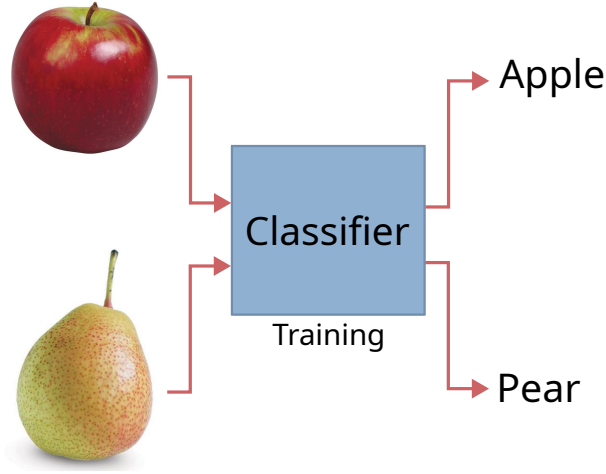
Tackling Distribution Shift with

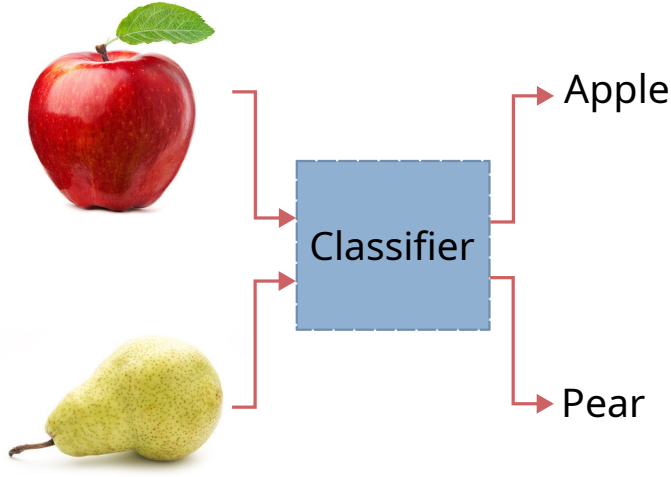
Théo Gnassounou, Antoine Collas, Rémi Flamary

PyData Paris, 30-09-2025

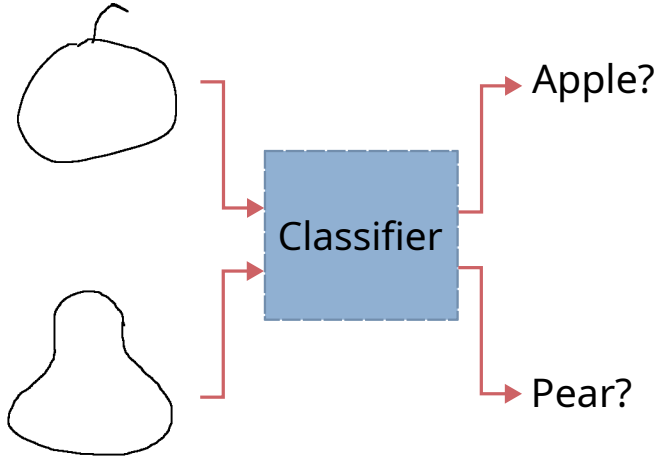
Machine Learning: a powerful tool



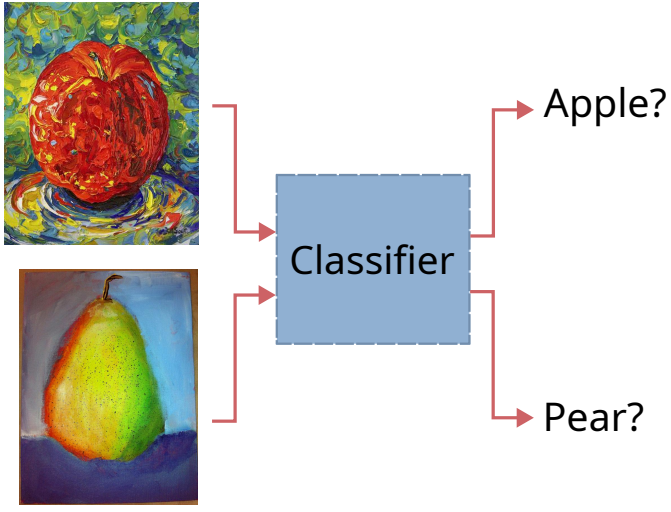
Machine Learning: a powerful tool



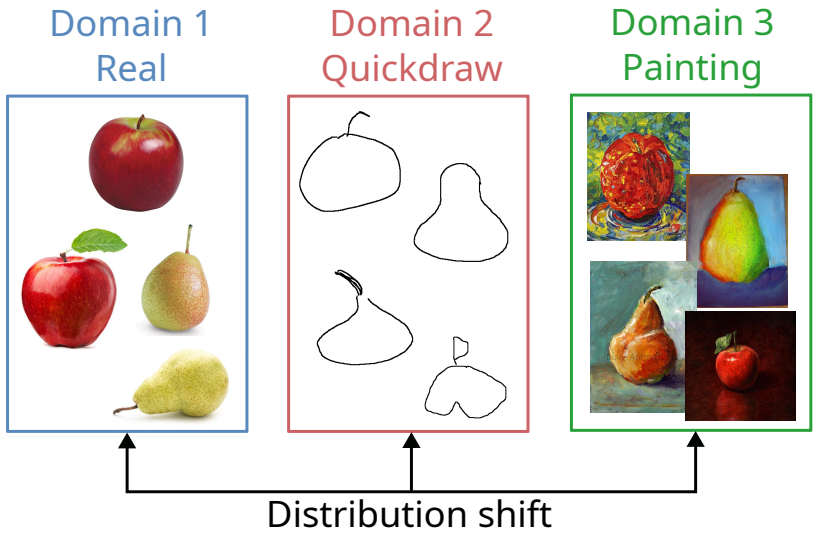
Machine Learning: a powerful tool but shift happens . . .



Machine Learning: a powerful tool but shift happens . . .

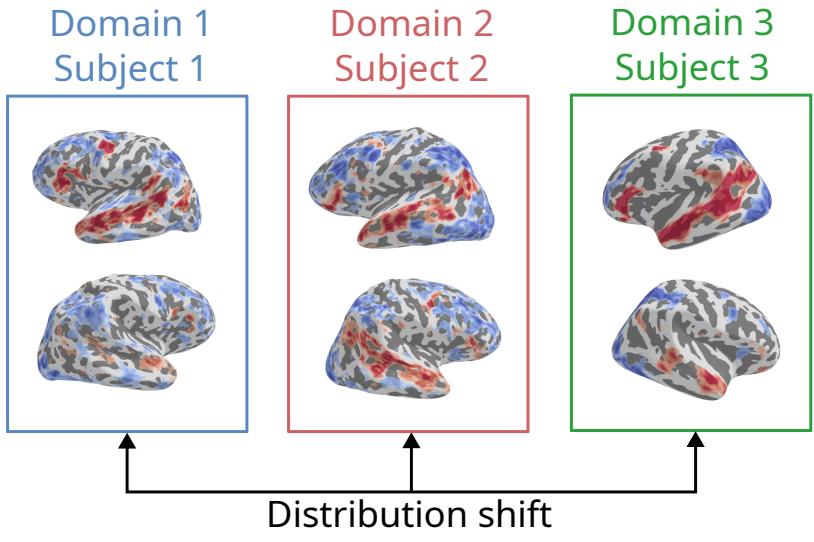


Distribution shift in Image classification[1]



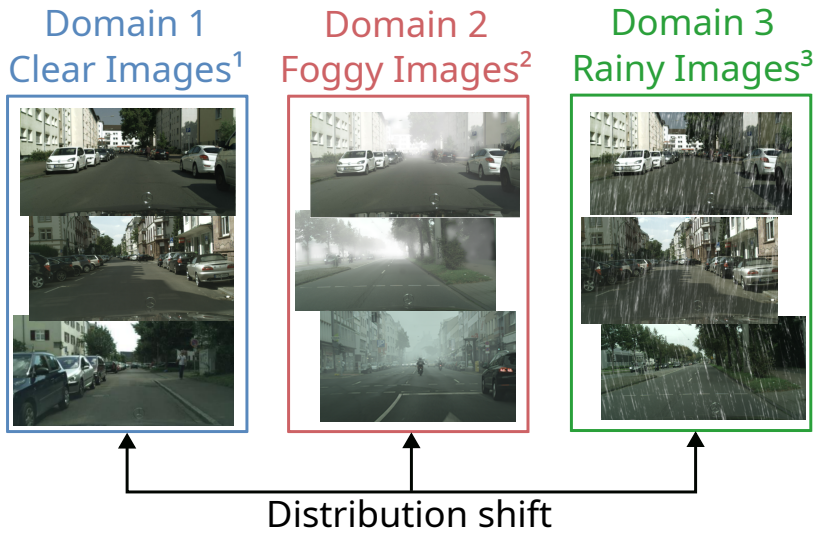
[1] *Dataset: DomainNet (Peng et. al., 2019)*

Distribution shift in Functional MRI[1]



[1] *Dataset: IBC Project (Pinho et. al., 2018)*

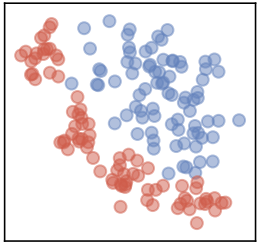
Distribution shift in Autonomous Driving



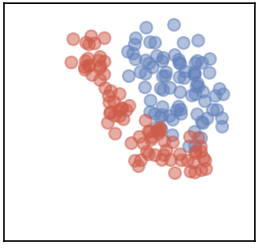
Dataset: [1] Cityscapes (Cordts et. al., 2016) [2] Foggy Cityscapes (Sakaridis et. al., 2018) [3] Rainy Cityscapes (Li et. al., 2024)

Impact of Distribution Shift

Source data $(\mathbf{X}_s, \mathbf{y}_s)$

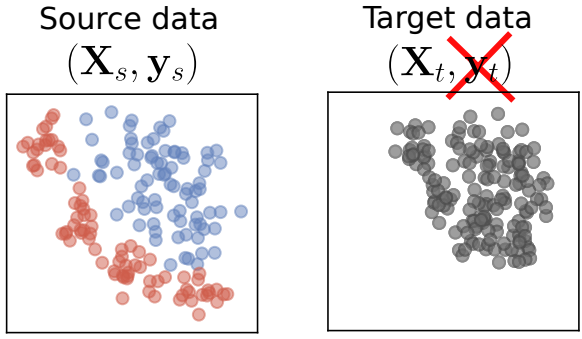


Target data $(\mathbf{X}_t, \mathbf{y}_t)$



- Suppose classification problem with two classes (**blue** and **red**).
- Source domain (left): **labeled** data.

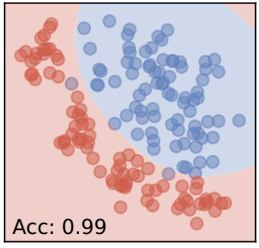
Impact of Distribution Shift



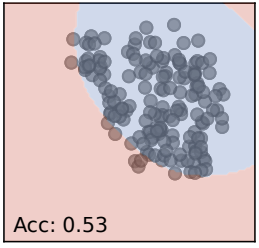
- Suppose classification problem with two classes (**blue** and **red**).
- Source domain (left): **labeled** data.
- Target domain (right): **unlabeled** data.

Impact of Distribution Shift

Source data
$(\mathbf{X}_s, \mathbf{y}_s)$



Target data
$(\mathbf{X}_t, ?)$



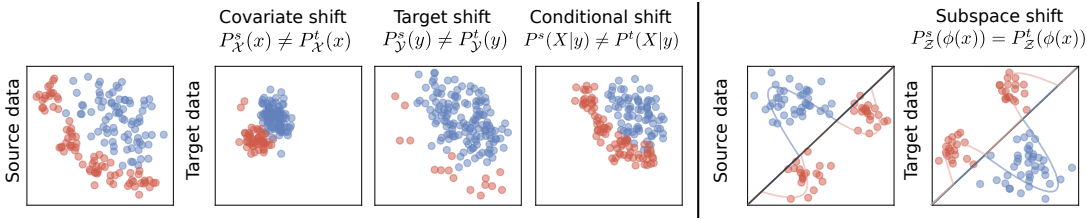
- Suppose classification problem with two classes (**blue** and **red**).
- Source domain (left): **labeled** data.
- Target domain (right): **unlabeled** data.
- Drop in performance when applying source classifier to target data.

How to use Domain Adaptation?

Open-source libraries	Shallow methods	Deep methods	Backend	Sklearn-Compatibility	DA Scorer	Number of methods	Number of contributors	Last commit
Salad[1]	✗	✓	torch	✗	✗	6	3	>4y
ADA[2]	✗	✓	torch	✗	✗	5	3	>2y
Pytorch-Adapt[3]	✗	✓	torch	✗	✗	$\sim$ 10	2	>2y
Tllib[4]	✗	✓	torch	✗	✗	$\sim$ 40	11	>1y
Adapt[5]	✓	✓	tf + numpy	✗	✗	$\sim$ 30	8	>1mo
Skada (ours)	✓	✓	torch + numpy	✓	✓	$\sim$ 30	13	<1mo

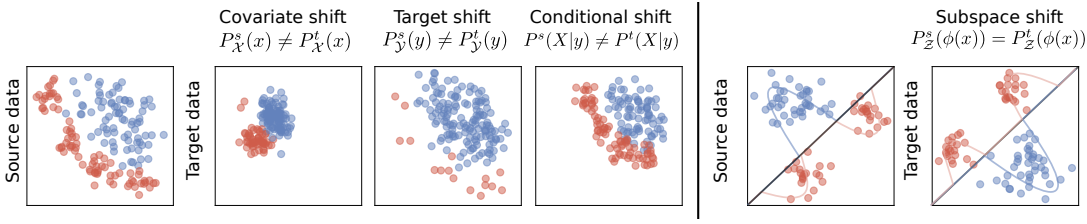
[1] *Schneider et. al., 2018* [2] *ADA Library, 2020*
[3] *Musgrave et. al., 2022* [4] *Wang et. al., 2020*
[5] *Demathelin et. al., 2024*

Existing Distribution Shifts



- **Covariate shift** : Distribution of **data** changes over domains.
- **Target shift** : Distribution of **labels** changes over domains.
- **Conditional shift** : Distribution of **data conditioned on labels** changes over domains.
- **Subspace Assumption** : Distribution of data of each class lies in a **subspace that is invariant** over domains.

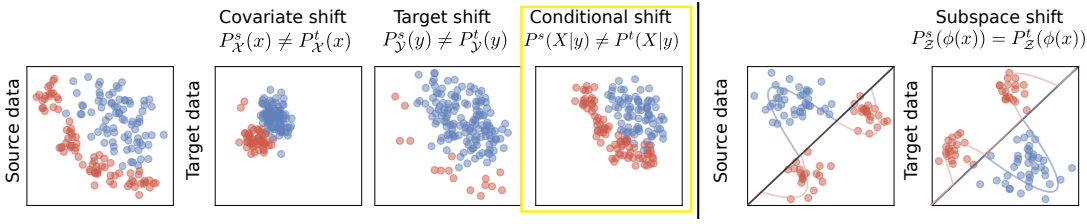
Methods available in SKADA



→ **20 Shallow methods** available in skada

- **Covariate shift** → Reweighting methods
- **Target shift** → Reweighting methods
- **Conditional shift** → Mapping methods
- **Subspace Assumption** → Subspace methods

Methods available in SKADA

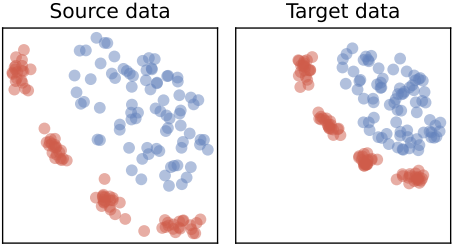


→ **20 Shallow methods** available in skada

- **Covariate shift** → Reweighting methods
- **Target shift** → Reweighting methods
- **Conditional shift** → Mapping methods
- **Subspace Assumption** → Subspace methods

How to use SKADA: Special Data format

```python
from skada.datasets import
    make_shifted_datasets

X, y, sample_domain = make_shifted_datasets(
    shift='conditional_shift',
    return_dataset=True
)
```



X , y , sample_domain

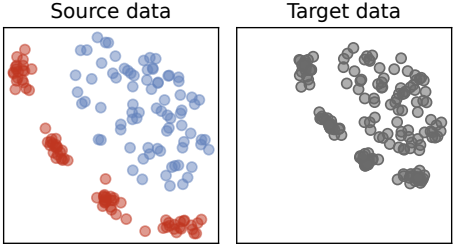
Domain label: (n_{samples})

Label: (n_{samples})

Data: $(n_{\text{samples}}, n_{\text{features}})$

How to use SKADA: Special Data format

```python
from skada.datasets import
↪  make_shifted_datasets

dataset = make_shifted_datasets(
    shift='conditional_shift',
    return_dataset=True
)

X, y, sample_domain = dataset.pack(
    as_sources=['s'],
    as_targets=['t'],
    mask_target_labels=True
)
```



$$\text{sample\_domain} = [1, 1, 1, 1, 1, \ldots, -2, -2, -2]$$

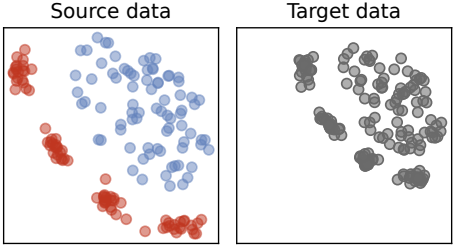
$$y = [\, 0, 1, 1, 0, 1 \,, \ldots, \, -1, -1, -1 \,]$$

True Labels

Masked Labels

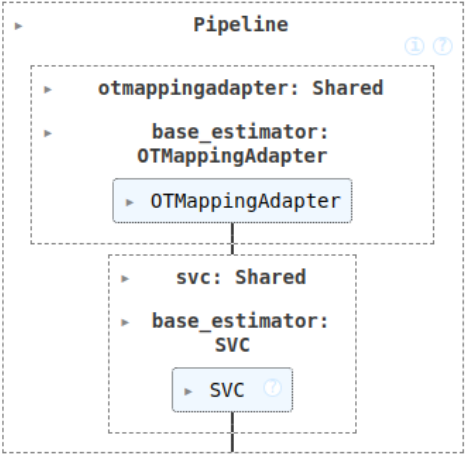
How to use SKADA: Sklearn-like estimator interface

```python
from skada import OTMapping

estimator = OTMapping()
```



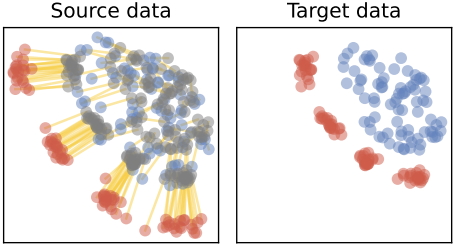
How to use SKADA: Sklearn-like estimator interface

```python
from sklearn.svm import SVC
from skada import OTMappingAdapter
from skada import make_da_pipeline

estimator = make_da_pipeline(
    OTMappingAdapter(),
    SVC(),
)

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



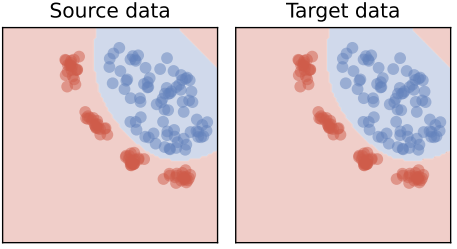
How to use SKADA: Sklearn-like estimator interface

```python
from sklearn.svm import SVC
from skada import OTMappingAdapter
from skada import make_da_pipeline

estimator = make_da_pipeline(
    OTMappingAdapter(),
    SVC(),
)

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



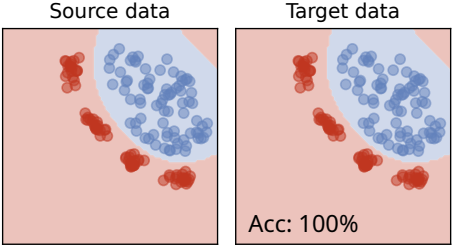
How to use SKADA: Sklearn-like estimator interface

```python
from sklearn.svm import SVC
from skada import OTMappingAdapter
from skada import make_da_pipeline

estimator = make_da_pipeline(
    OTMappingAdapter(),
    SVC(),
)

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



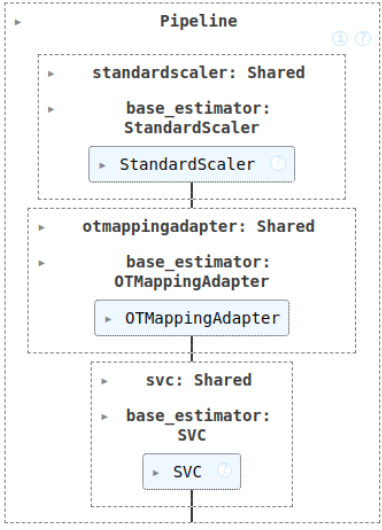
How to use SKADA: Sklearn-like estimator interface

```python
X, y, sample_domain = dataset.pack(
    as_sources=[],
    as_targets=['t'],
    mask_target_labels=False
)
estimator.score(X, y,
    sample_domain=sample_domain)
```



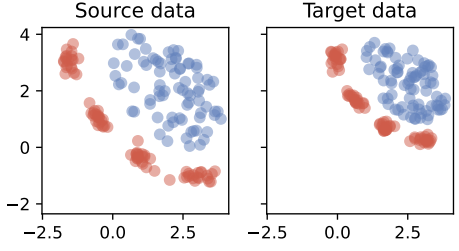
How to use SKADA: Sklearn-like DA Pipeline

```python
from sklearn.svm import SVC
from sklearn.preprocessing import
↪ StandardScaler
from skada import OTMappingAdapter
from skada import make_da_pipeline

estimator = make_da_pipeline(
    StandardScaler(),
    OTMappingAdapter(),
    SVC(),
)
```



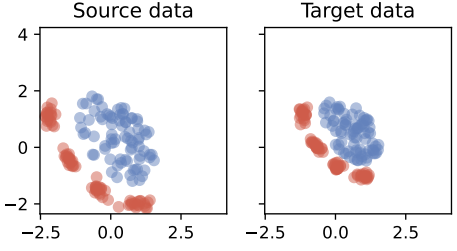
How to use SKADA: Sklearn-like DA Pipeline

```python
from sklearn.svm import SVC
from sklearn.preprocessing import
↪  StandardScaler
from skada import OTMappingAdapter
from skada import make_da_pipeline

estimator = make_da_pipeline(
    StandardScaler(),
    OTMappingAdapter(),
    SVC(),
)

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



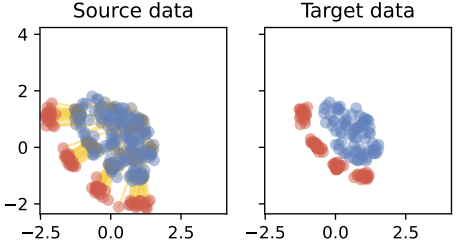
How to use SKADA: Sklearn-like DA Pipeline

```python
from sklearn.svm import SVC
from sklearn.preprocessing import
↪  StandardScaler
from skada import OTMappingAdapter
from skada import make_da_pipeline

estimator = make_da_pipeline(
    StandardScaler(),
    OTMappingAdapter(),
    SVC(),
)

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



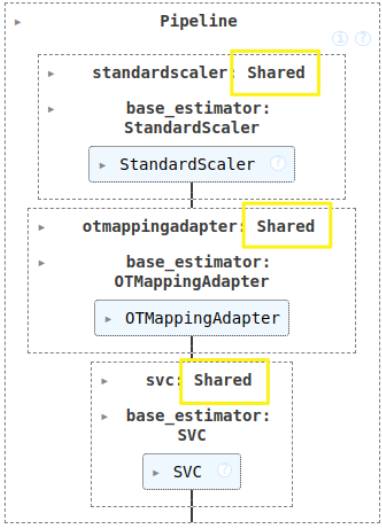
How to use SKADA: Sklearn-like DA Pipeline

```python
from sklearn.svm import SVC
from sklearn.preprocessing import
↪  StandardScaler
from skada import OTMappingAdapter
from skada import make_da_pipeline

estimator = make_da_pipeline(
    StandardScaler(),
    OTMappingAdapter(),
    SVC(),
)

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



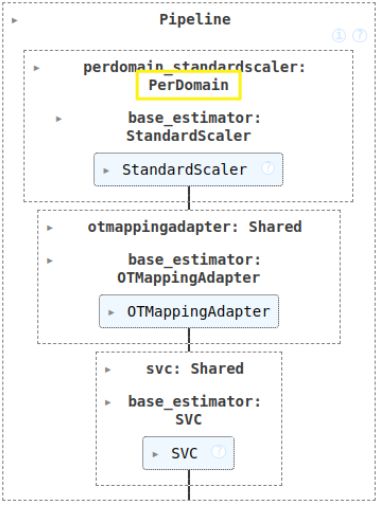
How to use SKADA: Sklearn-like DA Pipeline

```python
from sklearn.svm import SVC
from sklearn.preprocessing import
↪  StandardScaler
from skada import OTMappingAdapter
from skada import make_da_pipeline

estimator = make_da_pipeline(
    StandardScaler(),
    OTMappingAdapter(),
    SVC(),
)
```



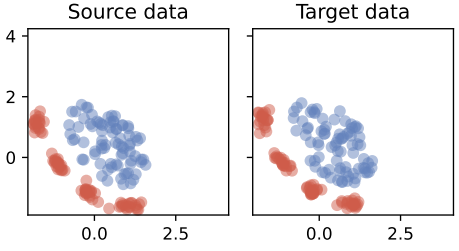
How to use SKADA: Sklearn-like DA Pipeline

```python
from sklearn.svm import SVC
from sklearn.preprocessing import
↪  StandardScaler
from skada import OTMappingAdapter
from skada import make_da_pipeline
from skada import PerDomain

estimator = make_da_pipeline(
    PerDomain(StandardScaler()),
    OTMappingAdapter(),
    SVC(),
)
```



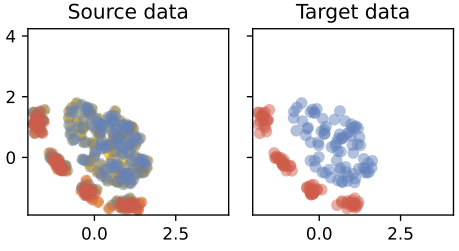
How to use SKADA: Sklearn-like DA Pipeline

```python
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from skada import OTMappingAdapter
from skada import make_da_pipeline
from skada import PerDomain

estimator = make_da_pipeline(
    PerDomain(StandardScaler()),
    OTMappingAdapter(),
    SVC(),
)

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

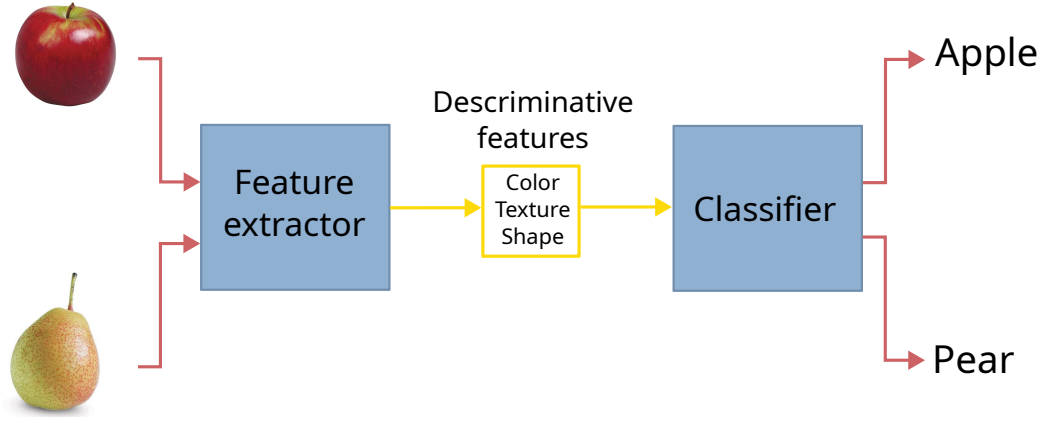


How to use SKADA: Sklearn-like DA Pipeline

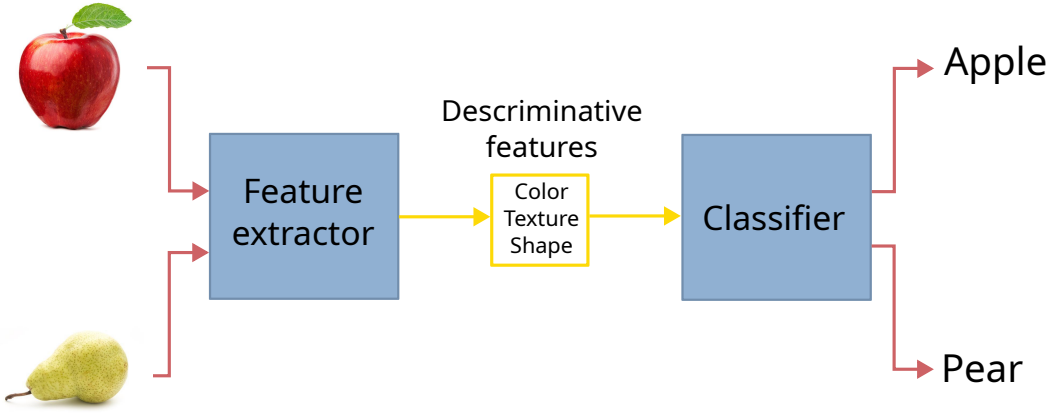
```python
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from skada import OTMappingAdapter
from skada import make_da_pipeline
from skada import PerDomain

estimator = make_da_pipeline(
    PerDomain(StandardScaler()),
    OTMappingAdapter(),
    SVC(),
)

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



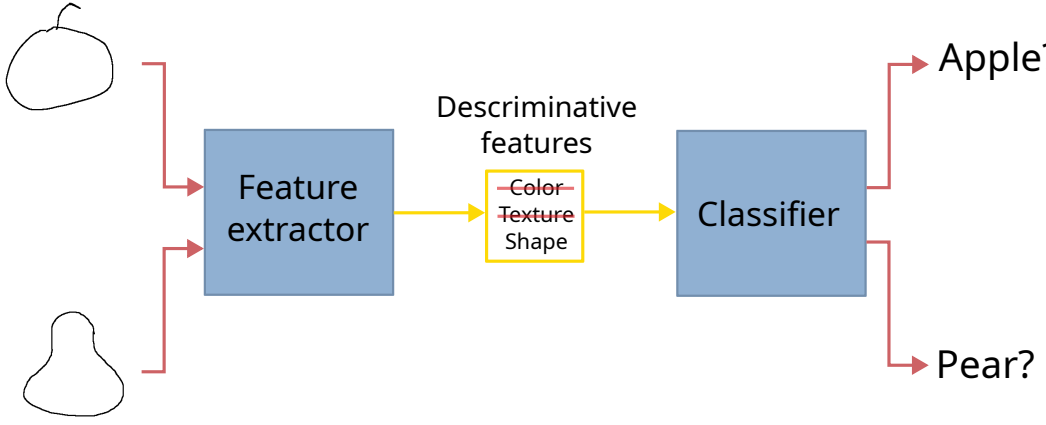
Deep Dive in SKADA



Deep Dive in SKADA



Deep Dive in SKADA



Deep Dive in SKADA

Feature extractor

Descriminative features

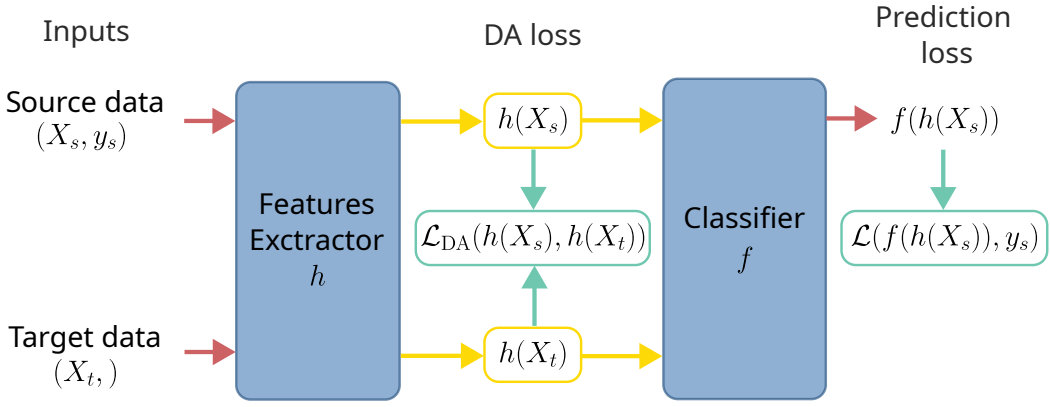
~~Color~~
~~Texture~~
Shape

Classifier

Apple?

Pear?

Deep Learning in DA



How to use SKADA: Deep DA with skorch

```python
from skorch import NeuralNetClassifier
from skada.deep.modules import ToyCNN

model = NeuralNetClassifier(
    ToyCNN(),
    batch_size=32,
    max_epochs=5,
    lr=1e-3,
)

model.fit(X_source, y_source)
```

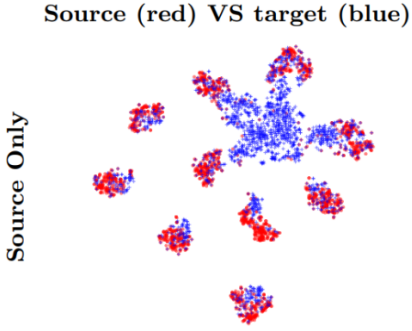


image from Damodaran et. al., 2018

How to use SKADA: Deep DA with skorch

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

model = DeepJDOT(
    ToyCNN(),
    batch_size=32,
    max_epochs=5,
    lr=1e-3,
    reg=1,
    layer_name="feature_extractor",
)

model.fit(X, y, sample_domain=sample_domain)
```

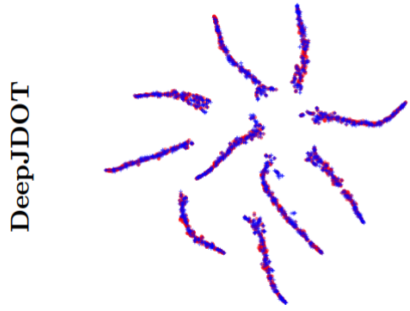


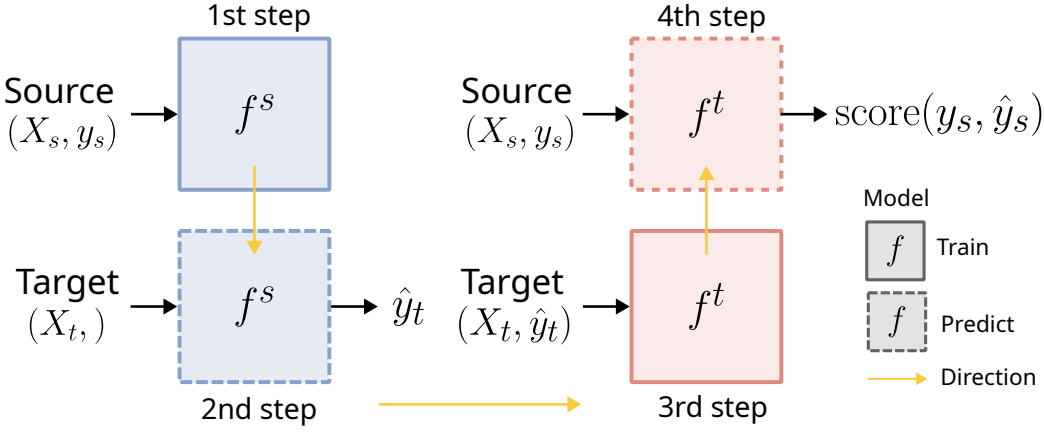
image from Damodaran et. al., 2018

Validation procedure in DA methods[1]

	Method	Validation Procedure
Reweighting	Density Reweight	**None**
	Discriminative Reweight	NA
	Gaussian Reweight	**None**
	KLIEP	Integrated CV
	KMM	**None**
	NN Reweight	**None**
	MMDTarS	CV
Mapping	Coral	NA
	OT mapping	**CV target/CircCV**
	Lin. OT mapping	NA
	MMD-LS	CV
Subsp.	SA	**2-fold CV on source**
	TCA	**Validation on target**
	TSL	**None**
Other	JDOT	Reverse CV
	OT label prop	NA
	DASVM	Circular Validation

[1] *Table adapted from Lalou et. al. 2025*

Circular Validation Procedure[1]



[1] *Bruzzone et. al. 2009*

How to use SKADA: DA Scorer

```python
from skada import EntropicOTMapping
from skada import CircularValidation

estimator = EntropicOTMapping(base_estimator=SVC())
cv = ShuffleSplit(n_splits=5, test_size=0.3)

reg_e = [0.01, 0.03, 0.05, 0.08, 0.1]

grid_search = GridSearchCV(
    estimator,
    {"entropicotmappingadapter__reg_e": reg_e},
    cv=cv,
    scoring=CircularValidation(),
)

grid_search.fit(X, y, sample_domain=sample_domain)
```

Conclusion and future works

- SKADA: **20+ shallow DA methods** and **10+ deep DA methods** .
- **Sklearn-like interface** for easy use and integration with existing workflows.
- Still need test from the community to improve the library: **contributions and/or experience feedbacks are welcome!**
- Future works: **more methods** and **more tutorials** !

SKADA GITHUB

SKADA

Skada-Bench Paper