A STUDY OF UNSUPERVISED EVALUATION METRICS FOR PRACTICAL AND AUTOMATIC DOMAIN ADAPTATION
- SUPPLEMENTARY MATERIAL -

Anonymous authors
Paper under double-blind review

A IMPLEMENTATIONS OF UNSUPERVISED DOMAIN ADAPTATION METRICS

Metric	w. Source Accuracy	Hard to Attack	Input-level
$\mathcal{A}$-distanceBen-David et al. (2006)	✓	✓	
$\mathcal{H}\Delta\mathcal{H}$-divergence or MCD Ben-David et al. (2010); Saito et al. (2018)	✓	✓	
MDD Zhang et al. (2019)	✓	✓	
Deep Embedded Validation (DEV) You et al. (2019)	✓		
DEVN Musgrave et al. (2022)	✓		
Entropy Grandvalet & Bengio (2004); Vu et al. (2019)			
Soft Neighborhood Density (SND) Saito et al. (2021)			
Mutual Information Shi & Sha (2012)			
BNM Musgrave et al. (2022)			
ClassAMI Musgrave et al. (2022)			
ISM (ours)	✓	✓	
ACM (ours)	✓	✓	✓

Table 1: The metrics of UDA studied in the paper. We implement previous metrics according to their papers and modify them to be positively correlated with target accuracy.

A.1 DISCREPANCY-BASED METRIC:

Ben-David's theory Ben-David et al. (2006; 2010) shows that the error rate of a classifier on the target domain can be bounded by the error rate on the source domain and the domain divergence:

$$\epsilon_T(h) \leq \epsilon_S(h) + d_{\mathcal{H}}\left(\mathcal{D}_S, \mathcal{D}_T\right) + \lambda \tag{1}$$

where $\lambda = \lambda_T + \lambda_S$, and λ_T and λ_S are the errors of $h^* = \text{argmin}_{h\in H}\left(\epsilon_T(h) + \epsilon_S(h)\right)$ with respect to $\mathcal{D}_T$ and $\mathcal{D}_S$ respectively. Later works Ganin et al. (2016) exploits this bound to optimize the domain divergence and source error to minimize the target error. Inspired by this formula, we think that the target error can be approximated by domain divergence and source error. In other words, we can utilize domain divergence and source accuracy as the evaluation metric to measure target accuracy.

We transform these discrepancy-based UDA methods Ben-David et al. (2006; 2010); Saito et al. (2018); Zhang et al. (2019) into UDA metrics. These metrics are composited by source accuracy and domain divergence. We can formalize the UDA metrics as:

$$\mathcal{M}(\mathcal{D}_S, \mathcal{D}_T, \boldsymbol{M}) = A_S(\mathcal{D}_S, \boldsymbol{M}) - d_{\mathcal{M}}(\mathcal{D}_S, \mathcal{D}_T, \boldsymbol{M}) \tag{2}$$

where $\boldsymbol{M}$ is the model to be evaluated, A_S is source accuracy and $d_{\mathcal{M}}$ is the domain divergence. The model is composed of a feature generator and a classifier: $\boldsymbol{M} = \mathbf{f}(\mathbf{g}(\cdot))$. Different metrics for UDA have different $d_{\mathcal{M}}$ terms, we describe each $d_{\mathcal{M}}$ terms in the following.

1) $\mathcal{A}$-distance Ben-David et al. (2006):

$$d_{\mathcal{A}} = 2\sup_{h\in\mathcal{H}} \left|\mathbb{E}_{\mathcal{D}_s} I\left[h=1\right] + \mathbb{E}_{\mathcal{D}_t} I\left[h=0\right]\right|$$

A domain discriminator h is trained and the accuracy of the domain discriminator is used as the metric. We use one linear layer to model the domain discriminator the same as Jiang et al. (2020). Notably, when evaluating metrics, we only have the validation set of the source and the target domain, but we need to train the domain discriminator on a training set and evaluate it on the other set. So we use 3-fold validation: we split the validation set into three parts, and each time we train the domain discriminator on two parts and evaluate it on the left part. If not specified, for the following metric that needs training additional networks, we use this 3-fold validation to get the metric score.

2) $\mathcal{H}\Delta\mathcal{H}$-divergence or MCD Ben-David et al. (2010); Saito et al. (2018):

$$d_{\mathcal{H}\Delta\mathcal{H}}=\sup_{h,h'\in\mathcal{H}}\left|\mathbb{E}_{\mathcal{D}_s}I\left[h'\neq h\right]-\mathbb{E}_{\mathcal{D}_t}I\left[h'\neq h\right]\right|$$

Two additional classifiers are trained on the top of the feature. Apart from supervised training on source features, they also need to agree on the source domain and disagree on the target domain. These two classifiers are modeled by one linear layer.

3) Maximum Mean Discrepancy (MDD) Zhang et al. (2019):

$$d_{f,\mathcal{F}}^{(\rho)}(\mathcal{D}_s,\mathcal{D}_t)=\sup_{f'\in\mathcal{F}}\left(\operatorname{disp}_{\mathcal{D}_s}^{(\rho)}\left(f,f'\right)-\operatorname{disp}_{\mathcal{D}_t}^{(\rho)}\left(f,f'\right)\right)$$

where f is the classifier of the evaluated model and $\operatorname{disp}^{(\rho)}$ is the margin error. An additional classifier f' is trained to agree f on the source domain and disagree f on the target domain. f' is modeled by a linear layer and trained by the algorithm: Eq. (30) in the original paper Zhang et al. (2019).

A.2 Importance Weighted Validation Metric:

4) Deep Embedded Validation (DEV) You et al. (2019):

DEV is based on the Importance-Weighted cross-validation (IWCV) of the source domain. It needs to train a two-layer domain discriminator h first, and compute IWCV:

$$\ell(\mathbf{x}_i^s)=w_h\left(\mathbf{x}_i^s\right)I\left(\hat{y}_i^s\neq y_i^s\right)$$
$$w_h\left(\mathbf{x}_i^s\right)=\frac{n_s}{n_t}\frac{1-h\left(\boldsymbol{z}_i^s\right)}{h\left(\boldsymbol{z}_i^s\right)}$$

Then DEV adds IWCV and the variance of the risk estimation as follows:

$$\boldsymbol{DEV}=\operatorname{mean}(\ell)+\eta\operatorname{mean}(W)-\eta \tag{3}$$

$$\eta=-\frac{\widehat{\operatorname{Cov}}\left(\ell,w_h\right)}{\widehat{\operatorname{Var}}\left[w_h\right]} \tag{4}$$

We use the negative DEV to be positively related to accuracy.

5) DEV with normalization (DEVN) Musgrave et al. (2022):

In Musgrave et al. (2022), they propose to normalize the weights by either max normalization or standardization to avoid large η. We implement DEVN with standardization:

$$W_{st}=\frac{W-\bar{W}}{\sigma_W}+1 \tag{5}$$

Then W_{st} is used in $\boldsymbol{DEV}$.

A.3 Entropy-based Metric:

6) Entropy Morerio et al. (2017); Vu et al. (2019):

$$\boldsymbol{Ent}=-\mathbb{E}_{\mathcal{D}_t}[H(\boldsymbol{p})]=\mathbb{E}_{\mathcal{D}_t}[\sum_k\boldsymbol{p}_k\log\boldsymbol{p}_k]$$

We compute the negative entropy of the predicted probability $\boldsymbol{p}$ of $\boldsymbol{M}$ on target samples.

7) Soft Neighborhood Density (SND) Saito et al. (2021):

In their work, they define the soft neighborhoods as the similarity distribution between target samples and estimate the density by computing the entropy of the distribution. The similarity is defined as $S_{ij} = \langle \boldsymbol{p}_i^t, \boldsymbol{p}_j^t \rangle$. The similarity distribution is computed as follows:

$$P_{ij} = \frac{\exp\left(S_{ij}/\tau\right)}{\sum_{j'} \exp\left(S_{ij'}/\tau\right)} \tag{6}$$

Then SND is defined as:

$$\boldsymbol{SND} = -\frac{1}{N_t} \sum_{i=1}^{N_t} \sum_{j=1}^{N_t} P_{ij} \log P_{ij} \tag{7}$$

8) Mutual Information Shi & Sha (2012):

The Mutual Information of the model prediction:

$$\boldsymbol{MI} = H(\mathbb{E}_{\mathcal{D}_t}[\boldsymbol{p}]) - \mathbb{E}_{\mathcal{D}_t}[H(\boldsymbol{p})]$$

A.4 OTHER METRIC:

9) Batch nuclear-norm maximization (BNM) Cui et al. (2020); Musgrave et al. (2022):

BNM is a UDA algorithm that aims to generate diverse and confident predictions. It approaches this via singular value decomposition:

$$\boldsymbol{BNM} = \|P\|_* \tag{8}$$

where P is the $\tilde{N}_t \times K$ prediction matrix ($\tilde{N}_t$ is the target validation set size, and K is the number of classes), and $\|P\|_*$ is the nuclear norm (the sum of the singular values) of P.

10) ClassAMI Musgrave et al. (2022):

They propose computing the Adjusted Mutual Information (AMI) between target cluster labels and the predicted labels:

$$\boldsymbol{ClassAMI} = \mathrm{AMI}(P, \mathrm{kmeans}(F).\mathrm{labels}) \tag{9}$$

$$P_i = \underset{k}{\mathrm{argmax}}[\boldsymbol{p}_i] \tag{10}$$

where P is the predicted labels for the target data, $\boldsymbol{p}_i$ is the i-th prediction vector, and F is the set of target features.

A.5 OUR METRIC:

11) Inception Score Metric for UDA (ISM):

Its formula is presented in the main paper. The MLP classifier $\boldsymbol{h}$ has two layers with a hidden size equal to the feature size (bottleneck dimension). The classifier is trained by the LBFGS optimizer for 200 steps on the source validation set.

12) Augmentation Consistency Metric (ACM):

Its formula is presented in the main paper. For the data-augmented sample, we use a series of random data augmentation to get it, including Random Resize and Crop, Horizontal Flip, Random Color Jitter, and Random Gaussian Blur.

Office31	A	W	D	
Training	1,971	556	498	
Validation	846	239	498	
OfficeHome	Ar	Cl	Pr	Rw
Training	1,698	3,055	3,107	3,049
Validation	729	1,310	1,332	1,308
DomainNet	c	p	r	s
Training	33,525	50,416	120,906	48,212
Validation	14,604	21,850	52,041	20,916
VisDA	Syn	Real		
Training	106,677	55,388		
Validation	45,720	72,372		

Table 2: The statistic of the training set and the validation set of the datasets used in the paper. Following the 70%/30% scheme, we split Office31, OfficeHome, and the "Synthetic" domain of VisDA into no overlapping training and validation sets.

B DETAILS OF UDA TRAINING

B.1 DATASETS SPLITTING

Generally speaking, the validation set contains different samples with the training set to show generalization. However, Office31 and OfficeHome do not split the training and validation set, so previous works report the accuracy of the target training set. To solve this historical issue, we follow a 70%/30% split scheme to split the training and the validation set for Office31 and OfficeHome (except for "D" domain of Office31, due to limited samples) and report the target accuracy on the validation set. We also split the training and validation set for the source domain of VisDA2017, as metrics will utilize the source validation set. The training sets and validation sets are listed in Tab 2

B.2 TRAINING IMPLEMENT DETAILS

Following Transfer-Learning-Library Jiang et al. (2020), we train all five methods (Source only, DANN, CDAN, MDD, MCC) through SGD with 0.9 momentum, and the learning rate of ResNet backbone is scaled by 0.1. We schedule the learning rate with the commonly used strategy: the learning rate is adjusted by $\eta_p = \frac{\eta_0}{(1+\alpha q)^\beta}$, where q is the training progress linearly changing from 0 to 1, $\eta_0 = 0.01$, $\alpha = 10$, $\beta = 0.75$. The batch size is set to 32 for all training. We train each model with one V100 GPU. For the architecture of the model, $\boldsymbol{M} = \mathbf{f}(\mathbf{g}(\cdot))$, the feature generator $\mathbf{g}$ contain a ResNet He et al. (2016) backbone and a bottleneck layer, and the classifier $\mathbf{f}$ is a linear layer. We use ResNet50 as the backbone for Office31 and OfficeHome, and ResNet101 for VisDA and DomainNet. We train every model for 3000 steps on Office31, OfficeHome, and VisDA, and 6000 steps on DomainNet in total.

C EXPERIMENTAL RESULTS

C.1 MORE CONSISTENCY RESULTS

In Section 4.2 of the main paper, we show the results of comparing our ISM and ACM to previous metrics on three datasets. In the Appendix, we add three more previous metrics Musgrave et al. (2022): DEVN, BNM, and ClassAMI. We show the consistency of each metric with target accuracy on four UDA datasets with five UDA training methods. Tab. 1, Tab. 5, Tab. 4 and Tab. 6 show the results on VisDA2017, DomainNet, OfficeHome and Office31 respectively.

We find the performance of the metric can vary largely for different training methods. DEVN, BNM, and ClassAMI demonstrate excellent performance in certain cases, yet they may also exhibit significant errors under other conditions. Our ISM and ACM show decent performance for all training methods on all datasets. We find the results of all metrics

Datasets	OfficeHome				VisDA2017				DomainNet			
Train Method	MI		AC		MI		AC		MI		AC	
Metric	corr	dev	corr	dev	corr	dev	corr	dev	corr	dev	corr	dev
MI	0.67	7.97	0.55	5.76	- 0.21	15.3	0.87	1.99	-0.75	18.9	0.95	1.62
ISM	0.89	1.73	**0.97**	1.43	0.87	1.77	0.89	1.57	0.88	**0.0**	**0.97**	**0.62**
ACM	**0.92**	**1.15**	0.62	**1.37**	**0.99**	**0.0**	**0.89**	**0.59**	**0.91**	**0.0**	0.96	1.55

Table 3: A test of whether metrics are attackable on different datasets. These three metrics are transformed into training loss, and then trained models are evaluated by themselves.

Training Method	Source only		DANN		CDAN		MDD		MCC		ALL	
Metric	corr	dev	corr	dev	corr	dev	corr	dev	corr	dev	corr	dev
MDD	-0.65	7.39	0.58	6.29	-0.11	4.73	0.66	0.85	0.34	4.38	0.36	5.93
DEVN	-0.6	7.39	-0.16	6.67	-0.15	8.99	0.55	4.97	0.47	4.99	0.37	32.51
BNM	0.11	6.35	0.60	3.65	-0.03	3.72	0.85	**0.00**	0.02	3.41	0.48	3.41
ClassAMI	-0.40	7.39	0.74	6.29	-0.12	9.81	**0.94**	1.29	0.15	7.19	0.61	7.76
ISM	**0.84**	**0.31**	0.75	3.92	0.42	1.23	0.75	0.40	0.88	**0.66**	0.59	**1.66**
ACM	0.80	2.38	**0.79**	**1.18**	**0.61**	**0.98**	0.85	**0.0**	**0.93**	1.66	**0.76**	**1.66**

Figure 1: Consistency between metrics of UDA and target accuracy on VisDA2017.

Training Method	Source only		DANN		CDAN		MDD		MCC		ALL	
Metric	corr	dev	corr	dev	corr	dev	corr	dev	corr	dev	corr	dev
MDD	0.53	4.44	0.72	1.67	0.83	1.88	0.83	1.13	0.05	5.99	0.3	6.2
DEVN	0.18	3.48	-0.06	6.38	0.08	7.10	0.89	12.03	0.27	8.45	0.52	10.14
BNM	-0.59	6.63	0.48	4.53	0.88	1.30	0.93	1.98	0.37	17.19	0.54	17.28
ClassAMI	**0.79**	3.57	**0.77**	2.93	0.86	1.25	0.93	1.08	0.71	**1.16**	0.81	**1.24**
ISM	0.72	1.47	0.6	1.68	**0.91**	1.15	**0.97**	1.45	0.70	1.96	0.88	1.96
ACM	0.75	**1.37**	**0.77**	**1.16**	0.90	**1.13**	0.95	**0.93**	**0.94**	1.36	**0.93**	1.73

Table 4: The "corr" and "dev" results are averaged over the 12 transfer tasks of OfficeHome.

Training Method	Source only		DANN		CDAN		MDD		MCC		ALL	
Metric	corr	dev	corr	dev	corr	dev	corr	dev	corr	dev	corr	dev
MDD	0.26	2.12	0.54	3.46	0.72	1.11	0.7	12.75	-0.55	10.39	0.54	4.99
DEVN	0.81	1.47	0.67	5.40	0.80	7.99	0.87	0.38	0.50	9.95	0.0	2.45
BNM	0.34	3.83	0.58	6.65	0.65	4.28	0.34	14.28	0.74	1.37	0.59	2.85
ClassAMI	0.57	**1.27**	0.52	3.46	0.60	5.62	0.61	**0.04**	0.81	1.37	0.65	3.55
ISM	0.85	1.33	**0.87**	0.7	0.96	0.41	**0.98**	0.28	**0.92**	0.6	**0.91**	1.13
ACM	**0.94**	1.41	0.8	**0.27**	**0.98**	**0.29**	0.93	**0.04**	0.84	**0.15**	0.87	**0.29**

Table 5: The "corr" and "dev" results are averaged over 12 transfer tasks of DomainNet.

Training Method	DANN		CDAN		ALL	
Metric	corr	dev	corr	dev	corr	dev
$\mathcal{A}$-distance	0.37	1.69	0.34	1.3	0.35	3.11
MCD	0.46	1.48	0.32	1.71	0.48	2.20
MDD	0.53	2.46	0.38	1.34	0.63	2.22
DEV	NaN	-	NaN	-	NaN	-
DEVN	NaN	-	NaN	-	NaN	-
Entropy	0.4	2.47	0.57	2.59	0.55	2.01
SND	0.43	6.70	0.44	3.09	0.57	6.14
MI	0.38	2.26	0.58	1.51	0.53	2.74
BNM	0.29	2.99	0.54	1.75	0.32	3.59
ClassAMI	0.56	1.83	0.61	2.06	0.67	3.46
ISM	**0.73**	1.52	**0.63**	**1.04**	0.71	**1.41**
ACM	0.71	**1.46**	0.59	1.21	**0.75**	1.84

Table 6: Consistency between metrics of UDA and target accuracy on Office31. The results are averaged across 6 transfer tasks of Office31.

decrease on Office31, which may be due to small validation sets. DEV and DEVN will collapse on Office31 because source accuracy can be 1.

Method	c → p	c → r	c → s	p → c	p → r	p → s	r → c	r → p	r → s	s → c	s → p	s → r	Avg
DANN (default)	35.9	54.3	43.8	38.0	54.9	35.5	49.8	50.1	38.3	54.4	43.8	53.2	46.0
DANN (searched)	38.0	54.5	44.7	40.7	56.1	37.9	50.7	50.7	38.3	55.0	44.7	53.7	47.1
Gains (+Δ)	+2.1	+0.2	+0.9	+2.7	+1.2	+2.4	+0.9	+0.6	+0.0	+0.6	+0.9	+0.5	**+1.1**
CDAN (default)	40.0	55.8	44.6	44.2	57.3	39.8	55.2	53.3	41.5	56.9	46.3	55.5	49.2
CDAN (searched)	40.6	56.5	45.1	45.5	58.4	40.3	55.4	53.1	42.3	57.1	46.6	56.4	49.8
Gains (+Δ)	+0.6	+0.7	+0.5	+1.3	+1.1	+0.5	+0.2	-0.2	+0.8	+0.2	+0.3	+0.9	**+0.6**
MDD (default)	42.3	58.4	46.6	48.5	60.1	43.6	56.8	56.3	46.3	57.2	44.8	57.2	51.5
MDD (searched)	42.5	58.6	47.0	48.5	60.1	43.6	57.3	55.9	46.6	57.5	45.0	57.2	51.7
Gains (+Δ)	+0.2	+0.2	+0.4	+0.0	+0.0	+0.0	+0.5	-0.4	+0.3	+0.3	+0.2	+0.0	**+0.2**
MCC (default)	35.1	49.2	40.6	41.0	56.0	36.2	48.3	49.0	36.3	51.9	38.9	49.9	44.4
MCC (searched)	41.2	53.6	44.5	51.1	59.9	40.7	58.5	54.8	38.2	61.7	47.6	55.0	50.6
Gains (+Δ)	+6.1	+4.4	+3.9	+10.1	+3.9	+4.5	+10.2	+5.8	+1.9	+9.8	+8.7	+5.1	**+6.2**

Table 7: The hyper-parameters found by our metric v.s. the default hyper-parameters in original papers on DomainNet.

Method	Ar → Cl	Ar → Pr	Ar → Rw	Cl → Ar	Cl → Pr	Cl → Rw	Pr → Ar	Pr → Cl	Pr → Rw	Rw → Ar	Rw → Cl	Rw → Pr	Avg
DANN (default)	49.0	61.3	72.9	53.5	66.6	68.6	55.0	50.4	75.2	67.1	56.3	79.3	62.9
DANN (searched)	51.2	62.3	74.2	56.7	66.0	70.8	58.7	52.7	76.0	67.5	57.8	80.8	64.5
Gains (+Δ)	+2.2	+1.0	+1.3	+3.2	-0.6	+2.2	+3.7	+2.3	+0.8	+0.4	+1.5	+1.5	**+1.6**
CDAN (default)	50.4	69.4	73.5	56.7	69.4	69.1	57.3	50.5	75.5	70.6	55.8	80.6	64.9
CDAN (searched)	51.1	69.2	74.3	58.4	70.3	69.7	61.6	50.6	77.5	71.4	56.7	81.1	66.0
Gains (+Δ)	+0.7	-0.3	+0.8	+1.7	+0.7	+0.6	+4.3	+0.1	+2.0	+0.8	+0.9	+0.5	**+1.1**
MDD (default)	51.1	70.6	72.1	57.3	70.6	76.6	59.5	53.9	74.9	70.5	58.6	81.7	66.4
MDD (searched)	52.9	72.2	75.2	58.8	71.9	76.6	58.7	52.4	76.8	69.8	59.2	81.8	67.2
Gains (+Δ)	+1.8	+1.6	+3.1	+1.5	+1.3	+0.0	-0.8	-1.5	+1.9	-0.7	+0.6	+0.1	**+0.8**
MCC (default)	55.5	77.7	80.2	62.8	75.2	75.8	61.7	50.6	78.3	69.7	56.3	83.4	68.9
MCC (searched)	56.1	78.5	79.0	63.6	75.2	76.6	64.1	52.3	78.3	71.6	56.1	83.5	69.5
Gains (+Δ)	+0.6	+0.8	-1.2	+0.8	+0.0	+0.8	+2.4	+1.7	+0.0	+1.9	-0.2	+0.1	**+0.6**

Table 8: The hyper-parameters found by our metric v.s. the default hyper-parameters in original papers on OfficeHome. The target accuracy of 12 transfer tasks is reported.

Method	A → W	A → D	W → A	W → D	D → A	D → W	Avg
DANN (default)	90.4	81.7	69.6	97.8	72.3	93.7	84.3
DANN (searched)	90.6	83.9	69.6	98.6	72.3	95.0	85.1
Gains (+Δ)	+0.2	+2.2	+0.0	+0.8	+0.0	+1.3	**+0.8**
CDAN (default)	91.2	93.0	68.2	100.0	72.1	97.1	86.9
CDAN (searched)	91.6	91.5	69.6	100.0	74.1	97.5	87.4
Gains (+Δ)	+0.4	-1.5	+1.4	+0.0	+2.0	+0.4	**+0.5**

Table 9: The hyper-parameters found by our metric v.s. the default hyper-parameters in original papers on Office31.

C.2 ROBUSTNESS RESULTS

For the "Robustness" property of metrics, we transform MI and ACM into two training methods. We employ these metrics to select the trade-off λ from $\{0.1, 0.3, 1.0, 3.0, 10.0\}$ for these methods. We show the implementation of these methods here. The loss of the "Mutual Information" method is as follows:

$$L_{MI} = \mathbb{E}_{(\boldsymbol{x}^s, y^s)}[-\log \boldsymbol{p}_{y^s}] + \lambda(\sum_k \hat{\boldsymbol{p}}_k \log \hat{\boldsymbol{p}}_k - \mathbb{E}_{\boldsymbol{x}^t}[\sum_k \boldsymbol{p}_k \log \boldsymbol{p}_k]), \tag{11}$$

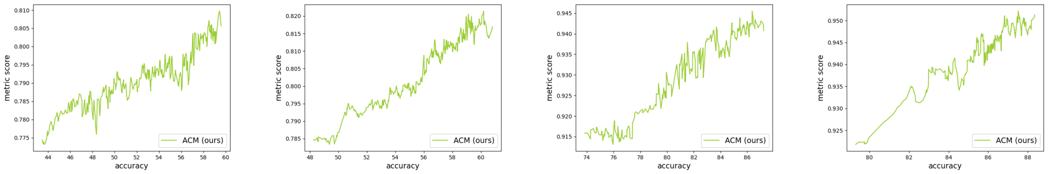
where $\hat{\boldsymbol{p}}_k$ is the average prediction for class k within a batch.

The loss of the "Augment Consist" method is as follows:

$$L_{AC} = \mathbb{E}_{(\boldsymbol{x}^s, y^s)}[-\log \boldsymbol{p}_{y^s}] - \lambda \mathbb{E}_{\boldsymbol{x}^t} \sum_k I[k = \underset{k}{\operatorname{argmax}}(\boldsymbol{p}^t)] \log \boldsymbol{p}^{t\prime}$$

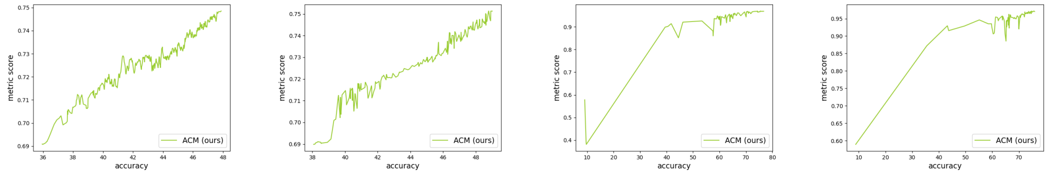
where $\boldsymbol{p}^{t\prime}$ is the prediction of the model on the sample $\boldsymbol{x}^{t\prime}$, which is the augmented version of $\boldsymbol{x}^t$. We use the same random data augmentation as the "ACM" metric.

We show the study of the Robustness property on OfficeHome, VisDA2017, and DomainNet in Tab 3 for "MI", "ISM" and "ACM" metrics. When the models are trained with the "MI" method, the "MI" metric is inconsistent with the target accuracy. Meanwhile, the "ISM" metric is robust to this attack. "ACM" is also robust against the attack against it.



(a) DANN on OfficeHome (b) CDAN on OfficeHome (c) DANN on Office31 (d) CDAN on Office31

Figure 2: The visualization of the relation between the ACM score and target accuracy. Each sub-figure contains models trained by DANN or CDAN methods on OfficeHome or Office31 datasets.



(a) DANN on DomainNet (b) CDAN on DomainNet (c) DANN on VisDA (d) CDAN on VisDA

Figure 3: The visualization of the relation between the ACM score and target accuracy. Each sub-figure contains models trained by DANN or CDAN methods on DomainNet or VisDA2017 datasets.

C.3 Hyperparameter Searching Results

We show the results of hyper-parameter searching on DomainNet, OfficeHome, and Office31 in Tab 7, Tab 8, and Tab 9. The hyper-parameters found by our ACM metric outperform the default hyper-parameters for all four training methods.

D Visualizations

We visualize the consistency between the metric score of our ACM and target accuracy and get some sense of Pearson's correlation between them. In Fig. 2 and Fig. 3, we plot the metric score according to the target accuracy of the models trained by the DANN (CDAN) method on various datasets.

As we can see from the figures, it is clear that the ACM score is positively related to target accuracy. Therefore, when the target accuracy increases, the ACM score tends to increase. This correlation is especially obvious in OfficeHome and DomainNet datasets.

E Limitations and Future Works

Although we have studied various UDA metrics and proposed new metrics for UDA evaluation, the best derivation of the best model ("dev") remains 1%-2%. It is desirable to propose a new metric that better meets the three criteria of robust metrics. One possible direction is combining multiple metrics to evaluate the model. Meanwhile, the time cost of evaluating the metric should also be considered, and metrics in the paper require, at most, to train a simple network. In the paper, we use a simple TPE searcher and relatively small search spaces. More advanced searching strategies and neural architecture searching Zoph & Le (2016) for UDA can be explored. The paper mainly focuses on the single-source UDA for close-set classification. The unsupervised metric for more transfer learning scenarios can be studied, e.g., Partial UDA, Source-Free UDA, UDA for object detection, semantic segmentation, and depth estimation.

REFERENCES

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando C Pereira. Analysis of representations for domain adaptation. In NeurIPS, 2006.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando C Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. Machine Learning, 79:151–175, 2010.

Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations. In CVPR, 2020.

Yaroslav Ganin, E. Ustinova, Hana Ajakan, Pascal Germain, H. Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. Domain-adversarial training of neural networks. In Journal of Machine Learning Research, 2016.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In NeurIPS, 2004.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

Junguang Jiang, Baixu Chen, Bo Fu, and Mingsheng Long. Transfer-learning-library. `https://github.com/thuml/Transfer-Learning-Library`, 2020.

Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. ArXiv, abs/1711.10288, 2017.

Kevin Musgrave, Serge J. Belongie, and Ser Nam Lim. Three new validators and a large-scale benchmark ranking for unsupervised domain adaptation. ArXiv, 2022.

Kuniaki Saito, Kohei Watanabe, Y. Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In CVPR, 2018.

Kuniaki Saito, Donghyun Kim, Piotr Teterwak, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Tune it the right way: Unsupervised validation of domain adaptation via soft neighborhood density. In ICCV, 2021.

Yuan Shi and Fei Sha. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In ICML, 2012.

Tuan-Hung Vu, Himalaya Jain, Max Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In CVPR, 2019.

Kaichao You, Ximei Wang, Mingsheng Long, and Michael I. Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In ICML, 2019.

Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael I. Jordan. Bridging theory and algorithm for domain adaptation. In ICML, 2019.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In ICLR, 2016.