



Boulder

# Computer Vision; Image Classification; Domain Adaptation

---

**Maziar Raissi**

**Assistant Professor**

Department of Applied Mathematics

University of Colorado Boulder

[maziar.raissi@colorado.edu](mailto:maziar.raissi@colorado.edu)



# Learning Transferable Features with Deep Adaptation Networks

Boulder

## Unsupervised Domain Adaptation

$$\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s} \rightarrow \text{source domain}$$

$$\mathcal{D}_t = \{x_j^t\}_{j=1}^{n_t} \rightarrow \text{target domain}$$

Build a classifier which can minimize target risk!

$$y = c_\theta(x) \rightarrow \text{classifier}$$

$$\varepsilon_t(\theta) = \Pr_{(x,y) \sim \mathcal{D}_t}[c_\theta(x) \neq y] \rightarrow \text{target risk}$$

## Semi-supervised Adaptation

$$\mathcal{D}_a = \{(x_i^a, y_i^a)\}_{i=1}^{n_a} \rightarrow n_a \text{ annotated examples of source and target domains}$$

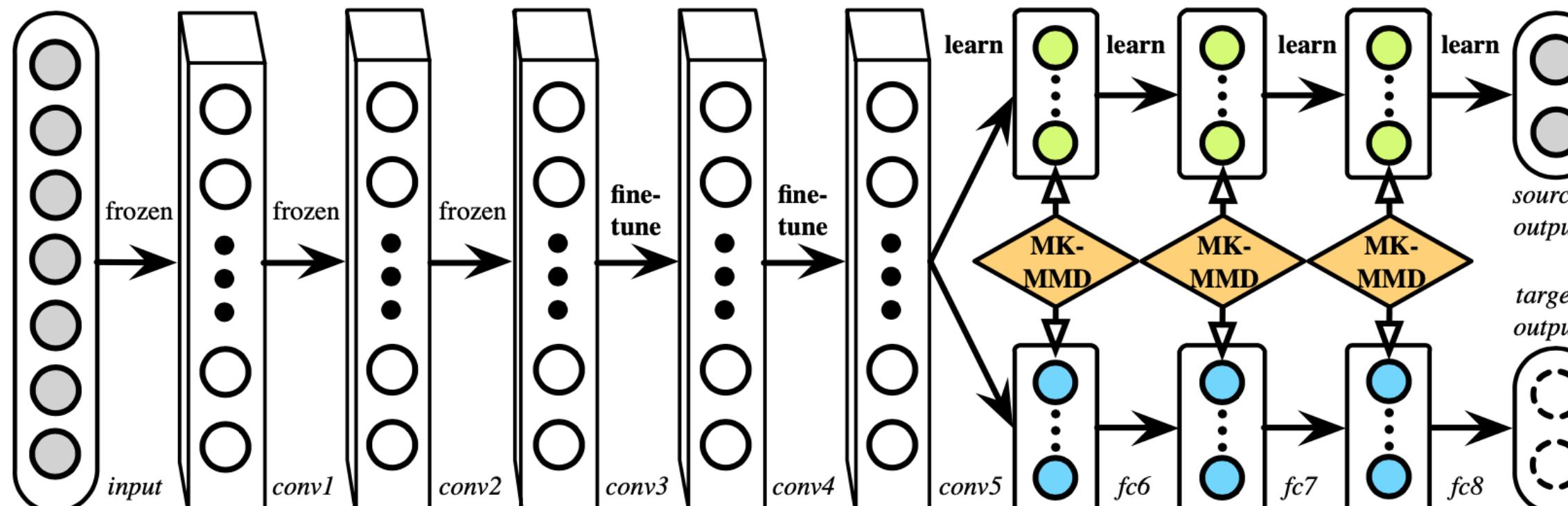
## Multi-Layer Adaptation & Multi-Kernel MMD

MMD → Maximum Mean Discrepancy

$$\min_{\theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J(c_\theta(x_i^a), y_i^a) + \lambda \sum_{l=l_1}^{l_2} d_k^2(\mathcal{D}_s^l, \mathcal{D}_t^l)$$

cross-entropy loss

layer indices ( $l_1 = 6$  &  $l_2 = 8$ )



$$\mathcal{D}_s^l = \{h_i^{sl} := \phi^l(x_i^s)\}_{i=1}^{n_s}$$

$$\mathcal{D}_t^l = \{h_j^{tl} := \phi^l(x_j^t)\}_{j=1}^{n_t}$$

$k \rightarrow$  convex combination of  $m$  kernels

$$k = \sum_{u=1}^m \beta_u k_u, \sum_{u=1}^m \beta_u = 1, \beta_u \geq 0$$

$$k_u(h_i, h_j) = \exp\left(-\frac{\|h_i - h_j\|^2}{\gamma_u}\right)$$

↳ Gaussian kernels

$$d_k^2(\mathcal{D}_s^l, \mathcal{D}_t^l) = \frac{1}{n_s^2} \sum_{i=1}^{n_s} \sum_{j=1}^{n_s} k(h_i^{sl}, h_j^{sl}) + \frac{1}{n_t^2} \sum_{i=1}^{n_t} \sum_{j=1}^{n_t} k(h_i^{tl}, h_j^{tl}) - \frac{2}{n_s n_t} \sum_{i=1}^{n_s} \sum_{j=1}^{n_t} k(h_i^{sl}, h_j^{tl})$$

incurs a complexity of  $O(n^2)$

An unbiased estimate of MK-MMD can be computed with linear complexity (see the paper!)

We can learn the optimal kernel parameters  $\beta_u$  by  $\max_{k \in \mathcal{K}} d_k^2(\mathcal{D}_s^l, \mathcal{D}_t^l) \sigma_k^{-2}$   
 $\sigma_k \rightarrow$  estimation variance

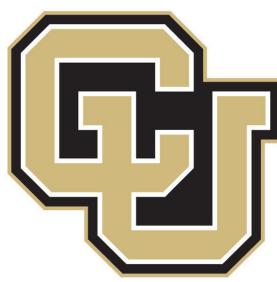
Minimizing MMD is equivalent to maximizing the error of classifying the source from the target (two-sample classifier)

Method	A → C	W → C	D → C	C → A	C → W	C → D	Average
TCA	42.7 ± 0.0	34.1 ± 0.0	35.4 ± 0.0	54.7 ± 0.0	50.5 ± 0.0	50.3 ± 0.0	44.6
GFK	41.4 ± 0.0	26.4 ± 0.0	36.4 ± 0.0	56.2 ± 0.0	43.7 ± 0.0	42.0 ± 0.0	41.0
CNN	83.8 ± 0.3	76.1 ± 0.5	80.8 ± 0.4	91.1 ± 0.2	83.1 ± 0.3	89.0 ± 0.3	84.0
LapCNN	83.6 ± 0.6	77.8 ± 0.5	80.6 ± 0.4	92.1 ± 0.3	81.6 ± 0.4	87.8 ± 0.4	83.9
DDC	84.3 ± 0.5	76.9 ± 0.4	80.5 ± 0.2	91.3 ± 0.3	85.5 ± 0.3	89.1 ± 0.3	84.6
DAN <sub>7</sub>	84.7 ± 0.3	78.2 ± 0.5	81.8 ± 0.3	91.6 ± 0.4	87.4 ± 0.3	88.9 ± 0.5	85.4
DAN <sub>8</sub>	84.4 ± 0.3	80.8 ± 0.4	81.7 ± 0.2	91.7 ± 0.3	90.5 ± 0.4	89.1 ± 0.4	86.4
DAN <sub>SK</sub>	84.1 ± 0.4	79.9 ± 0.4	81.1 ± 0.5	91.4 ± 0.3	86.9 ± 0.5	89.5 ± 0.3	85.5
DAN	86.0 ± 0.5	81.5 ± 0.3	82.0 ± 0.4	92.0 ± 0.3	92.0 ± 0.4	90.5 ± 0.2	87.3

Accuracy on *Office-10 + Caltech-10* dataset with standard unsupervised adaptation protocol

Method	A → W	D → W	W → D	A → D	D → A	W → A	Average
TCA	21.5 ± 0.0	50.1 ± 0.0	58.4 ± 0.0	11.4 ± 0.0	8.0 ± 0.0	14.6 ± 0.0	27.3
GFK	19.7 ± 0.0	49.7 ± 0.0	63.1 ± 0.0	10.6 ± 0.0	7.9 ± 0.0	15.8 ± 0.0	27.8
CNN	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
LapCNN	60.4 ± 0.3	94.7 ± 0.5	99.1 ± 0.2	63.1 ± 0.6	51.6 ± 0.4	48.2 ± 0.5	69.5
DDC	61.8 ± 0.4	95.0 ± 0.5	98.5 ± 0.4	64.4 ± 0.3	52.1 ± 0.8	52.2 ± 0.4	70.6
DAN <sub>7</sub>	63.2 ± 0.2	94.8 ± 0.4	98.9 ± 0.3	65.2 ± 0.4	52.3 ± 0.4	52.1 ± 0.4	71.1
DAN <sub>8</sub>	63.8 ± 0.4	94.6 ± 0.5	98.8 ± 0.6	65.8 ± 0.4	52.8 ± 0.4	51.9 ± 0.5	71.3
DAN <sub>SK</sub>	63.3 ± 0.3	95.6 ± 0.2	99.0 ± 0.4	65.9 ± 0.7	53.2 ± 0.5	52.1 ± 0.4	71.5
DAN	68.5 ± 0.4	96.0 ± 0.3	99.0 ± 0.2	67.0 ± 0.4	54.0 ± 0.4	53.1 ± 0.3	72.9

Amazon (A), Webcam (W), DSLR (D)



Boulder

# Domain-Adversarial Training of Neural Networks

## Domain Adaptation

Data at training and test time come from similar but different distributions!

$X \rightarrow$  input space

$Y = \{0, 1, \dots, L - 1\} \rightarrow$  set of  $L$  possible labels

$\mathcal{D}_S \rightarrow$  source domain

$\mathcal{D}_T \rightarrow$  target domain

$\mathcal{D}_S, \mathcal{D}_T \rightarrow$  distributions over  $X \times Y$

$S \rightarrow$  labeled source sample drawn i.i.d from  $\mathcal{D}_S$

$T \rightarrow$  unlabeled target sample drawn i.i.d from  $\mathcal{D}_T^X$

$\mathcal{D}_T^X \rightarrow$  marginal distribution of  $\mathcal{D}_T$  over  $X$

$S = \{(x_i, y_i)\}_{i=1}^n \sim (\mathcal{D}_S)^n$

$T = \{x_i\}_{i=n+1}^N \sim (\mathcal{D}_T^X)^{n'}$

$N = n + n' \rightarrow$  total number of samples

Build a classifier  $\eta : X \rightarrow Y$  with a low target risk:

$$\mathcal{R}_{\mathcal{D}_T}(\eta) = \Pr_{(x,y) \sim \mathcal{D}_T}(\eta(x) \neq y)$$

## Theorem

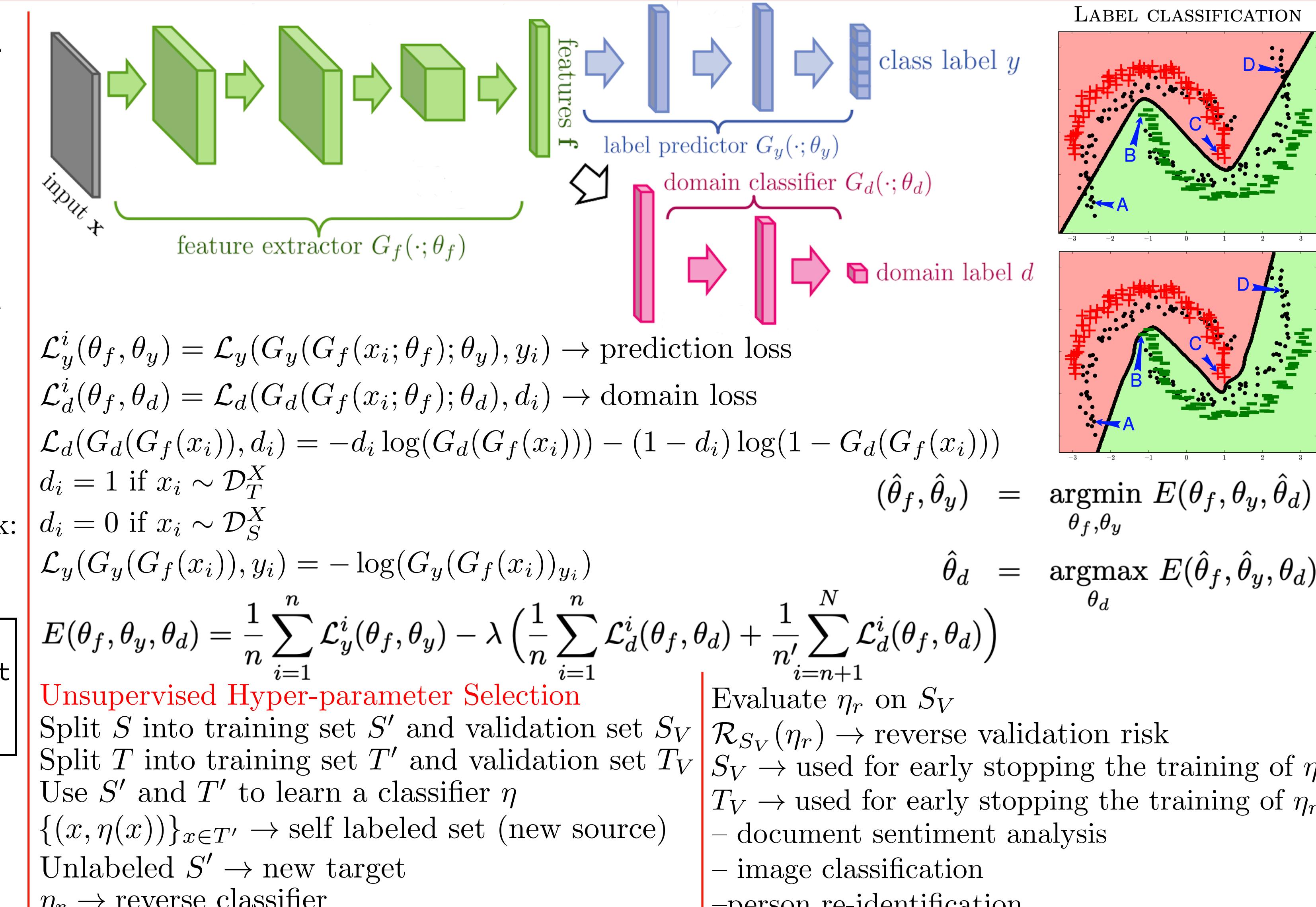
For effective domain transfer to be achieved, predictions must be made based on features that cannot discriminate between the training (source) and test (target) domains.

## Domain-Adversarial Neural Network (DANN)

$G_f(\cdot; \theta_f) \rightarrow$  feature extractor

$G_y(\cdot; \theta_y) \rightarrow$  label predictor

$G_d(\cdot; \theta_d) \rightarrow$  domain classifier



## Unsupervised Hyper-parameter Selection

Split  $S$  into training set  $S'$  and validation set  $S_V$

Split  $T$  into training set  $T'$  and validation set  $T_V$

Use  $S'$  and  $T'$  to learn a classifier  $\eta$

$\{(x, \eta(x))\}_{x \in T'} \rightarrow$  self labeled set (new source)

Unlabeled  $S' \rightarrow$  new target

$\eta_r \rightarrow$  reverse classifier

Evaluate  $\eta_r$  on  $S_V$

$\mathcal{R}_{S_V}(\eta_r) \rightarrow$  reverse validation risk

$S_V \rightarrow$  used for early stopping the training of  $\eta$

$T_V \rightarrow$  used for early stopping the training of  $\eta_r$

- document sentiment analysis

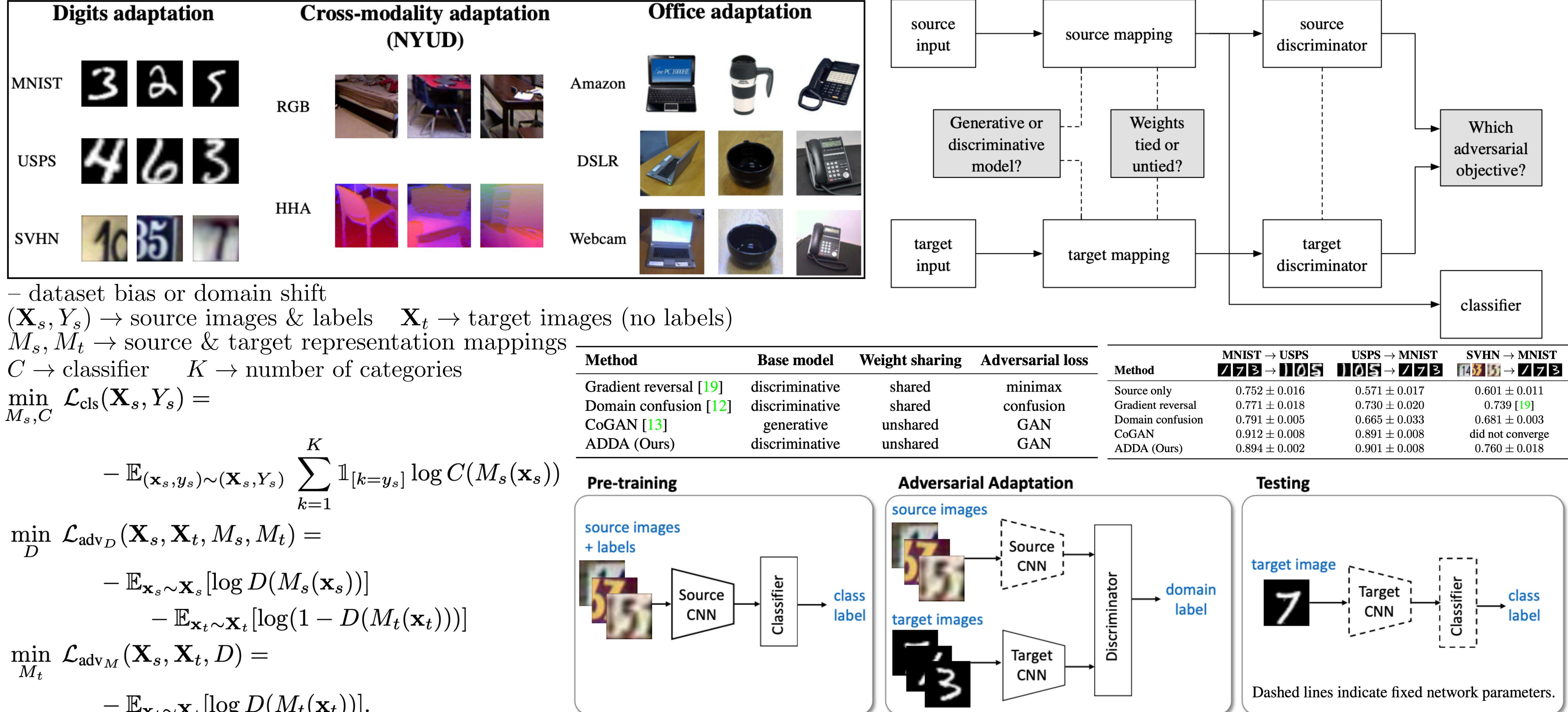
- image classification

- person re-identification



Boulder

# Adversarial Discriminative Domain Adaptation





Boulder

# Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks

Synthetic data (game engines or renderers): ground-truth annotations are generated automatically

- map representations between the two domains
- learn to extract features that are domain-invariant
- transformation in the pixel space from one domain to the other

## Unsupervised Pixel-level Domain Adaptation (PixelDA)

Assumption: The differences between the domains are primarily low-level (due to noise, resolution, illumination, color) rather than high-level (types of objects, geometric variations, etc).

$X^s = \{x_i^s, y_i^s\}_{i=1}^{N^s}$  → labeled dataset of  $N^s$  samples from the source domain

$X^t = \{x_i^t\}_{i=1}^{N^t}$  → unlabeled dataset of  $N^t$  samples from the target domain

$x^f = G(x^s, z; \theta_G)$  → generator

↳ adapted or fake image

$z \sim p_z$  → noise vector

$X^f = \{G(x^s, z), y^s\}$  → new dataset of any size

↳ adapted dataset

The task-specific classifier can be trained as if the training and test data were from the same distribution.

## Learning

$d = D(x; \theta_d)$  → discriminator

↳ likelihood that a given image has been sampled from the target domain  
distinguish between “fake” images produced by the generator and “real” images from the target domain

$\hat{y} = T(x; \theta_T)$  → classifier (assigns task specific labels  $\hat{y}$  to images  $x \in X^f \cup X^t$ )

$$\min_{\theta_G, \theta_T} \max_{\theta_D} \alpha \mathcal{L}_d(D, G) + \beta \mathcal{L}_t(G, T) + \gamma \mathcal{L}_c(G)$$

$$\mathcal{L}_d(D, G) = \mathbb{E}_{\mathbf{x}^t} [\log D(\mathbf{x}^t; \theta_D)] + \mathbb{E}_{\mathbf{x}^s, \mathbf{z}} [\log(1 - D(G(\mathbf{x}^s, \mathbf{z}; \theta_G); \theta_D))] \quad \text{domain loss}$$

$$\mathcal{L}_t(G, T) = \mathbb{E}_{\mathbf{x}^s, \mathbf{y}^s, \mathbf{z}} [-\mathbf{y}^s^\top \log T(G(\mathbf{x}^s, \mathbf{z}; \theta_G); \theta_T) - \mathbf{y}^s^\top \log T(\mathbf{x}^s); \theta_T] \quad \text{task-specific loss (cross-entropy)}$$

$y^s$  → one-hot-encoding

## Content-similarity loss

**Prior knowledge:** Single objects are rendered on black backgrounds and consequently we expect images adapted from these renderings to have similar foregrounds and different backgrounds from the equivalent source images.

$$\mathcal{L}_c(G) = \mathbb{E}_{\mathbf{x}^s, \mathbf{z}} \left[ \frac{1}{k} \|(\mathbf{x}^s - G(\mathbf{x}^s, \mathbf{z}; \theta_G)) \circ \mathbf{m}\|_2^2 - \frac{1}{k^2} ((\mathbf{x}^s - G(\mathbf{x}^s, \mathbf{z}; \theta_G))^\top \mathbf{m})^2 \right]$$

masked-PMSE (pairwise mean squared error) loss

$m \in \mathbb{R}^k$  → binary mask (foreground vs background)

Model	MNIST to USPS	MNIST to MNIST-M
Source Only	78.9	63.6 (56.6)
CORAL [41]	81.7	57.7
MMD [44, 31]	81.1	76.9
DANN [14]	85.1	77.4
DSN [5]	91.3	83.2
CoGAN [30]	91.2	62.0
Our PixelDA	<b>95.9</b>	<b>98.2</b>
Target-only	96.5	96.4 (95.9)

nearest neighbors in  
the MNIST-M training  
set of the generated  
samples





Boulder

# CyCADA: Cycle-Consistent Adversarial Domain Adaptation

Adversarial Adaptation Models:

- discovering domain invariant representations (feature space methods)
- mapping between unpaired image domains (image space methods)

## Unsupervised Adaptation

$X_S \rightarrow$  source data       $Y_S \rightarrow$  source labels  
 $X_T \rightarrow$  target data      no target labels!

Learn a model  $f_T$  that correctly predicts the label for the target data  $X_T$

## Pre-train Source Task Model

$$\mathcal{L}_{\text{task}}(f_S, X_S, Y_S) = -\mathbb{E}_{(x_s, y_s) \sim (X_S, Y_S)} \sum_{k=1}^K \mathbb{1}_{[k=y_s]} \log \left( \sigma(f_S^{(k)}(x_s)) \right)$$

softmax function

domain shift leads to reduced performance when evaluating on target data

## Pixel-level Adaptation

$G_{S \rightarrow T} \rightarrow$  mapping from source to target

$\mathcal{L}$  trained to produce target samples that fool adversarial discriminator  $D_T$

$$\mathcal{L}_{\text{GAN}}(G_{S \rightarrow T}, D_T, X_T, X_S) = \mathbb{E}_{x_t \sim X_T} [\log D_T(x_t)] + \mathbb{E}_{x_s \sim X_S} [\log(1 - D_T(G_{S \rightarrow T}(x_s)))]$$

$$\mathcal{L}_{\text{task}}(f_T, G_{S \rightarrow T}(X_S), Y_S) \rightarrow \text{learn a target model } f_T$$

No way to guarantee that  $G_{S \rightarrow T}(x_s)$  preserves the structure and content of  $x_s$ !

$$\mathcal{L}_{\text{GAN}}(G_{T \rightarrow S}, D_S, X_S, X_T)$$

$$\begin{aligned} \mathcal{L}_{\text{cyc}}(G_{S \rightarrow T}, G_{T \rightarrow S}, X_S, X_T) = & \mathbb{E}_{x_s \sim X_S} [||G_{T \rightarrow S}(G_{S \rightarrow T}(x_s)) - x_s||_1] \\ & + \mathbb{E}_{x_t \sim X_T} [||G_{S \rightarrow T}(G_{T \rightarrow S}(x_t)) - x_t||_1] \end{aligned}$$

## Prevent Label Flipping

semantic consistency  $\leftarrow \mathcal{L}_{\text{sem}}(G_{S \rightarrow T}, G_{T \rightarrow S}, X_S, X_T, f_S) =$

$$\mathcal{L}_{\text{task}}(f_S, G_{T \rightarrow S}(X_T), p(f_S, X_T))$$

$$\begin{aligned} & + \mathcal{L}_{\text{task}}(f_S, G_{S \rightarrow T}(X_S), p(f_S, X_S)) \\ & p(f, X) = \arg \max f(x) \end{aligned}$$

$\mathcal{L}$  predicted label  
 $f_S \rightarrow$  pretrained and fixed

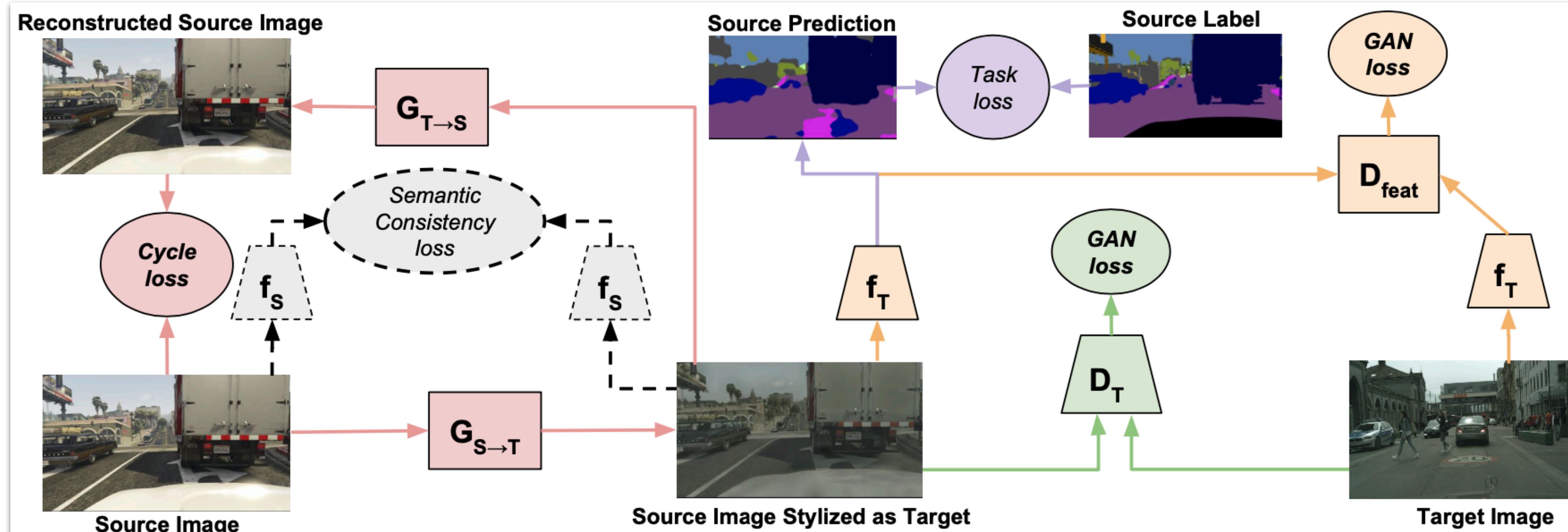
## Feature-level Adaptation

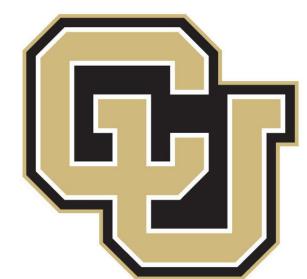
$$\mathcal{L}_{\text{GAN}}(f_T, D_{\text{feat}}, f_S(G_{S \rightarrow T}(X_S)), X_T)$$

## Complete Objective

$$f_T^* = \arg \min_{f_T} \min_{G_{S \rightarrow T}} \max_{D_S, D_T} \mathcal{L}_{\text{CyCADA}}$$

Model	Accuracy (%)
Source only	67.1
CyCADA - no feat adapt, no semantic loss	70.3
CyCADA - no feat adapt	71.2
CyCADA - no cycle consistency	75.7
CyCADA - no pixel adapt	83.8
CyCADA (Full)	90.4
Target Fully Supervised	99.2





Boulder

# Questions?

---