Conditional Adversarial Domain Adaptation
(2018)

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan]

Summary

Contributions

Given a source and target domain, where the source domain is labeled and the target domain is unlabeled. We want to train a model to perform well on target, even if we have a distribution shift between source and target, for examples synthetic and real images. The popular approach of Adversarial Domain Adaptation tries to solve this by forcing a feature extractor f to produce the same features regardless of the domain. But this is a very strong requirements, since we push the feature extractor to suppress rich information if its not present in both domain, and we might need this information to produce good results. This paper proposed CDAN, where their enforce a consistency of features, but this time over both predicted labels and features rather than only the features.

Method

In unsupervised domain adaptation, we are given a source domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, \mathbf{y}_i^s)\}_{i=1}^{n_s}$ of n_s labeled examples and a target domain $\mathcal{D}_t = \left\{\mathbf{x}_j^t\right\}_{j=1}^{n_t}$ with n_t unlabeled examples. The source domain and target domain are sampled from joint distributions $P\left(\mathbf{x}^s, \mathbf{y}^s\right)$ and $Q\left(\mathbf{x}^t, \mathbf{y}^t\right)$. With a feature extractor $\mathbf{f} = F(\mathbf{x})$ and a classifier $\mathbf{g} = G(\mathbf{x})$, the objective is to reduce the target risk $\epsilon_t(G) = \mathbb{E}_{(\mathbf{x}^t, \mathbf{y}^t)\sim Q}\left[G\left(\mathbf{x}^t\right) \neq \mathbf{y}^t\right]$. Now the objective is to push the feature extractor to produce features that are stable across domain, in traditional DANN, we train a discriminator to differentiate between the feature of each domain, and the feature extractor needs to confuse it. And the whole model (feature extractor, discriminator, classifier) are trained jointly. The feature extractor and discriminator on the whole dataset and the classifier only on the labels source examples.

CDAN proposes to feed the classifier the features conditioned on the classifier output, the lass in this case is:

$$\mathcal{E}(G) = \mathbb{E}_{\left(\mathbf{x}_i^*, \mathbf{y}_i^*\right)\sim\mathcal{D}_s} L\left(G\left(\mathbf{x}_i^s\right), \mathbf{y}_i^s\right)$$
$$\mathcal{E}(D, G) = -\mathbb{E}_{\mathbf{x}_i^*\sim\mathcal{D}_s} \log\left[D\left(\mathbf{f}_i^s, \mathbf{g}_i^s\right)\right] - \mathbb{E}_{\mathbf{x}_j'\sim\mathcal{D}_t} \log\left[1 - D\left(\mathbf{f}_j^t, \mathbf{g}_j^t\right)\right]$$

The we train the discriminator and feature extractor + classifier in a minimax framework just like GaNs (but this time with a gradient reversal layer).

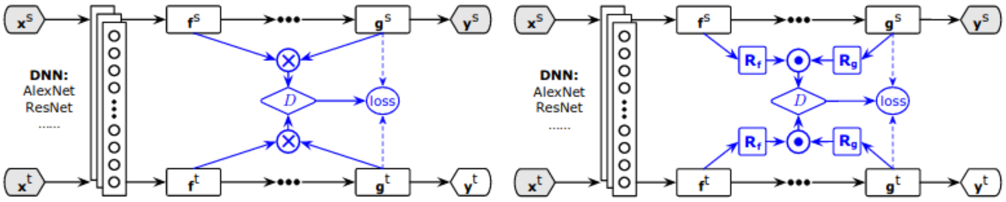
$$\min_G \mathcal{E}(G) - \lambda\mathcal{E}(D, G)$$
$$\min_D \mathcal{E}(D, G)$$

But how do we condition the features on the predicted labels? The authors say that simple concatenation is not rich enough, and propose to ways, either tensorial product $T_\otimes$ or, in case we have very large dimensions (the output of the feature extractor for ResNet for examples can be 4096), we apply random projection before the tensorial product.

$$T_\odot(\mathbf{f}, \mathbf{g}) = \frac{1}{\sqrt{d}}\left(\mathbf{R_f f}\right) \odot \left(\mathbf{R_g g}\right)$$

SO the conditioning is done as follows, and illustrated in the Figure bellow.

$$T(\mathbf{h}) = \begin{cases} T_{\otimes}(\mathbf{f}, \mathbf{g}) & \text{if } d_f \times d_g << 4096 \\ T_{\odot}(\mathbf{f}, \mathbf{g}) & \text{otherwise} \end{cases}$$



(a) Multilinear Conditioning

(b) Randomized Multilinear Conditioning

Figure 1: Architectures of Conditional Domain Adversarial Networks (**CDAN**) for domain adaptation, where domain-specific feature representation **f** and classifier prediction **g** embody the cross-domain gap to be reduced jointly by the conditional domain discriminator D. (a) Multilinear (M) Conditioning, applicable to lower-dimensional scenario, where D is conditioned on classifier prediction **g** via multilinear map $\mathbf{f} \otimes \mathbf{g}$; (b) Randomized Multilinear (RM) Conditioning, fit to higher-dimensional scenario, where D is conditioned on classifier prediction **g** via randomized multilinear map $\frac{1}{\sqrt{d}}(\mathbf{R_f f}) \odot (\mathbf{R_g g})$. Entropy Conditioning (dashed line) leads to **CDAN+E** that prioritizes D on easy-to-transfer examples.

Results

Table 2: Accuracy (%) on ImageCLEF-DA for unsupervised domain adaptation (AlexNet and ResNet)

Method	I → P	P → I	I → C	C → I	C → P	P → C	Avg
AlexNet [27]	66.2±0.2	70.0±0.2	84.3±0.2	71.3±0.4	59.3±0.5	84.5±0.3	73.9
DAN [29]	67.3±0.2	80.5±0.3	87.7±0.3	76.0±0.3	61.6±0.3	88.4±0.2	76.9
DANN [13]	66.5±0.6	81.8±0.3	89.0±0.4	79.8±0.6	63.5±0.5	88.7±0.3	78.2
JAN [30]	67.2±0.5	82.8±0.4	91.3±0.5	80.0±0.5	63.5±0.4	91.0±0.4	79.3
CDAN	**67.7**±0.3	83.3±0.1	91.8±0.2	**81.5**±0.2	63.0±0.2	91.5±0.3	79.8
CDAN+E	67.0±0.4	**84.8**±0.2	**92.4**±0.3	81.3±0.3	**64.7**±0.3	**91.6**±0.4	**80.3**
ResNet-50 [20]	74.8±0.3	83.9±0.1	91.5±0.3	78.0±0.2	65.5±0.3	91.2±0.3	80.7
DAN [29]	74.5±0.4	82.2±0.2	92.8±0.2	86.3±0.4	69.2±0.4	89.8±0.4	82.5
DANN [13]	75.0±0.6	86.0±0.3	96.2±0.4	87.0±0.5	74.3±0.5	91.5±0.6	85.0
JAN [30]	76.8±0.4	88.0±0.2	94.7±0.2	89.5±0.3	74.2±0.3	91.7±0.3	85.8
CDAN	76.7±0.3	90.6±0.3	97.0±0.4	90.5±0.4	**74.5**±0.3	93.5±0.4	87.1
CDAN+E	**77.7**±0.3	**90.7**±0.2	**97.7**±0.3	**91.3**±0.3	74.2±0.2	**94.3**±0.3	**87.7**

Table 3: Accuracy (%) on Office-Home for unsupervised domain adaptation (AlexNet and ResNet)

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
AlexNet [27]	26.4	32.6	41.3	22.1	41.7	42.1	20.5	20.3	51.1	31.0	27.9	54.9	34.3
DAN [29]	31.7	43.2	55.1	33.8	48.6	50.8	30.1	35.1	57.7	44.6	39.3	63.7	44.5
DANN [13]	36.4	45.2	54.7	35.2	51.8	55.1	31.6	39.7	59.3	45.7	46.4	65.9	47.3
JAN [30]	35.5	46.1	57.7	36.4	53.3	54.5	33.4	40.3	60.1	45.9	47.4	67.9	48.2
CDAN	36.2	47.3	58.6	37.3	54.4	**58.3**	33.2	**43.9**	62.1	48.2	48.1	70.7	49.9
CDAN+E	**38.1**	**50.3**	**60.3**	**39.7**	**56.4**	57.8	**35.5**	43.1	**63.2**	**48.4**	**48.5**	**71.1**	**51.0**
ResNet-50 [20]	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DAN [29]	43.6	57.0	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3
DANN [13]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
JAN [30]	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
CDAN	49.0	69.3	74.5	54.4	66.0	68.4	55.6	48.3	75.9	68.4	55.4	80.5	63.8
CDAN+E	**50.7**	**70.6**	**76.0**	**57.6**	**70.0**	**70.0**	**57.4**	**50.9**	**77.3**	**70.9**	**56.7**	**81.6**	**65.8**

Table 4: Accuracy (%) on Digits and VisDA-2017 for unsupervised domain adaptation (ResNet-50)

Method	M → U	U → M	S → M	Avg	Method	Synthetic → Real
UNIT [28]	**96.0**	93.6	**90.5**	93.4	JAN [30]	61.6
CyCADA [22]	95.6	96.5	90.4	94.2	GTA [43]	69.5
CDAN	93.9	96.9	88.5	93.1	**CDAN**	66.8
CDAN+E	95.6	**98.0**	89.2	**94.3**	**CDAN+E**	**70.0**

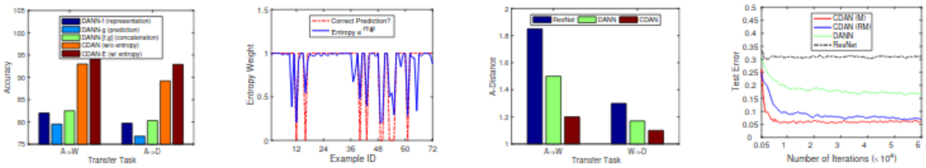
Table 1: Accuracy (%) on Office-31 for unsupervised domain adaptation (AlexNet and ResNet)

Method	A → W	D → W	W → D	A → D	D → A	W → A	Avg
AlexNet [27]	61.6±0.5	95.4±0.3	99.0±0.2	63.8±0.5	51.1±0.6	49.8±0.4	70.1
DAN [29]	68.5±0.5	96.0±0.3	99.0±0.3	67.0±0.4	54.0±0.5	53.1±0.5	72.9
RTN [31]	73.3±0.3	96.8±0.2	99.6±0.1	71.0±0.2	50.5±0.3	51.0±0.1	73.7
DANN [13]	73.0±0.5	96.4±0.3	99.2±0.3	72.3±0.3	53.4±0.4	51.2±0.5	74.3
ADDA [51]	73.5±0.6	96.2±0.4	98.8±0.4	71.6±0.4	54.6±0.5	53.5±0.6	74.7
JAN [30]	74.9±0.3	96.6±0.2	99.5±0.2	71.8±0.2	**58.3**±0.3	55.0±0.4	76.0
CDAN	77.9±0.3	96.9±0.2	**100.0**±.0	75.1±0.2	54.5±0.3	**57.5**±0.4	77.0
CDAN+E	**78.3**±0.2	**97.2**±0.1	**100.0**±.0	**76.3**±0.1	57.3±0.2	57.3±0.3	**77.7**
ResNet-50 [20]	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 [29]	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 [31]	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 [13]	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 [51]	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 [30]	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 [43]	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
CDAN	93.1±0.2	98.2±0.2	**100.0**±.0	89.8±0.3	70.1±0.4	68.0±0.4	86.6
CDAN+E	**94.1**±0.1	**98.6**±0.1	**100.0**±.0	**92.9**±0.2	71.0±0.3	69.3±0.3	**87.7**

Table 5: Accuracy (%) of CDAN variants on Office-31 for unsupervised domain adaptation (ResNet)

Method	A → W	D → W	W → D	A → D	D → A	W → A	Avg
CDAN+E (w/ gaussian sampling)	93.0±0.2	98.4±0.2	**100.0**±.0	89.2±0.3	70.2±0.4	67.4±0.4	86.4
CDAN+E (w/ uniform sampling)	94.0±0.2	98.4±0.2	**100.0**±.0	89.8±0.3	70.1±0.4	**69.4**±0.4	87.0
CDAN+E (w/o random sampling)	**94.1**±0.1	**98.6**±0.1	**100.0**±.0	**92.9**±0.2	**71.0**±0.3	69.3±0.3	**87.7**

domain discriminator plugged in feature layer **f** and classifier layer **g**. Figure 2(a) shows accuracies on **A** → **W** and **A** → **D** based on ResNet-50. The concatenation strategy is not successful, as it cannot capture the cross-covariance between features and classes, which are crucial to domain adaptation [10]. Figure 2(b) shows that the entropy weight $e^{-H(\mathbf{g})}$ corresponds well with the prediction correctness: entropy weight ≈ 1 if the prediction is correct, and much smaller than 1 when prediction is incorrect (uncertain). This reveals the power of the entropy conditioning to guarantee example transferability.



(a) Multilinear (b) Entropy (c) Discrepancy (d) Convergence

Figure 2: Analysis of conditioning strategies, distribution discrepancy, and convergence.