Domain Adaptation: Tackle Distribution Shift Without Access to Target Label

Théo Gnassounou

Reading Group Hi!Paris, 11-03-2025

What is Domain Adaptation (DA)?

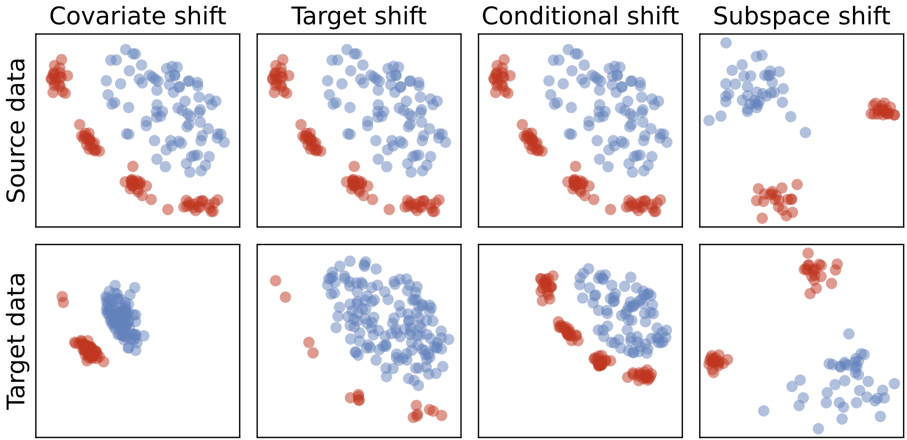
- Two type of domains: **Source** and **Target** .
- **Source** domains **with label** and **Target** domains **without label** .
- Assumption $\rightarrow$ **shift** between the distribution of the domain's data

4 different types of shifts

- **Covariate Shift** (CS): $\mathcal{P}^s_{\mathcal{X}}(x) \neq \mathcal{P}^t_{\mathcal{X}}(x)$, $\mathcal{P}^s(y|X) = \mathcal{P}^t(y|X)$
- **Target Shift** (TS): $\mathcal{P}^s_{\mathcal{Y}}(x) \neq \mathcal{P}^t_{\mathcal{Y}}(x)$, $\mathcal{P}^s(X|y) = \mathcal{P}^t(X|y)$
- **Conditional Shift** (CondS): $\mathcal{P}^s(y|X) \neq \mathcal{P}^t(y|X)$ or $\mathcal{P}^s(X|y) \neq \mathcal{P}^t(X|y)$
- **Subspace Shift** (SS): $\mathcal{P}^s(X) \neq \mathcal{P}^t(X)$ but it exists a subspace projection W such that $\mathcal{P}^s(WX) = \mathcal{P}^t(WX)$

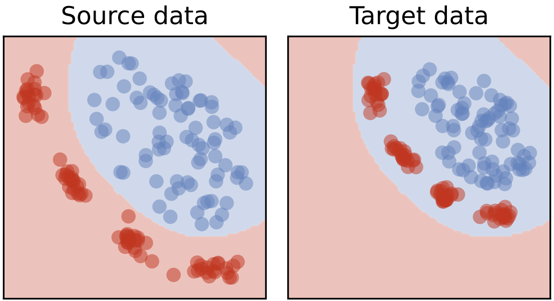
What is Domain Adaptation (DA)?

- Two type of domains: **Source** and **Target** .
- **Source** domains **with label** and **Target** domains **without label** .
- Assumption $\rightarrow$ **shift** between the distribution of the domain's data



What is Domain Adaptation (DA)?

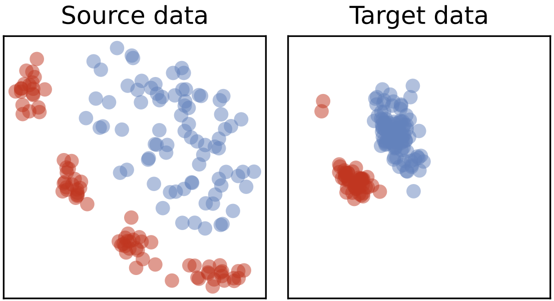
- Two type of domains: **Source** and **Target** .
- **Source** domains **with label** and **Target** domains **without label** .
- Assumption → **shift** between the distribution of the domain's data



→ **Problem:** **Drop in performance** when applying a model trained on the source to the target.

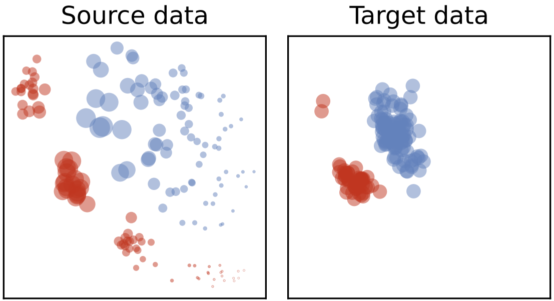
Traditional DA methods: Covariate Shift and target shift

- **One** source (X_s, y_s) and **one** target $(X_t,)$
- **Adapt** the source to the target via: **Reweighting**



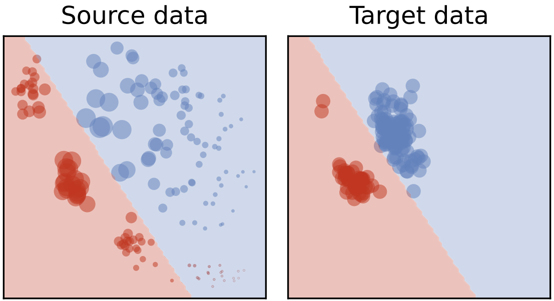
Traditional DA methods: Covariate Shift and target shift

- **One** source (X_s, y_s) and **one** target $(X_t,)$
- **Adapt** the source to the target via: **Reweighting**



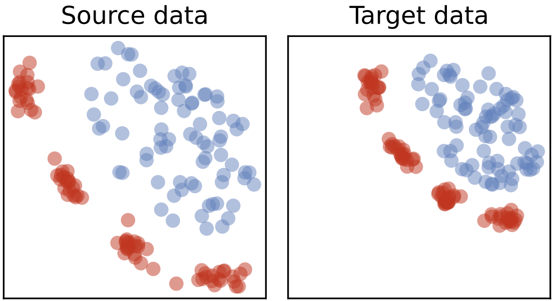
Traditional DA methods: Covariate Shift and target shift

- **One** source (X_s, y_s) and **one** target $(X_t,)$
- **Adapt** the source to the target via: **Reweighting**



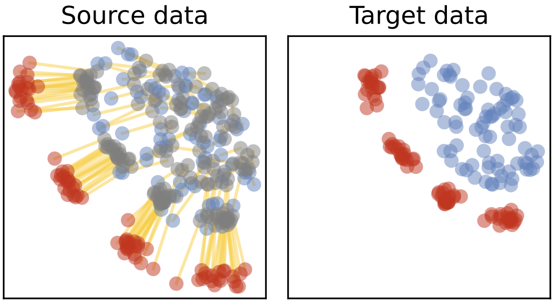
Traditional DA methods: Conditional Shift

- **One** source (X_s, y_s) and **one** target $(X_t,)$
- **Adapt** the source to the target via: **Mapping**



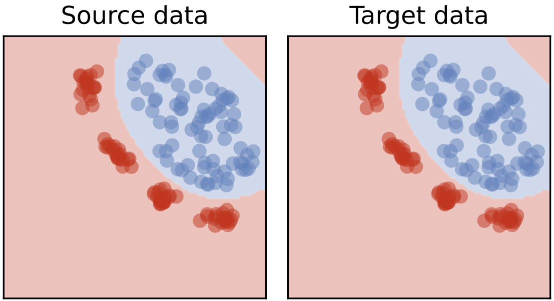
Traditional DA methods: Conditional Shift

- **One** source (X_s, y_s) and **one** target $(X_t,)$
- **Adapt** the source to the target via: **Mapping**



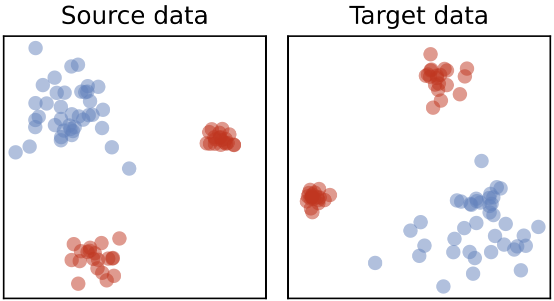
Traditional DA methods: Conditional Shift

- **One** source (X_s, y_s) and **one** target $(X_t,)$
- **Adapt** the source to the target via: **Mapping**



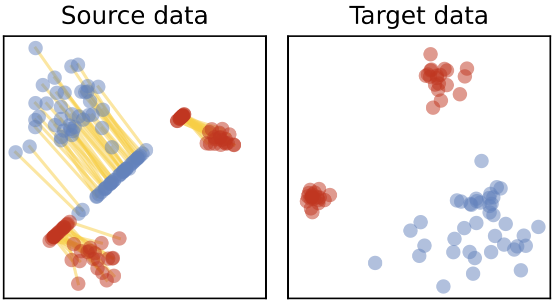
Traditional DA methods: Subspace shift

- **One** source (X_s, y_s) and **one** target (X_t,)
- **Adapt** the source to the target via: **Subspace**



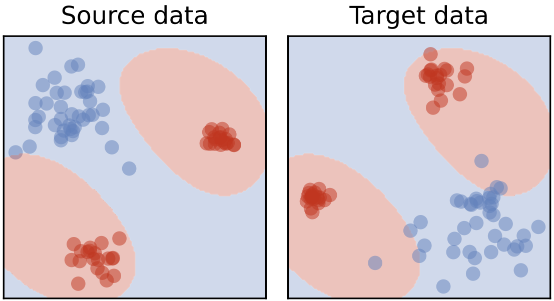
Traditional DA methods: Subspace shift

- **One** source (X_s, y_s) and **one** target $(X_t,)$
- **Adapt** the source to the target via: **Subspace**

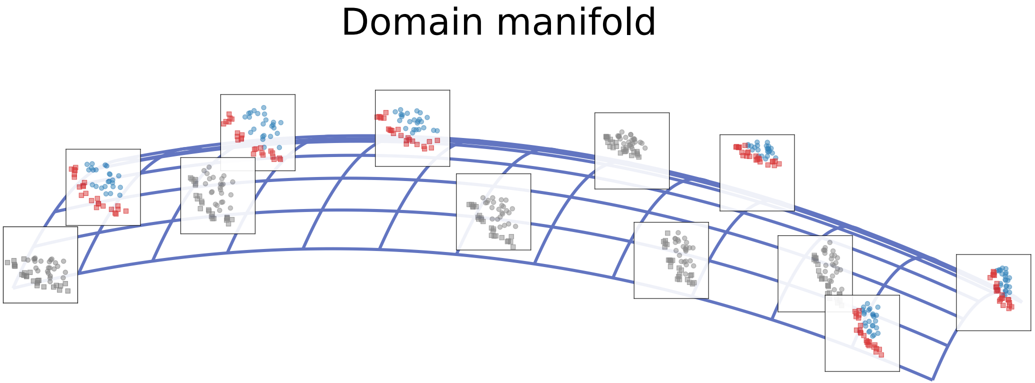


Traditional DA methods: Subspace shift

- **One** source (X_s, y_s) and **one** target $(X_t,)$
- **Adapt** the source to the target via: **Subspace**



Multi-source multi-target Domain Adaptation



Source-free Domain Adaptation (or Test-Time DA)

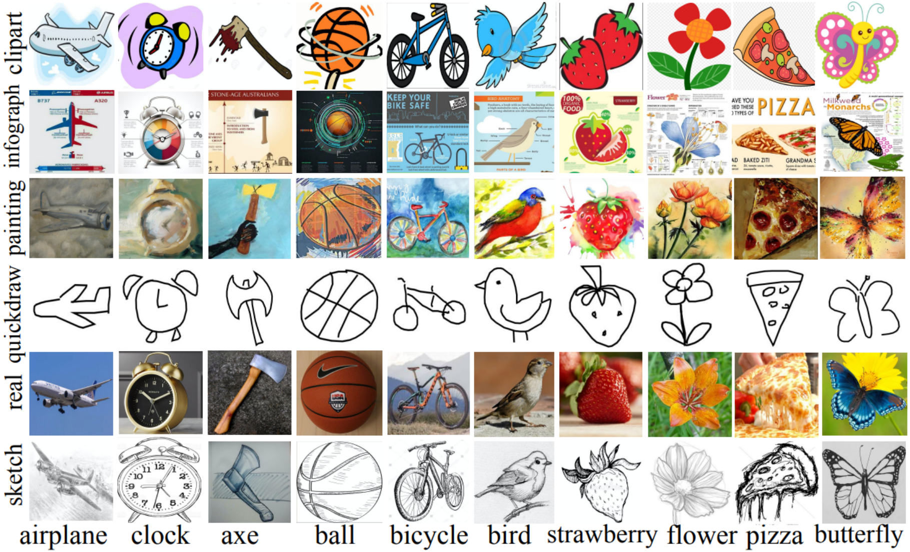
1. Train-time

- Acces to **Source** domains with labels
- **No** access to **Target** domains
- **Train** a model on the source domains with labels

2. Test-time

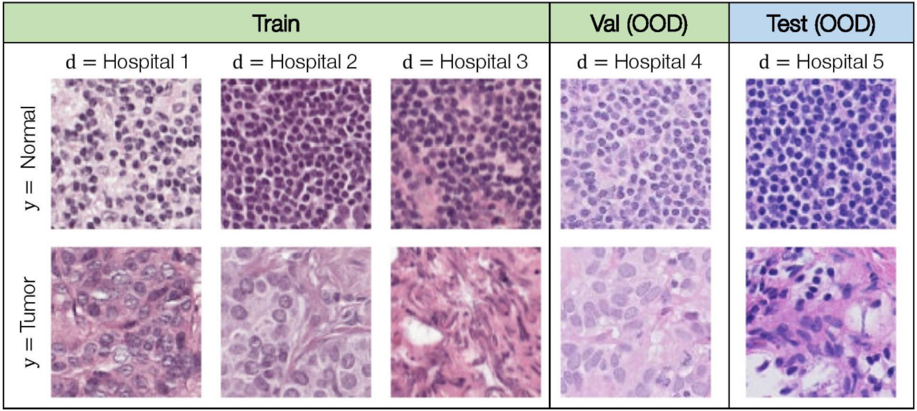
- **No** access to **Source** domains
- Acces to **Target** domains **without** labels
- **Finetune** the model on the target domains without access to the target labels

Real-world applications: Computer Vision



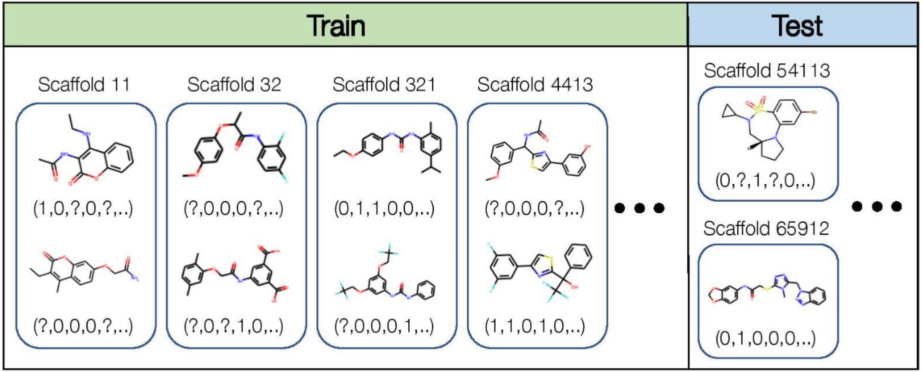
Source: Peng et. al., 2019

Real-world applications: Computer Vision



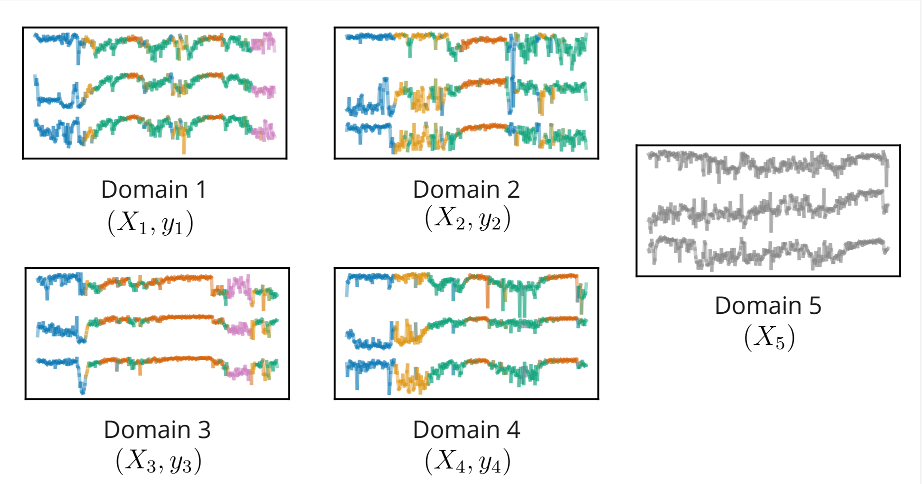
Source: Koh et. al., WILDS, 2020

Real-world applications: Biology



Source: Koh et. al., WILDS, 2020

Real-world applications: Time series



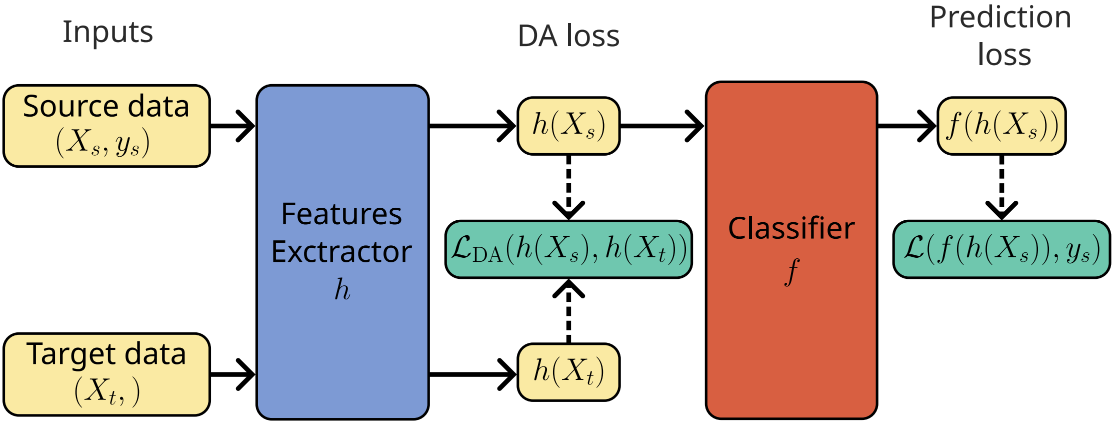
Example: Sleep stage classification from EEG signals[1]

[1] *Gnassounou et. al., 2023*

Deep Learning for DA

Why use Deep Learning?

- More **fine-grained feature** extraction
- **End-to-end** learning: no need for adaptation step



Deep learning for DA

$$\mathcal{L}_{\text{tot}}(h, f) = \underbrace{\mathcal{L}(f(h(X_s)), y_s)}_{\text{Prediction Loss}} - \overbrace{\lambda}^{\text{Regularization}} \underbrace{\mathcal{L}_{\text{DA}}(f(h(X_s)), f(h(X_t)))}_{\text{DA loss}} ,$$

- **Loss** $\rightarrow$ **Cross-entropy** loss
- **Regularization** $\rightarrow$ **Threshold** between the loss and the DA loss
- **DA loss** $\rightarrow$ **Reduce divergence** between source and target features

$$(\hat{h}, \hat{f}) = \underset{h,f}{\operatorname{argmin}}\ \mathcal{L}_{\text{tot}}(h, f)$$

How to reduce the divergence between source and target features?

Domain-Adversarial Training of Neural Networks

Yaroslav Ganin GANIN@SKOLTECH.RU
Evgeniya Ustinova EVGENIYA.USTINOVA@SKOLTECH.RU
Skolkovo Institute of Science and Technology (Skoltech)
Skolkovo, Moscow Region, Russia

Hana Ajakan

Deep CORAL: Correlation Alignment for Deep Domain Adaptation

Baochen Sun* and Kate Saenko**

University of Massachusetts Lowell, Boston University

Abstract. Deep neural networks are able to learn powerful representations from large quantities of labeled input data, however they cannot always generalize well across changes in input distributions. Domain adaptation algorithms have been proposed to compensate for the degradation in performance due to domain shift. In this paper, we address the case when the target domain is unlabeled, requiring unsupervised adaptation. CORAL[1] is a "frustratingly easy" unsupervised domain adaptation method that aligns the second-order statistics of the source and target distributions with a linear transformation. Here, we extend CORAL to learn a nonlinear transformation that aligns correlations of layer activations in deep neural networks (Deep CORAL). Experiments on standard benchmark datasets show state-of-the-art performance.

DeepJDOT: Deep Joint Distribution Optimal Transport for Unsupervised Domain Adaptation

Bharath Bhushan Damodaran[1]*, Benjamin Kellenberger[2]*, Rémi Flamary[3], Devis Tuia[2], Nicolas Courty[1]

[1] Université de Bretagne Sud, IRISA, UMR 6074, CNRS, France
[2] Wageningen University, the Netherlands
[3] Université Côte d'Azur, OCA, UMR 7293, CNRS, Laboratoire Lagrange, France
{bharath-bhushan.damodaran@irisa.fr, benjamin.kellenberger@wur.nl}

Abstract. In computer vision, one is often confronted with problems of domain shifts, which occur when one applies a classifier trained on a source dataset to target data sharing similar characteristics (e.g. same classes), but also different latent data structures (e.g. different acquisition conditions). In such a situation, the model will perform poorly on the new data, since the classifier is specialized to recognize visual cues specific to the source domain. In this work we explore a solution, named DeepJDOT, to tackle this problem: through a measure of discrepancy on joint deep representations/labels based on optimal transport, we not only learn new data representations aligned between the source and target domain, but also simultaneously preserve the discriminative information used by the classifier. We applied DeepJDOT to a series of visual recognition tasks, where it compares favorably against state-of-the-art deep domain adaptation methods.

Abstract

We introduce a new representation learning approach for domain adaptation, in which data at training and test time come from similar but different distributions. Our approach is directly inspired by the theory on domain adaptation suggesting that, for effective do-

DeepCoral: Correlation Alignment[1]

With d the dimension of the feature space, the **Coral loss** is defined as:

$$\mathcal{L}_{\mathrm{DA}}(h) = \frac{1}{4d^2} \| \underbrace{C(h(X_s))}_{\text{Source Covariance}} - \overbrace{C(h(X_t))}^{\text{Target Covariance}} \|_F^2 ,$$

with $\|\cdot\|_F$ the Frobenius norm. The covariance matrices are defined as:

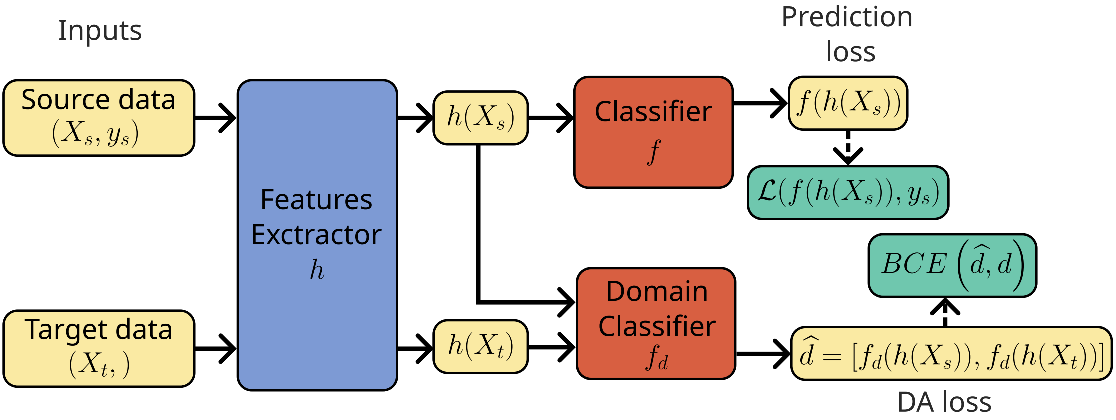
$$C(X) = \frac{1}{N-1} \left(X^\top X - \frac{1}{N} (\mathbf{1}^\top X)^\top (\mathbf{1}^\top X) \right) ,$$

with $\mathbf{1}$ a vector of ones and N the number of samples.

[1] *Sun et. al., 2016*

DANN: Domain Adversarial Neural Network[1]

- **Adversarial** training with a **domain classifier**
- **Binary** classification: $\mathbf{d = 0}$ for **source** and $\mathbf{d = 1}$ for **target**



[1] *Ganin et. al., 2016*

DANN: Domain Adversarial Neural Network[1]

Adversarial loss with binary cross entropy (BCE) loss:

$$\mathcal{L}_{\text{DA}}(g, f, f^d) = \underbrace{-}_{\text{Reverse gradient}} \overbrace{BCE([f_d(h(X_s)), f_d(h(X_t))], [\mathbf{0}, \mathbf{1}])}^{\text{Binary cross entropy}} .$$

$$(\widehat{h}, \widehat{f}) = \underset{h,f}{\text{argmin}}\, \mathcal{L}_{\text{tot}}(g, f, f^d) \ ,$$
$$\widehat{f_d} = \underset{f_d}{\text{argmax}}\, \mathcal{L}_{\text{DA}}(g, f, f^d) \ .$$

In practice:

- **Joint** optimization of the feature extractor and the domain classifier
- **Reverse Gradient Layer**
- $f_d \rightarrow$ 3 layers of fully connected layers

[1] *Ganin et. al., 2016*

DeepJDOT: Joint Distribution Optimal Transport[1]

→ **Optimal Transport** to align the source and target distributions

Brief reminder on Optimal Transport

The relaxed version of the Kantorovich Problem is[2]:

$$\gamma_0 = \underset{\gamma \in \Pi}{\operatorname{argmin}} \int_{\mathcal{X}_s \times \mathcal{X}_t} \underbrace{c(X_s, X_t)}_{\text{Cost matrix}} \, d \, \overbrace{\gamma(X_s, X_t)}^{\text{Transport plan}} \; ,$$

where **Π** is the set of all the **couplings** between marginal distributions μ_s and μ_t.
For discrete OT, introducing the cost matrix $(C)_{i,j} = c(X_s^i, x_t^j)$ the Kantorovich Problem becomes:

$$\gamma_0 = \underset{\gamma \in \mathcal{B}}{\operatorname{argmin}} \langle \gamma, \mathbf{C} \rangle_F \; , \tag{1}$$

[1] *Damodaran et. al., 2018*
[2] *Peyré et. al., 2019*

DeepJDOT: Joint Distribution Optimal Transport[1]

The **Joint Distribution Optimal Transport** loss is defined as:

$$\mathcal{L}_{\mathrm{DA}}(h, f) = \langle \gamma, \mathbf{C_{h,f}} \rangle_F \ ,$$

$C_{h,f}$ the cost matrix defines with the **feature extractor** and the **classifier** :

$$C_{h,f}(X_s^i, X_t^j) = \overbrace{\alpha}^{\text{Regularization}} \underbrace{\|h(X_s^i) - h(X_t^j)\|^2}_{\text{Distance}} + \overbrace{\beta}^{\text{Regularization}} \underbrace{\mathcal{L}(f(h(X_t^i)), y_s^j)}_{\text{Pseudo-labeling loss}} \ ,$$

- Cost matrix $\rightarrow$ reduce **Distance** between source and target features
- Cost matrix $\rightarrow$ Map target with **same** predicted **label** as source
- **Regularization** between distance and pseudo-labeling loss

[1] *Damodaran et. al., 2018*

DeepJDOT: Joint Distribution Optimal Transport[1]

Two steps optimization:

1. Compute the **optimal transport plan** γ between the source and target **batches**

$$\gamma = \underset{\gamma}{\operatorname{argmin}} \, \langle \gamma, \mathbf{C}_{\mathbf{h},\mathbf{f}} \rangle_F \ ,$$

Can be done using **POT** [2] library.

2. Update the **feature extractor** and the **classifier** by minimizing the **total loss**

$$(\hat{h}, \hat{f}) = \underset{h,f}{\operatorname{argmin}} \, \mathcal{L}_{\text{tot}}(h, f)$$

[1] *Damodaran et. al., 2018*
[2] *Flamary et. al., 2017*

In practice: How to choose the best regularization?

No labels in the target domain $\rightarrow$ **No** way to **tune** the **hyperparameters** .

- DeepCoral $\rightarrow$ 1 hyperparameter
- DANN $\rightarrow$ 1 hyperparameter
- DeepJDOT $\rightarrow$ 2 hyperparameters

Solutions in paper:

- DeepCoral $\rightarrow$ λ fixed: $\mathcal{L} \sim \mathcal{L}_{\mathrm{DA}}$ at the end of the training
- DANN $\rightarrow$ Reversed cross-validation[1]
- DeepJDOT $\rightarrow$ "Fixed experimentally" ...

[1] *Zhong et. al., 2010*

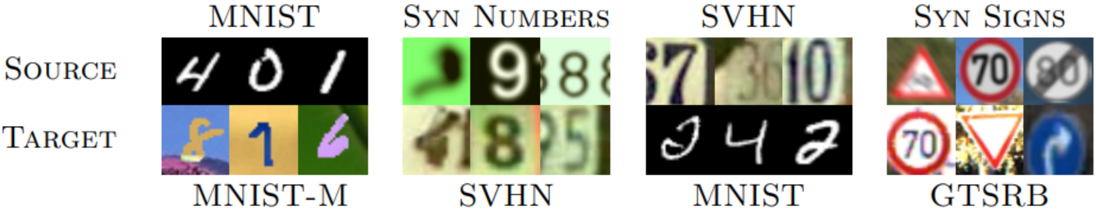
DA scorer: Metric to set the best hyperparameters

- **Import Weighted** [1] $\rightarrow$ Score as a **reweighted** accuracy on labeled sources data
- **Depp Embedded Validation** [2] $\rightarrow$ IW strategy in the **latent space** with variance reduction strategy
- **Prediction Entropy** [3] $\rightarrow$ Reduce **Entropy** of the **prediction** on the target domain to reduce uncertainty
- **Circular Validation** [4] $\rightarrow$ Adapt Source to Target then Target to Source with predicted labels

[1] *Sugiyama et. al., 2007*
[2] *You et. al., 2019*
[3] *Morerio et. al., 2017*
[4] *Bruzzone et. al., 2010*

Experimental results: Digits dataset

Digits dataset



Source: Ganin et. al., 2016

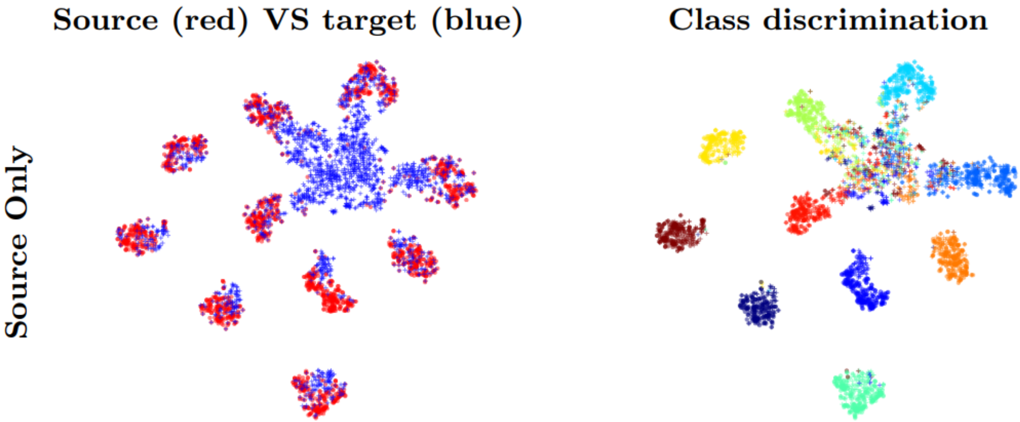
- Classification of **10** classes over **5** domains
- Shift between the domains: **Font** , **Color** , **Style**

Experimental results: Digits dataset

Method	Adaptation:source→target			
	MNIST → USPS	USPS → MNIST	SVHN → MNIST	MNIST → MNIST-M
Source only	94.8	59.6	60.7	60.8
DeepCORAL [6]	89.33	91.5	59.6	66.5
MMD [14]	88.5	73.5	64.8	72.5
DANN [8]	*95.7*	90.0	70.8	75.4
ADDA [21]	92.4	93.8	76.0[5]	78.8
AssocDA [16]	-	-	*95.7*	*89.5*
Self-ensemble[4] [42]	88.14	92.35	93.33	-
DRCN [40]	91.8	73.6	81.9	-
DSN [41]	91.3	-	82.7	83.2
CoGAN [9]	91.2	89.1	-	-
UNIT [18]	**95.9**	*93.5*	90.5	-
GenToAdapt [19]	95.3	90.8	92.4	-
I2I Adapt [20]	92.1	87.2	80.3	-
StochJDOT	93.6	90.5	67.6	66.7
DeepJDOT (ours)	*95.7*	**96.4**	**96.7**	**92.4**
target only	95.8	98.7	98.7	96.8

Source: Damodaran et. al., 2018

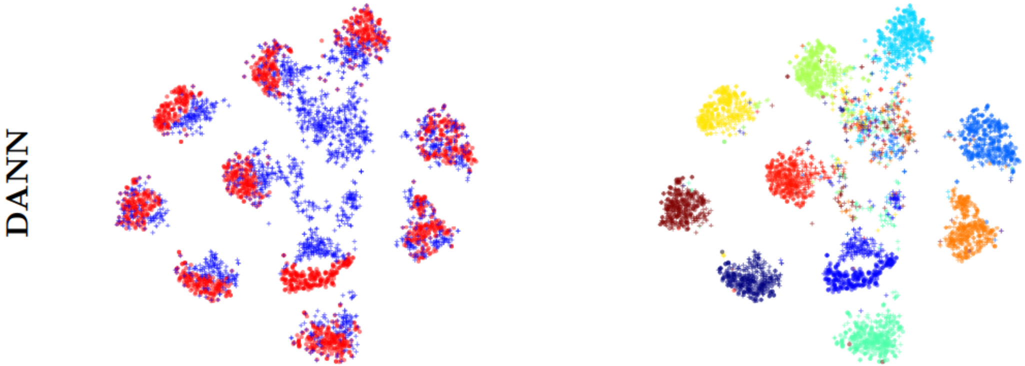
Experimental results: TSNE visualization with Source Only



- **Target** domain samples are **not clustered**

Source: Damodaran et. al., 2018

Experimental results: TSNE visualization with DANN



- **Target** domain samples are **more clustered** but few are misclassified

Source: Damodaran et. al., 2018

Experimental results: TSNE visualization with DeepJDOT

- **Target** domain samples are **perfectly clustered** !

Source: Damodaran et. al., 2018

How to use DA?

Skada [1] is a **Python** library to **easily** use DA methods.

- **Homogeneous API** for all DA methods (Shallow and Deep learning).
- **Sklearn-like API** with estimator class (.fit, .predict, . . .), pipeline, grid search . . .
- **DA scorer** to validate hyper-parameters without using target label.

SKADA

[1] *Gnassounou et. al., 2024*

Data format in Skada

- `X` → 2D array of shape `(n_samples, n_features)`
- `y` → 1D array of shape `(n_samples,)`
- `sample_domain` → 1D array of shape `(n_samples,)` giving the **domain** of each **sample**

```python
from skada.datasets import make_shifted_datasets

X, y, sample_domain = make_shifted_datasets(
    20, 20, shift='covariate_shift', random_state=42
)
```

- All shift are available in `make_shifted_datasets` function

Shallow DA in Skada

- Initialize the estimator
- Fit the model
- Don't forget to give the **sample domain**

```
from skada import LinOT

estimator = LinOT()
estimator.fit(X, y, sample_domain=sample_domain)
```

- $\sim$ 20 shallow methods available in Skada

Pipeline DA in Skada

- Can be used with **Pipeline**

```
from skada import make_da_pipeline
from skada import LinOTAdapter, GaussianReweightAdapter
from sklearn.linear_model import LogisticRegression

pipeline = Pipeline(
    LinOTAdapter(),
    LogisticRegression()
)
pipeline.fit(X, y, sample_domain=sample_domain)
```

- Possibility to mixed DA adapters

```
pipeline = Pipeline(
   LinOTAdapter(),
   GaussianReweightAdapter(),
   LogisticRegression()
)
```

DA scorer in Skada

- Possibility to use `cross_val_score` with **DA scorers**
- DA scorers are used to **validate** the **hyperparameters** without using the target labels

```
from skada.scorers import ImportanceWeightedScorer

scorer = ImportanceWeightedScorer()
score = cross_val_score(pipeline, X, y, sample_domain=sample_domain,
↪ scoring=scorer)
```

- 6 DA scorers available in Skada

Deep DA method in Skada

- Use **Skorch** $\rightarrow$ **Pytorch** wrapper for **Sklearn**
- Give an **architecture** and **hyperparameters**

```python
from skada.deep import DeepCoral
from skada.deep.modules import ToyCNN

model = DeepCoral(
    ToyCNN(),
    batch_size=32,
    max_epochs=5,
    lr=1e-3,
    reg=1,
    layer_name="feature_extractor",
)
model.fit(X, y, sample_domain=sample_domain)
```

- $\sim$ 10 Deep DA methods available in Skada

Conclusion

- Distribution shift is a **challenging** problem
- Deep learning methods reduce the shift in the **feature space**
- Modern DA methods are more focus on **Test-Time DA**
- Try **Skada** to easily use DA methods
- Don't hesitate to contribute to the library!