

Unsupervised Domain Adaptation by Backpropagation

Index

1. Introduction
2. Related work
3. Deep Domain Adaptation
4. Experiments
5. Discussion

Introduction

- **domain adaptation (DA)**

training and test distributions 같지 않을 때, discriminative classifier/other predictor 를 학습하는 것

- domain adaptation 의 장점

- target domain 의 data 가 레이블이 지정되지 않았거나(unsupervised domain annotation)
- labeled samples 이 거의 없는 경우(semi-supervised domain adaptation)

→ a mapping between domains 을 학습할 수 있는 기능

- 기존의 논문과의 차이점

기존 방식은 parameter 가 고정된 feature representations 으로 domain adaptation 을 적용했지만, 이 논문은 [**domain adaptation + deep feature learning**] 을 하나의 training 과정 내에서 해결

- 목표

- source and the target domains 이 유사할 때, domain adaptation 을 learning representation 과정에 포함하여 최종 classification decisions 이 domains 변경에 대해 영향받지 않도록 이루어짐.

→ 획득한 feed-forward network 는 **two domains** 간의 이동에 방해받지 않고 **target domain** 에 적용 가능

Introduction

- (i) discriminativeness [차별성] + (ii) domaininvariance [도메인 불변] features 학습

- class labels 을 예측하고 training and at test time 모두 사용되는 label predictor
- training 과정에서 source and the target domains 구별하는 domain classifier

→ deep feature mapping 의 parameters 는 [label classifier의 loss ↓ + domain classifier 의 loss ↑]
domain 을 구별할 수 없도록 만듦.

- 구현

- backpropagation 이 가능한 기존 feed-forward architecture 에 domain adaptation 을 추가
- gradient 에 특정 음의 상수를 곱해서 backpropagation 수행

2. Related work

Related work

- **unsupervised domain adaptation**

source / target domains 의 feature distributions 일치시켜 domain adaptation 수행

Our approach,

feature space distributions 을 일치시키려 하지만,

기존의 방식처럼 source domain 에서 유효한 것만 샘플링하거나 source distribution을 target distribution 으로 매핑하는 feature transformation 을 찾는 것이 아니라 **feature representation** 자체를 변형함으로서 달성됨

→ distribution matching 의 중요한 점: (비)유사성을 측정하는 방식

two domains 사이의 feature distributions 불일치를 측정하고 최소화하여 distribution matching 을 진행

- **supervised domain adaptation**

target domain 의 labeled data → source domain 에서 trained network 를 fine-tune 할 때 사용

Our approach,

labeled target-domain data 는 필요 없지만 이러한 데이터를 사용할 수 있을 때 두 도메인을 쉽게 통합 가능

3. Deep Domain Adaptation

3.1 The model

dataset

$$S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \sim (\mathcal{D}_S)^n; \quad T = \{\mathbf{x}_i\}_{i=n+1}^N \sim (\mathcal{D}_T^X)^{n'},$$

L 개의 label 분류하는 classification 문제라고 가정

- X : input space
- Y : label space, $\{0, 1, \dots, L-1\}$ set of L possible labels
- $N = (n + n')$: total number of samples.

$\mathbf{x} \in X$: input space 의 data

$y \in Y$: label space 의 특정 레이블

$X \otimes Y$ 공간 상에서 2개의 다른 분포, unsupervised domain adaptation learning 이라면,

- \mathcal{D}_S : source domain, labeled source sample S
- \mathcal{D}_T : target domain, unlabeled target sample T
- d_i : i-th domain label(binary variable, **0 or 1**)

→ \mathbf{x}_i 가 source distribution ($\mathbf{x}_i \sim S(\mathbf{x})$ if $d_i=0$) 에서 오는지, target distribution ($\mathbf{x}_i \sim T(\mathbf{x})$ if $d_i=1$) 에서 오는지

source distribution ($d_i=0$) sample 의 해당 레이블 $y_i \in Y$ 는 training time 에 알려져 있지만, target domains 의 sample인 경우 training time 에 레이블을 알지 못하며 test time 에 레이블을 예측하려 함.

3.1 The model

The goal of the learning algorithm

$$R_{\mathcal{D}_T}(\eta) = \Pr_{(\mathbf{x}, y) \sim \mathcal{D}_T} \left(\eta(\mathbf{x}) \neq y \right),$$

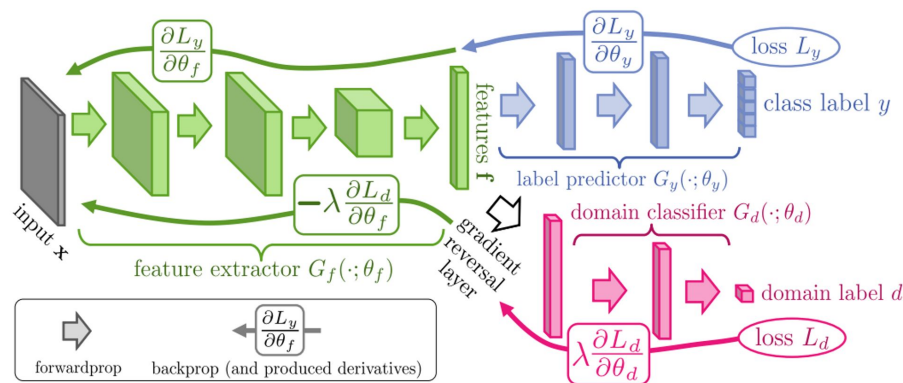
the labels of \mathcal{D}_T 에 대한 정보없이 a classifier $\eta : X \rightarrow Y$ with a low target risk 를 빌드

⇒ target domain 에 대한 risk function R 이 최소화되는 classifier η 빌드

3.1 The model

- architecture

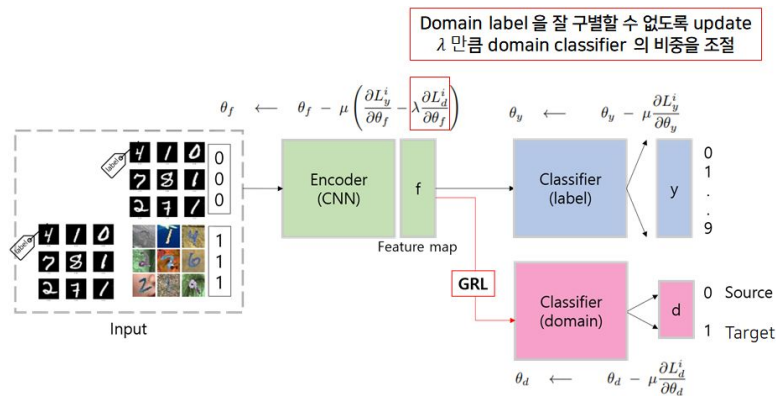
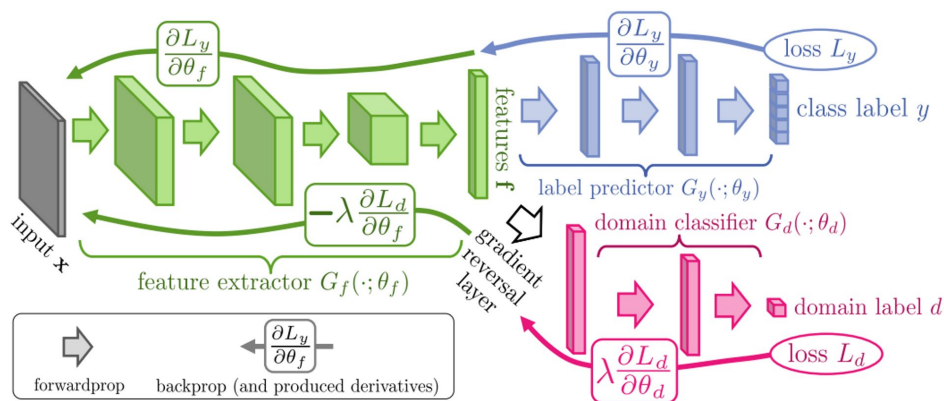
→ input x 에 대해 label $y \in Y$ 과 domain label $d \in \{0, 1\}$ 를 예측하는 deep feed-forward architecture 정의




- a standard feed-forward architecture

- green = a deep feature extractor (숫자로 이루어진 해당 데이터의 feature 추출)
 - input x 는 mapping G_f (**feature extractor**) 에 의해 D -dimensional feature vector $f \in \mathbb{R}^D$ 에 mapping
→ feature mapping 은 여러개의 feed-forward layers 포함할 수 있으며, 이 매핑의 all layers 의 parameters 벡터: θ_f , i.e. $f = G_f(x; \theta_f)$
- blue = a deep label predictor
 - feature vector f 는 mapping G_y (**label predictor**)에 의해 **label y** 에 mapping
→ 이 mapping의 parameter: θ_y

3.1 The model



Unsupervised domain adaptation

- red = a domain classifier
 -  = a gradient reversal layer (gradient 에 특정 음의 상수 λ 를 곱해서 backpropagation 수행)
- 2-2. feature vector f 는 G_d (domain classifier) 에 의해 domain label d 에 mapping
 \rightarrow 0 mapping의 parameter θ_d

Gradient reversal




- 기능: two domains 에 대한 feature distributions 이 유사하게 (domain classifier 가 구별할 수 없도록)
- 없다면? : 표준대로 학습진행
 \rightarrow label prediction loss (for source examples) + domain classification loss (for all samples) 최소화

3.1 The model

During the learning stage,

- 목표: **source part** 의 **label prediction loss** 최소화 + **features f** 를 **domain-invariant** 만듦.
 1. feature extractor + source domain 의 label predictor parameters
→ source domain samples 에 대한 empirical loss 를 최소화하기 위해 optimized 됨.
 2. 분포 $S(f) = \{G_f(x; \theta_f) | x \sim S(x)\}$ \rightleftharpoons 분포 $T(f) = \{G_f(x; \theta_f) | x \sim T(x)\}$

→ dissimilarity 의 측정하는 방법: **G_d (domain classifier) 의 loss 값 측정**

- A. domain-invariant features 을 얻기 위해 G_d (domain classifier) 의 loss 값  feature mapping 의 parameter θ_f 를 찾음. → 2 feature distributions 을 유사하게
- B. G_d (domain classifier) 의 loss 값  G_d 의 parameter θ_d 를 찾음.
- C. G_y (label predictor) 의 loss 값 

3.1 The model

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) &= \sum_{\substack{i=1..N \\ d_i=0}} L_y(G_y(G_f(\mathbf{x}_i; \theta_f); \theta_y), y_i) - \\ &\quad \lambda \sum_{i=1..N} L_d(G_d(G_f(\mathbf{x}_i; \theta_f); \theta_d), y_i) = \\ &= \sum_{\substack{i=1..N \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d) \end{aligned} \quad (1)$$

- $L_y(\cdot, \cdot)$: label prediction 의 loss (e.g. multinomial)
- $L_d(\cdot, \cdot)$: domain classification 의 loss (e.g. logistic)
- L_y^i and L_d^i : i-th training example 에서 평가된 loss functions
- parameter λ : learning 동안 features 를 형성하는 two objectives 사이의 균형 제어

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d) \quad (2)$$

$$\hat{\theta}_d = \arg \max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d). \quad (3)$$

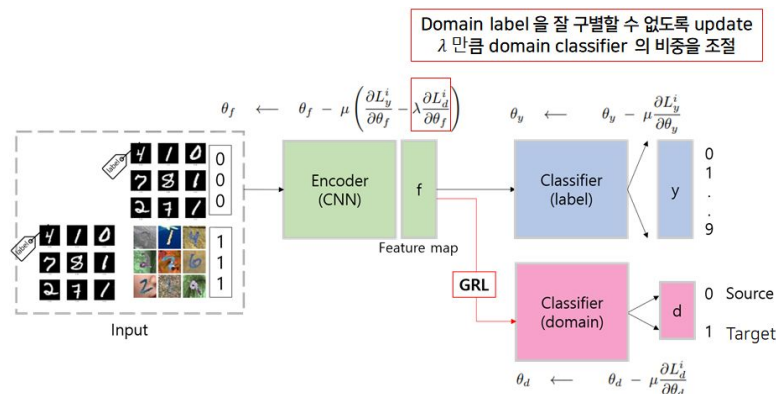
- G_d (domain classifier) 의 parameter θ_d
: domain classification loss ↓
→ - 부호와 함께 (1)에 들어가기 때문
- G_y (label predictor) 의 parameter θ_y
: label prediction loss ↓
- G_f (feature extractor) 의 parameter θ_f
: label prediction loss ↓ (features are discriminative) +
domain classification loss ↑ (features are domain-invariant)

3.2 Optimization with backpropagation

$$\theta_f \leftarrow \theta_f - \mu \left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right) \quad (4)$$

$$\theta_y \leftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y} \quad (5)$$

$$\theta_d \leftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d} \quad (6)$$



- μ : learning rate
- **gradient reversal layer (GRL)**
 - parameters x (backpropagation 의해 업데이트 되지 않는 meta-parameter λ 제외)
 - During propagation \rightarrow identity transform 수행
 - During backpropagation \rightarrow subsequent level gradient 에 $-\lambda$ 를 곱하고 preceding layer 에 전달
 - forwardprop (identity transform)
 - backprop (multiplying by a constant)
 - parameter update (nothing)
- backpropagation process 가 통과함에 따라 GRL downstream (i.e. L_d) GRL upstream 에 있는 layer parameters 인 (i.e. θ_f) 는 $-\lambda$ 로 곱해짐. $\frac{\partial L_d}{\partial \theta_f} \rightarrow -\lambda \frac{\partial L_d}{\partial \theta_f}$
- running SGD \rightarrow updates (4)-(6) 이 구현되고 안정적으로 (1) 이 수렴

3.2 Optimization with backpropagation

- gradient reversal layer → **pseudo-function** $R_\lambda(\mathbf{x})$ 로 정의 가능

$$R_\lambda(\mathbf{x}) = \mathbf{x} \quad (7)$$

$$\frac{dR_\lambda}{d\mathbf{x}} = -\lambda \mathbf{I} \quad (8)$$

○ \mathbf{I} : 항등 행렬

- $(\theta_f, \theta_y, \theta_d)$ objective pseudo-function

$$\begin{aligned} \tilde{E}(\theta_f, \theta_y, \theta_d) = & \sum_{\substack{i=1..N \\ d_i=0}} L_y (G_y(G_f(\mathbf{x}_i; \theta_f); \theta_y), y_i) + \\ & \sum_{i=1..N} L_d (G_d(R_\lambda(G_f(\mathbf{x}_i; \theta_f))); \theta_d), y_i) \end{aligned} \quad (9)$$

- Running updates (4)-(6) → (9) 에 대한 SGD 를 실행하는 것처럼 구현
- After the learning,
label predictor $y(\mathbf{x}) = G_y(G_f(\mathbf{x}; \theta_f); \theta_y)$ 를 사용해서 target domain samples 의 labels 예측 가능

3.3. Relation to H Δ H-distance

source / target distributions 간 거리 (Divergence) 를 정의하기 위해 H-divergence 사용

- **H-divergence** : two domains 이 얼마나 다른지 수치화한 값

$$d_{\mathcal{H}}(\mathcal{D}_S^X, \mathcal{D}_T^X) = 2 \sup_{\eta \in \mathcal{H}} \left| \Pr_{\mathbf{x} \sim \mathcal{D}_S^X} [\eta(\mathbf{x}) = 1] - \Pr_{\mathbf{x} \sim \mathcal{D}_T^X} [\eta(\mathbf{x}) = 1] \right|.$$

여기서 the hypothesis class \mathcal{H} 는 binary classifiers $\eta : X \rightarrow \{0, 1\}$ 들의 집합. $\eta \in \mathcal{H}$

- **input** = \mathcal{D}_S^X 와 \mathcal{D}_T^X
- **output** = 0 혹은 양수
- **sup=supremum**= 상계에 속하는 값 중 가장 작은 값
ex. $0 < x < 1$ 이라고 했을 때, $\sup = 1$
- 위 수식의 뜻

\mathcal{H} 라는 hypothesis class 안에 여러 η (classifier)들이 존재

→ 모든 η 에 대해 \mathcal{D}_S^X 와 \mathcal{D}_T^X 의 sample들을 넣고 위의 절대값 안의 수식을 계산해서 가장 큰 값이 무엇인지를 확인하면 그 값을 H-divergence라고 정의

3.3. Relation to H Δ H-distance

- empirical H-divergence

$$\hat{d}_{\mathcal{H}}(S, T) = 2 \left(1 - \min_{\eta \in \mathcal{H}} \left[\frac{1}{n} \sum_{i=1}^n I[\eta(\mathbf{x}_i)=0] + \frac{1}{n'} \sum_{i=n+1}^N I[\eta(\mathbf{x}_i)=1] \right] \right)$$

- $2(1-\min[0+0]) = 2$, domain distribution 구별 잘함
source domain 의 sample 이 들어갔을 때, 0이라고 예측할 확률 = 0 (1이라 예측할 확률=1)
target domain 의 sample 이 들어갔을 때, 1이라고 예측할 확률 = 0 (0이라 예측할 확률=1)
- $2(1-\min[1/2+1/2]) = 0$, domain distribution 구별 못함

⇒ domain adaption 을 잘 수행하기 위해 (= target risk function 을 작게 하려면)

💡 source domain 에서 classification 성능이 높고($\mathbf{R}_s(\eta)$ ↓) source / target domain 을 구별하지 못하는 ($\mathbf{d}_{\mathcal{H}}(\mathbf{S}, \mathbf{T})$ ↓) 모델 구축

3. Experiments

3. Experiments

popular image datasets 에 대한 광범위한 평가를 수행함 (+ modified datasets)

- large-scale datasets of small-images
- OFFICE datasets (Saenko et al., 2010)

METHOD	SOURCE	MNIST	SYN NUMBERS	SVHN	SYN SIGNS
	TARGET	MNIST-M	SVHN	MNIST	GTSRB
SOURCE ONLY		.5225	.8674	.5490	.7900
SA (FERNANDO ET AL., 2013)		.5690 (4.1%)	.8644 (−5.5%)	.5932 (9.9%)	.8165 (12.7%)
PROPOSED APPROACH		.7666 (52.9%)	.9109 (79.7%)	.7385 (42.6%)	.8865 (46.4%)
TRAIN ON TARGET		.9596	.9220	.9942	.9980

baselines

1. **Source-only model** : target domain 에 대한 고려 없이 training 됨.
2. **Train-on-target model** : target domain 의 class 가 revealed 된 채로 training 됨. => DA method 의 upper bound
3. unsupervised DA method based on subspace alignment (**SA**) (Fernando et al., 2013)

3. Experiments

CNN architectures

- 2,3 개의 Convolutional Layer 로 feature extractor 를 구성함
- Domain adaptator \Rightarrow 3개의 fully connected layer 로 구성 ($x \rightarrow 1024 \rightarrow 1024 \rightarrow 2$)
- Loss function
 - L_y : Logistic regression loss
 - L_d : binomial cross-entropy

CNN training procedure

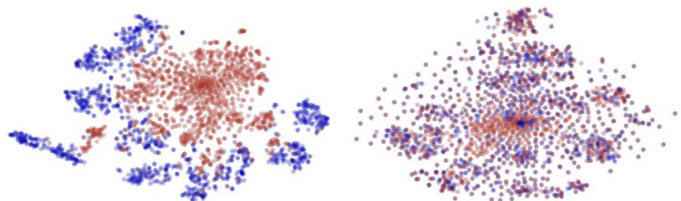
- **batch size** : 128
 - 각 batch 의 절반은 known label 한 source domain 으로부터, 나머지 절반은 unknown label 한 target domain 으로 한다.
- **image 전처리** : mean subtraction
- **adaptation factor** 람다 사용 \Rightarrow 학습 과정에서 domain classifier 의 noise 를 줄이기 위
$$\lambda_p = \frac{2}{1 + \exp(-\gamma \cdot p)} - 1,$$
 - 고정된 값이 아닌, 0 에서 1 로 점진적으로 변하는 값을 사용

3. Experiments

Visualization

- t-SNE projection 사용 \Rightarrow feature distribution 을 시각화하기 위해. (domain 을 색깔로 구별함)

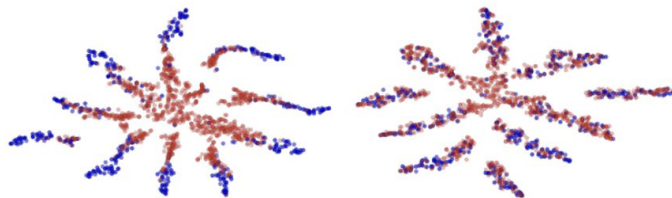
MNIST \rightarrow MNIST-M: top feature extractor layer



(a) Non-adapted

(b) Adapted

SYN NUMBERS \rightarrow SVHN: last hidden layer of the label predictor



(a) Non-adapted

(b) Adapted

- 파란색점 : source domain 의 sample
- 빨간색점 : target domain 의 sample

Results

source data set 으로 학습하고, 상당한 shift 가 있는 target data set 에 대해서 test 한다.

METHOD	SOURCE	MNIST	SYN NUMBERS	SVHN	SYN SIGNS
	TARGET	MNIST-M	SVHN	MNIST	GTSRB
SOURCE ONLY		.5225	.8674	.5490	.7900
SA (FERNANDO ET AL., 2013)		.5690 (4.1%)	.8644 (−5.5%)	.5932 (9.9%)	.8165 (12.7%)
PROPOSED APPROACH		.7666 (52.9%)	.9109 (79.7%)	.7385 (42.6%)	.8865 (46.4%)
TRAIN ON TARGET		.9596	.9220	.9942	.9980

1. MNIST → MNIST-M
2. SYN NUM → SVHN
3. SVHN → MNIST
4. SYN SIGNS → GTSRB

Results

1. MNIST \Rightarrow MNIST-M

- MNIST-M : BSDS500 의 컬러사진에서 random 하게 추출한 patch 와 digits 을 섞음. $I_{ijk}^{out} = |I_{ijk}^1 - I_{ijk}^2|$
 - image I_1, I_2 / i, j 는 픽셀의 좌표 / k 는 채널의 인덱스
 - output sample 은 사진에서 패치를 가져와서 숫자의 픽셀에 대응하는 위치에서 픽셀을 반전시킴으로써 생성된다.
- MNIST (source data set) 으로 훈련시킨 모델의 관점에서는 MNIST-M 데이터셋은 source 와 충분히 구별되는 target dataset 임.



METHOD	SOURCE	MNIST
	TARGET	MNIST-M
SOURCE ONLY		.5225
SA (FERNANDO ET AL., 2013)		.5690 (4.1%)
PROPOSED APPROACH		.7666 (52.9%)
TRAIN ON TARGET		.9596

Results

2. SYN NUMBERS ⇒ SVHN

합성 데이터에 대한 일반적인 시나리오와 **real data**에 대한 **testing**을 위해

synthetic digit을 source로 하고 target domain는 SVHN으로 한다.

- SYN NUM : Synthetic number ⇒ windowsTM font로 자체 생성한 50만개의 이미지로 구성됨
 - 한자리, 두자리, 세자리 숫자
 - 위치, 방향, 배경색, 스트로크 색, blur를 다양하게 변경할 수 있음.
- SVHN : Street-View House Number ⇒ 길거리에서 보이는 숫자. 라는 의미(실제 **real world** 숫자)
 - 두 데이터는 구별됨. 가장 큰 차이점은 SVHN 배경의 ‘구조화된 혼란스러움’이다.



METHOD	SOURCE	SYN NUMBERS
	TARGET	SVHN
SOURCE ONLY		.8674
SA (FERNANDO ET AL., 2013)		.8644 (−5.5%)
PROPOSED APPROACH		.9109 (79.7%)
TRAIN ON TARGET		.9220

source data로만 학습시키는 것과 target data로만 학습시키는 것의 격차의 80%(79.7%)를 cover함.

Results

3. SVHN \Rightarrow MNIST

source 와 target domain 의 차이를 더 크게 하기 위해 한 눈에 봐도 다른 SVHN 과 MNIST 에 대해 test 한다.

- adaptation 이 없어도 SVHN 의 training 은 어렵다. \Rightarrow 150 epoch 동안 classification error 가 높게 나옴.
 \Rightarrow **poor local minimum** 으로 학습이 끝나지 않게 하기 위해서 **learning rate annealing** 을 사용하지 않는다.
- SVHN 이 학습하기는 더 어렵지만, MNIST 보다 더 다양한 dataset 이므로 모델의 훈련이 더 일반적으로 될 것이고, MNIST 에 대해 더 합리적인 수행을 하게 될 것이다.
- MNIST \rightarrow SVHN 으로는 adaptation 을 통해서 성능을 향상시키는 것이 안 됨.

METHOD	SOURCE	SVHN
	TARGET	MNIST
SOURCE ONLY		.5490
SA (FERNANDO ET AL., 2013)		.5932 (9.9%)
PROPOSED APPROACH		.7385 (42.6%)
TRAIN ON TARGET		.9942



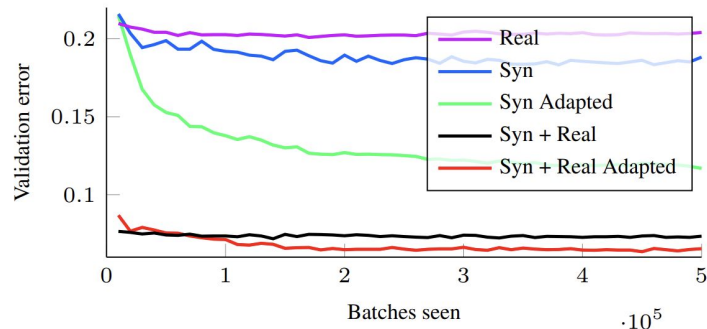
Results

4. SYN SIGNS \Rightarrow GTSRB

앞선 실험과 setting 이 유사함.

- but, class 개수가 43개로 앞의 실험(10개) 보다 상당히 많다 \rightarrow feature 의 distribution 이 앞 실험보다 좀 더 복잡하다.
- SYN SIGNS : 다양한 image condition 을 보여주는 합성된 10만개의 image
- target domain GTSRB \rightarrow 31,367 random training samples
- 추가 실험 : semi-supervised learning
 - 클래스당 10개의 labeled sample 추가

METHOD	SOURCE	SYN SIGNS
	TARGET	GTSRB
SOURCE ONLY		.7900
SA (FERNANDO ET AL., 2013)		.8165 (12.7%)
PROPOSED APPROACH		.8865 (46.4%)
TRAIN ON TARGET		.9980



Results

4. OFFICE Dataset

- OFFICE Dataset : Amazon, DSLR, Webcam 의 모음

METHOD	SOURCE	AMAZON	DSLR	WEBCAM
	TARGET	WEBCAM	WEBCAM	DSLR
GFK(PLS, PCA) (GONG ET AL., 2012)		.214	.691	.650
SA* (FERNANDO ET AL., 2013)		.450	.648	.699
DLID (S. CHOPRA & GOPALAN, 2013)		.519	.782	.899
DDC (TZENG ET AL., 2014)		.605	.948	.985
DAN (LONG & WANG, 2015)		.645	.952	.986
SOURCE ONLY		.642	.961	.978
PROPOSED APPROACH		.730	.964	.992

가장 까다로운 **AMAZON** → **Webcam** 실험에서도 이전
방법들보다 좋은 성능을 보여줌

5. Discussion

5. Discussion

1. **deep feed-forward architectures** 의 **unsupervised domain adaptation** 에 대한 새로운 접근 방식을 제안함.

⇒ source domain 의 annotated data(labeled data) 와 target domain 의 unannotated data(unlabeled data) 으로 training 하는 방식

⇒ 두 도메인의 feature 분포를 standard backpropagation training 을 통해 정렬함으로써 달성됨.

2. 이 방식은 **scalable(확장 가능)** 하며, **deep learning package** 를 활용하여 구현할 수 있다.

⇒ 이를 위해 Gradient Reversal layer 에 대한 소스코드를 공개함.