HARVARD MEDICAL SCHOOL

MASSACHUSETTS GENERAL HOSPITAL

Gordon Center for Medical Imaging

Energy-constrained Self-training for Unsupervised Domain Adaptation

Xiaofeng Liu, Bo Hu, Xiongchang Liu, Jun Lu, Jane You, Lingsheng

optimizing the likelihood $\log p_{\mathbf{w}}(\mathrm{x},\mathrm{y}) = \boxed{\log p_{\mathbf{w}}(\mathrm{x})} + \boxed{\log p_{\mathbf{w}}(\mathrm{y}|\mathrm{x})}$ can be helpful for both the discrimi nation and generation task.

Cross-entropy loss

The additional optimization objective $\log p_{\mathbf{w}}(\mathrm{x})$ has been proven and evidenced that can improve the confidence calibration and robustness for conventional classification task [15].

Considering the target samples do not have ground truth label, the self-training methods [12] utilize the inaccurate pseudo label to calculate the cross-entropy loss. Therefore, optimizing $\log p_{\mathbf{w}}(\mathrm{x})$ can potentially be more helpful for UDA setting. Actually, $\log p_{\mathbf{w}}(\mathrm{x})$ is adaptive w.r.t. the input x and network parameter w, and irrelevant to the inaccurate pseudo label, which can be an ideal regularizer of self-training based UDA.

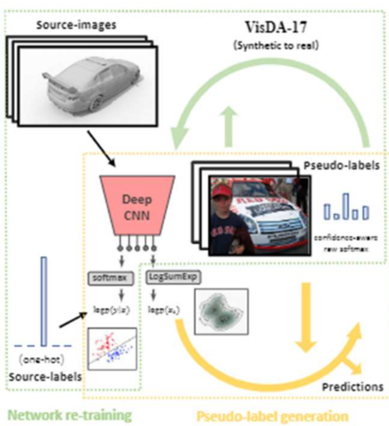

Fig. 1: The illustration of our Energy-constrained Self-training framework for UDA. Minimizing the pseudo label-irrelevant energy of $E_{\mathbf{w}}(\mathbf{x}_t)$ is introduced as additional objective for the target sample.

However, how to modeling $\log p_{\mathbf{w}}(\mathrm{x})$ can be a challenging

Considering $\frac{\partial \log p_{\mathbf{w}}(\mathrm{x})}{\partial \mathbf{w}}$ can be approximated with $-\frac{\partial E_{\mathbf{w}}(\mathrm{x})}{\partial \mathbf{w}}$ [15], it is possible to modeling the energy function $E_{\mathbf{w}}(\mathrm{x})$ instead of $\log p_{\mathbf{w}}(\mathrm{x})$. Following [15], we can define an EBM of the joint distribution $p_{\mathbf{w}}(\mathrm{x},\mathrm{y}) = \exp(f_{\mathbf{w}}(\mathrm{x})[k])/Z(\mathrm{w})$, by defining $E_{\mathbf{w}}(\mathrm{x},\mathrm{y}) = -f_{\mathbf{w}}(\mathrm{x})[k]$. By marginalizing out y, we have $p_{\mathbf{w}}(\mathrm{x}) = \frac{\sum_k \exp(f_{\mathbf{w}}(\mathrm{x})[k])}{Z(\mathrm{w})}$ [15]. Considering $p_{\mathbf{w}}(\mathrm{x}) = \exp(-E_{\mathbf{w}}(\mathrm{x}))/Z(\mathrm{w})$, the energy function of x can be

$$E_{\mathbf{w}}(\mathrm{x}) = -\log \sum_k \exp(f_{\mathbf{w}}(\mathrm{x})[k]) \tag{1}$$

In this setting, $p_{\mathbf{w}}(\mathrm{x}|\mathrm{y}) = \frac{p_{\mathbf{w}}(\mathrm{x},\mathrm{y})}{p_{\mathbf{w}}(\mathrm{x})} = \frac{\exp(f_{\mathbf{w}}(\mathrm{x})[k])/Z(\mathrm{w})}{\sum_k \exp(f_{\mathbf{w}}(\mathrm{x})[k])/Z(\mathrm{w})}$. The normalization constant $Z(\mathrm{w})$ will be canceled out and yielding the standard softmax function, which bridges the EMB and conventional classifiers.

$$E_{\mathbf{w}}(\mathrm{x}) = -\log \sum_k \exp(f_{\mathbf{w}}(\mathrm{x})[k]) \tag{1}$$

In this setting, $p_{\mathbf{w}}(\mathrm{x}|\mathrm{y}) = \frac{p_{\mathbf{w}}(\mathrm{x},\mathrm{y})}{p_{\mathbf{w}}(\mathrm{x})} = \frac{\exp(f_{\mathbf{w}}(\mathrm{x})[k])/Z(\mathrm{w})}{\sum_k \exp(f_{\mathbf{w}}(\mathrm{x})[k])/Z(\mathrm{w})}$. The normalization constant $Z(\mathrm{w})$ will be canceled out and yielding the standard softmax function, which bridges the EMB and conventional classifiers.

Following the formulation in our CRST [12], the self-training with EBM regularization (R-EBM) for target sample, i.e., $E_{\mathbf{w}}(\mathrm{x}_t)$, can be formulated as

$$\min_{\mathbf{w},\hat{\mathbf{Y}}_T} \mathcal{L}_{R-EBM}(\mathbf{w},\hat{\mathbf{Y}}) = -\sum_{s\in S}\sum_{k=1}^{K} y_s^{(k)} \log p_{\mathbf{w}}(k|\mathbf{x}_s)$$
$$-\sum_{t\in T}\{\sum_{k=1}^{K}[\hat{y}_t^{(k)} \log p_{\mathbf{w}}(k|\mathbf{x}_t) - \hat{y}_t^{(k)} \log \lambda_k] - \alpha E_{\mathbf{w}}(\mathbf{x}_t)\}$$
$$s.t.\ \hat{\mathbf{y}}_t \in \Delta^{K-1} \cup \{\mathbf{0}\}, \forall t \tag{2}$$

Step 1) Pseudo-label generation Fix w and solve:

$$\min_{\hat{\mathbf{Y}}_T} -\sum_{t\in T}\{\sum_{k=1}^{K} \hat{y}_t^{(k)}[\log p_{\mathbf{w}}(k|\mathbf{x}_t) - \log \lambda_k] - \alpha E_{\mathbf{w}}(\mathbf{x}_t)\}$$
$$s.t.\ \hat{y}_t \in \Delta^{K-1} \cup \{\mathbf{0}\}, \forall t \tag{3}$$

For solving step **1)**, there is a global optimizer for arbitrary $\hat{\mathbf{y}}_t = (\hat{y}_t^{(1)}, ..., \hat{y}_t^{(K)})$ as [12]:

$$\hat{y}_t^{(k)*} = \begin{cases} 1, \text{ if } k = \underset{k}{\operatorname{argmax}} \dfrac{p_{\mathbf{w}}(k|\mathbf{x}_t)}{\lambda_k} \\ \qquad \text{and} \quad p_{\mathbf{w}}(k|\mathbf{x}_t) > \lambda_k \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

Step 2) Network retraining Fix $\hat{\mathbf{Y}}_T$ and minimize

$$-\sum_{s\in S}\sum_{k=1}^{K} y_s^{(k)} \log p_{\mathbf{w}}(k|\mathbf{x}_s) - \sum_{t\in T}\sum_{k=1}^{K} \hat{y}_t^{(k)} \log p_{\mathbf{w}}(k|\mathbf{x}_t) \tag{5}$$

w.r.t. w. Carrying out step **1)** and **2)** for one time is defined as one round in self-training.

Method	Base Net	Road	SW	Build	Wall	Fence	Pole	TL	TS	Veg.	Terrain	Sky	PR	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU
Source	DRN26	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
CyCADA [47]		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	DRN105	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
MCD [42]		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	PSPNet	69.9	22.3	75.6	15.8	20.1	18.8	28.2	17.1	75.6	8.00	73.5	55.0	2.9	66.9	**34.4**	30.8	0.0	18.4	0.0	33.3
DCAN [48]		85.0	30.8	81.3	25.8	21.2	22.2	25.4	26.6	83.4	36.7	76.2	58.9	24.9	80.7	29.5	42.9	2.50	26.9	11.6	41.7
Source	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
AdaptSegNet [49]		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 [50]	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
Source	DeepLabv2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	29.2
FCAN [51]		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	46.6
Source	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
DPR [52]		92.3	51.9	**82.1**	29.2	25.1	24.5	33.8	**33.0**	82.4	32.8	**82.2**	58.6	27.2	84.3	33.4	**46.3**	2.2	29.5	32.3	46.5
Source	DeepLabv2	73.8	16.0	66.3	12.8	22.3	29.0	30.3	10.2	77.7	19.0	50.8	55.2	20.4	73.6	28.3	25.6	0.1	27.5	12.1	34.2
PyCDA [53]		90.5	36.3	84.4	32.4	**28.7**	34.6	36.4	31.5	86.8	37.9	78.5	62.3	21.5	**85.6**	27.9	34.8	18.0	22.9	**49.3**	47.4
Source	DeepLabv2	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
CBST [31]		89.9	55.0	79.9	29.5	20.6	37.8	32.9	13.9	84.0	31.2	75.5	60.2	27.1	81.8	29.7	40.5	7.62	28.7	41.4	45.6
CBST+R_{EBM}		91.1	53.9	80.6	31.6	21.0	40.4	35.0	19.8	86.8	35.9	76.4	**63.3**	31.4	83.0	22.5	38.6	24.2	32.2	39.4	47.8
Source	DeepLabv2	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
CRST [12]		89.0	51.2	79.4	**31.7**	19.1	38.5	34.1	20.4	84.7	35.4	76.8	61.3	30.2	80.7	27.4	39.4	10.2	32.2	43.3	46.6
CRST+R_{EBM}		**92.5**	**56.6**	80.9	26.2	20.5	**40.5**	**35.3**	24.4	**86.9**	**37.3**	77.5	63.4	**30.5**	81.3	28.8	39.2	**24.6**	**33.5**	41.3	**48.5**

TABLE II: Experimental results for GTA5 to Cityscapes.

https://liu-xiaofeng.github.io/ liuxiaofengcmu@gmail.com