Unsupervised_Domain_Adaptation_for_Semantic

Paper Title	Unsupervised Domain Adaptation for Semantic Segmentation via Class-Balanced Self-Training
Authors	Yang Zou , Zhiding Yu , B.V.K. Vijaya Kumar , and Jinsong Wang
Conference	ECCV-2018
Association	CMU, NVIDIA, General Motors R&D

Keywords:

- **Domain Gap**: models face challenges in real world “wild tasks” where large difference between labeled training/source data and unseen test/target data exists.
- **Class Balancing** to avoid gradual dominance of large classes on pseudo-label generation.
- **Spatial Priors** for refining generated labels.

Competition on major open benchmark datasets tend to overfit the model to the benchmark data. So the model encounter challenges in practical applications such as autonomous driving, where it needs good performance of the perception module.

Benchmark datasets are usually biased to specific environments, while the testing scenario may encounter large domain differences caused by:

- Change of geological position
- Illumination
- Camera condition
- Weather condition

Domain change problems cannot be remediated by further building up the model power.

Natural idea to improve network’s generalization:

- Annotate data covering more diverse scenes. But they are time consuming and labor intensive.
- Efficiently generate densely annotated images synthetically, from GTA5 and SYNTHIA. But simulated and real domain produce different performance.

This paper focuses on an approach where we train a model on semantic segmentation on a labeled source domain and perform domain adaptation on a target domain where we do not have labels.

Usual approach:

1. **Adversarial Training**: These methods match the distributions of both source and target features. They aim to minimize a domain adversarial loss to reduce both the global and class-wise discrepancy between source and target feature distributions, while retaining good performance on source domain task by minimizing the task-specific loss. But instead of trying to adapt by confusing

the domain discriminator, this paper kind of unifies the feature space alignment and the task itself together under a single, unified loss.

2. **CNN + self-training**: The feature learning in self-training guided by softmax cross-entropy loss not only encourages global closeness of source and target features but also the class-wise feature alignment.

Self-Training assumes that target samples with larger prediction probability have better prediction accuracy.

Different countries may have different construction views and plants, but traffic lights and vehicles are similar. So it’s harder for the source pre-trained models to learn transferable knowledge for construction and plants than for traffic lights and vehicles.

The imbalanced class distribution of source domain, and difference between source distribution and target distribution can also cause different degree of difficulty in transferring knowledge among different classes.

Since ST selects pseudo-labels with large confidence, it tends to be biased towards easy-to-transfer classes ignoring other classes and have inferior adaptation performance.

Traffic scene has its own spatial structure

Fine-Tuning for supervised domain adaptation:

If the labels for the same task in both source and target are available, possibly the most direct way to perform domain adaptation is supervised fine-tuning the model on both domains. For semantic segmentation networks with softmax output, the adaptation problem can be formulated as minimizing the following loss function.
Eq(1):

$$\min_{\mathbf{w}} \mathcal{L}_S(\mathbf{w}) = -\sum_{s=1}^{S}\sum_{n=1}^{N} \mathbf{y}_{s,n}^{\top} \log\left(\mathbf{p}_n\left(\mathbf{w}, \mathbf{I}_s\right)\right) - \sum_{t=1}^{T}\sum_{n=1}^{N} \mathbf{y}_{t,n}^{\top} \log\left(\mathbf{p}_n\left(\mathbf{w}, \mathbf{I}_t\right)\right)$$

where $\mathbf{I}_s$ denotes the image in source domain indexed by $s = 1, 2, \ldots, S, \mathbf{y}_{s,n}$ the ground truth label for the $n-th$ pixel $(n = 1, 2, \ldots, N)$ in $\mathbf{I}_s$, and $\mathbf{W}$ contains the network weights. $\mathbf{p}_n\left(\mathbf{w}, \mathbf{I}_s\right)$ is the softmax output containing the class probabilities at pixel n. Similar definitions apply for $\mathbf{I}_t, \mathbf{y}_{t,n}$ and $\mathbf{p}_n\left(\mathbf{w}, \mathbf{I}_t\right)$.

Self-training for unsupervised domain adaptation:
In the case of unsupervised domain adaptation, the target ground truth labels are not available. An alternate way to fine-tune the segmentation model is to consider the target labels as hidden variables that can be learned. Accordingly the problem can be formulated as follows:
Eq(2):

$$\begin{aligned}&\min_{\mathbf{w}, \mathbf{y}} \mathcal{L}_U(\mathbf{w}, \hat{\mathbf{y}}) = -\sum_{s=1}^{S}\sum_{n=1}^{N} \mathbf{y}_{s,n}^{\top} \log\left(\mathbf{p}_n\left(\mathbf{w}, \mathbf{I}_s\right)\right) - \sum_{t=1}^{T}\sum_{n=1}^{N} \hat{\mathbf{y}}_{t,n}^{\top} \log\left(\mathbf{p}_n\left(\mathbf{w}, \mathbf{I}_t\right)\right)\\ &\text{s.t. } \hat{\mathbf{y}}_{t,n} \in \left\{\mathbf{e}^{(i)} \middle| \mathbf{e}^{(i)} \in \mathbb{R}^C\right\}, \forall t, n\end{aligned}$$

where $\hat{\mathbf{y}}$ indicates the set of target labels, C is the number of classes, and $\mathbf{e}^{(i)}$ a one-hot vector. By minimizing the loss above with respect to $\hat{\mathbf{y}}$, the optimized $\hat{\mathbf{y}}$ should approximate the underlying true target ground truth. Domain adaptation can then be performed similarly to Eq(1). We call $\hat{\mathbf{y}}$ "pseudo-labels" and regard such training strategy as self-training.

Proposed Methods

- **Self-Training(ST) with self-paced learning:**
  - Jointly learning the model and optimizing pseudo-labels on unlabled data is naturally difficult as it is not possible to completely guarantee the correctness of the generated pseudo-labels.
  - A better strategy is **"easy-to-hard" scheme via self-paced curriculum learning**, where one seeks to generate pseudo-labels from the most confident predictions and hope they are mostly correct.
  - **Once the model is updated and better adapted to the target domain, the scheme then explores the remaining pseudo-lables with less confidence.**
  - To incorporate curriculum learning, we consider the following revised self-training formulation: **Eq(3)**

$$\begin{aligned}\min_{\mathbf{w},\hat{\mathbf{y}}} \mathcal{L}_{ST}(\mathbf{w},\hat{\mathbf{y}}) = & -\sum_{s=1}^{S}\sum_{n=1}^{N} \mathbf{y}_{s,n}^{\top} \log\left(\mathbf{p}_n\left(\mathbf{w},\mathbf{I}_s\right)\right) \\ & -\sum_{t=1}^{T}\sum_{n=1}^{N}\left[\hat{\mathbf{y}}_{t,n}^{\top}\log\left(\mathbf{p}_n\left(\mathbf{w},\mathbf{I}_t\right)\right)+k\left|\hat{\mathbf{y}}_{t,n}\right|_1\right] \\ & \text{s.t. } \hat{\mathbf{y}}_{t,n} \in \left\{\left\{\mathbf{e}^{(i)} | \mathbf{e}^{(i)} \in \mathbb{R}^C\right\} \cup \mathbf{0}\right\}, \forall t,n \\ & k > 0\end{aligned}$$

where assigning $\mathbf{y}_{s,n}$ (*maybe this should be* $\mathbf{y}_{t,n}$) as 0 leads to ignoring this pseudo-label in model training, and the L_1 regularization serves as a negative sparse promoting term to prevent the trivial solution of ignoring all pseudo-labels. k is a hyperparameter controlling the amount of ignored pseudo-labels. A larger k encourages the selection of more pseudo-labels for model training. To minimize the loss in Eq(3), we take the following alternate block coordinate descent algorithm:

- a) Fix (initialize) $\mathbf{w}$ and minimize the loss in Eq(3) with respect to $\hat{\mathbf{y}}_{t,n}$.
- b) Fix $\hat{\mathbf{y}}_{t,n}$ and optimize the objective in Eq(3) with respect to $\mathbf{w}$.

Co-ordinate descent:
It is an optimization algorithm that successively minimizes along coordinate directions to find the minimum of a function. At each iteration, the algorithm determines a coordinate or coordinate block via a coordinate selection rule, then exactly or inexactly minimizes over the corresponding coordinate hyperplane while fixing all other coordinates or coordinate blocks.

Coordinate descent is applicable in both differentable and derivative-free contexts.

Differentiable case:

- Choose an initial parameter vector x.
- Until convergence is reached, or for some fixed number of iterations:
    - Choose an index i from 1 to n
    - Choose a step size α.
    - Update x_i to $x_i - \alpha \frac{\partial F}{\partial x_i}(\mathbf{x})$.

We call one step of a) followed by one step of b) as one **round**. We repeat this for multiple rounds.

Intuitively, **step a) selects a certain portion of most confident pseudo-labels from the target domain. step b) trains the network model given the pseudo-labels selcted in step a).**

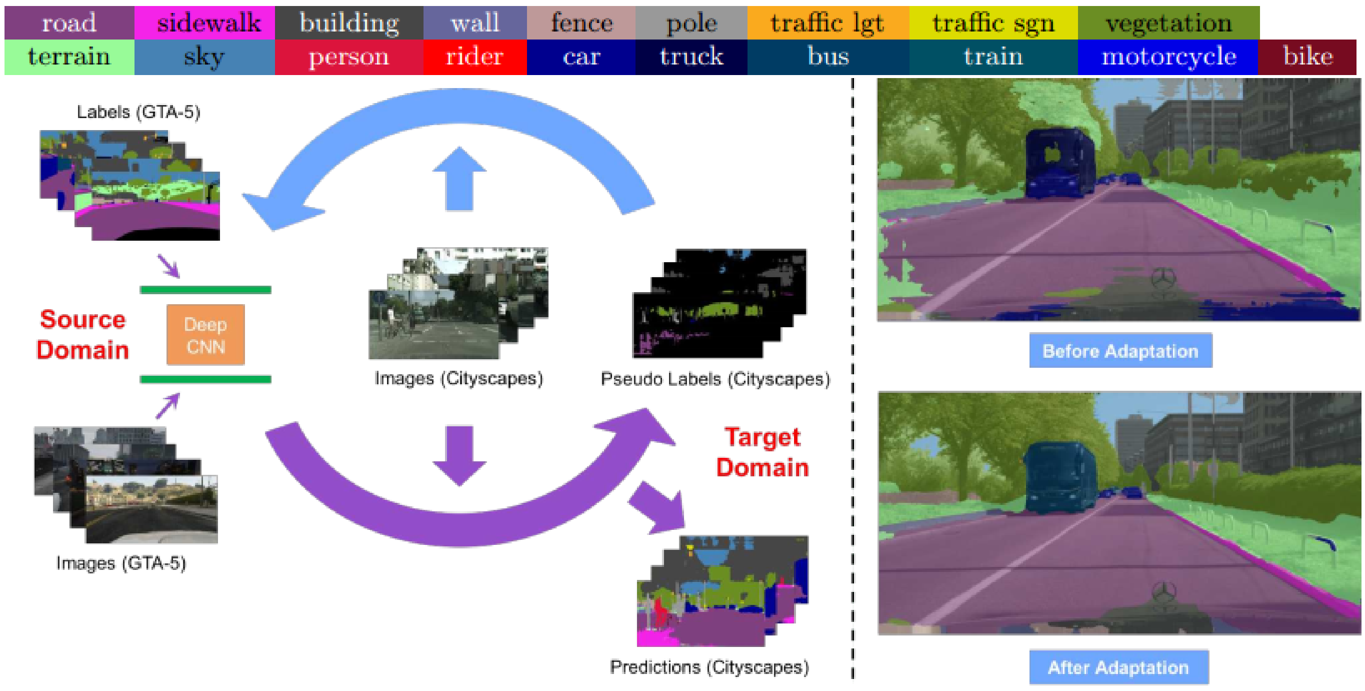

Fig. 1: Illustration of the proposed itertive self-training framework for unsupervised domain adaptation. Left: algorithm workflow. Right figure: semantic segmentation results on Cityscapes before and after adaptation.

Solving *step b) leads to network learning with stochastic gradient descent*. However, solving *step a) requires a nonlinear integer programming given the optimization over discrete variables*. Given $k > 0$, step a) can be rewritten as:

$$\begin{gathered}\min_{\hat{\mathbf{y}}} -\sum_{t=1}^{T}\sum_{n=1}^{N}\left[\sum_{c=1}^{C}\hat{y}_{t,n}^{(c)}\log\left(p_n\left(c|\mathbf{w},\mathbf{I}_t\right)\right)+k\left|\hat{\mathbf{y}}_{t,n}\right|_1\right] \\ \text{s.t. } \hat{\mathbf{y}}_{t,n}=\left[\hat{y}_{t,n}^{(1)},\ldots,\hat{y}_{t,n}^{(C)}\right]\in\left\{\left\{\mathbf{e}^{(i)}|\mathbf{e}^{(i)}\in\mathbb{R}^C\right\}\cup\mathbf{0}\right\},\forall t,n \\ k>0\end{gathered}$$

since $\hat{\mathbf{y}}_{t,n}$ is required to be either a discrete one-hot vector or a zero vector, the pseudo-label configuration can be optimized via the following solver:

$$\hat{y}_{t,n}^{(c)*}=\begin{cases}1, \text{ if } c=\arg\max_c p_n\left(c|\mathbf{w},\mathbf{I}_t\right) \\ p_n\left(c|\mathbf{w},\mathbf{I}_t\right)>\exp(-k) \\ 0, \text{ otherwise}\end{cases}$$

- CNN based self-training can learn not only domain invariant classifier but also domain-invariant features.
- The softmax loss implicitly tries to reduce the domain difference in feature space.

---

1. Class-balanced self-training (CBST)
   Easy-to-transfer classes will result in higher prediction confidence scores among difficult classes due to visual domain gap and skewed class distribution. **Thus it is difficult to perform ST on multi-class segmentation adaptation problem.**
   Solution: Class-balanced self-training framework where class-wise confidence levels are normalized.
   Eqn(4):

$$\begin{aligned}\min_{\mathbf{w},\hat{\mathbf{y}}} \mathcal{L}_{CB}(\mathbf{w},\hat{\mathbf{y}}) = &-\sum_{s=1}^{S}\sum_{n=1}^{N}\mathbf{y}_{s,n}^{\top}\log\left(\mathbf{p}_n\left(\mathbf{w},\mathbf{I}_s\right)\right)\\ &-\sum_{t=1}^{T}\sum_{n=1}^{N}\sum_{c=1}^{C}\left[\hat{y}_{t,n}^{(c)}\log\left(p_n\left(c|\mathbf{w},\mathbf{I}_t\right)\right)+k_c\hat{y}_{t,n}^{(c)}\right]\\ \text{s.t. } \hat{\mathbf{y}}_{t,n} = \left[\hat{y}_{t,n}^{(1)},\ldots,\hat{y}_{t,n}^{(C)}\right] &\in \left\{\left\{\mathbf{e}^{(i)}|\mathbf{e}^{(i)}\in\mathbb{R}^C\right\}\cup\mathbf{0}\right\},\forall t,n\\ k_c > 0, \forall c\end{aligned}$$

where k_c is a separate parameter determining the proportion of selected pseudo-labels in class c. As one may observe, **it is the difference between k_c that introduces different levels of class-wise bias for pseudo-label selection, and addresses the issue of inter-class balance.**
The optimization flow of class-balanced self-training is same as in Eq(3) except for pseudo-label generation. The step of pseudo-label optimization is:

$$\begin{gathered}\min_{\hat{\mathbf{y}}} -\sum_{t=1}^{T}\sum_{n=1}^{N}\sum_{c=1}^{C}\left[\hat{y}_{t,n}^{(c)}\log\left(p_n\left(c|\mathbf{w},\mathbf{I}_t\right)\right)+k_c\hat{y}_{t,n}^{(c)}\right]\\ \text{s.t. } \hat{\mathbf{y}}_{t,n} = \left[\hat{y}_{t,n}^{(1)},\ldots,\hat{y}_{t,n}^{(C)}\right] \in \left\{\left\{\mathbf{e}|\mathbf{e}\in\mathbb{R}^C\right\}\cup\mathbf{0}\right\},\forall t,n\\ k_c > 0, \forall c\end{gathered}$$

For this we need a class-balanced solver:

$$\hat{y}_{t,n}^{(c)*} = \begin{cases} 1, \text{ if } c = \arg\max_c \frac{p_n(c|\mathbf{w},\mathbf{I}_t)}{\exp(-k_c)} \\ \qquad \frac{p_n(c|\mathbf{w},\mathbf{I}_t)}{\exp(-k_c)} > 1 \\ \qquad 0, \text{ otherwise}\end{cases}$$

The pseudo-label generation in Eq(4) is no longer dependent on the output $p_n\left(c|\mathbf{w},\mathbf{I}_t\right)$, but hinges on the normalized output $\frac{p_n(c|\mathbf{w},\mathbf{I}_t)}{\exp(-k_c)}$.

This way of pseudo-label assignment owns the benefit of balancing towards the class with relatively low score but having high within-class confidence.

k_c should be set in a way that exp(-k_c) encodes the response strength of each class to balance different classes.

For CBST, the pseudo-label of any pixel is only filtered when all the balanced responses are smaller than 1. There could also be multiple classes with $\frac{p_n(c|\mathbf{w},\mathbf{I}_t)}{\exp(-k_c)} > 1$. In this case, the class with the maximum balanced response is selected.

Self-paced learning policy design

Determination of k in ST:

- k plays a key role in filtering out pseudo-labels with probabilities smaller than k.
- We take the maximum output probability at each pixel, and sort such probabilities across all pixel locations and all target images in descending order. We then set k such that $\exp(-k)$ equals the probability ranked at round$(p * T * N)$, where p is a proportion number between $[0, 1]$. In this case, pseudo-label optimization produces $p \times 100\%$ most confident pseudo-labels for network training.