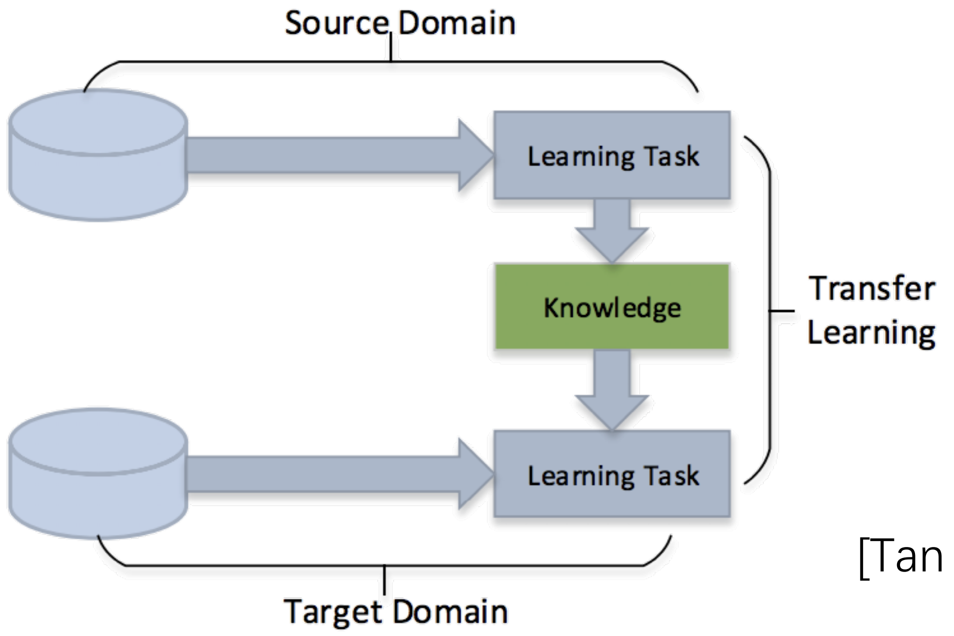
Part III: Domain Adaptation

Yuhong Guo

Introduction

Domain Adaptation/Transfer Learning

- ***Definition*** [Pan et al., IJCAI13]:
  Ability of a system to recognize and apply knowledge and skills learned in previous domains/tasks to novel domains/tasks
- .



[Tan et al, CVPR18]

Fig. 1. Learning process of transfer learning.

S. Pan, Q. Yang and W. Fan. Tutorial: Transfer Learning with Applications, IJCAI 2013.
Tan, Chuanqi, et al. "A survey on deep transfer learning." International Conference on Artificial Neural Networks. Springer, Cham, 2018.

Why Domain Adaptation

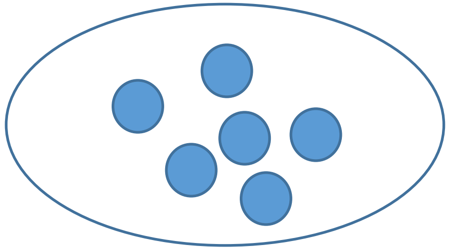
- Successful Application of ML in industry depends on learning from large amount of labeled data
  - ➢ Expensive, time consuming to collect labels
  - ➢ Difficult or dangerous to collect data in certain scenarios, e.g, auto driving
- Domain Adaptation/Transfer Learning provides essential ability of
  - ✓ Reusing existing labeled resources
  - ✓ Adapting to changing environment
  - ✓ Learning from simulations

Transfer Learning vs Traditional ML

Traditional ML

(Semi-)Supervised Learning

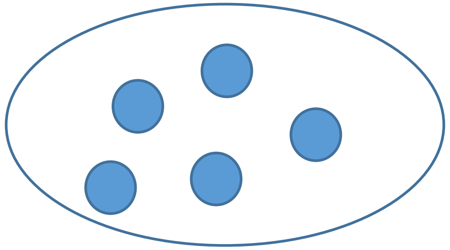
Training domain/task A



=

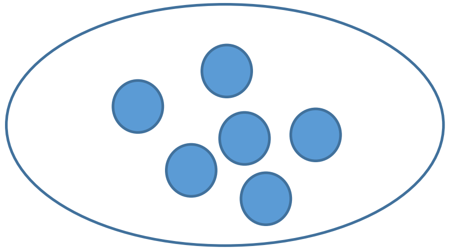
- same feature space
- same feature distribution
- same label space

Test domain/task B



Transfer Learning/Domain Adaptation

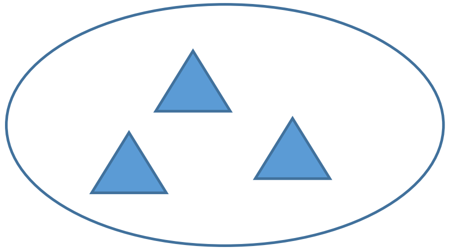
Training domain/task A



≠

- different feature spaces
- different feature distributions
- different label spaces

Test domain/task B



Motivation Examples

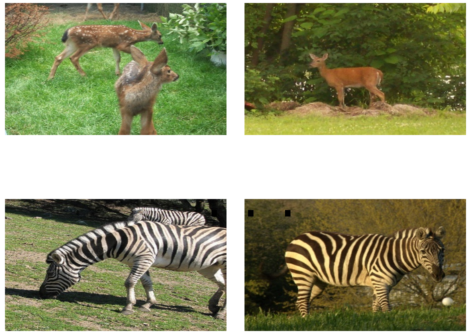
Different feature distributions

Source domain

Target domain

[Bebru et al, ICCV 17]

Different label spaces



C_{train}

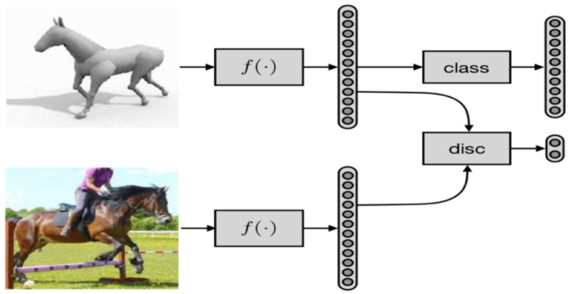
C_{test}

[Ravi et al, ICLR 17]

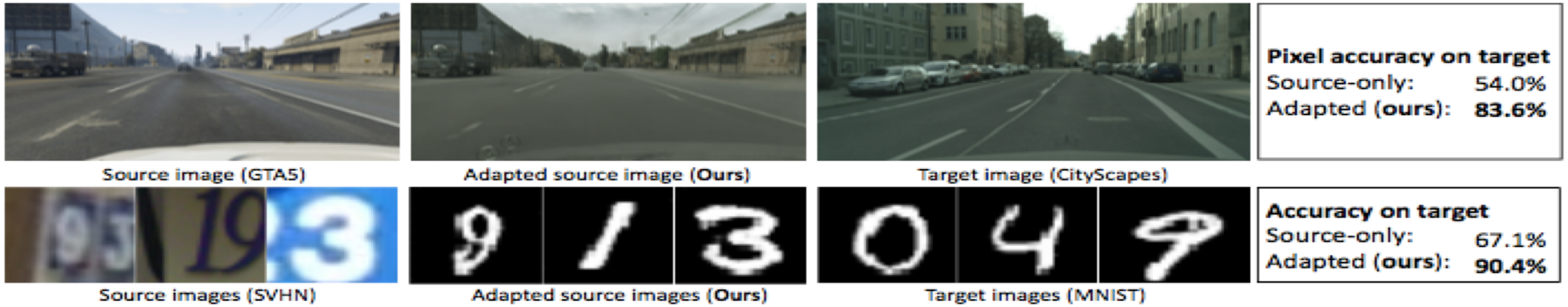
Applications in Computer Vision

Adapting to New Domains

- Reuse existing datasets, hence the annotation information
  - Object Recognition
  - Object Detection
  - Person Re-Identification
  - Image Segmentation
  - Image Classification … …



L. Wei et al, CVPR18]



[J. Hoffman et al, ICML18]

Learning from Simulations

http://ruder.io/transfer-learning/index.html

- Gathering data and training model are either too expensive, time-consuming, or too dangerous
- **Solution:** create data, learning from simulations
  - Auto driving

    OpenAI's Universe will potentially allow us to train a self-driving car using GTA 5 or other video games.

Udacity's self-driving car simulator (source: TechCrunch)

  - Robotic

    Training models on real robotics is too slow and expensive

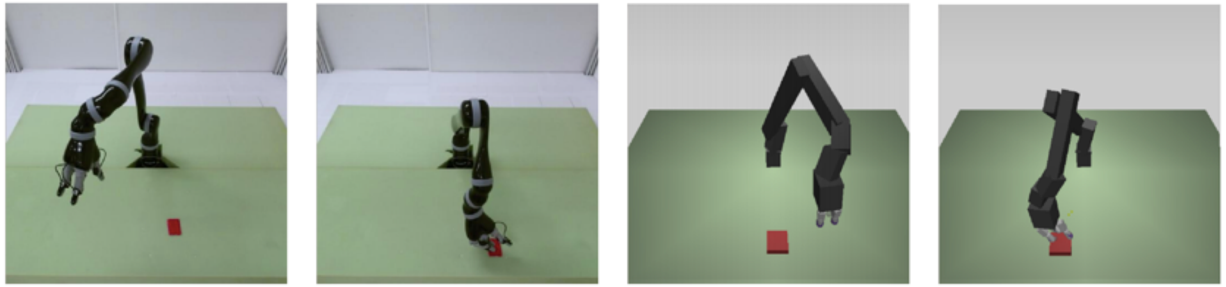


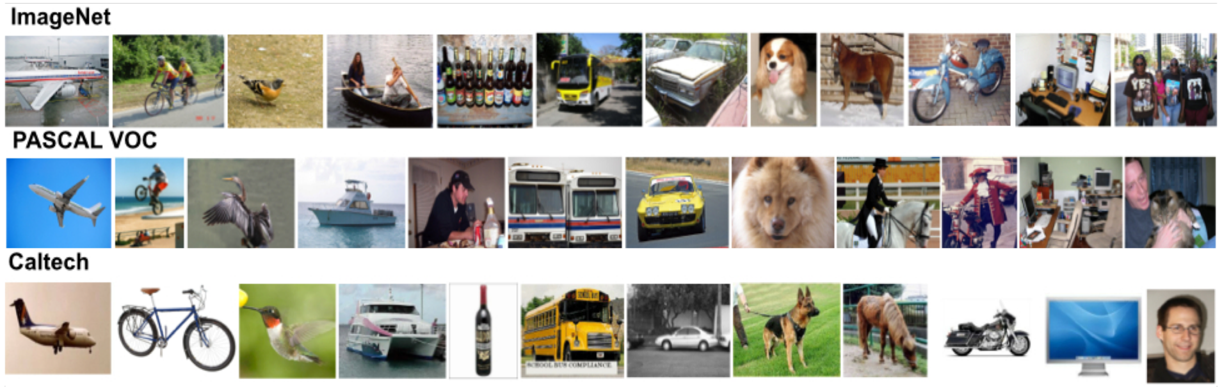
Figure 8: Robot and simulation images (Rusu et al., 2016)

Common Datasets

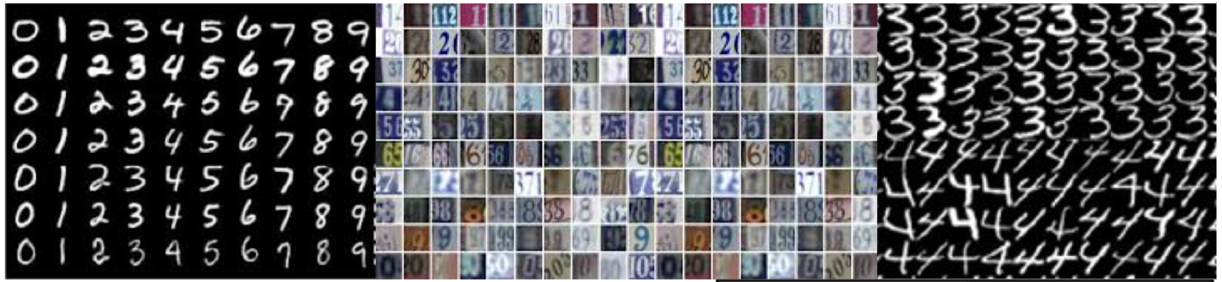
- Object recognition:

  Office-31:
  - Amazon (A)
  - Webcam (W)
  - DSLR (D)

  ImageCLEF-DA:
  - ImageNet ILSVRC 2012 (I)
  - Pascal VOC 2012 (P)
  - Caltech-256 (C)



- Digits: MNIST, SVHN, USPS

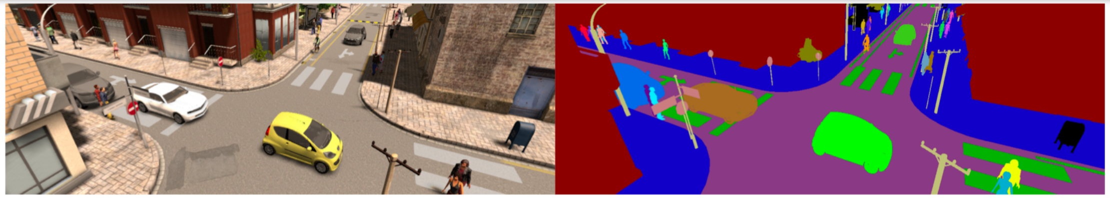


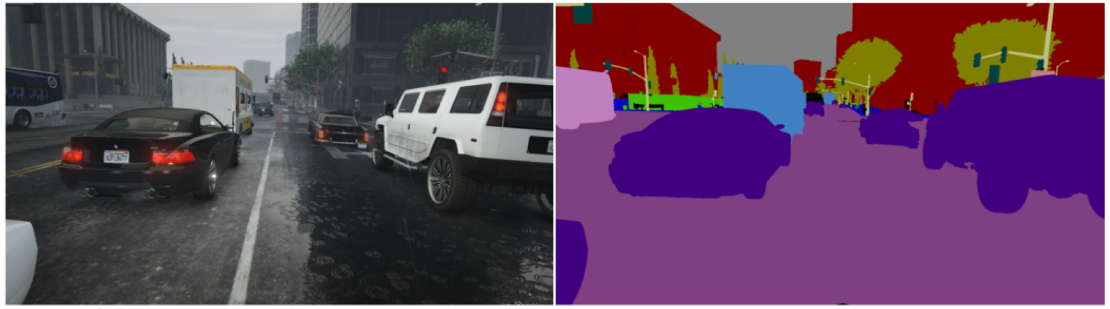
- Syn2Real dataset – a new dataset for object recognition

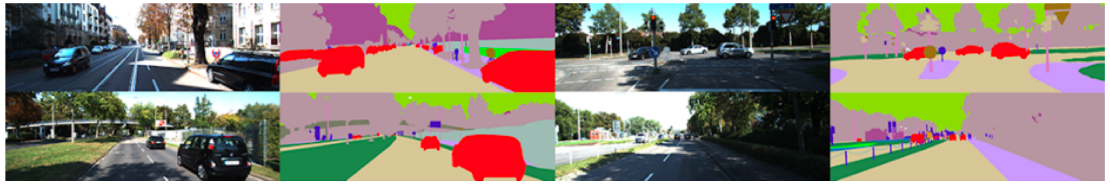
[Peng et al, 2018]

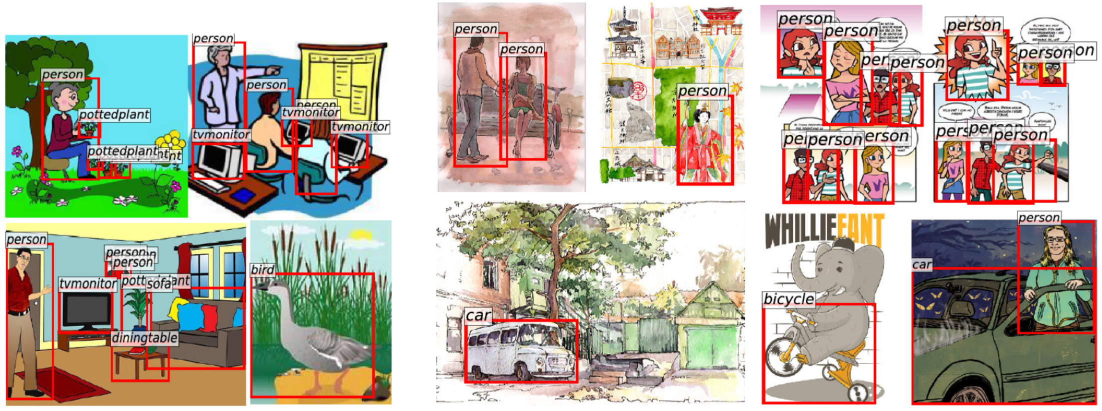
Common Datasets

- Semantic Segmentation/object detection:
  - SYNTHIA/GTA5/SIM10K
  - Cityscapes/Foggy Cityscapes/KITTI
  - Watercolor datasets constructed using Amazon Mechanical Turk:
    - Clipart1k, Watercolor2k, Comic2k
  - Visual domain adaptation challenge dataset VisDA-2017
  - CVPR19 domain adaptation challenge: BDD100K, D^2-City









(a) Clipart1k (b) Watercolor2k (c) Comic2k

Domain Adaptation Methods

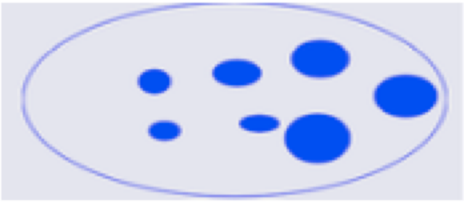
Categories of DA Methods

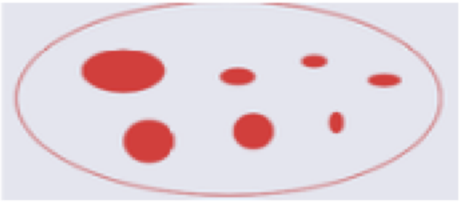
Three main classes:

- Reweighting/Instance-based Methods
  - ✓Reweight source labeled instances to match cross-domain feature distributions
- Feature-based/Representation Learning Methods
  - ✓Seek a good representation of data to minimize the gap between the source and target distributions (via projection, deep learning, etc)
- Parameter/Model- based Methods
  - ✓Transfer models/parameters between source and target domains

Start with Instance Reweighting

- Context
  - Domains share the same input space
  - Exist distribution shift across source and target domains, caused by sampling bias / shift between marginals $P_S(\mathbf{x}) \neq P_T(\mathbf{x})$





5 1 0.2

- Idea
  - **Reweight** source labeled instances to reduce the discrepancy between the source and target domains $P_S(\phi(\mathbf{x})) \approx P_T(\phi(\mathbf{x}))$

Simple Math Analysis

- h() – prediction function, x --- input , y – output
- Expected risk in target domain:

$$
\begin{aligned}
R_T(h) &= \mathbf{E}_{(\mathbf{x},y)\sim P_T} I[h(\mathbf{x}) \neq y] \\
&= \mathbf{E}_{(\mathbf{x},y)\sim P_T} \frac{P_S(\mathbf{x},y)}{P_S(\mathbf{x},y)} I[h(\mathbf{x}) \neq y] \\
&= \mathbf{E}_{(\mathbf{x},y)} P_T(\mathbf{x},y) \frac{P_S(\mathbf{x},y)}{P_S(\mathbf{x},y)} I[h(\mathbf{x}) \neq y] \\
&= \mathbf{E}_{(\mathbf{x},y)\sim P_S} \frac{P_T(\mathbf{x},y)}{P_S(\mathbf{x},y)} I[h(\mathbf{x}) \neq y]
\end{aligned}
$$

Covariate Shift

[Shimodaira,00]

- Assume shared conditional distribution $P_S(y|\mathbf{x}) = P_T(y|\mathbf{x})$

$$
\begin{aligned}
R_T(h) &= \mathbf{E}_{(\mathbf{x},y)\sim P_S} \frac{P_T(\mathbf{x},y)}{P_S(\mathbf{x},y)} I[h(\mathbf{x}) \neq y] \\
&= \mathbf{E}_{(\mathbf{x},y)\sim P_S} \frac{P_T(\mathbf{x})P_T(y|\mathbf{x})}{P_S(\mathbf{x})P_S(y|\mathbf{x})} I[h(\mathbf{x}) \neq y] \\
&= \mathbf{E}_{(\mathbf{x},y)\sim P_S} \frac{P_T(\mathbf{x})}{P_S(\mathbf{x})} I[h(\mathbf{x}) \neq y] \\
&= \mathbf{E}_{\mathbf{x}\sim D_S} \frac{P_T(\mathbf{x})}{P_S(\mathbf{x})} \mathbf{E}_{y\sim P_S(y|\mathbf{x})} I[h(\mathbf{x}) \neq y]
\end{aligned}
$$

Training in source domain

- To minimize target risk, source instance can be reweighted:

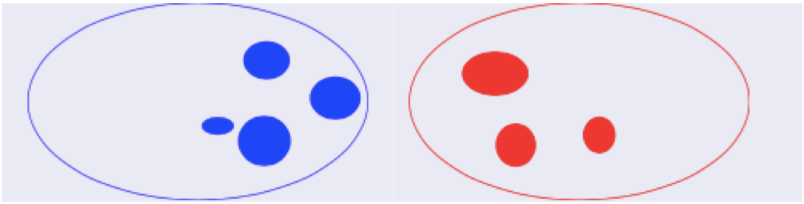
$$
\omega(\mathbf{x}) = \frac{P_T(\mathbf{x})}{P_S(\mathbf{x})}
$$

Assumptions

- Assume shared conditional distribution $P_S(y|\mathbf{x}) = P_T(y|\mathbf{x})$
- In addition, note $\omega(\mathbf{x}) = \frac{P_T(\mathbf{x})}{P_S(\mathbf{x})}$
  - **If P_S(x) = P_T(x) , ω(x) = 1 , need no adaptation**
  - **If P_S(x) ≠ P_T(x) , ω(x) ≠ 1 ,** adaptation across domain

Matching cross-domain Marginal distributions

- **Assumption of support:**
  - However, **problematic** if $\exists x$, P_T(x) >0, but P_S(x) =0



  - **shared support in the source domain**: P_S(x) =0 **iff** P_T(x) = 0

Weight Estimation

- **Density ratio estimation** [Sugiyama *et al.*, NIPS-07]
  - Estimate the density $\boldsymbol{P}(\mathbf{x})$ with some standard models, e.g., mixture Gaussian
  - Then compute the weight $\boldsymbol{\omega}(\mathbf{x})$
- **Direct weight estimation**
  - Learning weights with a **binary domain discrimination function**

[Bickel et al., ICML07]

$$\boldsymbol{\omega}(\mathbf{x}) = \boldsymbol{P_T}(\mathbf{x}) \;/\; \boldsymbol{P_S}(\mathbf{x}) \propto \boldsymbol{P(s=0|x)/P(s=1|x)}$$

$\boldsymbol{P(s=0|x)}$: **prob. of instance x belonging to target domain**
$\boldsymbol{P(s=1|x)}$: **prob. of instance x belonging to the source domain**

Learning Weights Directly: MMD

- **Maximum Mean Discrepancy (MMD)** [Gretton et al. 2012]
  - ➢A test statistic for measuring the difference of two distributions p, q.
  - ➢MMD defined in Reproducing Kernel Hilbert Spaces [Gretton et al, 2012]:

Lemma 4 Assume existence of *the mean embeddings* μ_p, μ_q

$$\mathrm{MMD}^2[\mathcal{F},p,q] = \|\mu_p - \mu_q\|^2_{\mathcal{H}}.$$

- **Lemma 5** Let $\mathcal{F}$ be a unit ball in a universal RKHS $\mathcal{H}$, defined on the compact metric space $\mathcal{X}$, with associated continuous kernel k(·, ·). Then *MMD [F, p, q] = 0 if and only if p = q.*

Learning Weights Directly: MMD

- **MMD for domain adaptation**

  - A widely used first order **metric for matching the cross-domain distributions**

$$\mathrm{MMD}^2(s,t) = \sup_{\|\phi\|_{\mathcal{H}} \le 1} \left\| E_{\mathbf{x}^s \sim p_s}[\phi(\mathbf{x}^s)] - E_{\mathbf{x}^t \sim p_t}[\phi(\mathbf{x}^t)] \right\|_{\mathcal{H}}^2$$

  - Learn weights by minimizing the empirical MMD

$$P_T(\mathrm{x}) \sim \omega(\mathrm{x}) P_S(\mathrm{x})$$

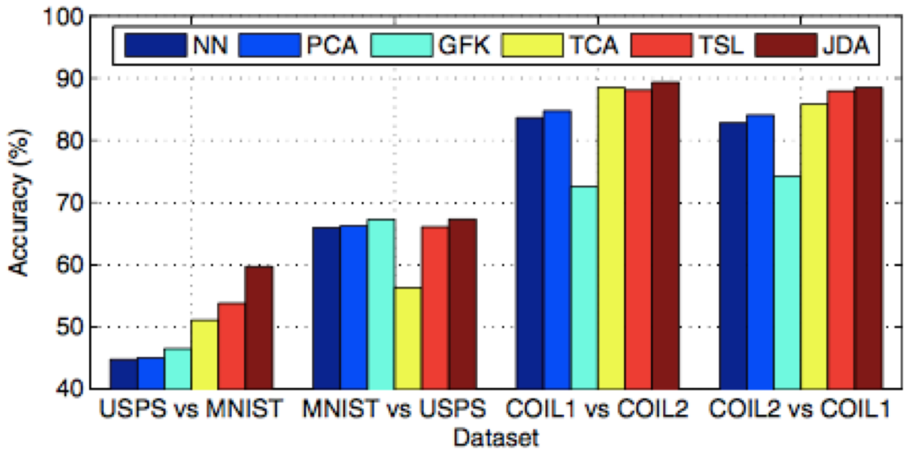
Extend to Representation Learning

- Extend MMD to learn representation function $\emptyset(x)$
  - ➢E.g., simple **feature transformation** [Long et al. CVPR13]

Conditional distribution adaptation (pseudo labels)

$$\left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{A}^{\mathrm{T}} \mathbf{x}_i - \frac{1}{n_t} \sum_{j=n_s+1}^{n_s+n_t} \mathbf{A}^{\mathrm{T}} \mathbf{x}_j \right\|^2 = \mathrm{tr}\left(\mathbf{A}^{\mathrm{T}} \mathbf{X} \mathbf{M}_0 \mathbf{X}^{\mathrm{T}} \mathbf{A} \right)$$

$$\left\| \frac{1}{n_s^{(c)}} \sum_{\mathbf{x}_i \in \mathcal{D}_s^{(c)}} \mathbf{A}^{\mathrm{T}} \mathbf{x}_i - \frac{1}{n_t^{(c)}} \sum_{\mathbf{x}_j \in \mathcal{D}_t^{(c)}} \mathbf{A}^{\mathrm{T}} \mathbf{x}_j \right\|^2 = \mathrm{tr}\left(\mathbf{A}^{\mathrm{T}} \mathbf{X} \mathbf{M}_c \mathbf{X}^{\mathrm{T}} \mathbf{A} \right)$$



Long et al. " Transfer feature learning with joint distribution adaptation ", CVPR 13

Recent Feature-based Methods

- Representation learning methods present larger capacity in bridging domain discrepancy
- Widely applied in transfer learning for computer vision tasks
- Recent development of representation learning based domain adaptation
  - **Exploit adversarial loss**
  - **Use generative models**
  - **Exploit pseudo-labels**

Adversarial Loss-based Adaptation Framework

- **Main idea:**
  - Use adversarial loss to reduce cross-domain discrepancy

$$\min_G \max_D \; L_{adv}(G, D) = \mathbb{E}_{x\sim D_S} \log D(G(x)) + \mathbb{E}_{x\sim D_T} \log(1 - D\big(G(x)\big))$$

    - $G(x)$ is a feature extractor; e.g., a deep network; or $(G_S,\ G_T)$
    - D(.) is the domain discrimination function

  - Theorem 1 of [Goodfellow et al, 2014] suggests:

The global minimum is achieved if and only if $\mathrm{p}_S\,(G(x)) = \mathrm{p}_T(G(x))$

Goodfellow et al. " **Generative Adversarial Networks** ", 2014

Theoretical Connection

[Kifer et al., VLDB04]

- A-distance, measure of distance between probability distribution

$$d_{\mathcal{A}}(\mathcal{D}, \mathcal{D}') = 2 \sup_{A \in \mathcal{A}} |\Pr_{\mathcal{D}}[A] - \Pr_{\mathcal{D}'}[A]|$$

- Bound on target domain error
  - .Theorem 2: [Ben-David et al., NIPS06]

$\lambda = \min_h [\epsilon_s(h) + \epsilon_t(h)]$;

$$\epsilon_T(h) \le \hat{\epsilon}_S(h) + \frac{4}{m}\sqrt{\left(d \log \frac{2em}{d} + \log \frac{4}{\delta}\right)} + \lambda + d_{\mathcal{H}}(\tilde{\mathcal{U}}_S, \tilde{\mathcal{U}}_T) + 4\sqrt{\frac{d \log(2m') + \log(\frac{4}{\delta})}{m'}}$$

Sample A-distance between domains

  - Computing A-distance on real data

$$d_{\mathcal{A}}(\tilde{\mathcal{U}}_S, \tilde{\mathcal{U}}_T) = 2\left(1 - 2 \min_{h' \in \mathcal{H}} \mathrm{err}(h')\right)$$

Binary classification error of discriminating points sampled from two domains

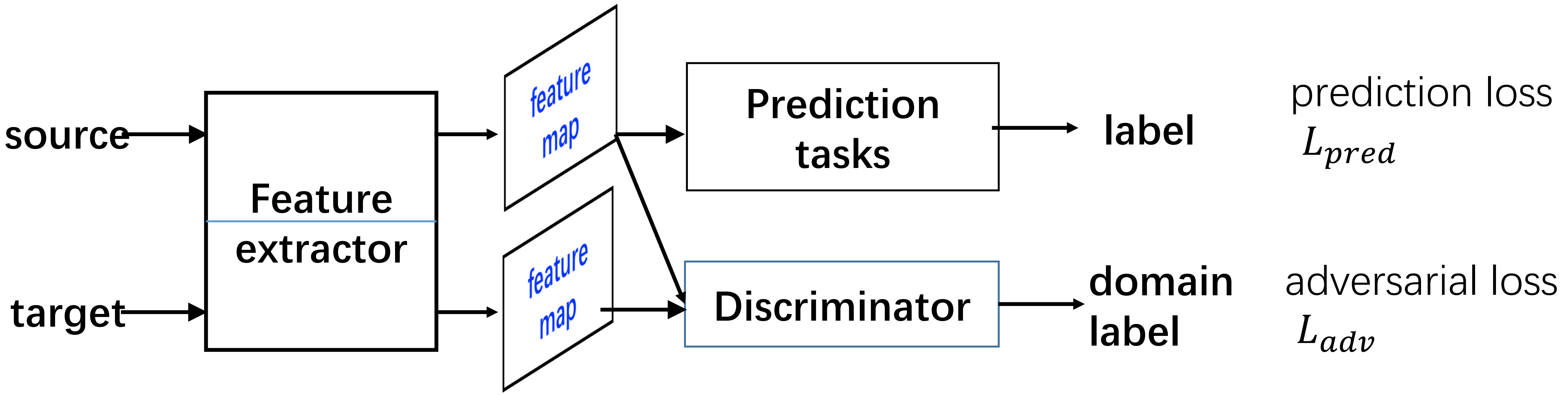
Kifer et al. Detecting change in data streams. In *Very Large Databases (VLDB)*, 2004.

Ben-David et al. "Analysis of Representations for Domain Adaptation", NIPS 06

Adversarial Loss-based Adaptation Framework

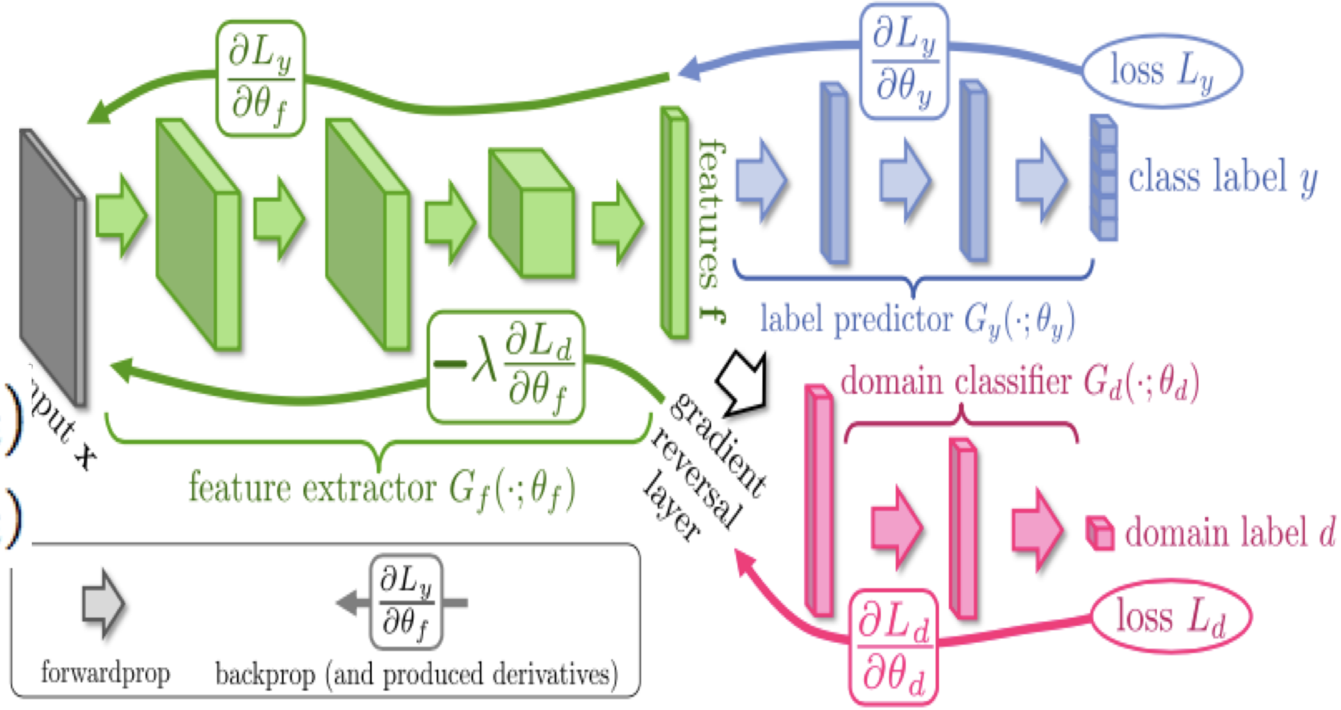
- **Main idea:**
  - Use adversarial regularized prediction loss for training

$$\min_{G,F} \max_D \quad L = L_{pred}(G,F) + \lambda L_{adv}(G,D)$$



Domain Adversarial Neural Network (DANN)

- **DANN**: Adversarial is implemented via GRL (gradient reverse layer)



$$\mathcal{L}_y^i(\theta_f, \theta_y) = \mathcal{L}_y(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i)$$
$$\mathcal{L}_d^i(\theta_f, \theta_d) = \mathcal{L}_d(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), d_i)$$

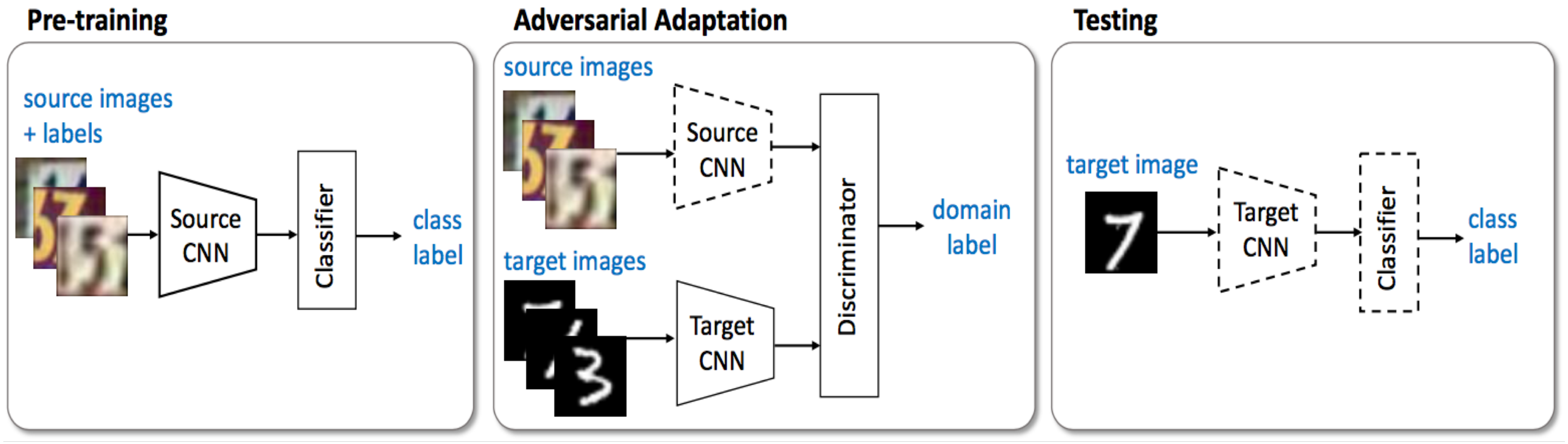
$$E(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_y^i(\theta_f, \theta_y) - \lambda \left(\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_d^i(\theta_f, \theta_d) + \frac{1}{n'}\sum_{i=n+1}^{N} \mathcal{L}_d^i(\theta_f, \theta_d) \right)$$

Ganin et al. "Domain-adversarial training of neural networks", *JMLR 2016*

Ganin, Y. and Lempitsky, V. "Unsupervised domain adaptation by backpropagation". ICML2015

Model Sharing and Adversarial Adaptation

- *Adversarial Discriminative Domain Adaptation* (**ADDA**)

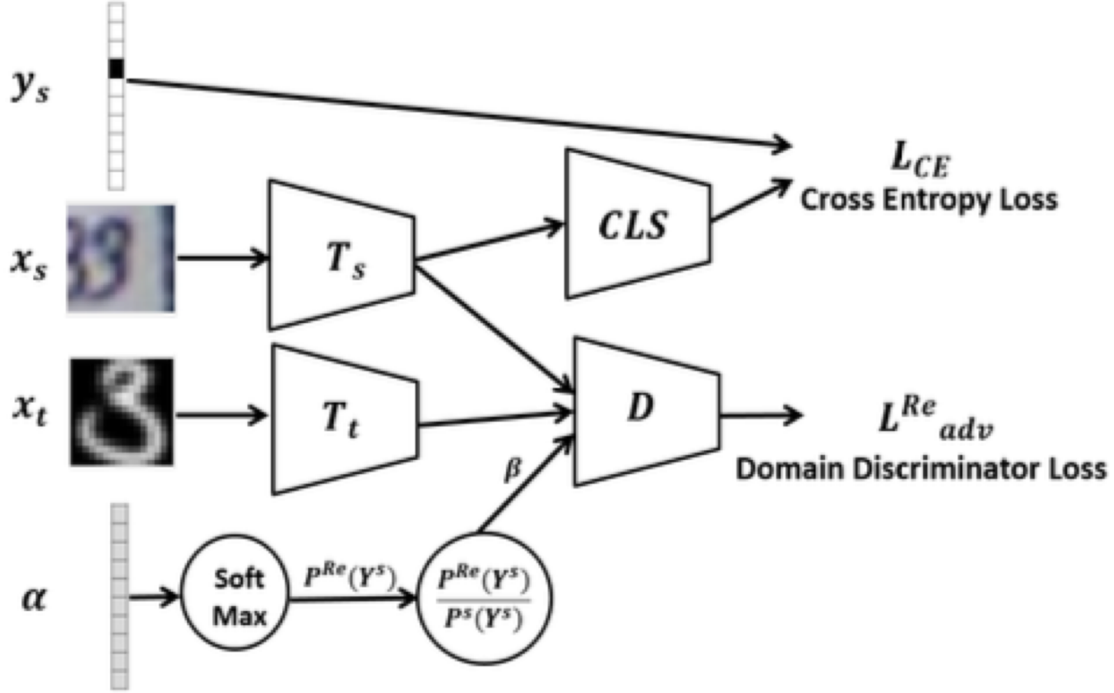


source CNN is trained without sacrificing any discriminativity

Tzeng et al. "Adversarial Discriminative Domain Adaptation" ,CVPR2017

Reweighted Adversarial Adaptation [Chen et al, CVPR 18]

- Re-weight source domain label distribution to help reduce domain discrepancy and adapt classifier
- Reweighted adversarial loss (**RAAN**)



$$\min_{\mathcal{T}_t} \max_{\mathcal{D},\boldsymbol{\beta}} \mathcal{L}_{adv}^{Re}, \ where$$

$$\mathcal{L}_{adv}^{Re} = \mathop{\mathbb{E}}_{(\boldsymbol{x}^s, y^s) \sim P^s(\boldsymbol{X}^s, \boldsymbol{Y}^s)} \boldsymbol{\beta}(y^s) \mathcal{D}(\mathcal{T}_s(\boldsymbol{x}^s)) - \mathop{\mathbb{E}}_{\boldsymbol{x}^t \sim P^t(\boldsymbol{X}^t)} \mathcal{D}(\mathcal{T}_t(\boldsymbol{x}^t))$$

$$s.t. \| \nabla_{\mathcal{T}_t(\boldsymbol{x}^t)} \mathcal{D}(\mathcal{T}_t(\boldsymbol{x}^t)) \|_2 \leq 1,$$

$$\| \nabla_{\mathcal{T}_s(\boldsymbol{x}^s)} \mathcal{D}(\mathcal{T}_s(\boldsymbol{x}^s)) \|_2 \leq 1. \qquad (14)$$

Chen, et al. " Re-weighted Adversarial Adaptation Network for Unsupervised Domain Adaptation ", CVPR 18

+/-: with/without reweighting

Alternative Adversarial Terms

[Saito et al, CVPR 18]

- Maximum Classifier Discrepancy **(MCD):**
  - Use **multiple (two) classifiers** F1, F2:
  - Exploit **prediction disagreement in target domain** as an **adversarial term**

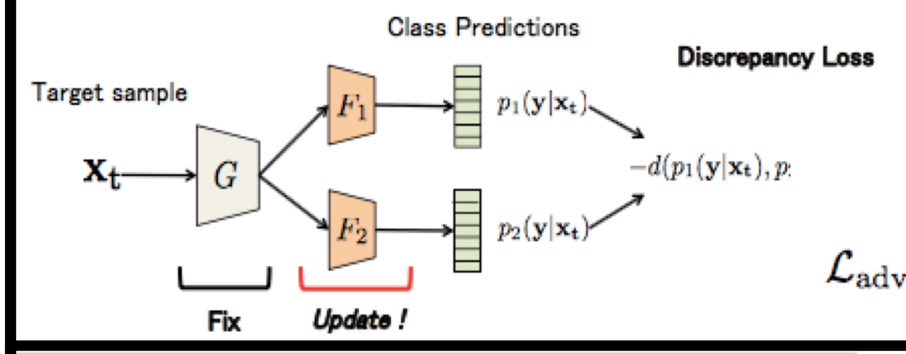
Step A

Train both classifiers and generator to classify the source samples correctly

$$\min_{G,F_1,F_2} \mathcal{L}(X_s, Y_s).$$

$$\mathcal{L}(X_s, Y_s) = -\mathbb{E}_{(\mathbf{x}_s, y_s)\sim(X_s,Y_s)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log p(\mathbf{y}|\mathbf{x}_s)$$

Step B : *Maximize* **discrepancy on target (Fix G)**

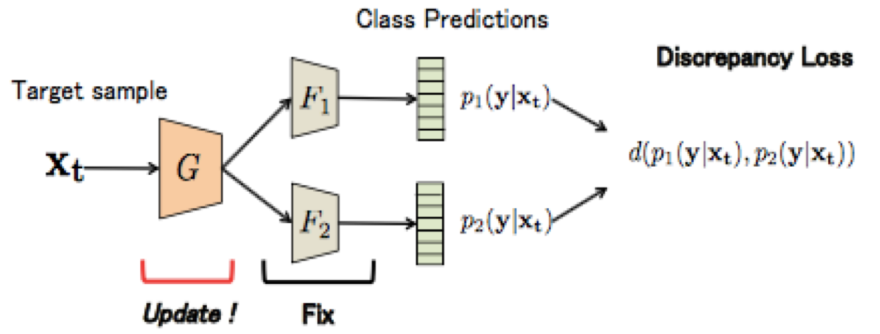


$$d(p_1, p_2) = \frac{1}{K}\sum_{k=1}^{K} |p_{1_k} - p_{2_k}|$$

$$\min_{F_1,F_2} \mathcal{L}(X_s, Y_s) - \mathcal{L}_{\text{adv}}(X_t).$$

$$\mathcal{L}_{\text{adv}}(X_t) = \mathbb{E}_{\mathbf{x_t}\sim X_t}[d(p_1(\mathbf{y}|\mathbf{x_t}), p_2(\mathbf{y}|\mathbf{x_t}))]$$

Step C : *Minimize* **discrepancy on target (Fix F_1 , F_2)**



$$\min_{G} \mathcal{L}_{\text{adv}}(X_t).$$

- Adversarial loss:

Target domain prediction discrepancy

K. Saito, et al. " Maximum Classifier Discrepancy for Unsupervised Domain Adaptation", CVPR 18

Conditional Adversarial Domain Adaptation

- Conditional Domain Adversarial Networks (**CDANs**) [NeurIPS 18]:
  - When feature distribution is multimodal under multi-class classification, **exploit classifier prediction** for the domain adversarial discriminator with multilinear map

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

$$\begin{aligned} &\min_{G} \mathbb{E}_{(\mathbf{x}_i^s, \mathbf{y}_i^s)\sim\mathcal{D}_s} L\left(G\left(\mathbf{x}_i^s\right), \mathbf{y}_i^s\right) \\ &\quad + \lambda\left(\mathbb{E}_{\mathbf{x}_i^s\sim\mathcal{D}_s} \log\left[D\left(T\left(\mathbf{h}_i^s\right)\right)\right] + \mathbb{E}_{\mathbf{x}_j^t\sim\mathcal{D}_t} \log\left[1 - D\left(T\left(\mathbf{h}_j^t\right)\right)\right]\right) \\ &\max_{D} \mathbb{E}_{\mathbf{x}_i^s\sim\mathcal{D}_s} \log\left[D\left(T\left(\mathbf{h}_i^s\right)\right)\right] + \mathbb{E}_{\mathbf{x}_j^t\sim\mathcal{D}_t} \log\left[1 - D\left(T\left(\mathbf{h}_j^t\right)\right)\right], \end{aligned}$$

Long et al. "Conditional Adversarial Domain Adaptation". NeurIPS 2018

DA Recognition Results

Office-Home (Classification Accuracy (%) on Office-Home with 40% Mixed Corruption)

Methods	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Average
ResNet-50	27.1	50.7	61.7	41.1	53.8	56.3	40.9	28.0	61.8	51.3	33.0	65.9	47.6
DANN(Ganin et.al. JMLR16)	32.9	50.6	60.1	38.6	49.2	50.6	39.9	32.6	60.4	50.5	38.4	67.4	47.6
ADDA(Tzeng et al.CVPR17)	32.6	52.0	60.6	42.6	53.5	54.3	43.0	31.6	63.1	52.7	37.7	67.5	49.3
TCL(Shu.et.al.AAAI19)	38.8	62.1	69.4	46.5	58.5	59.8	51.3	39.9	72.3	63.4	43.5	74.0	56.6

Digit

Methods	MNIST→USPS	USPS→MNIST	SVHN→MNIST	SYNSIG→GTSRB
Source only	75.2±1.6	57.1±1.7	60.1±1.1	
DANN (Ganin et.al. JMLR16)	77.1±1.8	73.0±0.2	73.85	88.65
ADDA (Tzeng et al.CVPR17)	89.4±0.2	90.1±0.8	76.0±1.8	
MCD_DA(Saito et al.CVPR18)	94.2±0.7	94.1±0.1	96.2±0.4	94.4±0.3
RANN(-)(Chen et al.CVPR18)	88.3	91.5	80.7	
RANN(+)(Chen et al.CVPR18)	89.0	92.1	89.2	
+/- : with/without reweighting				
CDAN(Long et al. NeurIPS18)	93.9	96.9	88.5	
CDAN+E(Long et al. NeurIPS18)	95.6	98.0	89.2	

VisDA-2017

Methods	Airplane	Bicycle	Bus	Car	Horse	Knife	motorcycle	Person	Plant	Skateboard	Train	Truck	Average
ResNet101	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
DANN(Ganin et.al. JMLR16)	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
MCD(n=4)(Saito et al.CVPR18)	87.0	60.9	83.7	64.0	88.9	79.6	84.7	76.9	88.6	40.3	83.0	25.8	71.9
CDAN (Long et al. NeurIPS18)	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.7

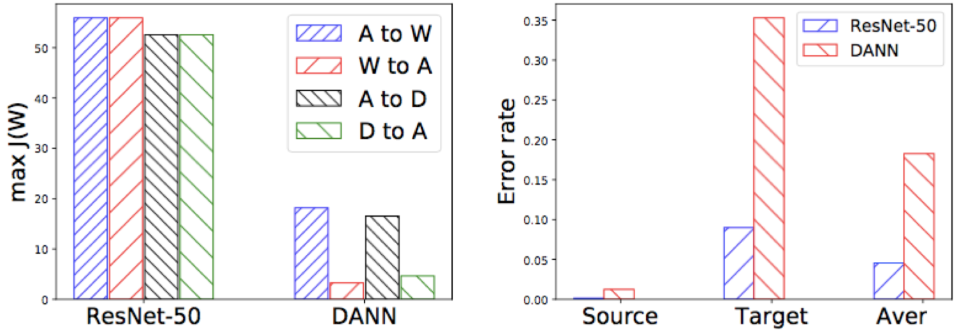
Office-31

Methods	A→W	D→W	W→D	A→D	D→A	W→A	Average
ResNet-50 (2016)	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
DANN(Ganin et al.JMLR16)	79.3	97.3	99.6	80.7	65.3	63.2	80.9
ADDA(Tzeng et al.CVPR17)	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
CDAN(Long et al. NeurIPS18)	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(Long et al. NeurIPS18)	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

Question Raised: Transferabiliy vs Discriminability

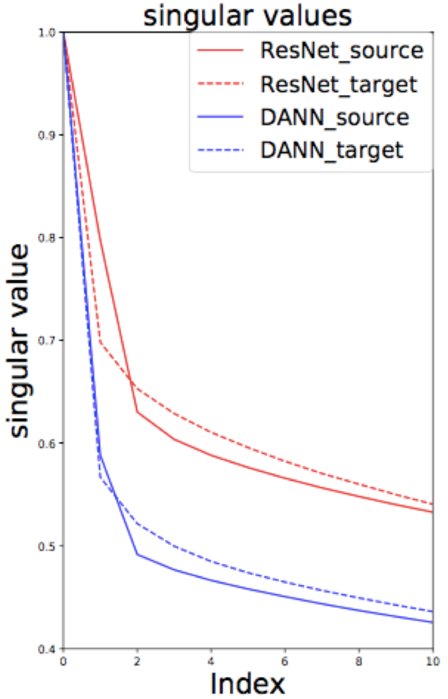
BSP+DANN[ICML19]:

- In DANN, the top eigenvectors of feature matrix dominate the transferability, at the cost of discriminability



(a) max $J(\mathbf{W})$ (b) Error Rate

Figure 1. Two experiments measuring discriminability of fea



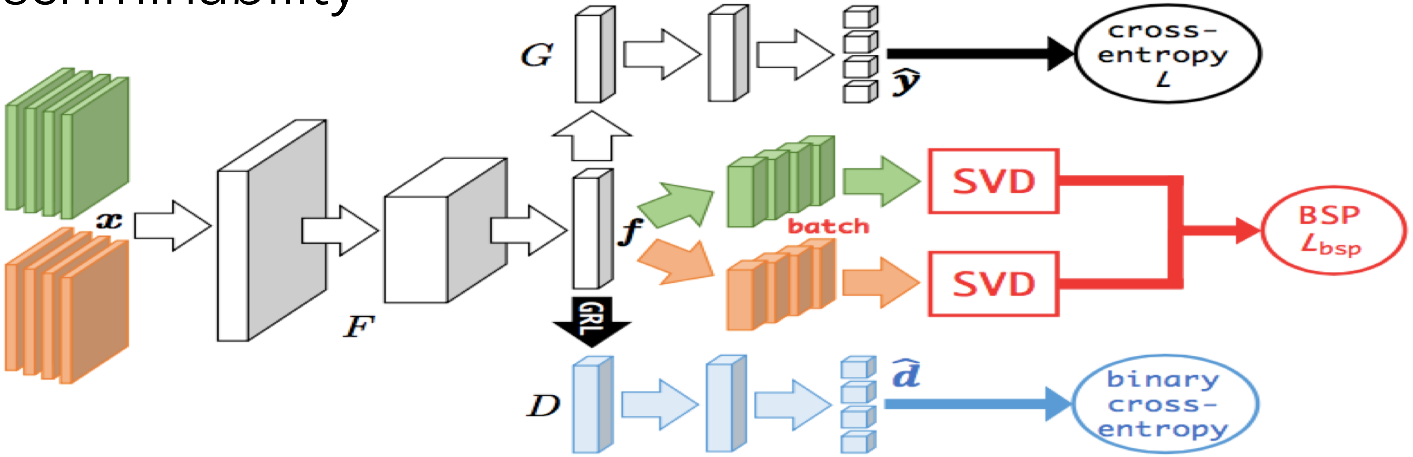


Figure 3. The architecture of **BSP+DANN** where BSP enhances discriminability while learning transferable features via domain adversarial network (DANN). BSP is a lightweight module readily pluggable into any deep domain adaptation networks, which is end-to-end trainable with the support of **differentiable SVD** in **PyTorch**. GRL denotes Gradient Reversal Layer widely used in adversarial domain adaptation.

Chen et al. "Transferability vs. Discriminability: Batch Spectral Penalization for Adversarial Domain Adaptation" . ICML2019

Batch Spectral Penalization (BSP)

- **Batch Spectral Penalization (BSP):** penalize the k largest singular values of source and target feature matrix within each batch

$$\min_{F,G} \mathcal{E}(F,G) + \delta \text{dist}_{P \leftrightarrow Q}(F,D) + \beta L_{\text{bsp}}(F)$$
$$\max_{D} \text{dist}_{P \leftrightarrow Q}(F,D),$$

$$L_{\text{bsp}}(F) = \sum_{i=1}^{k} (\sigma_{s,i}^2 + \sigma_{t,i}^2)$$

Table 3. Accuracy (%) on VisDA-2017 for unsupervised domain adaptation (ResNet-101).

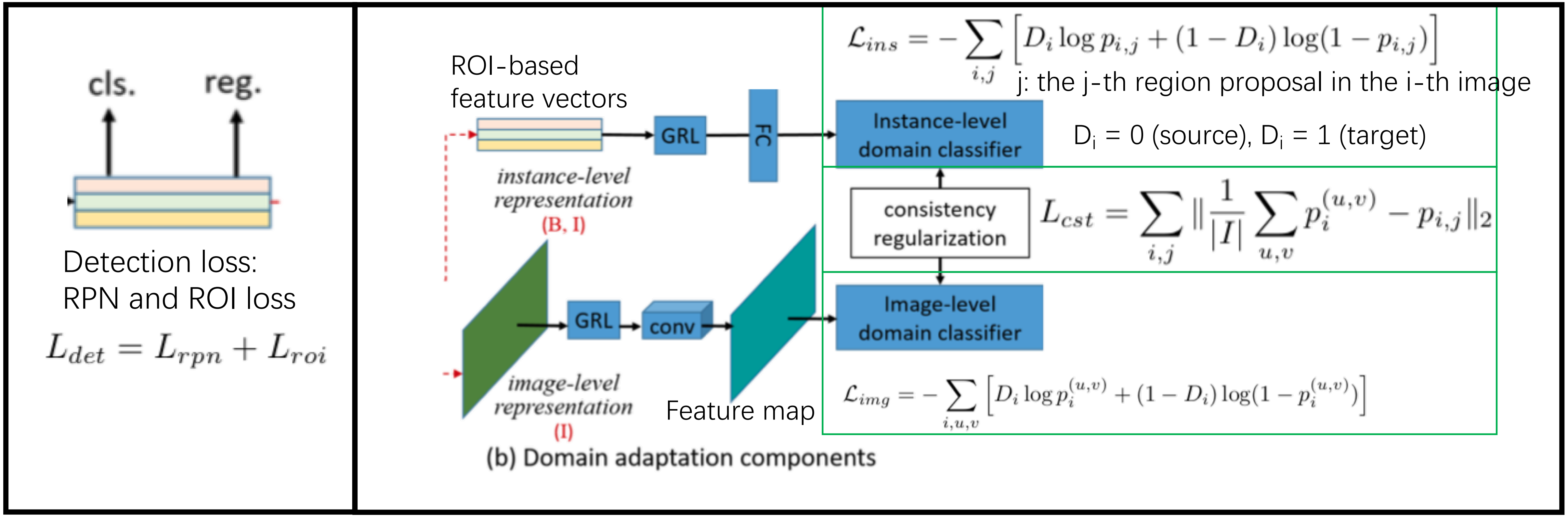
Method	plane	bcybl	bus	car	horse	knife	mcyle	person	plant	sktbrd	train	truck	mean
ResNet-101 (He et al., 2016)	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
DAN (Long et al., 2015)	87.1	63.0	76.5	42.0	**90.3**	42.9	85.9	53.1	49.7	36.3	**85.8**	20.7	61.1
DANN (Ganin et al., 2016)	81.9	**77.7**	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
MCD (Saito et al., 2018)	87.0	60.9	83.7	**64.0**	88.9	79.6	84.7	76.9	**88.6**	40.3	83.0	25.8	71.9
CDAN (Long et al., 2018)	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.7
BSP+DANN (Proposed)	92.2	72.5	**83.8**	47.5	87.0	54.0	86.8	72.4	80.6	66.9	84.5	37.1	72.1
BSP+CDAN (Proposed)	**92.4**	61.0	81.0	57.5	89.0	**80.6**	**90.1**	**77.0**	84.2	**77.9**	82.1	**38.4**	**75.9**

Chen et al. "Transferability vs. Discriminability: Batch Spectral Penalization for Adversarial Domain Adaptation". ICML2019

Multi-Level Adversarial Adaptation [Chen et al, CVPR 18]

Object detection **DA-Faster**-R-CNN

- Adversarial loss （**via GRL**）at both image level and instance level
- Consistent regularization at the two levels

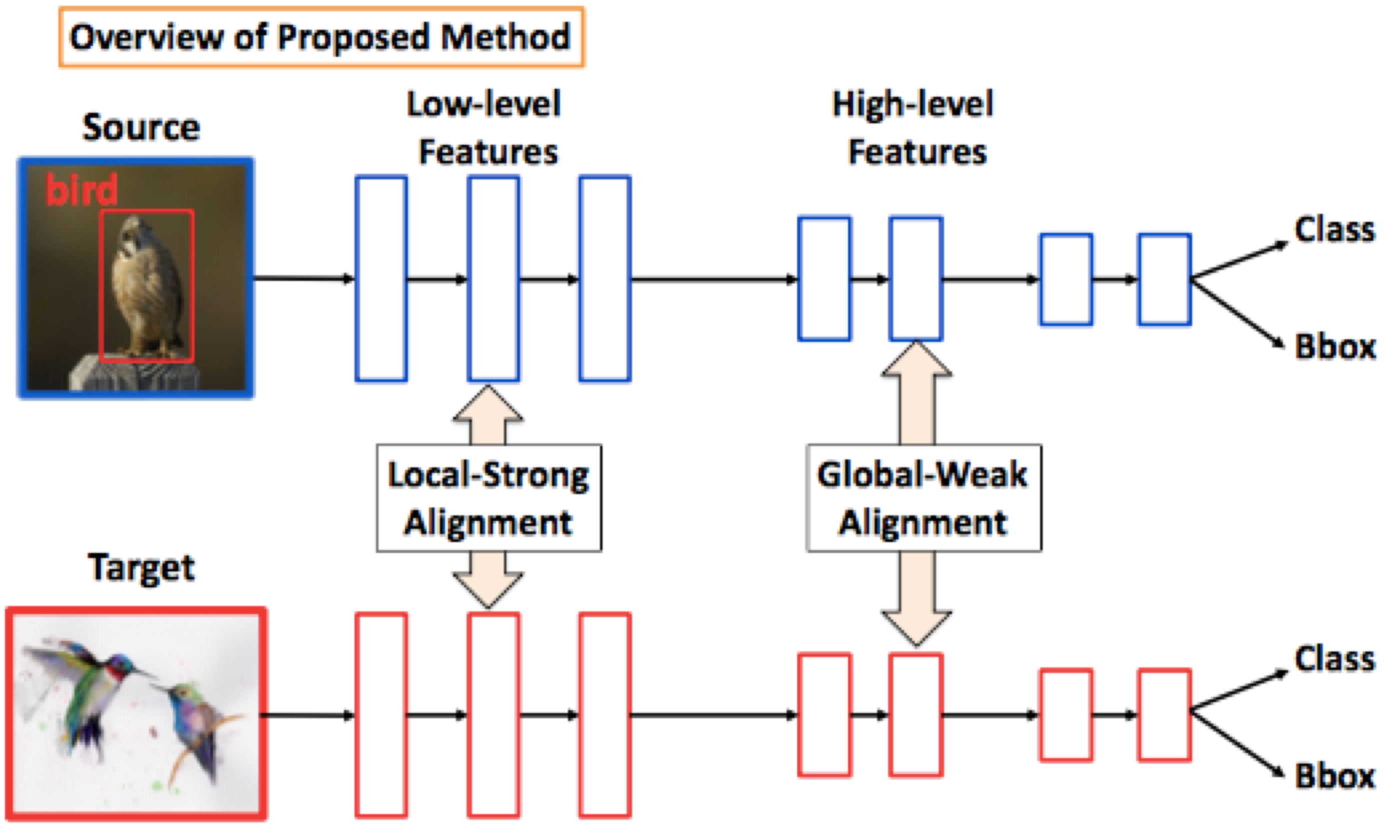


Chen, et al. " Domain Adaptive Faster R-CNN for Object Detection in the Wild:", CVPR 18

Multi-Level Adversarial Alignment [Saito et al, CVPR 19]

Object detection: Strong-Weak

- Domains can have distinct scene layouts and different combinations of objects.
- Local features such as texture and color do not change category level semantics.



Extract global features just before the RPN and local features from lower layers

Learn domain-invariant features that are
- strongly aligned at the local patch level
- weakly (partially) aligned at the global scene level.

Saito, et al. " Strong Weak Distribution Alignment for Adaptive Object Detection :", CVPR 19

Multi-Level Adversarial Alignment

[Saito et al, CVPR 19]

Object detection Extract global features just before the RPN and local features from lower layers

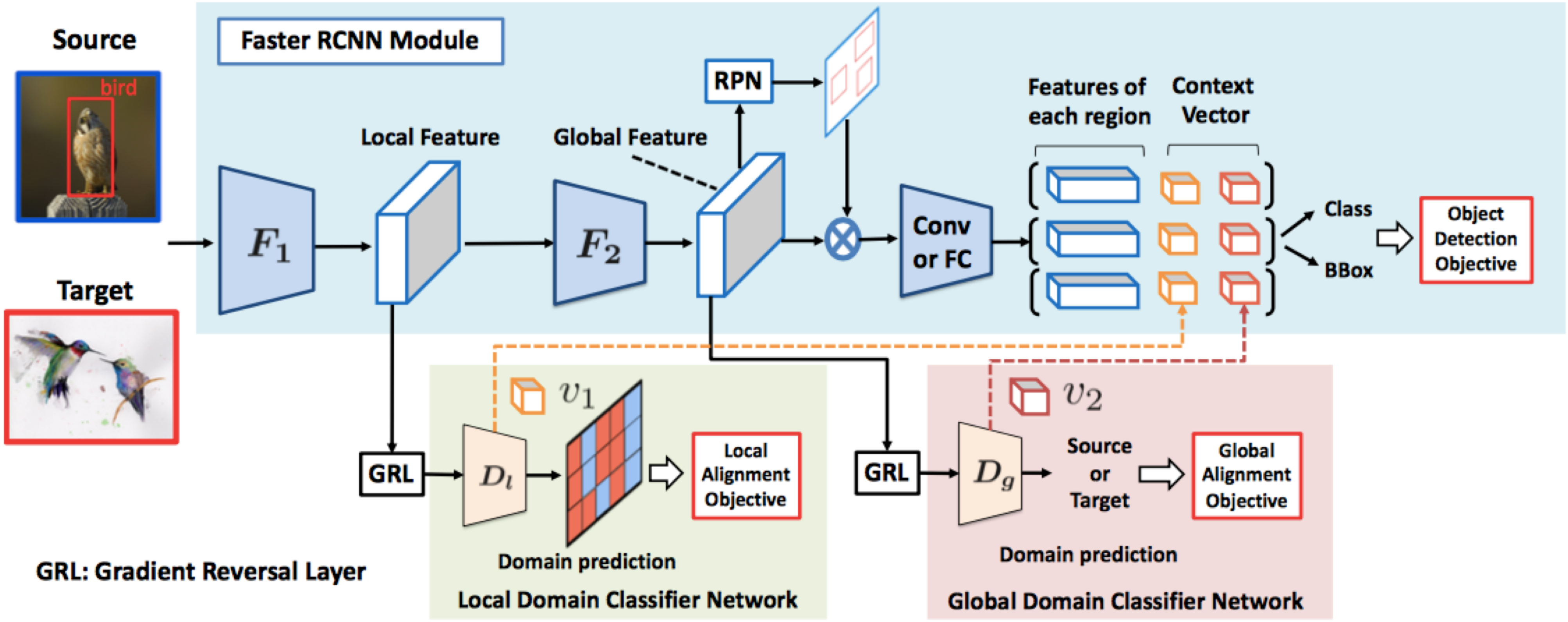


Figure 3. Proposed Network Architecture. Our method performs strong-local alignment by a local domain classifier network and weak-global alignment by a global domain classifier. The context vector is extracted by the domain classifiers and is concatenated in the layer before the final fully connected layer.

$$\max_{D} \min_{F,R} \mathcal{L}_{cls}(F,R) - \lambda \mathcal{L}_{adv}(F,D)$$

Saito, et al. " Strong Weak Distribution Alignment for Adaptive Object Detection :", CVPR 19

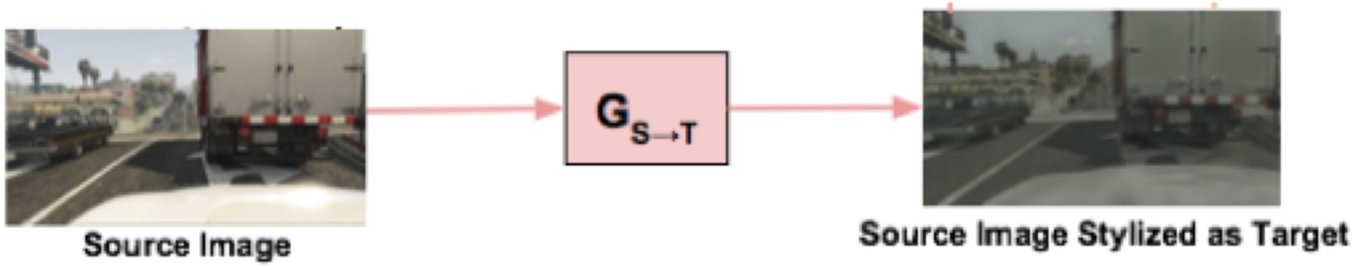
DA Detection Results

Object detection

mAP	SIM10K → Cityscapes	Cityscapes → FoggyCityscapes	KITTI → Cityscapes	Cityscapes → KITTI	PASCALVOC → Clipart	PASCALVOC → WaterColor
Faster R-CNN	34.6	20.3	30.2	53.5	27.8	44.6
DA-Faster (Chen et al, CVPR 18)	38.9	27.6	38.5	64.1	19.8	46.0
Strong-Weak (Saito et al, CVPR 19)	47.7	34.3	-	-	38.1	53.3

Generative Model based Methods

- **Main idea**:
  - Use generative models to generate data in either domains, or transform data to another domain



  - The generated/transformed data can then be used to complement existing data and align the domains

Cycle-Consistent Adversarial DA

Hoffman et al. ICML18

- Limitation of domain alignment techniques:
  - aligning marginal distributions does not enforce semantic consistency
  - alignment at higher levels of a deep representation can fail to model aspects of low-level appearance variance
- CyCADA:
  - transform data from one domain to the other domain
  - adaptation ***at both pixel level*** and ***feature level***
  - **Integrate multiple Losses**:

    cycle-consistency loss, **semantic consistency loss**,

    adversarial loss (**pixel** & **feature level**), **prediction loss**

Hoffman et al. " CyCADA: Cycle-Consistent Adversarial Domain Adaptation :", ICML 18

Cycle-Consistent Adversarial DA

Hoffman et al. ICML18

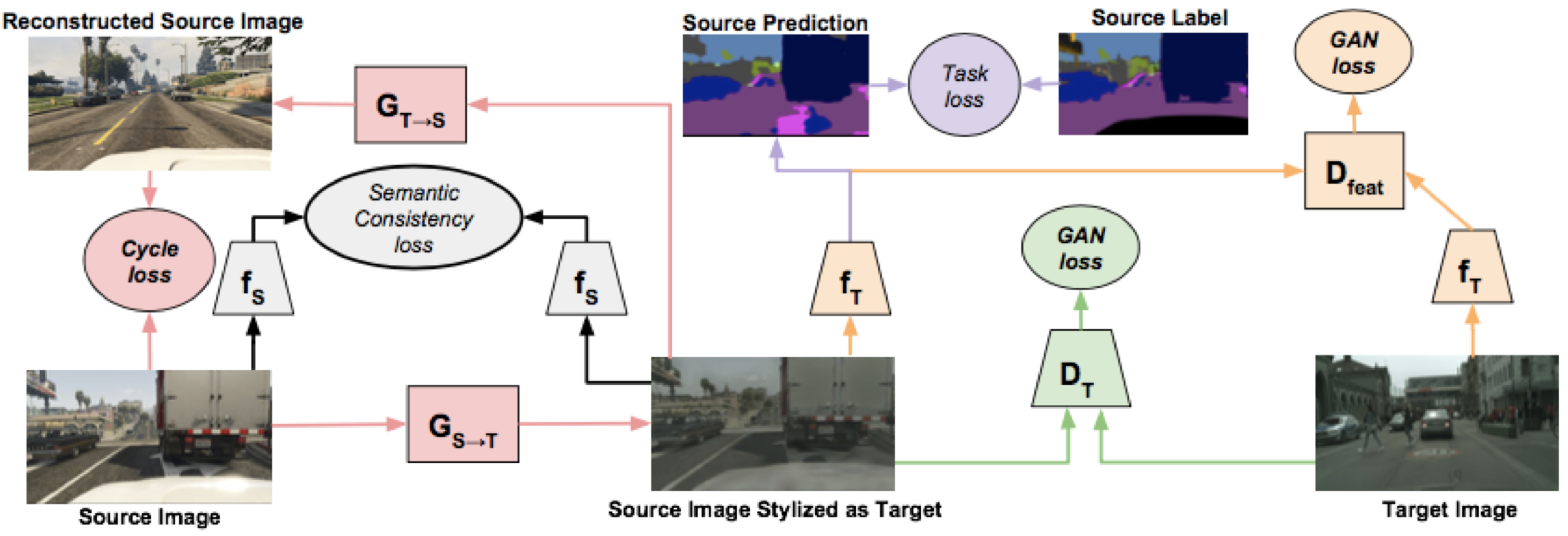


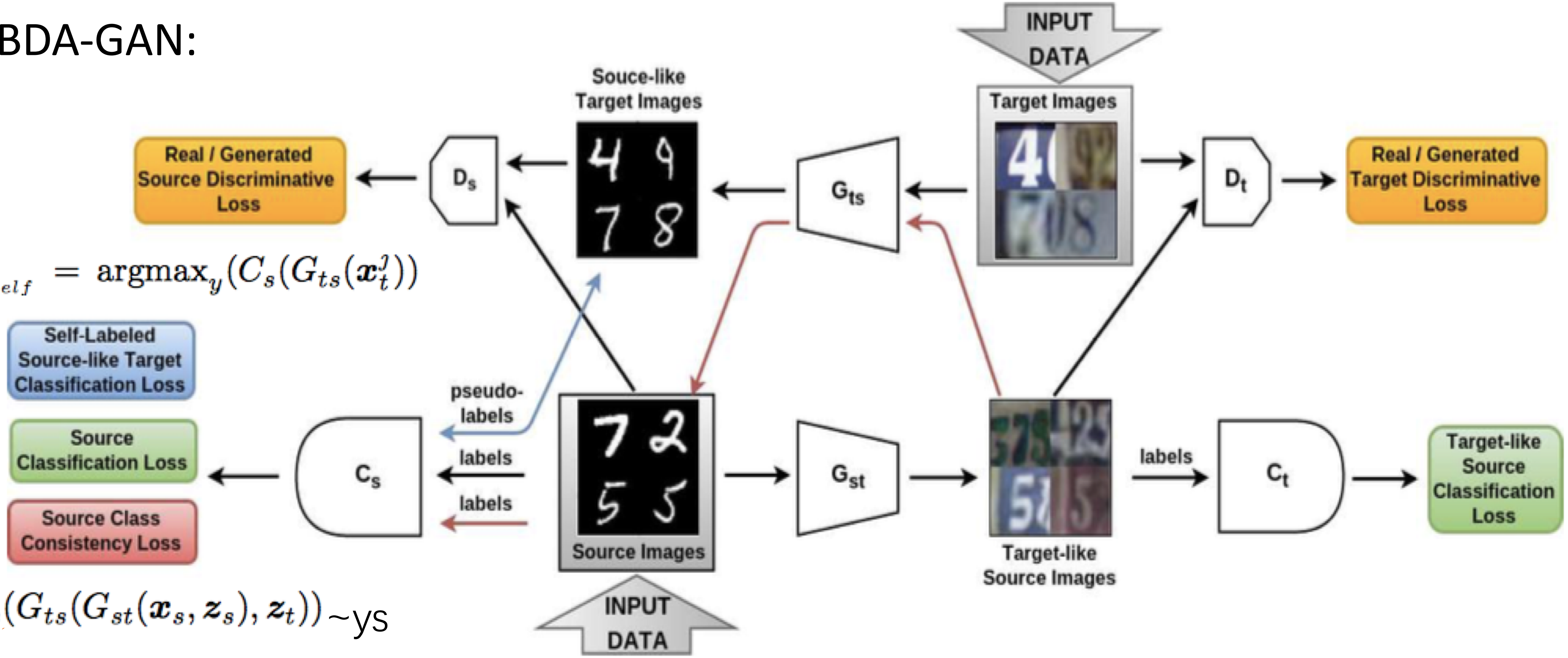
image-level **GAN loss (green**), the feature level **GAN loss** (**orange**), the **source and target semantic consistency losses (black),** the **source cycle loss (red),** and the **source task loss (purple).**

For clarity the target cycle is omitted.

Hoffman et al. " CyCADA: Cycle-Consistent Adversarial Domain Adaptation :", ICML 18

Symmetric Bi-Directional Adaptive GAN

- SBDA-GAN:



$$y^{j}_{t_{self}} = \mathrm{argmax}_y(C_s(G_{ts}(\boldsymbol{x}^j_t)))$$

$$C_s(G_{ts}(G_{st}(\boldsymbol{x}_s, \boldsymbol{z}_s), \boldsymbol{z}_t)) \sim ys$$

- combine bi-directional image transformation with target self-labeling
- use class consistency loss to align the generators in the two directions

Russo et al. " From source to target and back: Symmetric Bi-Directional Adaptive GAN", CVPR 18

DA Recognition Results

Methods	MNIST→USPS	USPS→MNIST	SVHN→MNIST	SYNSIG→GTSRB
DANN (Ganin et.al. JMLR16)	77.1±1.8	73.0±0.2	73.85	88.65
ADDA (Tzeng et al.CVPR17)	89.4±0.2	90.1±0.8	76.0±1.8	
MCD_DA(Saito et al.CVPR18)	94.2±0.7	94.1±0.1	96.2±0.4	94.4±0.3
CDAN(Long et al. NeurIPS18)	93.9	96.9	88.5	
CyCADA (Hoffman.et.al. ICML18)	95.6±0.2	96.5±0.1	90.4±0.4	
GTA(Sankaranarayanan.et.CVPR18)	92.8±0.9	90.8±1.3	92.4±0.9	
SBAD-GAN (Russo et al. CVPR18)	97.6	95.0	76.1	96.7

Office-31							
Methods	A→W	D→W	W→D	A→D	D→A	W→A	Average
ResNet-50 (2016)	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
DANN(Ganin et al.JMLR16)	79.3	97.3	99.6	80.7	65.3	63.2	80.9
ADDA(Tzeng et al.CVPR17)	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
CDAN(Long et al. NeurIPS18)	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
GTA(Sankaranarayanan.et.CVPR18)	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

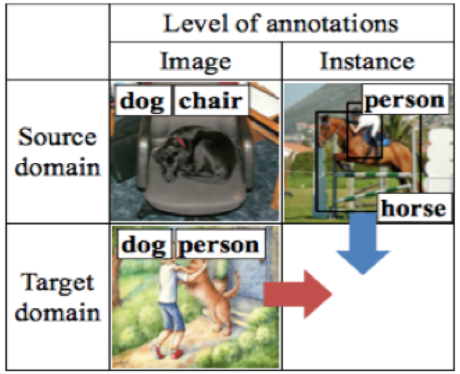
Pseudo-Label based Methods

- Use target domain **unlabeled data with predicted pseudo-labels** to augment labeled training data in the training process

- It is a **general semi-supervised learning strategy**; many methods can be extended to exploit pseudo-labels

Some positive application in domain adaptation:

➢ Progressive domain adaptation for Object detection



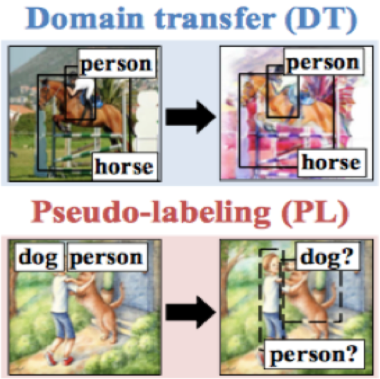


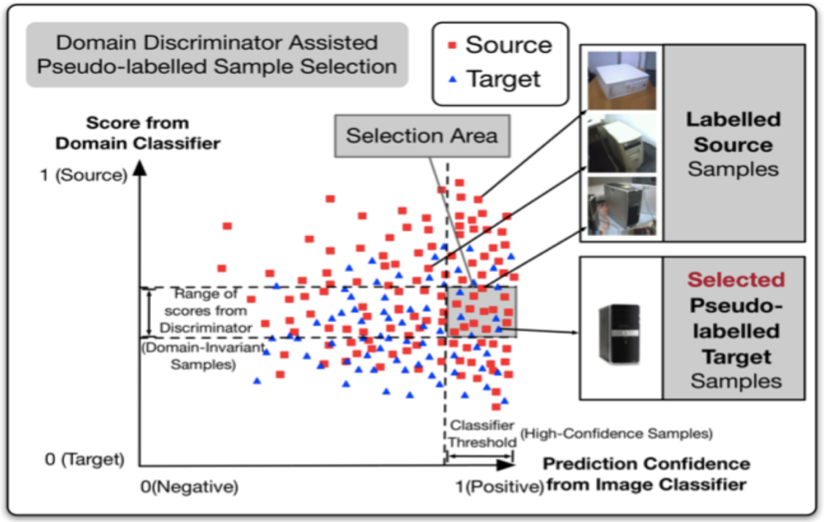
Table 3: Results of our methods on the different baseline FSDs in terms of mAP [%] in Clipart1k.

Method	SSD300	YOLOv2	Faster R-CNN
Baseline	26.8	25.5	26.2
DT	38.0	31.5	32.1
PL	36.4	34.0	29.8
DT+PL	**46.0**	**39.9**	**34.9**
Ideal case	55.4	51.2	50.0

➢ For recognition:

Model	A→W	W→A	A→D	D→A	W→D	D→W	Avg.
ResNet50[16]	73.5	59.8	76.5	56.7	99.0	93.6	76.5
DDC[27]	76.0	63.7	77.5	67.0	98.2	94.8	79.5
DAN[19]	80.5	62.8	78.6	63.6	99.6	97.1	80.4
RTN[20]	84.5	64.8	77.5	66.2	99.4	96.8	81.6
DANN[13]	79.3	63.2	80.7	65.3	99.6	97.3	80.9
JAN[21]	86.0	70.7	85.1	69.2	99.7	96.7	84.6
CAN(ours)	81.5	63.4	85.5	65.9	99.7	98.2	82.4
iCAN(ours)	92.5	69.9	90.1	72.1	100.0	98.8	87.2

Table 1. Comparison of different methods for unsupervised domain adaptation on the Office-31 dataset.



Zhang et al. " Collaborative and Adversarial Network for Unsupervised domain adaptation :", CVPR 18
Inoue et al. " Cross-Domain Weakly-Supervised Object Detection through Progressive Domain Adaptation", CVPR 18

Summary

- Unsupervised domain adaptation has received a lot of attention
- Open domain learning remains to be challenging, but starts drawing attentions
- Most study has focused on classification problems
- Much less effort has been made on more complex tasks such as object detection