Uncertainty-aware self-training with expectation maximization basis transformation

Zijia Wang
Dell Technologies
Shanghai, China
Zijia_Wang@Dell.com

Wenbin Yang
Dell Technologies
Shanghai, China
ralph.yang@dell.com

Zhisong Liu
Dell Technologies
Shanghai, China
zhisong.liu@dell.com

Zhen Jia
Dell Technologies
Shanghai, China
z_jia@dell.com

Abstract

Self-training is a powerful approach to deep learning. The key process is to find a pseudo-label for modeling. However, previous self-training algorithms suffer from the over-confidence issue brought by the hard labels, even some confidence-related regularizers cannot comprehensively catch the uncertainty. Therefore, we propose a new self-training framework to combine uncertainty information of both model and dataset. Specifically, we propose to use Expectation-Maximization (EM) to smooth the labels and comprehensively estimate the uncertainty information. We further design a basis extraction network to estimate the initial basis from the dataset. The obtained basis with uncertainty can be filtered based on uncertainty information. It can then be transformed into the real hard label to iteratively update the model and basis in the retraining process. Experiments on image classification and semantic segmentation show the advantages of our methods among confidence-aware self-training algorithms with 1-3 percentage improvement on different datasets.

1 Introduction

Deep neural networks have been developed for many years and achieved great outcomes. However, its superiority relies on large-scale data labeling. In some real situations, like agriculture, it is difficult to obtain labeled data. To alleviate the burden of data labeling, many methods like domain adaption (1; 2; 3; 4; 5), and self-training (6; 7; 8; 9; 10; 11) have been proposed. For example, BERT (12) and GPT (13; 14; 15), directly leverage a large amount of unlabeled data to pretrain the model. However, they cannot be generally applied in other areas. Among these methods, self training methods(16; 17) show promising results and it attracts much attention.

Self training is a semi-supervised learning method (18), which iteratively generates task specific pseudo-labels using a model trained on some labeled data. It then retrains the model using the labeled data. However, there are many issues in this bootstrap process, one of them is the noise in the pseudo-labeled data. Some researchers resolve this problem by learning from noisy labels (19; 20; 21; 22). It can also be optimized by sample selection (23) or label smoothing (24). However, none of the previous works focused on data properties. Recently, a novel knowledge distillation (25) is proposed to distill the large dataset into a small one (26; 27).The intuition of these methods is to find the key samples, like means in the feature spaces, to capture the data properties. These means

Preprint. Under review.

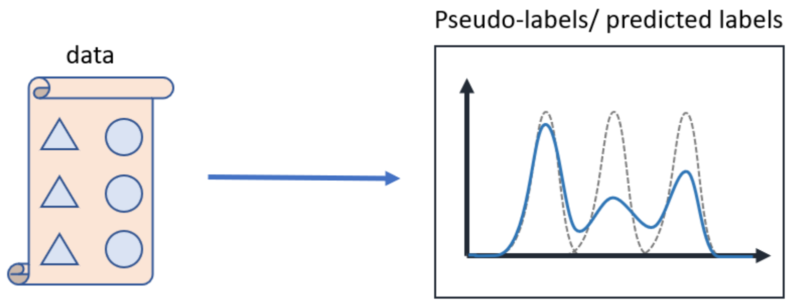

Figure 1: Uncertainty-aware representations. In the right part of this figure, dashed curves represent the basis distributions while the blue curve represent the uncertainty-aware representation and uncertainty-aware labels of the data. The expectation of the labels could be used as the final label and the variance could be used to evaluate the uncertainty.

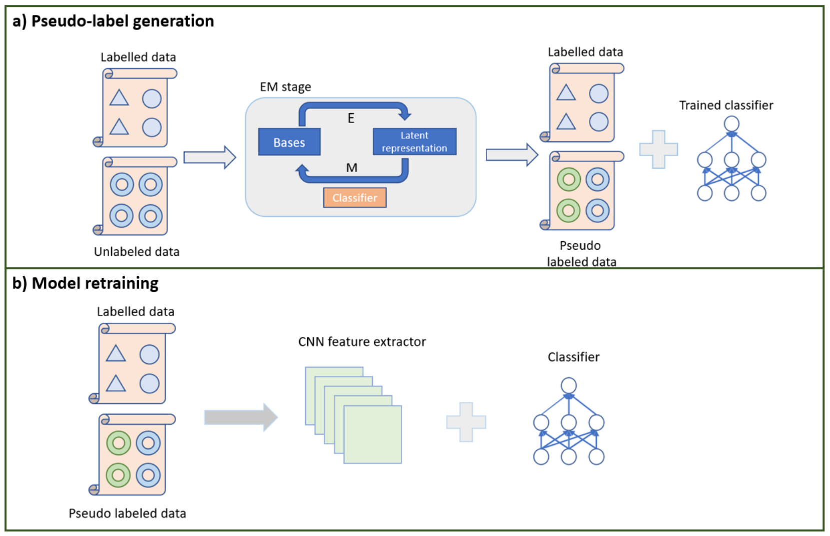

Figure 2: One self training round. Pseudo-label generation (a) use EM algorithm to update the Gaussian basis and the classifier, then it generates some pseudo-labels with uncertainty information while the classifier is also trained in this stage. Then in model retraining stage (b), an uncertainty-aware training strategy is used to update the whole model (CNN and classifier).

could also be referred as basis of the data. They can be used to formulate the latent representations of the data in a probabilistic way using expectation maximization algorithm (28; 29).

Therefore, as shown in figure 1, we propose a probabilistic model to extract uncertainty for self-training. Concretely, expectation maximization algorithm is adapted to get the probabilistic latent representations of the data and their corresponding pseudo-label distributions can be obtained. Then the samples are selected based on the variance of the (pseudo-)label distribution where distributions with lower variance represent good (pseudo-)labels. Finally, an uncertainty-aware training process is used to retrain the model using the new dataset where the expectation of distributions becomes the final pseudo-labels. Overall, our contributions in this paper are:

- Adapt Expectation Maximization algorithm to perform basis transformation on data features. We use neural networks for expectation maximization process to generate the latent probabilistic representations of the data using base transformation. These representations are low-rank while keeping the uncertainty information and deprecating the noises.

- A novel regularizer is used for pseudo-label generation. Variance and classification loss are combined in the pseudo-label generation process to get the best pseudo-label distributions which contain comprehensive uncertainty information.
- A basis generation process with basis regularizer is proposed. An attention-like module (ATT block) is introduced here to extract basis from the dataset or feature space. To make the basis more robust, we propose a basis regularizer to make all basis orthogonal, which could lower the rank of final latent representations.

2 Related work

Self-training: Self-training is a wide and meaningful research area in semi-supervised learning (30; 31; 32), one basic direction in this area is to train a student net using a teacher net (33; 34; 35), some other works use a pseudo-label-based method for self-training (11). In this paper, we choose to use pseudo-label-based method while keeping the uncertainty information in the label, an iterative training framework is proposed according to the self-training paradigm and uncertainty information to improve the network performance.

Expectation-Maximization and Gaussian Mixture Model: Expectation-maximization (EM) (36) is to find solutions for latent variables models using likelihood maximization algorithm while Gaussian mixture model (GMM) (37) is also one kind of EM algorithm with specific constraints. Latent variables models with GMM could naturally capture the uncertainty information considering the data properties. In GMM, the data could be represented in the distribution form:

$$p(\hat{x_n}) = \sum_{k=1}^{K} z_{nk} \mathcal{N}(x_n|\mu_k, \Sigma_k), \tag{1}$$

where the latent representation $\hat{x_n}$ is viewed as a linear superposition of k Gaussian basis $\mathcal{N}(x_n|\mu_k, \Sigma_k)$ and K is the basis number, z_{nk} represents the weight of this linear composition. In the GMM, z_{nk} could be updated in the E step:

$$z_{nk}^{new} = \frac{\mathcal{N}(\mu_k^{new}, \Sigma_k)}{\sum_{j=1}^{K} \mathcal{N}(\mu_j^{new}, \Sigma_j)}, \tag{2}$$

Notably, the Σ_k in the Gaussian basis is set to be identity matrix $\mathbf{I}$ in this paper, so the Σ update process is ignored in our algorithm.

3 Problem definition

In this part, we formally define the uncertainty-aware self-training problem. Given a set of labeled samples $\{\mathbf{X}_L, \mathbf{Y}_L\}$ and a set of unlabeled data $\mathbf{X}_U$ where $\mathbf{X}_U$ and $\mathbf{X}_L$ belong to same domain. Then the goal is to find a latent representation $\hat{\mathbf{X}}$ and uncertainty-aware pseudo-labels $\mathbf{Y}_U$ by using a CNN feature extractor and a simple classifier.

As shown in Figure 2, our problem could be solved by alternating the following steps (24):

a) **Pseudo-label generation:** Given all the data, EM algorithm is used to generate the pseudo-labels with uncertainty information while the classifier is also trained in this process based on a combined loss to reduce the variance of pseudo-labels and optimize the classification accuracy for labeled data.

b) **Network retraining**. Data are sampled from the pseudo-labeled data based on the label variance, then the sampled data, along with the original labeled data, are used to train the whole classification network.

4 Uncertainty-aware self training

To generate the pseudo-label for unlabeled data $\mathbf{X}_U$, we first use a base extraction net trained on labeled data to get basis for $\mathbf{X}_L$, then these bases could be used as the initialized $\mu(0)$ of EM stage to

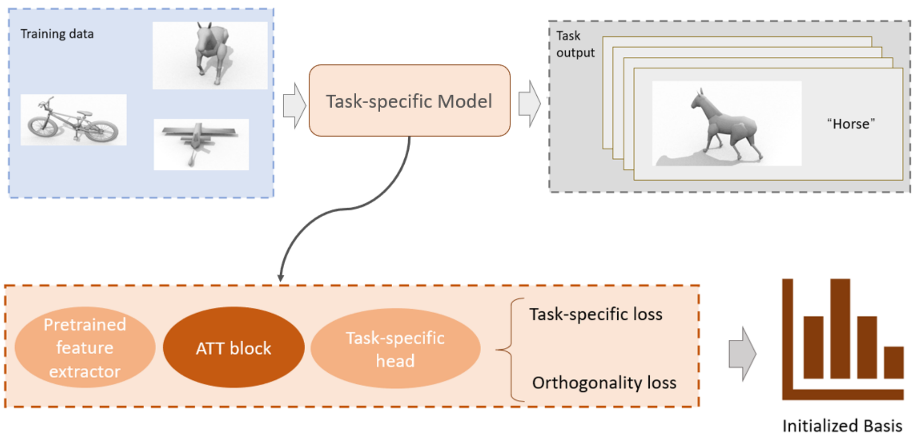

Figure 3: **Whole training process for basis initialization net.** Concretely, we train the model like classical machine learning training process and add a small module (attention block) to extract the processed weights which then become the initialized basis of EM algorithm.

speed up the convergence. Notably, as mentioned in related work section, the Σ is set to be identity matrix and not updated in our algorithm considering a good basis should have identical variance. After the initialization, the EM algorithm is adapted to update the μ while the prediction net is simultaneously updated in the EM stage.

Concretely, the details of base extraction net is shown in section 4.1, then two losses which are used in the EM stage to update the pseudo label generator parameters (classifier in figure 2 a) are demonstrated in section 4.2. After the definition of losses, the whole EM stage is described in section 4.2.1.

4.1 Basis Extraction net

As shown in figure 3, we demonstrate the generalized basis initialization net. In this paper, we use classification as an example where the model trained in this stage has 3 components:

- **Feature extractor.** In fig 3, CNN functions as the feature extractor. The weights we extracted are from this part.
- **Classifier.** The fully connected layer could be the classifier in our setting, this part is for the original machine learning tasks like classification.
- **Weight extractor.** An additional ATT block is added to extract the informative basis from the feature space.

Clearly in training process, there are 2 tasks: classification and weights extraction. For classification, we use classical classification loss - negative log likelihood loss (L_{nll}). Then for weight extraction part, we want our weights to be basis with low rank, so they need to be orthogonal:

$$L_2 = W * W^T - I \tag{3}$$

Where W is the weight and I is the unity matrix. Therefore, the loss becomes:

$$L_{s1} = L_{nll} + L_2 \tag{4}$$

In Attention block (ATT block), given a matrix $X \in R^{N \times d}$ which contains the features of all data samples, we try to extract the inherent low-rank properties of features by basis extraction. The basis extraction, says the problem to find the most informative projection of features, can be formally expressed as

$$min_{\mu} \left\| X - \mu Z \right\|_F s.t. \mu^T \mu = IZ = \mu^T X \tag{5}$$

where $\mu \in R^{K \times d}$ represents the basis matrix of the latent features. Through the process, the inherent data structure can be founded. However, as an unsupervised method, the problem is reported easily suffer from the model collapse problems. Considering the important label information in classification problems. then we can modify the problem above into a semi-supervised manner as

$$min_{\mu}\left\|X-\mu Z\right\|_F+\left\|ZZ^T-YY^T\right\|_F+\left\|\mu^T\mu-I\right\|_F s.t Z=\mu^T X \tag{6}$$

where Y donates all the labels. We can solve the problems above with standard gradient decent methods. Then, after stage I, we generated some basis which the latent space features of data samples effectively and precisely.

4.2 Pseudo-label generation

Recall that the latent representation should be transformed into the pseudo label using a function f_θ. Given a latent representation $\hat{x_n}$ will obey the fallowing distribution:

$$p(\hat{x_n})=\sum_{k=1}^{K} z_{nk}\mathcal{N}(x_n|\mu_k,\Sigma_k), \tag{7}$$

where K is the number of basis, $\mathcal{G}(\mu,\Sigma)$ is the final distribution basis representation. Then the corresponding pseudo label for sample $\hat{x_n}(m)$ is $\hat{y_n}(m)=f_\theta(\hat{x_n}(m))$. With the will know re-parameter trick, distribution $p(y_n)$ can be formally expressed as

$$p(y_n)=p(y_n|x_n)p(x_n|\epsilon)dx_n d\epsilon, \epsilon\sim\mathcal{N}(0,I) \tag{8}$$

where

$$p(x_n|\epsilon)=\sum_{k=1}^{K} z_{nk}\mu_k+\Sigma_k\epsilon \tag{9}$$

Then, we could easily compute the variance $VAR(\hat{y_n})$ and expectation $E(\hat{y_n})$ using these sampled pseudo label. For latent representations in X_L which have label y_n, the loss function for f_θ is:

$$Loss_L=E(\hat{y_n})-y_n \tag{10}$$

For latent representations in X_U which don't have label, the loss is basically the variance, therefore the final loss for pseudo label prediction model is:

$$L=\lambda Loss_L+(1-\lambda)VAR(\hat{y_n}), \tag{11}$$

where $\lambda=1$ if the latent representation is from X_U and vice versa.

4.2.1 Expectation-Maximization

Now we can get the ideally orthogonal base vectors from weights and use them as initialized μ in the base generation block and compute the loss. Then in this section, we formally define the adapted EM process. At first, we need to update z_{nk}:

$$z_{nk}^{new}=\frac{\mathcal{K}(x_n,\mu_k)}{\sum_{j=1}^{K}\mathcal{K}(x_n,\mu_j)}, \tag{12}$$

where $\mathcal{K}(a,b)$ is a kernel function to evaluate the similarity between a and b. Then in the algorithm, the t-th Z could be formulated as:

$$z^{(t)}=softmax(\lambda X(\mu^{(t-1)})^T), \tag{13}$$

where λ is manually set to control Z distribution. Then in the M step (likelihood maximization), we update the μ based on the weighted summation of $\mathbf{X}$ to make them in one space. Then the update process in t-th iteration could be formulated as:

$$\mu_k^{(t)}=\frac{z_{nk}^{(t)}x_n}{\sum_{m=1}^{N} z_{mk}^{(t)}} \tag{14}$$

After T iterations, we could get the final basis $\mu_k(T), \Sigma_k(T)$ and the prediction model $\theta_k(T)$. The generated pseudo label for each sample is a distribution, which can be formulated as:

$$y_n = f_\theta(x_n), \tag{15}$$

where f_θ is a linear transformation, so distribution of y_n could be easily calculated. The whole process of pseudo-label generation is summarized in algorithm 1.

Algorithm 1: Pseudo-label generation

```
Input  : $X_L, X_U, Y_L, f_\theta$
Output : $\mu_k(T), \Sigma_k(T), \theta_k(T)$
Initialize   $\mu_k(0), \Sigma_k(0), \theta(0)$
for $t \leftarrow 1$ to $T$ do
    update $z_{nk}(t)$ (eq 13)
    compute $\hat{x_n}(t)$ (eq 10)
    compute pseudo-label $y_n$ (eq 15)
    compute loss function (eq 11)
    update $\theta$(t) using back propagation
    update $\mu_k(t)$ (eq 14)
return
```

4.3 Network retraining

Because in section 4.1, we define the problem as a classification task, so in this part we simply use classification as our final task. Considering we have the distribution for pseudo-labels, there are mainly two steps in the retraining part - sample selection and model retraining.

4.3.1 Sample selection

After pseudo-label generation process, the generated pseudo-labels are formulated in a distribution format (Gaussian form) shown in equation 8 which contains variance and mean information. Then for classification task, a class-dependent selection (49) could be performed to construct a dataset with hard labels $D_{S,U} = \{x_{u,s} \in S_{u,c}, y_u\}$. Here, $S_{u,c} \in X_U$ is constructed based on the score rank of each sample, if the sample's pseudo-label has higher variance, then it's more likely to be discarded. For y_u, one can simply use its mean as its hard pseudo label, but here we want to accurately model the uncertainty information. Therefore, we randomly sample hard labels from the pseudo-label distribution to incorporate the uncertainty information encoded in the distribution.

4.3.2 Uncertainty aware retraining

After the sample selection, a retraining dataset is derived as $D_r = \{X_L, Y_L\} \bigcup \{x_{u,s}, y_u\}$, then for the retraining part, the final goal is to minimize following loss:

$$min_W \quad L_L + \frac{L_U}{Var(y)} \tag{16}$$

Method	A→W	D→W	W→D	A→D	D→A	W→A	Mean
ResNet-50 (38)	68.4±0.2	96.7±0.1	99.3±0.1	68.9±0.2	62.5±0.3	60.7±0.3	76.1
DAN (39)	80.5±0.4	97.1±0.2	99.6±0.1	78.6±0.2	63.6±0.3	62.8±0.2	80.4
RTN (40)	84.5±0.2	96.8±0.1	99.4±0.1	77.5±0.3	66.2±0.2	64.8±0.3	81.6
DANN (41)	82.0±0.4	96.9±0.2	99.1±0.1	79.7±0.4	68.2±0.4	67.4±0.5	82.2
ADDA (42)	86.2±0.5	96.2±0.3	98.4±0.3	77.8±0.3	69.5±0.4	68.9±0.5	82.9
JAN (43)	85.4±0.3	97.4±0.2	99.8±0.2	84.7±0.3	68.6±0.3	70.0±0.4	84.3
GTA (44)	89.5±0.5	97.9±0.3	99.8±0.4	87.7±0.5	72.8±0.3	71.4±0.4	86.5
MRKLD+LRENT (45)	89.4±0.7	98.9±0.4	100±0.0	88.7±0.8	72.6±0.7	70.9±0.5	86.8
[HTML]EFEFEF Ours	**92.2±0.5**	98.2±0.3	99.6±0.4	87.2±0.5	**72.8±0.3**	**72.4±0.4**	**87.1**

Table 1: Comparison on Office-31 experiments

Method	Aero	Bike	Bus	Car	Horse	Knife	Motor	Person	Plant	Skateboard	Train	Truck	Mean
Source (46)	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81	26.5	73.5	8.5	52.4
MMD (39)	87.1	63	76.5	42	90.3	42.9	85.9	53.1	49.7	36.3	85.8	20.7	61.1
DANN (41)	81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
ENT (32)	80.3	75.5	75.8	48.3	77.9	27.3	69.7	40.2	46.5	46.6	79.3	16	57
MCD (47)	87	60.9	83.7	64	88.9	79.6	84.7	76.9	88.6	40.3	83	25.8	71.9
ADR (46)	87.8	79.5	83.7	65.3	92.3	61.8	88.9	73.2	87.8	60	85.5	32.3	74.8
SimNet-Res152(48)	94.3	82.3	73.5	47.2	87.9	49.2	75.1	79.7	85.3	68.5	81.1	50.3	72.9
GTA-Res152 (44)	-	-	-	-	-	-	-	-	-	-	-	-	77.1
MRKLD+LRENT (45)	88.0	79.2	61.0	60.0	87.5	81.4	86.3	78.8	85.6	86.6	73.9	68.8	78.1
[HTML]EFEFEF Ours	89.1	81.7	82.1	57.7	83.2	79.7	83.9	77.2	86.2	82.7	83.8	65.9	**79.4**

Table 2: Comparison on VisDA17 experiments

Method	Backbone	Road	SW	Build	Wall	Fence	Pole	TL	TS	Veg.	Terrain	Sky	PR	Rider	Car	Truck	Bus	Train	Motor	Bike	mIoU
Source		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	5	1.4	21.7
CyCADA (3)	-2*DRN-26	79.1	33.1	77.9	23.4	17.3	32.1	33.3	31.8	81.5	26.7	69	62.8	14.7	74.5	20.9	25.6	6.9	18.8	20.4	39.5
Source		36.4	14.2	67.4	16.4	12	20.1	8.7	0.7	69.8	13.3	56.9	37	0.4	53.6	10.6	3.2	0.2	0.9	0	22.2
MCD (47)	-2*DRN-105	90.3	31	78.5	19.7	17.3	28.6	30.9	16.1	83.7	30	69.1	58.5	19.6	81.5	23.8	30	5.7	25.7	14.3	39.7
Source		75.8	16.8	77.2	12.5	21	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	36.6
AdaptSegNet (50)	-2*DeepLabv2	86.5	36	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 (51)	DeepLabv2	89.4	33.1	81	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		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	29.2
FCAN (52)	-2*DeepLabv2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	46.6
[HTML]EFEFEF Ours	[HTML]EFEFEFDeepLabv2	87	47.7	80.3	25.9	26.3	47.9	34.7	29	80.9	45.7	80.3	60	29.2	81.7	37.9	47.5	37.2	29.8	47.7	**50.4**

Table 3: Adaptation results of experiments transferring from GTA5 to Cityscapes.

Where W is the model parameter, L_L and L_U represent the task loss for labeled data and unlabeled data respectively, here in this classification example, they represent same classification loss like cross entropy. $Var(y)$ represents the sample uncertainty, for samples $x \in X_U$, variance is same to the variance in the distribution to catch the uncertainty information of teacher model. In this setting, samples with higher variance, which basically means that the previous model is not confident on this sample, have lower weights in the back propagation process of training. After the retraining, one round shown in figure 2 is completed. Then we simply repeat the whole process until the ideal results are derived.

5 Experiment

In this section, we demonstrate the advantages of proposed methods by comparing the performance of proposed methods with the SOTA confidence-aware self-training strategy on 2 tasks - image classification and image segmentation. To make the results comparative, we basically follow the settings in (45) which achieves SOTA results in confidence-aware self-training domain, details will be illustrated in following sections.

5.1 Dataset and evaluation metric

5.1.1 Image classification.

For domain adaption in image classification task, VisDA17 (53) and Office-31 (54) are used to evaluate the algorithm performance. In VisDA17, there are 12 classes with 152, 409 virtual images for training while 55, 400 real images from MS-COCO (55) are target dataset. For Office-31, 31 classes collected from Amazon(A, 2817 images), Webcam(W, 795 images) and DSLR(D, 498 images) domains are included. We strictly follow the settings in (54; 44; 45) which evaluate the domain adaption performance on $A \rightarrow W, D \rightarrow W, W \rightarrow D, A \rightarrow D, D \rightarrow A, W \rightarrow A$. For evaluation, we simply use the accuracy for each class and mean accuracy across all classes as the evaluation metric.

Method	Backbone	Road	SW	Build	Wall*	Fence*	Pole*	TL	TS	Veg.	Sky	PR	Rider	Car	Bus	Motor	Bike	mIoU	mIoU*
Source	2*DRN-105	14.9	11.4	58.7	1.9	0	24.1	1.2	6	68.8	76	54.3	7.1	34.2	15	0.8	0	23.4	26.8
MCD (47)		84.8	43.6	79	3.9	0.2	29.1	7.2	5.5	83.8	83.1	51	11.7	79.9	27.2	6.2	0	37.3	43.5
Source	2*DeepLabv2	55.6	23.8	74.6	-	-	-	6.1	12.1	74.8	79	55.3	19.1	39.6	23.3	13.7	25	-	38.6
AdaptSegNet(50)		84.3	42.7	77.5	-	-	-	4.7	7	77.9	82.5	54.3	21	72.3	32.2	18.9	32.3	-	46.7
Source	2*ResNet-38	32.6	21.5	46.5	4.8	0.1	26.5	14.8	13.1	70.8	60.3	56.6	3.5	74.1	20.4	8.9	13.1	29.2	33.6
CBST (45)		53.6	23.7	75	12.5	0.3	36.4	23.5	26.3	84.8	74.7	67.2	17.5	84.5	28.4	15.2	55.8	42.5	48.4
AdvEnt (51)	DeepLabv2	85.6	42.2	79.7	8.7	0.4	25.9	5.4	8.1	80.4	84.1	57.9	23.8	73.3	36.4	14.2	33	41.2	48
Source	2*DeepLabv2	64.3	21.3	73.1	2.4	1.1	31.4	7	27.7	63.1	67.6	42.2	19.9	73.1	15.3	10.5	38.9	34.9	40.3
Ours		68	29.9	76.3	**10.8**	**1.4**	**33.9**	22.8	29.5	77.6	78.3	60.6	28.3	81.6	23.5	18.8	39.8	**42.6**	**48.9**

Table 4: Adaptation results of experiments transferring from SYNTHIA to Cityscapes.

5.1.2 Semantic segmentation

For domain adaption in image segmentation tasks, 2 virtual datasets GTA5 (56), SYNTHIA (57) and 1 real dataset Cityscapes (58) are used to evaluate the performance of proposed method. Concretely, GTA5 contains 24, 966 images based on the game GTA5, SYNTHIA-RAND-CITYSCAPES (subset of SYNTHIA) has 9400 images. For the experiment setup, we also strictly follow (3; 50; 45) which use Cityscapes as target domain and view virtual datasets (GTA5 and CITYSCAPES) as training domain. For evaluation, the Intersection over Union (IoU) is used to measure the performance of models where.

5.2 Experiment setup

To make our results comparable with current SOTA confidence-aware method, we adapt the settings in (45). Besides, all the training process is performed on 4 Tesla V100 GPUs which have 32GB memory.

Image Classification: ResNet101/ ResNet-50 (38) are used as backbones, which are pretrained based on ImageNet (59). Then in source domain, we fine-tune the model using SGD while the learning rate is 1×10^{-4}, weight decay is set to be 5×10^{-5}, momentum is 0.8 and the batch size is 32. In the self-training round, the parameters are same except for the different learning rates which are 5×10^{-4}.

Image Segmentation: In image segmentation part, we mainly use the older DeepLab v2 (60) as backbone to align with previous results. DeepLab v2 is first pretrained on ImageNet and then finetuned on source domain using SGD. Here we set learning rate as 5×10^{-4}, weight decay is set to be 1×10^{-5}, momentum is 0.9, the batch size is 8 while the patch size is 512×1024. In self-training, we basically run 3 rounds which has 4 retraining epochs.

5.3 Experiment results

Comparison on image classification. As shown in table 1 and table 2, compared with previous SOTA result in confidence-aware self-training and other self-training algorithms, although our algorithm does not achieve best performance in all sub-tasks, the mean results (87.1 and 79.4 for Office-31 and VisDA17 respectively) achieves SOTA while our results (derivations and means) are obtained from 5 runs of the experiment.

Comparison on image segmentation. As shown in table 3 and 4, in semantic segmentation task, our results of average IoU (mIoU) achieves SOTA among confidence-aware self-training algorithms.

6 Conclusion and future work

In this paper, we propose a new confidence-aware self-training framework and compare our algorithm with current SOTA results of confidence-aware self-training which proves that our pseudo-label could better catch the uncertainty information and thus alleviate the over-confident issue in self-training. Furthermore, the idea underlying our method could be used in many self-training related tasks while the over-confidence is a common question faced by most self-training algorithms.

References

[1] Chen, Y., Li, W., Sakaridis, C., Dai, D., Van Gool, L.: Domain adaptive faster r-cnn for object detection in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 3339–3348

[2] Chen, Y.H., Chen, W.Y., Chen, Y.T., Tsai, B.C., Frank Wang, Y.C., Sun, M.: No more discrimination: Cross city adaptation of road scene segmenters. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 1992–2001

[3] Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A., Darrell, T.: Cycada: Cycle-consistent adversarial domain adaptation. In: International conference on machine learning, PMLR (2018) 1989–1998

[4] Kim, M., Sahu, P., Gholami, B., Pavlovic, V.: Unsupervised visual domain adaptation: A deep max-margin gaussian process approach. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 4380–4390

[5] Long, M., Cao, Z., Wang, J., Jordan, M.I.: Conditional adversarial domain adaptation. arXiv preprint arXiv:1705.10667 (2017)

[6] Busto, P.P., Iqbal, A., Gall, J.: Open set domain adaptation for image and action recognition. IEEE transactions on pattern analysis and machine intelligence **42** (2018) 413–429

[7] Chen, C., Xie, W., Huang, W., Rong, Y., Ding, X., Huang, Y., Xu, T., Huang, J.: Progressive feature alignment for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 627–636

[8] Inoue, N., Furuta, R., Yamasaki, T., Aizawa, K.: Cross-domain weakly-supervised object detection through progressive domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 5001–5009

[9] Lee, D.H., et al.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on challenges in representation learning, ICML. Volume 3. (2013)

[10] Saito, K., Ushiku, Y., Harada, T.: Asymmetric tri-training for unsupervised domain adaptation. In: International Conference on Machine Learning, PMLR (2017) 2988–2997

[11] Zou, Y., Yu, Z., Kumar, B., Wang, J.: Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: Proceedings of the European conference on computer vision (ECCV). (2018) 289–305

[12] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[13] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. (2018)

[14] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI blog **1** (2019) 9

[15] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)

[16] Scudder, H.: Probability of error of some adaptive pattern-recognition machines. IEEE Transactions on Information Theory **11** (1965) 363–371

[17] He, J., Gu, J., Shen, J., Ranzato, M.: Revisiting self-training for neural sequence generation. arXiv preprint arXiv:1909.13788 (2019)

[18] Chapelle, O., Scholkopf, B., Zien, A.: Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. IEEE Transactions on Neural Networks **20** (2009) 542–542

[19] Natarajan, N., Dhillon, I.S., Ravikumar, P., Tewari, A.: Learning with noisy labels. In: NIPS. Volume 26. (2013) 1196–1204

[20] Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., Rabinovich, A.: Training deep neural networks on noisy labels with bootstrapping. arXiv preprint arXiv:1412.6596 (2014)

[21] Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., Fergus, R.: Training convolutional networks with noisy labels. arXiv preprint arXiv:1406.2080 (2014)

[22] Yu, Z., Liu, W., Zou, Y., Feng, C., Ramalingam, S., Kumar, B., Kautz, J.: Simultaneous edge alignment and learning. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 388–404

[23] Mukherjee, S., Awadallah, A.H.: Uncertainty-aware self-training for text classification with few labels. arXiv preprint arXiv:2006.15315 (2020)

[24] Zou, Y., Yu, Z., Liu, X., Kumar, B., Wang, J.: Confidence regularized self-training. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 5982–5991

[25] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network (2015)

[26] Sucholutsky, I., Schonlau, M.: Improving dataset distillation. CoRR **abs/1910.02551** (2019)

[27] Wang, T., Zhu, J., Torralba, A., Efros, A.A.: Dataset distillation. CoRR **abs/1811.10959** (2018)

[28] Li, X., Zhong, Z., Wu, J., Yang, Y., Lin, Z., Liu, H.: Expectation-maximization attention networks for semantic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 9167–9176

[29] Moon, T.K.: The expectation-maximization algorithm. IEEE Signal processing magazine **13** (1996) 47–60

[30] Amini, M.R., Gallinari, P.: Semi-supervised logistic regression. In: ECAI. (2002) 390–394

[31] Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: 33rd annual meeting of the association for computational linguistics. (1995) 189–196

[32] Grandvalet, Y., Bengio, Y., et al.: Semi-supervised learning by entropy minimization. In: CAP. (2005) 281–296

[33] Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. arXiv preprint arXiv:1610.02242 (2016)

[34] Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. arXiv preprint arXiv:1703.01780 (2017)

[35] Luo, Y., Zhu, J., Li, M., Ren, Y., Zhang, B.: Smooth neighbors on teacher graphs for semi-supervised learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 8896–8905

[36] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: Series B (Methodological) **39** (1977) 1–22

[37] Richardson, S., Green, P.J.: On bayesian analysis of mixtures with an unknown number of components (with discussion). Journal of the Royal Statistical Society: series B (statistical methodology) **59** (1997) 731–792

[38] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778

[39] Long, M., Cao, Y., Wang, J., Jordan, M.: Learning transferable features with deep adaptation networks. In: International conference on machine learning, PMLR (2015) 97–105

[40] Long, M., Zhu, H., Wang, J., Jordan, M.I.: Unsupervised domain adaptation with residual transfer networks. arXiv preprint arXiv:1602.04433 (2016)

[41] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The journal of machine learning research **17** (2016) 2096–2030

[42] Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 7167–7176

[43] Long, M., Zhu, H., Wang, J., Jordan, M.I.: Deep transfer learning with joint adaptation networks. In: International conference on machine learning, PMLR (2017) 2208–2217

[44] Sankaranarayanan, S., Balaji, Y., Castillo, C.D., Chellappa, R.: Generate to adapt: Aligning domains using generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 8503–8512

[45] Zou, Y., Yu, Z., Liu, X., Kumar, B.V.K.V., Wang, J.: Confidence regularized self-training. CoRR **abs/1908.09822** (2019)

[46] Saito, K., Ushiku, Y., Harada, T., Saenko, K.: Adversarial dropout regularization. arXiv preprint arXiv:1711.01575 (2017)

[47] Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 3723–3732

[48] Pinheiro, P.O.: Unsupervised domain adaptation with similarity learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 8004–8013

[49] Mukherjee, S., Awadallah, A.H.: Uncertainty-aware self-training for text classification with few labels. CoRR **abs/2006.15315** (2020)

[50] Tsai, Y.H., Hung, W.C., Schulter, S., Sohn, K., Yang, M.H., Chandraker, M.: Learning to adapt structured output space for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 7472–7481

[51] Vu, T.H., Jain, H., Bucher, M., Cord, M., Pérez, P.: Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 2517–2526

[52] Zhang, Y., Qiu, Z., Yao, T., Liu, D., Mei, T.: Fully convolutional adaptation networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 6810–6818

[53] Peng, X., Usman, B., Kaushik, N., Wang, D., Hoffman, J., Saenko, K.: Visda: A synthetic-to-real benchmark for visual domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. (2018) 2021–2026

[54] Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: European conference on computer vision, Springer (2010) 213–226

[55] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755

[56] Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: European conference on computer vision, Springer (2016) 102–118

[57] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 3234–3243

[58] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 3213–3223

[59] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, Ieee (2009) 248–255

[60] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40** (2017) 834–848

A Appendix

A.1 Basis extraction illustration

In this paper, an attention-like module is added (ATT block in figure 3) to extract the basis from features, which are then used to be the initialized basis in Gaussian Mixture Model. Before demonstrating the details of our pseudo-label generation process, we first illustrate the intuition of using 'weights' as initialized basis.

A.1.1 Logistic regression and data information

A small dataset could contain the information of a huge dataset while this 'small dataset' could be viewed as the basis of original dataset. In this section, we'll explain why this could happen based on logistic regression, which is a simple machine learning model and can be easily understood.

In logistic regression,

Assuming the labels are [0, 1], the loss function could be written as follows without considering the differentiation:

$$L(w, x) = \sum_{i=1}^{N} \left[h\left(w, x_i\right) - y_i\right]^2 \tag{17}$$

This could be seen as a specific example of attention-based model, which only has one kind of weight. Now, assuming we have already gained the best weight w_0 and we make it the only data sample:

$$x_i = w_0, i = 1 \tag{18}$$

If the label of x_i is 1, then the new data sample becomes $x_i, 1$. Then, if we retrain the logistic model using only this one sample, when the model converges, we could find the new weight w', and apparently $w' = w_0$. Because in equation 17, we just change N to 0, and only when $w' = w_0$, the loss would be 0.

The above result means that the model trained on only one synthesized data sample could achieve similar or even same performance as the model trained on the original whole dataset. Because the weight of the logistic model could be seen as the projection of the original data, while the data whose label is 1 should have more correlation with weight (according to the loss, h(w,x) could be seen as a correlation equation).

A.1.2 Support Vector Machine (SVM) and base

SVM could also be helpful in understanding our idea.

In SVM, the most important data points are called support vectors, while these vectors are also data in initial dataset. And the loss function is:

$$L(\alpha) = \sum_{i=1}^{N} \left| sgn\left[\sum_{j=1}^{N} \alpha_j y_j \Phi\left(x_j, x_i\right)\right] - y_i \right|^2 + \lambda \sum_{i=1}^{N} \alpha_i \tag{19}$$

Now, let's assume a self attention model with 4 weights w_1, w_2, w_3, w_4:

$$y_i = sgn\left[\sum_{k=1}^{K} h\left(w_h, x_i\right) \beta_b\right] \tag{20}$$

The loss function then becomes:

$$L(W) = \sum_{i=1}^{N} \left[sgn\left[\sum_{k=1}^{K} h\left(w_k, x_i\right) \beta_k\right] - y_i \right]^2 \tag{21}$$

Apparently, two loss functions (equation 19 and 21) are similar, the only difference is the optimized parameter. In SVM, we need to find the support vectors while in attention model, we need to find the optimized matrix. Then if you come back to the logistic part, you could find that in logistic regression, we want to find the basis (weights in previous section), these bases are called support vectors in SVM.

Therefore, using weights to be the initialized basis is a reasonable direction while these bases could contain the information inside the original dataset or in the feature space. However, the uncertainty information is the thing we try to extract in this paper. Therefore, GMM is used to represent the data uncertainty information while EM is used to optimize the result. Following sections will illustrate technical details of this idea.