From theoretical to practical transfer learning: the ADAPT library

Antoine de Mathelin, Francois Deheeger, Mathilde Mougeot, Nicolas Vayatis

From theoretical to practical transfer learning: the ADAPT library

Antoine de Mathelin, Francois Deheeger, Mathilde Mougeot, Nicolas Vayatis

Abstract In traditional machine learning, the learner assumes that the training and testing datasets are drawn according to the same distribution. However, in most practical scenarios, the two datasets are drawn according to two different distributions, the source distribution and the target distribution. In this context, the use of classical machine learning algorithms often fails as models trained on the source data provide poor performances on the target data. To solve this problem, many transfer learning techniques have been developed following one of the three main strategies: parameter-based transfer, instance-based transfer and feature-based transfer. The choice of the appropriate strategy is mainly determined by the nature of the shift between the source and target distributions. For example, to deal with the problem of sampling bias, when part of the population is over- or under-represented in the training set, instance-based approaches are useful to adequately reweight the source data in the training phase. If the shift is caused by a change in data acquisition, such as sensor drift, feature-based methods help to correct the shift by learning a common feature representation for the source and target data. For a real application, it is really a challenge to choose in advance the best transfer learning strategy and one often needs to evaluate different models in practice. As the different transfer methods were introduced by various contributors, no common framework is today available for a rapid development. To tackle this issue, we propose a Python library for transfer learning: ADAPT (Awesome Domain Adaptation Python Toolbox), which allows practitioners to compare the results of many methods on their particular problem.

Antoine de Mathelin
Michelin, Clermont-Ferrand, e-mail: antoine.de-mathelin-de-papigny@michelin.com

Francois Deheeger
Michelin, Clermont-Ferrand, e-mail: francois.deheeger@michelin.com

Mathilde Mougeot
Centre Borelli ENS Paris-Saclay, Gif-sur-Yvette, e-mail: mathilde.mougeot@ens-paris-saclay.fr

Nicolas Vayatis
Centre Borelli ENS Paris-Saclay, Gif-sur-Yvette, e-mail: nicolas.vayatis@ens-paris-saclay.fr

ADAPT is an open-source library providing the implementation of several transfer learning methods. The library is suited for scikit-learn estimator objects (objects which implement fit and predict methods) and tensorflow models. It allows to evaluate very easily the benefits of transfer learning methods on real data. In this chapter, we propose to illustrate the different features of the ADAPT library on both synthetic and real datasets.

1 Introduction

Models of machine learning are often deployed on data distributions different from the distribution used for training. This scenario is characterized by a distribution shift that often induces a degradation on model performances. Examples of shift appear in medical applications such as survival prediction [4] or cancer detection [27] as well as in industrial applications [30, 40, 41]. To correct this shift, transfer learning or domain adaptation methods have been developed recently. Their goal is to transfer information from a source domain where a lot of data are available to a target domain where few or no data are available [38, 46]. In recent years, these methods have been widely developed and used in a large number of applications such as image segmentation and classification [22, 58], sentiment analysis [12] or speech recognition [63, 68]. At the same time, the interest of the industry for this type of method is growing and transfer learning has already been used, in this area, in computer design [30], aircraft design [41] and photovoltaic design [29].

However, the deployment of transfer methods in industry is not obvious. Most of transfer methods are based on very precise assumptions which are hard to check in practice. For example, some methods consider the *covariate shift* hypothesis which assumes that the labeling functions are the same on both domains [27, 64]. Other methods consider the *conditional shift* or *hidden covariate shift* hypothesis which assumes that the labeling function of both domain are matching after a specific transformation of the input features [14, 22]. Verifying these assumptions in real use-cases is an open problem [20, 26, 51]. When practitioners face a new application, it is then particularly difficult to choose the appropriate method. In practice, one would like to be able to evaluate different transfer methods in order to select the method suited for the problem at hand. Therefore, in order to reasonably consider the deployment of transfer models in industry, it is necessary to facilitate the access to the different existing methods and the comparison in a common framework. Moreover, practitioners need to be guided in their choice of transfer method based on practical considerations derived from the problem at hand.

Some implementations of transfer methods are already publicly available in open source repositories as MDD [79] [1] or WDGRL [62] [2]. However, most of the available implementations have been developed to allow the reproduction of the experiments

[1] https://github.com/thuml/MDD

[2] https://github.com/RockySJ/WDGRL

presented in the corresponding research works. Thus, it often requires extra efforts to use these implementations for other problems with other machine learning models or other network architectures. Moreover, the different available implementations use different formalisms and libraries (Scikit-learn [50], Tensorflow [3] or Pytorch [49]) which makes particularly difficult the comparison of the methods on a common basis.

To avoid these difficulties, some Python libraries have been developed in order to group methods under the same formalism. One can find for instance TLlib [28], ADA [69] or Salad [60]. However, these three repositories propose nowadays only Pytorch implementations. The most developed repository, TLlib, proposes essentially deep learning methods and is intended for Pytorch users.

Facing these challenges, and motivated by real world problems, we propose an open-source Python library: ADAPT[3] to facilitate the access to transfer methods for a large public, including industrial players. For this purpose, we propose in ADAPT a wide range of methods, compatible with scikit-learn and tensorflow objects. The methods are implemented with a "fit" and a "predict" functions which can take any type of dataset and estimator. Moreover, the deep methods have all the advantages of the Keras models [10]: speed of calculation, large possibility of modifying the hyper-parameters and monitoring the training, large choice of loss functions which allows to use the methods for both classification and regression tasks. Finally, a detailed documentation with a user guide helps to quickly select the appropriate transfer method based on practical considerations[4].

In this work we present the different features of the ADAPT library and show how they allow to answer a wide range of transfer problems that can be encountered in the industry. The organization of the paper is as follows: We first recall the transfer learning framework and the variety of transfer scenario that can be encountered. We then describe the main guidelines of the ADAPT library and we provide examples of its installation and usage. We show how it is possible, from a user point of view, to handle different transfer problems with the ADAPT library. Finally, we present a benchmark of transfer methods on several transfer learning datasets.

2 Transfer Learning Framework

The transfer learning framework is characterized by the presence of a source domain (Q, f_Q) and a target domain (P, f_P) where Q, P define two distributions on the input space and f_Q, f_P are two labeling functions returning the labels for each domain: $y = f_Q(x)$ for any $x \sim Q$ and $y = f_P(x)$ for any $x \sim P$. f_Q and f_P are also denoted in their probabilistic form $f_Q = P^Q_{Y|X}$ and $f_P = P^P_{Y|X}$. The main assumption for transfer learning is that the source and target marginal distributions differ: $P \neq Q$.

[3] https://github.com/adapt-python/adapt

[4] https://adapt-python.github.io/adapt

2.1 Main assumptions of domain adaptation

The shift between distributions $P \neq Q$ can be caused by different factors. For example, in image processing, the source distribution Q can be the distribution of a set of images on which it is easy to obtain labels, either because they are synthetically generated (as GTA images [78]) or because they are in a rich semantics (as Amazon images [56]). In both cases the images are characterized by a shape close but different from the one of real target images. The shift can also occurs due to technological changes, like for example in design space exploration, where the learned models are applied on unknown design spaces [30]. In the same way, shifts also appear when one is aiming to generalize a model to different products or places, for example between images recorded by different traffic cameras [80, 17]... There exist also sub-population shifts which occur, for example, when one applies a model trained on a large domain (as Imagenet [18]) on a specific sub-domain [45], or when one tries to correct existing biases in the training dataset when one class is more represented than others for example [27, 11, 42]. Finally, the shift can also be the result of a change in the acquisition of the input data caused by sensor drift for instance [14].

To characterize all previous cases, several theoretical assumptions on the nature of the domain adaptation shifts are described in the literature and different types of transfer methods may be then advised.

- **Covariate Shift** assumes that the target and source labeling functions are the same ($f_Q = f_P$); from a probabilistic point of view that the conditional distributions are similar ($P^Q_{Y|X} = P^P_{Y|X}$). This assumption is generally made in cases of sub-population shift or sample bias. In these cases, it is often considered that P and Q have the same support. The correction of the marginal distribution difference is often proposed through importance weighting [27, 64].
- **Hidden Covariate Shift** [5] or Conditional Shift [76] considers that the labeling functions matches under a specific transformation of the input features ($\exists\, \phi,\ f_Q = f_P \circ \phi$) which is related to the conditional shift assumption $P^Q_{X|Y} \neq P^P_{X|Y}$ and $P^Q_Y = P^P_Y$. This kind of hypothesis is mainly considered for shifts between real and synthetic data [22, 71] or in sensor drift [14].
- **Label Shift**, **Target Shift** or **class imbalance** assumes that $P^Q_Y \neq P^P_Y$ [76, 32]. It is generally supposed besides that $P^Q_{X|Y} = P^P_{X|Y}$ which refers to class imbalance problems. The case where an entire class is not present in the target dataset is referred as partial domain adaptation [8, 7]. In general, target shift problem also occurs with conditional shift, we then speak of generalized target shift [32].
- **Joint Probabilistic Shift** considers the cases where shifts exist between both the marginal and the conditional distributions: $P^Q_X \neq P^P_X$ and $P^Q_{Y|X} \neq P^P_{Y|X}$. This assumption is encountered in a large amount of practical applications [23]. Thus, transfer methods which focus on correcting these two shifts, generally provide improved results over methods correcting only the marginals [36]. One difficulty encountered by these methods is the estimation of the target conditional

distribution $P^P_{Y|X}$. Some of the methods need a small labeled target sample to make this estimation [44, 43, 57, 70] others use pseudo-labels [13, 23, 35, 36, 79].

2.2 Classification of transfer methods based on user needs

One limitation of the definition of transfer problems linked to the previous assumptions is the difficulty to evaluate the corresponding hypotheses in real scenarios [20, 26, 51]. For many cases, it is not obvious for one practitioner to know the nature of the shift between the training dataset and the data on which the model is applied. Moreover, in practice, many methods are not strictly limited by their assumptions. For instance, a transfer strategy dedicated to hidden-covariate shift may still work in the covariate or the target shift settings. Practitioners have often a different point of view which mainly relies on the available source or target data. They will prefer to follow a more practical transfer learning pathway directly linked to the encountered operational constraints of their problem as follows:

- **Source free** assumes that the source data are not available. The learner has only access to a pre-trained source model. This occurs typically when a pre-trained network trained on Imagenet is reused to get deep features for another task [9, 45, 33].
- **Multi-source** offers source data supposed to belong to several different sources [39, 25, 54]. For example, the source data can be composed of images from several cameras watching the traffic [80].
- **Homogeneous and Heterogeneous Transfer** refer respectively to the cases where the source and target input feature spaces are the same or not. For instance, the source and target datasets can be composed of images of same resolution (homogeneous) or different resolution (heterogeneous).
- **Supervised**, **Semi-supervised**, **Unsupervised Domain Adaptation** (SDA, SSDA, UDA): in the supervised setting, only a few labeled target data are available [43, 43, 17] and a lot of unlabelled target data in the unsupervised setting [47, 34, 71, 79]. In the semi-supervised setting, both kind of data are available [57, 70].

Following our experience of the end-users, we design a new presentation of transfer learning needs as presented in Figure 1. This diagram is built from a user point of view, where the choice of transfer method is driven by the characteristics of the available datasets. It should be underlined that it is difficult for a practitioner to evaluate the presence of target shift in a dataset, but anyone can easily check for the availability of source and/or target labels.

The classification of transfer methods that we propose in Figure 1 is provided with the ADAPT documentation to allow the user to quickly identify which type of method could be used in a specific case. This contributes to the main goal of the ADAPT library, which is to help the practitioners find the transfer learning method adapted to their needs.

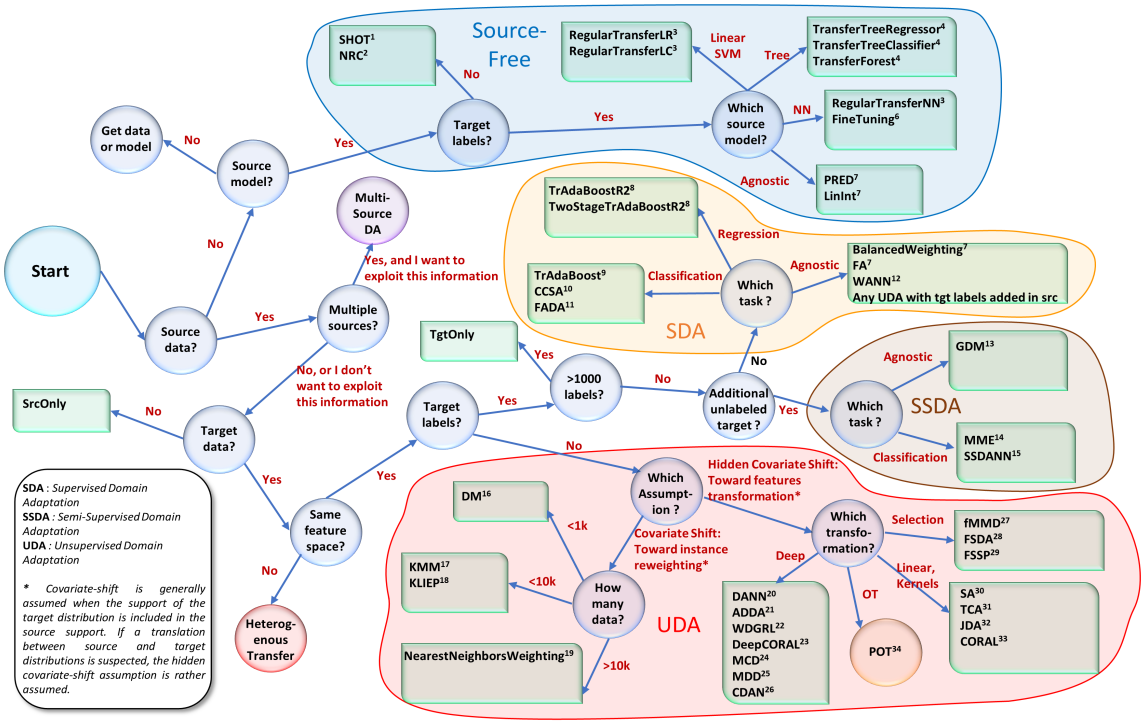

Fig. 1 A classification of transfer methods from a user perspective. Corresponding references: [1][33], [2][75], [3][9], [4][61], [6][45], [7][16], [8][48], [9][15], [10][44], [11][43], [12][17], [13][12], [14][57], [15][70], [16][38], [17][27], [18][64], [19][37], [20][22], [21][71], [22][62], [23][66], [24][58], [25][79], [26][35], [27][72], [28][67], [29][31], [30][19], [31][47], [32][36], [33][65], [34][21]

The proposed classification does not pretend to exhaust the classification of transfer learning methods but to highlight the four main categories: Source-Free, SDA, SSDA and UDA. This corresponds to a data driven classification which is easy to identify for any transfer problem.

2.3 Transfer learning strategies

As previously mentioned, from a user point of view, the choice of methods is essentially guided by the access or not to labeled/non-labeled source/target data. However, for more experimented users, this decision criterion may be completed with the choice of the appropriate strategy between the instance-based and feature-based strategies. It should be underlined that, for certain transfer problems, a wrong choice of strategy can cause negative transfer [17].

- **The instance-based strategy** consists in correcting the shift between source and target distribution by reweighting the source instances in the loss during training. Source data are weighted depending on their relation to the target data. This strategy is mostly used in covariate-shift problems and often for regression tasks. Most methods assume that the supports of the marginal distributions are the same or at least that the target distribution is included in the source distribution (sub-population shift). These methods are also useful under the target shift assumption to correct the imbalance between classes.
- **The feature-based strategy** consists in finding a new representation of the input features in which the source and target distributions match. This representation can be obtained with different transformation as optimal transport, feature reduction or deep encoding. This strategy is often used with the hidden covariate-shift assumption. Moreover, it is assumed that the transformation needed to match the distributions is not arbitrary but has a certain regularity. This type of method is mostly used in classification. It can be used to solve sensor drift problems for example or when the brightness or the background of images is different from one domain to another. It is also useful when the source domain is a specific representation of the target domain, like adaptation cases between simulated data and real data.

2.4 Hyper-parameters selection

Transfer methods are very popular for image recognition problems [22, 71, 70, 58, 35]. Results are very promising for example on the product classification problem using Amazon images [22] and similarly in segmentation, excellent performances are also obtained [58]. These applications show that the performance of a source

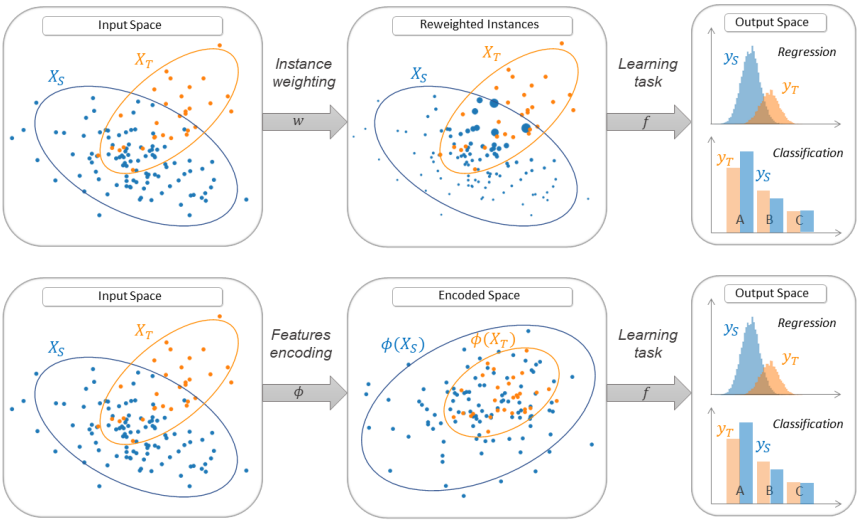

Fig. 2 Illustration of the instance-based (top) and feature-based (bottom) strategies.

model can be really improved on a target domain without the expense of any target labels.

However, to apply these methods for real deployed applications in industry, one is facing several challenges. The most promising unsupervised methods (DANN, CDAN, MDD...) appear to be very sensitive to the hyper-parameters selection and often lead to negative transfer [55, 17].

The choice of hyper-parameters appears to be particularly crucial for transfer learning methods because one of the goal is to learn a model on a target domain for which few or no labels are available. The classical cross-validation methods are then not suitable because of the difficulty of computing an error on the target domain in the absence of labels. This problem of hyper-parameters selection is, as far as we know, little reported in the literature, although some methods and metrics have been proposed, like the reverse validation [22] or the J-score [64].

3 ADAPT features

The large number of existing methods to deal with domain adaptation and the variety of shifts that can be encountered in real problems makes it difficult to select the appropriate transfer method and hyper-parameters. Most of the time, the learner faces to try several methods to evaluate which one fits best.

The ADAPT library is designed to answer this problem by allowing to evaluate quickly, in the same framework, several methods. Moreover it proposes unsupervised

metrics (J-score [64], reverse-validation [22], linear discrepancy [38]...) allowing to evaluate the capacity of a method to adapt well without using labels in the target domain. These metrics are particularly useful for the hyper-parameter selection and early stopping in the unsupervised framework.

3.1 ADAPT Guideline

ADAPT is designed to ease the use of transfer methods on a variety of problems. To achieve this goal, all transfer tools are implemented in the `scikit-learn` style with a `fit` and `predict` methods [50]. As `scikit-learn` objects, the hyper-parameters can be given at the instantiation step. Thus, grid-search can be applied with the `GridSearchCV` tool of `scikit-learn`. In a model deployment perspective, the objects can be duplicated and easily saved, either in the `pickle` or the `tensorflow` format for deep learning methods.

ADAPT is the only transfer library compatible with both `scikit-learn` and `tensorflow` objects. In comparison to other transfer repositories, ADAPT fundamentally differs by its "user friendly" approach providing a detailed documentation with small examples for each method and a high test coverage.

TLlib [28] is, as far as we know, the most developed transfer library which can be compared to ADAPT. It makes a great work of proposing almost 40 methods of domain adaptation with their corresponding documentation. However, at the difference of ADAPT, TLlib is mostly dedicated to researcher and is not designed thanks to end-user needs:

- TLlib implements only deep learning methods. Many industrial methods often rely on other types of model as Gaussian Processes [52] or Decision Trees [6] as industrial problems are often characterized by small sample sizes and needs of interpretability [74].
- The examples provided in TLlib present mainly experiments on image datasets published in research papers. Understanding how to use other datasets and network architecture than the ones already provided is not straightforward as no minimal examples are given in the documentation.
- Training the methods from TLlib requires the user to be familiar with `pytorch` as the code of the training loop had to be written by the user. In ADAPT, the practitioner only needs to call the `fit` method on the desired dataset.

3.2 ADAPT for real applications

To illustrate the use of ADAPT for industrial needs, we consider several real cases already presented in the literature:

3.2.1 Transfer between group of patients in falls detection

A floor provider company aims at developing floor sensor to detect falls of elderly people [42]. They first design a model trained with labeled data recorded from young people simulating falls in a controlled environment. However, when applied to real falls, the model fails to generalize. In this case, a small sample of real labeled data along with the source model (trained on simulated falls) are available. In order to solve this domain adaptation issue, the User-Guide in Figure 1 advises to use a source-free method as proposed by Minvielle et al. [42] with an adaptation of the source-free method TransferTree [61].

3.2.2 Generalization to new lines in tire design

A tire company faces the issue of generalization in design space exploration [40]. They observed that a model trained to predict the tire performances is limited to the domain defined by the training data and does not generalize well to unseen regions of the design space. They consider the case where a few labeled data is available in a target design region of potential innovative products. As the labeled data from previous developed products are also available, the User-Guide in Figure 1 suggests using supervised methods (SDA). In this problem, the authors of [40] consider both UDA and SDA cases, and show improvements with the regression SDA method TrAdaBoostR2 [48].

3.2.3 Adaptation between houses for non-intrusive load monitoring

A power company aims at monitoring the consumption of their clients [53]. They want to identify the consumption of one particular item (the washing machine) knowing the temporal total consumption of one house. This task faces a domain shift issue between different houses (different habits lead to different consumption patterns). The company has access to a labeled set for several houses (fully monitored with sensors) and wants to adapt a model for each house. Following Figure 1 pathway, as no labels are available for a new house, the problem should be treated with UDA methods. In these particular cases, Richard et al. [53] consider a hidden covariate-shift assumption: they suppose that the shift between the consumption of houses is a kind of translation, they then derive a deep feature-based methods to handle the problem. Moreover, as the source data set is composed of data from several houses, they propose a multi-source method.

3.2.4 Unsupervised domain adaptation for predicting stars formation history

In astrophysics, one of the main goal is to model the formation of the universe from the BigBang to our days based on the observation of stellar radiations which come

on earth [59]. To find the relationship between the stellar radiations (which can be observed) and their corresponding star formation histories, one cannot use real labeled data as no complete star formation can be observed. Thus, astrophysicists use simulations to produce pairs of simulated radiation and star formation history and then build machine learning models to learn the relationship between both. However, to use the model on real stellar radiations, the shift that exists between simulated and real data should be corrected. This problem is a purely UDA issue. To deal with it the authors of [59] consider the instance-based UDA approach KLIEP [64] which choice is motivated by the fact that the support of the target distribution is included in the support of the source distribution.

3.3 ADAPT installation and usage

As far as we know, the ADAPT library is for now, the only well provided library of transfer learning available on Pypi [1]. The installation simply consists in running the following command in a shell: `python3 -m pip install adapt` or `pip install adapt` in an Anaconda environment [2].

To use ADAPT in a Python environment, the library can be imported using: `import adapt`. Figure 3 presents an example of usage on a simple case of binary classification in one dimension proposed in [38]. In this problem, the source and target distributions are two Gaussians centered respectively in -1 and +1 with a standard deviation of 2. The labeling function is common for both and is equal to 1 in $[-1, 1]$ and -1 elsewhere. We use logistic regression as base estimator. We present in Figure 3 how to use the Kernel Mean Matching (KMM) method to solve this problem. The result of this synthetic experiment is given in Figure 4. KMM is a UDA approach for which hyper-parameters has to be set. Here we show how to select the bandwidth parameter of the gaussian kernel with a grid-search using the J-score [64] (cf Figure 5).

```python
# Import standard librairies
import numpy as np
from sklearn.linear_model import LogisticRegression

# Import KMM method form adapt.instance_based module
from adapt.instance_based import KMM

np.random.seed(0)

# Create source dataset (Xs ~ N(-1, 2))
# ys = 1 for ys in [-1, 1] else, ys = 0
Xs = np.random.randn(1000, 1)*2-1
ys = (Xs[:, 0] > -1.) & (Xs[:, 0] < 1.)

# Create target dataset (Xt ~ N(1, 2)), yt ~ ys
Xt = np.random.randn(1000, 1)*2+1
yt = (Xt[:, 0] > -1.) & (Xt[:, 0] < 1.)

# Instantiate and fit a source only model for comparison
src_only = LogisticRegression(penalty="none")
src_only.fit(Xs, ys)

# Instantiate a KMM model : estimator and target input
# data Xt are given as parameters with the kernel parameters
adapt_model = KMM(
    estimator=LogisticRegression(penalty="none"),
    Xt=Xt,
    kernel="rbf",  # Gaussian kernel
    gamma=1.,      # Bandwidth of the kernel
    verbose=0,
    random_state=0
)

# Fit the model.
adapt_model.fit(Xs, ys);
```

Fig. 3 Example of usage on a 1D synthetic dataset.

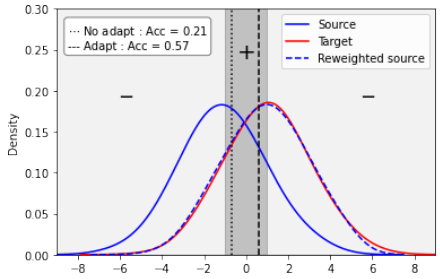

Fig. 4 Plotting results of the synthetic 1D experiment (the Python code used to produce the figure is given in Appendix). The dotted and dashed lines are respectively the class separation of the "source only" and KMM models. Note that the predicted positive class is on the right of the dotted line for the "source only" model but on the left of the dashed line for KMM. The input distributions in blue and red are smooth approximations of their corresponding empirical distributions. We observe that the reweighted source distribution is very close to the target distribution which induces a better target classification of accuracy 57 % instead of 21%.

```python
# Import standard librairies
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Import KMM and the unsupervised metric j_score
from adapt.instance_based import KMM
from adapt.metrics import j_score, make_uda_scorer

np.random.seed(0)

# Instantiate KMM model with a random gamma parameter
adapt_model = KMM(
    estimator=LogisticRegression(penalty="none"),
    Xt=Xt,
    kernel="rbf",  # Gaussian kernel
    gamma=0.,      # Bandwidth of the kernel
    verbose=0,
    random_state=0
)

# Create a score function from the j_score metric and Xs, Xt
score = make_uda_scorer(j_score, Xs, Xt)

# Launch the gridsearch on three gamma parameters [0.5, 1., 2.]
gs = GridSearchCV(adapt_model, {"gamma": [0.5, 1., 2.]},
                  scoring=score,
                  return_train_score=True, cv=5, verbose=0)
gs.fit(Xs, ys)

# Print results (the j_score is given in "train_score")
keys = ["params", 'mean_train_score', 'std_train_score']
results =  [v for k, v in gs.cv_results_.items() if k in keys]
best = results[1].argmin()
print("Best Params %s -- Score %.3f (%.3f)"%
      (str(results[0][best]), results[1][best], results[2][best]))
print("-"*50)
for p, mu, std in zip(*results):
    print("Params %s -- Score %.3f (%.3f)"%(str(p), mu, std))
```

```
Best Params {'gamma': 1.0} -- Score -0.021 (0.003)
--------------------------------------------------
Params {'gamma': 0.5} -- Score 0.002 (0.004)
Params {'gamma': 1.0} -- Score -0.021 (0.003)
Params {'gamma': 2.0} -- Score -0.015 (0.003)
```

Fig. 5 Example of grid-search usage on the synthetic dataset with the KMM model. To select the appropriate bandwidth "gamma" of the kernel used in KMM, the unsupervised metric "J-score" is computed between the reweighted source distribution and the target distribution for each gamma in [0.5, 1, 2]. The configuration gamma=1 gives the best J-score.

4 Application / Illustration

This section presents the use and the results of the different ADAPT methods under different transfer scenarios: the Supervised Domain Adaptation scenario on the CityCam dataset [77] and the Unsupervised Domain Adaptation scenario on the MNIST vs MNIST-M dataset [22] and the Office dataset [56].

4.1 Supervised and Semi-Supervised Domain Adaptation

To compare the transfer methods in the SDA and SSDA setting, we consider the transfer problem on the CityCam dataset [77] proposed in [17]. In this experiment, the images from three traffic video cameras from CityCam are used as source data whereas the images from a fourth camera are used as targets. The task consists in predicting the number of cars appearing on the image (see Figure 6). A small number of target labels are available along with the source labeled data and the target unlabeled data. We follow the settings from [17], considering the same architecture of neural networks, optimizer and hyper-parameters. We conduct each experiment 5 times to compute standard deviation for the resulting target mean absolute error (MAE). We conduct four different experiments, each corresponding to one of the four cameras selected as target; each camera is referenced with a number: 495, 253, 511 or 572. We consider the transfer learning methods: KLIEP, KMM, WANN, DANN, MDD, ADDA, DeepCORAL and CORAL. The UDA methods are used in the SSDA setting by adding the target labeled data to the source dataset. The results are reported in Figure 7.

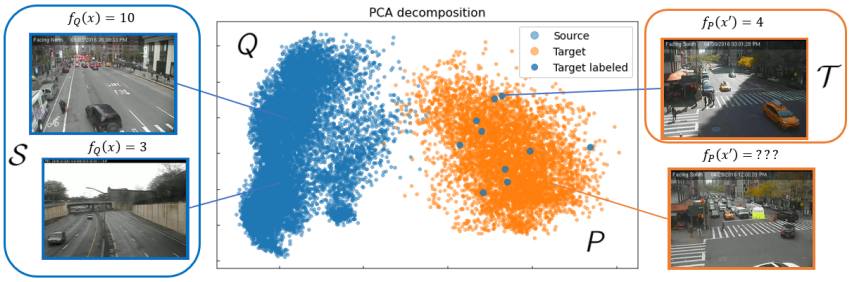

Fig. 6 Illustration of the CityCam experiment [77]. The two first PCA components of the input space are represented.

We observe in Figure 7 that the target error is decreasing with the number of labeled data, particularly for instance-based approaches which outperform the feature-based methods. In this case, as the learner has access to several target labels, using an

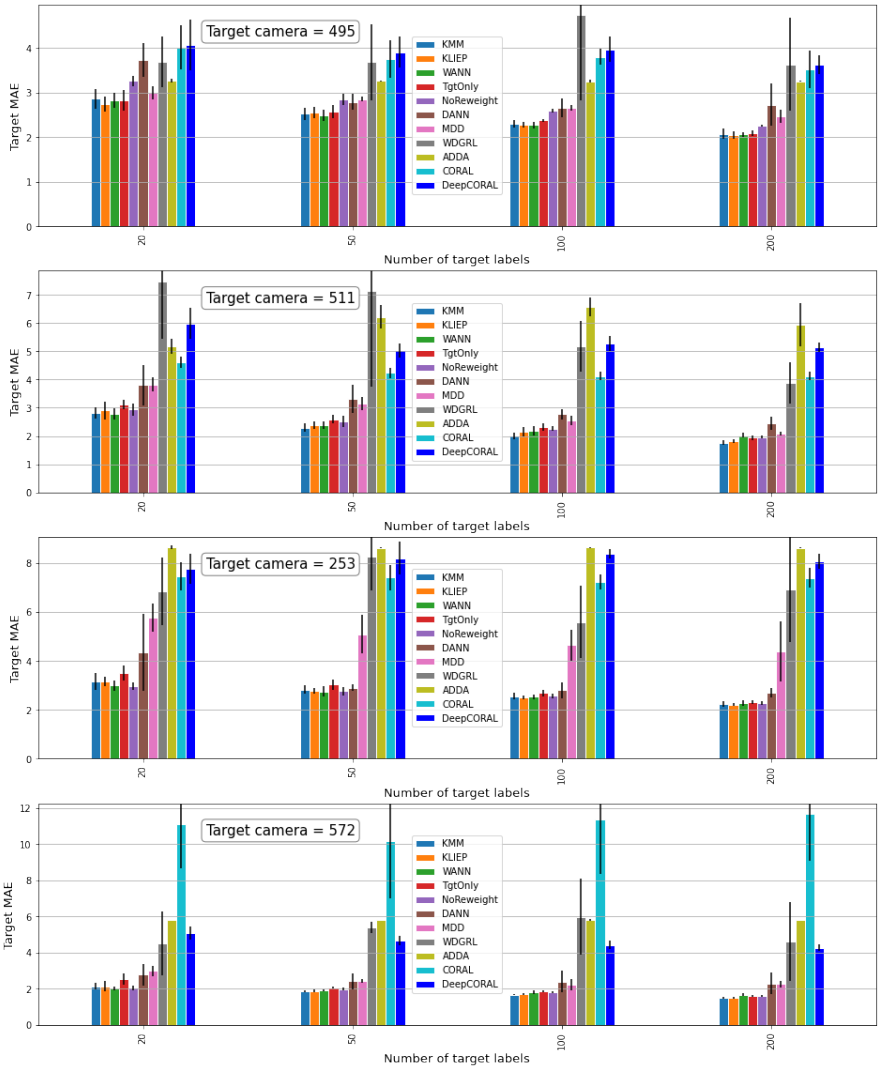

Fig. 7 Target mean absolute error for the transfer between traffic cameras from CityCam. The experiments are conducted for different numbers of target labeled data. TgtOnly refers to the model trained with labeled target data only and NoReweight to the model trained with all labeled data without transfer.

instance-weight strategy to give more importance to the target labeled data is well suited.

4.2 Unsupervised Domain Adaptation

To compare the transfer methods under the unsupervised domain adaptation setting, we consider two public datasets: MNIST vs MNIST-M [22] and Office [56].

4.3 MINST vs M-MNIST

In the MNIST vs MNIST-M experiment proposed in [22], the task consists in predicting the value of the digits in the images (see Figure 8). Here, we suppose that no target labels are available. We use the same network architecture and optimizer than [22] and 3000 data from each data set. The hyper-parameters used for each transfer methods are reported in Appendix. The experiments are conducted 5 times to compute standard deviation of the accuracy on the MNIST-M data for each method. The results are reported in Table 1.

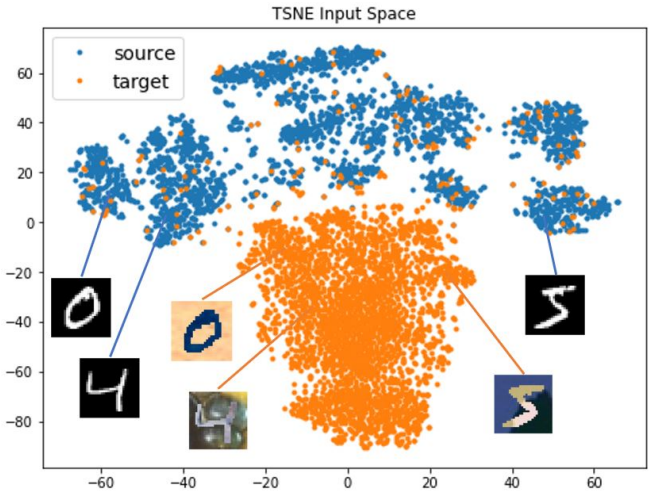

Fig. 8 Illustration of the MNIST to MNIST-M experiment. The two first tSNE [73] components of the input space are represented.

We observe in Table 1 that the feature-base methods outperform the instance-based ones. In particular, DANN, ADDA and CDAN+E improve substantively the accuracy on the target domain without using any target labels. In this scenario, indeed, as the supports of the target and source distribution differ, it is recommended to consider the feature-based strategies.

Table 1 Target Accuracy for the transfer from MNIST to MNIST-M. SrcOnly refers to the model trained without transfer.

SrcOnly	MCD	MDD	CDAN+E	CORAL	WDGRL
0.334 (0.036)	0.402 (0.011)	0.485 (0.017)	0.52 (0.026)	0.311 (0.042)	0.457 (0.031)
KMM	KLIEP	DANN	ADDA	DeepCORAL	
0.297 (0.016)	0.338 (0.038)	0.519 (0.033)	**0.593 (0.015)**	0.348 (0.059)	

4.4 Office

The office dataset [56] is composed of photos of office items (see Figure 9). The data come from three different domains: `amazon`, `webcam` and `dslr`. In the experiment conducted here, we look at the adaptation from the `amazon` domain to the `webcam` domain. As the labels are easy to access in the `amazon` domain (the class of the item is provided in the description of the object) we consider it as the source domain whereas `webcam` is the target domain. We consider the experimental setup from [79] using a fine-tuned ResNet50 network as encoder [24].

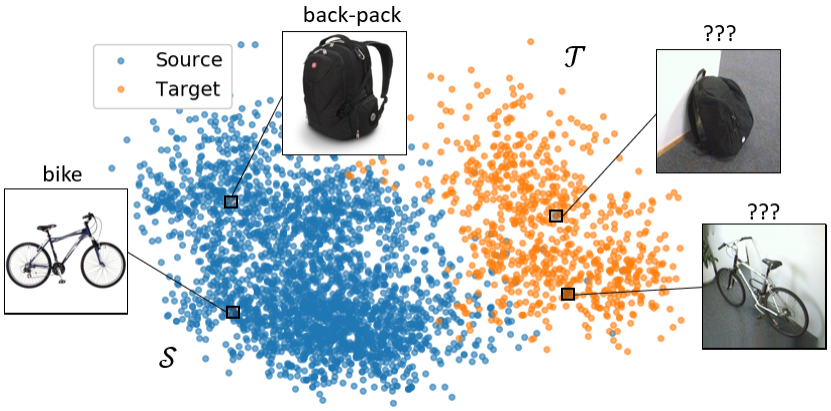

Fig. 9 Illustration of the Office experiment [56]. The two first PCA components of the input space are represented.

We try several unsupervised domain adaptation models from the ADAPT package: DANN, ADDA, MDD, DeepCORAL and CDAN; all are feature-based methods. We compare their results with the ones of the baseline SrcOnly trained with source data only. The evolution of the target accuracy through the epochs is reported in Figure 10.A along with the evolution of the linear discrepancy [38], an unsupervised metric which evaluates the similarity between the encoded source and target distributions. We observe that DANN and WDGRL significantly improve the accuracy on the target domain compared to the SrcOnly baseline. We also observe that the linear

discrepancy is very small for these two methods, which shows that unsupervised metric can help to discriminate between methods. Notice, however, that CDAN has a small discrepancy but does not improve the target accuracy. This, indeed, can happen if the encoder learns a wrong pairing between source and target data and projects target data on source data from the wrong class. Finding a way to avoid this effect is an open problem in unsupervised domain adaptation.

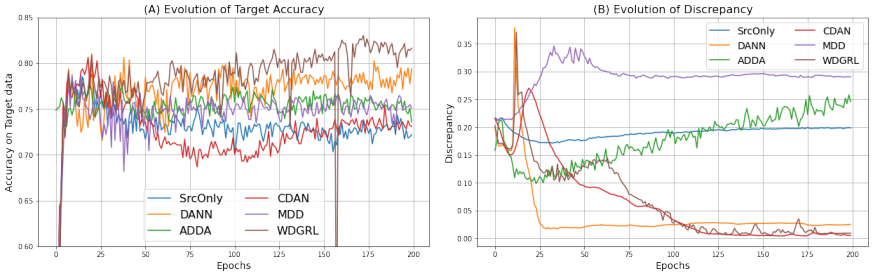

Fig. 10 Results of the Office experiments. Figure (A) presents the evolution of the target accuracy in function of the number of epochs. Figure (B) presents the evolution of the linear discrepancy [38] between the encoded source and target domains in function of the number of epochs.

5 Conclusion

This work presents ADAPT, a library which implements, in a pythonic fashion, a large collection of transfer learning methods. This library helps to compare several methods on real problems. ADAPT appears to be an efficient tool for practitioners to find the right transfer method to use for a particular problem. Since its creation, ADPAT has been used in a large range of fields from tire design [40] to astrophysics [59].

References

[1] Python package index - pypi.
[2] Anaconda software distribution, 2020.
[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal

Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] Alexis Bellot and Mihaela van der Schaar. Boosting transfer learning with survival data from heterogeneous domains. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 57–65. PMLR, 16–18 Apr 2019.

[5] Victor Bouvier, Philippe Very, Céline Hudelot, and Clément Chastagnol. Hidden covariate shift: A minimal assumption for domain adaptation. *arXiv preprint arXiv:1907.12299*, 2019.

[6] L Breiman, JH Friedman, R Olshen, and CJ Stone. Classification and regression trees. 1984.

[7] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Partial transfer learning with selective adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2724–2732, 2018.

[8] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.

[9] Ciprian Chelba, Jorge Silva, and Alex Acero. Soft indexing of speech content for search in spoken documents. *Comput. Speech Lang.*, 21(3):458–478, July 2007.

[10] Francois Chollet et al. Keras, 2015.

[11] Corinna Cortes and Mehryar Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519, 2014.

[12] Corinna Cortes, Mehryar Mohri, and Andrés Muñoz Medina. Adaptation based on generalized discrepancy. *J. Mach. Learn. Res.*, 20(1):1–30, January 2019.

[13] Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 3730–3739, 2017.

[14] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016.

[15] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, volume 227, pages 193–200, 01 2007.

[16] Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[17] Antoine de Mathelin, Guillaume Richard, Francois Deheeger, Mathilde Mougeot, and Nicolas Vayatis. Adversarial weighting for domain adaptation in regression. *arXiv preprint arXiv:2006.08251*, 2020.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[19] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 2960–2967, 2013.

[20] Clément Feutry, Pablo Piantanida, Florence Alberge, and Pierre Duhamel. A simple statistical method to detect covariate shift. In *XXVIIème Colloque francophone de traitement du signal et des images (Gretsi 2019)*, 2019.

[21] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[22] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, January 2016.

[23] Te Han, Chao Liu, Wenguang Yang, and Dongxiang Jiang. Deep transfer network with joint distribution adaptation: a new intelligent fault diagnosis framework for industry application. *ISA Transactions*, 08 2019.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Judy Hoffman, Mehryar Mohri, and Ningshan Zhang. Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems*, pages 8246–8256, 2018.

[26] Xiaoyu Hu and Jing Lei. A distribution-free test of covariate shift using conformal prediction. *arXiv preprint arXiv:2010.07147*, 2020.

[27] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex J. Smola. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 601–608. MIT Press, 2007.

[28] Mingsheng Long Junguang Jiang, Bo Fu. Transfer-learning-library. https://github.com/thuml/Transfer-Learning-Library, 2020.

[29] Mine Kaya and Shima Hajimirza. Using bayesian optimization with knowledge transfer for high computational cost design: A case study in photovoltaics. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 59186, page V02AT03A015. American Society of Mechanical Engineers, 2019.

[30] D. Li, S. Wang, S. Yao, Y. Liu, Y. Cheng, and X. Sun. Efficient design space exploration by knowledge transfer. In *2016 International Conference*

on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pages 1–10, Oct 2016.

[31] Jingjing Li, Jidong Zhao, and Ke Lu. Joint feature selection and structure preservation for domain adaptation. In *IjCAI*, pages 1697–1703, 2016.

[32] Yitong Li, Michael Murias, Samantha Major, Geraldine Dawson, and David E Carlson. On target shift in adversarial domain adaptation. *arXiv preprint arXiv:1903.06336*, 2019.

[33] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020.

[34] M. Long, J. Wang, G. Ding, S. J. Pan, and P. S. Yu. Adaptation regularization: A general framework for transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1076–1089, May 2014.

[35] Mingsheng Long, ZHANGJIE CAO, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1640–1650. Curran Associates, Inc., 2018.

[36] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207, 2013.

[37] Marco Loog. Nearest neighbor-based importance weighting. In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2012.

[38] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.

[39] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1041–1048. Curran Associates, Inc., 2009.

[40] Antoine De mathelin, François Deheeger, Mathilde MOUGEOT, and Nicolas Vayatis. Handling distribution shift in tire design. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[41] A. T. W. Min, R. Sagarna, A. Gupta, Y. Ong, and C. K. Goh. Knowledge transfer through machine learning in aircraft design. *IEEE Computational Intelligence Magazine*, 12(4):48–60, 2017.

[42] Ludovic Minvielle, Mounir Atiq, Sergio Peignier, and Mathilde Mougeot. Transfer learning on decision tree with class imbalance. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1003–1010, 2019.

[43] Saeid Motiian, Quinn Jones, Seyed Mehdi Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *Proceedings of the 31st Interna-*

tional Conference on Neural Information Processing Systems, NIPS'17, page 6673–6683, Red Hook, NY, USA, 2017. Curran Associates Inc.

[44] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5715–5725, 2017.

[45] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[46] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[47] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

[48] David Pardoe and Peter Stone. Boosting for regression transfer. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, June 2010.

[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[51] Stephan Rabanser, Stephan Günnemann, and Zachary C Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. *arXiv preprint arXiv:1810.11953*, 2018.

[52] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[53] Guillaume Richard. *Transfer Learning methods for temporal data*. PhD thesis, Université Paris-Saclay, 2021.

[54] Guillaume Richard, Antoine de Mathelin, Georges Hébrail, Mathilde Mougeot, and Nicolas Vayatis. Unsupervised multi-source domain adaptation for regression. In Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020, Ghent, Belgium, September 14-18, 2020, Proceedings, Part I*, volume 12457 of *Lecture Notes in Computer Science*, pages 395–411. Springer, 2020.

[55] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, pages 1–4, 2005.

[56] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, page 213–226, Berlin, Heidelberg, 2010. Springer-Verlag.

[57] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko. Semi-supervised domain adaptation via minimax entropy. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8049–8057, 2019.

[58] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.

[59] Sabine Bellstedt Sankalp Gilda, Antoine de Mathelin and Guillaume Richard. Benefits of unsupervised domain adaptation in constraining galaxy star formation histories. Space and Artificial Intelligence, online conference, September 13th, 2021 Organized by CLAIRE and ESA, in association with ECML PKDD 2021, 2021.

[60] Steffen Schneider, Alexander S. Ecker, Jakob H. Macke, and Matthias Bethge. Salad: A toolbox for semi-supervised adaptive learning across domains, 2018.

[61] N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv. Learn on source, refine on target: A model transfer learning framework with random forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1811–1824, 2017.

[62] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[63] Yusuke Shinohara. Adversarial multi-task learning of deep neural networks for robust speech recognition. In *Interspeech*, pages 2369–2372. San Francisco, CA, USA, 2016.

[64] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1433–1440, Red Hook, NY, USA, 2007. Curran Associates Inc.

[65] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[66] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[67] Feng Sun, Hanrui Wu, Zhihang Luo, Wenwen Gu, Yuguang Yan, and Qing Du. Informative feature selection for domain adaptation. *IEEE Access*, 7:142551–142563, 2019.

[68] Sining Sun, Binbin Zhang, Lei Xie, and Yanning Zhang. An unsupervised deep domain adaptation approach for robust speech recognition. *Neurocomputing*, 257:79–87, 2017.

[69] Anne-Marie Tousch and Christophe Renaudin. (yet) another domain adaptation library, 2020.

[70] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4068–4076, 2015.

[71] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[72] Selen Uguroglu and Jaime Carbonell. Feature selection for transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 430–442. Springer, 2011.

[73] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[74] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.

[75] Shiqi Yang, Joost van de Weijer, Luis Herranz, Shangling Jui, et al. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. *Advances in Neural Information Processing Systems*, 34, 2021.

[76] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827, 2013.

[77] Shanghang Zhang, Guanhang Wu, Joao P Costeira, and Jose MF Moura. Understanding traffic density from large-scale web camera data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5898–5907, 2017.

[78] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2020–2030, 2017.

[79] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7404–7413, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[80] Han Zhao, Shanghang Zhang, Guanhang Wu, José M. F. Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8559–8570. Curran Associates, Inc., 2018.

Appendix

Table 2 List of the implemented methods in the ADAPT library.

	Method	Cov. Shift	Cond. Shift	Tgt Shift	Mul. Src.	Supervision
Feature	FE [16]	✓	✓	✓	✓	SDA
	CORAL [65]		✓			UDA
	DeepCORAL [66]		✓			UDA
	DANN [22]		✓			UDA
	ADDA [71]		✓			UDA
	MCD [58]		✓			UDA
	MDD [79]		✓			UDA
	CDAN [35]		✓			UDA
	WDGRL [62]		✓			UDA
	MSDA* [80]		✓		✓	UDA
	AHD-MSDA* [54]		✓		✓	UDA
Instance	KMM [27]	✓				UDA
	KLIEP [64]	✓				UDA
	TrB. [15]	✓		✓		SDA
	TrBR2 [48]	✓		✓		SDA
	2Stg-TrBR2 [48]	✓		✓		SDA
	WANN [17]	✓				SDA
Param.	RT LR [9]	✓	✓	✓		SDA
	RT LC [9]	✓	✓	✓		SDA
	RT NN [9, 45]	✓	✓	✓		SDA
	TTC* [61]	✓	✓	✓		SDA
	TTR* [61]	✓	✓	✓		SDA
	TTF* [61]	✓	✓	✓		SDA

* Methods in development.

Table 3 Hyper-parameters for the MNIST to MNIST-M experiment.

SrcOnly	default
DANN	λ = increasing ; γ = 10
ADDA	default
DeepCORAL	λ = 10
MDD	λ = 0.1 ; γ = 1
MCD	λ = 0.1
WDGRL	λ = 1 ; γ = 0.000001
CDAN+E	λ = 1
CORAL	λ = 1000
KLIEP	default
KMM	default

```python
import matplotlib.pyplot as plt
import seaborn as sns

weights = adapt_model.predict_weights()
Xs_weighted = np.random.choice(Xs.ravel(), 1000, p=weights/weights.sum())
limit_noadapt = src_only.intercept_ / src_only.coef_
limit_adapt = adapt_model.estimator_.intercept_ / adapt_model.estimator_.coef_
acc_noadapt = src_only.score(Xt, yt)
acc_adapt = adapt_model.estimator_.score(Xt, yt)

k1 = sns.kdeplot(Xs.ravel(), color="blue", bw_method=0.5, label="Source")
k2 = sns.kdeplot(Xt.ravel(), color="red", bw_method=0.5, label="Target")
k3 = sns.kdeplot(Xs_weighted, color="blue", ls="--",
                 bw_method=0.5, label="Reweighted source")

plt.plot([limit_noadapt[0, 0]]*2, [0, 1], ls=":", c="k")
plt.plot([limit_adapt[0, 0]]*2, [0, 1], ls="--", c="k")
plt.fill_between([-1, 1], [0, 0], [1, 1], alpha=0.2, color="k")
plt.fill_between([-9, 9], [0, 0], [1, 1], alpha=0.05, color="k")
plt.text(-6, 0.2, "_", fontsize=20)
plt.text(-0.45, 0.24, "+", fontsize=20)
plt.text(5.5, 0.2, "_", fontsize=20)
plt.ylim(0, 0.3); plt.xlim(-9, 9)

plt.text(-8.5, 0.25,
         (r"$\cdots$ No adapt : Acc = %.2f"%acc_noadapt + "\n" +
          r"--- Adapt : Acc = %.2f"%acc_adapt),
         bbox=dict(boxstyle='round', fc='w', ec="gray"))
plt.legend(); plt.show();
```

Fig. 11 Python code used to generate Figure 4.

Table 4 Hyper-parameters for the CityCam experiment.

Tgt Only	default
Unif. Weight.	default
WANN	$C = 1.$
DANN	$\lambda = 0.1$
ADDA	default
DeepCORAL	$\lambda = 10$
MDD	$\lambda = 0.0001$; $\gamma = 4$
MCD	$\lambda = 0.1$
WDGRL	$\lambda = 0.1$; $\gamma = 0.000001$
CDAN+E	$\lambda = 1$
CORAL	$\lambda = 1000$
KLIEP	$\sigma = 0.001$
KMM	$\sigma = 0.001$

Table 5 Hyper-parameters for the Office experiment.

Src Only	default
DANN	$\lambda = 1.$
ADDA	default
DeepCORAL	$\lambda = 100$
MDD	$\lambda = 0.1$; $\gamma = 4$
WDGRL	$\lambda = 1$; $\gamma = 0.01$
CDAN+E	$\lambda = 1$