Adversarial Instance Re-weighting for Unsupervised Domain Adaptation

Itzel Belderbos

29 Oct, 2021

DKE supervisors:
Drs. Mirela Popa
Dr. Christof Seiler

External supervisor:
Dr. Fabian Jansen

Maastricht University

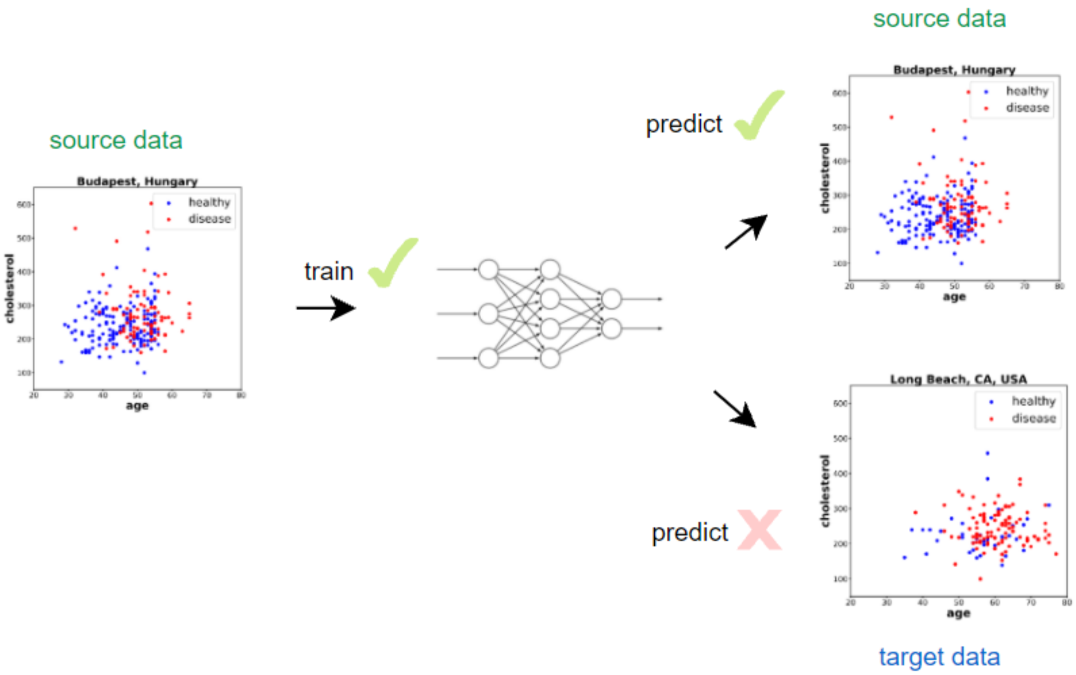
Contents

Problem statement

- **Wanted:**
    - Train model on *source* dataset, test on *target* dataset
- **Problem:**
    - Target dataset should have same distribution!

→ Labels target dataset not available

→ Data annotation is expensive



Examples

- Computer vision
    - From one weather condition to another
- Sentiment analysis
    - From book reviews to DVD reviews
- Speech recognition
    - From one speaker to another speaker

→ Focus in this work on **images**

Domain adaptation

- Domain: a distribution within the feature space
- This work focuses to two types of DA:

Covariate shift

$$
\begin{aligned}
\mathcal{X}^s &= \mathcal{X}^t \\
P(Y|X)^s &= P(Y|X)^t \\
P(X)^s &\neq P(X)^t
\end{aligned}
$$

Prior probability shift

$$
\begin{aligned}
\mathcal{X}^s &= \mathcal{X}^t \\
P(X|Y)^s &= P(X|Y)^t \\
P(Y)^s &\neq P(Y)^t
\end{aligned}
$$

Research goal

- Introduce *weight network* to transform distribution sample-wise

→ **Goal**

Investigate to which extent this weight network is able to benefit the domain adaptation, particularly for covariate shift

Research questions

1. Can we use non-generative adversarial networks to improve covariate shift adaptation?
   a. How can we train a weight function, which converts source into target distribution?
   b. What is the effect of the critic?

1. How can we measure the performance of the reconstructed target distribution and the adversarial network?

1. Are there other ways than a weight function to express source data into target data?

Methodology for proof-of-concept

The Wasserstein GAN vs our network

- Wasserstein GAN
  - Adversarial game between critic & generator

$$\min_g \max_c \mathbb{E}_{x\sim p_{data}(x)} [c(x)] - \mathbb{E}_{\sim p_z(z)} \boxed{[c(g(z))]}$$

- Our weighted network
  - Adversarial game between critic & weight

$$\min_w \max_c \mathbb{E}_{x\sim s(x)} \boxed{[c(x)w(x)]} - \mathbb{E}_{x\sim t(x)} [c(x)]$$

Experiments

- **Experiment 1:** One-dimensional Gaussians
- **Experiment 2:** MNIST subset & class imbalance
- **Experiment 3:** MNIST subset & covariate shift

Evaluation metric

→ **Binary domain classifier** to evaluate *domain invariance*

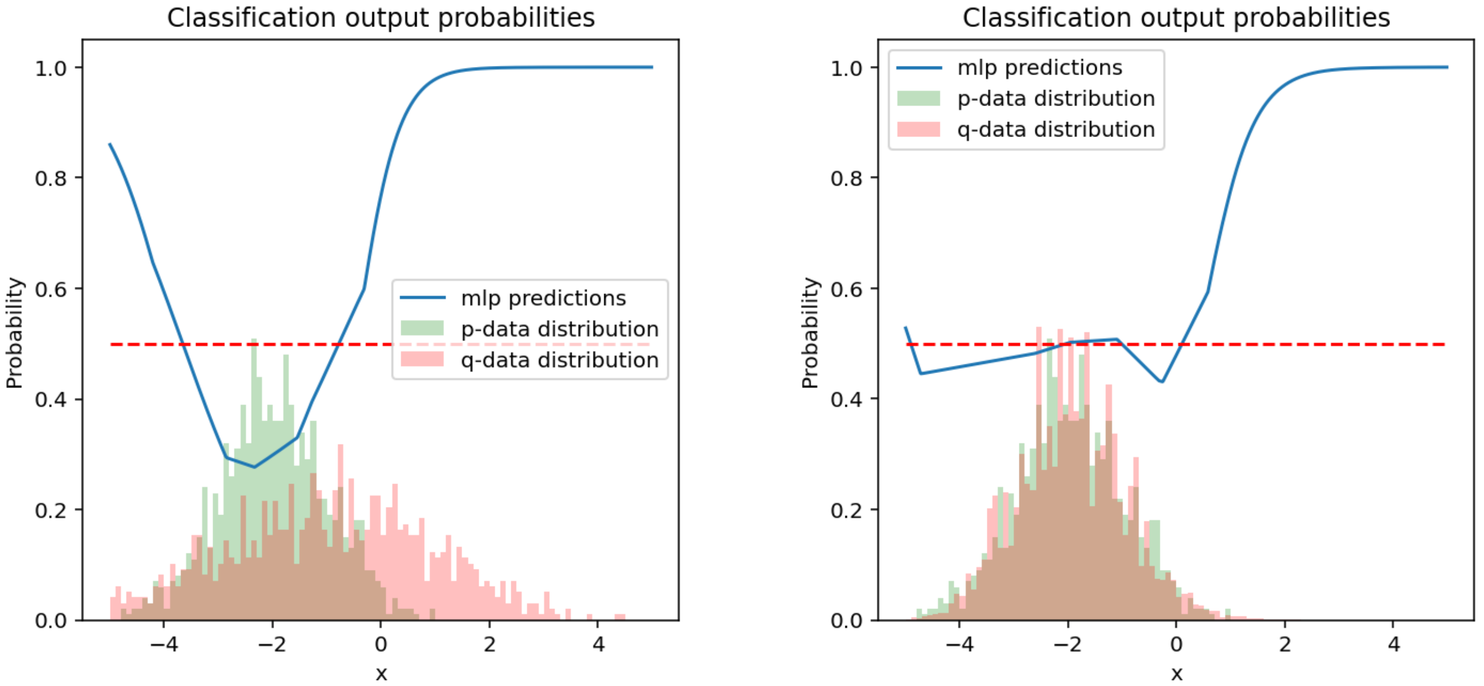
- Combine source and target dataset into 1 dataset
- Label target as 0, source as 1
- Check whether classifier can see difference
- Ideal situation: accuracy of 50%

Results for proof-of-concept

Experiment 1 - Results

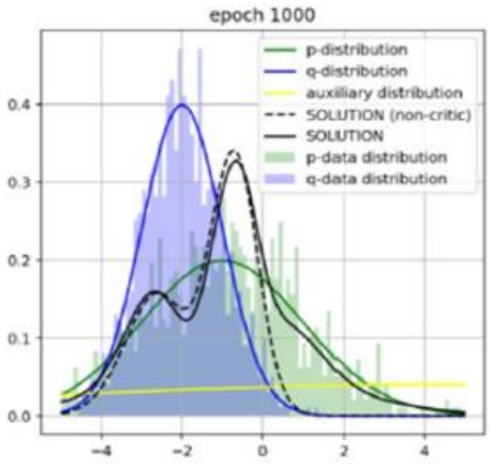
- Weighted distribution: binary domain classifier cannot see difference

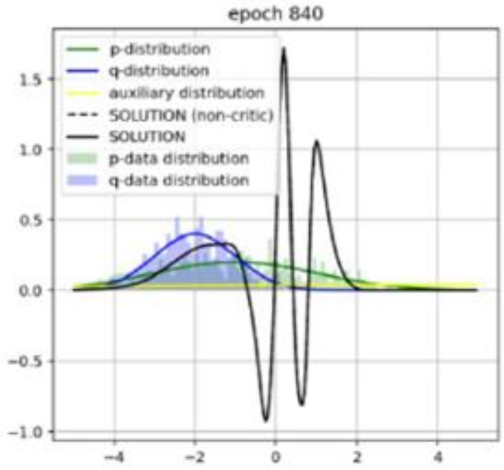
green = target
red = source



Experiment 1 - Limitations

- Target dataset cannot contain values which are not present is source dataset
    - Cannot re-weight values which do not exist





Experiment 2 - Prior probability shift

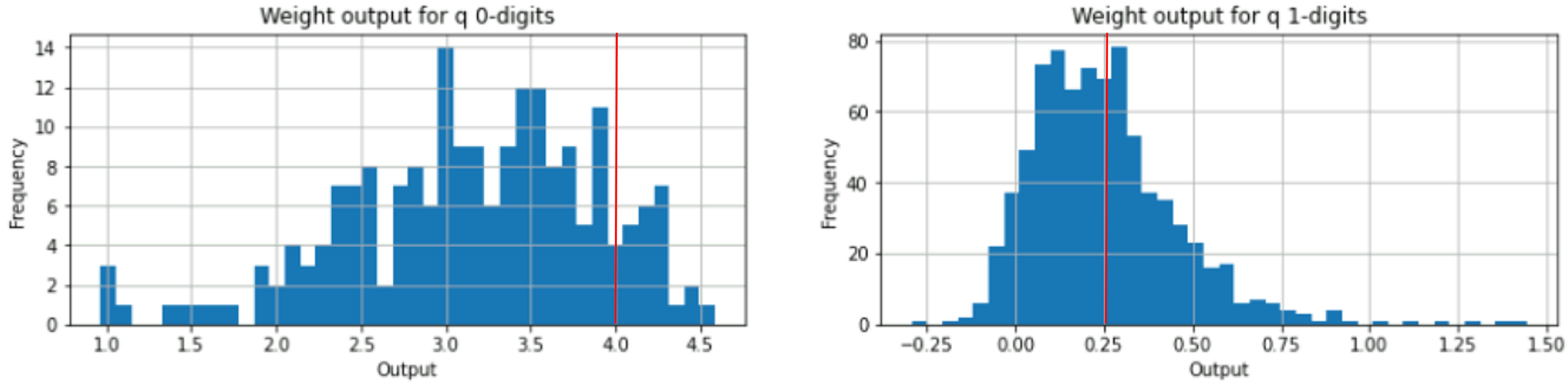
- 1000 samples; classes 0 and 1
- Imbalance between 2 classes

k	$P(Y=k)^s$	$P(Y=k)^t$	$\mathbb{E}[w(x)]$
0	0.20	0.80	4
1	0.80	0.20	$\frac{1}{4}$

Experiment 2 - Results

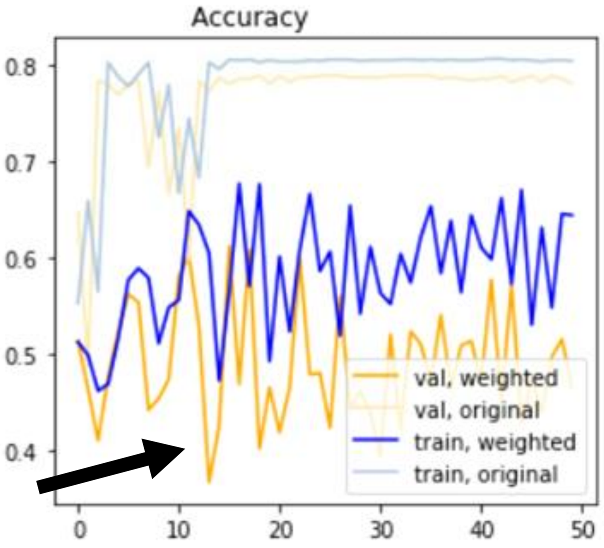
- Histogram of weight outputs

k	$P(Y=k)^s$	$P(Y=k)^t$	$\mathbb{E}[w(x)]$
0	0.20	0.80	4
1	0.80	0.20	$\frac{1}{4}$



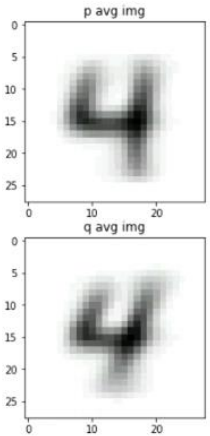
Experiment 2 - Results

- Binary domain classifier
  - **x-axis:** epochs
  - **y-axis:** accuracy



Experiment 3 - Covariate shift

- Source/target datasets consist of 10,000 samples with class 0-4
- Use KMeans to obtain 2 **subclasses** per class
- Create imbalance between 2 subclasses



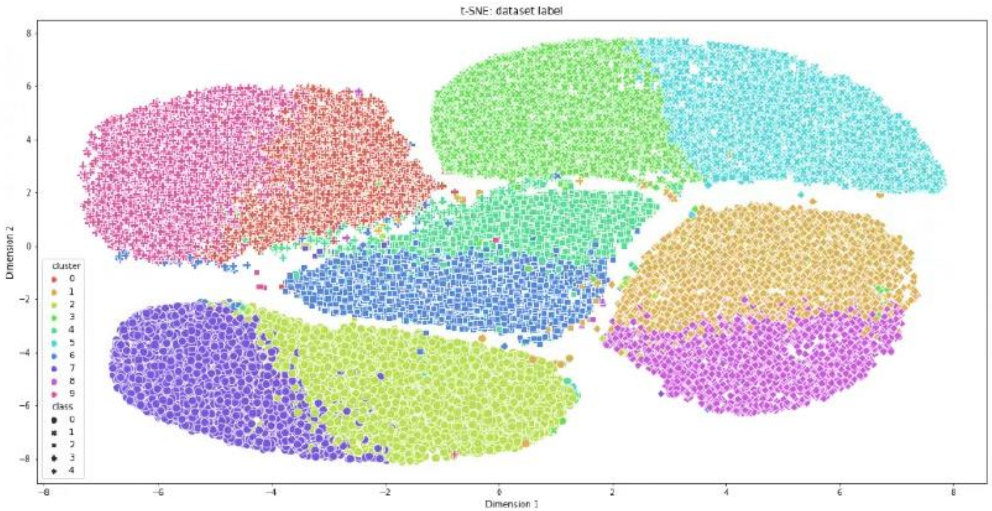
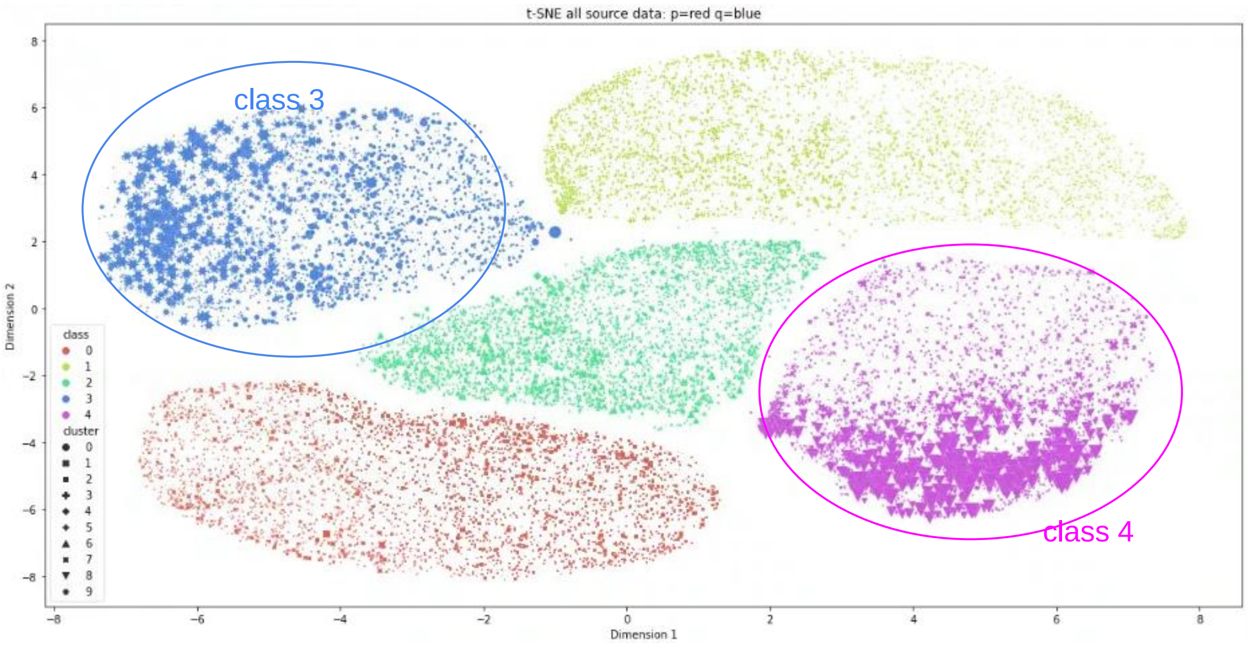


Figure 3.17: t-SNE of subset with 5 classes

Experiment 3 - Results

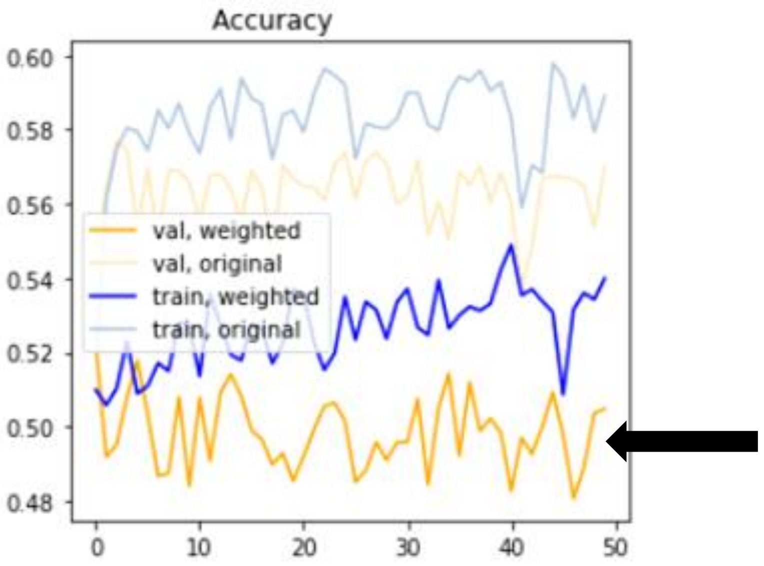


class	cluster	expected weight	actual weight
0	2	1	0.97
0	7	1	0.98
1	3	1	0.98
1	5	1	0.99
2	4	1	0.90
2	6	1	0.99
3	0	0.33	0.55
3	9	3	1.92
4	1	0.33	0.56
4	8	3	2.03

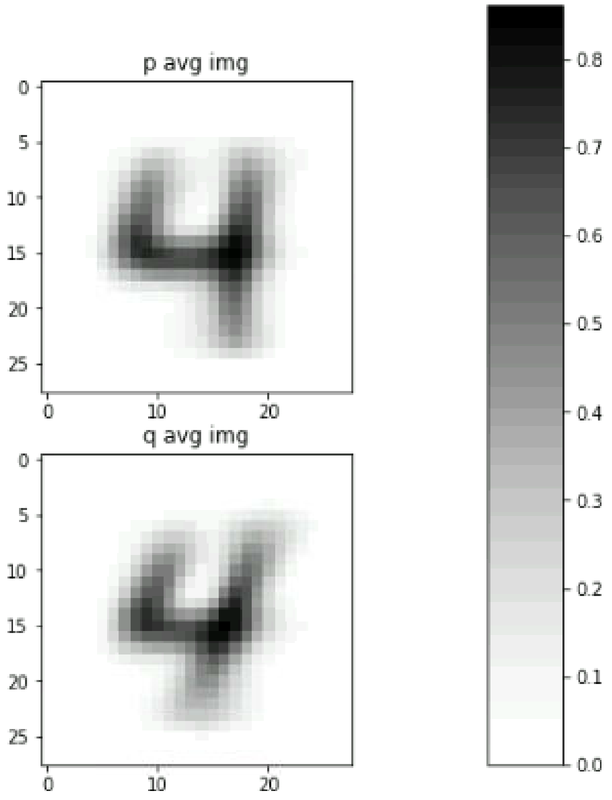
Table 3.3: Expected weight vs actual weight per cluster

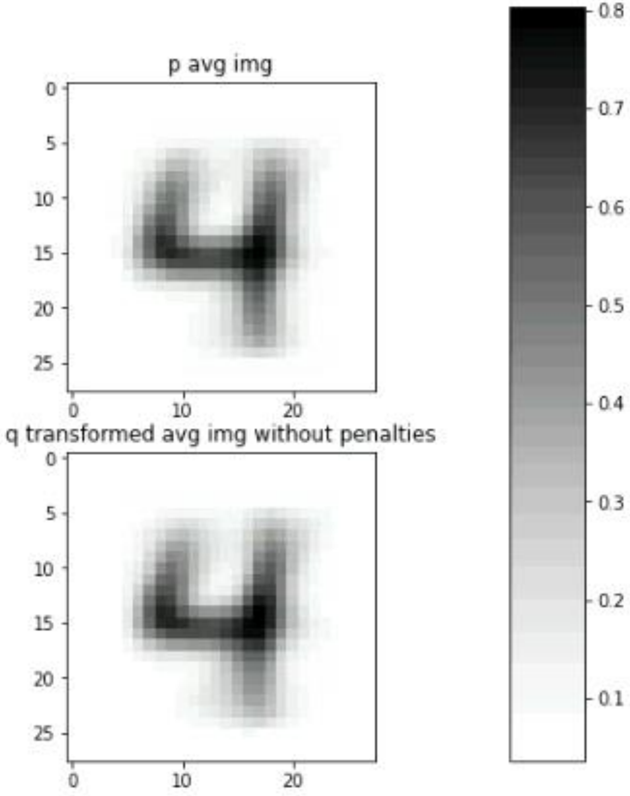
Experiment 3 - Results

- Binary domain classifier



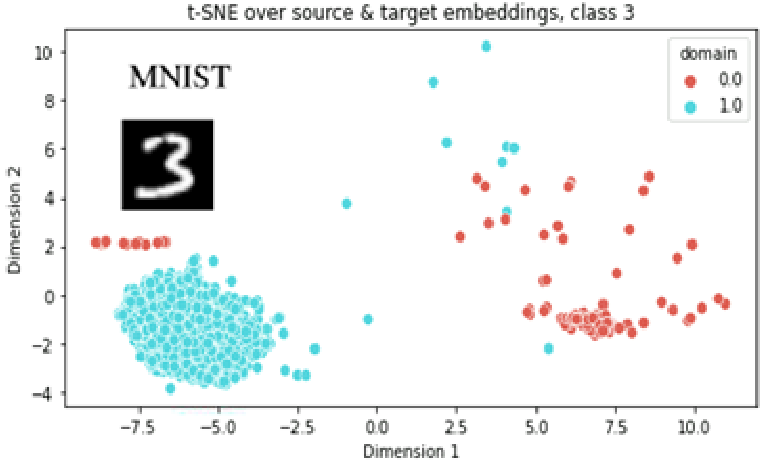
Experiment 3 - Results

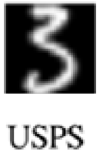


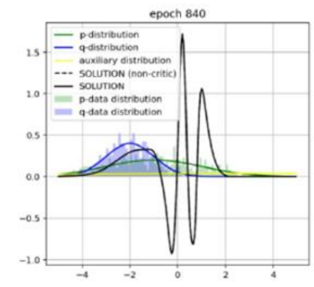


Methodology for state-of-the-art comparison

Large domain shift: different features



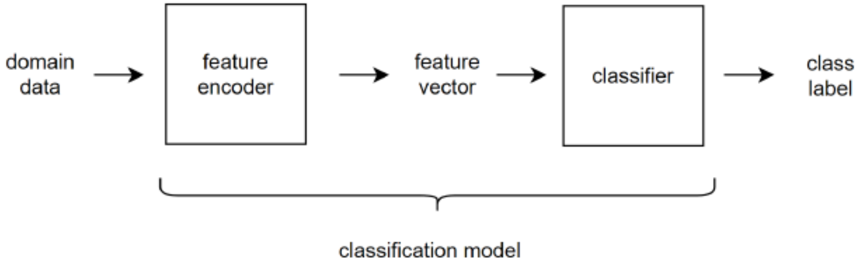




What to do when **target features are not present is source data**?

Combination with feature-based domain adaptation

- Learn a domain-invariant feature embedding with a **feature encoder**



$$P(X^s) \neq P(X^t)$$

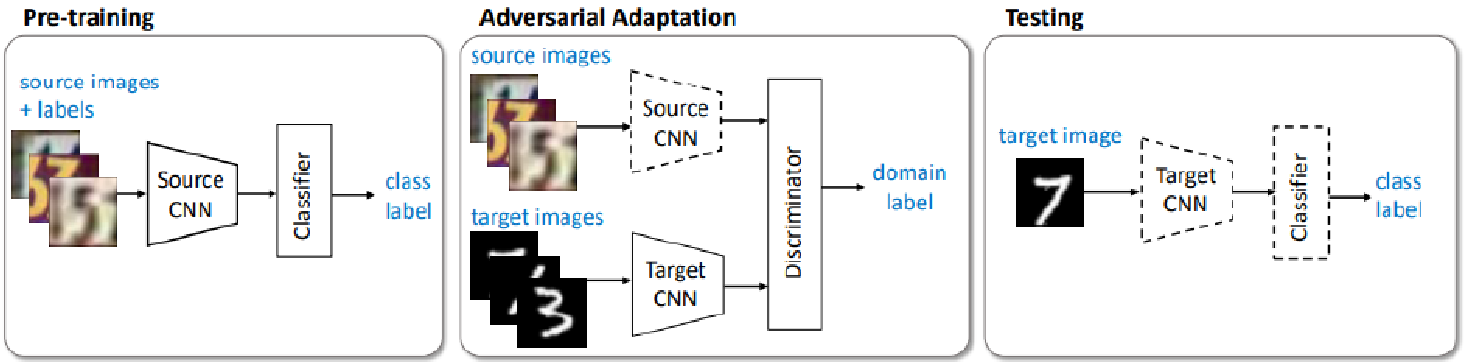
$$P(\tau(X^s)) = P(\tau(X^t))$$

Baseline models

- Adversarial Discriminative Domain Adaptation (ADDA)
- Wasserstein Guided Respresentation Learning framework (WDGRL)
    - Adversarial game between **discriminator/critic** & **feature encoder**

ADDA

- Works like a GAN, instead of *fake image* it learns a *fake feature representation*
- Binary cross-entropy loss
- 2 separate encoders for source and target (*asymmetric*)



ADDA

- Works like a GAN, instead of *fake image* it learns a *fake feature representation*
- Binary cross-entropy loss
- 2 separate encoders for source and target (*asymmetric*)

discriminator	$\min_{d} L^{d} = -\mathbb{E}[\log d\boxed{(\tau^{s}(x^{s}))]} - \mathbb{E}[\log(1 - d\boxed{(\tau^{t}(x^{t}))]}$
target encoder	$\min_{\tau^{t}} L^{\tau_t} = -\mathbb{E}[\log d\boxed{(\tau^{t}(x^{t}))]}$

WDGRL

- Uses Wasserstein distance instead of BCE loss
- 1 encoder for both source and target (*symmetric*)
- Adds classification loss in feature loss
  - Feature loss = Wasserstein loss + source classification loss

domain invariance

class segregation

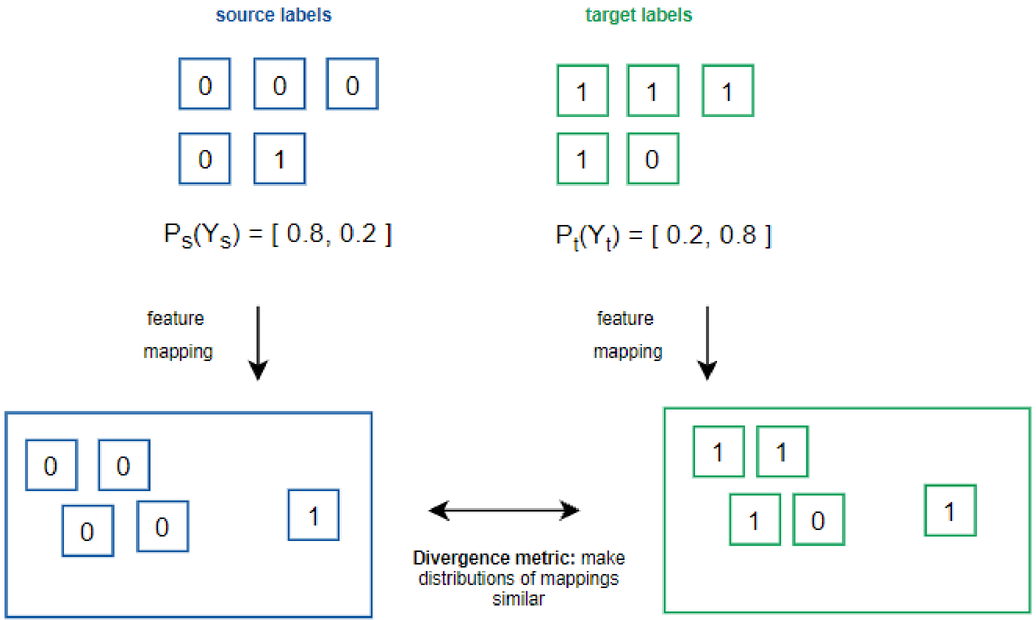
WDGRL

- Uses Wasserstein distance instead of BCE loss
- 1 encoder for both source and target (*symmetric*)
- Adds classification loss in feature loss
  - Feature loss = Wasserstein loss + source classification loss

critic	$\min_{d} L^{d} = \underbrace{\frac{1}{T}\sum_{x\in X^{t}} d(\boxed{\tau^{s,t}(x^{t})}) - \frac{1}{S}\sum_{x\in X^{s}} d(\boxed{\tau^{s,t}(x^{s})})}_{\text{Wasserstein loss}} + \nu \underbrace{\frac{1}{E}\sum_{x\in X^{s}} (\Vert \nabla_{e} d \Vert_{2} - 1)^{2}}_{\text{gradient penalty}}$
feature encoder	$\min_{\tau^{s,t},c} L^{ce,wdgrl} L^{\tau} = \underset{\text{balancing parameter}}{\lambda} \underbrace{\left[\frac{1}{S}\sum_{x\in X^{s}} d(\boxed{\tau^{s,t}(x^{s})}) - \frac{1}{T}\sum_{x\in X^{t}} d(\boxed{\tau^{s,t}(x^{t})})\right]}_{\text{Wasserstein loss}} + \underbrace{L^{ce,wdgrl}}_{\text{classification loss}}$

Introduction of weight network

- Regular domain adaptation methods assume class balance
- Mapping between target data and representations might go wrong



Proposed method

- Addition of weight
- Combination of ADDA and WDGRL
  - **WDGRL:** Wasserstein distance & addition of classification loss in feature loss
  - **ADDA:** 2 separate encoders for source and target

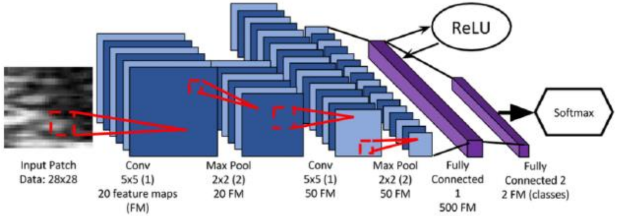
Proposed method

- 2 separate encoders for source and target (ADDA)
- Addition of classification loss in feature loss (WDGRL)
- Addition of weight

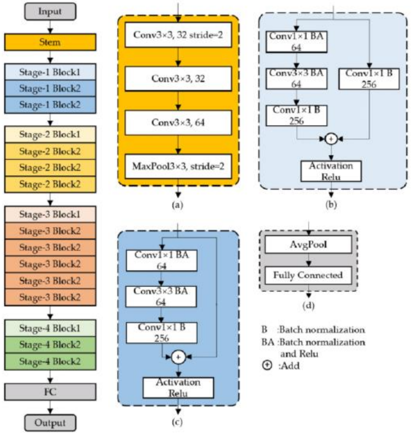
critic	$$\min_d L^d = \underbrace{\frac{1}{T}\sum_{x\in X^t} d\left(\boxed{\tau^t(x^t)}\right) - \frac{1}{S}\sum_{x\in X^s} d\left(\boxed{\tau(x^s)}\right)\boxed{w(\tau^s(x^s))}}_{\text{Wasserstein loss}} + \underbrace{\nu\frac{1}{E}\sum_{x\in X^e}(\lVert \nabla_e d(e)\rVert_2 - 1)^2}_{\text{gradient penalty}}$$
feature encoder	$$\min_{\tau^t} L^\tau = \lambda^{\tau^t}\underbrace{\left[-\frac{1}{T}\sum_{x\in X^t} d\left(\boxed{\tau^t(x^t)}\right)\right]}_{\text{Wasserstein loss}} + \underbrace{L^{ce,ours}}_{\text{classification loss}}$$
weight	$$\min_w L^w = \lambda^w \underbrace{\left[\frac{1}{S}\sum_{x\in X^s} d\left(\boxed{\tau^s(x^s)}\right)\boxed{w(\tau^s(x^s))}\right]}_{\text{Wasserstein loss}} + \underbrace{L^{ce,ours}}_{\text{classification loss}}$$

Datasets & backbone architectures

modified LeNet



ResNet-50



Experiments

- **Experiment 4:** unweighted feature-based adaptation
- **Experiment 5:** weighted feature-based adaptation
- **Experiment 6:** weighted feature-based *imbalanced* adaptation

Results for state-of-the-art comparison

Experiment 4: unweighted feature-based model

- MNIST-USPS

Method		$M \to U$	$U \to M$
	Source only	0.761	0.588
Previous work	DANN [7]	0.771	0.750
	DDC [16]	0.791	0.698
	ADDA [1]	0.896	0.909
	WDGRL [6]	-	-
	CoGAN [8]	0.912	0.899
	DRAnet [11]	**0.978**	**0.991**
This work	WDGRL^{ours}	0.920	0.898
	ADDA^{ours}	0.901	0.923
	ours	**0.958**	**0.936**

Experiment 4: unweighted feature-based model

- MNIST-USPS

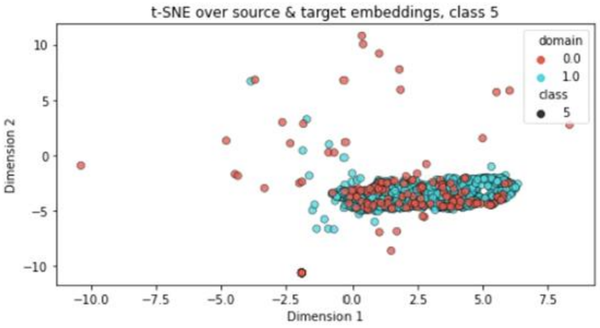


Figure 5.3: Example feature representation for ADDA

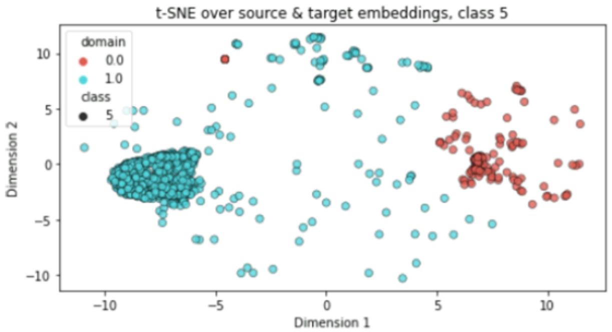


Figure 5.4: Example feature representation for WDGRL

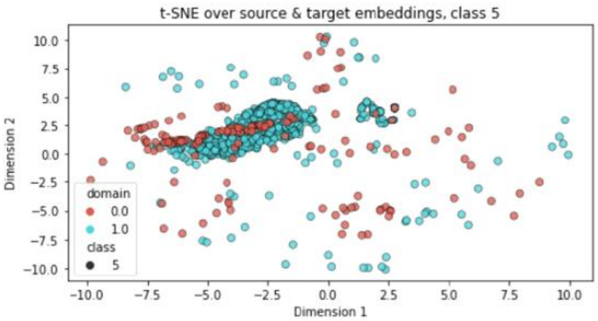


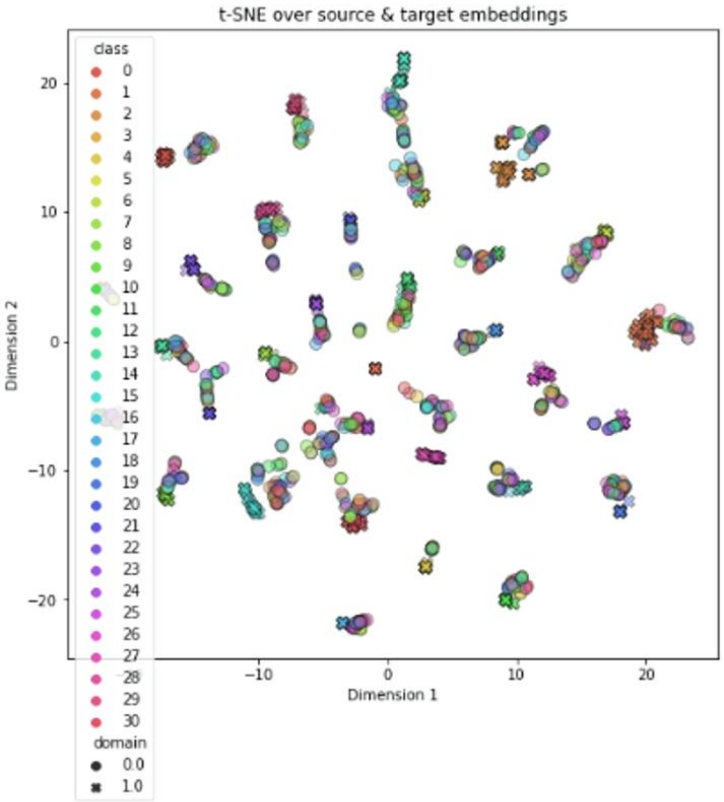
Figure 5.5: Example feature representation for our method

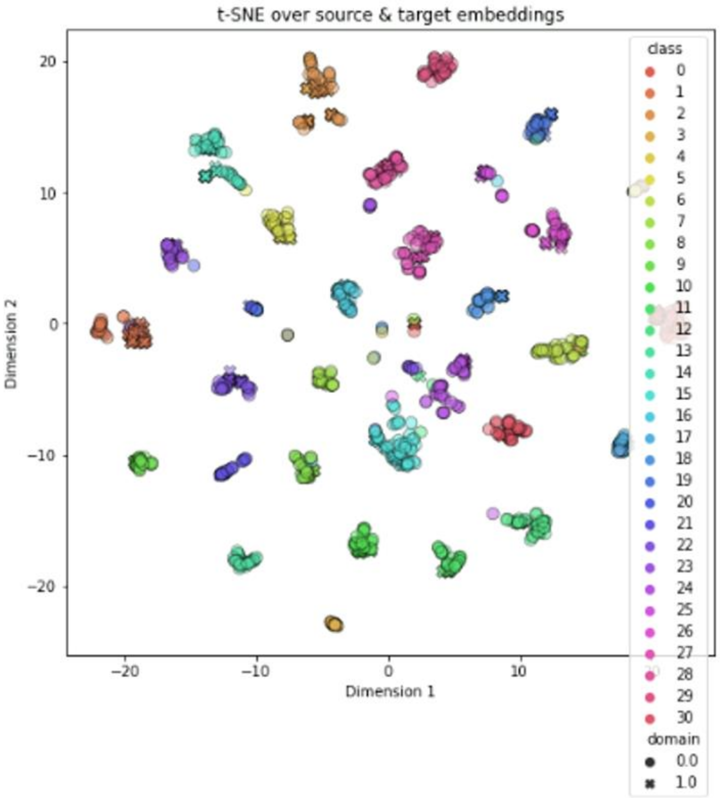
Experiment 4: unweighted feature-based model

- Office-31

Method		$A \to W$	$D \to W$	$W \to A$
	Source only	0.626	0.961	0.610
Previous work	DANN [7]	0.730	0.964	0.675
	ADDA [1]	0.751	0.970	-
	DDC [16]	0.618	0.950	-
	WDGRL [6]	0.895	0.979	0.937
	DADA [18]	**0.924**	**0.993**	**0.743**
This work	ADDA^{ours}	0.765	0.972	0.697
	ours	**0.803**	**0.980**	**0.709**

Experiment 4: Office-31





Experiment 5: integration with weight

- Large design choice space

critic	$\min_d L^d = \frac{1}{T}\sum_{x\in X^t} d(\tau^t(x^t)) - \frac{1}{S}\sum_{x\in X^s} d(\tau(x^s))w(\tau^s(x^s)) + \nu\frac{1}{E}\sum_{x\in X^e}(\lVert\nabla_e d(e)\rVert_2 - 1)^2$
feature encoder	$\min_{\tau^t} L^\tau = \boxed{\lambda^{\tau^t}}\left[-\frac{1}{T}\sum_{x\in X^t} d(\tau^t(x^t))\right] + \boxed{L^{ce,ours}}$
weight	$\min_w L^w = \boxed{\lambda^w}\left[\frac{1}{S}\sum_{x\in X^s} d(\tau^s(x^s))w(\tau^s(x^s))\right] + \boxed{L^{ce,ours}}$

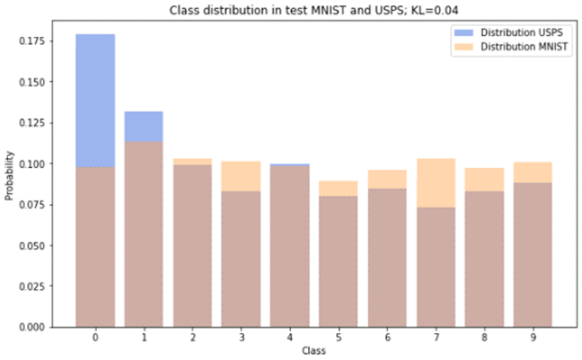
Experiment 5: integration with weight

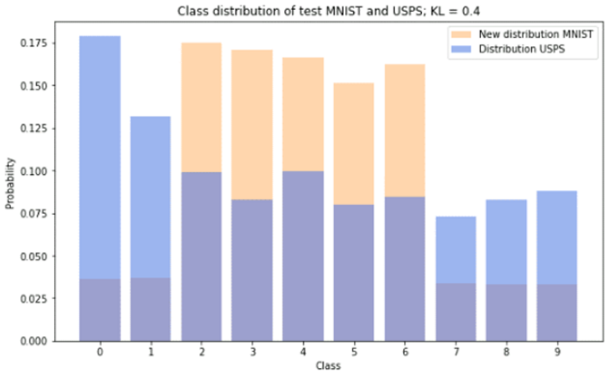
Method	λ_τ	λ_w	$L^{ce,ours}$ for τ	$L^{ce,ours}$ for w	w in $L^{ce,ours}$	w_{pass}	activation	acc_{target}
Source-only								0.761
Unweighted$_{ours}$								0.958
trial 1	✓	✓	✓	×	✓	×	softmax	0.834
trial 2	✓	×	✓	×	✓	×	softmax	0.818
trial 3	×	×	✓	✓	✓	×	softmax	0.792
trial 4	✓	×	✓	✓	✓	×	relu	0.801
trial 5	✓	×	✓	×	✓	×	relu	0.799
trial 6	✓	✓	✓	✓	✓	×	softmax	**0.842**
trial 7	✓	✓	✓	✓	✓	✓	softmax	**0.862**

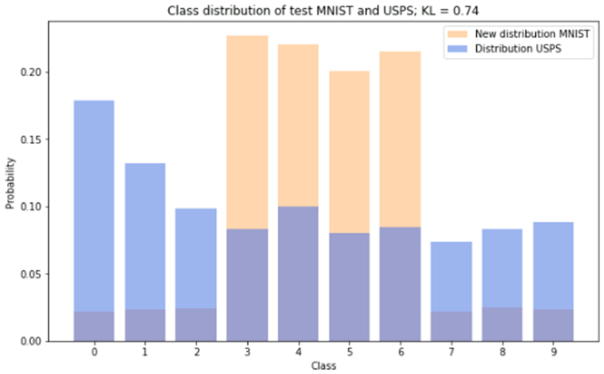
Table 5.3: Results for weighted domain adaptation on MNIST-USPS transfer

Experiment 6: imbalanced domain adaptation

- **x-axis:** class
- **y-axis:** probability







KL = 0.04

KL = 0.40

KL = 0.74

Experiment 6: imbalanced domain adaptation

Method		$M \to U$	
$KL(S,T)$	0.04	0.40	0.74
Unweighted	0.872	0.609	0.640
Weighted	0.865	**0.714**	0.643

Discussion & Conclusion

- Research focus
  - Investigate whether an adversarial instance re-weighting framework, inspired by Wasserstein GAN, is able to enhance unsupervised domain adaptation problem
- Proof-of-concept: small domain shift
  - **Experiment 1:** one-dimensional Gaussians
    - Source and target domain should have sufficient overlap
  - **Experiment 2:** MNIST, class imbalance
    - Algorithm is able to re-weight based on class
  - **Experiment 3:** MNIST, covariate shift
    - Algorithm is able to re-weight based on features

Discussion & Conclusion

- Comparison with state-of-the-art: large domain shift
  - **Experiment 4:** unweighted feature-based adaptation
    - Our method improved ADDA and WDGRL for MNIST-USPS and Office-31
    - Did not improve state-of-the-art
  - **Experiment 5:** weighted feature-based adaptation
    - Large choice space leads to difficult optimization
    - Improves source-only; does not improve unweighted methods
  - **Experiment 6:** weighted feature-based & imbalanced adaptation
    - For a KL-divergence of 0.40 between MNIST and USPS, weighted method outperforms unweighted method