
SKADA-Bench: Benchmarking Unsupervised Domain Adaptation Methods with Realistic Validation

Yanis Lalou^{1*}, Theo Gnassounou^{2*}, Antoine Collas^{2*}, Antoine de Mathelin^{3*},
Oleksii Kachaiev⁴, Ambroise Odonnat⁵, Alexandre Gramfort²,
Thomas Moreau², Rémi Flamary¹

¹École Polytechnique, IP Paris, CMAP, UMR 7641, 91120 Palaiseau, France

²Université Paris-Saclay, Inria, CEA, 91120 Palaiseau, France

³Centre Borelli, ENS Paris-Saclay, Gif-sur-Yvette, 91190, France

⁴Independent Researcher ⁵Huawei Noah's Ark Lab, Paris, France

Abstract

Unsupervised Domain Adaptation (DA) consists of adapting a model trained on a labeled source domain to perform well on an unlabeled target domain with some data distribution shift. While many methods have been proposed in the literature, fair and realistic evaluation remains an open question, particularly due to methodological difficulties in selecting hyperparameters in the unsupervised setting. With SKADA-Bench, we propose a framework to evaluate DA methods and present a fair evaluation of existing shallow algorithms, including reweighting, mapping, and subspace alignment. Realistic hyperparameter selection is performed with nested cross-validation and various unsupervised model selection scores, on both simulated datasets with controlled shifts and real-world datasets across diverse modalities, such as images, text, biomedical, and tabular data with specific feature extraction. Our benchmark highlights the importance of realistic validation and provides practical guidance for real-life applications, with key insights into the choice and impact of model selection approaches. SKADA-Bench is open-source, reproducible, and can be easily extended with novel DA methods, datasets, and model selection criteria without requiring re-evaluating competitors. SKADA-Bench is available on GitHub at <https://github.com/scikit-adaptation/skada-bench>.

1 Introduction

Given some training –or *source*– data, supervised learning consists in estimating a function that makes good predictions on *target* data. However, performance often drops when the source distribution used for training differs from the target distribution used for testing. This shift can be due, for instance, to the collection process or non-stationarity in the data, and is ubiquitous in real-life settings. It has been observed in various application fields, including tabular data [16], clinical data [18], or computer vision [15].

Domain adaptation. Unsupervised Domain Adaptation (DA) addresses this problem by adapting a model trained on a labeled source dataset –or *domain*– so that it performs well on an unlabeled target domain, assuming some distribution shifts between the two [2, 41, 42]. As illustrated in Figure 1, source and target distributions can exhibit various types of shifts [36]: changes in feature distributions (covariate shift), class proportions (target shift), conditional distributions (conditional shift), or in distributions in particular subspaces (subspace shift). Depending on the type of shift, existing DA methods attempt to align the source distribution closer to the target using reweighting [46, 50],

*Equal contribution

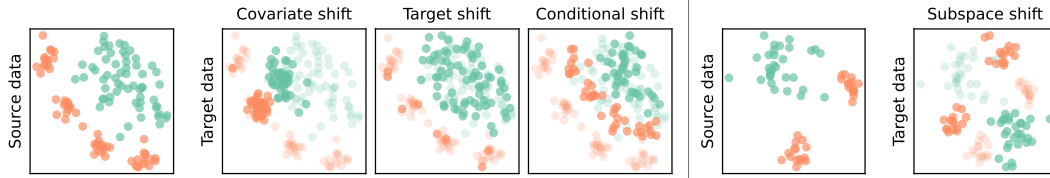


Figure 1: Illustration of the different data shifts studied in the DA literature using the simulated datasets used in the numerical experiments.

mapping [7, 52], or dimension reduction [12, 39] methods. More recently, it has been proposed to mitigate shifts in a feature space learned by deep learning [9, 15, 29, 53], primarily focusing on computer vision applications. Regardless of the core algorithm used to address the domain shift, hyperparameters must be tuned for optimal performance. Indeed, a critical challenge in applying DA methods to real-world cases is selecting the appropriate method and tuning its hyperparameters, especially given the unknown shift type and the absence of labels in the target domain.

Model selection in DA settings. Without distribution shifts, classical model selection strategies—including hyperparameter optimization—rely on evaluating the generalization error with an independent labeled validation set. However, in DA, validating the hyperparameters in a supervised manner on the target domains is impossible due to the lack of labels. While it is possible to validate the hyperparameters on the source domain, it generally leads to a suboptimal model selection because of the distribution shift. In the literature, this problem is often raised but not always addressed. Some papers choose not to validate the parameters [39], while others validate on the source domain [52] or propose custom cross-validation methods [51]. Few papers focus specifically on DA model selection criteria, which we will call *scorers* in this paper. These scorers are used to select the methods’ hyperparameters, and mainly consists of reweighting methods on source [49, 59], prediction entropy [37, 45] or circular validation [3]. One of the goals of our benchmark is to evaluate these approaches in realistic scenarios.

Benchmarks of DA. As machine learning continues to flourish, new methods constantly emerge, making it essential to develop benchmarks that facilitate fair comparisons [22, 31, 35, 40]. In DA and related fields, several benchmarks have been proposed. Numerous papers focus on Out-of-distribution (OOD) datasets for different modalities: computer vision, text, graphs [25, 44], time-series [14], AI-aided drug discovery [23] or tabular dataset [16]. Due to the type of data considered, existing benchmarks are mainly focused on Deep DA methods [11, 24, 38, 56], offering an incomplete evaluation of DA literature. Moreover, only a few benchmarks propose a comparison of Deep unsupervised DA methods with realistic parameters selection, on computer vision [20, 38] and time series [11] data. Those benchmarks have shown the importance of validating with unsupervised scores and reveal that deep DA methods achieve much lower performance in realistic scenarios. In the present work, we focus on “shallow” DA methods, addressing a gap in the DA literature and its evaluation.

Contributions. In the following, we propose SKADA-Bench, an ambitious and fully reproducible benchmark with the following features: **1.** a set of 4 simulated and 8 real-life datasets with different modalities (CV, NLP, tabular, biomedical) totaling 51 realistic shift scenarios, **2.** a wide range of 20 shallow DA methods designed to handle different types of shifts, **3.** a realistic model selection procedure using 5 different unsupervised scorers with nested cross-validation for hyperparameter selection, **4.** an open-source code and publicly available datasets, easy to extend for new DA methods and datasets without the need to re-run the whole experiment.

In addition, we provide a detailed analysis of the results and derive guidelines for practitioners to select the best methods depending on the type of shifts, and the best scorer to perform unsupervised model selection. In particular, the effects of model selection and the scorer’s choice on the final performances are highlighted, showing a clear gap between the unsupervised realistic scorers versus using target labels for supervised validation.

2 Domain adaptation and model selection without target labels

In this section, we first discuss the specificities of the unsupervised domain adaptation problem and introduce several types of data shift and their corresponding DA methods. Next, we discuss the different validation strategies used in the literature and the need for realistic scorers to compare DA methods.

2.1 Data shifts and DA strategies

Domain Adaptation problem and theory. The theoretical framework of DA is well established [2, 41, 42]. The main results highlight that the performance discrepancy of an estimator between the source and target domains is linked to the divergence between both distributions. This has motivated the majority of DA methods to search for a universal (or domain invariant) predictor by minimizing the divergence between the two domains through the adaptation of the distributions. This is done in practice by modeling and estimating the shift between the source and target distributions and then compensating for this shift before training a predictor.

Data Shifts and DA methods. A wide variety of shifts between the source and target distributions are possible. They are usually expressed as a relation between the joint distributions $P^s(x, y) = P^s(x|y)P_y^s(y) = P^s(y|x)P_x^s(x)$ in the source domain and $P^t(x, y) = P^t(x|y)P_y^t(y) = P^t(y|x)P_x^t(x)$ in the target domain. We now discuss the main types of shifts and the strategies proposed in the literature to mitigate them. Figure 1 illustrates these shifts.

In **Covariate shift** the conditionals probabilities are equal (*i.e.*, $P^s(y|x) = P^t(y|x)$), but the feature marginals change (*i.e.*, $P_x^s(x) \neq P_x^t(x)$). **Target shift** is similar, but the label marginals change $P_y^s(y) \neq P_y^t(y)$ while the conditionals are preserved. For classification problems, it corresponds to a change in the proportion of the classes between the two domains. Both of those shifts can be compensated by reweighting methods that assign different weights to the samples of the source domain to make it closer to the target domain [46, 50].

In **Conditional shift**, conditional probabilities differ between domain (*i.e.*, $P^s(x|y) \neq P^t(x|y)$ or $P^s(y|x) \neq P^t(y|x)$). This shift is typically harder to compensate for, necessitating explicit modeling to address it effectively. For instance, several approaches model the shift as a mapping m between the source and target domain such that $P^s(y|m(x)) = P^t(y|x)$ [7, 52]. The estimated mapping is then applied to the source data before training a predictor.

Subspace shift, also known as domain invariant representation, assumes that while probabilities are different between the domains, there exists a subspace \mathcal{Z} and a function $\phi : \mathcal{X} \rightarrow \mathcal{Z}$ such that $P^s(y|\phi(x)) = P^t(y|\phi(x))$. This implies that a classifier trained on this subspace will perform well across both domains. Subspace methods aim to identify the subspace \mathcal{Z} and the function ϕ , as developed in [12, 39]. Note that, as discussed in the introduction, a natural extension of this idea is to learn a non-linear invariant subspace using deep learning [15, 53].

2.2 DA model selection strategies

As seen above, the different DA methods are usually designed for one type of shift, yet in a practical problem, one does not know what shift is present. This raises the problem of method and parameter selection when facing a new problem. In this section, we discuss the validation strategies proposed in the literature to compare DA methods, focusing on realistic scorers that do not use target labels.

Realistic DA scorers. In the literature, few papers propose realistic DA scorers to validate the parameters of the methods, *i.e.*, unsupervised scorers that do not require target labels. The *Importance Weighted (IW)* scorer [49] computes the score as a reweighted accuracy on labeled sources data. The *Deep Embedded Validation (DEV)* [59] can be seen as an IW in the latent space with a variance reduction strategy. DEV was originally proposed for deep learning models but can be used on shallow DA methods that compute features from the data (mapping/subspaces). The *Prediction Entropy (PE)* scorer [37] measures the uncertainty associated with model predictions on the target data. *Soft Neighborhood Density (SND)* [45] also computes an entropy but on a normalized pairwise similarity matrix between probabilistic predictions on target. The *Circular Validation (CircV)* scorer [3] performs DA by first adapting the model from the source to the target domain and predicting

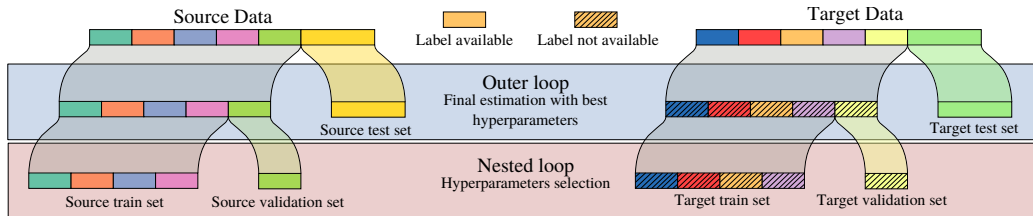


Figure 2: Visualization of nested cross-validation strategy. Both source and target data are split into an outer loop and then a nested loop. The nested loop tunes hyperparameters for the domain adaptation method, while the outer loop trains a final classifier with the best hyperparameters and evaluates its accuracy on both source and target data. Note: Target sets have no labels during the nested loop, reflecting unsupervised domain adaptation.

the target labels. Next, it adapts back from the target to the source using these estimated labels. Performance is measured as the accuracy between the recovered and true source labels.

DA validation in the literature. The model selection problem in DA has been widely discussed in the literature. Yet, this literature constitutes a subfield of DA and has seldom been used to validate new DA methods. Indeed, there is no consensus on the best validation strategy and many papers do not properly validate their methods, leading to over-estimated performances. Some authors do not discuss the validation procedure [46, 50] or consider fixed hyperparameters [21]. While some methods rely on custom validation technique [51], others use cross-validation, either on the source or the target [7, 52], or alternatively other validation strategies proposed in the literature [3, 6]. A complete picture of the model selection procedures used to validate the methods considered in SKADA-Bench in their original papers is presented in Table A.1 in Appendix. The goal of SKADA-Bench is therefore to constitute a dedicated benchmark to compare scorers from the literature and report performances that can be expected in real use cases for the considered methods.

3 A realistic benchmark for DA

In this section, we present our benchmark framework. First, we introduce the parameter validation strategies. Then, we present the compared DA methods followed by a description of the datasets used in the benchmark.

3.1 Nested cross-validation loop and implementation

We discuss below the nested cross-validation and the implementation details of the benchmark.

Hyperparameter validation loop. We propose a nested loop cross-validation procedure, depicted in Figure 2. First, the source and target data are split into multiple outer test and train sets (outer loop in Figure 2). The test sets are kept to compute the final accuracy for both the source and target domains. For each split in the outer loop, we use a nested loop to select the DA methods’ parameters. Here, the training sets are further divided into nested train and validation sets (nested loop in Figure 2). Note that no labels are available for the target nested train and validation sets in this loop. The target training set is used to train the DA method, while the target validation set allows to compute the unsupervised score and select the best model.

For both loops, the data is split randomly 5 times using stratified sampling with an 80%/20% train/test split. For one given method, we evaluate all the unsupervised scorers discussed earlier, as well as a supervised scorer that uses target labels, over all the nested splits. After averaging, the scores over the splits, the best hyperparameters are selected according to each scorer and then used to train a final classifier on the outer training sets. Although the supervised scorer cannot be used in practice, it is included in our results to actually evaluate the performance drop due to the absence of target labels. To limit complexity and perform a fair comparison of the methods, we set a timeout of 4 hours for performing the nested loop.

Base estimator. Existing domain adaptation methods typically rely on either a base estimator trained on the adapted data or an iterative estimation process to adapt this estimator to the target data. The choice of the base estimator is crucial, as it significantly impacts the final performance. Before validating the hyperparameters of the DA methods, we determined the best estimator for each dataset using a grid-search on the source data. We tested multiple hyperparameters for Logistic Regression, SVM with RBF kernel, and XGBoost [5], selecting the ones that maximize the average accuracy on the source test sets. Note that for some methods that specifically require an SVM estimator (*i.e.*, JDOT and DASVM), we only validate SVM as the base estimator. We validated the base estimator separately from the DA methods parameters to reduce computational complexity and avoid too complex hyperparameter grids that can compromise the reliability of DA scorers.

Best scorer selection and statistical test. For all methods, we select the best validation scorer as the one that maximizes the averaged accuracy on the target domains for all real datasets. This provides a reasonable and actionable choice of scorer for each DA method for practitioners. For all methods and datasets, we perform a paired Wilcoxon signed-rank test at the 0.05 level to detect significant gain or drop in performance with respect to the no DA approach, denoted by “Train Src” in the following. The test is done using the accuracy measures of the DA method with the selected scorer and the Train Src for all shifts and outer splits, ensuring between 10 and 60 values depending on the dataset.

Python implementation. The benchmark code will be made available on GitHub upon publication of the paper.* Our benchmark is implemented following the `benchopt` framework [35], which provides standardized ways of organizing and running benchmarks for ML in Python. This framework facilitates reproducing the benchmark’s results, with tools to install the dependencies, run the methods in parallel, or cache the results to prevent redundant computations. It also makes it easy to extend the benchmark with additional datasets and methods, enabling it to evolve to account for the advances in the field. In the supplementary materials, we provide examples demonstrating how to add DA methods or datasets to the benchmark. Using this framework, we aim to make SKADA-Bench a reference benchmark to evaluate new DA methods in realistic scenarios with valid performance estimations.

3.2 Compared DA methods

In this section, we present the different families of domain adaptation methods that we compare in our benchmark. We group the methods into four categories: reweighting methods, mapping methods, subspace methods, and others. We provide a brief description of each method and the corresponding references.

Reweighting methods. These methods aim to reweight the source data to make it closer to the target data. The weights are estimated using different methods such as kernel density estimation (Dens. RW) [50], Gaussian estimation (Gauss. RW) [46], discriminative estimation (Discr. RW) [46], or nearest-neighbors (NN RW) [30]. Other reweighting estimate weights by minimizing a divergence between the source and target distributions such as Kullback-Leibler Importance Estimation Procedure (KLIEP) [51] or Kernel Mean Matching (KMM) [21]. Finally, we also include the MMDTarS method [60] that uses a Maximum Mean Discrepancy (MMD) to estimate the weights under the target shift hypothesis.

Mapping methods. These methods aim to find a mapping between the source and target data that minimizes the distribution shift. The Correlation Alignment method (CORAL) [52] aligns the second-order statistics of source and target distributions. The Maximum Mean Discrepancy (MMD-LS) method [60] minimizes the MMD to estimate an affine Location-Scale mapping. Finally, the Optimal Transport (OT) mapping methods [7] use the optimal transport plan to align with a non-linear mapping of the source and target distributions with exact OT (MapOT), entropic regularization (EntOT), or class-based regularization (ClassRegOT). Finally, the Linear OT method [13] uses a linear mapping to align the source and target distributions, assuming Gaussian distributions.

Subspace methods. These methods aim to learn a subspace where the source and target data have the same distribution. The Transfer Component Analysis (TCA) method [39] searches for a kernel

*Our code is available in supplementary materials.

Table 1: Characteristics of the real-world datasets used in SKADA-Bench.

Dataset	Modality	Preprocessing	# adapt	# classes	# samples	# features
Office 31 [26]	CV	Decaff [10] + PCA	6	31	470 ± 350	100
Office Home [55]	CV	ResNet [19] + PCA	12	65	3897 ± 850	100
MNIST/USPS [28]	CV	Vect + PCA	2	10	3000 / 10000	50
20 Newsgroup [27]	NLP	LLM [43, 57] + PCA	6	2	3728 ± 174	50
Amazon Review [32, 33]	NLP	LLM [43, 57] + PCA	12	4	2000	50
Mushrooms [8]	Tabular	One Hot Encoding	2	2	4062 ± 546	117
Phishing [34]	Tabular	NA	2	2	5527 ± 1734	30
BCI [54]	Biosignals	Cov+TS [1]	9	4	288	253

embedding that minimizes the MMD divergence between the domains while preserving the variance. The Subspace Alignment (SA) method [12] aims to learn a subspace where the source and target have their covariance matrices aligned. The Transfer Subspace Learning (TSL) method [47] aims to learn a subspace using classical supervised loss functions on the source (*e.g.*, PCA, Fisher LDA) but regularized so that the source and target data have the same distribution once projected on the subspace. Finally, the Joint Principal Component Analysis (JPCA) method is a simple baseline that concatenates source and target data before applying a PCA on all data.

Others. We also include other methods that do not fit into the previous categories. The Domain Adaptation SVM (DASVM) method [3] is a self-labeling method that iteratively updates SVM estimators by adding new target samples with predicted labels and removing source samples. The Joint Distribution Optimal Transport (JDOT) method [6] aims to learn a target predictor that minimizes an OT loss between the joint source and target distributions. The Optimal Transport Label Propagation (OTLabelProp) method [48] uses the optimal transport plan to propagate labels from the source to the target domain.

3.3 Compared datasets

In this section, we present the datasets used in our experiments. We first introduce the synthetic datasets that implement different known shifts. Then, we describe the real-world datasets from various modalities and tasks such as Computer Vision (CV), Natural language Processing (NLP), tabular data, and biosignals.

Simulated datasets. The objective of the simulated datasets is to evaluate the performance of the DA methods under different types of shifts. Knowing that multiple DA methods have been built to handle specific shifts, evaluating them with this dataset will demonstrate whether they perform as expected and if they are properly validated.

The 4 simulated shifts in 2D, covariate (Cov. shift), target (Tar. shift) conditional (Cond. shift) and Subspace (Sub. shift) shift are illustrated in Figure 1. The source domain is represented by two non-linearly separable classes generated from one large and several smaller Gaussian blobs. In the experiments, the level of noise has been adjusted from Figure 1 to make the problem more difficult. For the subspace shift scenario, the source domain consists of one class represented by a large Gaussian blob and another class comprising Gaussian blobs positioned along the sides of the large one. The target domain is flipped along the diagonal, making the task challenging in the original space but feasible upon diagonal projection.

Real-word datasets. The real-world datasets used in our benchmark are summarized in Table 1. We select 8 datasets from different modalities and tasks: Computer Vision (CV) with Office31 [26], Office Home [55], and MNIST/USPS [28], Natural Language Processing (NLP) with 20Newsgroup [27] and Amazon Review [33], Tabular Data with Mushrooms [8] and Phishing [34], and Biosignals with BCI Competition IV [54]. The datasets are chosen to represent a wide range of shifts and to evaluate the performance of the methods on different types of data.

The datasets are preprocessed with feature extraction to ensure reasonable performance when trained on each domain. For example, images are embedded using pre-trained models followed by a PCA (except MNIST/USPS where only PCA is used), and textual data is embedded using Large Language Models (LLM) [43, 57] before applying a PCA. The tabular data are one-hot encoded to transform categorical data into numerical data. The biosignals from Brain-Computer Interface (BCI) data are embedded using the state-of-the-art tangent space representation proposed in [1]. The datasets are

Table 2: Accuracy score for all datasets compared for all the methods for simulated and real-life datasets. The color indicates the amount of the improvement. A white color means the method is not statistically different from Train Src (Train on source). Blue indicates that the score improved with the DA methods, while red indicates a decrease. The darker the color, the more significant the change.

		<u>Cov. shift</u>	<u>Tar. shift</u>	<u>Cond. shift</u>	<u>Sub. shift</u>	Office31	OfficeHome	MNIST/USPS	20NewsGroups	AmazonReview	Mushrooms	Phishing	BCI	Selected Scorer	Rank
	Train Src	0.88	0.85	0.66	0.19	0.59	0.56	0.54	0.59	0.7	0.72	0.91	0.55		9.75
	Train Tgt	0.92	0.93	0.82	0.98	0.88	0.8	0.96	1.0	0.73	1.0	0.97	0.64		1.06
Reweighting	Dens. RW [50]	0.88	0.86	0.66	0.18	0.57	0.55	0.54	0.58	0.7	0.71	0.91	0.55	IW	10.76
	Disc. RW [46]	0.85	0.83	0.71	0.18	0.58	0.53	0.5	0.6	0.68	0.75	0.91	0.56	CircV	11.12
	Gauss. RW [46]	0.89	0.86	0.65	0.21	0.2	0.44	0.11	0.54	0.6	0.51	0.46	0.25	CircV	19.42
	KLIEP [51]	0.88	0.86	0.66	0.19	0.59	0.56	0.54	0.6	0.69	0.72	0.91	0.55	CircV	10.36
	KMM [21]	0.89	0.87	0.64	0.15	0.58	0.55	0.52	0.7	0.57	0.74	0.91	0.52	CircV	12.11
	NN RW [30]	0.89	0.86	0.67	0.15	0.58	0.55	0.54	0.59	0.66	0.71	0.91	0.54	CircV	11.91
	MMDTarS [60]	0.88	0.86	0.64	0.2	0.56	0.55	0.54	0.59	0.7	0.74	0.91	0.55	IW	9.51
Mapping	CORAL [52]	0.66	0.84	0.66	0.19	0.59	0.57	0.62	0.73	0.69	0.72	0.92	0.62	CircV	7.10
	MapOT [7]	0.72	0.57	0.82	0.02	0.55	0.51	0.61	0.76	0.67	0.63	0.84	0.47	PE	10.98
	EntOT [7]	0.71	0.6	0.82	0.12	0.58	0.58	0.6	0.83	0.62	0.75	0.86	0.54	CircV	9.75
	ClassRegOT [7]	0.74	0.58	0.81	0.11	0.61	0.53	0.62	0.96	0.68	0.82	0.88	0.52	IW	8.71
	LinOT [13]	0.73	0.73	0.76	0.18	0.59	0.57	0.64	0.82	0.7	0.76	0.91	0.61	CircV	5.33
	MMD-LS [60]	0.65	0.68	0.81	0.52	0.55	0.54	0.52	0.97	0.68	0.86	0.88	0.56	IW	9.66
	Subspace	JPCA	0.88	0.85	0.66	0.15	0.55	0.47	0.51	0.77	0.69	0.78	0.9	0.54	PE
SA [12]		0.74	0.68	0.8	0.11	0.59	0.57	0.56	0.88	0.66	0.88	0.89	0.53	CircV	8.53
TCA [39]		0.46	0.48	0.55	0.56	0.04	NA	0.11	0.57	0.6	0.45	NA	0.27	CircV	19.57
TSL [47]		0.88	0.85	0.66	0.19	0.59	0.2	0.25	0.68	0.7	0.56	0.86	0.25	IW	14.65
Other		JDOT [6]	0.72	0.57	0.82	0.14	0.6	0.51	0.63	0.77	0.67	0.63	0.8	0.46	DEV
	OTLabelProp [48]	0.72	0.59	0.81	0.05	0.61	0.56	0.62	0.86	0.67	0.64	0.86	0.5	CircV	10.49
	DASVM [3]	0.89	0.86	0.65	0.14	NA	NA	NA	0.68	NA	0.78	0.88	NA	CircV	11.00

split into pairs of source and target domains totaling 51 adaptation tasks in the benchmark. More details about the datasets and pre-processing are available in Appendix B.

4 Benchmark results

We now present the results of the benchmark. Training and evaluation across all experiments required 1,215 CPU-hours on a standard Slurm [58] cluster. We first discuss and compare the performances of the methods on the different datasets. Then, a detailed study of the unsupervised scorers is provided.

4.1 Performance of the DA methods

Results table. First, we report the realistic performances of the different methods when using their selected scorer on the different datasets in Table 2. The cells showcasing a significant change in performance with the Wilcoxon test are highlighted with colors. Blue indicates an increase in performance, while red indicates a loss. The intensity of the color corresponds to the magnitude of the gain or loss - the darker the shade, the larger the positive or negative change. Cells with a NA values indicate that the method was not applicable to the dataset (DASVM is limited to binary classification) or that the method has reached a timeout. We also report the best scorer and the average rank of the methods for all datasets. In addition to Table 2 providing realistic performance estimations, we also report in Table D.12 (Appendix D) the results when using the non-realistic supervised scorer.

Simulated data with known shifts. On simulated data with known shifts (*i.e.*, underlined datasets), DA methods tend to show a small but significant gain on the shift they were designed for. However, they rarely reach the Train Tgt performances, and as expected, the methods perform poorly on the shift they were not designed for. For Cov. shift, the improvement with reweighting methods is very limited. We believe that using a complex base estimator (SVM with an RBF kernel) enables us to train an estimator that works well on both source and target, leading to less impressive results. The Appendix D contains detailed tables for two other base estimators (*i.e.*, Logistic Regression in Table D.9 and XGBoost in Table D.11). These tables show that the gain is larger for Logistic

Regression, a linear classifier, although the overall performance is lower. Note that in the literature, the base estimator used is often a simple method, which can explain the performance gains reported in various papers.

Real life datasets. On the real-life datasets, we observe that DA methods do not consistently outperform the Train Src approach. The only exceptions are the LinOT and CORAL methods, which either result in a significant improvement or perform comparably to the Train Src approach. OT-based mapping methods show gains on MNIST/USPS and 20NewsGroups. Other methods, such as ClassRegOT, MMD-LS, JPCA, and SA, occasionally show better accuracy than Train Src. However, interestingly, DA methods fail to improve performance on datasets like Office31, AmazonReview, and Phishing. For the AmazonReview dataset, Train Src performs exceptionally well, possibly because the selected LLM has been partially trained on similar Amazon review data, leading to invariant representations [57].

Looking at the average rank of the methods, we observe that the Train Src approach has a relatively small rank of 9.75 out of 20, while the best-rank DA methods are LinOT with a rank of 5.33 and CORAL with 7.10. Other methods with a lower rank than Train Src are ClassRegOT, MMDTarS, JPCA, and SA.

Selected scorer per DA method. We observe that the best scorer differs across methods, Circular Validation has been selected 12 out of 20 times as the best scorer, followed by Importance Weighting 5 out of 20 times. Table D.12 in the supplementary material provides the accuracy results with the supervised scorer. It is worth noting that the supervised scorer generally outperforms the unsupervised ones, and several methods significantly outperform Train Src in each dataset. It is crucial to choose the model realistically to avoid producing overly optimistic results, as many data analysis papers have done (see table A.1).

These results show the methods’ sensitivity to parameter selections and the difficulty of using realistic scorers. This might also explain why DA methods are not widely used in practice: they are very difficult to tune and might decrease performances compared with no adaptation.

4.2 Study of validation scorers

We now investigate the performance of the various scorers to select hyperparameters of the DA method. First, we consider the relationship between the cross-val score and the accuracy for each inner split. In Figure 3, we plot for each scorer the cross-val score as a function of the accuracy computed on the test set and report the Pearson correlation coefficient ρ . As expected, the supervised scorer is highly correlated with the accuracy ($\rho = 0.98$), as it has access to the target labels. We observe that SND, DEV, and PE do not provide a good proxy to select hyperparameters that give the best-performing models ($\rho \leq 0.06$). On the contrary, IW and CircV are correlated with the accuracy, $\rho = 0.53$ and $\rho = 0.71$ respectively. This is coherent with their selection as the best scorer in most scenarios in Table 2. Still, while those scorers are well correlated with the target accuracy, it is important to note that they have a large variance. For instance, a score close to 1 in IW or CircV corresponds to an accuracy between 0.5 and 1.0.

Furthermore, we provide in Figures D.3 and D.4, from Appendix D, several visualizations that illustrate the relationship between the accuracy achieved when using a supervised scorer and the accuracy obtained when using different unsupervised scorers. We also visualize in Figure D.2 the drop in performance when using the best-unsupervised scorer instead of the supervised scorer. Interestingly some methods such as KMM, EntOT, and ClassRegOT can lose up to 10% accuracy when using realistic scorers, which might come from their high number of parameters or their sensitivity to them.

Our results thus show that most scorers have poor results when evaluated on many datasets. Of the five methods under consideration, only two achieve satisfactory performance, although incurring large variance in their results. This shows that proper hyperparameter selection is still an open question, that needs attention from the research community to guide practitioners toward real life applications of unsupervised DA technics.

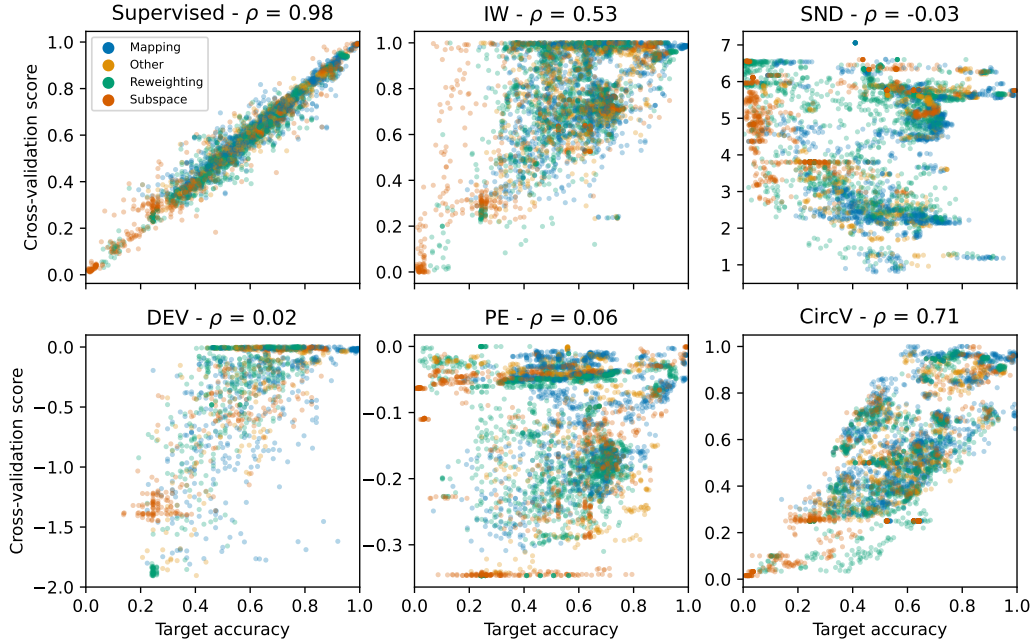


Figure 3: Cross-val score as a function of the accuracy for different supervised and unsupervised scorers. The Pearson correlation coefficient is reported for each scorer by ρ . Each point represents an inner split with a DA method (color of the points) and a dataset. A good score should correlate with the target accuracy.

5 Conclusion

In this work, we introduced SKADA-Bench, a comprehensive benchmark for unsupervised domain adaptation, focusing on shallow DA methods and the impact of their model selection. Our findings reveal that few DA methods consistently perform well across diverse datasets and that model selection scorers significantly influence their effectiveness. For each DA method, we provide the optimal model selection scorer for unsupervised hyperparameter tuning based on our experiments. We believe this benchmark will aid the community in understanding the critical role of model selection in DA performance and inspire the development of methods that are more robust to hyperparameter choices.

Acknowledgments

This work was supported by grant ANR-22-PESN-0012 to AC under the France 2030 program, ANR-20-CHIA-0016 and ANR-20-IADJ-0002 to AG, and ANR-23-ERCC-0006 to YL, TG, and RF, all managed by the Agence Nationale de la Recherche (ANR). Additionally, this project received funding from the European Union’s Horizon Europe research and innovation program under grant agreement 101120237 (ELIAS).

All the datasets used for this work were accessed and processed on the Inria and IP Paris (IDCS) compute infrastructures.

References

- [1] Barachant, A., Bonnet, S., Congedo, M., and Jutten, C. (2012). Multiclass brain–computer interface classification by riemannian geometry. *IEEE Transactions on Biomedical Engineering*, 59(4):920–928.
- [2] Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2006). Analysis of representations for domain adaptation. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press.

- [3] Bruzzone, L. and Marconcini, M. (2010a). Domain adaptation problems: A dasvm classification technique and a circular validation strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):770–787.
- [4] Bruzzone, L. and Marconcini, M. (2010b). Domain adaptation problems: A dasvm classification technique and a circular validation strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):770–787.
- [5] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.
- [6] Courty, N., Flamary, R., Habrard, A., and Rakotomamonjy, A. (2017a). Joint distribution optimal transportation for domain adaptation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- [7] Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2017b). Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865.
- [8] Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2007). Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning, ICML '07*, page 193–200, New York, NY, USA. Association for Computing Machinery.
- [9] Damodaran, B. B., Kellenberger, B., Flamary, R., Tuia, D., and Courty, N. (2018). Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, page 467–483, Berlin, Heidelberg. Springer-Verlag.
- [10] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 647–655, Beijing, China. PMLR.
- [11] Fawaz, H. I., Del Grosso, G., Kerdoncuff, T., Boisbunon, A., and Saffar, I. (2023). Deep unsupervised domain adaptation for time series classification: a benchmark. *arXiv preprint arXiv:2312.09857*.
- [12] Fernando, B., Habrard, A., Sebban, M., and Tuytelaars, T. (2013). Unsupervised visual domain adaptation using subspace alignment. In *2013 IEEE International Conference on Computer Vision*, pages 2960–2967.
- [13] Flamary, R., Lounici, K., and Ferrari, A. (2020). Concentration bounds for linear monge mapping estimation and optimal transport domain adaptation.
- [14] Gagnon-Audet, J.-C., Ahuja, K., Bayazi, M. J. D., Mousavi, P., Dumas, G., and Rish, I. (2023). WOODS: Benchmarks for out-of-distribution generalization in time series. *Transactions on Machine Learning Research*. Featured Certification.
- [15] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.
- [16] Gardner, J. P., Popovi, Z., and Schmidt, L. (2023). Benchmarking distribution shift in tabular data with tableshift. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- [17] Gnassounou, T., Kachaiev, O., Flamary, R., Collas, A., Lalou, Y., Mathelin, A., Gramfort, A., Bueno, R., Michel, F., Mellot, A., Loison, V., Odonnat, A., and Moreau, T. (2024). Skada : Scikit adaptation.
- [18] Harutyunyan, H., Khachatrian, H., Kale, D. C., Ver Steeg, G., and Galstyan, A. (2019). Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1).

- [19] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.
- [20] Hu, D., Liang, J., Liew, J. H., Xue, C., Bai, S., and Wang, X. (2023). Mixed samples as probes for unsupervised model selection in domain adaptation. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 37923–37941. Curran Associates, Inc.
- [21] Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., and Smola, A. (2006). Correcting sample selection bias by unlabeled data. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press.
- [22] Hutson, M. (2018). Artificial intelligence faces reproducibility crisis.
- [23] Ji, Y., Zhang, L., Wu, J., Wu, B., Li, L., Huang, L.-K., Xu, T., Rong, Y., Ren, J., Xue, D., Lai, H., Liu, W., Huang, J., Zhou, S., Luo, P., Zhao, P., and Bian, Y. (2023). Drugood: Out-of-distribution dataset curator and benchmark for ai-aided drug discovery – a focus on affinity prediction problems with noise annotations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):8023–8031.
- [24] Jiang, J., Shu, Y., Wang, J., and Long, M. (2022). Transferability in deep learning: A survey.
- [25] Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B., Haque, I., Beery, S. M., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. (2021). Wilds: A benchmark of in-the-wild distribution shifts. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR.
- [26] Koniusz, P., Tas, Y., and Porikli, F. (2017). Domain adaptation by mixture of alignments of second-or higher-order scatter tensors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7139–7148.
- [27] Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.
- [28] Liao, Z. and Carneiro, G. (2015). Competitive multi-scale convolution.
- [29] Long, M., Cao, Y., Wang, J., and Jordan, M. (2015). Learning transferable features with deep adaptation networks. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 97–105, Lille, France. PMLR.
- [30] Loog, M. (2012). Nearest neighbor-based importance weighting. In SN, editor, *The 2012 IEEE workshop on machine learning for signal processing (MLSP) proceedings*, pages 1–6, United States. IEEE. The 2012 IEEE workshop on machine learning for signal processing (MLSP) ; Conference date: 23-09-2012 Through 26-09-2012.
- [31] Mattson, P., Reddi, V. J., Cheng, C., Coleman, C., Damos, G., Kanter, D., Micikevicius, P., Patterson, D., Schmuelling, G., Tang, H., Wei, G.-Y., and Wu, C.-J. (2020). Mlperf: An industry standard benchmark suite for machine learning performance. *IEEE Micro*, 40(2):8–16.
- [32] McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.
- [33] McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’15*, page 43–52, New York, NY, USA. Association for Computing Machinery.
- [34] Mohammad, R. M., Thabtah, F., and McCluskey, L. (2012). An assessment of features related to phishing websites using an automated technique. In *2012 International Conference for Internet Technology and Secured Transactions*, pages 492–497.

- [35] Moreau, T., Massias, M., Gramfort, A., Ablin, P., Bannier, P.-A., Charlier, B., Dagr eou, M., la Tour, T. D., Durif, G., Dantas, C. F., Klopfenstein, Q., Larsson, J., Lai, E., Lefort, T., Mal ezieux, B., Moufad, B., Nguyen, B., Rakotomamonjy, A., Ramzi, Z., Salmon, J., and Vaiter, S. (2022). Benchopt: Reproducible, efficient and collaborative optimization benchmarks. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.
- [36] Moreno-Torres, J. G., Raeder, T., Alaiz-Rodr guez, R., Chawla, N. V., and Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530.
- [37] Morerio, P., Cavazza, J., and Murino, V. (2017). Minimal-entropy correlation alignment for unsupervised deep domain adaptation.
- [38] Musgrave, K., Belongie, S., and Lim, S.-N. (2021). Unsupervised domain adaptation: A reality check. *arXiv preprint arXiv:2111.15672*.
- [39] Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.
- [40] Pineau, J., Sinha, K., Fried, G., Ke, R. N., and Larochelle, H. (2019). Iclr reproducibility challenge 2019. *ReScience C*, 5(2):5.
- [41] Quinero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2008). *Dataset shift in machine learning*. MIT Press.
- [42] Redko, I., Morvant, E., Habrard, A., Sebban, M., and Bennani, Y. (2022). A survey on domain adaptation theory: learning bounds and theoretical guarantees.
- [43] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *EMNLP/IJCNLP (1)*, pages 3980–3990. Association for Computational Linguistics.
- [44] Sagawa, S., Koh, P. W., Lee, T., Gao, I., Xie, S. M., Shen, K., Kumar, A., Hu, W., Yasunaga, M., Marklund, H., Beery, S., David, E., Stavness, I., Guo, W., Leskovec, J., Saenko, K., Hashimoto, T., Levine, S., Finn, C., and Liang, P. (2022). Extending the WILDS benchmark for unsupervised adaptation. In *International Conference on Learning Representations*.
- [45] Saito, K., Kim, D., Teterwak, P., Sclaroff, S., Darrell, T., and Saenko, K. (2021). Tune it the right way: Unsupervised validation of domain adaptation via soft neighborhood density. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9164–9173.
- [46] Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244.
- [47] Si, S., Tao, D., and Geng, B. (2010). Bregman divergence-based regularization for transfer subspace learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(7):929–942.
- [48] Solomon, J., Rustamov, R., Guibas, L., and Butscher, A. (2014). Wasserstein propagation for semi-supervised learning. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 306–314, Beijing, China. PMLR.
- [49] Sugiyama, M., Krauledat, M., and M ller, K.-R. (2007a). Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005.
- [50] Sugiyama, M. and M ller, K.-R. (2005). Input-dependent estimation of generalization error under covariate shift. *Statistics & Risk Modeling*, 23(4):249–279.
- [51] Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P., and Kawanabe, M. (2007b). Direct importance estimation with model selection and its application to covariate shift adaptation. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.
- [52] Sun, B., Feng, J., and Saenko, K. (2017). *Correlation Alignment for Unsupervised Domain Adaptation*, pages 153–171. Springer International Publishing, Cham.

- [53] Sun, B. and Saenko, K. (2016). Deep coral: Correlation alignment for deep domain adaptation. In Hua, G. and Jégou, H., editors, *Computer Vision – ECCV 2016 Workshops*, pages 443–450, Cham. Springer International Publishing.
- [54] Tangermann, M., Müller, K.-R., Aertsen, A., Birbaumer, N., Braun, C., Brunner, C., Leeb, R., Mehring, C., Miller, K., Mueller-Putz, G., Nolte, G., Pfurtscheller, G., Preissl, H., Schalk, G., Schlögl, A., Vidaurre, C., Waldert, S., and Blankertz, B. (2012). Review of the BCI Competition IV. *Frontiers in Neuroscience*, 6.
- [55] Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [56] Wang, J. (2018). Everything about transfer learning and domain adaptation.
- [57] Xiao, S., Liu, Z., Zhang, P., and Muennighoff, N. (2023). C-pack: Packaged resources to advance general chinese embedding.
- [58] Yoo, A. B., Jette, M. A., and Grondona, M. (2003). Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer.
- [59] You, K., Wang, X., Long, M., and Jordan, M. (2019). Towards accurate model selection in deep unsupervised domain adaptation. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7124–7133. PMLR.
- [60] Zhang, K., Schölkopf, B., Muandet, K., and Wang, Z. (2013). Domain adaptation under target and conditional shift. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 819–827, Atlanta, Georgia, USA. PMLR.
- [61] Zhong, E., Fan, W., Yang, Q., Verscheure, O., and Ren, J. (2010). Cross validation framework to choose amongst models and datasets for transfer learning. In Balcázar, J. L., Bonchi, F., Gionis, A., and Sebag, M., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 547–562, Berlin, Heidelberg. Springer Berlin Heidelberg.

Appendix

Reproducibility. The entire code and results of SKADA-Bench are open-sourced at <https://github.com/scikit-adaptation/skada-bench>. The implementation of the DA methods and scorers is provided along with access to the simulated and real-world datasets. All the performance tables and figures can be reproduced effortlessly, and guidelines with minimal working examples are given to add new DA methods and datasets.

Roadmap. In this appendix, we provide additional information regarding the validation procedure used in the literature for each DA method implemented in SKADA-Bench in Section A. We provide a detailed description of the data and preprocessing used in SKADA-Bench in Section B. In Section C, we give minimal working Python examples to add a new DA method and dataset in SKADA-Bench. Finally, we provide the detailed benchmark results in Section D. In particular, the results per dataset can be found in Section D.1. We discuss in Section D.2 the impact of the choice of base estimator on the performance of DA methods for the simulated datasets. The results of each DA method with the supervised scorer on all the datasets are given in Table D.12 of Section D.3, which parallels Table 2. A thorough analysis of the effect of using realistic unsupervised scorers is also provided in Section D.4. Finally, the computational efficiency of each DA method is studied in Section D.5 and the hyperparameters used for grid search are given in Section D.6. We display the corresponding table of contents below.

Table of Contents for Appendix

A	Model selection in Domain Adaptation	15
B	Datasets description and preprocessing	15
C	Adding new methods and datasets to SKADA-Bench	16
	C.1 Adding a new DA method	16
	C.2 Adding a new dataset	17
D	Benchmark detailed results	18
	D.1 Results per datasets	18
	D.2 Impact of the base estimators on the simulated datasets	27
	D.3 Unrealistic validation with supervised scorer	30
	D.4 Comparisons between supervised and unsupervised scorers	30
	D.5 Computational efficiency of the DA methods	32
	D.6 Hyperparameters grid search for the DA methods	33

A Model selection in Domain Adaptation

Table A.1: Validation procedure in Domain Adaptation methods. NA stands for *not applicable* and means that there are no hyperparameters. None means that no validation procedure has been conducted or that it is not specified in the original paper.

	Method	Validation Procedure	Comment
Reweighting	Density Reweight [50]	None	Bandwidth fixed by Silverman method
	Discriminative Reweight [46]	NA	No hyperparameters
	Gaussian Reweight [46]	None	Not specified in [46]
	KLIEP [51]	Integrated CV	Likelihood CV [51] on target
	KMM [21]	None	Fixed data-dependent hyperparameters
	NN Reweight [30]	None	Number of neighbors fixed to one
	MMDTarS [60]	CV	Not specified if done on source or target
Mapping	Coral [52]	NA	No hyperparameters
	OT mapping [7]	CV target/CircCV	Unclear in the text
	Lin. OT mapping [13]	NA	No hyperparameters
	MMD-LS [60]	CV	Not specified if done on source or target
Subsp.	SA [12]	2-fold CV on source	-
	TCA [39]	Validation on target	Target subset used to tune parameters
	TSL [47]	None	Not specified in [47]
Other	JDOT [6]	Reverse CV [61]	-
	OT label prop [48]	NA	No hyperparameters
	DASVM [3]	Circular Validation [3]	-

In Table A.1, we provide additional information on the validation procedures used in the original papers that proposed the different domain adaptation methods implemented in SKADA-Bench. The first column is the name of the method, the second column contains the procedure used to select hyperparameters and the last column provides additional details. What is striking is that many methods do not conduct or specify a validation procedure to select the hyperparameters, which limits the performance of the proposed method on a novel dataset. Several others rely on cross-validation using target data. However, since target labels are typically unavailable in practical scenarios, this validation approach is unrealistic. Overall, many methods have been evaluated with unrealistic or not reproducible validation procedures, making the performance of the proposed methods appear over-optimistic. A key contribution of our work is the extensive comparison of realistic, unsupervised scorers for selecting optimal hyperparameters and base estimators in DA methods.

B Datasets description and preprocessing

The simulated dataset proves that DA methods can work well under the proper shift (see Table 2). However, in real-world applications, we do not have prior knowledge of the type of data shift. Hence, finding the appropriate domain-adaptation method between reweighting, mapping, and subspace methods is a challenging task. In this section, we introduce 8 real-world datasets coming from different fields. Table 1 summarizes the 8 classification datasets used in this benchmark with the corresponding data modality, preprocessing, number of source-target pairs (# adapt), number of classes, samples, and feature dimensions.

Computer Vision. First, three computer vision datasets are proposed: Office31 [26], Office Home [55], and MNIST/USPS [28]. We create embeddings for Office31 using the Decaff preprocessing method [10] and for Office Home using a pre-trained ResNet50 [19]. These embeddings, as well as vectorized MNIST/USPS, are dimensionally reduced with a Principal Component Analysis (PCA). These three datasets encompass 3, 4, and 3 domains, respectively and all pairs of adaptations are used as DA problems. MNIST/USPS contain clear and blurry images digits, Office31 differentiates between images captured by various devices, while for OfficeHome, its by image style.

NLP. The second task is Natural Language Processing (NLP). Two datasets are studied: 20News-group [27] and Amazon Review [33]. The 20Newsdataset contains 20.000 documents categorized into 4 categories: *talk*, *rec*, *comp*, and *sci*. The learning task is to classify documents across

categories. First, the documents are embedded using a Large Language Model (LLM) [43, 57], and then PCA is applied for dimensionality reduction.

For the Amazon Review dataset, the task is to classify comment ratings. This dataset spans four domains (Books, DVDs, Kitchen, Electronics), and the domain shift results from these varying types of objects. Similar to the 20Newsgroup dataset, comments are embedded using the same LLM and then reduced in dimensionality using a PCA.

Tabular data. We propose two tabular datasets. The first one is the Mushroom dataset [8], where the task is to classify whether a mushroom is poisonous or not. The two domains are separated according to the mushroom’s stalk shape (enlarging vs. tapering). The tabular data are one-hot-encoded to transform categorical data into numerical data. The second dataset is Phishing [34]. The classification problem involves determining whether a webpage is a phishing or a legitimate one. The domains are separated according to the availability of the IP address. Since the data are already numerical, no preprocessing is done on this dataset.

Biosignals. The last task is BCI Motor Imagery. The dataset used is BCI Competition IV [54], often used in the literature [1]. The task is to classify four kinds of motor imagery (right hand, left hand, feet, and tongue) from EEG data. In this dataset, nine subjects are available. The domains are separated based on session number. For each subject, session 1 is considered as the source domain and session 2 is considered as the target domain. The data are multivariate signals. To embed the data, we first compute the covariance and then project this covariance on the Tangent Space as proposed in [1].

C Adding new methods and datasets to SKADA-Bench

Using the `benchopt` framework for this benchmark allows users to easily add novel domain adaptation (DA) methods and datasets. To that end, users should adhere to the `benchopt` [35] conventions. We provide below the guidelines with examples in Python to add a new DA method and a new dataset to SKADA-Bench.

C.1 Adding a new DA method

A new DA method can be easily added with the following:

- Create file with a class called `Solver` that inherits from `DA Solver` and place it in the `solvers` folder.
- This class should implement a `get_estimator()` function, which returns a class inheriting from `sklearn.BaseEstimator` and accepts `sample_weight` as fit parameter. In the benchmark we used the Domain Adaptation toolbox SKADA [17] that provides many esDA estimatos with correct interface.

We provide below an example of Python implementation to add a new DA method to SKADA-Bench.


```

# Python snippet code to add a DA method
from benchmark_utils.base_solver import DASolver
from sklearn.base import BaseEstimator

class MyDAEstimator(BaseEstimator):
    def __init__(self, param1=10, param2='auto'):
        self.param1 = param1
        self.param2 = param2

    def fit(self, X, y, sample_weight=None):
        # sample_weight<0 are source samples
        # sample_weight>=0 are target samples
        # y contains -1 for masked target samples
        # Your code here : store stuff in self for later predict
        return self

    def predict(self, X):
        # do prediction on target domain here
        return ypred

    def predict_proba(self, X):
        # do probabilistic prediction on target domain here
        return proba

class Solver(DASolver):
    name = "My_DA_method"

    # Param grid to validate
    default_param_grid = {
        'param1': [10, 100],
        'param2': ['auto', 'manual']
    }

    def get_estimator(self):
        return MyDAEstimator()

```

C.2 Adding a new dataset

A new DA dataset can be easily added with the following:

- Create a file with a class called Dataset that inherits from BaseDataset and place it in the datasets folder.
- This class should implement a get_data() function, which returns a dictionary with keys X, y, and sample_domain.

We provide below an example of Python implementation to add a new dataset to SKADA-Bench.

```

# Python snippet code to add a dataset

from benchopt import BaseDataset
from sklearn.datasets import make_blobs
import numpy as np

class Dataset(BaseDataset):
    name = "example_dataset"

    def get_data(self):
        X_source, y_source = make_blobs(
            n_samples=100, centers=3,
            n_features=2, random_state=0
        )

        X_target, y_target = make_blobs(
            n_samples=100, centers=5,
            n_features=2, random_state=42
        )
        # sample_domain is negative for target sampels and positive for source
        sample_domain = np.array([1]*len(X_source) + [-2]*len(X_target))

        return dict(
            X=np.concatenate((X_source, X_target), axis=0)
            y=np.concatenate((y_source, y_target))
            sample_domain=sample_domain
        )

```

By following these guidelines, users can seamlessly integrate their own datasets and DA methods into SKADA-Bench. It results in a user-friendly benchmark that enables fast, reproducible, and reliable comparisons of common and novel DA methods and datasets. We will provide users with precomputed result files and utilities, allowing them to run only the new methods or datasets. This will speed up new comparisons and avoid unnecessary computations.

D Benchmark detailed results

D.1 Results per datasets

In Table 2 of the main paper, the reported performance for each method on a given dataset is an average over the number of shifts, i.e., the number of source-target pairs denoted by #adapt in Table 1. In this section, we provide additional details on the performance of methods for each shift in each dataset. These results are presented in separate tables for each dataset

These detailed tables where cell in green denote a gain wrt Train Src (average outside of standard deviation of Train Src) better illustrate the challenges of domain adaptation (DA) methods. They show that not all shifts are equivalent within a given dataset. For example, Table D.5 reveals that only 4 shifts in the AmazonReview dataset present a DA problem (defined as a $> 3\%$ difference in accuracy between Train Src and Train Tgt). While for the other shifts, we achieve similar performance whether we train on source or target data. Additionally, some specific shifts present a DA problem that no method can successfully address. This can be seen in the dsl \rightarrow amz shift in the Office31 dataset, as shown in Table D.2. Finally, some DA methods perform consistently across all shifts within a dataset, as demonstrated by the results for the 20Newsgroup dataset in Table D.4.

Table D.1: Accuracy score for MNIST/USPS dataset for each shift compared for all the methods. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

		MNIST→USPS	USPS→MNIST	Mean	Rank
	Train Src	0.66 ± 0.02	0.43 ± 0.02	0.54 ± 0.02	11.50
	Train Tgt	0.96 ± 0.0	0.96 ± 0.01	0.96 ± 0.01	1.00
Reweighting	Dens. RW	0.66 ± 0.02	0.42 ± 0.02	0.54 ± 0.02	12.75
	Disc. RW	0.6 ± 0.01	0.4 ± 0.02	0.5 ± 0.02	18.00
	Gauss. RW	0.11 ± 0.01	0.11 ± 0.01	0.11 ± 0.01	20.50
	KLIEP	0.66 ± 0.02	0.43 ± 0.02	0.54 ± 0.02	12.50
	KMM	0.64 ± 0.02	0.41 ± 0.03	0.52 ± 0.02	17.00
	NN RW	0.66 ± 0.02	0.42 ± 0.02	0.54 ± 0.02	11.50
	MMDTarS	0.66 ± 0.02	0.42 ± 0.02	0.54 ± 0.02	11.25
Mapping	CORAL	0.74 ± 0.01	0.51 ± 0.01	0.62 ± 0.01	5.50
	MapOT	0.69 ± 0.02	0.54 ± 0.02	0.61 ± 0.02	4.00
	EntOT	0.66 ± 0.02	0.54 ± 0.02	0.6 ± 0.02	5.00
	ClassRegOT	0.69 ± 0.03	0.54 ± 0.03	0.62 ± 0.03	4.50
	LinOT	0.74 ± 0.02	0.53 ± 0.02	0.64 ± 0.02	3.25
	MMD-LS	0.63 ± 0.05	0.41 ± 0.04	0.52 ± 0.05	15.50
Subspace	JPCA	0.66 ± 0.02	0.37 ± 0.02	0.51 ± 0.02	12.50
	SA	0.71 ± 0.03	0.41 ± 0.02	0.56 ± 0.02	10.50
	TCA	0.1 ± 0.05	0.12 ± 0.03	0.11 ± 0.04	20.50
	TSL	0.31 ± 0.05	0.19 ± 0.01	0.25 ± 0.03	19.00
Other	JDOT	0.73 ± 0.02	0.53 ± 0.02	0.63 ± 0.02	3.00
	OTLabelProp	0.71 ± 0.03	0.53 ± 0.02	0.62 ± 0.02	6.50

Table D.2: Accuracy score for Office31 dataset for each shift compared for all the methods. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

	amz-dsl	amz-web	dsl-amz	dsl-web	web-amz	Mean	Rank
Train Src	0.55 ± 0.04	0.51 ± 0.04	0.5 ± 0.02	0.9 ± 0.02	0.49 ± 0.01	0.59 ± 0.03	9.4
Train Tgt	0.93 ± 0.03	0.95 ± 0.01	0.78 ± 0.01	0.95 ± 0.02	0.77 ± 0.01	0.88 ± 0.02	1.0
Dens. RW	0.49 ± 0.05	0.51 ± 0.04	0.5 ± 0.03	0.85 ± 0.09	0.48 ± 0.01	0.57 ± 0.04	11.6
Disc. RW	0.57 ± 0.07	0.51 ± 0.04	0.49 ± 0.03	0.87 ± 0.04	0.46 ± 0.0	0.58 ± 0.03	12.6
Gauss. RW	0.34 ± 0.04	0.1 ± 0.03	0.07 ± 0.01	0.38 ± 0.03	0.1 ± 0.01	0.2 ± 0.03	20.0
KLIEP	0.55 ± 0.04	0.51 ± 0.04	0.49 ± 0.02	0.9 ± 0.02	0.49 ± 0.01	0.59 ± 0.02	10.8
KMM	0.56 ± 0.06	0.54 ± 0.03	0.49 ± 0.03	0.86 ± 0.02	0.47 ± 0.02	0.58 ± 0.03	11.7
NN RW	0.53 ± 0.04	0.5 ± 0.03	0.49 ± 0.02	0.91 ± 0.02	0.48 ± 0.01	0.58 ± 0.02	12.7
MMDTarS	0.54 ± 0.05	0.51 ± 0.04	0.48 ± 0.02	0.76 ± 0.14	0.48 ± 0.01	0.56 ± 0.05	13.0
CORAL	0.53 ± 0.04	0.53 ± 0.03	0.5 ± 0.02	0.91 ± 0.03	0.5 ± 0.02	0.59 ± 0.03	6.6
MapOT	0.46 ± 0.04	0.52 ± 0.03	0.47 ± 0.02	0.84 ± 0.03	0.43 ± 0.02	0.55 ± 0.03	12.0
EntOT	0.52 ± 0.02	0.56 ± 0.02	0.49 ± 0.03	0.88 ± 0.03	0.47 ± 0.01	0.58 ± 0.02	12.5
ClassRegOT	0.59 ± 0.08	0.62 ± 0.03	0.49 ± 0.03	0.86 ± 0.04	0.49 ± 0.01	0.61 ± 0.04	7.0
LinOT	0.54 ± 0.05	0.55 ± 0.04	0.5 ± 0.02	0.9 ± 0.02	0.49 ± 0.01	0.59 ± 0.03	8.0
MMD-LS	0.46 ± 0.07	0.49 ± 0.06	0.49 ± 0.03	0.89 ± 0.04	0.43 ± 0.04	0.55 ± 0.05	14.7
JPCA	0.56 ± 0.03	0.51 ± 0.04	0.42 ± 0.02	0.84 ± 0.03	0.44 ± 0.02	0.55 ± 0.03	11.6
SA	0.56 ± 0.03	0.51 ± 0.03	0.49 ± 0.02	0.89 ± 0.01	0.49 ± 0.01	0.59 ± 0.02	11.3
TCA	0.04 ± 0.02	0.03 ± 0.01	0.04 ± 0.0	0.04 ± 0.0	0.04 ± 0.0	0.04 ± 0.0	21.0
TSL	0.55 ± 0.04	0.51 ± 0.03	0.5 ± 0.02	0.9 ± 0.02	0.49 ± 0.01	0.59 ± 0.03	7.4
JDOT	0.63 ± 0.04	0.64 ± 0.01	0.5 ± 0.02	0.77 ± 0.03	0.49 ± 0.01	0.6 ± 0.02	4.2
OTLabelProp	0.58 ± 0.03	0.63 ± 0.01	0.48 ± 0.04	0.88 ± 0.02	0.46 ± 0.01	0.61 ± 0.02	11.2

Table D.3: Accuracy score for OfficeHome dataset for each shift compared for all the methods. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

	art→clipart	art→product	art→realworld	clipart→art	clipart→product	clipart→realworld	product→art	product→clipart	product→realworld	realworld→art	realworld→clipart	realworld→product	Mean	Rank
Train Src	0.34 ± 0.02	0.63 ± 0.01	0.7 ± 0.01	0.46 ± 0.02	0.6 ± 0.01	0.63 ± 0.01	0.5 ± 0.02	0.36 ± 0.02	0.71 ± 0.01	0.64 ± 0.01	0.38 ± 0.01	0.74 ± 0.02	0.56 ± 0.02	10.08
Train Tgt	0.71 ± 0.02	0.92 ± 0.01	0.84 ± 0.01	0.73 ± 0.03	0.91 ± 0.01	0.83 ± 0.0	0.73 ± 0.01	0.7 ± 0.01	0.83 ± 0.01	0.75 ± 0.02	0.7 ± 0.01	0.91 ± 0.01	0.8 ± 0.01	1.00
Reweighting	Dens. RW	0.34 ± 0.02	0.63 ± 0.02	0.7 ± 0.01	0.46 ± 0.02	0.56 ± 0.07	0.63 ± 0.01	0.5 ± 0.02	0.36 ± 0.02	0.71 ± 0.01	0.64 ± 0.01	0.36 ± 0.04	0.55 ± 0.02	9.62
	Disc. RW	0.32 ± 0.02	0.56 ± 0.03	0.7 ± 0.01	0.43 ± 0.01	0.53 ± 0.01	0.56 ± 0.02	0.52 ± 0.02	0.35 ± 0.01	0.72 ± 0.01	0.63 ± 0.01	0.26 ± 0.01	0.53 ± 0.01	14.33
	Gauss. RW	0.22 ± 0.01	0.43 ± 0.01	0.6 ± 0.02	0.36 ± 0.02	0.46 ± 0.02	0.49 ± 0.02	0.41 ± 0.02	0.25 ± 0.02	0.61 ± 0.01	0.53 ± 0.01	0.67 ± 0.01	0.44 ± 0.02	18.83
	KLIEP	0.34 ± 0.02	0.63 ± 0.02	0.7 ± 0.01	0.46 ± 0.02	0.6 ± 0.01	0.63 ± 0.02	0.51 ± 0.02	0.35 ± 0.02	0.71 ± 0.01	0.63 ± 0.01	0.38 ± 0.01	0.56 ± 0.02	10.08
	KMM	0.31 ± 0.03	0.62 ± 0.02	0.69 ± 0.01	0.48 ± 0.03	0.6 ± 0.01	0.65 ± 0.02	0.48 ± 0.01	0.35 ± 0.02	0.71 ± 0.01	0.59 ± 0.01	0.36 ± 0.01	0.55 ± 0.02	14.00
	NN RW	0.34 ± 0.02	0.63 ± 0.01	0.7 ± 0.01	0.46 ± 0.02	0.59 ± 0.01	0.62 ± 0.02	0.51 ± 0.02	0.35 ± 0.02	0.71 ± 0.01	0.63 ± 0.01	0.37 ± 0.01	0.55 ± 0.02	12.50
Mapping	MMDIarS	0.34 ± 0.02	0.63 ± 0.02	0.63 ± 0.01	0.46 ± 0.01	0.6 ± 0.01	0.63 ± 0.01	0.51 ± 0.02	0.36 ± 0.02	0.71 ± 0.01	0.64 ± 0.01	0.38 ± 0.01	0.55 ± 0.03	8.67
	CORAL	0.39 ± 0.02	0.64 ± 0.01	0.7 ± 0.01	0.48 ± 0.02	0.59 ± 0.01	0.6 ± 0.01	0.54 ± 0.02	0.38 ± 0.01	0.72 ± 0.01	0.64 ± 0.01	0.43 ± 0.01	0.57 ± 0.01	6.62
	MapOT	0.35 ± 0.02	0.54 ± 0.02	0.64 ± 0.01	0.41 ± 0.02	0.51 ± 0.01	0.54 ± 0.01	0.45 ± 0.02	0.37 ± 0.02	0.64 ± 0.02	0.6 ± 0.02	0.41 ± 0.01	0.51 ± 0.02	12.67
	EntOT	0.38 ± 0.02	0.65 ± 0.01	0.71 ± 0.01	0.51 ± 0.01	0.62 ± 0.02	0.66 ± 0.02	0.55 ± 0.02	0.38 ± 0.01	0.72 ± 0.02	0.64 ± 0.01	0.42 ± 0.01	0.58 ± 0.01	4.92
	ClassRegOT	0.33 ± 0.04	0.61 ± 0.07	0.67 ± 0.05	0.44 ± 0.05	0.54 ± 0.05	0.62 ± 0.05	0.51 ± 0.06	0.34 ± 0.03	0.66 ± 0.02	0.62 ± 0.03	0.38 ± 0.04	0.69 ± 0.05	13.67
	LinOT	0.39 ± 0.02	0.64 ± 0.02	0.7 ± 0.01	0.5 ± 0.02	0.59 ± 0.01	0.61 ± 0.01	0.54 ± 0.02	0.38 ± 0.01	0.72 ± 0.01	0.64 ± 0.01	0.43 ± 0.01	0.75 ± 0.03	5.67
Subspace	MMD-I-S	0.34 ± 0.02	0.6 ± 0.04	0.68 ± 0.03	0.46 ± 0.01	0.56 ± 0.09	0.58 ± 0.1	0.5 ± 0.02	0.32 ± 0.06	0.71 ± 0.01	0.63 ± 0.01	0.36 ± 0.05	0.54 ± 0.04	12.79
	JPCA	0.33 ± 0.02	0.63 ± 0.01	0.7 ± 0.01	0.44 ± 0.02	0.58 ± 0.01	0.61 ± 0.02	0.18 ± 0.02	0.17 ± 0.01	0.65 ± 0.01	0.48 ± 0.05	0.19 ± 0.05	0.47 ± 0.02	13.54
	SA	0.39 ± 0.02	0.64 ± 0.01	0.71 ± 0.01	0.47 ± 0.02	0.6 ± 0.01	0.63 ± 0.01	0.52 ± 0.01	0.38 ± 0.01	0.72 ± 0.01	0.64 ± 0.01	0.39 ± 0.02	0.57 ± 0.01	6.38
	TCA	0.02 ± 0.0	0.01 ± 0.0	0.02 ± 0.0	0.02 ± 0.01	0.02 ± 0.0	0.02 ± 0.01	0.01 ± 0.01	NA	NA	0.04 ± 0.01	NA	0.02 ± 0.0	21.00
Other	TSL	0.09 ± 0.02	0.17 ± 0.05	0.32 ± 0.01	0.11 ± 0.01	0.15 ± 0.05	0.16 ± 0.04	0.16 ± 0.03	0.12 ± 0.01	0.35 ± 0.01	0.26 ± 0.09	0.13 ± 0.01	0.2 ± 0.03	20.00
	JDOT	0.35 ± 0.02	0.5 ± 0.01	0.62 ± 0.02	0.42 ± 0.03	0.54 ± 0.01	0.55 ± 0.02	0.48 ± 0.01	0.37 ± 0.02	0.64 ± 0.02	0.59 ± 0.02	0.4 ± 0.01	0.51 ± 0.02	9.08
OTLabelProp	0.32 ± 0.03	0.57 ± 0.01	0.66 ± 0.02	0.5 ± 0.02	0.63 ± 0.01	0.63 ± 0.02	0.53 ± 0.01	0.39 ± 0.01	0.7 ± 0.02	0.63 ± 0.02	0.43 ± 0.01	0.73 ± 0.02	0.56 ± 0.02	9.67

Table D.4: Accuracy score for 20NewsGroups dataset for each shift compared for all the methods. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

	Rec→sci	Rec→talk	sci→rec	sci→talk	talk→rec	talk→sci	Mean	Rank	
Train Src	0.52 ± 0.03	0.56 ± 0.05	0.54 ± 0.05	0.7 ± 0.01	0.52 ± 0.05	0.71 ± 0.03	0.59 ± 0.04	18.33	
Train Tgt	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.99 ± 0.0	1.0 ± 0.0	0.99 ± 0.0	1.0 ± 0.0	1.00	
Reweighting	Dens. RW	0.49 ± 0.03	0.56 ± 0.05	0.53 ± 0.05	0.69 ± 0.03	0.71 ± 0.03	0.58 ± 0.04	18.33	
	Disc. RW	0.45 ± 0.03	0.61 ± 0.07	0.44 ± 0.02	0.76 ± 0.01	0.57 ± 0.04	0.6 ± 0.03	15.50	
	Gauss. RW	0.7 ± 0.07	0.45 ± 0.0	0.66 ± 0.11	0.49 ± 0.01	0.44 ± 0.0	0.48 ± 0.02	0.54 ± 0.04	18.50
	KLJEP	0.52 ± 0.03	0.56 ± 0.05	0.57 ± 0.1	0.7 ± 0.01	0.52 ± 0.05	0.71 ± 0.03	0.6 ± 0.05	17.25
	KMM	0.65 ± 0.06	0.71 ± 0.03	0.7 ± 0.06	0.74 ± 0.02	0.65 ± 0.14	0.75 ± 0.03	0.7 ± 0.06	12.17
Reweighting	NN RW	0.52 ± 0.06	0.57 ± 0.06	0.54 ± 0.03	0.7 ± 0.01	0.52 ± 0.06	0.59 ± 0.04	17.08	
	MMDTarS	0.52 ± 0.03	0.57 ± 0.05	0.54 ± 0.05	0.7 ± 0.01	0.52 ± 0.06	0.59 ± 0.04	14.58	
Mapping	CORAL	0.61 ± 0.03	0.79 ± 0.02	0.62 ± 0.01	0.76 ± 0.01	0.78 ± 0.01	0.73 ± 0.02	11.83	
	MapOT	0.67 ± 0.02	0.83 ± 0.01	0.68 ± 0.02	0.78 ± 0.02	0.83 ± 0.01	0.79 ± 0.02	7.08	
	EntOT	0.82 ± 0.0	0.82 ± 0.02	0.78 ± 0.05	0.87 ± 0.01	0.82 ± 0.02	0.87 ± 0.01	0.83 ± 0.02	6.50
	ClassRegOT	0.97 ± 0.02	0.99 ± 0.01	0.98 ± 0.0	0.94 ± 0.04	0.99 ± 0.01	0.92 ± 0.06	0.96 ± 0.02	2.50
	LinOT	0.72 ± 0.02	0.92 ± 0.0	0.72 ± 0.01	0.79 ± 0.11	0.91 ± 0.01	0.83 ± 0.07	0.82 ± 0.04	6.83
	MMD-LS	0.99 ± 0.0	0.98 ± 0.0	0.99 ± 0.0	0.94 ± 0.01	0.98 ± 0.0	0.93 ± 0.01	0.97 ± 0.01	2.50
Subspace	JPCA	0.69 ± 0.03	0.85 ± 0.06	0.71 ± 0.03	0.75 ± 0.03	0.87 ± 0.05	0.77 ± 0.04	6.33	
	SA	0.9 ± 0.08	0.91 ± 0.08	0.87 ± 0.06	0.87 ± 0.01	0.85 ± 0.06	0.87 ± 0.01	0.88 ± 0.05	4.83
	TCA	0.58 ± 0.13	0.58 ± 0.05	0.56 ± 0.1	0.6 ± 0.07	0.56 ± 0.02	0.56 ± 0.01	0.57 ± 0.06	17.25
	TSL	0.75 ± 0.04	0.64 ± 0.05	0.74 ± 0.04	0.68 ± 0.02	0.62 ± 0.05	0.68 ± 0.03	0.68 ± 0.04	11.83
Other	JDOT	0.67 ± 0.02	0.84 ± 0.01	0.68 ± 0.02	0.79 ± 0.02	0.84 ± 0.01	0.8 ± 0.01	6.33	
	OTLabelProp	0.78 ± 0.01	0.95 ± 0.01	0.78 ± 0.01	0.84 ± 0.05	0.95 ± 0.01	0.83 ± 0.04	5.00	
	DASVM	0.57 ± 0.11	0.75 ± 0.02	0.52 ± 0.06	0.72 ± 0.01	0.81 ± 0.03	0.73 ± 0.03	0.68 ± 0.04	14.00

Table D.5: Accuracy score for AmazonReview dataset for each shift compared for all the methods. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

	books→electronics	books→kitchen	dvd→books	dvd→electronics	dvd→kitchen	electronics→books	electronics→dvd	electronics→kitchen	kitchen→books	kitchen→dvd	kitchen→electronics	Mean	Rank
Train Src	0.65 ± 0.03	0.71 ± 0.02	0.72 ± 0.01	0.66 ± 0.02	0.72 ± 0.01	0.69 ± 0.01	0.71 ± 0.01	0.75 ± 0.01	0.71 ± 0.02	0.7 ± 0.02	0.71 ± 0.01	0.7 ± 0.01	3.45
Train Tgt	0.71 ± 0.02	0.77 ± 0.01	0.74 ± 0.01	0.69 ± 0.03	0.76 ± 0.01	0.72 ± 0.01	0.72 ± 0.02	0.76 ± 0.01	0.72 ± 0.01	0.72 ± 0.01	0.71 ± 0.01	0.73 ± 0.01	1.00
Dens. RW	0.65 ± 0.03	0.71 ± 0.02	0.72 ± 0.01	0.65 ± 0.02	0.72 ± 0.01	0.69 ± 0.01	0.71 ± 0.01	0.74 ± 0.01	0.71 ± 0.02	0.7 ± 0.02	0.71 ± 0.01	0.7 ± 0.02	7.05
Disc. RW	0.63 ± 0.01	0.71 ± 0.01	0.72 ± 0.01	0.64 ± 0.02	0.7 ± 0.01	0.68 ± 0.03	0.62 ± 0.05	0.73 ± 0.01	0.7 ± 0.02	0.69 ± 0.01	0.7 ± 0.01	0.68 ± 0.02	7.77
Gauss. RW	0.56 ± 0.03	0.63 ± 0.01	0.54 ± 0.06	0.44 ± 0.08	0.58 ± 0.08	0.64 ± 0.03	0.62 ± 0.02	0.69 ± 0.03	0.66 ± 0.03	0.68 ± 0.0	0.61 ± 0.02	0.6 ± 0.04	16.18
KLJEP	0.65 ± 0.03	0.69 ± 0.04	0.72 ± 0.01	0.66 ± 0.02	0.72 ± 0.01	0.69 ± 0.03	0.68 ± 0.04	0.75 ± 0.01	0.65 ± 0.06	0.68 ± 0.03	0.71 ± 0.01	0.69 ± 0.03	6.64
KMM	0.52 ± 0.04	0.63 ± 0.01	0.6 ± 0.04	0.5 ± 0.02	0.64 ± 0.04	0.28 ± 0.2	0.57 ± 0.19	0.63 ± 0.05	0.66 ± 0.03	0.64 ± 0.03	0.58 ± 0.02	0.57 ± 0.06	17.09
NN RW	0.59 ± 0.05	0.69 ± 0.01	0.71 ± 0.03	0.51 ± 0.1	0.66 ± 0.06	0.68 ± 0.03	0.68 ± 0.02	0.71 ± 0.04	0.7 ± 0.02	0.65 ± 0.02	0.68 ± 0.02	0.66 ± 0.03	10.36
MMDIars	0.65 ± 0.03	0.7 ± 0.02	0.72 ± 0.01	0.67 ± 0.01	0.72 ± 0.01	0.7 ± 0.01	0.71 ± 0.02	0.74 ± 0.01	0.71 ± 0.02	0.7 ± 0.02	0.71 ± 0.02	0.7 ± 0.02	7.50
CORAL	0.62 ± 0.03	0.72 ± 0.01	0.73 ± 0.01	0.61 ± 0.01	0.72 ± 0.01	0.69 ± 0.01	0.7 ± 0.02	0.74 ± 0.01	0.7 ± 0.01	0.7 ± 0.01	0.7 ± 0.02	0.69 ± 0.01	4.59
MapOT	0.6 ± 0.01	0.69 ± 0.0	0.7 ± 0.01	0.59 ± 0.01	0.71 ± 0.01	0.69 ± 0.02	0.69 ± 0.01	0.73 ± 0.02	0.69 ± 0.02	0.69 ± 0.01	0.64 ± 0.01	0.67 ± 0.01	9.59
EntOT	0.53 ± 0.01	0.62 ± 0.0	0.64 ± 0.0	0.53 ± 0.0	0.62 ± 0.0	0.63 ± 0.0	0.66 ± 0.01	0.74 ± 0.01	0.64 ± 0.0	0.64 ± 0.0	0.54 ± 0.0	0.62 ± 0.0	14.68
ClassRegOT	0.66 ± 0.02	0.73 ± 0.02	0.72 ± 0.01	0.64 ± 0.02	0.73 ± 0.01	0.63 ± 0.02	0.62 ± 0.02	0.68 ± 0.03	0.66 ± 0.01	0.66 ± 0.03	0.69 ± 0.02	0.68 ± 0.02	12.27
LinOT	0.64 ± 0.03	0.73 ± 0.02	0.73 ± 0.01	0.61 ± 0.01	0.73 ± 0.0	0.69 ± 0.02	0.7 ± 0.02	0.74 ± 0.01	0.71 ± 0.02	0.7 ± 0.02	0.7 ± 0.02	0.7 ± 0.02	3.59
MMD-LS	0.58 ± 0.08	0.67 ± 0.03	0.72 ± 0.01	0.66 ± 0.02	0.58 ± 0.18	0.69 ± 0.01	0.71 ± 0.01	0.75 ± 0.01	0.71 ± 0.02	0.7 ± 0.02	0.71 ± 0.01	0.68 ± 0.04	9.14
JPCA	0.62 ± 0.02	0.7 ± 0.01	0.72 ± 0.01	0.63 ± 0.01	0.7 ± 0.02	0.7 ± 0.01	0.71 ± 0.01	0.75 ± 0.01	0.69 ± 0.02	0.68 ± 0.01	0.69 ± 0.03	0.69 ± 0.02	6.73
SA	0.63 ± 0.0	0.7 ± 0.01	0.69 ± 0.01	0.63 ± 0.0	0.7 ± 0.01	0.64 ± 0.02	0.64 ± 0.02	0.69 ± 0.01	0.68 ± 0.01	0.67 ± 0.01	0.64 ± 0.01	0.66 ± 0.01	10.64
TCA	0.52 ± 0.0	0.62 ± 0.0	0.64 ± 0.0	0.52 ± 0.0	0.62 ± 0.0	0.64 ± 0.0	0.64 ± 0.0	0.62 ± 0.0	0.64 ± 0.0	0.64 ± 0.0	0.52 ± 0.0	0.6 ± 0.0	17.36
TSL	0.65 ± 0.03	0.71 ± 0.02	0.72 ± 0.01	0.66 ± 0.02	0.72 ± 0.01	0.69 ± 0.01	0.71 ± 0.01	0.75 ± 0.01	0.71 ± 0.02	0.7 ± 0.02	0.71 ± 0.01	0.7 ± 0.01	5.86
JDOT	0.61 ± 0.01	0.7 ± 0.01	0.71 ± 0.01	0.6 ± 0.01	0.7 ± 0.01	0.63 ± 0.01	0.64 ± 0.02	0.72 ± 0.03	0.69 ± 0.02	0.69 ± 0.01	0.67 ± 0.03	0.67 ± 0.01	10.27
OTLabelProp	0.59 ± 0.02	0.69 ± 0.01	0.7 ± 0.01	0.59 ± 0.01	0.7 ± 0.01	0.69 ± 0.02	0.68 ± 0.02	0.74 ± 0.01	0.69 ± 0.02	0.69 ± 0.01	0.63 ± 0.01	0.67 ± 0.01	9.41

Table D.6: Accuracy score for Mushrooms dataset for each shift compared for all the methods. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

		enl→tap	tap→enl	Mean	Rank
	Train Src	0.67 ± 0.01	0.77 ± 0.01	0.72 ± 0.01	9.75
	Train Tgt	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.00
Reweighting	Dens. RW	0.67 ± 0.01	0.76 ± 0.0	0.71 ± 0.01	10.50
	Disc. RW	0.73 ± 0.06	0.78 ± 0.01	0.75 ± 0.04	5.00
	Gauss. RW	0.56 ± 0.0	0.46 ± 0.0	0.51 ± 0.0	20.75
	KLIEP	0.66 ± 0.02	0.77 ± 0.01	0.72 ± 0.01	12.50
	KMM	0.7 ± 0.02	0.78 ± 0.01	0.74 ± 0.01	7.00
	NN RW	0.67 ± 0.05	0.75 ± 0.01	0.71 ± 0.03	14.00
	MMDTarS	0.7 ± 0.02	0.77 ± 0.01	0.74 ± 0.01	7.50
Mapping	CORAL	0.66 ± 0.02	0.77 ± 0.01	0.72 ± 0.02	13.50
	MapOT	0.65 ± 0.01	0.62 ± 0.02	0.63 ± 0.01	16.50
	EntOT	0.82 ± 0.01	0.67 ± 0.01	0.75 ± 0.01	9.00
	ClassRegOT	0.92 ± 0.01	0.72 ± 0.01	0.82 ± 0.01	8.00
	LinOT	0.72 ± 0.01	0.81 ± 0.01	0.76 ± 0.01	4.50
	MMD-LS	0.88 ± 0.01	0.83 ± 0.01	0.86 ± 0.01	3.50
Subspace	JPCA	0.76 ± 0.03	0.81 ± 0.0	0.78 ± 0.02	6.00
	SA	0.93 ± 0.05	0.84 ± 0.01	0.88 ± 0.03	2.50
	TCA	0.45 ± 0.04	0.46 ± 0.0	0.45 ± 0.02	21.75
	TSL	0.62 ± 0.08	0.51 ± 0.01	0.56 ± 0.04	19.00
Other	JDOT	0.67 ± 0.02	0.6 ± 0.01	0.63 ± 0.01	14.50
	OTLabelProp	0.68 ± 0.01	0.61 ± 0.01	0.64 ± 0.01	13.00
	DASVM	0.71 ± 0.02	0.85 ± 0.0	0.78 ± 0.01	4.00

Table D.7: Accuracy score for Phishing dataset for each shift compared for all the methods. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

		$ip_address \rightarrow no_ip_address$	$no_ip_address \rightarrow ip_address$	Mean	Rank
	Train Src	0.94 ± 0.01	0.88 ± 0.01	0.91 ± 0.01	7.0
	Train Tgt	0.97 ± 0.01	0.97 ± 0.01	0.97 ± 0.01	1.0
Reweighting	Dens. RW	0.94 ± 0.01	0.88 ± 0.01	0.91 ± 0.01	6.5
	Disc. RW	0.94 ± 0.01	0.88 ± 0.01	0.91 ± 0.01	9.5
	Gauss. RW	0.51 ± 0.0	0.41 ± 0.0	0.46 ± 0.0	20.0
	KLIEP	0.94 ± 0.01	0.89 ± 0.01	0.91 ± 0.01	5.0
	KMM	0.94 ± 0.01	0.89 ± 0.02	0.91 ± 0.01	6.5
	NN RW	0.94 ± 0.01	0.89 ± 0.01	0.91 ± 0.01	6.5
	MMDTarS	0.94 ± 0.01	0.88 ± 0.01	0.91 ± 0.01	5.5
Mapping	CORAL	0.93 ± 0.01	0.91 ± 0.01	0.92 ± 0.01	5.5
	MapOT	0.83 ± 0.01	0.84 ± 0.03	0.84 ± 0.02	15.0
	EntOT	0.87 ± 0.04	0.85 ± 0.03	0.86 ± 0.04	16.0
	ClassRegOT	0.87 ± 0.02	0.89 ± 0.02	0.88 ± 0.02	9.0
	LinOT	0.91 ± 0.01	0.9 ± 0.02	0.91 ± 0.02	7.0
	MMD-LS	NA	0.88 ± 0.01	0.88 ± 0.01	11.5
Subspace	JPCA	0.92 ± 0.01	0.89 ± 0.01	0.9 ± 0.01	8.0
	SA	0.9 ± 0.02	0.88 ± 0.02	0.89 ± 0.02	11.0
	TSL	0.88 ± 0.02	0.84 ± 0.02	0.86 ± 0.02	14.0
Other	JDOT	0.8 ± 0.02	0.8 ± 0.01	0.8 ± 0.02	18.0
	OTLabelProp	0.86 ± 0.01	0.86 ± 0.01	0.86 ± 0.01	16.0
	DASVM	NA	0.88 ± 0.01	0.88 ± 0.01	15.0

Table D.8: Accuracy score for BCI dataset for each shift compared for all the methods. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

	λ	ν	ζ	δ	ξ	ζ	δ	ξ	ζ	δ	ξ	ζ	δ	ξ	Mean	Rank
Train Src	0.59 ± 0.05	0.51 ± 0.05	0.72 ± 0.05	0.49 ± 0.06	0.39 ± 0.04	0.41 ± 0.06	0.62 ± 0.07	0.69 ± 0.08	0.54 ± 0.1	0.55 ± 0.06	8.44					
Train Tgt	0.74 ± 0.05	0.61 ± 0.07	0.82 ± 0.03	0.52 ± 0.04	0.41 ± 0.08	0.47 ± 0.06	0.75 ± 0.03	0.8 ± 0.03	0.61 ± 0.06	0.64 ± 0.05	1.44					
Reweighting ^g	Dens. RW	0.59 ± 0.06	0.51 ± 0.05	0.72 ± 0.04	0.5 ± 0.06	0.39 ± 0.04	0.4 ± 0.07	0.63 ± 0.06	0.67 ± 0.06	0.53 ± 0.09	9.72					
	Disc. RW	0.63 ± 0.06	0.51 ± 0.06	0.77 ± 0.05	0.49 ± 0.04	0.4 ± 0.05	0.4 ± 0.05	0.61 ± 0.05	0.7 ± 0.08	0.56 ± 0.09	6.22					
	Gauss. RW	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	0.25 ± 0.01	20.56					
	KLIEP	0.59 ± 0.05	0.52 ± 0.04	0.71 ± 0.04	0.48 ± 0.07	0.39 ± 0.04	0.41 ± 0.08	0.63 ± 0.07	0.69 ± 0.08	0.55 ± 0.08	8.11					
	KMM	0.61 ± 0.06	0.45 ± 0.08	0.74 ± 0.05	0.39 ± 0.1	0.31 ± 0.09	0.36 ± 0.05	0.64 ± 0.09	0.64 ± 0.01	0.52 ± 0.07	11.44					
	NN RW	0.58 ± 0.06	0.45 ± 0.05	0.69 ± 0.03	0.47 ± 0.09	0.36 ± 0.06	0.41 ± 0.05	0.63 ± 0.07	0.68 ± 0.07	0.54 ± 0.08	10.67					
MMDTars	0.59 ± 0.06	0.52 ± 0.04	0.73 ± 0.05	0.5 ± 0.06	0.39 ± 0.04	0.41 ± 0.06	0.62 ± 0.07	0.69 ± 0.07	0.54 ± 0.08	0.55 ± 0.06	8.11					
Mapping	CORAL	0.77 ± 0.08	0.56 ± 0.06	0.81 ± 0.04	0.49 ± 0.08	0.45 ± 0.05	0.44 ± 0.06	0.73 ± 0.05	0.75 ± 0.1	0.56 ± 0.06	2.67					
	MapOT	0.61 ± 0.08	0.35 ± 0.04	0.64 ± 0.05	0.39 ± 0.03	0.32 ± 0.04	0.29 ± 0.05	0.53 ± 0.04	0.61 ± 0.08	0.49 ± 0.05	11.00					
	EntOT	0.67 ± 0.07	0.45 ± 0.05	0.79 ± 0.06	0.37 ± 0.04	0.33 ± 0.06	0.32 ± 0.02	0.67 ± 0.01	0.74 ± 0.07	0.52 ± 0.06	9.39					
	ClassRegOT	0.66 ± 0.09	0.44 ± 0.07	0.66 ± 0.07	0.4 ± 0.06	0.36 ± 0.03	0.32 ± 0.06	0.62 ± 0.08	0.69 ± 0.08	0.53 ± 0.08	12.78					
	LinOT	0.76 ± 0.1	0.55 ± 0.06	0.79 ± 0.04	0.48 ± 0.09	0.46 ± 0.06	0.46 ± 0.07	0.71 ± 0.04	0.76 ± 0.09	0.53 ± 0.07	3.78					
	MMD-LS	0.68 ± 0.11	0.51 ± 0.05	0.72 ± 0.05	0.49 ± 0.06	0.39 ± 0.04	0.41 ± 0.06	0.62 ± 0.07	0.69 ± 0.08	0.54 ± 0.1	7.67					
Subspace	JPCA	0.62 ± 0.04	0.48 ± 0.09	0.71 ± 0.03	0.51 ± 0.04	0.36 ± 0.05	0.39 ± 0.08	0.63 ± 0.06	0.64 ± 0.05	0.56 ± 0.09	5.44					
	SA	0.65 ± 0.07	0.59 ± 0.05	0.73 ± 0.11	0.39 ± 0.09	0.32 ± 0.07	0.28 ± 0.05	0.59 ± 0.05	0.69 ± 0.06	0.5 ± 0.04	11.06					
	TCA	0.3 ± 0.1	0.26 ± 0.06	0.25 ± 0.04	0.26 ± 0.03	0.22 ± 0.04	0.23 ± 0.06	0.25 ± 0.05	0.36 ± 0.06	0.26 ± 0.06	19.56					
	TSL	0.26 ± 0.02	0.25 ± 0.04	0.25 ± 0.01	0.24 ± 0.04	0.22 ± 0.07	0.26 ± 0.02	0.27 ± 0.02	0.25 ± 0.05	0.25 ± 0.05	20.11					
Other	JDOT	0.59 ± 0.08	0.37 ± 0.06	0.59 ± 0.07	0.39 ± 0.05	0.28 ± 0.05	0.3 ± 0.04	0.54 ± 0.09	0.62 ± 0.05	0.51 ± 0.08	15.56					
	OTLabelProp	0.63 ± 0.06	0.41 ± 0.05	0.64 ± 0.05	0.41 ± 0.04	0.35 ± 0.07	0.33 ± 0.06	0.59 ± 0.06	0.66 ± 0.04	0.5 ± 0.09	13.11					

D.2 Impact of the base estimators on the simulated datasets

As mentioned in the main paper, it is possible to partly compensate for the shift by choosing the right base estimator. In this part, we provide the results on the Simulated dataset for three different base estimators: Logistic Regression (LR) in Table D.9, SVM in Table D.10, and XGBoost in Table D.11. Observing the two first rows for covariate shift, we see that with LR (Table D.9), there is a significant drop in performance between training on the source v.s. training on the target ($\sim 10\%$), while using SVC (Table D.10) only leads to a drop ($\sim 3\%$). Finally, using XGBoost (Table D.11) maintains the performance. The reweighting DA methods help compensate for the shift when using a simpler LR estimator. However when using an SVC, as shown in the main paper, the reweighting does not help to compensate for the covariate shift. If we look at the other shifts, the problem is harder. The subspace methods help with subspace shift, and the mapping methods help with the conditional shift.

These Tables show the importance of choosing the right base estimator. It is clear that choosing an appropriate base estimator can partially compensate for some shifts.

Table D.9: Accuracy score for simulated datasets compared for all the methods with LR. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

		<i>Cov. shift</i>	<i>Tar. shift</i>	<i>Cond. shift</i>	<i>Sub. shift</i>	Mean	Rank
	Train Src	0.8 \pm 0.02	0.81 \pm 0.03	0.68 \pm 0.03	0.06 \pm 0.01	0.59 \pm 0.02	10.50
	Train Tgt	0.91 \pm 0.02	0.92 \pm 0.01	0.79 \pm 0.03	0.97 \pm 0.01	0.9 \pm 0.02	2.00
Reweighting	Dens. RW	0.88 \pm 0.03	0.84 \pm 0.04	0.66 \pm 0.03	0.07 \pm 0.02	0.61 \pm 0.03	7.50
	Disc. RW	0.55 \pm 0.02	0.78 \pm 0.05	0.7 \pm 0.04	0.06 \pm 0.01	0.52 \pm 0.03	13.25
	Gauss. RW	0.89 \pm 0.02	0.85 \pm 0.03	0.64 \pm 0.03	0.06 \pm 0.01	0.61 \pm 0.02	8.00
	KLIEP	0.8 \pm 0.02	0.81 \pm 0.04	0.69 \pm 0.03	0.07 \pm 0.02	0.59 \pm 0.03	8.25
	KMM	0.84 \pm 0.03	0.82 \pm 0.05	0.66 \pm 0.04	0.07 \pm 0.02	0.6 \pm 0.04	7.88
	NN RW	0.81 \pm 0.02	0.82 \pm 0.04	0.67 \pm 0.03	0.07 \pm 0.01	0.59 \pm 0.03	7.75
	MMDTarS	0.8 \pm 0.02	0.84 \pm 0.04	0.66 \pm 0.03	0.07 \pm 0.02	0.59 \pm 0.03	10.75
Mapping	CORAL	0.73 \pm 0.05	0.68 \pm 0.11	0.75 \pm 0.08	0.04 \pm 0.02	0.55 \pm 0.06	12.25
	MapOT	0.73 \pm 0.03	0.6 \pm 0.04	0.79 \pm 0.03	0.03 \pm 0.01	0.54 \pm 0.03	13.75
	EntOT	0.72 \pm 0.05	0.61 \pm 0.04	0.79 \pm 0.03	0.03 \pm 0.01	0.54 \pm 0.03	12.50
	ClassRegOT	0.87 \pm 0.08	0.59 \pm 0.04	0.79 \pm 0.03	0.03 \pm 0.01	0.57 \pm 0.04	11.50
	LinOT	0.77 \pm 0.03	0.65 \pm 0.06	0.76 \pm 0.04	0.04 \pm 0.02	0.56 \pm 0.04	12.00
	MMD-LS	0.7 \pm 0.1	0.64 \pm 0.06	0.78 \pm 0.04	0.38 \pm 0.22	0.63 \pm 0.1	10.75
Subspace	JPCA	0.8 \pm 0.02	0.81 \pm 0.03	0.68 \pm 0.03	0.06 \pm 0.01	0.59 \pm 0.02	11.25
	SA	0.8 \pm 0.02	0.62 \pm 0.04	0.78 \pm 0.03	0.04 \pm 0.02	0.56 \pm 0.03	11.25
	TCA	0.44 \pm 0.29	0.49 \pm 0.06	0.54 \pm 0.11	0.54 \pm 0.23	0.5 \pm 0.17	15.50
	TSL	0.8 \pm 0.02	0.81 \pm 0.03	0.68 \pm 0.03	0.06 \pm 0.01	0.59 \pm 0.02	11.00
Other	OTLabelProp	0.73 \pm 0.03	0.59 \pm 0.04	0.79 \pm 0.03	0.03 \pm 0.01	0.53 \pm 0.03	13.50

Table D.10: Accuracy score for simulated datasets compared for all the methods with SVC. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

		<i>Cov. shift</i>	<i>Tar. shift</i>	<i>Cond. shift</i>	<i>Sub. shift</i>	Mean	Rank
	Train Src	0.88 ± 0.03	0.85 ± 0.04	0.66 ± 0.02	0.19 ± 0.03	0.65 ± 0.03	9.38
	Train Tgt	0.92 ± 0.02	0.93 ± 0.02	0.82 ± 0.03	0.98 ± 0.01	0.91 ± 0.02	1.25
Reweighting	Dens. RW	0.88 ± 0.03	0.86 ± 0.04	0.66 ± 0.02	0.18 ± 0.04	0.64 ± 0.03	8.88
	Disc. RW	0.85 ± 0.04	0.83 ± 0.04	0.72 ± 0.04	0.18 ± 0.03	0.64 ± 0.04	10.75
	Gauss. RW	0.89 ± 0.03	0.86 ± 0.04	0.65 ± 0.02	0.21 ± 0.04	0.65 ± 0.03	7.00
	KLIEP	0.88 ± 0.03	0.86 ± 0.04	0.66 ± 0.02	0.19 ± 0.03	0.65 ± 0.03	8.12
	KMM	0.89 ± 0.03	0.87 ± 0.04	0.64 ± 0.04	0.15 ± 0.05	0.64 ± 0.04	9.50
	NN RW	0.89 ± 0.03	0.86 ± 0.04	0.67 ± 0.02	0.15 ± 0.04	0.64 ± 0.03	9.12
	MMDTarS	0.88 ± 0.03	0.86 ± 0.04	0.64 ± 0.03	0.2 ± 0.04	0.65 ± 0.03	9.12
Mapping	CORAL	0.74 ± 0.04	0.7 ± 0.11	0.76 ± 0.08	0.18 ± 0.04	0.59 ± 0.07	11.50
	MapOT	0.72 ± 0.04	0.57 ± 0.04	0.82 ± 0.03	0.02 ± 0.01	0.53 ± 0.03	14.25
	EntOT	0.71 ± 0.04	0.6 ± 0.04	0.82 ± 0.03	0.12 ± 0.06	0.56 ± 0.05	12.75
	ClassRegOT	0.74 ± 0.09	0.58 ± 0.04	0.81 ± 0.03	0.11 ± 0.06	0.56 ± 0.06	12.75
	LinOT	0.73 ± 0.05	0.73 ± 0.08	0.76 ± 0.06	0.18 ± 0.04	0.6 ± 0.06	11.75
	MMD-LS	0.65 ± 0.08	0.68 ± 0.11	0.79 ± 0.05	0.55 ± 0.31	0.67 ± 0.14	10.75
Subspace	JPCA	0.88 ± 0.03	0.85 ± 0.04	0.66 ± 0.02	0.15 ± 0.05	0.64 ± 0.04	11.25
	SA	0.74 ± 0.04	0.68 ± 0.04	0.8 ± 0.03	0.11 ± 0.03	0.58 ± 0.03	12.50
	TCA	0.46 ± 0.21	0.48 ± 0.09	0.55 ± 0.11	0.56 ± 0.2	0.51 ± 0.15	15.62
	TSL	0.88 ± 0.03	0.85 ± 0.04	0.66 ± 0.02	0.19 ± 0.03	0.65 ± 0.03	9.62
Other	OTLabelProp	0.72 ± 0.04	0.58 ± 0.04	0.81 ± 0.04	0.04 ± 0.05	0.54 ± 0.04	14.00

Table D.11: Accuracy score for simulated datasets compared for all the methods with XGBoost. A white color means the method does not increase the performance compared to Train Src (Train on the source). Green indicates that the performance improved with the DA methods. The darker the color, the more significant the change.

		<i>Cov. shift</i>	<i>Tar. shift</i>	<i>Cond. shift</i>	<i>Sub. shift</i>	Mean	Rank
	Train Src	0.89 ± 0.02	0.84 ± 0.04	0.66 ± 0.03	0.21 ± 0.03	0.65 ± 0.03	9.25
	Train Tgt	0.89 ± 0.02	0.93 ± 0.02	0.77 ± 0.03	0.98 ± 0.01	0.89 ± 0.02	2.25
Reweighting	Dens. RW	0.88 ± 0.03	0.84 ± 0.03	0.67 ± 0.03	0.22 ± 0.04	0.65 ± 0.03	8.25
	Disc. RW	0.68 ± 0.06	0.84 ± 0.03	0.66 ± 0.03	0.2 ± 0.03	0.6 ± 0.04	12.25
	Gauss. RW	0.87 ± 0.03	0.84 ± 0.03	0.67 ± 0.03	0.22 ± 0.03	0.65 ± 0.03	9.12
	KLIEP	0.88 ± 0.03	0.84 ± 0.03	0.67 ± 0.03	0.21 ± 0.03	0.65 ± 0.03	7.12
	KMM	0.87 ± 0.04	0.84 ± 0.04	0.67 ± 0.04	0.22 ± 0.04	0.65 ± 0.04	7.62
	NN RW	0.88 ± 0.03	0.84 ± 0.04	0.66 ± 0.03	0.2 ± 0.03	0.65 ± 0.03	10.50
	MMDTarS	0.88 ± 0.03	0.86 ± 0.04	0.63 ± 0.03	0.22 ± 0.03	0.65 ± 0.03	7.50
Mapping	CORAL	0.71 ± 0.04	0.71 ± 0.11	0.74 ± 0.08	0.17 ± 0.05	0.58 ± 0.07	12.75
	MapOT	0.7 ± 0.04	0.59 ± 0.03	0.8 ± 0.03	0.17 ± 0.05	0.56 ± 0.04	13.25
	EntOT	0.69 ± 0.05	0.61 ± 0.04	0.8 ± 0.03	0.2 ± 0.02	0.57 ± 0.04	12.25
	ClassRegOT	0.82 ± 0.11	0.59 ± 0.03	0.8 ± 0.03	0.16 ± 0.04	0.59 ± 0.05	12.00
	LinOT	0.72 ± 0.04	0.68 ± 0.06	0.76 ± 0.04	0.19 ± 0.04	0.59 ± 0.05	12.00
	MMD-LS	0.64 ± 0.07	0.68 ± 0.08	0.78 ± 0.04	0.59 ± 0.25	0.67 ± 0.11	10.25
Subspace	JPCA	0.88 ± 0.03	0.84 ± 0.03	0.67 ± 0.03	0.14 ± 0.05	0.63 ± 0.03	10.50
	SA	0.72 ± 0.04	0.69 ± 0.04	0.78 ± 0.03	0.13 ± 0.04	0.58 ± 0.04	11.75
	TCA	0.48 ± 0.05	0.5 ± 0.05	0.51 ± 0.05	0.51 ± 0.06	0.5 ± 0.05	15.50
	TSL	0.89 ± 0.02	0.84 ± 0.04	0.66 ± 0.03	0.21 ± 0.03	0.65 ± 0.03	9.25
Other	OTLabelProp	0.72 ± 0.05	0.59 ± 0.04	0.81 ± 0.04	0.04 ± 0.05	0.54 ± 0.04	13.00

D.3 Unrealistic validation with supervised scorer

Table D.12 shows the results when we choose the supervised scorer that is when validating on target labels. It is important to highlight that this choice is impossible in real life applications due to the lack of target labels.

When using the target labels, the method’s parameters are better validated. This can be seen by the significant increase in the table (blue values), which are numerous in this table compared to the one with the selected realistic scorer. For example, the method MMDTarS, which is made for Target shift, compensates all the shift simulated covariate shifts when we select the model with a supervised scorer.

When looking at the rank, 11 DA methods have a higher rank than Train Src compared to 8 when using realistic scorer.

Table D.12: Accuracy score for all datasets compared for all the methods for simulated and real-life datasets. In this table, each DA method is validated with the supervised scorer. The color indicates the amount of the improvement. A white color means the method is not statistically different from Train Src (Train on source). Blue indicates that the performance improved with the DA methods, while red indicates a decrease. The darker the color, the more significant the change.

		Cov. shift	Tar. shift	Cond. shift	Sub. shift	Office31	OfficeHome	MNIST/USPS	20NewsGroups	AmazonReview	Mushrooms	Phishing	BCI	Rank
	Train Src	0.88	0.85	0.66	0.19	0.59	0.56	0.54	0.59	0.7	0.72	0.91	0.55	11.27
	Train Tgt	0.92	0.93	0.82	0.98	0.88	0.8	0.96	1.0	0.73	1.0	0.97	0.64	1.12
Reweighting	Dens. RW	0.89	0.87	0.67	0.2	0.59	0.56	0.54	0.59	0.7	0.76	0.91	0.55	11.05
	Disc. RW	0.86	0.84	0.73	0.23	0.58	0.53	0.54	0.62	0.69	0.78	0.91	0.56	12.19
	Gauss. RW	0.89	0.86	0.65	0.21	0.2	0.44	0.11	0.54	0.6	0.51	0.46	0.25	20.03
	KLIEP	0.89	0.88	0.66	0.2	0.59	0.56	0.54	0.58	0.7	0.75	0.92	0.55	10.93
	KMM	0.9	0.87	0.67	0.2	0.58	0.55	0.53	0.71	0.66	0.75	0.92	0.54	12.12
	NN RW	0.89	0.86	0.67	0.15	0.58	0.55	0.55	0.59	0.66	0.72	0.91	0.54	13.79
	MMDTarS	0.88	0.93	0.66	0.27	0.59	0.56	0.52	0.59	0.7	0.74	0.91	0.55	10.99
Mapping	CORAL	0.66	0.84	0.66	0.19	0.6	0.57	0.62	0.75	0.7	0.72	0.92	0.62	8.34
	MapOT	0.87	0.63	0.81	0.14	0.54	0.51	0.6	0.77	0.67	0.63	0.84	0.47	14.87
	EntOT	0.89	0.61	0.82	0.47	0.61	0.58	0.63	0.88	0.68	0.81	0.87	0.53	9.22
	ClassRegOT	0.91	0.59	0.82	0.15	0.61	0.59	0.66	0.98	0.68	0.89	0.9	0.52	6.40
	LinOT	0.89	0.81	0.81	0.19	0.6	0.58	0.65	0.88	0.71	0.81	0.91	0.6	5.68
	MMD-LS	0.64	0.79	0.81	0.77	0.59	0.56	0.56	0.97	0.68	0.86	0.88	0.58	9.09
Subspace	JPCA	0.88	0.85	0.66	0.19	0.59	0.56	0.56	0.83	0.7	0.8	0.9	0.55	9.13
	SA	0.74	0.81	0.8	0.13	0.6	0.58	0.56	0.93	0.7	0.91	0.89	0.59	7.26
	TCA	0.44	0.47	0.53	0.5	0.04	NA	0.11	0.56	0.6	0.43	NA	0.27	20.29
	TSL	0.88	0.85	0.66	0.86	0.59	0.21	0.3	0.7	0.7	0.61	0.88	0.26	15.12
Other	JDOT	0.72	0.57	0.82	0.14	0.61	0.51	0.64	0.77	0.68	0.63	0.8	0.46	13.43
	OTLabelProp	0.9	0.76	0.81	0.14	0.61	0.56	0.64	0.89	0.67	0.69	0.86	0.51	11.14
	DASVM	0.89	0.86	0.65	0.14	NA	NA	NA	0.68	NA	0.78	0.88	NA	12.39

D.4 Comparisons between supervised and unsupervised scorers

Impact on the cross-validation score. We observe in Figure D.1 the cross-validation score as a function of the final accuracy for various DA methods type and for both supervised and unsupervised scorers. As expected, we observe a good correlation between accuracy and cross-validation score with the supervised scorer. An important remark is that the Circular Validation (CircV) [4] shows some correlation between accuracy and cross-validation score. It indicates that this unsupervised scorer might be the most suitable choice for hyperparameter selection. This is supported by our extended experimental results in Table 2 for which the CircV is selected as the best scorer the most often. A similar trend can be observed for the Importance Weighted (IW) [49] which is also confirmed in Table 2.

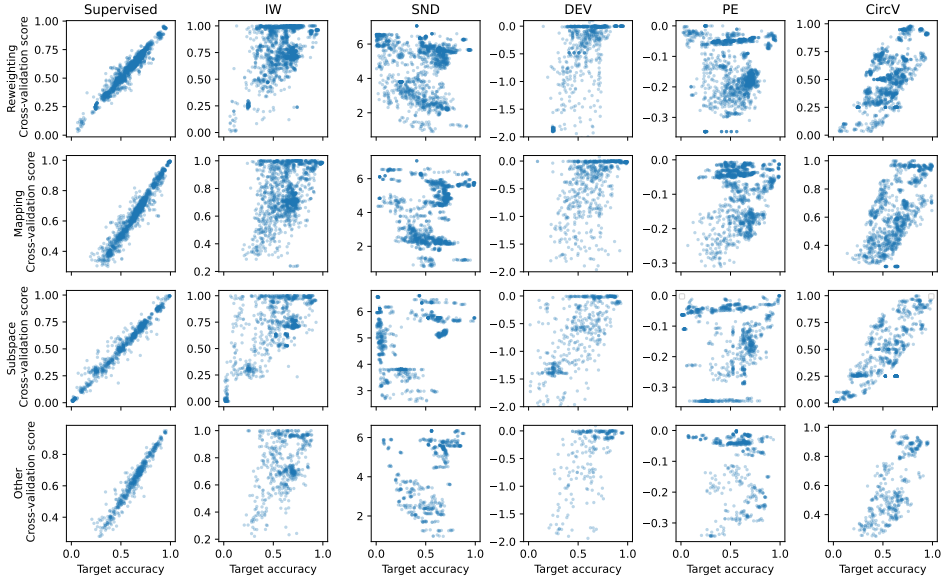


Figure D.1: Cross-val score as a function of the accuracy for various DA methods and different supervised and unsupervised scorers. Each point represents an inner split with a DA method (color of the points) and a dataset. A good scorer should have a score that correlates with the target accuracy.

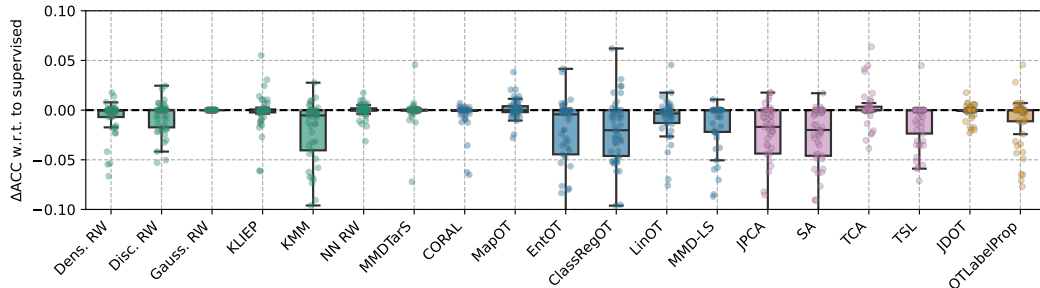


Figure D.2: Change of accuracy of the DA methods with the best realistic unsupervised scorer (Table 2) w.r.t. the supervised scorer.

Supervised scorer v.s. the best realistic unsupervised scorer. We plot the loss in performance of the DA methods with the best realistic unsupervised scorer compared to using the supervised scorers in Figure D.2.

Supervised scorer v.s. realistic unsupervised scorers. We present a scatter plot in Figure D.3 and Figure D.4, the accuracy of different DA methods using both supervised scorer and unsupervised scorer. In this figure, points below the diagonal indicate a decrease in performance when using the unsupervised scorer compared to the supervised one. The colors represent different types of DA methods. We can see that the SND, DEV and PE scorers all lead to a large performance loss compared to the supervised scorer. While IW and CircV results are much more concentrated near the diagonal, indicating a small loss in performance. This concentration explains why these two scorers have been selected as the best scorers for most of the methods in Table 2.

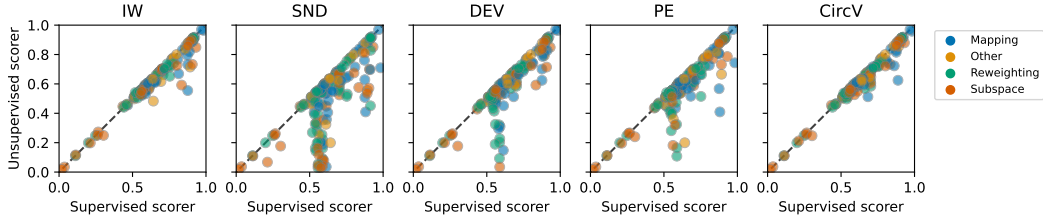


Figure D.3: Accuracy of the DA methods using unsupervised scorers as a function of the accuracy with the supervised scorer. Colors represent the type of DA methods.

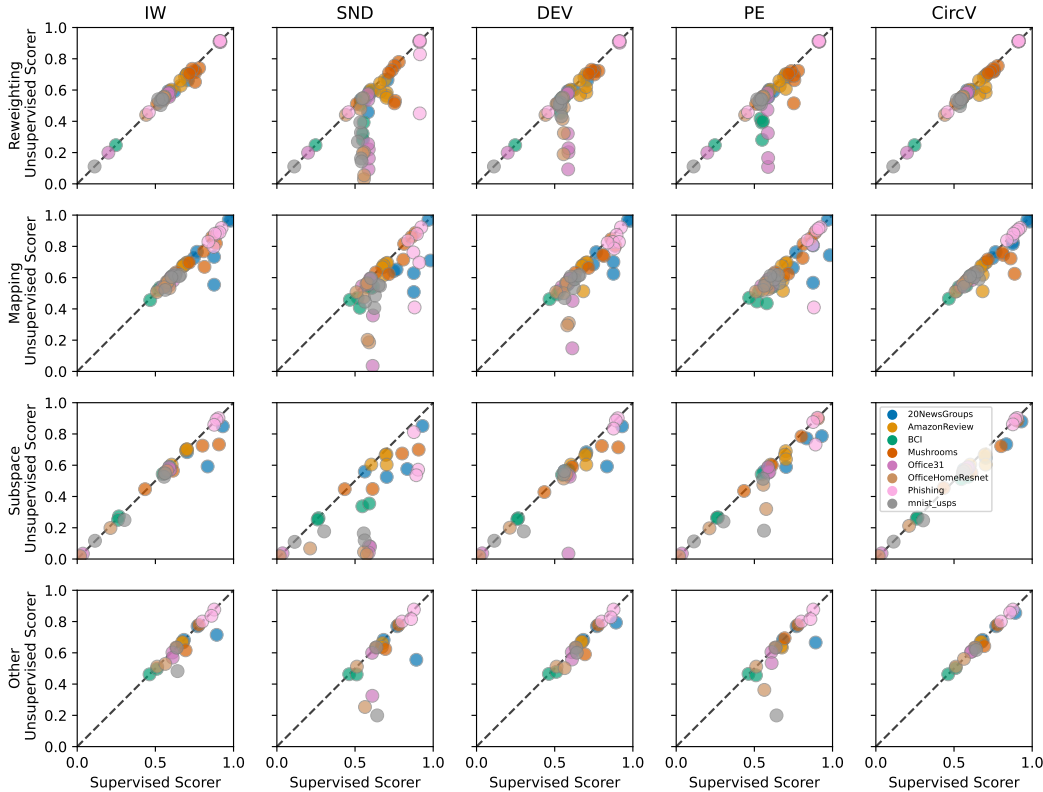


Figure D.4: Accuracy of the DA methods using unsupervised scorers as a function of the accuracy with the supervised scorer for the different types of DA methods. Points below the diagonal represent a decrease in performance when using the unsupervised scorer compared to the supervised one. Colors represent the dataset on which the DA method is applied.

D.5 Computational efficiency of the DA methods

Figure D.5 shows the average computation time for training and testing each method. These results are based on one outer split, while we ran the benchmarks for five outer splits. Each method has a different time complexity. Interestingly, more time-consuming methods are not necessarily more performant than others. For instance, the highest-ranked methods—LinOT, CORAL, and SA—also have some of the lowest training and testing times. It’s also worth noting that during the experiments, we enforced a 4-hour timeout. Thus, the more time-intensive methods might have been even slower without this timeout.

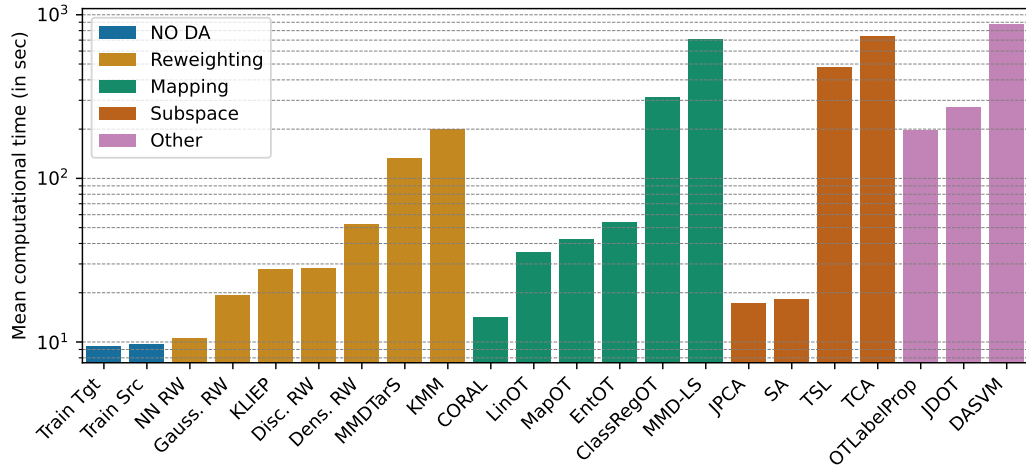


Figure D.5: Mean computing time to train and test each method for every experiment outer split.

D.6 Hyperparameters grid search for the DA methods

In this section, we report the grids of hyperparameters used in our grid search for each DA method. These values are also available in our code that is provided in the supplementary material.

Table D.13: Hyperparameter grids used in the grid search for each DA method. The hyperparameter grids were designed to be minimal yet expressive, allowing each method to perform optimally. We selected parameters based on what seemed most reasonable, according to our best knowledge.

Method	Hyperparameter Grid
KLIEP	'cv': [5], 'gamma': [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0, 'auto', 'scale'], 'max_iter': [1000], 'n_centers': [100], 'random_state': [0], 'tol': [1e-06]
KMM	'B': [1000.0], 'gamma': [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0, None], 'max_iter': [1000], 'smooth_weights': [False], 'tol': [1e-06]
NN RW	'laplace_smoothing': [True, False]
MapOT	'max_iter': [1000000], 'metric': ['sqeuclidean', 'cosine', 'cityblock'], 'norm': ['median']
JPCA	'n_components': [1, 2, 5, 10, 20, 50, 100]
SA	'n_components': [1, 2, 5, 10, 20, 50, 100]
TCA	'kernel': ['rbf'], 'mu': [10, 100], 'n_components': [1, 2, 5, 10, 20, 50, 100]
CORAL	'assume_centered': [False, True], 'reg': ['auto']
MMDTarS	'gamma': [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0, None], 'max_iter': [1000], 'reg': [1e-06], 'tol': [1e-06]
ClassRegOT	'max_inner_iter': [1000], 'max_iter': [10], 'metric': ['sqeuclidean', 'cosine', 'cityblock'], 'norm': ['l1'], 'tol': [1e-06], '(reg_cl, reg_e)': [(0.1, 0.1), (0.5, 0.5), (1.0, 1.0)]
Dens. RW	'bandwidth': [0.01, 0.1, 1.0, 10.0, 100.0, 'scott', 'silverman']
Disc. RW	'domain_classifier': ['LR', 'SVC', 'XGB']
Gauss. RW	'reg': ['auto']
DASVM	'max_iter': [200]
JDOT	'alpha': [0.1, 0.3, 0.5, 0.7, 0.9], 'n_iter_max': [100], 'thr_weights': [1e-07], 'tol': [1e-06]
EntOT	'max_iter': [1000], 'metric': ['sqeuclidean', 'cosine', 'cityblock'], 'norm': ['median'], 'reg_e': [0.1, 0.5, 1.0], 'tol': [1e-06]
LinOT	'bias': [True, False], 'reg': [1e-08, 1e-06, 0.1, 1, 10]
TSL	'base_method': ['flda'], 'length_scale': [2], 'max_iter': [300], 'mu': [0.1, 1, 10], 'n_components': [1, 2, 5, 10, 20, 50, 100], 'reg': [0.0001], 'tol': [0.0001]
MMD-LS	'gamma': [0.01, 0.1, 1, 10, 100], 'max_iter': [20], 'reg_k': [1e-08], 'reg_m': [1e-08], 'tol': [1e-05]
OTLabelProp	'metric': ['sqeuclidean', 'cosine', 'cityblock'], '(n_iter_max, reg)': [(10000, [None]), (100, [0.1, 1])]