Instance Adaptive Self-Training for Unsupervised Domain Adaptation

Ke Mei, Chuang Zhu, Jiaqi Zou, and Shangshang Zhang

School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China

Abstract

The divergence between labeled training data and unlabeled testing data is a significant challenge for recent deep learning models. Unsupervised domain adaptation (UDA) attempts to solve such a problem. Recent works show that self-training is a powerful approach to UDA. However, existing methods have difficulty in balancing scalability and performance. In this paper, we propose an instance adaptive self-training framework for UDA on the task of semantic segmentation. Our method is so concise and efficient that it is easy to be generalized to other unsupervised domain adaptation methods. Experiments on 'GTA5 to Cityscapes' and 'SYNTHIA to Cityscapes' demonstrate the superior performance of our approach compared with the state-of-the-art methods.

Methods

Instance adaptive self-training framework (IAST) mainly has two contributions:

- Pseudo-Label Generation Strategy with an Instance Adaptive Selectore(IAS): IAS selects an adaptive pseudo-label threshold for each semantic category in units of images and dynamically reduces the proportion of "hard" classes, to eliminate noise in the pseudo-labels.
- Region-guided Regularization: Region-guided regularization is designed to smooth the prediction of the confident region and sharpen the prediction of the ignored region.

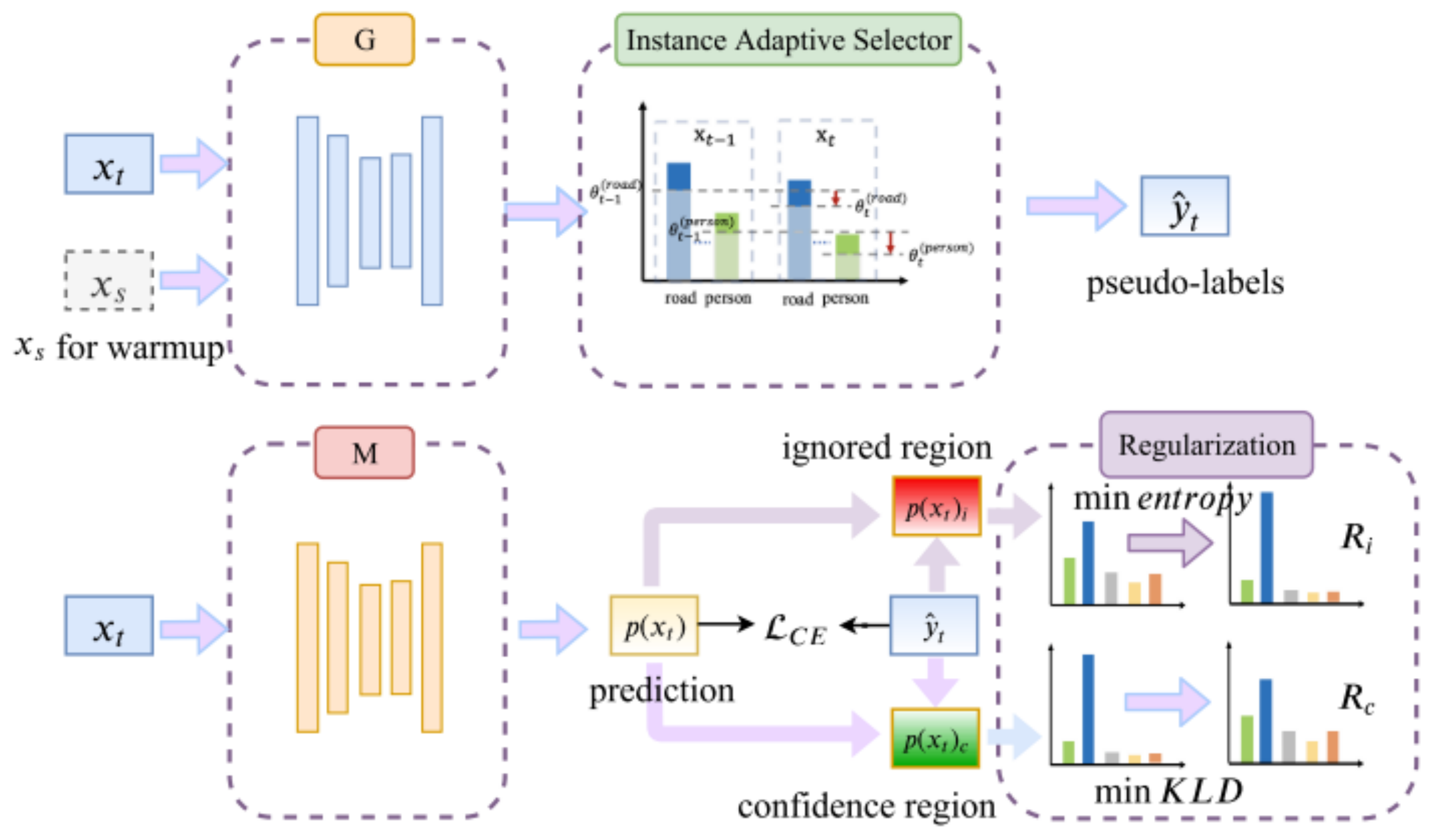

Fig. 1. Proposed IAST framework overview

The IAST training process consists of three phases:

a) In the warm-up phase, a non-self-training method uses both the source data and the target data to train an initial segmentation model M0 as the initial pseudo-label generator G0.
b) In the pseudo-label generation phase, G is used to obtain the prediction result of the target data, and a pseudo-label is generated by an instance adaptive selector.
c) In the self-training phase, the segmentation model M is trained using the target data.

Results

Comparison with the state-of-the-art methods. The results of IAST and some other state-of-the-art methods on GTA5 to Cityscapes are present in Table1. From the overall results, IAST has the best mIoU 52.2% and has obvious advantages over other methods. Compared with some adversarial training methods AdaptSegNet and SIBAN , IAST improves by 9.6% mIoU and have significant gains in almost all classes. Compared with the same self-training methods such as MRKLD, IAST improves by 4.8% mIoU. In addition, BLF is a method that combines adversarial training and self-training, which has the second-best 48.5% mIoU. Compared to BLF, IAST still has a significant improvement.

Table 1. Results of our proposed method IAST and other state-of-the-art methods (GTA5 to Cityscapes).

Method	Arch.	Road	SW	Build	Wall	Fence	Pole	TL	TS	Veg.	Terrain	Sky	PR	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU
Source [27]	AT	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 [27]		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
SIBAN [22]		88.5	35.4	79.5	26.3	24.3	28.5	32.5	18.3	81.2	40.0	76.5	58.1	25.8	82.6	30.3	34.3	3.4	21.6	21.5	42.6
SSF-DAN [6]		90.3	38.9	81.7	24.8	22.9	30.5	37.0	21.2	84.8	38.8	76.9	58.8	30.7	85.7	30.6	38.1	5.9	28.3	36.9	45.4
AdvEnt [29]		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	45.4
APODA [30]		85.6	32.8	79.0	29.5	25.5	26.8	34.6	19.9	83.7	40.6	77.9	59.2	28.3	84.6	34.6	49.2	8.0	32.6	39.6	45.9
Source [36]	ST	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 [36]		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
PyCDA[20]		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
MRKLD [35]		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
BLF [19]	A&S	91.0	44.7	84.2	34.6	27.6	30.2	36.0	36.0	85.0	**43.6**	83.0	58.6	31.6	83.3	35.3	49.7	3.3	28.8	35.6	48.5
AdaptMR [34]		90.5	35.0	84.6	34.3	24.0	36.8	**44.1**	**42.7**	84.5	33.6	82.5	**63.1**	**34.4**	85.8	32.9	38.2	2.0	27.1	41.8	48.3
PatchAlign [28]		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	26.3	2.2	29.5	32.3	46.5
Source(ours)	A&S	64.8	21.7	74.3	15.4	21.2	18.2	30.7	13.0	80.9	33.7	76.3	55.6	20.0	43.9	27.0	35.5	4.4	24.9	14.3	35.6
IAST(ours)		93.8	57.8	85.1	39.5	26.7	26.2	43.1	34.7	84.9	32.9	88.0	62.6	29.0	87.3	39.2	49.6	23.2	34.7	39.6	51.5
IAST-MST(ours)		**94.1**	**58.8**	**85.4**	**39.7**	**29.2**	25.1	43.1	34.2	84.8	34.6	**88.7**	62.7	30.3	**87.6**	**42.3**	**50.3**	24.7	**35.2**	40.2	**52.2**

Apply to other UDA methods. Because IAST has no special structure or model dependencies, it can be directly used to decorate other UDA methods. We chose two typical adversarial training methods, AdapSeg and AdvEnt for experiments. As shown in Table 2, these two methods have significantly improved performance under the IAST framework.

Table 2. Semi-supervised learning results on the Cityscapes val set. 1/8, 1/4 and 1/2 mean the proportion of labeled images

Method	Data Amount			
	1/8	1/4	1/2	Full
Baseline	57.3	59.0	61.2	70.2
Univ-full[12]	55.9	-	-	-
AdvSemi[11]	58.8	62.3	65.7	67.7
IAST(ours)	**64.6**	**66.7**	**69.8**	**70.2**

Extension: other tasks. The self-training method can also be applied to semi-supervised semantic segmentation task. We use the same configuration in Cityscapes for semi-supervised training with different proportions of data as labeled data. As shown in Table 3, we have significantly better performance than others.

Table 3.Extension analysis, applying IAST to non-self-learning UDA methods(test on Cityscapes), and Source means training IAST without warmup.

Method	GTA5			SYNTHIA		
	Base	+IAST	Δ	Base	+IAST	Δ
AdaptSeg[27]	42.4	50.2	+7.8	46.7	54.7	+8.0
AdvEnt[29]	45.4	49.8	+4.4	48.0	55.1	+7.1
Source	35.6	48.8	+13.2	42.2	54.2	+12.0

Software and hardware environment: In our experiments, we implement IAST using PyTorch1.0 on an NVIDIA Tesla V100.