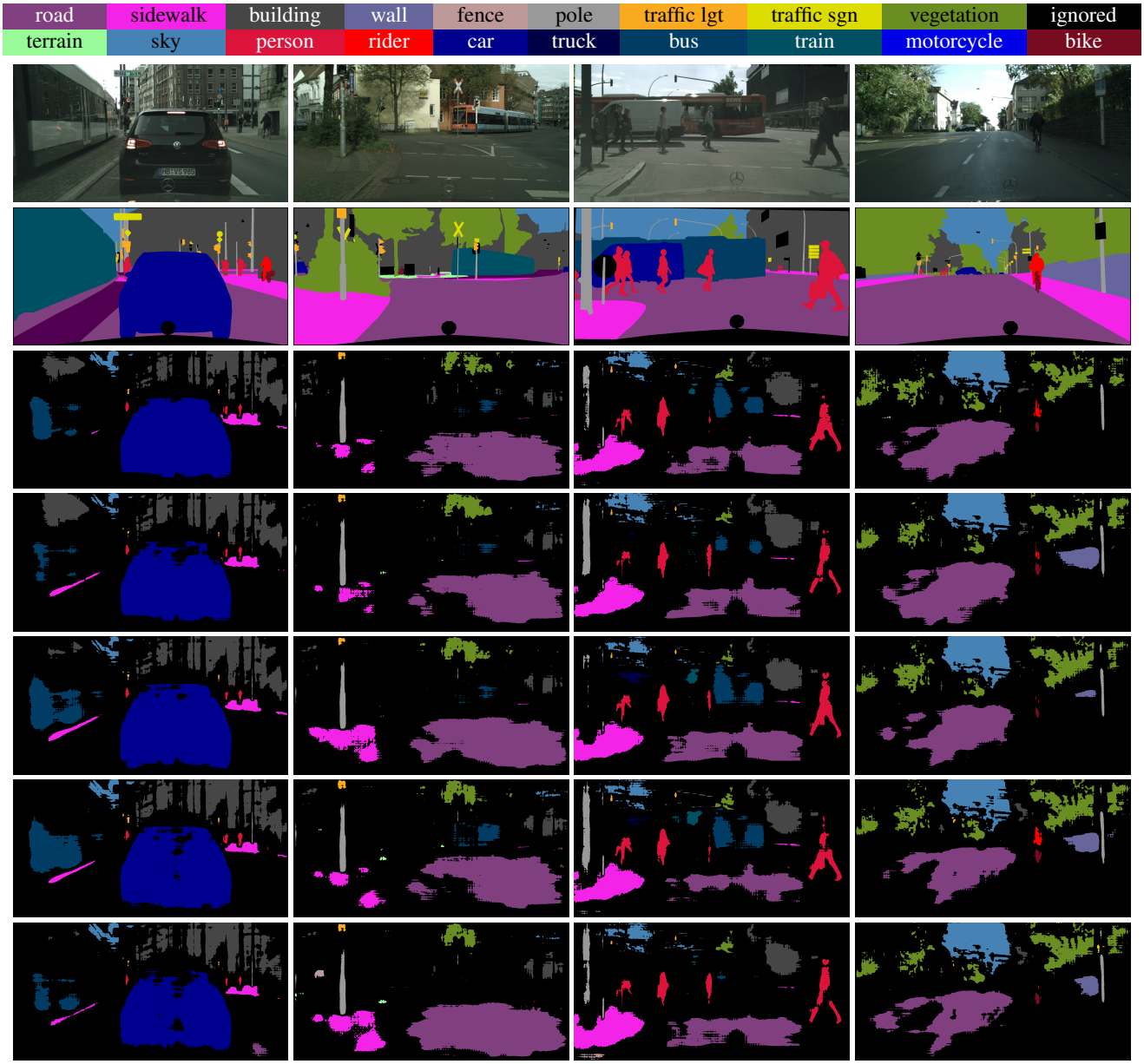

Figure 6: Adaptation results on GTA5 $\rightarrow$ Cityscapes. Rows correspond to sample images in Cityscapes. From top to bottom, rows correspond to original images, ground truth, and pseudo-label maps of CBST, MRL2, MRENT, MRKLD, LRENT.