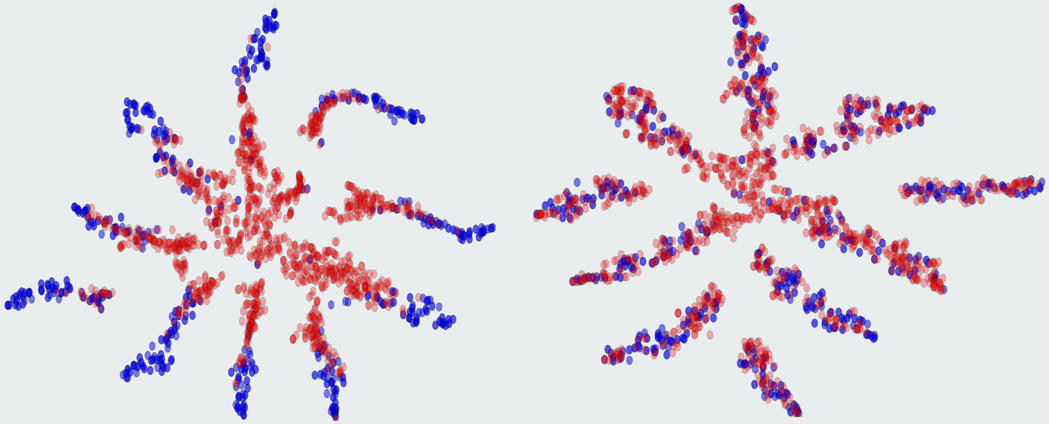
Unsupervised Domain Adaptation by Backpropagation

Chih-Hui Ho, Xingyu Gu, Yuan Qi



Outline

- Introduction
- Related works
- Proposed solution
- Experiments
- Conclusions

Problems

Deep network: requires massive **labeled** training data.

Labeled data:

- **Available** sometimes:
  - Image recognition
  - Speech recognition
  - Recommendation
- **Difficult to collect** sometimes:
  - Robotics
  - Disaster
  - Medical diagnosis
  - Bioinformatics

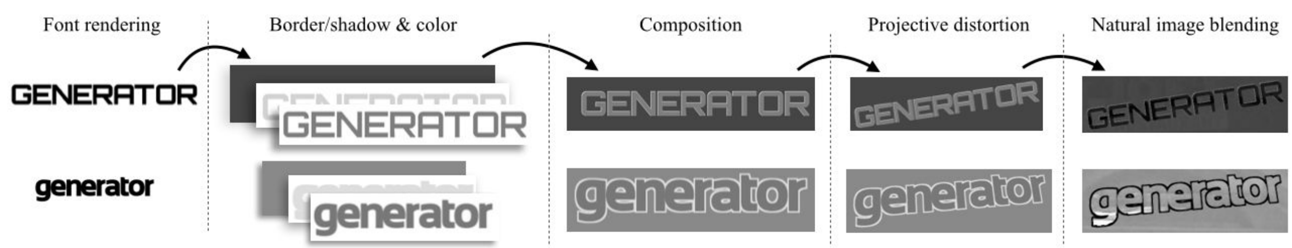
Problems

Test time failure: distribution of actual data is different from training data.

Example: Model is

- **Trained** on **synthetic data** (abundant and fully labeled), but
- **Tested** on **real data**.

MJSynth
(synthetic)

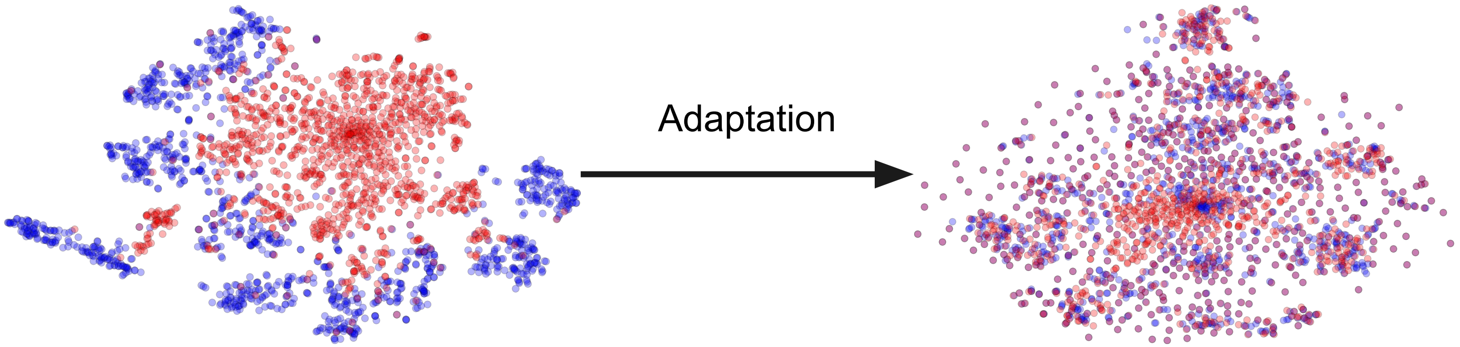


IIIT5K
(real)

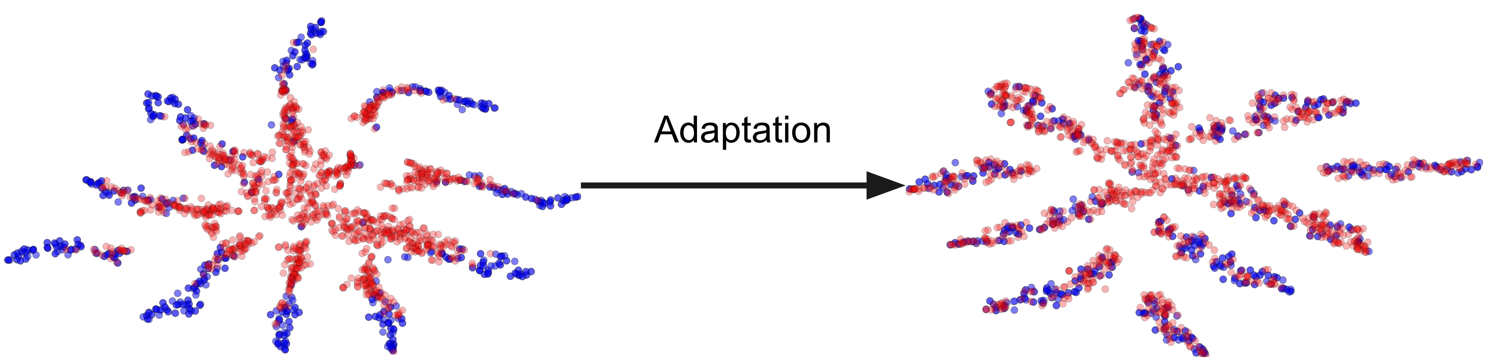
Results

- Source datapoint
- Target datapoint

- MNIST → MNIST-M (extracted features)



- SYN NUMBERS → SVHN (label classifier's last hidden layer)



Objective

Given:

- Lots of **labeled** data in the **source** domain (e.g. synthetic images)
- Lots of **unlabeled** data in the **target** domain (e.g. real images)

Domain Adaptation (**DA**):

In the presence of a ***shift*** between source and target domain,
Train a network on **source** domain that performs well on **target** domain.

Objective

Example: Office dataset

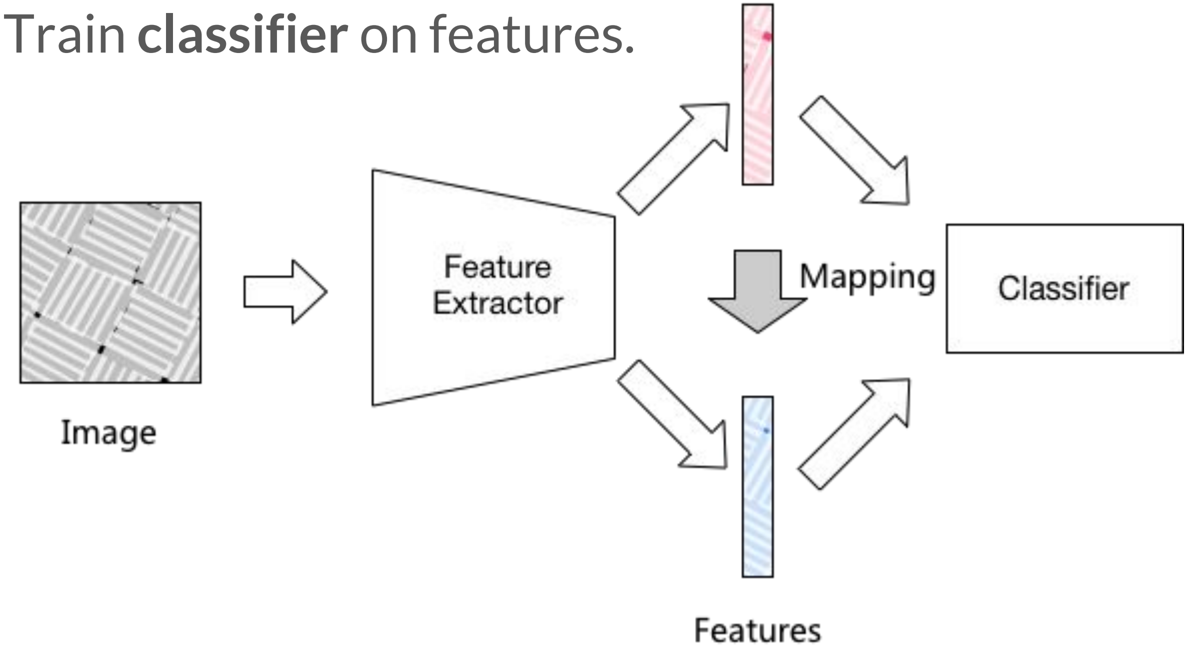
- **Source**:
  Amazon photos of office objects
  (on white background)

- **Target**:
  Consumer photos of office objects
  (taken by DSLR camera / webcam)

Previous Approaches - DLID

Deep **L**earning by **I**nterpolating between **D**omains

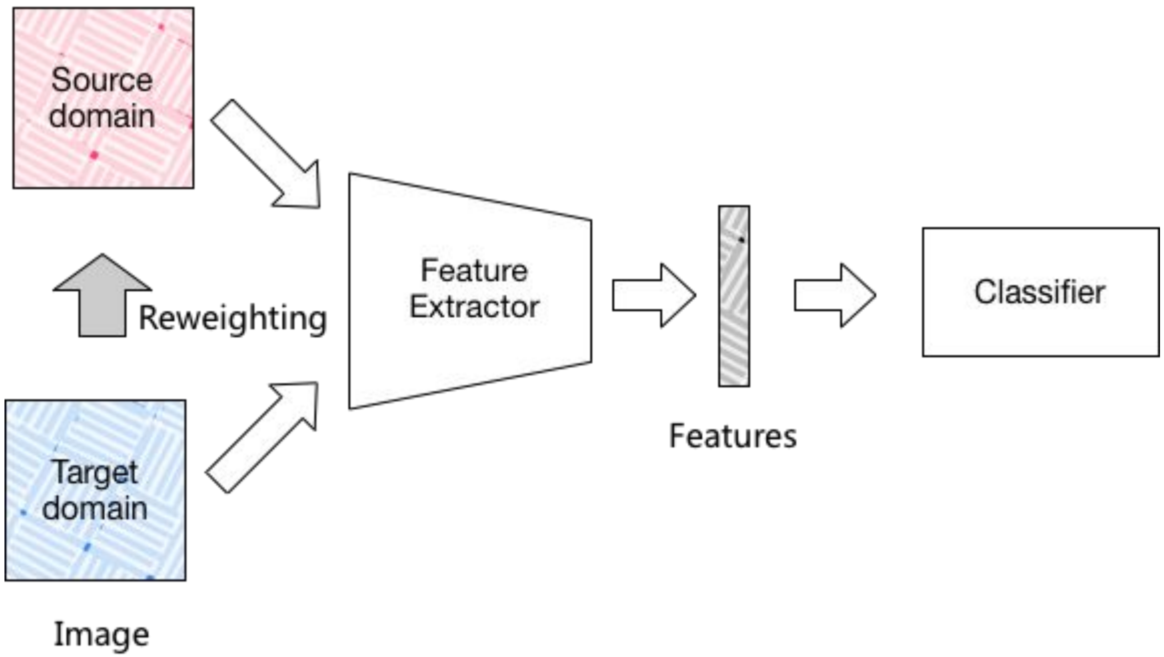
- **Feature transformation** mapping source into target.
  - Train **feature extractor** layer-wise.
  - Gradually replacing source samples with target samples.
  - Train **classifier** on features.



Previous Approaches - MMD

Maximum **M**ean **D**iscrepancy (measures domain-distance)

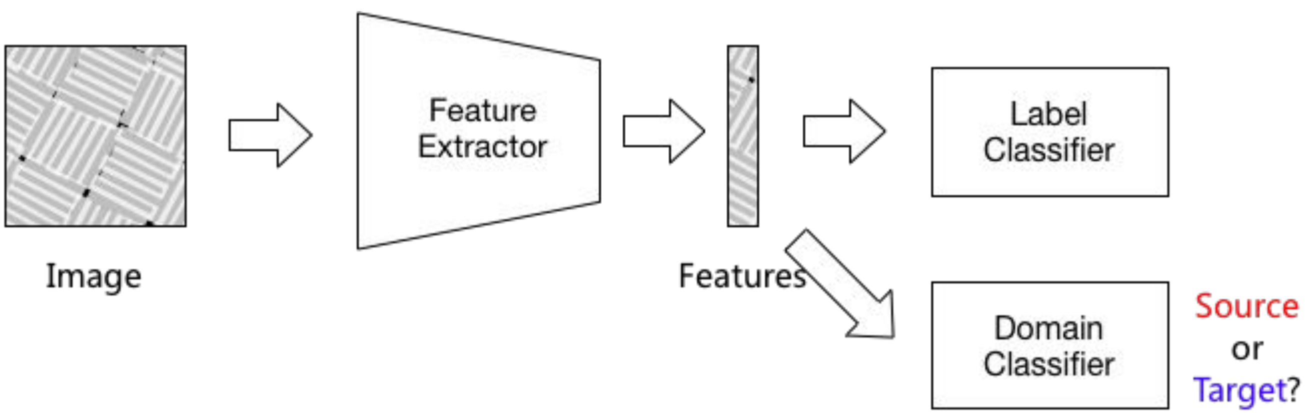
- **Reweighting** target domain images.
  - **Distance** between source and target distributions.
  - Explicit distance measurement (e.g. kernel Hilbert space).



Proposed Solution - Deep Domain Adaptation (DDA)

Standard CNN + **domain classifier.**

- An **implicit** way to measure similarity between source and target.
  - If domain classifier performs **good**: **dissimilar** features.
  - If domain classifier performs **bad**: **similar** features.
- Objective: feature is **best** for label classifier, and **worst** for domain classifier.



Improvement

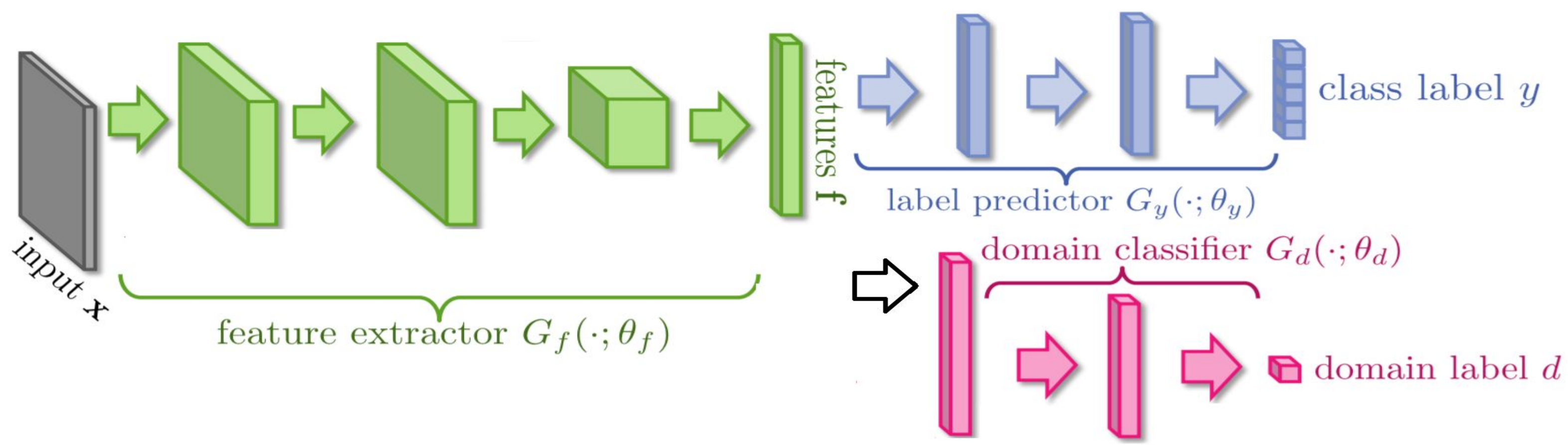
	Previous approaches	Proposed solution
Measurement of similarity between domains	**Explicit** (distance in Hilbert space)	**Implicit** (performance of domain classifier)
Training steps	**Separate** feature extractor and label classifier	**Jointly** trained by backpropagation
Architecture	**Complicated**	**Simple** (standard CNN + domain classifier)

Proposed Solution

- Notation
  - x_i :training samples (from both source and target domain)
  - y_i : class label (only source domain has labels)
  - d_i : 0 (source domain) or 1 (target domain)
- x_i in source domain has d_i = 0 and y_i
- x_i in target domain has d_i = 1 (target domain has no label)

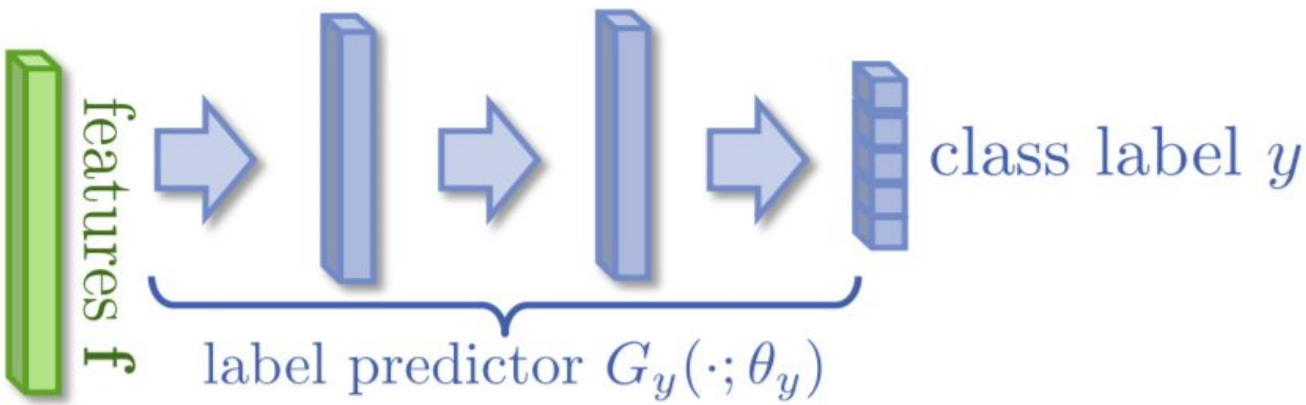
Proposed Solution

- G_f : feature extractor
- G_y : label predictor
- G_d : domain classifier



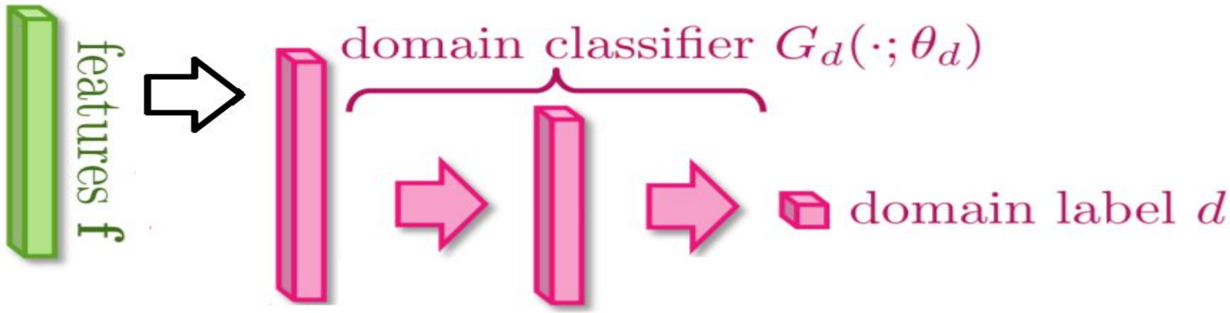
Proposed Solution – Label predictor

- θ_y denotes the parameters in label classifier
- Given a feature, label classifier tries to predict the label of the feature
- Multi-class classification task
- The loss is categorical cross entropy
- θ_y minimize the loss the label classifier
- θ_y is updated as $\theta_y \longleftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y}$
- μ is the learning rate



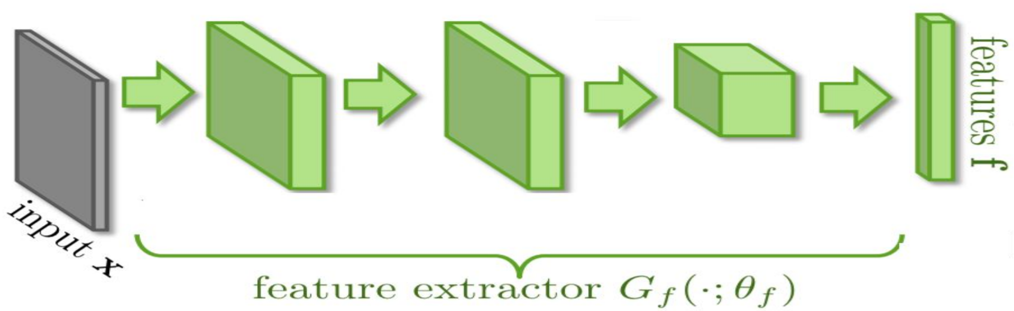
Proposed Solution

- θ_d denotes the parameters in domain classifier
- Given a feature, domain classifier tries to predict whether the feature comes from source or target domain
- Binary classification task
- The loss is binary cross entropy
- θ_d minimizes the loss the domain classifier
- θ_d is updated as $\theta_d \longleftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d}$
- μ is the learning rate



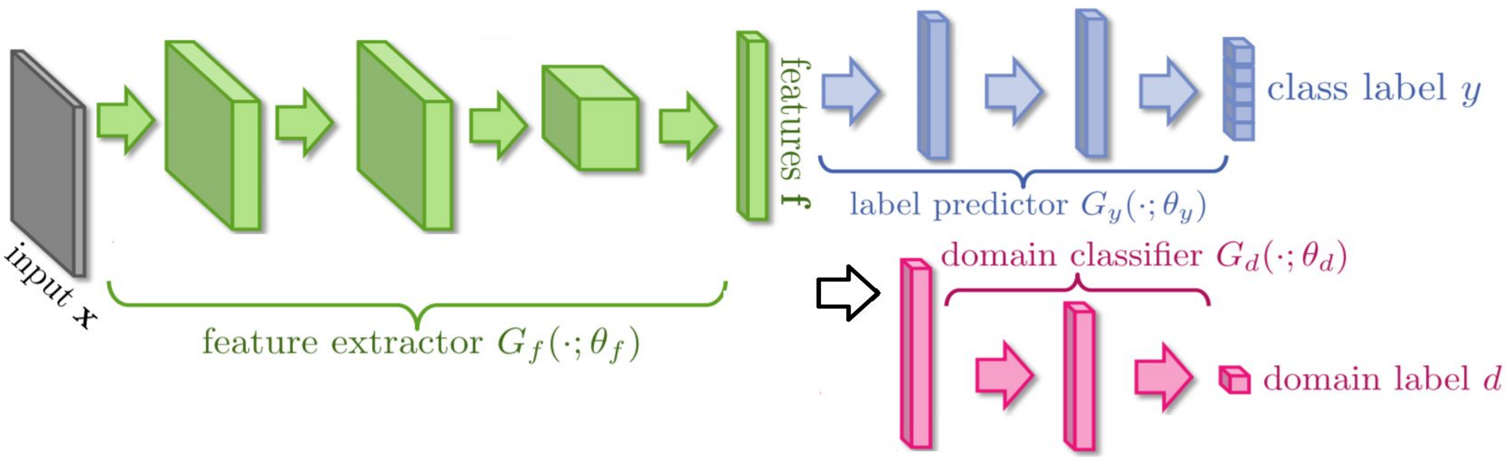
Proposed Solution

- θ_f denotes the parameters in feature extractor
- Given an image, feature extractor generates a deep feature for the image
- θ_f minimizes the loss the label classifier
- θ_f maximize the loss the domain classifier
- θ_d is updated as $\theta_f \longleftarrow \theta_f - \mu \left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right)$
- μ is the learning rate
- The parameter λ controls the trade-off between the two objectives



Proposed Solution

Consider an image from source domain

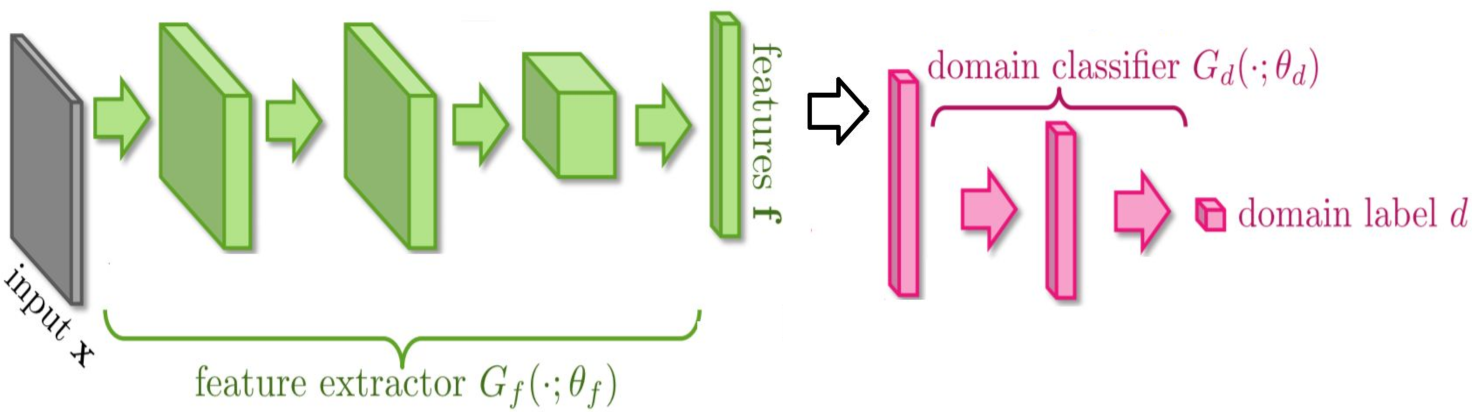


$$E(\theta_f,\theta_y,\theta_d) = \sum_{\substack{i=1..N\\ d_i=0}} L_y\left(\hat{y}_i, y_i\right) -$$

$$\lambda \sum_{i=1..N} L_d\left(\hat{d}_i, \overset{d_i}{\cancel{y_i}}\right) =$$

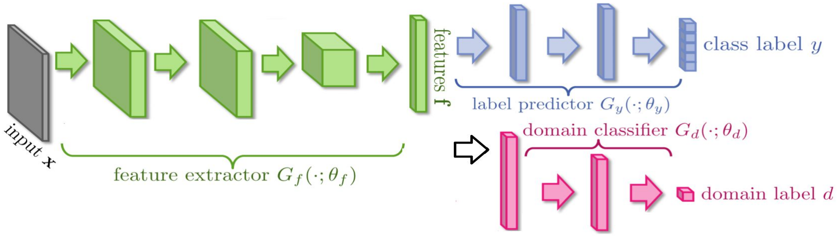
Proposed Solution

Consider an image from target domain



$$E(\theta_f, \theta_y, \theta_d) = \lambda \sum_{i=1..N} L_d\left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), \overset{d_i}{\cancel{y_i}}\right)$$

Proposed Solution



$$E(\theta_f, \theta_y, \theta_d) = \sum_{\substack{i=1..N \\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i\right) -$$

$$\lambda \sum_{i=1..N} L_d\left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), y_i\right) =$$

$$= \sum_{\substack{i=1..N \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d)$$

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d)$$

$$\hat{\theta}_d = \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)\,.$$

Proposed Solution

- At saddle point
  - θ_d minimizes domain classification loss
  - θ_y minimizes label prediction loss
  - θ_f minimizes label prediction loss and maximize domain classification loss

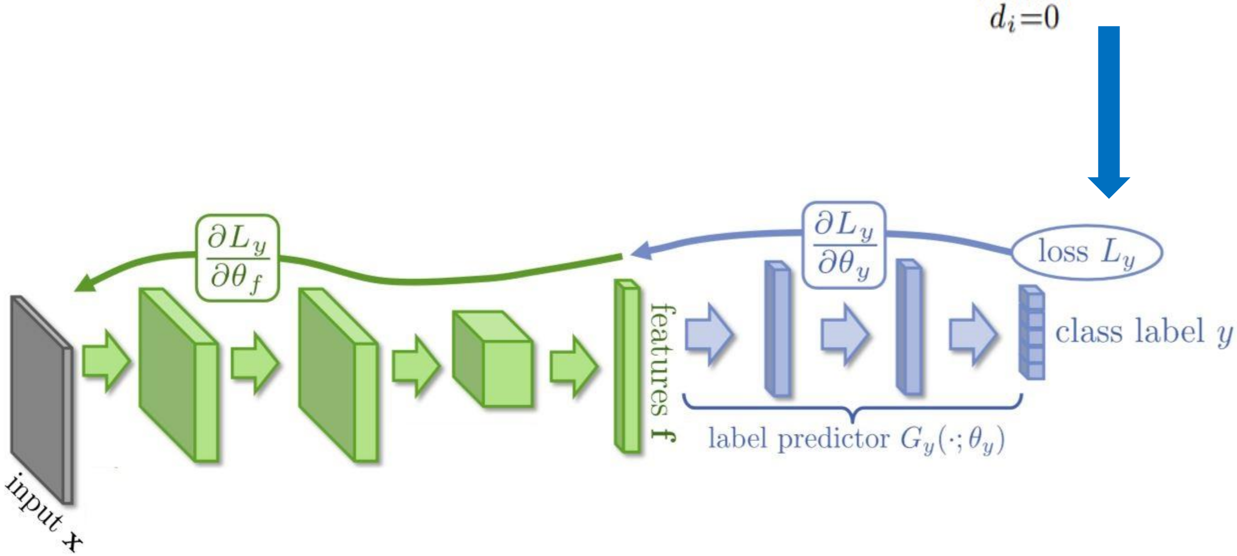
$$E(\theta_f, \theta_y, \theta_d) = \sum_{\substack{i=1..N \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d)$$

$$\begin{aligned}
\theta_f &\longleftarrow \theta_f - \mu \left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right) \\
\theta_y &\longleftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y} \\
\theta_d &\longleftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d}
\end{aligned}$$

Proposed Solution

- How to backpropagate the label classifier loss?
- Consider only the upper architecture
- This is typical backpropagation

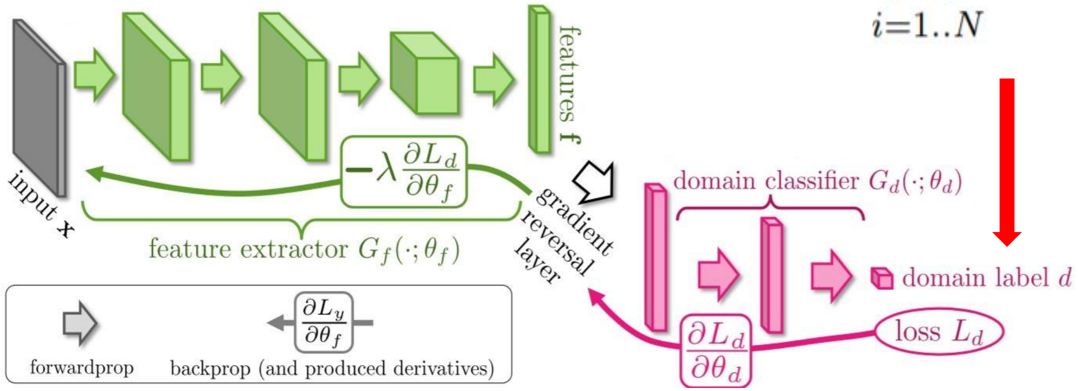
$$\sum_{\substack{i=1..N\\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i\right)$$



Proposed Solution

- How to backpropagate the domain classifier loss?
- Consider only the upper architecture
- Define gradient reversal layer (GRL)

$$\sum_{i=1..N} L_d^i(\theta_f, \theta_d)$$



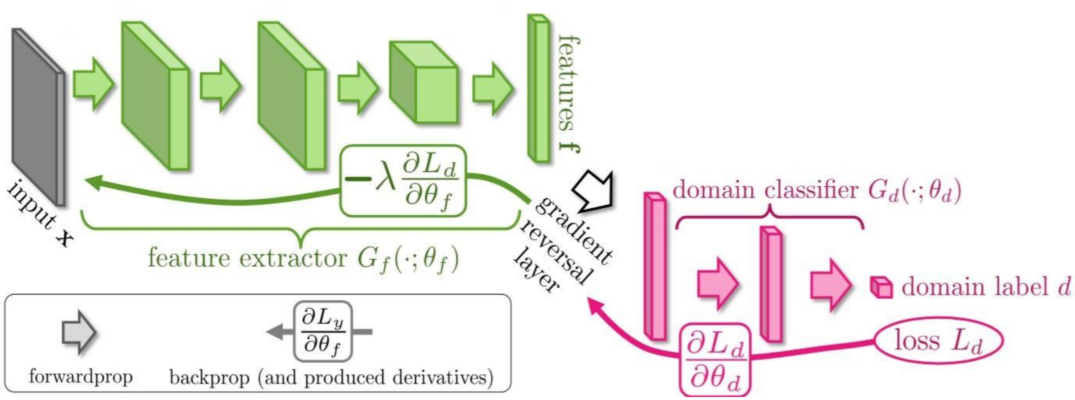
$$\theta_f \longleftarrow \theta_f + \lambda \frac{\partial L_d^i}{\partial \theta_f} \qquad \theta_d \longleftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d}$$

Proposed Solution

- Forward : GRL is an identity transformation
- Backward: GRL takes gradient from subsequent level, multiply by λ and pass it to previous layer
- Treat GRL as a pseudo function $R_\lambda(x)$

Forward $$R_\lambda(\mathbf{x}) = \mathbf{x}$$

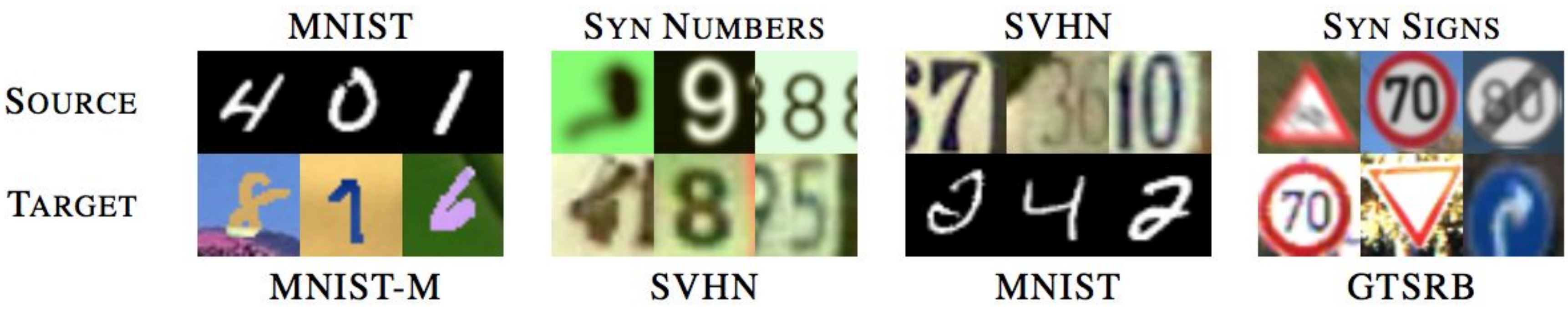
Backward $$\frac{dR_\lambda}{d\mathbf{x}} = -\lambda \mathbf{I}$$



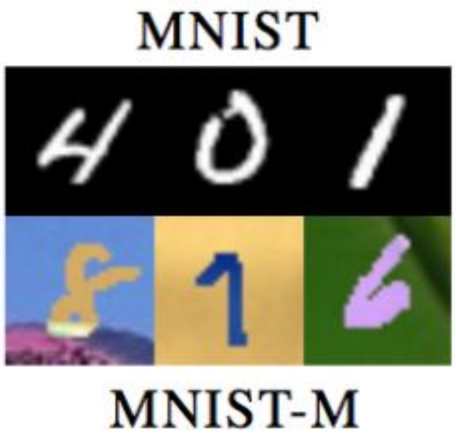
Proposed Solution

- After training, the label predictor can be used to predict labels for samples from either source or target domain
- Experiment results

Source & Target Datasets



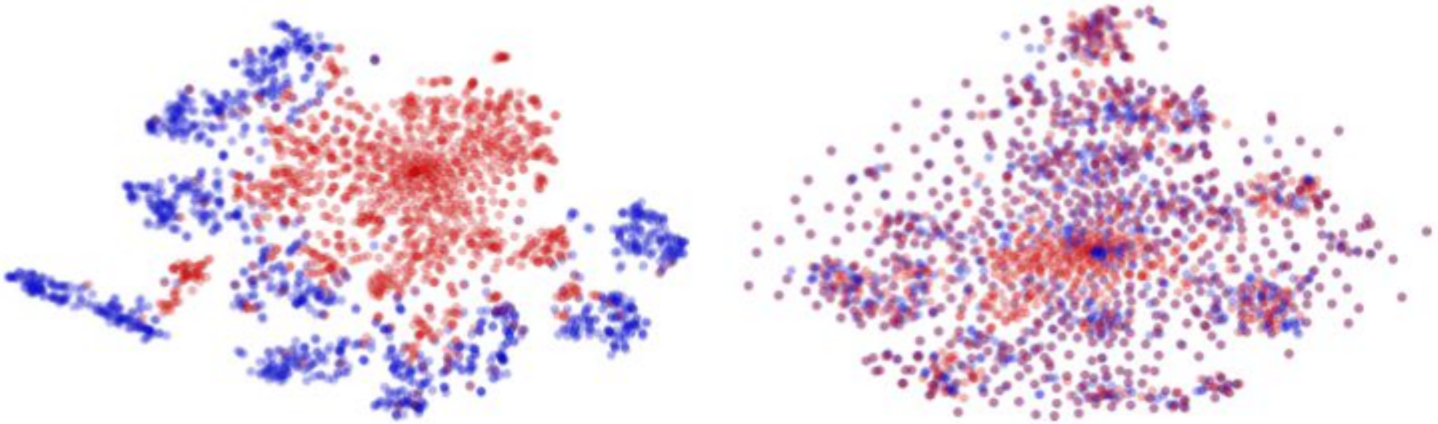
MNIST → MNIST-M



METHOD SOURCE / TARGET	MNIST / MNIST-M
SOURCE ONLY	.5749
SA (FERNANDO ET AL., 2013)	.6078 (7.9%)
PROPOSED APPROACH	**.8149** (57.9%)
TRAIN ON TARGET	.9891

MNIST → MNIST-M

MNIST → MNIST-M: top feature extractor layer



(a) Non-adapted (b) Adapted

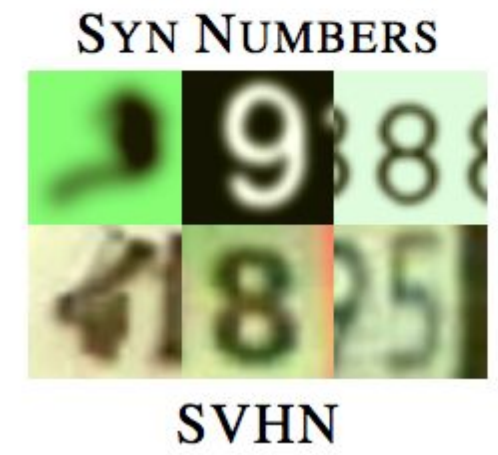
Synthetic numbers → SVHN

SYN NUMBERS

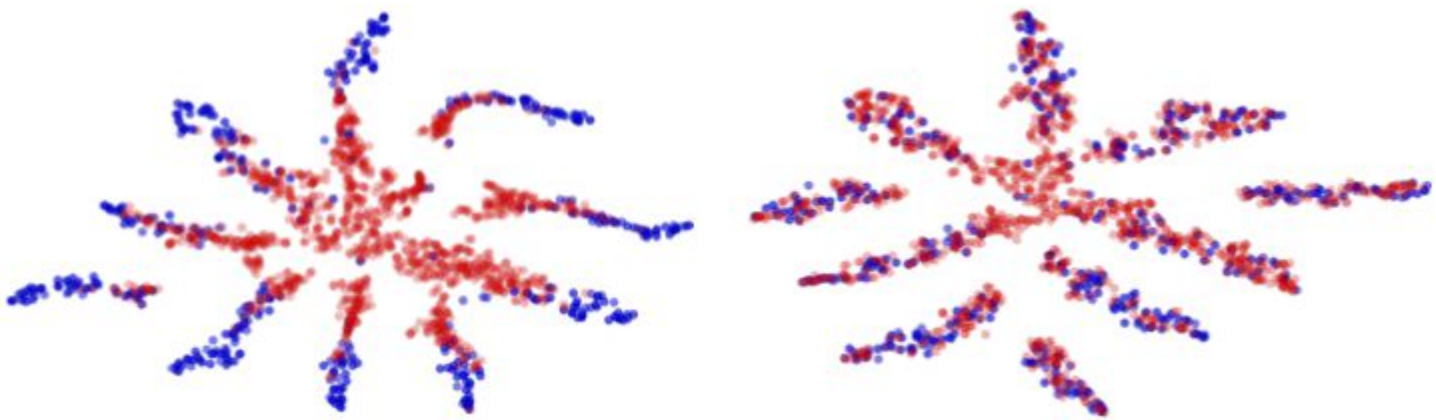
SVHN

METHOD (SOURCE / TARGET)	MNIST / MNIST-M	SYN NUMBERS / SVHN
SOURCE ONLY	.5749	.8665
SA (FERNANDO ET AL., 2013)	.6078 (7.9%)	.8672 (1.3%)
PROPOSED APPROACH	**.8149** (57.9%)	**.9048** (66.1%)
TRAIN ON TARGET	.9891	.9244

Synthetic numbers → SVHN



SYN NUMBERS → SVHN: last hidden layer of the label predictor



(a) Non-adapted (b) Adapted

MNIST ↔ SVHN

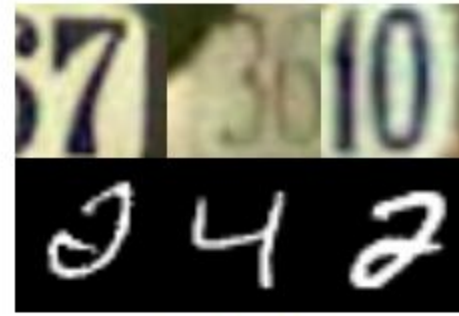
The two directions (MNIST → SVHN and SVHN → MNIST) are not equally difficult.

SVHN is more diverse, a model trained on SVHN is expected to be more generic and to perform reasonably on the MNIST dataset.

Unsupervised adaptation from MNIST to SVHN gives a failure example for this approach.

SVHN → MNIST

SVHN



MNIST

METHOD (SOURCE / TARGET)	MNIST / MNIST-M	SYN NUMBERS / SVHN	SVHN / MNIST
SOURCE ONLY	.5749	.8665	.5919
SA (FERNANDO ET AL., 2013)	.6078 (7.9%)	.8672 (1.3%)	.6157 (5.9%)
PROPOSED APPROACH	**.8149** (57.9%)	**.9048** (66.1%)	**.7107** (29.3%)
TRAIN ON TARGET	.9891	.9244	.9951

Synthetic Signs → GTSRB

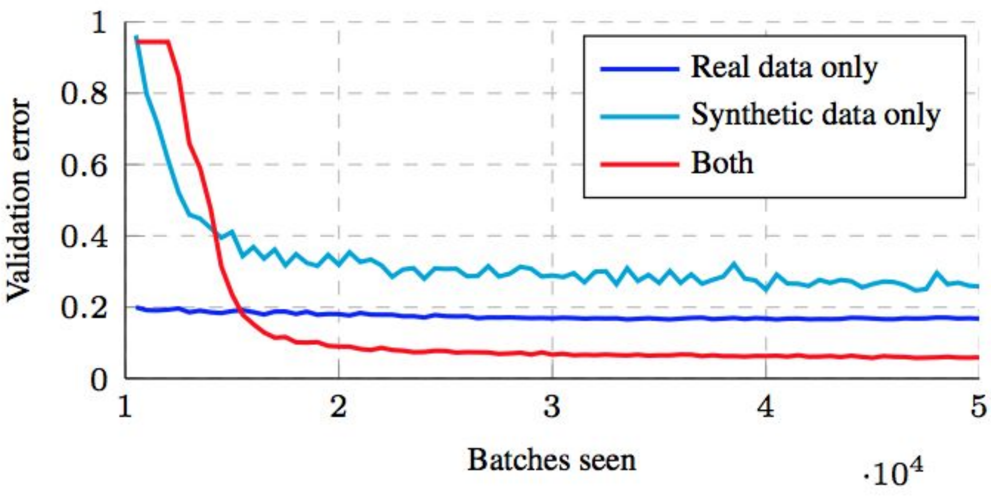
SYN SIGNS

GTSRB

METHOD	SOURCE / TARGET	MNIST / MNIST-M	SYN NUMBERS / SVHN	SVHN / MNIST	SYN SIGNS / GTSRB
SOURCE ONLY		.5749	.8665	.5919	.7400
SA (FERNANDO ET AL., 2013)		.6078 (7.9%)	.8672 (1.3%)	.6157 (5.9%)	.7635 (9.1%)
PROPOSED APPROACH		**.8149** (57.9%)	**.9048** (66.1%)	**.7107** (29.3%)	**.8866** (56.7%)
TRAIN ON TARGET		.9891	.9244	.9951	.9987

Synthetic Signs → GTSRB

This paper also evaluates the proposed algorithm for semi-supervised domain adaptation, i.e. when one is additionally provided with a small amount of labeled target data.



Office dataset

Method (Source / Target)	Amazon / Webcam	DSLR / Webcam	Webcam / DSLR
GFK(PLS, PCA) (Gong et al., 2012)	$.464 \pm .005$	$.613 \pm .004$	$.663 \pm .004$
SA (Fernando et al., 2013)	.450	.648	.699
DA-NBNN (Tommasi & Caputo, 2013)	$.528 \pm .037$	$.766 \pm .017$	$.762 \pm .025$
DLID (S. Chopra & Gopalan, 2013)	.519	.782	.899
DeCAF$_6$ Source Only (Donahue et al., 2014)	$.522 \pm .017$	$.915 \pm .015$	–
DaNN (Ghifary et al., 2014)	$.536 \pm .002$	$.712 \pm .000$	$.835 \pm .000$
DDC (Tzeng et al., 2014)	$.594 \pm .008$	$.925 \pm .003$	$.917 \pm .008$
Proposed Approach	$\mathbf{.673 \pm .017}$	$\mathbf{.940 \pm .008}$	$\mathbf{.937 \pm .010}$

Conclusions

- Proposed a new approach to unsupervised domain adaptation of deep feed-forward architectures;
- Unlike previous approaches, this approach is accomplished through standard backpropagation training;
- The approach is scalable, and can be implemented using any deep learning package.