ÖAW AUSTRIAN ACADEMY OF SCIENCES

RICAM – JOHANN RADON INSTITUTE FOR COMPUTATIONAL AND APPLIED MATHEMATICS

Ensemble Learning for Domain Adaptation by Importance Weighted Least Squares

M.-C. Dinu, M. Holzleitner, M. Beck,
D.H. Nguyen, A. Huber, H. Eghbal-zadeh,
B.A. Moser, S.V. Pereverzyev,
S. Hochreiter, W. Zellinger

RICAM-Report 2022-10

Ensemble Learning for Domain Adaptation by Importance Weighted Least Squares

Marius-Constantin Dinu[1,2] **Markus Holzleitner**[1] **Maximilian Beck**[1]

Duc Hoan Nguyen[4] **Andrea Huber**[1] **Hamid Eghbal-zadeh**[1]

Bernhard A. Moser[5] **Sergei V. Pereverzyev**[4] **Sepp Hochreiter**[1,3] **Werner Zellinger**[4]

[1]ELLIS Unit Linz and LIT AI Lab, Institute for Machine Learning,
Johannes Kepler University Linz
[2]Dynatrace Research
[3] Institute of Advanced Research in Artificial Intelligence
[4]Johann Radon Institute for Computational and Applied Mathematics,
Austrian Academy of Sciences
[5]Software Competence Center Hagenberg GmbH

Abstract

We study ensemble learning for unsupervised domain adaptation, i.e., with labeled data in a source domain and unlabeled data in a target domain, drawn from a different input distribution. An open problem is to find an optimal aggregation of given models without making strong assumptions on the model classes. While several heuristics exist, methods are still missing that rely on thorough theories for bounding the target error. In this turn, we propose a method that extends the theory of weighted least squares to linear aggregations and vector-valued functions. Our method is asymptotically error-rate-optimal, in the sense that the error of the computed aggregation is asymptotically not worse than twice the error of the unknown optimal aggregation. In experiments, we compare our method to (1) classical ensemble learning on source data only, (2) majority voting on target predictions, (3) ensemble learning based on pseudo-labels, (4) importance weighted validation, and, (5) deep embedded validation; on several datasets including language, images and time-series. As a result, our method sets a new state-of-the-art performance for ensemble learning in unsupervised domain adaptation under theoretical error guarantees.

1 Introduction

The goal of *unsupervised domain adaptation* is to learn a model on unlabeled data from a *target* input distribution using labeled data from a different *source* distribution [1]. If this goal is achieved, medical diagnostic systems can successfully be trained on unlabeled images using labeled images with different modality [2, 3]; natural language models can be learned from unlabeled biomedical abstracts by means of labeled data from financial journals [4]; models for industrial quality inspection of new, unlabelled, products can be constructed when utilizing data from related products [5, 6].

However, missing target labels combined with distribution shift makes algorithm design a hard problem [7, 8, 9, 10]. Typically, one often ends up with a sequence of models, e.g. originating from

Preprint. Under review.

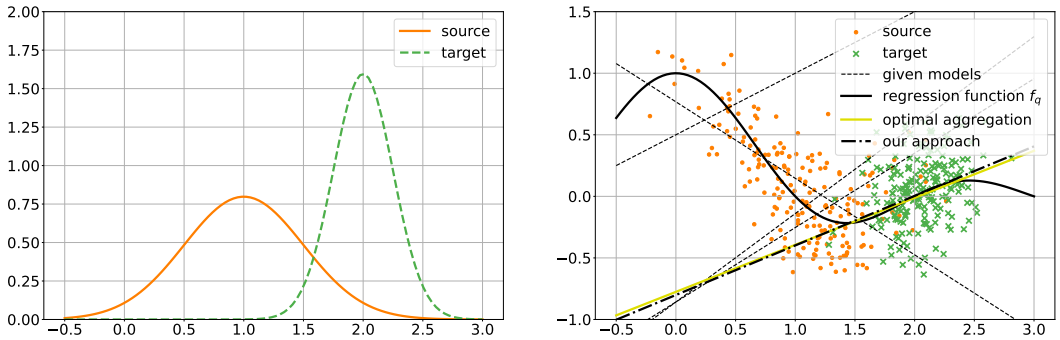

Figure 1: Unsupervised domain adaptation problem [40, 8, 9]. **Left:** Source distribution (solid) and target distribution (dashed). **Right:** A sequence of different linear models (dashed) is used to find the optimal linear aggregation of the models (solid). Model selection methods [8, 16, 9, 10] cannot outperform the best single model in the sequence, confidence values as used in [26] are not available, and, approaches based on averages or tendencies of majorities of models [27] suffer from a high fraction of large-error-models in the sequence. In contrast, our approach (dotted-dashed) is nearly optimal. In addition, the model computed by our method provably approaches the optimal linear aggregation for increasing sample size, at optimal error rate.

different domain adaptation algorithms or hyper-parameter configurations [11, 12, 13, 14, 15]. In this work, we study the problem of constructing an optimal aggregation using all models in the sequence.

Although methods with general performance guarantees have been proposed to select the best model in the sequence [8, 16, 9, 10], methods for learning aggregations of the models are either heuristics or their theory guarantees are limited by severe assumptions (cf. [17]). Typical approaches are (a) to learn ensembles on source data only [18], (b) to learn an ensemble on a set of (unknown) labeled target examples [19, 20, 21, 22], (c) to learn an aggregation on target examples (pseudo-)labeled based on confidence measures of the given models [23, 24, 25, 26, 27], (d) to aggregate the models based on data-structure specific transformations [28, 29], and, (e) to use specific (possibly not available) knowledge about the given models, such as information obtained at different time-steps of its gradient-based optimization process [30, 31, 32, 33, 34] or the information that the given models are trained on different (source) distributions [35, 36, 37, 38, 39]. One problem shared among all methods mentioned above is that they cannot guarantee a small error, even if the sample size grows to infinity. See Figure 1 for a simple illustrative example.

In this work, we propose (to the best of our knowledge) the first ensemble learning algorithm for unsupervised domain adaptation of vector-valued models with target error guarantees. We extend the *importance weighted least squares algorithm* [40] and corresponding recently proposed error bounds [41] to linear aggregations of vector-valued models. The importance weights are the values of an estimated ratio between target and source density evaluated at the examples. Every method for density-ratio estimation can be used as a basis for our approach, e.g. [42, 43, 41, 42] and references therein. Our error bound proves the optimality of the error rate of the proposed method, in the sense that the target error of the computed aggregation is asymptotically at most twice the target error of the optimal aggregation. In addition, we perform extensive empirical evaluations on several datasets with academic data (Transformed Moons [10]), text data (Amazon Reviews [4]), images (MiniDomainNet [10]), electroencephalography signals (Sleep-EDF [44, 45]), body sensor signals (UCI-HAR [46], WISDM [47]), and, sensor signals from mobile phones and smart watches (HHAR [48]). Our method[1] outperforms or is on par with classical heuristic ensemble approaches, including regression on source data only, ensemble learning on pseudo-labels on all datasets, and, sets a new state of the art for methods with theoretical error guarantees, namely importance weighted validation [8] and deep embedded validation [16].

[1]Source code is available at `https://anonymous.4open.science/r/iwa-4C76`

2 Summary of Results

Notation and Setup Let $\mathcal{X} \subset \mathbb{R}^{d_1}$ denote an *input space* and $\mathcal{Y} \subset \mathbb{R}^{d_2}$ denote a *label space* with inner product $\langle ., . \rangle_{\mathcal{Y}}$ such that for the associated norm $\|y\|_{\mathcal{Y}} \leq y_0$ holds for all $y \in \mathcal{Y}$ and some $y_0 > 0$. Following [49], we consider two datasets: A *source dataset* $(\mathbf{x}, \mathbf{y}) = ((x_1, y_1), \ldots, (x_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$ independently drawn according to some source distribution (Borel probability measure) p on $\mathcal{X} \times \mathcal{Y}$ and an unlabeled *target* dataset $\mathbf{x}' = (x'_1, \ldots, x'_m) \in \mathcal{X}^m$ with elements independently drawn according to the marginal distribution[2] $q_{\mathcal{X}}$ of some target distribution q on $\mathcal{X} \times \mathcal{Y}$. The marginal distribution of p on $\mathcal{X}$ is analogously denoted as $p_{\mathcal{X}}$. We further denote by $\mathcal{R}_q(f) = \int_{\mathcal{X} \times \mathcal{Y}} \|f(x) - y\|_{\mathcal{Y}}^2 \, \mathrm{d}q(x, y)$ the *expected target risk* of a vector valued function $f : \mathcal{X} \to \mathcal{Y}$ w.r.t. the least squares loss. It is interesting to highlight the generality of our work that is not restricted on a specific norm, e.g. the euclidean norm. Even more, $\mathcal{Y}$ can be any separable Hilbert space. We refer to Section 1 in the Supplementary Material for technical details. All vectors are column-vectors.

Problem Given a set $f_1, \ldots, f_l : \mathcal{X} \to \mathcal{Y}$ of models, the labeled source sample $(\mathbf{x}, \mathbf{y})$ and the unlabeled target sample $\mathbf{x}'$, the problem considered in this work is to find a model $f : \mathcal{X} \to \mathcal{Y}$ with a minimal target error $\mathcal{R}_q(f)$.

Main Assumptions We rely (a) on the *covariate shift* assumption that the source conditional distribution $p(y|x)$ equals the target conditional distribution $q(y|x)$, and, (b) on the *bounded density ratio* assumption that there is a function $\beta : \mathcal{X} \to [0, B]$ with $B > 0$ such that $\mathrm{d}q_{\mathcal{X}}(x) = \beta(x) \, \mathrm{d}p_{\mathcal{X}}(x)$.

Approach Our goal is to compute the linear aggregation $f = \sum_{i=1}^{l} c_i f_i$ for $c_1, \ldots, c_l \in \mathbb{R}$ with minimal squared target risk $\mathcal{R}_q\left(\sum_{i=1}^{l} c_i f_i\right)$. Our approach relies on the fact that $\arg\min_{c_1, \ldots, c_l \in \mathbb{R}} \mathcal{R}_q\left(\sum_{i=1}^{l} c_i f_i\right) = \arg\min_{c_1, \ldots, c_l \in \mathbb{R}} \int_{\mathcal{X}} \left\| \sum_{i=1}^{l} c_i f_i(x) - f_q(x) \right\|_{\mathcal{Y}}^2 \mathrm{d}q_{\mathcal{X}}(x)$ for the *regression function* $f_q(x) = \int_{\mathcal{Y}} y \, \mathrm{d}q(y|x) = \int_{\mathcal{Y}} y \, \mathrm{d}p(y|x) = f_p(x)$[3], see e.g. [51, Proposition 1]. Borrowing an idea from *importance sampling* and applying our main assumptions (bounded density ratio and covariate shift), we observe that

$$\operatorname*{arg\,min}_{c_1, \ldots, c_l \in \mathbb{R}} \mathcal{R}_q\left(\sum_{i=1}^{l} c_i f_i\right) = \operatorname*{arg\,min}_{c_1, \ldots, c_l \in \mathbb{R}} \int_{\mathcal{X}} \beta(x) \left\| \sum_{i=1}^{l} c_i f_i(x) - f_p(x) \right\|_{\mathcal{Y}}^2 \mathrm{d}p_{\mathcal{X}}(x) \tag{1}$$

which extends importance weighted least squares [40, 52] to linear aggregations $\sum_{i=1}^{l} c_i f_i$ of vector-valued functions $f_1, \ldots, f_l$. The unique minimizer of Eq. (1) can be approximated based on available data analogously to classical least squares estimation as detailed in Algorithm 1. In the following, we call Algorithm 1 Importance Weighted Least Squares Linear Aggregation (IWA).

Properties of Algorithm 1 IWA satisfies the following properties

- IWA is general in the sense that it does not rely on a specific model class, e.g. support vector machines, decision trees and neural networks, can be used simultaneously.
- IWA is the first (to the best of our knowledge) ensemble learning algorithm for unsupervised domain adaptation with a non-trivial target error bound for vector-valued models.
- IWA has asymptotically optimal error rate, in the sense that the computed ensemble has error at most twice the error of the optimal ensemble plus a term converging linearly to zero for increasing sample size. Our proof extends error bounds for regularized least squares in [53], to the importance weighted case [40], using arguments from [41, 54].

In addition, IWA sets a new state of the art for methods with theoretical error guarantees, namely importance weighted validation [8] and deep embedded validation [16], and, outperforms or is on par with classical heuristic ensemble approaches, including regression on source data only, and ensemble learning on pseudo-labels on all datasets.

[2] The existence of the conditional probability density $q(y|x)$ with $q(x, y) = q(y|x) q_{\mathcal{X}}(x)$ is guaranteed by the fact that $\mathcal{X} \times \mathcal{Y}$ is Polish, i.e., a separable and complete metric space, c.f. [50, Theorem 10.2.2.].

[3] $\mathcal{Y}$-valued integrals are defined in the sense of Lebesgue-Bochner.

Algorithm 1: Importance Weighted Least Squares Linear Aggregation (IWA)

Input : Set $f_1, \ldots, f_l : \mathcal{X} \to \mathcal{Y}$ of models, labeled source sample $(\mathbf{x}, \mathbf{y})$ and unlabeled target sample $\mathbf{x}'$.

Output : Linear aggregation $\widetilde{f} = \sum_{i=1}^{l} \widetilde{c}_i f_i$ with weights $\widetilde{c} = (\widetilde{c}_1, \ldots, \widetilde{c}_l) \in \mathbb{R}^l$.

Step 1 *Use unlabeled samples* $\mathbf{x}$ *and* $\mathbf{x}'$ *to approximate density ratio* $\frac{\mathrm{d}q_{\mathcal{X}}}{\mathrm{d}p_{\mathcal{X}}}$ *by some function* $\beta : \mathcal{X} \to [0, \infty)$ *using a classical algorithm, e.g. [42].*

Step 2 *Compute weight vector*

$$\widetilde{c} = \widetilde{G}^{-1} \widetilde{g}$$

with empirical Gram matrix $\widetilde{G}$ *and vector* $\widetilde{g}$ *defined by*

$$\widetilde{G} = \left(\frac{1}{m} \sum_{k=1}^{m} \langle f_i(x'_k), f_j(x'_k) \rangle_{\mathcal{Y}} \right)_{i,j=1}^{l} \qquad \widetilde{g} = \left(\frac{1}{n} \sum_{k=1}^{n} \beta(x_k) \langle y_k, f_i(x_k) \rangle_{\mathcal{Y}} \right)_{i=1}^{l} .$$

Return : Linear aggregation $\widetilde{f} = \sum_{i=1}^{l} \widetilde{c}_i f_i$.

Related Work It is well known, that aggregations of models in an ensemble often outperform individual models [55, 56]. Traditional ensemble methods that have shown the advantage of aggregation are Boosting [57, 58], Bootstrap Aggregating (bagging) [59, 60] and Stacking [61, 62]. For example, averages of multiple models pre-trained on data from a distribution different from the target one have recently been shown to achieve state-of-the-art performance on ImageNet [63] and their good generalization properties can be related to flat minima [64, 65]. However, most such methods don't take into account a present distribution shift. Although some ensemble learning methods exist, which take into account a present distribution shift, in contrast to our work, they are either relying on labeled target data [18, 19, 21, 20, 66], are restricted by fixing the aggregation weights to be the same [67], make assumptions on the models in the sequence or the corresponding process for learning the models [28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39], or, learn an aggregation based on the heuristic approach of (pseudo-)labeling some target data based on confidence measures of models in the sequence [23, 24, 25, 26, 27]. Another crucial difference of all methods above is that none of these methods can guarantee a small target error in the general setting (distribution shift, vector valued models, different classes, single source domain) described above, even if the sample size grows to infinity. Another branch of research are methods which aim at selecting the best model in the sequence. Although, such methods with error bounds have been proposed for the general setting above [8, 9, 10], they cannot overcome a limited performance of the best model in the given sequence (cf. Figure 1 and Section 6 in the Supplementary Material of [10]). In contrast, our method can outperform the best model in the sequence, and our empirical evaluations show that this is indeed the case in practical examples. An algorithm most similar to ours can be found in [41], where a method with theoretical error guarantees for aggregating regression models in unsupervised domain adaptation is proposed. However, in contrast to their approach, our method allows a much more general form of vector-valued models and can therefore be applied to practical classification tasks. Another recent work discussing least squares regression under domain shift in reproducing kernel spaces is [68]. However, the authors consider the choice of the regularization parameter under the assumption of known rate of decays of eigenvalues in the expansion of the kernel into eigenfunctions normalized with respect to (unknown) target measure (cf. Eq. 7 in [68]), and don't discuss the situation, where such information is not available. One can overcome this drawback by applying the ensemble learning approach proposed here. Our work employs technical tools developed in [53, 69]. In fact, we extend them to deal with importance weighted least squares. Finally, it is important to note [70], where a core Lemma of our proofs is proposed.

3 Target Error Bound for Algorithm 1

Let us start by introducing some further notation: $L^2(p)$ refers to the Lebesgue-Bochner space of functions from $\mathcal{X}$ to $\mathcal{Y}$, associated to a measure p on $\mathcal{X}$ with corresponding inner product $\langle ., . \rangle_{L^2(p)}$

(this space basically consists of all $\mathcal{Y}$-valued functions whose $\mathcal{Y}$-norms are square integrable with respect to the given measure p). Moreover, let us introduce the (positive semi-definite) Gram matrix $G = \left(\langle f_i, f_j \rangle_{L^2(q_\mathcal{X})}\right)_{i,j=1}^{l}$ and the vector $\overline{g} = \left(\langle \beta f_p, f_i \rangle_{L^2(p_\mathcal{X})}\right)_{i=1}^{l}$. We can assume that G is indeed invertible (and thus positive definite), since otherwise some models are too similar to others and can be withdrawn from consideration (see Supplementary Material Section 4). Next, we recall that the minimizer of Eq. (1) is $c^* = (c_1^*, \ldots, c_l^*) = G^{-1}\overline{g}$ (a proof of this observation is provided in Lemma 4 in the Supplementary Material).

However, neither G nor the vector $\overline{g}$ is accessible in practice, because there is no access to the target measure $q_\mathcal{X}$. Driven by the law of large numbers we try to approximate them by averages over our given data and therefore arrive at the formulas for $\widetilde{G}$ and $\widetilde{g}$ given in Algorithm 1. This leads to the approximation $\widetilde{f}$. Up to this point, we were only considering an intuitive perspective on the problem setting, therefore, we will now formally discuss statements on the distance between the model $\widetilde{f}$ and the optimal linear model $f^* = \sum_{i=1}^{l} c_i^* f_i$, measured in terms of target risks, and how this distance behaves with increasing sample sizes. This is what we attempt with our main result:

Theorem 1. *With probability* $1 - \delta$ *it holds that*

$$\mathcal{R}_q(\tilde{f}) - \mathcal{R}_q(f_q) \leq 2\left(\mathcal{R}_q\left(f^*\right) - \mathcal{R}_q(f_q)\right) + C\left(\log\frac{1}{\delta}\right)\left(n^{-1} + m^{-1}\right) \tag{2}$$

for some coefficient $C > 0$ *not depending on* m, n *and* δ.

Before we give an outline of the proof of Theorem 1 (full proof in Section 1 in the Supplementary Material), let us briefly comment on its main statement. First, we observe that $\mathcal{R}_q(f) - \mathcal{R}_q(f_q) = \|f - f_q\|^2_{L^2(q_\mathcal{X})}$ (cf. again e.g. [51, Proposition 1]) can be interpreted as the total target error made by Algorithm 1, sometimes called *excess risk*. Indeed, in the deterministic setting of labeling functions, f_q equals the target labeling function and the excess risk equals the target error of [49]. Eq. (2) compares this error for the aggregation $\widetilde{f}$, computed by Algorithm 1, to the error for the optimal aggregation f^*. Note that the error of the optimal aggregation f^* is unavoidable in the sense that it is determined by the decision of searching for linear aggregations of $f_1, \ldots, f_l$ only. However, if the models $f_1, \ldots, f_l$ are sufficiently different, then this error can be expected to be small. Theorem 1 tells us that the error of $\widetilde{f}$ approaches the one of f^* with increasing target and source sample size. Since the error functional is convex, $\widetilde{f}$ approaches the optimal linear aggregation f^* for increasing sample size. The rate of convergence is at least linear.

Let us now give a brief outline for the proof of Theorem 1. One key part concerns the existence of a Hilbert space $\mathcal{H}$ with associated inner product $\langle ., . \rangle_\mathcal{H}$ (a reproducing kernel space of functions from $\mathcal{X} \to \mathcal{Y}$) which contains all given models $f_1, \ldots, f_l$ and the regression function $f_q = f_p$. Note that Algorithm 1 does not need any knowledge of $\mathcal{H}$; its existence is not a restriction and used only for the proofs, so that we can apply many arguments developed in [53, 69]. Technical details explaining the role of $\mathcal{H}$ are deferred to Sections 1 and 2 in the Supplementary Material. Moreover, in this setting one can express the excess risk as follows: $\mathcal{R}_q\left(f\right) - \mathcal{R}_q(f_q) = \|A(f - f_q)\|^2_\mathcal{H}$ for some bounded linear operator $A : \mathcal{H} \to \mathcal{H}$. This also allows us to formulate the entries of G and $\bar{g}$ in terms of the inner product $\langle ., . \rangle_\mathcal{H}$ instead. Using properties related to the operators that appear in the construction of $\mathcal{H}$, in combination with Hoeffding-like concentration bounds in Hilbert spaces and bounds that measure e.g. the deviation between empirical averages in source and target domain (as done in [70, Lemma 4]), we can now quantify differences between the entries of G and $\widetilde{G}$ (and $\bar{g}$ and $\widetilde{g}$ respectively) in terms of n and m and an error probability δ. This ultimately leads to Eq. (2).

Let us finally place our theoretical findings in the existing literature and also discuss on the novelty: Overall, a similar strategy was already used in [41], however, only for univariate regression. Our main contribution is therefore the non-trivial extension to models with multidimensional output. Another advantage of our approach is, that we need no explicit knowledge on the operations used for the construction of the given models, which is also a core difference to [41]. Our construction even allows infinitely dimensional $\mathcal{Y}$, which turns out to be more involved. In this way, to the best of our knowledge, Theorem 1 extends the state of the art.

4 Empirical Evaluations

We now empirically evaluate the performance of our approach compared to classical ensemble learning baselines and state-of-the-art model selection methods. Therefore, we structure our empirical evaluation as follows. First, we outline our experimental setup for unsupervised domain adaptation and introduce all domain adaption methods for our analysis. Second, we describe the ensemble learning and model selection baselines, and third, we present the datasets used for our experiments. We then conclude with our results and a detailed discussion thereof.

4.1 Experimental Setup

To assess the performance of our ensemble learning Algorithm 1 IWA we perform numerous experiments with different domain adaptation algorithms on different datasets. By changing the hyper-parameters of each algorithm, we obtain, as results of applying these algorithms, sequences of models. The goal of our method is to find optimal models based on combinations of candidates from each sequence. For example, consider Figure 2, where we compare the results of Algorithm 1 to the approach proposed in [9]. The given sequence of models is obtained from applying the algorithm proposed in [11], with different choices for the domain adaptation hyper-parameter λ, to the Transformed Moons dataset. As domain adaptation algorithms, we use Central Moment Discrepancy (CMD) [14], Maximum Mean Discrepancy (MMD) [71], and Domain-Adversarial Neural Networks (DANN) [11] for our experiments on language and image datasets, and we consider the AdaTime benchmark suite for time-series data [72]. This suite comprises a collection of 10 domain adaptation algorithms and four datasets. For evaluation we follow their setup and use the introduced algorithms Deep Domain Confusion (DDC) [73], Correlation Alignment via Deep Neural Networks (Deep-Coral) [74], Higher-order Moment Matching (HoMM) [75], Minimum Discrepancy Estimation for Deep Domain Adaptation (MMDA) [76], Deep Subdomain Adaptation (DSAN) [77], Domain-Adversarial Neural Networks (DANN) [11], Conditional Adversarial Domain Adaptation (CDAN) [78], A DIRT-T Approach to Unsupervised Domain Adaptation (DIRT-T) [79], Convolutional deep Domain Adaptation model for Time-Series data (CoDATS) [80], and Adversarial Spectral Kernel Matching (AdvSKM) [81]. In addition to the sequence of models, our method, IWA, requires an estimate of the density ratio between source and target domain. To compute this quantity we follow [82] and [9, Section 4.3], and, train a classifier discriminating between source and target data. The output of this classifier is then used to approximate the density ratio denoted as β in Algorithm 1. Overall, to compute the results in our tables we trained 11981 models over approximately a timeframe of 1000 GPU/hours using computation resources of NVIDIA P100 16GB GPUs.

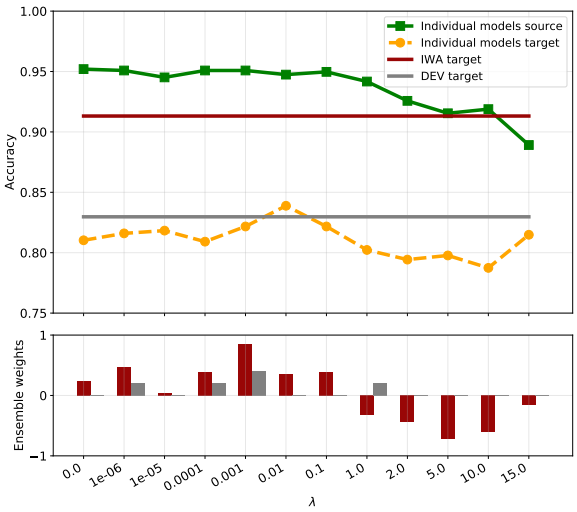

Figure 2: **Top:** Mean classification accuracy of our method (IWA), deep embedded validation (DEV) and individual models (green:source acc, orange: target acc) used in the ensemble for Transformed Moons over 5 seeds. The single models are trained with DANN [11] for different λ values. **Bottom:** Ensemble weights for individual models computed by IWA and DEV. Instead of selecting single best models in the ensemble, IWA effectively uses all models in the ensemble, thus outperforms DEV and all other model selection procedures.

4.2 Ensemble Learning Baselines

As representatives for the most prominent methods discussed in Section 1, we compare our method, IWA, to ensemble learning methods that use linear regression and majority voting as *heuristic* for model aggregation, and, model selection methods with *theoretical error guarantees*.

Heuristic Baselines Our first heuristic ensemble learning baseline is majority voting on target data (TMV). It aggregates the predictions of all models by counting the overall class predictions and selecting the class with the maximum prediction count as ensemble output. In addition, we implement three heuristic baselines which aggregate the vector-valued output, i.e. probabilities, of all classifiers using weights learned via linear regression. The final ensemble prediction is then made by selecting the class with the highest probability. The three heuristic regression baselines differ in the input used for the performed regression. Source-only regression (SOR) trains a regression model on classifier predictions (of the given models) and labels from the source domain only. Target majority voting regression (TMR) uses the same voting procedure as explained above to generate pseudo-labels on the target domain, which are then further used to train a linear regression model. In contrast, target confidence average regression (TCR) selects the highest average class probability over all classifiers to pseudo-label the target samples, which is then used for training the linear regression model.

Baselines with Theoretical Error Guarantees We compare IWA to the model selection methods importance weighted validation (IWV) [8] and deep embedded validation (DEV) [16], which select models according to their (importance weighted) target risk. Both methods assume the knowledge of an estimated density ratio between target and source domains. In our experiments we follow [82, 9] and estimate this ratio, by using a classifier that discriminates between source and target domain (see Supplementary Material Section 3 for more details). For deep embedded validation (DEV) the variance of the target risk estimate is further reduced by the technique of control variate [9].

4.3 Datasets

We evaluate the previously mentioned methods according to a diverse set of datasets, including language, image and time-series data. All datasets have a train, evaluation and test split, with results only presented on the held-out test sets. For additional details on the model architectures and datasets, we refer the reader to the Supplementary Material Section 3.

Academic Dataset We use the proposed Transformed Moons dataset from [10]. The source domain of this dataset consists of two-dimensional input data points and their transformations to two opposing moon-shaped forms. In Figure 2 we illustrate the ensemble weights and reference the reader for further ablations to the Supplementary Material Section 3.

Language Dataset To evaluate our method on a language task, we rely on the widely used Amazon Reviews [4] dataset. This dataset consists of text reviews from four domains: books (B), DVDs (D), electronics (E), and kitchen appliances (K). Reviews are encoded in feature vectors of bag-of-words unigrams and bigrams with binary labels: label 0 if the product is ranked by 1 to 3 stars, and label 1 if the product is ranked by 4 or 5 stars. From the four categories we obtain twelve domain adaptation tasks where each category serves once as source domain and once as target domain.

Image Dataset Our third dataset is MiniDomainNet, which is based on the DomainNet-2019 dataset [15] consisting of six different image domains (Quickdraw, Real, Clipart, Sketch, Infograph, and Painting). We follow [10] and rely on the reduced version of DomainNet-2019, referred to as MiniDomainNet, which reduces the number of classes to the top-five largest representatives in the training set of each class across all six domains.

Time-Series Dataset We rely on the AdaTime benchmark suite [72] which is a large-scale evaluation of domain adaptation algorithms on time-series data. It evaluates 10 state-of-the-art methods on four representative datasets spanning 20 cross-domain real-world scenarios, i.e., human activity recognition and sleep stage classification. The four datasets used by the benchmark are UCI-HAR [46], WISDM[47], HHAR [48], and Sleep-EDF [44, 45].

4.4 Results

We separate the applied methods into two groups, namely *heuristic* and methods with *theoretical error guarantees*. All tables show accuracies of source-only (SO) and target-best (TB) models, where source-only denotes training without domain adaptation and target-best the best performing model obtained among all parameter settings. We highlight in bold the performance of the best performing

method with theoretical error guarantees, and in italic the best performing heuristic. The full tables and additional details can be found in the Supplementary Material Section 4.

In summary, on all datasets, our method outperforms on average other methods with theoretical guarantees (IWV, DEV), setting a new state of the art. Even more, on Transformed Moons, Sleep-EDF, HHAR, UCI-HAR and WISDM, our method in addition outperforms all heuristic approaches. On the rest of the datasets, our method is on par with heuristics.

Table 1: Mean and standard deviation (after $\pm$) of target classification error on Transformed Moons dataset over 5 seeds with different random initializations of model weights.

	Transformed Moons								
		Heuristic				**Theoretical error guarantees**			
Method	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
CMD	0.800(±0.018)	0.801(±0.003)	0.799(±0.007)	0.809(±0.003)	*0.845(±0.022)*	0.827(±0.024)	0.827(±0.024)	**0.874(±0.026)**	0.822(±0.022)
MMD	0.800(±0.018)	0.815(±0.010)	0.810(±0.011)	0.805(±0.010)	*0.872(±0.022)*	0.825(±0.025)	0.825(±0.025)	**0.896(±0.038)**	0.818(±0.026)
DANN	0.810(±0.030)	0.808(±0.010)	0.809(±0.009)	0.809(±0.005)	*0.871(±0.028)*	0.830(±0.035)	0.830(±0.035)	**0.913(±0.019)**	0.839(±0.045)
Avg.	0.803(±0.022)	0.808(±0.008)	0.806(±0.009)	0.808(±0.006)	*0.862(±0.024)*	0.827(±0.028)	0.827(±0.028)	**0.894(±0.027)**	0.826(±0.826)

Table 2: Mean and standard deviation (after $\pm$) of target classification error on Amazon Reviews dataset over 5 seeds with different random initializations of model weights.

	Amazon Reviews								
		Heuristic				**Theoretical error guarantees**			
Method	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
CMD	0.752(±0.048)	0.790(±0.042)	0.785(±0.043)	0.785(±0.043)	*0.801(±0.044)*	0.759(±0.058)	0.726(±0.104)	**0.789(±0.043)**	0.794(±0.039)
MMD	0.752(±0.048)	0.768(±0.049)	0.770(±0.047)	0.766(±0.047)	*0.782(±0.046)*	0.730(±0.080)	0.635(±0.115)	**0.762(±0.082)**	0.763(±0.046)
DANN	0.750(±0.048)	*0.771(±0.047)*	0.770(±0.046)	0.770(±0.047)	0.764(±0.055)	0.749(±0.060)	0.662(±0.126)	**0.768(±0.051)**	0.757(±0.045)
Avg.	0.751(±0.048)	0.776(±0.046)	0.775(±0.046)	0.773(±0.046)	*0.782(±0.048)*	0.746(±0.066)	0.674(±0.115)	**0.773(±0.058)**	0.771(±0.771)

Table 3: Mean and standard deviation (after $\pm$) of target classification error on MiniDomainNet dataset over 3 seeds with different random initializations of model weights.

	MiniDomainNet								
		Heuristic				**Theoretical error guarantees**			
Method	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
CMD	0.677(±0.202)	0.667(±0.171)	0.676(±0.168)	0.686(±0.168)	*0.687(±0.169)*	0.676(±0.168)	0.476(±0.266)	**0.688(±0.168)**	0.677(±0.202)
MMD	0.677(±0.202)	0.666(±0.200)	0.668(±0.202)	0.665(±0.201)	*0.674(±0.194)*	0.662(±0.198)	0.655(±0.200)	**0.672(±0.194)**	0.677(±0.202)
DANN	0.671(±0.211)	0.681(±0.196)	0.682(±0.192)	*0.682(±0.192)*	0.681(±0.196)	0.659(±0.199)	0.652(±0.210)	**0.679(±0.199)**	0.671(±0.211)
Avg.	0.675(±0.205)	0.671(±0.189)	0.675(±0.187)	0.678(±0.187)	*0.681(±0.186)*	0.666(±0.188)	0.595(±0.226)	**0.680(±0.187)**	0.675(±0.675)

4.5 Discussion

In general, our method outperforms others, however, occasionally it is outperformed by heuristic choices. The question appears why this is the case. We observe that in some of these scenarios learning without domain adaptation (SO) is not significantly different from TB. This hints that domain adaptation may, either, have not been appropriately addressed by the underlying domain adaptation methods, or, that the dataset is too hard for the state-of-the-art methods. This phenomenon is known as *negative transfer* [1], and has been reported on in the domain adaptation literature (cf. Table 2 in [15] Source Only versus DAN, DANN; Table 3 AlexNet versus others; Table 4 Source Only versus DAN; and the Supplementary Material in [10]). Moreover, this phenomenon might serve as a an intuition why heuristics based on source data only (e.g. SOR) can work well in such scenarios: they profit from an implicit bias towards the best model (SO). However, in general, when ignoring such ill-posed problem settings, one can summarize, that our method is the most stable choice when approaching uncertain hyper-parameter-choices in unsupervised domain adaptation.

5 Conclusion and Future Work

In this paper, we present a constructive theory-based method for ensemble learning in the setting of unsupervised domain adaptation. Its theoretical approach relies on the extension of weighted least squares theory to vector-valued functions in reproducing kernel spaces. The resulting ensemble learning method distinguishes itself by a wide scope of admissible model classes without strong assumptions, e.g. support vector machines, decision trees and neural networks. A broad empirical evaluation on benchmark datasets for language, images and time-series underpins the theory-based

Table 4: Mean and standard deviation (after $\pm$) of target classification error on four time series datasets, each over 3 seeds with different random initializations of model weights.

	Sleep-EDF								
		Heuristic				Theoretical error guarantees			
Method	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
HoMM	0.659(±0.079)	0.701(±0.100)	0.696(±0.101)	0.696(±0.099)	*0.705(±0.096)*	0.713(±0.073)	0.703(±0.087)	**0.714(±0.092)**	0.700(±0.101)
AdvSKM	0.699(±0.056)	0.718(±0.079)	0.713(±0.077)	0.710(±0.078)	*0.719(±0.084)*	0.711(±0.065)	0.704(±0.061)	**0.727(±0.071)**	0.701(±0.079)
DIRT	0.637(±0.142)	0.750(±0.158)	0.753(±0.165)	0.754(±0.158)	*0.764(±0.117)*	0.734(±0.160)	0.699(±0.155)	**0.772(±0.114)**	0.747(±0.187)
DDC	0.662(±0.086)	*0.711(±0.064)*	0.703(±0.065)	0.698(±0.067)	0.695(±0.099)	0.695(±0.087)	0.669(±0.097)	**0.712(±0.069)**	0.702(±0.076)
MMDA	0.668(±0.093)	*0.708(±0.118)*	0.697(±0.126)	0.690(±0.129)	0.702(±0.142)	**0.711(±0.099)**	0.697(±0.104)	0.710(±0.113)	0.685(±0.141)
CoDATS	0.692(±0.114)	*0.729(±0.125)*	0.722(±0.129)	0.727(±0.126)	0.716(±0.094)	0.700(±0.126)	0.682(±0.196)	**0.739(±0.093)**	0.710(±0.096)
Deep-Coral	0.656(±0.086)	*0.705(±0.067)*	0.703(±0.061)	0.696(±0.067)	0.704(±0.093)	0.695(±0.087)	0.666(±0.094)	**0.706(±0.068)**	0.705(±0.072)
CDAN	0.639(±0.112)	0.722(±0.167)	0.723(±0.167)	0.726(±0.162)	*0.729(±0.104)*	0.690(±0.135)	0.662(±0.158)	**0.743(±0.127)**	0.724(±0.161)
DANN	0.646(±0.106)	0.695(±0.094)	*0.699(±0.092)*	0.694(±0.096)	0.689(±0.137)	**0.697(±0.084)**	0.666(±0.141)	0.691(±0.101)	0.680(±0.100)
DSAN	0.665(±0.095)	0.670(±0.179)	0.655(±0.192)	0.640(±0.204)	*0.720(±0.125)*	**0.713(±0.114)**	0.625(±0.170)	0.672(±0.187)	0.665(±0.095)
Avg.	0.662(±0.097)	0.711(±0.115)	0.706(±0.117)	0.703(±0.119)	*0.715(±0.109)*	0.706(±0.103)	0.677(±0.126)	**0.719(±0.103)**	0.702(±0.702)

	HHAR								
		Heuristic				Theoretical error guarantees			
Method	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
HoMM	0.719(±0.134)	*0.759(±0.169)*	0.758(±0.167)	0.758(±0.168)	0.726(±0.146)	0.739(±0.146)	0.722(±0.131)	**0.760(±0.168)**	0.782(±0.152)
AdvSKM	0.754(±0.119)	0.756(±0.136)	*0.761(±0.137)*	0.760(±0.136)	0.717(±0.126)	0.748(±0.124)	0.728(±0.168)	**0.761(±0.135)**	0.754(±0.119)
DIRT	0.744(±0.116)	0.803(±0.161)	0.801(±0.160)	*0.808(±0.155)*	0.763(±0.182)	0.731(±0.141)	0.754(±0.157)	**0.788(±0.177)**	0.832(±0.166)
DDC	0.694(±0.166)	*0.752(±0.167)*	0.752(±0.168)	0.751(±0.166)	0.726(±0.150)	0.734(±0.167)	0.731(±0.161)	**0.755(±0.167)**	0.759(±0.136)
MMDA	0.686(±0.135)	*0.783(±0.163)*	0.782(±0.160)	0.780(±0.160)	0.702(±0.139)	0.704(±0.132)	0.703(±0.131)	**0.780(±0.164)**	0.777(±0.139)
CoDATS	0.718(±0.142)	0.772(±0.243)	0.772(±0.244)	0.772(±0.246)	*0.790(±0.208)*	0.745(±0.203)	0.752(±0.202)	**0.772(±0.243)**	0.777(±0.238)
Deep-Coral	0.726(±0.140)	*0.775(±0.162)*	0.769(±0.170)	0.767(±0.175)	0.712(±0.168)	0.750(±0.138)	0.750(±0.131)	**0.774(±0.164)**	0.769(±0.159)
CDAN	0.733(±0.141)	0.763(±0.225)	0.759(±0.224)	0.759(±0.226)	*0.763(±0.179)*	0.726(±0.186)	0.713(±0.200)	**0.795(±0.191)**	0.803(±0.209)
DANN	0.750(±0.096)	0.772(±0.236)	0.779(±0.232)	*0.780(±0.232)*	0.741(±0.168)	0.750(±0.142)	0.745(±0.149)	**0.797(±0.208)**	0.783(±0.150)
DSAN	0.734(±0.158)	0.786(±0.227)	0.780(±0.235)	0.782(±0.234)	*0.810(±0.185)*	0.787(±0.191)	0.563(±0.313)	**0.830(±0.191)**	0.812(±0.192)
Avg.	0.726(±0.135)	*0.772(±0.189)*	0.771(±0.190)	0.772(±0.190)	0.745(±0.165)	0.741(±0.157)	0.716(±0.174)	**0.781(±0.181)**	0.785(±0.785)

	UCI-HAR								
		Heuristic				Theoretical error guarantees			
Method	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
HoMM	0.797(±0.182)	*0.838(±0.131)*	0.822(±0.153)	0.822(±0.148)	0.733(±0.205)	0.815(±0.154)	0.815(±0.154)	**0.840(±0.126)**	0.843(±0.136)
AdvSKM	0.786(±0.182)	*0.786(±0.193)*	0.782(±0.200)	0.786(±0.196)	0.741(±0.181)	0.761(±0.193)	0.769(±0.178)	**0.794(±0.186)**	0.788(±0.205)
DIRT	0.779(±0.183)	*0.892(±0.093)*	0.886(±0.103)	0.888(±0.103)	0.762(±0.215)	0.815(±0.157)	0.815(±0.157)	**0.889(±0.100)**	0.922(±0.070)
DDC	0.779(±0.193)	0.793(±0.189)	0.796(±0.192)	*0.797(±0.191)*	0.742(±0.157)	0.746(±0.206)	0.748(±0.209)	**0.797(±0.184)**	0.784(±0.211)
MMDA	0.804(±0.176)	0.797(±0.190)	0.794(±0.197)	*0.803(±0.191)*	0.772(±0.162)	0.799(±0.143)	0.801(±0.140)	**0.804(±0.190)**	0.819(±0.151)
CoDATS	0.773(±0.147)	*0.849(±0.142)*	0.831(±0.163)	0.825(±0.164)	0.793(±0.149)	0.787(±0.155)	0.815(±0.110)	**0.840(±0.150)**	0.867(±0.123)
Deep-Coral	0.748(±0.210)	0.790(±0.192)	*0.801(±0.181)*	0.794(±0.191)	0.782(±0.170)	0.751(±0.211)	0.751(±0.211)	**0.794(±0.192)**	0.808(±0.152)
CDAN	0.747(±0.203)	*0.838(±0.159)*	0.836(±0.163)	0.838(±0.167)	0.803(±0.121)	0.745(±0.141)	0.734(±0.178)	**0.850(±0.149)**	0.846(±0.153)
DANN	0.744(±0.217)	*0.835(±0.164)*	0.825(±0.170)	0.828(±0.164)	0.805(±0.173)	0.758(±0.226)	0.790(±0.186)	**0.840(±0.156)**	0.825(±0.155)
DSAN	0.785(±0.165)	*0.864(±0.132)*	0.839(±0.160)	0.843(±0.156)	0.773(±0.166)	0.812(±0.128)	0.726(±0.272)	**0.869(±0.125)**	0.906(±0.120)
Avg.	0.774(±0.186)	*0.828(±0.158)*	0.821(±0.168)	0.822(±0.167)	0.771(±0.170)	0.779(±0.172)	0.776(±0.179)	**0.832(±0.156)**	0.841(±0.841)

	WISDM								
		Heuristic				Theoretical error guarantees			
Method	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
HoMM	0.744(±0.115)	*0.765(±0.080)*	0.755(±0.089)	0.739(±0.071)	0.707(±0.103)	0.749(±0.110)	**0.771(±0.097)**	0.762(±0.076)	0.772(±0.062)
AdvSKM	0.743(±0.104)	*0.765(±0.090)*	0.739(±0.120)	0.759(±0.110)	0.736(±0.110)	0.742(±0.104)	0.742(±0.104)	**0.772(±0.085)**	0.770(±0.107)
DIRT	0.780(±0.106)	0.788(±0.064)	0.787(±0.064)	0.790(±0.060)	*0.802(±0.071)*	0.780(±0.106)	**0.786(±0.105)**	0.782(±0.065)	0.805(±0.061)
DDC	0.743(±0.130)	*0.764(±0.095)*	0.747(±0.115)	0.749(±0.105)	0.728(±0.114)	0.743(±0.130)	0.743(±0.130)	**0.767(±0.096)**	0.771(±0.090)
MMDA	0.749(±0.122)	*0.757(±0.067)*	0.740(±0.084)	0.743(±0.089)	0.750(±0.153)	0.749(±0.122)	0.760(±0.108)	**0.771(±0.079)**	0.749(±0.122)
CoDATS	0.679(±0.106)	*0.808(±0.094)*	0.793(±0.105)	0.798(±0.107)	0.721(±0.122)	0.703(±0.104)	0.710(±0.093)	**0.823(±0.106)**	0.805(±0.093)
Deep-Coral	0.693(±0.092)	*0.751(±0.080)*	0.740(±0.091)	0.726(±0.099)	0.707(±0.113)	0.698(±0.090)	0.695(±0.089)	**0.746(±0.083)**	0.755(±0.100)
CDAN	0.728(±0.115)	0.780(±0.069)	*0.783(±0.069)*	0.769(±0.063)	0.759(±0.080)	0.746(±0.114)	0.760(±0.096)	**0.774(±0.061)**	0.774(±0.087)
DANN	0.746(±0.114)	*0.776(±0.120)*	0.773(±0.113)	0.770(±0.113)	0.732(±0.164)	0.746(±0.120)	0.716(±0.169)	**0.772(±0.128)**	0.754(±0.081)
DSAN	0.749(±0.122)	0.711(±0.151)	0.718(±0.165)	0.701(±0.187)	*0.754(±0.121)*	**0.749(±0.122)**	0.713(±0.150)	0.689(±0.191)	0.749(±0.122)
Avg.	0.735(±0.113)	*0.767(±0.091)*	0.758(±0.102)	0.754(±0.100)	0.740(±0.115)	0.740(±0.112)	0.739(±0.114)	**0.766(±0.097)**	0.770(±0.770)

optimality claim. It is left to future research to further refine the theory and its estimates e.g. by exploiting reproducing kernel spaces for neural networks, concentration bounds from [70] or advanced density ratio estimators from [42].

Acknowledgments

The ELLIS Unit Linz, the LIT AI Lab, and the Institute for Machine Learning are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (LIT-2018-6-YOU-212), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), AIRI FG 9-N (FWF-36284, FWF-36235), and ELISE (H2020-ICT-2019-3 ID: 951847). We further thank Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, Software Competence Center Hagenberg GmbH, TÜV Austria, Frauscher Sensonic, and the NVIDIA Corporation. The research reported

in this paper has been funded by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Federal Ministry for Digital and Economic Affairs (BMDW), and the Province of Upper Austria in the frame of the COMET–Competence Centers for Excellent Technologies Programme and the COMET Module S3AI managed by the Austrian Research Promotion Agency FFG.

References

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

[2] T. Varsavsky, M. Orbes-Arteaga, C. H. Sudre, M. S. Graham, P. Nachev, and M. J. Cardoso. Test-time unsupervised domain adaptation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 428–436. Springer, 2020.

[3] D. Zou, Q. Zhu, and P. Yan. Unsupervised domain adaptation with dual-scheme fusion network for medical image segmentation. In *IJCAI*, pages 3291–3298, 2020.

[4] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128, 2006.

[5] J. Jiao, M. Zhao, J. Lin, and C. Ding. Classifier inconsistency-based domain adaptation network for partial transfer intelligent diagnosis. *IEEE Transactions on Industrial Informatics*, 16(9):5965–5974, 2019.

[6] W. Zellinger, T. Grubinger, M. Zwick, E. Lughofer, H. Schöner, T. Natschläger, and S. Saminger-Platz. Multi-source transfer learning of time series in cyclical manufacturing. *Journal of Intelligent Manufacturing*, 31(3):777–787, 2020.

[7] F. D. Johansson, D. Sontag, and R. Ranganath. Support and invertibility in domain-invariant representations. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 527–536, 2019.

[8] M. Sugiyamai, M. Krauledat, and K. M. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

[9] K. You, X. Wang, M. Long, and M. Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 7124–7133. PMLR, 2019.

[10] W. Zellinger, N. Shepeleva, M.-C. Dinu, H. Eghbal-zadeh, H. Nguyen, B. Nessler, S. Pereverzyev, and B. A. Moser. The balancing principle for parameter choice in distance-regularized domain adaptation. *Advances in Neural Information Processing Systems*, 34, 2021.

[11] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(Jan):1–35, 2016.

[12] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 137–144, 2007.

[13] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the International Conference on Machine Learning*, pages 97–105, 2015.

[14] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *International Conference on Learning Representations*, 2017.

[15] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[16] W. M. Kouw, J. H. Krijthe, and M. Loog. Robust importance-weighted cross-validation under sample selection bias. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2019.

[17] G. Wilson and D. J. Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

[18] D. Nozza, E. Fersini, and E. Messina. Deep learning and ensemble methods for domain adaptation. In *IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 184–189. IEEE, 2016.

[19] R. Xia, C. Zong, X. Hu, and E. Cambria. Feature ensemble plus sample selection: domain adaptation for sentiment classification. *IEEE Intelligent Systems*, 28(3):10–18, 2013.

[20] W. Dai, Q. Yang, G. R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, page 193–200, 2007.

[21] H. Daume III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research*, 26:101–126, 2006.

[22] L. Duan, I. W. Tsang, and D. Xu. Domain transfer multiple kernel learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):465–479, 2012.

[23] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 30:8008–8018, 2021.

[24] W. Ahmed, P. Morerio, and V. Murino. Cleaning noisy labels by negative ensemble learning for source-free unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1616–1625, 2022.

[25] W. Tuand S. Sun. Dynamical ensemble learning with model-friendly classifiers for domain adaptation. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1181–1184. IEEE, 2012.

[26] Y. Zou, Z. Yu, B. V. K. Kumar, and J. Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018.

[27] K. Saito, Y. Ushiku, and T. Harada. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997. PMLR, 2017.

[28] J. B. Yang, Q. Mao, Q. L. Xiang, I. W.-H. Tsangand K. M. A. Chai, and H. L. Chieu. Domain adaptation for coreference resolution: An adaptive ensemble approach. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 744–753, 2012.

[29] J. M. Ha and B. D. Youn. A health data map-based ensemble of deep domain adaptation under inhomogeneous operating conditions for fault diagnosis of a planetary gearbox. *IEEE Access*, 9:79118–79127, 2021.

[30] G. French, M. Mackiewicz, and M. Fisher. Self-ensembling for visual domain adaptation. *International Conference on Learning Representations*, 2018.

[31] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *International Conference on Learning Representations (ICLR)*, 2017.

[32] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[33] B. Athiwaratkun, M. Finzi, P. Izmailov, and A. G. Wilson. There are many consistent explanations of unlabeled data: Why you should average. *International Conference on Learning Representations (2019)*, 2019.

[34] S. Al-Stouhi and C. K. Reddy. Adaptive boosting for transfer learning using dynamic updates. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 60–75. Springer, 2011.

[35] J. Hoffman, M. Mohri, and N. Zhang. Algorithms and theory for multiple-source adaptation. *Advances in Neural Information Processing Systems*, 31, 2018.

[36] S. Rakshit, B. Banerjee, G. Roig, and S. Chaudhuri. Unsupervised multi-source domain adaptation driven by deep adversarial ensemble learning. In *German Conference on Pattern Recognition*, pages 485–498. Springer, 2019.

[37] R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3964–3973, 2018.

[38] G. Kang, L. Jiang, Y. Wei, Y. Yang, and A. G. Hauptmann. Contrastive adaptation network for single-and multi-source domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[39] K. Zhang, M. Gong, and B. Schölkopf. Multi-source domain adaptation: A causal view. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[40] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[41] E. R. Gizewski, L. Mayer, B. A. Moser, D. H. Nguyen, S. Pereverzyev Jr, S. V. Pereverzyev, N. Shepeleva, and W. Zellinger. On a regularization of unsupervised domain adaptation in RKHS. *Applied and Computational Harmonic Analysis*, 57:201–227, 2022.

[42] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[43] T. Kanamori, T. Suzuki, and M. Sugiyama. Statistical analysis of kernel-based least-squares density-ratio estimation. *Machine Learning*, 86(3):335–367, 2012.

[44] E. Eldele, Z. Chen, C. Liu, M. Wu, C.-K. Kwoh, X. Li, and C. Guan. An attention-based deep learning approach for sleep stage classification with single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2021.

[45] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 101(23):215–220, 2000.

[46] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. *European Symposium on Artificial Neural Networks*, pages 437–442, 2013.

[47] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *Sigkdd Explorations*, 12(2):74–82, 2011.

[48] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, page 127–140, New York, NY, USA, 2015. Association for Computing Machinery.

[49] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[50] R. M. Dudley. *Real analysis and probability*, volume 74. Cambridge University Press, 2002.

[51] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.

[52] T. Kanamori, S. Hido, and M. Sugiyama. A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10:1391–1445, 2009.

[53] A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

[54] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19, 2006.

[55] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, 2020.

[56] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[57] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[58] L. Breiman. Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, 26(3):801–849, 1998.

[59] L. Breiman. Bagging predictors. Technical Report 421, Department of Statistics, UC Berkeley, 1994.

[60] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[61] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:214–259, 1992.

[62] L. Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996.

[63] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, and L. Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*, 2022.

[64] S. Hochreiter and J. Schmidhuber. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994.

[65] S. Hochreiter and J. Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[66] A. Mayr, G. Klambauer, T. Unterthiner, and S. Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

[67] H. Razar and S. Samothrakis. Bagging adversarial neural networks for domain adaptation in non-stationary eeg. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2019.

[68] C. Ma, R. Pathak, and M. J. Wainwright. Optimally tackling covariate shift in rkhs-based nonparametric regression. *arXiv preprint arXiv:2205.02986*, 2022.

[69] A. Caponnetto and E. De Vito. Risk bounds for regularized least-squares algorithm with operatorvalued kernels. Technical report, CBCL paper 249/CSAIL-TR-2005-031, MIT, 2005.

[70] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems*, pages 513–520, 2006.

[71] I. O. Tolstikhin, B. K. Sriperumbudur, and B. Schölkopf. Minimax estimation of maximum mean discrepancy with radial kernels. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[72] M. Ragab, E. Eldele, W. L. Tan, C.-S. Foo, Z. Chen, M. Wu, C.-K. Kwoh, and X. Li. Adatime: A benchmarking suite for domain adaptation on time series data. *arXiv preprint arXiv:2203.08321*, 2022.

[73] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[74] B. Sun, J. Feng, and K. Saenko. Correlation alignment for unsupervised domain adaptation. *Domain Adaptation in Computer Vision Applications*, pages 153–171, 2017.

[75] C. Chen, Z. Fu, Z. Chen, S. Jin, Z. Cheng, X. Jin, and X.-S. Hua. Homm: Higher-order moment matching for unsupervised domain adaptation. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

[76] M. M. Rahman, C. Fookes, M. Baktashmotlagh, and S. Sridharan. On minimum discrepancy estimation for deep domain adaptation. *Domain Adaptation for Visual Understanding*, 2020.

[77] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, and Q. He. Deep subdomain adaptation network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 32(4):1713–1722, 2021.

[78] M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[79] R. Shu, H. Bui, H. Narui, and S. Ermon. A dirt-t approach to unsupervised domain adaptation. *International Conference on Learning Representations (ICLR)*, 2018.

[80] G. Wilson, J. R. Doppa, and D. J. Cook. Multi-source deep domain adaptation with weak supervision for time-series sensor data. *Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD)*, 2020.

[81] Q. Liu and H. Xue. Adversarial spectral kernel matching for unsupervised time series domain adaptation. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 30, 2021.

[82] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88, 2007.

Supplementary material for: Ensemble Learning for Domain Adaptation by Importance Weighted Least Squares

Marius-Constantin Dinu[1,2] **Markus Holzleitner**[1] **Maximilian Beck**[1]

Duc Hoan Nguyen[4] **Andrea Huber**[1] **Hamid Eghbal-zadeh**[1]

Bernhard A. Moser[5] **Sergei V. Pereverzyev**[4] **Sepp Hochreiter**[1,3] **Werner Zellinger**[4]

[1]ELLIS Unit Linz and LIT AI Lab, Institute for Machine Learning,
Johannes Kepler University Linz
[2]Dynatrace Research
[3] Institute of Advanced Research in Artificial Intelligence
[4]Johann Radon Institute for Computational and Applied Mathematics,
Austrian Academy of Sciences
[5]Software Competence Center Hagenberg GmbH

1 Notation and Proof of Main Result

The aim of this section is to give a full proof of our main result, Theorem 1 in the main paper. We start by introducing and summarizing the notation and the required concepts from functional analysis and measure theory, so that we can state and prove the required lemmas.

Summary of Notation

- *Spaces:* input space $\mathcal{X} \subset \mathbb{R}^{d_1}$ and label space $\mathcal{Y}$ with inner product $\langle ., . \rangle_{\mathcal{Y}}$. $\mathcal{Y}$ is assumed to be a separable Hilbert space such that for the associated norm $\|y\|_{\mathcal{Y}} \leq y_0$ holds for all $y \in \mathcal{Y}$ and some $y_0 > 0$. Note that this setting is more general than the one from the main text, where we assumed $\mathcal{Y} \subset \mathbb{R}^{d_2}$ (the simplification in the main text improves readability and respectation of space limits).
- *Datasets and Distributions:* Source data set: $(\mathbf{x}, \mathbf{y}) = ((x_1, y_1), \ldots, (x_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$ independently drawn according to source distribution p on $\mathcal{X} \times \mathcal{Y}$ and an unlabeled *target* dataset $\mathbf{x}' = (x'_1, \ldots, x'_m) \in \mathcal{X}^m$ independently drawn according marginal distribution $q_{\mathcal{X}}$ of target distribution q on $\mathcal{X} \times \mathcal{Y}$ (the corresponding marginal distribution of p on $\mathcal{X}$ is similarly denoted as $p_{\mathcal{X}}$).
- *Source Risk:* $\mathcal{R}_p(f) = \int_{\mathcal{X}\times\mathcal{Y}} \|f(x) - y\|^2_{\mathcal{Y}} \, \mathrm{d}p(x, y)$.
- *Source Regression function* $f_p(x) = \int_{\mathcal{Y}} y \, \mathrm{d}p(y|x)$. (Vector valued) integral in the sense of Lebesgue-Bochner.
- *Target Risk:* $\mathcal{R}_q(f) = \int_{\mathcal{X}\times\mathcal{Y}} \|f(x) - y\|^2_{\mathcal{Y}} \, \mathrm{d}q(x, y)$
- *Target Regression function* $f_q(x) = \int_{\mathcal{Y}} y \, \mathrm{d}q(y|x)$. (Vector valued) integral in the sense of Lebesgue-Bochner.

Preprint. Under review.

Problem

- *Given:* sequence $f_1, \dots, f_l : \mathcal{X} \to \mathcal{Y}$ of models, source sample $(\mathbf{x}, \mathbf{y})$ and unlabeled target sample $\mathbf{x}'$
- *Aim:* find aggretation $f = \sum_{i=1}^{l} c_i f_i$ with minimal $\mathcal{R}_q(f)$.

Main Assumptions

- *covariate shift:* $p(y|x) = q(y|x)$ and thus $f_p = f_q$.
- *bounded density ratio:* there is $\beta : \mathcal{X} \to [0, B]$ such that $\mathrm{d}q_{\mathcal{X}}(x) = \beta(x)\,\mathrm{d}p_{\mathcal{X}}(x)$.

Existence of the associated conditional probability measures is guaranteed by the fact that $\mathcal{X} \times \mathcal{Y}$ is Polish (a separable and complete metric space), c.f. [1, Theorem 10.2.2.].

Notation from functional analysis/operator theory Let $\mathbf{U}$ and $\mathbf{V}$ denote separable Hilbert spaces (i.e. they admit countable orthonormal bases) with associated inner products $\langle .,. \rangle_{\mathbf{U}}$ (or $\langle .,. \rangle_{\mathbf{V}}$, respectively). Let us briefly recall some notions from functional analysis that we need in order to set up our theory. There are lots of standard references on these aspects, e.g. [2] and [3]:

- $\mathcal{L}(\mathbf{U}, \mathbf{V})$: space of bounded linear operators $\mathbf{U} \to \mathbf{V}$ with uniform norm $\|.\|_{\mathcal{L}(\mathbf{U},\mathbf{V})}$. $\mathcal{L}(\mathbf{U})$: space of bounded linear operators $\mathbf{U} \to \mathbf{U}$.
- For $A \in \mathcal{L}(\mathbf{U}, \mathbf{V})$, its *adjoint* is denoted by $A^* \in \mathcal{L}(\mathbf{V}, \mathbf{U})$ (and uniquely defined by the equation $\langle Au, v \rangle_{\mathbf{V}} = \langle u, A^* v \rangle_{\mathbf{U}}$ for any $u \in \mathbf{U}$, $v \in \mathbf{V}$).
- If $A \in \mathcal{L}(\mathbf{U})$ and $A = A^*$: A is called *self-adjoint*.
- If $A \in \mathcal{L}(\mathbf{U})$ is self adjoint and $\langle Au, u \rangle_{\mathbf{U}} \geq 0$ for any $u \in \mathbf{U}$, then A is called *positive*. Equivalently: there exists (unique) bounded and self-adjoint $B := \sqrt{A} \in \mathcal{L}(\mathbf{U})$ such that $B^2 = A$.
- *Trace* of an operator $A \in \mathcal{L}(\mathbf{U})$: $Tr(A) = \sum_k \langle Ae_k, e_k \rangle_{\mathbf{U}}$ for any orthonormal basis $(e_k)_{k=1}^{\infty}$ of $\mathbf{U}$ (independent of choice of basis). If $Tr(A) < \infty$: A is called *trace class*.
- $\mathcal{L}_2(\mathbf{U})$: separable Hilbert space of *Hilbert-Schmidt operators* on $\mathbf{U}$ with scalar product $\langle A, B \rangle_{\mathcal{L}_2(\mathbf{U})} = Tr(B^* A)$ and norm $\|A\|_{\mathcal{L}_2(\mathbf{U})} = \sqrt{Tr(A^* A)} \geq \|A\|_{\mathcal{L}(\mathbf{U})}$.
- $A : \mathbf{U} \to \mathbf{V}$ is called *Hilbert-Schmidt*, if $A^* A$ is trace class. Also here: $\|A\|_{\mathcal{L}(\mathbf{U},\mathbf{V})} \leq \sqrt{Tr(A^* A)}$
- For (probability) measure q on $\mathcal{X}$ (or $\mathcal{Y}$) and appropriate functions $F : \mathcal{X} \to \mathbf{U}$ (e.g. strongly measurable and $\|F\|_{\mathbf{U}}$ is integrable wrt. q) we denote the usual ($\mathbf{U}$-valued) *Bochner integral* of F as $\int_{\mathcal{X}} F(x)\,\mathrm{d}q(x)$. We denote the associated L^p-spaces by $L^p(\mathcal{X}, q, \mathbf{U})$, or $L^p(q)$ for short, if the associated spaces are clear from the context.

Assumptions on models We assume that the regression function $f^* = f_p = f_q$ as well as the models $f_1, ..., f_l$ belong to a *hypothesis space* $\mathcal{H} \subseteq C(\mathcal{X}, \mathcal{Y}) \subseteq L^2(p_{\mathcal{X}}) \cap L^2(q_{\mathcal{X}})$, where $C(\mathcal{X}, \mathcal{Y})$ denotes the space of bounded continous functions $\mathcal{X} \to \mathcal{Y}$. The space $\mathcal{H}$ should satisfy the following assumptions, which are discussed in much greater detail in [4] and [5]:

Hypothesis 1. *[4] The space $\mathcal{H}$ is a separable Hilbert space of functions $f : \mathcal{X} \to \mathcal{Y}$ such that:*

- *For all $x \in \mathcal{X}$ there is a Hilbert-Schmidt operator $K_x : \mathcal{Y} \to \mathcal{H}$ satisfying*
$$f(x) = K_x^* f, \quad f \in \mathcal{H}, \tag{1}$$
- *The function from $\mathcal{X} \times \mathcal{X}$ to $\mathbb{R}$*
$$(x, t) \mapsto \langle K_t v, K_x w \rangle_{\mathcal{H}} \text{ is measurable } \forall v, w \in \mathcal{Y}; \tag{2}$$
- *There is $\kappa > 0$ such that*
$$Tr(K_x^* K_x) \leq \kappa, \quad \forall x \in \mathcal{X}. \tag{3}$$

Moreover we assume that the norms $\|f_k\|_{\mathcal{H}}$, $k = 1, 2, \dots, l$, are under our control, such that we can put a threshold $\gamma_l > 0$ and consider $\|f_k\|_{\mathcal{H}} \leq \gamma_l$.

Further useful observations Then we have

$$K_t^* K_x = K(t,x) \in \mathcal{L}_2(\mathcal{Y}) \quad \forall x,t \in \mathcal{X}. \tag{4}$$

Given $x \in \mathcal{X}$ the operator

$$T_x = K_x K_x^* \in \mathcal{L}_2(\mathcal{H}), \tag{5}$$

is a positive Hilbert-Schmidt operator and (5) ensures

$$\|T_x\|_{\mathcal{L}(\mathcal{H})} \le \|T_x\|_{\mathcal{L}_2(\mathcal{H})} = \|K(x,x)\|_{\mathcal{L}_2(\mathcal{Y})} \le \kappa. \tag{6}$$

Let $T_{q_\mathcal{X}} : \mathcal{H} \to \mathcal{H}$ be

$$T_{q_\mathcal{X}} = \int_\mathcal{X} T_x \, \mathrm{d}q_\mathcal{X}(x),$$

where the integral converges in $\mathcal{L}_2(\mathcal{H})$ to a positive trace class operator with

$$\|T_{q_\mathcal{X}}\|_{\mathcal{L}(\mathcal{H})} \le \|T_{q_\mathcal{X}}\|_{\mathcal{L}_2(\mathcal{H})} \le Tr(T_{q_\mathcal{X}}) = \int_\mathcal{X} Tr(T_x) \, \mathrm{d}q_\mathcal{X}(x) \le \kappa. \tag{7}$$

Following Proposition 1 in [4], we have the minimizers f_q of expected risk $\mathcal{R}_q$ are the solution of the following equation:

$$T_{q_\mathcal{X}} f_q = g,$$

where

$$g = \int_\mathcal{X} K_x f_q(x) \, \mathrm{d}q_\mathcal{X}(x) \in \mathcal{H},$$

with integral converging in $\mathcal{H}$.

Next we define the operators

$$T_{\mathbf{x}'} = \frac{1}{m} \sum_{j=1}^m K_{x'_j} K_{x'_j}^*,$$

$$T_{\mathbf{x},\beta} = \frac{1}{n} \sum_{i=1}^n \beta(x_i) K_{x_i} K_{x_i}^*,$$

$$g_{\mathbf{x},\mathbf{y},\beta} = \frac{1}{n} \sum_{i=1}^n \beta(x_i) K_{x_i} y_i.$$

In the sequel we adopt the convention that C denotes a generic positive coefficient, which can vary from appearance to appearance and may only depend on basic parameter such as $p_\mathcal{X}$, $q_\mathcal{X}$, κ, B, y_0 and others introduced below, but not on n, m and error probability $\delta > 0$.

We will need the following statements.

Lemma 1. *With probability at least $1-\delta$ we have*

$$\|T_{q_\mathcal{X}} - T_{\mathbf{x}'}\|_{\mathcal{L}(\mathcal{H})} \le \|T_{q_\mathcal{X}} - T_{\mathbf{x}'}\|_{\mathcal{L}_2(\mathcal{H})} \le C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) m^{-\frac{1}{2}}, \tag{8}$$

$$\|T_{\mathbf{x}'} - T_{\mathbf{x},\beta}\|_{\mathcal{L}(\mathcal{H})} \le C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) \left(n^{-\frac{1}{2}} + m^{-\frac{1}{2}} \right), \tag{9}$$

$$\|T_{\mathbf{x},\beta} f^* - g_{\mathbf{x},\mathbf{y},\beta}\|_\mathcal{H} \le C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) n^{-\frac{1}{2}}, \tag{10}$$

where $C > 0$ does not depend on n, m and δ.

The proof of Lemma 1 is based on Lemma 4 of [6], which we formulate in our notations as follows

Lemma 2. *([6]) Let ϕ be a map from $\mathbf{U}$ to $\mathbf{U}$ such that $\|\phi(x)\|_\mathbf{U} \le R$ for all $x \in \mathcal{X}$. Then with probability at least $1-\delta$ it holds*

$$\left\| \frac{1}{m} \sum_{j=1}^m \phi(x'_j) - \frac{1}{n} \sum_{i=1}^n \beta(x_i)\phi(x_i) \right\|_\mathbf{U} \le \left(1 + \sqrt{2 \log \frac{2}{\delta}} \right) R \sqrt{\frac{B^2}{n} + \frac{1}{m}}.$$

Moreover, we will need a concentration inequality that follows from [7], see also [8].

Lemma 3 (Concentration lemma)**.** *If $\xi_1, \xi_2, \ldots, \xi_n$ are zero mean independent random variables with values in a separable Hilbert space $\mathbf{U}$, and for some $D > 0$ one has $\|\xi_i\|_{\mathbf{U}} \leq D$, $i = 1, 2, \ldots, n$, then the following bound*

$$\left\| \frac{1}{n} \sum_{i=1}^{n} \xi_i \right\|_{\mathbf{U}} \leq \frac{D\sqrt{2\log\frac{2}{\delta}}}{\sqrt{n}}$$

holds true with probability at least $1 - \delta$.

Proof of Lemma 1.

Let us start by proving (8) by introducing the map $\xi : \mathcal{X} \to \mathcal{L}_2(\mathcal{H})$ as $\xi(x) = K_x K_x^* - T_{q_\mathcal{X}}$. From (6) and (7) it follows that

$$\|\xi(x)\|_{\mathcal{L}_2(\mathcal{H})} \leq \|K_x K_x^*\|_{\mathcal{L}_2(\mathcal{H})} + \|T_{q_\mathcal{X}}\|_{\mathcal{L}_2(\mathcal{H})} \leq 2\kappa.$$

Moreover, we have

$$\int_\mathcal{X} \xi(x) dq_\mathcal{X}(x) = \int_\mathcal{X} K_x K_x^* dq_\mathcal{X}(x) - T_{q_\mathcal{X}} = 0.$$

Therefore, for $x'_j, j = 1, 2, \ldots, m$, drawn i.i.d from the marginal probability measure $q_\mathcal{X}$, the corresponding operators $\xi_j = \xi(x'_j)$ can be treated as zero mean independent random variables in $\mathcal{L}_2(\mathcal{H})$, such that the condition of Concentration lemma are satisfied with $D = 2\kappa$, and

$$\|T_{\mathbf{x}'} - T_{q_\mathcal{X}}\|_{\mathcal{L}_2(\mathcal{H})} = \left\| \frac{1}{m} \sum_{j=1}^{m} K_{x'_j} K_{x'_j}^* - T_{q_\mathcal{X}} \right\|_{\mathcal{L}_2(\mathcal{H})} = \left\| \frac{1}{m} \sum_{j=1}^{m} \xi_j \right\|_{\mathcal{L}_2(\mathcal{H})} \leq \frac{2\kappa\sqrt{2\log\frac{2}{\delta}}}{\sqrt{m}}.$$

To obtain (9), for any $f \in \mathcal{H}$ we define a map $\phi = \phi_f : \mathcal{X} \to \mathcal{H}$ as $\phi_f(x) = K_x K_x^* f$. It clear that

$$\|\phi_f(x)\|_\mathcal{H} = \|K_x K_x^*\|_{\mathcal{L}(\mathcal{H})} \|f\|_\mathcal{H} \leq \kappa \|f\|_\mathcal{H}.$$

Therefore, for the map $\phi = \phi_f$ the condition of the above Lemma 2 is satisfied with $R = \kappa \|f\|_\mathcal{H}$. Then directly from that lemma for any $f \in \mathcal{H}$ we have

$$\begin{aligned}
\|T_{\mathbf{x}'} f - T_{\mathbf{x},\beta} f\|_\mathcal{H} &= \left\| \frac{1}{m} \sum_{j=1}^{m} \phi_f(x'_j) - \frac{1}{n} \sum_{i=1}^{n} \beta(x_i)\phi_f(x_i) \right\|_\mathcal{H} \\
&\leq \left(1 + \sqrt{2\log\frac{2}{\delta}}\right) \left(\sqrt{\frac{B^2}{n} + \frac{1}{m}}\right) \kappa \|f\|_\mathcal{H} \\
&\leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta}\right) \left(m^{-\frac{1}{2}} + n^{-\frac{1}{2}}\right) \|f\|_\mathcal{H},
\end{aligned}$$

that proves (9).

Consider now the map $\digamma : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$ defined by

$$\digamma(x, y) = \beta(x) K_x (f_p(x) - y).$$

Recall that $\|K_x\|_{\mathcal{L}(\mathcal{Y},\mathcal{H})} \leq \sqrt{Tr(K_x^* K_x)} \leq \sqrt{\kappa}$. Then we obtain:

$$\|\digamma(x, y)\|_\mathcal{H} \leq \|K_x\|_{\mathcal{L}(\mathcal{Y},\mathcal{H})} \left\| \int_\mathcal{Y} y' dp(y'|x) - y \right\|_\mathcal{Y} |\beta(x)| \leq 2y_0 B\sqrt{\kappa}.$$

Moreover, for $p(x, y) = p(y|x)p_\mathcal{X}(x)$ we have

$$\int_{\mathcal{X}\times\mathcal{Y}} \digamma(x, y) dp(x, y) = \int_\mathcal{X} K_x \beta(x) \int_\mathcal{Y} \left(\int_\mathcal{Y} y' dp(y'|x) - y \right) dp(y|x) dp_\mathcal{X}(x) = 0,$$

such that for (x_i, y_i), $i = 1, 2, \ldots, n$, drawn i.i.d from the measure $p(x, y)$ the corresponding values $\digamma_i = \digamma(x_i, y_i)$ are zero mean independent random variables in $\mathcal{H}$.

Then for the just defined $F_i = \beta(x_i)K_{x_i}(f_q(x_i) - y_i)$ the conditions of Lemma 3 are satisfied with $D = 2y_0 B\sqrt{\kappa}$, such that

$$\begin{aligned}\left\|\frac{1}{n}\sum_{i=1}^{n} F_i\right\|_{\mathcal{H}} &= \left\|\frac{1}{n}\sum_{i=1}^{n}\beta(x_i)K_{x_i}(f_q(x_i) - y_i)\right\|_{\mathcal{H}} \\ &= \left\|\sum_{i=1}^{n}\beta(x_i)K_{x_i}K_{x_i}^* f_q - \sum_{i=1}^{n}\beta(x_i)K_{x_i}y_i\right\|_{\mathcal{H}} \\ &= \|T_{\mathbf{x},\beta}f_q - g_{\mathbf{x},\mathbf{y},\beta}\|_{\mathcal{H}} \leq \frac{2y_0 B\sqrt{\kappa}\sqrt{2\log\frac{2}{\delta}}}{\sqrt{n}}.\end{aligned}$$

This bound gives us (10).

Aggregation for vector-valued functions Next we construct a new approximant in the form of a linear combination of approximants $f_1, f_2, \ldots, f_l$, computed for all tried parameter values. The linear combination of the approximants is computed as

$$f = \sum_{k=1}^{l} c_k f_k. \tag{11}$$

Since $f_1, f_2, \ldots, f_l$ belong to RKHS $\mathcal{H}$, it is clear that $f \in \mathcal{H}$. Now we want to argue on how close we can get to f_q. Following Proposition 1 in [4], we have

$$\mathcal{R}_q(f) - \mathcal{R}_q(f_q) = \|f - f_q\|^2_{L^2(q_\mathcal{X})} = \left\|\sqrt{T_{q_\mathcal{X}}}(f - f_q)\right\|^2_{\mathcal{H}}. \tag{12}$$

Next we observe that the best approximation f^* of the target regression function f_q by linear combinations corresponds to the vector $c^* = (c_1^*, \ldots, c_l^*)$ of ideal coefficients in (11) that solves the linear system $Gc^* = \bar{g}$ with the Gram matrix $G = \left(\left\langle\sqrt{T_{q_\mathcal{X}}}f_k, \sqrt{T_{q_\mathcal{X}}}f_u\right\rangle_{\mathcal{H}}\right)_{k,u=1}^{l}$ and the right-hand side vector $\bar{g} = \left(\left\langle\sqrt{T_{q_\mathcal{X}}}f_q, \sqrt{T_{q_\mathcal{X}}}f_k\right\rangle_{\mathcal{H}}\right)_{k=1}^{l}$. Let us provide a prove of this short observation in the next lemma. Note that the entries G and g can equivalently also be formulated in terms of $\langle .,. \rangle_{L^2(q_\mathcal{X})}$, as done in the main text. We are going to use this formulation in the next lemma in order to be compatible with the main text (switching to the inner products in terms of $\mathcal{H}$ would not change the argument of the proof at all):

Lemma 4. *The best $L^2(q_\mathcal{X})$-approximation f^* of the target regression function f_q by linear combinations corresponds to the vector $c^* = (c_1^*, \ldots, c_l^*) = G^{-1}\bar{g}$.*

Proof. Let us denote (12) by $f(c)$ and rewrite this expression appropriately:

$$f(c) = \sum_{i,j=1}^{l} c_i c_j \langle f_i, f_j\rangle_{L^2(q_\mathcal{X})} - 2\sum_{i=1}^{l} c_i \langle f_i, f_q\rangle_{L^2(q_\mathcal{X})} + \langle f_q, f_q\rangle_{L^2(q_\mathcal{X})}.$$

Taking the derivative with respect to c_i yield:

$$\frac{\partial f(c)}{\partial c_i} = 2\left(\sum_{j=1}^{l} c_j \langle f_i, f_j\rangle_{L^2(q_\mathcal{X})} - \langle f_i, f_q\rangle_{L^2(q_\mathcal{X})}\right).$$

Setting these derivatives to zero (for all $i = 1, ..l$) gives the claimed equation. Noting that the Hessian is equal to $2G$ (and thus positive-definite) ensures that c^* is a global minimum of f. □

But, of course, neither Gram matrix G nor the vector $\bar{g}$ is accessible, because there is no access to the target measure $q_\mathcal{X}$, so we switch to the empirical counterparts $\tilde{G}$ and $\tilde{g}$.

Then the following lemma is helpful to gain some information on the error made by the empirical average:

Lemma 5. *With probability* $1-\delta$ *we have*

$$\left|\left\langle\sqrt{T_{q\mathcal{X}}}f_u,\sqrt{T_{q\mathcal{X}}}f_k\right\rangle_{\mathcal{H}}-\frac{1}{m}\sum_{j=1}^{m}\left\langle f_k(x'_j),f_u(x'_j)\right\rangle_{\mathcal{Y}}\right|\leq C\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)m^{-\frac{1}{2}},$$

$$\left|\left\langle\sqrt{T_{q\mathcal{X}}}f_k,\sqrt{T_{q\mathcal{X}}}f_q\right\rangle_{\mathcal{H}}-\frac{1}{n}\sum_{i=1}^{n}\beta(x_i)\left\langle f_k(x_i),y_i\right\rangle_{\mathcal{Y}}\right|\leq C\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)(n^{-\frac{1}{2}}+m^{-\frac{1}{2}}),$$

where $C>0$ *does not depend on* n,m *and* δ.

Proof. Keeping in mind that $f_q,f_k\in\mathcal{H}$ we have

$$\begin{aligned}\left\langle\sqrt{T_{q\mathcal{X}}}f_u,\sqrt{T_{q\mathcal{X}}}f_k\right\rangle_{\mathcal{H}}&=\langle T_{\mathbf{x}'}f_k,f_u\rangle_{\mathcal{H}}+\langle(T_{q\mathcal{X}}-T_{\mathbf{x}'})f_u,f_k\rangle_{\mathcal{H}}\\&=\left\langle\frac{1}{m}\sum_{j=1}^{m}K_{x'_j}K^*_{x'_j}f_k,f_u\right\rangle_{\mathcal{H}}+\langle(T_{q\mathcal{X}}-T_{\mathbf{x}'})f_u,f_k\rangle_{\mathcal{H}}\\&=\frac{1}{m}\sum_{j=1}^{m}\left\langle K^*_{x'_j}f_k,K^*_{x'_j}f_u\right\rangle_{\mathcal{Y}}+\langle(T_{q\mathcal{X}}-T_{\mathbf{x}'})f_u,f_k\rangle_{\mathcal{H}}\\&=\frac{1}{m}\sum_{j=1}^{m}\left\langle f_k(x'_j),f_u(x'_j)\right\rangle_{\mathcal{Y}}+\langle(T_{q\mathcal{X}}-T_{\mathbf{x}'})f_u,f_k\rangle_{\mathcal{H}}.\end{aligned}$$

Moreover, from (8) with probability $1-\delta$ we have that

$$\left|\langle(T_{q\mathcal{X}}-T_{\mathbf{x}'})f_u,f_k\rangle_{\mathcal{H}}\right|\leq C\left\|f_u\right\|_{\mathcal{H}}\left\|f_k\right\|_{\mathcal{H}}\left(log^{\frac{1}{2}}\frac{1}{\delta}\right)m^{-\frac{1}{2}}.$$

Then

$$\left|\left\langle\sqrt{T_{q\mathcal{X}}}f_u,\sqrt{T_{q\mathcal{X}}}f_k\right\rangle_{\mathcal{H}}-\frac{1}{m}\sum_{j=1}^{m}\left\langle f_k(x'_j),f_u(x'_j)\right\rangle_{\mathcal{Y}}\right|\leq C\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)m^{-\frac{1}{2}}.$$

Now, we prove the second statement in Lemma 5. We have

$$\begin{aligned}\left\langle\sqrt{T_{q\mathcal{X}}}f_k,\sqrt{T_{q\mathcal{X}}}f_q\right\rangle_{\mathcal{H}}&=\langle f_k,T_{q\mathcal{X}}f_q\rangle_{\mathcal{H}}=\langle f_k,T_{q\mathcal{X}}f_q-g_{\mathbf{x},\mathbf{y},\beta}\rangle_{\mathcal{H}}+\langle f_k,g_{\mathbf{x},\mathbf{y},\beta}\rangle_{\mathcal{H}}\\&=\frac{1}{n}\sum_{i=1}^{n}\beta(x_i)\left\langle f_k,K_{x_i}y_i\right\rangle_{\mathcal{H}}+\langle f_k,T_{q\mathcal{X}}f_q-g_{\mathbf{x},\mathbf{y},\beta}\rangle_{\mathcal{H}}\\&=\frac{1}{n}\sum_{i=1}^{n}\beta(x_i)\left\langle K^*_{x_i}f_k,y_i\right\rangle_{\mathcal{Y}}+\langle f_k,T_{q\mathcal{X}}f_q-g_{\mathbf{x},\mathbf{y},\beta}\rangle_{\mathcal{H}}\\&=\frac{1}{n}\sum_{i=1}^{n}\beta(x_i)\left\langle f_k(x_i),y_i\right\rangle_{\mathcal{Y}}+\langle f_k,T_{q\mathcal{X}}f_q-g_{\mathbf{x},\mathbf{y},\beta}\rangle_{\mathcal{H}}.\end{aligned}$$

From Lemma 1, with probability $1-\delta$ we have

$$\begin{aligned}&\|T_{q\mathcal{X}}f_q-g_{\mathbf{x},\mathbf{y},\beta}\|_{\mathcal{H}}\\&\leq\|T_{q\mathcal{X}}f_q-T_{\mathbf{x}'}f_q\|_{\mathcal{H}}+\|T_{\mathbf{x}'}f_q-g_{\mathbf{x},\mathbf{y},\beta}\|_{\mathcal{H}}\\&\leq\|T_{q\mathcal{X}}f_q-T_{\mathbf{x}'}f_q\|_{\mathcal{H}}+\|T_{\mathbf{x}'}f_q-T_{\mathbf{x},\beta}f_q\|_{\mathcal{H}}+\|T_{\mathbf{x},\beta}f_q-g_{\mathbf{x},\mathbf{y},\beta}\|_{\mathcal{H}}\\&\leq C\left\|f_q\right\|_{\mathcal{H}}\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)m^{-\frac{1}{2}}+C\left\|f_q\right\|_{\mathcal{H}}\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)(n^{-\frac{1}{2}}+m^{-\frac{1}{2}})+\|T_{\mathbf{x},\beta}f_q-g_{\mathbf{x},\mathbf{y},\beta}\|_{\mathcal{H}}\\&\leq C\left\|f_q\right\|_{\mathcal{H}}\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)m^{-\frac{1}{2}}+C\left\|f_q\right\|_{\mathcal{H}}\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)(n^{-\frac{1}{2}}+m^{-\frac{1}{2}})+C\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)n^{-\frac{1}{2}}.\end{aligned}$$

Then

$$\langle f_k,T_{q\mathcal{X}}f_q-g_{\mathbf{x},\mathbf{y},\beta}\rangle_{\mathcal{H}}\leq C\left\|f_k\right\|_{\mathcal{H}}\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)\left(n^{-\frac{1}{2}}+m^{-\frac{1}{2}}\right).$$

Therefore,

$$\left| \left\langle \sqrt{T_{q_{\mathcal{X}}}} f_k, \sqrt{T_{q_{\mathcal{X}}}} f_q \right\rangle_{\mathcal{H}} - \frac{1}{n} \sum_{i=1}^{n} \beta(x_i) \left\langle f_k(x_i), y_i \right\rangle_{\mathcal{Y}} \right| \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}).$$

□

Towards our main generalization bound Next we mimic the proof of Theorem 4 from [9] to obtain our main result, Theorem 1. Lemma 5 suggests to approximate G and $\bar{g}$ by their empirical counterparts:

$$\tilde{G} = \left(\frac{1}{m} \sum_{j=1}^{m} \left\langle f_k(x_j^{'}), f_u(x_j^{'}) \right\rangle_{\mathcal{Y}} \right)_{k,u=1}^{l}, \tag{13}$$

$$\tilde{g} = \left(\frac{1}{n} \sum_{i=1}^{n} \beta(x_i) \left\langle y_i, f_k(x_i) \right\rangle_{\mathcal{Y}} \right)_{k=1}^{l} \tag{14}$$

which can be effectively computed from data samples. Moreover, again from Lemma 5 we can argue that with probability $1 - \delta$ it holds:

$$\|\bar{g} - \tilde{g}\|_{\mathbb{R}^l} \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}), \tag{15}$$

$$\|G - \tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)} \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) m^{-\frac{1}{2}}. \tag{16}$$

With the matrix $\tilde{G}$ at hand one can easily check whether or not it is well-conditioned and $\tilde{G}^{-1}$ exists (otherwise one needs to get rid of models with similar performance). Thus the norms $\left\|\tilde{G}\right\|_{\mathcal{L}(\mathbb{R}^l)}$ and $\left\|\tilde{G}^{-1}\right\|_{\mathcal{L}(\mathbb{R}^l)}$ can be bounded independently of m and n, due to the fact that all their entries can be bounded as follows (we only do the calculation for the entries of $\tilde{G}$):

$$\begin{aligned}
|\tilde{G}_{k,u}| &\leq \frac{1}{m} \sum_{j=1}^{m} \left| \left\langle f_k(x_j^{'}), f_u(x_j^{'}) \right\rangle_{\mathcal{Y}} \right| = \frac{1}{m} \sum_{j=1}^{m} \left| \left\langle K_{x_j^{'}}^{*} f_k, K_{x_j^{'}}^{*} f_u \right\rangle_{\mathcal{Y}} \right| \\
&= \frac{1}{m} \sum_{j=1}^{m} \left| \left\langle K_{x_j^{'}} K_{x_j^{'}}^{*} f_k, f_u \right\rangle_{\mathcal{H}} \right| = \frac{1}{m} \sum_{j=1}^{m} \left| \left\langle T_{x_j^{'}} f_k, f_u \right\rangle_{\mathcal{H}} \right| \\
&\leq \frac{1}{m} \sum_{j=1}^{m} \left\| T_{x_j^{'}} \right\|_{\mathcal{L}(\mathcal{H})} \|f_k\|_{\mathcal{H}} \|f_u\|_{\mathcal{H}} \leq \kappa \gamma_l^2,
\end{aligned}$$

where we used the reproducing property (1) to obtain the equality in the first line and (6) for the last inequality. Now assume that m is so large that with probability $1 - \delta$ we have

$$\|G - \tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)} < \frac{1}{\left\|\tilde{G}^{-1}\right\|_{\mathcal{L}(\mathbb{R}^l)}}. \tag{17}$$

Moreover we can use the following simple manipulation:

$$G^{-1} = \tilde{G}^{-1}(G\tilde{G}^{-1})^{-1} = \tilde{G}^{-1}(I - (I - G\tilde{G}^{-1}))^{-1} = \tilde{G}^{-1}(I - (\tilde{G} - G)\tilde{G}^{-1})^{-1}.$$

Then (17) ensures that the Neumann series for $(I - (\tilde{G} - G)\tilde{G}^{-1})^{-1}$ converges and we obtain the following bound:

$$\left\|G^{-1}\right\|_{\mathcal{L}(\mathbb{R}^l)} \leq \frac{\left\|\tilde{G}^{-1}\right\|_{\mathcal{L}(\mathbb{R}^l)}}{1 - \left\|\tilde{G}^{-1}\right\|_{\mathcal{L}(\mathbb{R}^l)} \left\|G - \tilde{G}\right\|_{\mathcal{L}(\mathbb{R}^l)}} = O(1). \tag{18}$$

Now we are in the position to formulate our main generalization bound for unsupervised domain adaptation:

Theorem 1. *Consider $\tilde{f} = \sum_{k=1}^{l} \tilde{c}_k f_k$, where $\tilde{c} = (\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_l) = \tilde{G}^{-1}\tilde{g}$. Then with probability $1-\delta$ it holds that*

$$\mathcal{R}_q(\tilde{f}) - \mathcal{R}_q(f_q) \leq 2\left(\mathcal{R}_q\left(f^*\right) - \mathcal{R}_q(f_q)\right) + C\left(\log\frac{1}{\delta}\right)(n^{-1} + m^{-1}) \tag{19}$$

for some coefficient $C > 0$ not depending on m, n and δ.

Proof. We have already discussed that the coefficients in the best approximation f^* to f_q are given by $c^* = (c_1^*, c_2^*, \ldots, c_l^*) = G^{-1}\bar{g}$. Since:

$$G^{-1}(\tilde{g} - \bar{g}) + G^{-1}(G - \tilde{G})\tilde{c} = G^{-1}\tilde{g} - c^* + \tilde{c} - G^{-1}\tilde{g} = \tilde{c} - c^*$$

then from (15)–(18) with probability $1-\delta$ we have

$$\begin{aligned}\|\tilde{c} - c^*\|_{\mathbb{R}^l} &= \leq \left\|G^{-1}\right\|_{\mathcal{L}(\mathbb{R}^l)}\left(\|\tilde{g} - \bar{g}\|_{\mathbb{R}^l} + \|G - \tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)}\|\tilde{c}\|_{\mathbb{R}^l}\right)\\ &\leq C\left(\log^{\frac{1}{2}}\frac{1}{\delta}\right)(n^{-\frac{1}{2}} + m^{-\frac{1}{2}}).\end{aligned} \tag{20}$$

Moreover:

$$\begin{aligned}\mathcal{R}_q(\tilde{f}) - \mathcal{R}_q(f_q) &= \left\|\sqrt{T_{q\mathcal{X}}}(\tilde{f} - f_q)\right\|_{\mathcal{H}}^2\\ &\leq \left(\left\|\sqrt{T_{q\mathcal{X}}}(f^* - f_q)\right\|_{\mathcal{H}} + \left\|\sqrt{T_{q\mathcal{X}}}(\tilde{f} - f^*)\right\|_{\mathcal{H}}\right)^2\\ &\leq 2\left\|\sqrt{T_{q\mathcal{X}}}(f^* - f_q)\right\|_{\mathcal{H}}^2 + 2\left\|\sqrt{T_{q\mathcal{X}}}(\tilde{f} - f^*)\right\|_{\mathcal{H}}^2\\ &= 2\left(\mathcal{R}_q\left(f^*\right) - \mathcal{R}_q(f_q)\right) + 2\left\|\sqrt{T_{q\mathcal{X}}}(\tilde{f} - f^*)\right\|_{\mathcal{H}}^2\\ &\leq 2\left(\mathcal{R}_q\left(f^*\right) - \mathcal{R}_q(f_q)\right) + 2\left(\sum_{k=1}^{l}|c_k^* - \tilde{c}_k|\left\|\sqrt{T_{q\mathcal{X}}}f_k\right\|_{\mathcal{H}}\right)^2\\ &\leq 2\left(\mathcal{R}_q\left(f^*\right) - \mathcal{R}_q(f_q)\right) + 2l\|c^* - \tilde{c}\|_{\mathbb{R}^l}^2\max_k\left\|\sqrt{T_{q\mathcal{X}}}f_k\right\|_{\mathcal{H}}^2\\ &\leq 2\left(\mathcal{R}_q\left(f^*\right) - \mathcal{R}_q(f_q)\right) + 2\left\|\sqrt{T_{q\mathcal{X}}}\right\|_{\mathcal{L}(\mathcal{H})}^2 l\gamma_l^2\|c^* - \tilde{c}\|_{\mathbb{R}^l}^2,\end{aligned} \tag{21}$$

The statement of the theorem follows now from (20)–(21) (using again the inequality $(a+b)^2 \leq 2(a^2+b^2)$). □

2 Construction of Function Spaces

Let us give a short discussion on the construction of our required function space mentioned in the previous Section 1, the reproducing kernel space $\mathcal{H}$. As mentioned already in the main text, the explicit knowledge of $\mathcal{H}$ is not required, we just need to rely on its existence. First, any of our models f can be regarded as an element of some reproducing kernel space (RKHS) $\tilde{\mathcal{H}}$ satisfying the assumptions 1. This is immediate if $f : \mathcal{X} \to \mathbb{R}$ is a real valued continuous function and we take $k(x, y) = f(x)f(y)$ as the associated reproducing kernel. In the case $f : \mathcal{X} \to \mathcal{Y}$ and $\mathcal{Y}$ is finite dimensional, it is not hard to see that a similar construction is possible, as this case can again be boiled down to the construction of a kernel with real-valued output, see e.g. Remark 1 in [4] for details.

Overall we end up with a finite sequence of spaces $(\mathcal{H}_k)_{k=1}^{l+1}$ of functions living on the same domain $\mathcal{X}$ (we have $l+1$ as we also take into account the regression function), and the existence of a RKHS containing all given models and the regression function is not a real restriction. For example, in case of real valued functions, this assumption is automatically satisfied, as linear combinations of

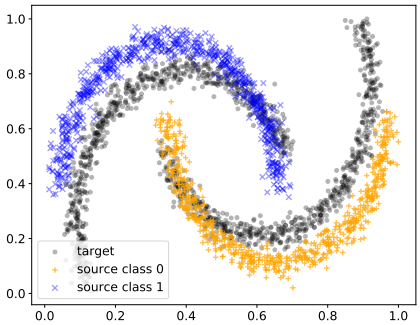

Figure 1: Transformed Moons dataset. Source data is depicted as blue $+$ and orange $\times$. Target data points are shown as black dots.

functions with the same domain which stem from a finite sequence of RKHSs belong to an RKHS. This follows from a classical result by N. Aronszajn and R. Godement, see e.g. [10, Theorem 1.4.].

There is also ongoing research on constructing function spaces (and especially associated reproducing kernels) for families of neural networks that are used in applications, see e.g. [11] for ReLU networks, [12] for recurrent networks and [13, 14] for convolutional neural networks. Incorporating these into our work may lead to refined generalization bounds that also reflect the nature of our models. We leave the details open for future work.

3 Datasets

This section provides an overview over all applied datasets from language, image, and time series domains.

Academic Dataset We rely on the Transformed Moons dataset [15], allowing us to visualize and address low-dimensional input data. The dataset consists of two-dimensional input data points forming two classes with a "moon-shaped" support. The shift from source to target domain is simulated by a transformation in input space as depicted in Figure 1.

Language Dataset To evaluate our method on a language task, we rely on the Amazon Reviews [16] dataset. This dataset consists of text reviews from four domains: books (B), DVDs (D), electronics (E), and kitchen appliances (K). Reviews are encoded in 5000 dimensional feature vectors of bag-of-words unigrams and bigrams with binary labels: label 0 if the product is ranked by 1 to 3 stars, and label 1 if the product is ranked by 4 or 5 stars. From the four categories we obtain twelve domain adaptation tasks, where each category serves once as source domain and once as target domain (see Table 9). We follow similar data splits as previous works [17, 18, 19]. In particular, we use 4000 labeled source examples and 4000 unlabeled target examples for training, and over 1000 examples for testing.

Image Dataset Our third dataset is MiniDomainNet, which is based on the DomainNet-2019 dataset [20] consisting of six different image domains (Quickdraw: Q, Real: R, Clipart: C, Sketch: S, Infograph: I, and Painting: P). We follow [15] and rely on the reduced version of DomainNet-2019, referred to as MiniDomainNet, which reduces the number of classes to the top-five largest representatives in the training set across all six domains. We further follow [20] and rely on a combined-source setting. That is, we define one domain as target domain and combine all others as combined source domain (CS), see Tabel 10. By choosing each domain once as a target domain, we obtain six domain adaptation tasks.

Time-Series Dataset The AdaTime benchmark suite [21] is a large-scale evaluation of domain adaptation algorithms on time-series data. It evaluates 10 state-of-the-art methods on four representative datasets spanning 20 cross-domain real-world scenarios, i.e., human activity recognition and sleep stage classification. The four datasets included in the benchmark are denoted by UCI-HAR, WISDM, HHAR, and Sleep-EDF. The first dataset is the *Human Activity Recognition* (HAR) [22]

dataset from the UC Irvine Repository as UCI-HAR, which contains data from three motion sensors (accelerometer, gyroscope and body-worn sensors) gathered using smartphones from 30 different subjects. It classifies their activities in several categories, namely, walking, walking upstairs, downstairs, standing, sitting, and lying down. The WISDM [23] dataset is a class-imbalanced variant from collected accelerometer sensors, including GPS data, from 29 different subjects which are performing similar activities as in the UCI-HAR dataset. The *Heterogeneity Human Activity Recognition* (HHAR) [24] dataset investigate sensor-, device- and workload-specific heterogeneities using 36 smartphones and smartwatches, consisting of 13 different device models from four manufacturers. Finally, the *Sleep Stage Classification* time-series setting aims to classify the electroencephalography (EEG) signals into five stages i.e., Wake (W), Non-Rapid Eye Movement stages (N1, N2, N3), and Rapid Eye Movement (REM). Analogous to [21, 25], we adopt the Sleep-EDF-20 dataset obtained from PhysioBank [26], which contains EEG readings from 20 healthy subjects. For all datasets, each subject is treated as an own domain, and adopt from a source subject to a target subject.

4 Experimental Setup

This section is meant to provide further details on the overall computational setting of our experiments. We start by giving an overview on the used computational resources for the specific data sets and the implementation tools. Next, we describe the network architectures for the individual data sets in greater detail. In the third subsection we elaborate on the construction of our models, and the final subsection is devoted to matrix inversion.

4.1 Computational Resources and Implementations

Overall, to compute the results in our tables, we trained 11981 models over approximately a timeframe of 1000 GPU/hours on one high-performance computing station using $8\times$NVIDIA P100 16GB, 512GB RAM, 40 Cores Xeon(R) CPU E5-2698 v4 @ 2.20GHz on CentOS Linux 7.

Transformed Moons: 3 methods $\times$ 12 parameters $\times$ 1 domain adaptation tasks $\times$ 5 seeds $+ 1 \times 5$ density estimator classifier $= 185$ trained models

Amazon Reviews: 3 methods $\times$ 16 parameters $\times$ 12 domain adaptation tasks $\times$ 5 seeds $+ 12 \times 5$ density estimator classifier $= 2940$ trained models

MiniDomainNet: 3 methods $\times$ 7 parameters $\times$ 6 domain adaptation tasks $\times$ 3 seeds $+ 6 \times 3$ density estimator classifier $= 396$ trained models

UCI-HAR: 10 methods $\times$ 14 parameters $\times$ 5 domain adaptation tasks $\times$ 3 seeds $+ 5 \times 3$ density estimator classifier $= 2115$ trained models

Sleep-EDF: 10 methods $\times$ 14 parameters $\times$ 5 domain adaptations $\times$ 3 seeds $+ 5 \times 3$ domain classifier $= 2115$

HHAR: 10 methods $\times$ 14 parameters $\times$ 5 domain adaptation tasks $\times$ 3 seeds $+ 5 \times 3$ density estimator classifier $= 2115$ trained models

WISDM: 10 methods $\times$ 14 parameters $\times$ 5 domain adaptation tasks $\times$ 3 seeds $+ 5 \times 3$ density estimator classifier $= 2115$ trained models

In Total: $185 + 2940 + 396 + 4 \times 2115 = 11981$ trained models

All methods have been implemented in Python using the *Pytorch* [27, BSD license] library. For monitoring the runs we used Weights & Biases [28, MIT license]. We use *Scikit-learn* [29] library for evaluation measures and toy datasets, and the *TQDM* [30] library, and *Tensorboard* [31] for keeping track of the progress of our experiments. We built parts of our implementation on the codebase of [15, MIT License] and [21, MIT License].

4.2 Architectures and Training Setup

In this subsection, we provide details on the model architectures and the training setup for every dataset.

Transformed Moons For the Transformed Moons dataset we use a feedforward neural network with fully connected layers and ReLU activation functions. The full architecture specification can be found in Table 1. The domain classifier (density ratio estimator) uses the same architecture. We train the models for 100 epochs with learning rate 0.001 and batchsize 256 using the Adam optimizer [32]. We reduce the learning rate at epochs 20 and 50 by a factor of 0.5 and use the same architecture and training setup for every domain adaption algorithms (CMD, MMD, DANN).

Table 1: Model architecture for the Transformed Moons dataset. Values in for neural network layers correspond to the number of output units.

Architecture		
	Layers	Values
Base Layers	Input units	2
	Fully-connected Layer	50
	ReLU Activation	
	Fully-connected Layer	50
	ReLU Activation	
	Dropout Layer	0.5
Adaptation Layer	Fully-connected Layer	25
	ReLU Activation	
	CMD	
Class Output Head	Fully-connected Layer	2
	MMD	
Class Output Head	Fully-connected Layer	2
	DANN	
Class Output Head	Fully-connected Layer	2
Domain Classifier Head	Fully-connected Layer	2

Amazon Reviews For the Amazon Reviews dataset we use a feedforward neural network with fully connected layers and ReLU activation functions similar to the setup for Transformed Moons. We also use the same architecture for the domain classifier. We train the model for 50 epochs with learning rate 0.001 and batchsize 256 using the Adam optimizer [32]. We use the same architechture and training setup for every domain adaption algorithm (CMD, MMD, DANN).

MiniDomainNet Following the pre-trained setup from [20], we use a frozen ResNet-18 backbone model which was trained on ImageNet [33], and operate subsequent computations on the 512 dimensional extracted features. To alleviate overfitting effects on pre-computed features, we perform data augmentation on each batch and forward the images through the backbone each time. We incorporate zero padding before resizing the images to 256x256 to avoid image distortions. Following the guidance for data augmentation techniques from [34], we perform random resized cropping to 224x224 with a random viewport between 70% and 100% of the original image, random horizontal flipping, color jittering of 0.25% on each RGB channel, and a ± 2 degree rotation.

After the ResNet-18 backbone output, we add several projection layers, and define the domain adaptation layers on which we use the domain adaptation methods to align the representations. The first layers are defined as a common architecture across the different domain adaptation methods. Additional layers are further added for the classification networks, according to the requirements of the individual domain adaptation methods in CMD or MMD. The number of layers/neurons in the upper layers of our architecture have been tuned in order to achieve the best performance in the source-only setup. See Table 3 for a detailed description of the architecture used. We perform experiments on all 6 domain adaptation tasks as defined in Section 3, and for each of the previously listed methods, and with 3 repetitions based on different random seeds. All methods have been trained for 50 epochs with Adam optimizer, an initial learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a MultiStep learning rate scheduler, halving the learning rate after 15 and 35 epochs.

AdaTime Unless stated otherwise, we follow the implementation and hyper-parameter settings as reported in [21]. The AdaTime suite comprises a collection of 10 domain adaptation algorithms.

Table 2: Model architecture for the Amazon Reviews dataset. Values in for neural network layers correspond to the number of output units.

Architecture		
	Layers	Values
Base Layers	Input units	5000
	Fully-connected Layer	64
	ReLU Activation	
	Fully-connected Layer	64
	ReLU Activation	
	Dropout Layer	0.5
Adaptation Layer	Fully-connected Layer	32
	ReLU Activation	
	CMD	
Class Output Head	Fully-connected Layer	2
	MMD	
Class Output Head	Fully-connected Layer	2
	DANN	
Class Output Head	Fully-connected Layer	2
Domain Classifier Head	Fully-connected Layer	2

We learned all domain adaptations models according to the following approaches (see also Table 4, Table 5 and Table 6): Deep Domain Confusion (DDC) [35], Correlation Alignment via Deep Neural Networks (Deep-Coral) [36], Higher-order Moment Matching (HoMM) [37], Minimum Discrepancy Estimation for Deep Domain Adaptation (MMDA) [38], Deep Subdomain Adaptation (DSAN) [39], Domain-Adversarial Neural Networks (DANN) [19], Conditional Adversarial Domain Adaptation (CDAN) [40], A DIRT-T Approach to Unsupervised Domain Adaptation (DIRT-T) [41], Convolutional deep Domain Adaptation model for Time-Series data (CoDATS) [42], and Adversarial Spectral Kernel Matching (AdvSKM) [43]. The backbone architecture of all models is a 1D-CNN network. It consists of three CNN blocks and each block has a 1D convolutional layer, followed by 1D batch normalization layer, ReLU activation function, 1D max pooling and dropout. In the first block, the kernel size of the convolutional layer is set according to the dataset as reported in [21]. After the convolutional blocks, we apply an 1D adaptive pooling layer. All methods are trained for 100 epochs on all datasets. The batch size is 32, except for Sleep-EDF, where we use batch size of 128. All models are trained with Adam optimizer [32] and weight decay of 10^{-4}. Additional hyper-parameters are reported in Table 8.

4.3 Model sequence

Our algorithm, IWA, constructs an ensemble from a sequence of different classifiers, e.g. obtained from a sequence of possible hyper-parameter configurations in domain adaptation algorithms. To obtain this sequence of models, we train multiple models for every domain adaptation task across all datasets with different hyper-parameter choices. For the experiments on the language, image and academic dataset, the values of the hyper-parameters are shown in Table 7. For the time series datasets we use the best hyper-parameters in [21], and, to obtain a good sequence of values, we multiply each parameter by $\lambda \in \{0, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 5, 10\}$. In this way, we generate a sequence of 14 hyper-parameter choices. The exact values are listed in Table 8.

4.4 Matrix Inversion

Matrix inversion is a well-known numerical task, especially in cases of limited computing precision and ill-conditioned matrices. In our case, similar models in the given sequence can cause numerical instability due to limited compute precision. That is, occasionally a tabula rasa inversion of the matrix $\tilde{G}$ in Algorithm 1 is numerically unstable. Various standard approaches can be applied to handle this common issue, including the exclusion of similar models and various regularization techniques. In our computational setup, we rely on on the Python routine *numpy.linalg.pinv*, which is based on

Table 3: Model architecture for the MiniDomainNet dataset. Values in for neural network layers correspond to the number of output units.

	Architecture	
	Layers	Values
Backbone Output Layer	ResNet-18 (Adaptive Average Pooling Layer)	512
Base Layers	Fully-connected Layer	1024
	Batch Normalization 1D Layer	
	ReLU	
	Fully-connected Layer	1024
	Batch Normalization 1D Layer	
	ReLU Activation	
	Dropout Layer	0.5
	Fully-connected Layer	1024
	Batch Normalization 1D Layer	
	ReLU Activation	
	Dropout Layer	0.5
	Fully-connected Layer	1024
	Batch Normalization 1D Layer	
	ReLU Activation	
	Dropout Layer	0.5
Adaptation Layers	Fully-connected Layer	512
	Batch Normalization 1D Layer	
	ReLU Activation	
	Dropout Layer	0.5
	Fully-connected Layer	512
	Batch Normalization 1D Layer	
	ReLU Activation	
	CMD	
Class Output Head	Fully-connected Layer	5
	MMD	
Class Output Head	Fully-connected Layer	5
	DANN	
Class Output Head	Fully-connected Layer	5
Domain Classifier Head	Fully-connected Layer	2

the eigendecompostion of $\tilde{G}$ (coinciding with the singular value decomposition in our case due to positive-definiteness) and an eigenvalue-based regularization based on a treshold value *rcond* for small eigenvalues, see [44, pages 138–140] for details. The choice of *rcond* depends on the scale of the Gram matrix and can therefore be chosen by source data only. Based on evaluating our method on source data only (target domain is fixed to be source domain) on several choices for *rcond*, we obtain a stable choice for *rcond* of 10^{-1} for all real-world datasets (AmazonReviews, MiniDomainNet, HAR, EEG, HHAR and WISDM). Due to the lower dimension of TransformedMoons, the scale of the matrix is different and we need a smaller value of 10^{-3} for numerical stabilization of *numpy.linalg.pinv*.

5 Detailed Empirical Results

In this section, we add all result tables for the datasets described in Section 4.3 of the main paper. Table 9 shows all domain adaptation tasks for the Amazon Review dataset. Table 10 shows all domain adaptation tasks for the MiniDomainNet experiments. Table 11, Table 12, Table 13, Table 14, Table 15, Table 16, Table 17, and Table 18 show all domain adaptation task results for the AdaTime datasets.

Baselines As addressed in the main paper, our method, IWA, is compared to ensemble learning methods that use linear regression and majority voting as *heuristic* for model aggregation, and, model selection methods with *theoretical error guarantees*. The heuristic baselines are majority voting on

Table 4: Model backbone for the AdaTime suite. Kernel size, stride, output channels of the convolutional layers are dataset dependent and are chosen according to [21].

	Architecture
	Layers
Conv Block 1	Convolutional 1D Layer Batch Normalization 1D Layer ReLU Activation Max Pooling 1D Layer Dropout
Conv Block 2	Convolutional 1D Layer Batch Normalization 1D Layer ReLU Activation Max Pooling 1D Layer Dropout
Conv Block 3	Convolutional 1D Layer Batch Normalization 1D Layer ReLU Activation Max Pooling 1D Layer Dropout
	Adaptive Pooling 1D Layer
	Methods See Table 5 and Table 6for details.

Table 5: Model architecture for the AdaTime dataset. Layer hyperparameters are dataset dependent and are chosen according to [21].

Method Architectures (Part 1)	
	DANN
Class Output Head Domain Classifier Head	Fully-connected Layer Fully-connected Layer ReLU Activation Fully-connected Layer ReLU Activation Fully-connected Layer
	DeepCoral
Class Output Head	Fully-connected Layer
	DDC
Class Output Head	Fully-connected Layer
	HoMM
Class Output Head	Fully-connected Layer
	CoDATS
Class Output Head	Fully-connected Layer ReLU Activation Fully-connected Layer ReLU Activation Fully-connected Layer
	DSAN
Class Output Head	Fully-connected Layer

target data (TMV), source-only regression (SOR), target majority voting regression (TMR), target confidence average regression (TCR). The model selection methods with theoretical error guarantees are importance weighted validation (IWV) [45] and deep embedded validation (DEV) [46]. The tables also provide a column for source-only (SO) performance and target-best (TB) performance.

Table 6: Model architecture for the AdaTime dataset. Hyper-parameters are dataset dependent and are chosen according to [21].

Method Architectures (Part 2)	
	AdvSKM
Class Output Head	Fully-connected Layer
AdvSKM Embedder 1	Fully-connected Layer
	Fully-connected Layer
	Batch Normalization 1D Layer
	Cosine Activation
	Fully-connected Layer
	Fully-connected Layer
	Batch Normalization 1D Layer
	Cosine Activation
AdvSKM Embedder 2	Fully-connected Layer
	Fully-connected Layer
	Batch Normalization 1D Layer
	ReLU Activation
	Fully-connected Layer
	Fully-connected Layer
	Batch Normalization 1D Layer
	ReLU Activation
	MMDA
Class Output Head	Fully-connected Layer
	CDAN
Class Output Head	Fully-connected Layer
Domain Classifier Head	Fully-connected Layer
	ReLU Activation
	Fully-connected Layer
	ReLU Activation
	Fully-connected Layer
	DIRT
Class Output Head	Fully-connected Layer
Domain Classifier Head	Fully-connected Layer
	ReLU Activation
	Fully-connected Layer
	ReLU Activation
	Fully-connected Layer

Table 7: Domain adaptation hyper-parameter sequences for experiments on the datasets Transformed Moons, AmazonReviews, and MiniDomainNet.

Dataset	**Method**	**Hyper-parameter λ**
Transformed Moons	CMD	$\{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 5, 10, 15\}$
	MMD	$\{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 5, 10, 15\}$
	DANN	$\{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 5, 10, 15\}$
MiniDomainNet	CMD	$\{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 4\}$
	MMD	$\{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 4\}$
	DANN	$\{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 4\}$
AmazonReviews	CMD	$\{0, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 1.5, 2, 5, 10\}$
	MMD	$\{0, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 1.5, 2, 5, 10\}$
	DANN	$\{0, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 1.5, 2, 5, 10\}$

We highlight in bold the performance of the best performing method with theoretical error guarantees, and in italic the best performing heuristic.

Table 8: Domain adaptation hyper-parameters for experiments on the AdaTime Benchmark suite. We multiply each hyper-paramter with a set of scaling factors $\lambda \in \{0, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 5, 10\}$ to obtain a sequence.

		Datasets			
Method	**Hyper-parameter**	**UCI-HAR**	**Sleep-EDF**	**WISDM**	**HHAR**
DANN	Classification loss weight	9.74	8.3	5.613	0.9603
	Domain loss weight	$\lambda \times 5.43$	$\lambda \times 0.324$	$\lambda \times 1.857$	$\lambda \times 0.9238$
DeepCoral	Classification loss weight	8.67	9.39	8.876	0.05931
	Coral loss weight	$\lambda \times 0.44$	$\lambda \times 0.19$	$\lambda \times 5.56$	$\lambda \times 8.452$
DDC	Classification loss weight	6.24	2.951	7.01	0.1593
	MMD loss weight	$\lambda \times 6.36$	$\lambda \times 8.923$	$\lambda \times 7.595$	$\lambda \times 0.2048$
HoMM	Classification loss weight	2.15	0.197	0.1913	0.2429
	Higher-order-MMD loss weight	$\lambda \times 9.13$	$\lambda \times 1.102$	$\lambda \times 4.239$	$\lambda \times 0.9824$
CoDATS	Classification loss weight	6.21	9.239	7.187	0.5416
	Adversarial loss weight	$\lambda \times 1.72$	$\lambda \times 1.342$	$\lambda \times 6.439$	$\lambda \times 0.5582$
DSAN	Classification loss weight	1.76	6.713	0.1	0.4133
	Local MMD loss weight	$\lambda \times 1.59$	$\lambda \times 6.708$	$\lambda \times 0.1$	$\lambda \times 0.16$
AdvSKM	Classification loss weight	3.05	2.5	3.05	0.4637
	Adversarial MMD loss weight	$\lambda \times 2.876$	$\lambda \times 2.5$	$\lambda \times 2.876$	$\lambda \times 0.1511$
MMDA	Classification loss weight	6.13	4.48	0.1	0.9505
	MMD loss weight	$\lambda \times 2.37$	$\lambda \times 5.951$	$\lambda \times 0.1$	$\lambda \times 0.5476$
	Conditional loss weight	$\lambda \times 7.16$	$\lambda \times 6.13$	$\lambda \times 0.4753$	$\lambda \times 0.5167$
	Coral loss weight	$\lambda \times 8.63$	$\lambda \times 3.36$	$\lambda \times 0.1$	$\lambda \times 0.5838$
CDAN	Classification loss weight	5.19	6.803	9.54	0.6636
	Adversarial loss weight	$\lambda \times 2.91$	$\lambda \times 4.726$	$\lambda \times 3.283$	$\lambda \times 0.1954$
	Conditional loss weight	$\lambda \times 1.73$	$\lambda \times 1.307$	$\lambda \times 0.1$	$\lambda \times 0.0124$
DIRT	Classification loss weight	7.0	9.183	0.1	0.9752
	Adversarial loss weight	$\lambda \times 4.51$	$\lambda \times 7.411$	$\lambda \times 0.1$	$\lambda \times 0.3892$
	Conditional loss weight	$\lambda \times 0.79$	$\lambda \times 2.564$	$\lambda \times 0.1$	$\lambda \times 0.09228$
	Virtual adversarial loss weight	$\lambda \times 9.31$	$\lambda \times 3.583$	$\lambda \times 0.1$	$\lambda \times 0.1947$

Table 9: Mean and standard deviation (after $\pm$) of target classification error on Amazon Reviews over 5 repetitions with different random initializations of model weights.

	CMD								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
B → D	0.79(±0.018)	0.806(±0.003)	0.806(±0.003)	0.806(±0.003)	*0.817(±0.011)*	0.79(±0.011)	0.732(±0.135)	**0.807(±0.004)**	0.8(±0.011)
B → E	0.711(±0.02)	0.758(±0.006)	0.758(±0.006)	0.754(±0.003)	*0.783(±0.008)*	0.727(±0.036)	0.747(±0.032)	**0.764(±0.006)**	0.782(±0.008)
B → K	0.742(±0.015)	0.793(±0.003)	0.793(±0.003)	0.794(±0.005)	*0.811(±0.01)*	0.713(±0.121)	0.775(±0.011)	**0.794(±0.007)**	0.793(±0.01)
D → B	0.766(±0.007)	0.79(±0.007)	0.79(±0.007)	0.792(±0.008)	*0.811(±0.006)*	0.772(±0.029)	0.769(±0.013)	**0.801(±0.008)**	0.8(±0.006)
D → E	0.741(±0.017)	0.791(±0.012)	0.791(±0.012)	0.79(±0.01)	*0.806(±0.015)*	0.741(±0.023)	0.702(±0.111)	**0.791(±0.012)**	0.819(±0.006)
D → K	0.754(±0.012)	0.787(±0.003)	0.787(±0.003)	0.786(±0.006)	*0.818(±0.012)*	0.751(±0.008)	0.728(±0.11)	**0.793(±0.003)**	0.809(±0.012)
E → B	0.712(±0.007)	0.735(±0.008)	0.735(±0.008)	0.732(±0.006)	*0.751(±0.016)*	0.735(±0.013)	0.636(±0.122)	**0.737(±0.009)**	0.743(±0.016)
E → D	0.72(±0.009)	0.75(±0.004)	0.75(±0.004)	0.755(±0.009)	*0.761(±0.016)*	0.742(±0.024)	0.671(±0.112)	**0.755(±0.007)**	0.764(±0.013)
E → K	0.848(±0.005)	0.871(±0.002)	0.871(±0.002)	0.866(±0.005)	*0.882(±0.003)*	0.854(±0.011)	0.712(±0.194)	**0.873(±0.003)**	0.866(±0.003)
K → B	0.69(±0.01)	0.728(±0.006)	0.728(±0.006)	0.728(±0.003)	*0.739(±0.009)*	0.705(±0.016)	0.714(±0.007)	**0.733(±0.004)**	0.738(±0.009)
K → D	0.719(±0.009)	0.757(±0.007)	0.757(±0.007)	0.754(±0.01)	*0.768(±0.014)*	0.739(±0.022)	0.684(±0.101)	**0.759(±0.005)**	0.777(±0.01)
K → E	0.826(±0.006)	0.859(±0.003)	0.859(±0.003)	0.859(±0.001)	*0.869(±0.004)*	0.834(±0.02)	0.839(±0.017)	**0.86(±0.003)**	0.854(±0.004)
Avg.	0.752(±0.048)	0.785(±0.043)	0.785(±0.043)	0.785(±0.043)	*0.801(±0.044)*	0.759(±0.058)	0.726(±0.104)	**0.789(±0.043)**	0.795(±0.037)

	MMD								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
B → D	0.79(±0.018)	0.808(±0.004)	0.808(±0.004)	0.805(±0.004)	*0.817(±0.006)*	0.787(±0.005)	0.509(±0.015)	**0.808(±0.003)**	0.795(±0.006)
B → E	0.711(±0.02)	0.736(±0.006)	0.736(±0.006)	0.721(±0.007)	*0.754(±0.014)*	0.722(±0.041)	0.645(±0.101)	**0.729(±0.008)**	0.745(±0.014)
B → K	0.742(±0.015)	0.765(±0.008)	0.765(±0.008)	0.759(±0.011)	*0.778(±0.009)*	0.647(±0.131)	**0.665(±0.134)**	0.661(±0.229)	0.769(±0.009)
D → B	0.766(±0.007)	0.783(±0.005)	0.783(±0.005)	0.778(±0.004)	*0.794(±0.006)*	0.705(±0.112)	0.618(±0.146)	**0.788(±0.009)**	0.783(±0.006)
D → E	0.741(±0.017)	0.775(±0.006)	0.775(±0.006)	0.761(±0.007)	*0.786(±0.012)*	0.743(±0.022)	0.615(±0.119)	**0.766(±0.007)**	0.766(±0.012)
D → K	0.754(±0.012)	0.767(±0.003)	0.767(±0.003)	0.765(±0.005)	*0.786(±0.006)*	0.692(±0.084)	0.635(±0.106)	**0.769(±0.005)**	0.784(±0.006)
E → B	0.712(±0.007)	0.723(±0.006)	0.723(±0.006)	0.727(±0.002)	*0.735(±0.007)*	0.708(±0.011)	0.583(±0.078)	**0.731(±0.008)**	0.713(±0.007)
E → D	0.72(±0.009)	0.738(±0.007)	0.738(±0.007)	0.734(±0.007)	*0.743(±0.006)*	0.688(±0.093)	0.655(±0.077)	**0.738(±0.008)**	0.73(±0.006)
E → K	0.848(±0.005)	0.861(±0.004)	0.861(±0.004)	0.86(±0.005)	*0.866(±0.004)*	0.825(±0.036)	0.835(±0.021)	**0.864(±0.004)**	0.852(±0.004)
K → B	0.69(±0.01)	0.705(±0.01)	0.705(±0.01)	0.702(±0.01)	*0.72(±0.004)*	0.684(±0.006)	0.538(±0.013)	**0.705(±0.006)**	0.708(±0.004)
K → D	0.719(±0.009)	0.732(±0.006)	0.732(±0.006)	0.735(±0.003)	*0.741(±0.008)*	0.725(±0.018)	0.677(±0.086)	**0.736(±0.005)**	0.733(±0.008)
K → E	0.826(±0.006)	0.849(±0.003)	0.849(±0.003)	0.844(±0.003)	*0.859(±0.002)*	0.832(±0.006)	0.648(±0.098)	**0.848(±0.002)**	0.832(±0.002)
Avg.	0.752(±0.048)	0.77(±0.047)	0.77(±0.047)	0.766(±0.047)	*0.782(±0.046)*	0.73(±0.08)	0.635(±0.115)	**0.762(±0.082)**	0.768(±0.043)

	DANN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
B → D	0.783(±0.008)	0.799(±0.003)	0.799(±0.003)	*0.802(±0.004)*	0.796(±0.018)	0.788(±0.005)	0.74(±0.13)	**0.799(±0.002)**	0.805(±0.009)
B → E	0.702(±0.008)	0.729(±0.006)	0.729(±0.006)	*0.73(±0.004)*	0.72(±0.006)	0.727(±0.01)	0.646(±0.104)	**0.732(±0.006)**	0.728(±0.006)
B → K	0.739(±0.01)	0.762(±0.003)	0.762(±0.003)	*0.765(±0.005)*	0.752(±0.024)	0.727(±0.044)	0.651(±0.138)	**0.766(±0.003)**	0.761(±0.003)
D → B	0.767(±0.009)	0.786(±0.006)	0.786(±0.006)	0.784(±0.006)	*0.794(±0.007)*	0.787(±0.014)	0.719(±0.128)	**0.788(±0.004)**	0.794(±0.012)
D → E	0.732(±0.006)	*0.773(±0.009)*	0.773(±0.009)	0.767(±0.013)	0.758(±0.027)	0.747(±0.027)	0.64(±0.102)	**0.769(±0.006)**	0.765(±0.009)
D → K	0.75(±0.005)	0.767(±0.004)	0.767(±0.004)	*0.768(±0.003)*	0.763(±0.019)	0.705(±0.096)	0.719(±0.07)	**0.738(±0.068)**	0.764(±0.003)
E → B	0.709(±0.007)	*0.728(±0.008)*	0.728(±0.008)	0.725(±0.008)	0.718(±0.024)	0.691(±0.043)	0.568(±0.084)	**0.727(±0.009)**	0.735(±0.009)
E → D	0.725(±0.011)	*0.74(±0.008)*	0.74(±0.008)	0.74(±0.01)	0.726(±0.023)	0.739(±0.013)	0.527(±0.104)	**0.743(±0.006)**	0.736(±0.006)
E → K	0.845(±0.004)	0.863(±0.003)	0.863(±0.003)	0.863(±0.002)	*0.867(±0.008)*	0.845(±0.013)	0.736(±0.142)	**0.864(±0.002)**	0.848(±0.008)
K → B	0.69(±0.012)	0.703(±0.007)	0.703(±0.007)	0.705(±0.002)	*0.708(±0.009)*	0.694(±0.009)	0.577(±0.103)	**0.703(±0.004)**	0.694(±0.009)
K → D	0.728(±0.01)	*0.742(±0.008)*	0.742(±0.008)	0.741(±0.007)	0.723(±0.057)	0.711(±0.027)	0.703(±0.07)	**0.744(±0.006)**	0.747(±0.018)
K → E	0.835(±0.005)	0.846(±0.005)	0.846(±0.005)	0.846(±0.004)	*0.848(±0.012)*	0.83(±0.007)	0.722(±0.172)	**0.849(±0.003)**	0.84(±0.003)
Avg.	0.75(±0.048)	*0.77(±0.046)*	0.77(±0.046)	0.77(±0.047)	0.764(±0.055)	0.749(±0.06)	0.662(±0.126)	**0.768(±0.051)**	0.768(±0.044)

Table 10: Mean and standard deviation (after $\pm$) of target classification error on MiniDomainNet over 3 repetitions with different random initializations of model weights.

	CMD								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
CS → S	0.702(±0.005)	0.713(±0.002)	0.713(±0.002)	0.717(±0.002)	*0.718(±0.002)*	0.704(±0.017)	0.71(±0.008)	**0.718(±0.003)**	0.71(±0.003)
CS → I	0.398(±0.03)	0.447(±0.014)	0.447(±0.014)	*0.456(±0.009)*	0.448(±0.003)	0.438(±0.016)	0.111(±0.0)	**0.451(±0.009)**	0.457(±0.003)
CS → C	0.767(±0.013)	0.763(±0.002)	0.763(±0.002)	0.763(±0.015)	*0.766(±0.008)*	0.763(±0.019)	0.717(±0.027)	**0.763(±0.011)**	0.767(±0.013)
CS → R	0.928(±0.018)	0.925(±0.013)	0.925(±0.013)	0.928(±0.013)	*0.93(±0.008)*	0.92(±0.025)	0.151(±0.0)	**0.926(±0.007)**	0.928(±0.008)
CS → Q	0.436(±0.02)	0.491(±0.028)	0.491(±0.028)	0.496(±0.026)	*0.507(±0.029)*	0.5(±0.035)	0.47(±0.018)	**0.505(±0.045)**	0.55(±0.074)
CS → P	0.829(±0.014)	0.717(±0.025)	0.717(±0.025)	*0.758(±0.006)*	0.754(±0.019)	0.733(±0.022)	0.697(±0.033)	**0.763(±0.003)**	0.829(±0.014)
Avg.	0.677(±0.202)	0.676(±0.168)	0.676(±0.168)	0.686(±0.168)	*0.687(±0.169)*	0.676(±0.168)	0.476(±0.266)	**0.688(±0.168)**	0.707(±0.16)

	MMD								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
CS → S	0.702(±0.005)	0.703(±0.007)	0.703(±0.007)	0.699(±0.009)	*0.71(±0.004)*	0.693(±0.007)	0.699(±0.003)	**0.707(±0.008)**	0.702(±0.004)
CS → I	0.398(±0.03)	0.438(±0.004)	0.438(±0.004)	0.436(±0.005)	*0.442(±0.005)*	**0.44(±0.016)**	0.44(±0.016)	0.438(±0.009)	0.447(±0.024)
CS → C	0.767(±0.013)	0.753(±0.01)	0.753(±0.01)	0.751(±0.014)	*0.755(±0.005)*	0.743(±0.035)	0.722(±0.024)	**0.749(±0.008)**	0.767(±0.013)
CS → R	0.928(±0.018)	0.927(±0.0)	0.927(±0.0)	0.924(±0.006)	*0.928(±0.002)*	**0.928(±0.018)**	0.928(±0.018)	0.927(±0.001)	0.928(±0.002)
CS → Q	0.436(±0.02)	0.382(±0.04)	0.382(±0.04)	0.378(±0.036)	*0.408(±0.012)*	0.384(±0.042)	0.367(±0.038)	**0.411(±0.024)**	0.436(±0.02)
CS → P	0.829(±0.014)	*0.803(±0.003)*	0.803(±0.003)	0.802(±0.004)	0.799(±0.018)	0.783(±0.011)	0.778(±0.003)	**0.801(±0.009)**	0.829(±0.014)
Avg.	0.677(±0.202)	0.668(±0.202)	0.668(±0.202)	0.665(±0.201)	*0.674(±0.194)*	0.662(±0.198)	0.655(±0.2)	**0.672(±0.194)**	0.685(±0.185)

	DANN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
CS → S	0.68(±0.01)	0.712(±0.008)	0.712(±0.008)	0.714(±0.003)	*0.725(±0.011)*	0.708(±0.022)	0.702(±0.007)	**0.715(±0.002)**	0.702(±0.011)
CS → I	0.366(±0.016)	0.447(±0.02)	0.447(±0.02)	*0.447(±0.017)*	0.429(±0.016)	0.414(±0.017)	0.329(±0.074)	**0.436(±0.017)**	0.444(±0.017)
CS → C	0.762(±0.018)	0.773(±0.013)	0.773(±0.013)	0.776(±0.011)	*0.791(±0.014)*	0.772(±0.041)	0.751(±0.017)	**0.787(±0.014)**	0.762(±0.014)
CS → R	0.928(±0.012)	0.93(±0.006)	0.93(±0.006)	0.929(±0.006)	*0.931(±0.007)*	0.916(±0.004)	0.916(±0.004)	**0.929(±0.007)**	0.928(±0.007)
CS → Q	0.441(±0.022)	0.427(±0.036)	0.427(±0.036)	0.424(±0.049)	*0.428(±0.034)*	0.397(±0.046)	**0.442(±0.043)**	0.411(±0.071)	0.457(±0.039)
CS → P	0.848(±0.003)	*0.804(±0.029)*	0.804(±0.029)	0.802(±0.022)	0.786(±0.019)	0.748(±0.068)	0.775(±0.022)	**0.8(±0.022)**	0.848(±0.003)
Avg.	0.671(±0.211)	0.682(±0.192)	0.682(±0.192)	*0.682(±0.192)*	0.681(±0.196)	0.659(±0.199)	0.652(±0.21)	**0.679(±0.199)**	0.69(±0.183)

Table 11: Mean and standard deviation (after ±) of target classification error on Sleep-EDF (Part 1) over 3 repetitions with different random initializations of model weights.

	HoMM								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.564(±0.015)	0.53(±0.018)	0.53(±0.018)	0.531(±0.017)	*0.634(±0.045)*	**0.609(±0.027)**	0.573(±0.049)	0.599(±0.108)	0.6(±0.045)
12 → 5	0.691(±0.041)	0.766(±0.008)	0.766(±0.008)	0.763(±0.015)	*0.796(±0.03)*	**0.771(±0.049)**	0.771(±0.049)	0.766(±0.004)	0.798(±0.033)
16 → 1	0.661(±0.075)	0.664(±0.008)	0.664(±0.008)	*0.667(±0.01)*	0.605(±0.011)	**0.682(±0.051)**	0.682(±0.051)	0.66(±0.005)	0.706(±0.073)
7 → 18	0.697(±0.063)	*0.711(±0.024)*	0.711(±0.024)	0.711(±0.027)	0.66(±0.014)	**0.721(±0.014)**	0.704(±0.02)	0.72(±0.024)	0.742(±0.042)
9 → 14	0.684(±0.116)	0.809(±0.027)	0.809(±0.027)	0.806(±0.018)	*0.831(±0.018)*	0.78(±0.049)	0.785(±0.046)	**0.823(±0.02)**	0.815(±0.018)
Avg.	0.659(±0.079)	0.696(±0.101)	0.696(±0.101)	0.696(±0.099)	*0.705(±0.096)*	0.713(±0.073)	0.703(±0.087)	**0.714(±0.092)**	0.732(±0.077)

	AdvSKM								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.618(±0.018)	0.617(±0.031)	0.617(±0.031)	0.611(±0.022)	*0.686(±0.043)*	0.66(±0.063)	0.663(±0.044)	**0.676(±0.088)**	0.643(±0.043)
12 → 5	0.741(±0.01)	0.73(±0.029)	0.73(±0.029)	0.715(±0.023)	*0.788(±0.022)*	**0.773(±0.027)**	0.728(±0.066)	0.74(±0.03)	0.75(±0.022)
16 → 1	0.674(±0.042)	0.67(±0.005)	0.67(±0.005)	*0.676(±0.013)*	0.611(±0.026)	0.653(±0.049)	0.653(±0.049)	**0.668(±0.01)**	0.717(±0.057)
7 → 18	0.699(±0.017)	*0.712(±0.024)*	0.712(±0.024)	0.711(±0.016)	0.685(±0.03)	0.693(±0.016)	0.693(±0.016)	**0.72(±0.006)**	0.73(±0.032)
9 → 14	0.76(±0.029)	0.833(±0.012)	0.833(±0.012)	*0.837(±0.006)*	0.827(±0.01)	0.776(±0.023)	0.784(±0.015)	**0.833(±0.008)**	0.811(±0.006)
Avg.	0.699(±0.056)	0.713(±0.077)	0.713(±0.077)	0.71(±0.078)	*0.719(±0.084)*	0.711(±0.065)	0.704(±0.061)	**0.727(±0.071)**	0.73(±0.054)

	DIRT								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.565(±0.083)	0.444(±0.054)	0.444(±0.054)	0.457(±0.034)	*0.565(±0.039)*	0.443(±0.088)	0.436(±0.098)	**0.564(±0.047)**	0.565(±0.083)
12 → 5	0.684(±0.128)	0.832(±0.043)	0.832(±0.043)	0.836(±0.035)	*0.872(±0.002)*	**0.866(±0.005)**	0.736(±0.131)	0.816(±0.036)	0.855(±0.002)
16 → 1	0.534(±0.237)	*0.78(±0.009)*	0.78(±0.009)	0.774(±0.009)	0.765(±0.013)	0.761(±0.028)	0.761(±0.028)	**0.784(±0.013)**	0.801(±0.025)
7 → 18	0.78(±0.022)	*0.83(±0.0)*	0.83(±0.0)	0.83(±0.0)	0.761(±0.054)	0.78(±0.022)	0.805(±0.044)	**0.83(±0.0)**	0.83(±0.0)
9 → 14	0.621(±0.064)	*0.878(±0.018)*	0.878(±0.018)	0.872(±0.024)	0.859(±0.007)	0.82(±0.034)	0.758(±0.075)	**0.866(±0.012)**	0.884(±0.015)
Avg.	0.637(±0.142)	0.753(±0.165)	0.753(±0.165)	0.754(±0.158)	*0.764(±0.117)*	0.734(±0.16)	0.699(±0.155)	**0.772(±0.114)**	0.787(±0.114)

	DDC								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.559(±0.022)	0.635(±0.044)	0.635(±0.044)	0.611(±0.022)	*0.648(±0.042)*	0.612(±0.076)	0.539(±0.027)	**0.621(±0.01)**	0.68(±0.075)
12 → 5	0.694(±0.044)	0.716(±0.012)	0.716(±0.012)	0.706(±0.008)	*0.777(±0.022)*	**0.749(±0.01)**	0.749(±0.01)	0.734(±0.004)	0.786(±0.052)
16 → 1	0.65(±0.075)	0.662(±0.017)	0.662(±0.017)	*0.666(±0.014)*	0.605(±0.024)	0.602(±0.043)	0.583(±0.035)	**0.657(±0.002)**	0.707(±0.073)
7 → 18	0.723(±0.039)	0.698(±0.05)	0.698(±0.05)	*0.704(±0.039)*	0.616(±0.054)	0.723(±0.039)	0.723(±0.039)	**0.736(±0.019)**	0.742(±0.011)
9 → 14	0.685(±0.132)	0.803(±0.002)	0.803(±0.002)	0.802(±0.008)	*0.829(±0.022)*	0.79(±0.035)	0.753(±0.033)	**0.81(±0.006)**	0.812(±0.022)
Avg.	0.662(±0.086)	*0.703(±0.065)*	0.703(±0.065)	0.698(±0.067)	0.695(±0.099)	0.695(±0.087)	0.669(±0.097)	**0.712(±0.069)**	0.746(±0.049)

	MMDA								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.557(±0.012)	0.521(±0.018)	0.521(±0.018)	0.512(±0.017)	*0.651(±0.033)*	**0.552(±0.016)**	0.523(±0.021)	0.547(±0.061)	0.557(±0.033)
12 → 5	0.681(±0.073)	*0.818(±0.018)*	0.818(±0.018)	0.818(±0.018)	0.803(±0.03)	0.77(±0.088)	0.77(±0.088)	**0.812(±0.014)**	0.845(±0.008)
16 → 1	0.652(±0.112)	0.699(±0.012)	0.699(±0.012)	*0.706(±0.017)*	0.669(±0.017)	**0.707(±0.018)**	0.705(±0.021)	0.702(±0.012)	0.732(±0.017)
7 → 18	0.73(±0.066)	*0.61(±0.039)*	0.61(±0.039)	0.591(±0.011)	0.541(±0.218)	**0.73(±0.066)**	0.723(±0.076)	0.654(±0.022)	0.73(±0.066)
9 → 14	0.72(±0.096)	0.837(±0.016)	0.837(±0.016)	0.824(±0.027)	*0.845(±0.012)*	0.798(±0.036)	0.764(±0.015)	**0.835(±0.022)**	0.799(±0.012)
Avg.	0.668(±0.093)	0.697(±0.126)	0.697(±0.126)	0.69(±0.129)	*0.702(±0.142)*	**0.711(±0.099)**	0.697(±0.104)	0.71(±0.113)	0.733(±0.098)

Table 12: Mean and standard deviation (after ±) of target classification error on Sleep-EDF (Part 2) over 3 repetitions with different random initializations of model weights.

	CoDATS								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.525(±0.079)	0.495(±0.067)	0.495(±0.067)	0.501(±0.054)	*0.617(±0.059)*	0.504(±0.031)	0.53(±0.059)	**0.595(±0.04)**	0.57(±0.059)
12 → 5	0.757(±0.038)	*0.829(±0.011)*	0.829(±0.011)	0.824(±0.017)	0.828(±0.008)	0.751(±0.033)	0.801(±0.035)	**0.818(±0.008)**	0.803(±0.011)
16 → 1	0.701(±0.052)	0.736(±0.048)	0.736(±0.048)	*0.741(±0.038)*	0.644(±0.028)	0.664(±0.132)	0.497(±0.342)	**0.706(±0.017)**	0.745(±0.018)
7 → 18	0.654(±0.06)	0.734(±0.007)	0.734(±0.007)	*0.74(±0.008)*	0.682(±0.038)	**0.772(±0.035)**	0.772(±0.035)	0.733(±0.009)	0.764(±0.035)
9 → 14	0.822(±0.03)	0.815(±0.024)	0.815(±0.024)	*0.828(±0.014)*	0.809(±0.01)	0.81(±0.011)	0.811(±0.005)	**0.844(±0.023)**	0.822(±0.023)
Avg.	0.692(±0.114)	0.722(±0.129)	0.722(±0.129)	*0.727(±0.126)*	0.716(±0.094)	0.7(±0.126)	0.682(±0.196)	**0.739(±0.093)**	0.741(±0.09)

	Deep-Coral								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.555(±0.018)	0.628(±0.018)	0.628(±0.018)	0.608(±0.032)	*0.658(±0.061)*	0.583(±0.016)	0.54(±0.037)	**0.621(±0.008)**	0.668(±0.072)
12 → 5	0.694(±0.039)	0.714(±0.009)	0.714(±0.009)	0.706(±0.01)	*0.777(±0.024)*	**0.75(±0.024)**	0.75(±0.024)	0.733(±0.006)	0.785(±0.051)
16 → 1	0.651(±0.079)	*0.666(±0.014)*	0.666(±0.014)	0.666(±0.017)	0.604(±0.02)	0.639(±0.089)	0.584(±0.033)	**0.659(±0.003)**	0.709(±0.073)
7 → 18	0.69(±0.062)	*0.71(±0.03)*	0.71(±0.03)	0.704(±0.035)	0.651(±0.029)	**0.715(±0.032)**	0.699(±0.01)	0.704(±0.015)	0.725(±0.016)
9 → 14	0.689(±0.136)	0.798(±0.002)	0.798(±0.002)	0.798(±0.018)	*0.831(±0.02)*	0.789(±0.031)	0.754(±0.035)	**0.811(±0.006)**	0.812(±0.02)
Avg.	0.656(±0.086)	0.703(±0.061)	0.703(±0.061)	0.696(±0.067)	*0.704(±0.093)*	0.695(±0.087)	0.666(±0.094)	**0.706(±0.068)**	0.74(±0.052)

	CDAN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.544(±0.044)	0.414(±0.01)	0.414(±0.01)	0.428(±0.013)	*0.595(±0.013)*	0.493(±0.062)	0.419(±0.084)	**0.529(±0.103)**	0.544(±0.013)
12 → 5	0.686(±0.114)	*0.841(±0.016)*	0.841(±0.016)	0.841(±0.016)	0.832(±0.02)	0.828(±0.026)	0.706(±0.116)	**0.833(±0.002)**	0.837(±0.016)
16 → 1	0.533(±0.121)	*0.742(±0.021)*	0.742(±0.021)	0.74(±0.023)	0.658(±0.033)	0.618(±0.091)	0.618(±0.091)	**0.732(±0.029)**	0.795(±0.017)
7 → 18	0.694(±0.043)	*0.762(±0.004)*	0.762(±0.004)	0.759(±0.015)	0.712(±0.042)	**0.781(±0.026)**	0.78(±0.024)	0.766(±0.004)	0.797(±0.017)
9 → 14	0.737(±0.069)	0.858(±0.012)	0.858(±0.012)	*0.859(±0.01)*	0.849(±0.01)	0.728(±0.069)	0.789(±0.082)	**0.857(±0.006)**	0.846(±0.01)
Avg.	0.639(±0.112)	0.723(±0.167)	0.723(±0.167)	0.726(±0.162)	*0.729(±0.104)*	0.69(±0.135)	0.662(±0.158)	**0.743(±0.127)**	0.764(±0.112)

	DANN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.553(±0.069)	0.611(±0.033)	0.611(±0.033)	0.6(±0.041)	*0.671(±0.042)*	**0.635(±0.05)**	0.599(±0.121)	0.629(±0.041)	0.63(±0.042)
12 → 5	0.702(±0.014)	0.75(±0.032)	0.75(±0.032)	0.747(±0.029)	*0.788(±0.027)*	**0.775(±0.062)**	0.76(±0.049)	0.754(±0.017)	0.755(±0.027)
16 → 1	0.605(±0.187)	*0.698(±0.022)*	0.698(±0.022)	0.691(±0.017)	0.64(±0.012)	0.664(±0.04)	**0.681(±0.066)**	0.674(±0.018)	0.721(±0.022)
7 → 18	0.679(±0.086)	*0.604(±0.033)*	0.604(±0.033)	0.597(±0.039)	0.509(±0.164)	**0.679(±0.086)**	0.491(±0.154)	0.566(±0.05)	0.679(±0.086)
9 → 14	0.689(±0.086)	0.831(±0.01)	0.831(±0.01)	0.833(±0.014)	*0.84(±0.014)*	0.73(±0.122)	0.798(±0.026)	**0.835(±0.006)**	0.815(±0.014)
Avg.	0.646(±0.106)	*0.699(±0.092)*	0.699(±0.092)	0.694(±0.096)	0.689(±0.137)	**0.697(±0.084)**	0.666(±0.141)	0.691(±0.101)	0.72(±0.063)

	DSAN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 11	0.559(±0.059)	0.458(±0.077)	0.458(±0.077)	0.478(±0.088)	*0.612(±0.026)*	0.582(±0.031)	0.462(±0.12)	**0.618(±0.031)**	0.559(±0.031)
12 → 5	0.648(±0.031)	0.85(±0.011)	0.85(±0.011)	0.853(±0.008)	*0.872(±0.014)*	**0.868(±0.016)**	0.697(±0.212)	0.853(±0.008)	0.855(±0.014)
16 → 1	0.734(±0.038)	*0.732(±0.039)*	0.732(±0.039)	0.705(±0.045)	0.632(±0.031)	0.627(±0.051)	0.627(±0.051)	**0.716(±0.014)**	0.753(±0.05)
7 → 18	0.73(±0.066)	0.421(±0.079)	0.421(±0.079)	0.352(±0.022)	*0.667(±0.154)*	**0.73(±0.066)**	0.66(±0.182)	0.352(±0.011)	0.73(±0.066)
9 → 14	0.652(±0.147)	0.812(±0.008)	0.812(±0.008)	0.811(±0.016)	*0.818(±0.037)*	0.759(±0.075)	0.678(±0.234)	**0.822(±0.026)**	0.806(±0.026)
Avg.	0.665(±0.095)	0.655(±0.192)	0.655(±0.192)	0.64(±0.204)	*0.72(±0.125)*	**0.713(±0.114)**	0.625(±0.17)	0.672(±0.187)	0.741(±0.101)

Table 13: Mean and standard deviation (after $\pm$) of target classification error on UCI-HAR (Part 1) over 3 repetitions with different random initializations of model weights.

HoMM

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.681(±0.012)	0.674(±0.024)	0.674(±0.024)	*0.688(±0.0)*	0.688(±0.0)	0.681(±0.012)	0.681(±0.012)	**0.688(±0.0)**	0.701(±0.032)
2 → 11	1.0(±0.0)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.854(±0.1)	**1.0(±0.0)**	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)
6 → 23	0.875(±0.036)	*0.91(±0.024)*	0.91(±0.024)	0.896(±0.0)	0.875(±0.036)	0.875(±0.036)	0.875(±0.036)	**0.91(±0.024)**	0.938(±0.0)
7 → 13	0.91(±0.032)	*0.903(±0.012)*	0.903(±0.012)	0.903(±0.012)	0.854(±0.036)	**0.91(±0.032)**	0.91(±0.032)	0.896(±0.0)	0.91(±0.024)
9 → 18	0.521(±0.072)	*0.625(±0.042)*	0.625(±0.042)	0.625(±0.042)	0.396(±0.182)	0.611(±0.06)	0.611(±0.06)	**0.708(±0.0)**	0.708(±0.042)
Avg.	0.797(±0.182)	*0.822(±0.153)*	0.822(±0.153)	0.822(±0.148)	0.733(±0.205)	0.815(±0.154)	0.815(±0.154)	**0.84(±0.126)**	0.851(±0.123)

AdvSKM

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.674(±0.012)	0.681(±0.012)	0.681(±0.012)	0.681(±0.012)	*0.688(±0.021)*	0.674(±0.012)	0.674(±0.012)	**0.688(±0.0)**	0.715(±0.048)
2 → 11	1.0(±0.0)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.844(±0.083)	0.938(±0.108)	0.938(±0.108)	**1.0(±0.0)**	1.0(±0.0)
6 → 23	0.84(±0.064)	*0.896(±0.0)*	0.896(±0.0)	0.889(±0.012)	0.889(±0.012)	0.84(±0.064)	0.84(±0.064)	**0.896(±0.0)**	0.903(±0.032)
7 → 13	0.896(±0.036)	0.875(±0.0)	0.875(±0.0)	*0.889(±0.024)*	0.854(±0.036)	**0.896(±0.036)**	0.896(±0.036)	0.889(±0.024)	0.903(±0.032)
9 → 18	0.521(±0.108)	0.458(±0.063)	0.458(±0.063)	*0.472(±0.043)*	0.431(±0.052)	0.458(±0.108)	**0.5(±0.095)**	0.5(±0.036)	0.549(±0.103)
Avg.	0.786(±0.182)	0.782(±0.2)	0.782(±0.2)	*0.786(±0.196)*	0.741(±0.181)	0.761(±0.193)	0.769(±0.178)	**0.794(±0.186)**	0.814(±0.162)

DIRT

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.667(±0.036)	*0.729(±0.055)*	0.729(±0.055)	0.729(±0.055)	0.715(±0.052)	0.667(±0.036)	0.667(±0.036)	**0.743(±0.032)**	0.84(±0.079)
2 → 11	1.0(±0.0)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.885(±0.036)	**1.0(±0.0)**	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)
6 → 23	0.854(±0.072)	0.924(±0.024)	0.924(±0.024)	*0.931(±0.012)*	0.903(±0.012)	0.91(±0.024)	0.91(±0.024)	**0.931(±0.012)**	0.944(±0.052)
7 → 13	0.868(±0.06)	*0.951(±0.012)*	0.951(±0.012)	0.951(±0.012)	0.924(±0.032)	0.868(±0.06)	0.868(±0.06)	**0.958(±0.0)**	0.965(±0.012)
9 → 18	0.507(±0.024)	*0.826(±0.012)*	0.826(±0.012)	0.826(±0.012)	0.382(±0.084)	0.632(±0.126)	0.632(±0.126)	**0.812(±0.0)**	0.938(±0.042)
Avg.	0.779(±0.183)	0.886(±0.103)	0.886(±0.103)	*0.888(±0.103)*	0.762(±0.215)	0.815(±0.157)	0.815(±0.157)	**0.889(±0.1)**	0.938(±0.053)

DDC

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.681(±0.012)	0.688(±0.0)	0.688(±0.0)	0.688(±0.0)	*0.694(±0.012)*	0.681(±0.012)	0.681(±0.012)	**0.688(±0.0)**	0.708(±0.021)
2 → 11	1.0(±0.0)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.833(±0.072)	0.896(±0.13)	0.906(±0.136)	**1.0(±0.0)**	1.0(±0.0)
6 → 23	0.903(±0.024)	*0.896(±0.0)*	0.896(±0.0)	0.896(±0.0)	0.826(±0.12)	0.882(±0.032)	0.882(±0.032)	**0.896(±0.0)**	0.91(±0.024)
7 → 13	0.833(±0.055)	0.896(±0.036)	0.896(±0.036)	*0.903(±0.032)*	0.868(±0.052)	0.875(±0.055)	0.875(±0.055)	**0.889(±0.024)**	0.896(±0.032)
9 → 18	0.479(±0.075)	*0.5(±0.127)*	0.5(±0.127)	0.5(±0.108)	0.486(±0.024)	0.396(±0.021)	0.396(±0.021)	**0.514(±0.098)**	0.493(±0.098)
Avg.	0.779(±0.193)	0.796(±0.192)	0.796(±0.192)	*0.797(±0.191)*	0.742(±0.157)	0.746(±0.206)	0.748(±0.209)	**0.797(±0.184)**	0.801(±0.181)

MMDA

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.681(±0.012)	0.681(±0.032)	0.681(±0.032)	*0.688(±0.0)*	0.681(±0.012)	0.681(±0.012)	0.681(±0.012)	**0.694(±0.012)**	0.771(±0.042)
2 → 11	1.0(±0.0)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.917(±0.048)	0.938(±0.062)	0.938(±0.062)	**1.0(±0.0)**	1.0(±0.0)
6 → 23	0.868(±0.048)	*0.896(±0.0)*	0.896(±0.0)	0.896(±0.0)	0.889(±0.012)	0.868(±0.048)	0.868(±0.048)	**0.896(±0.0)**	0.917(±0.021)
7 → 13	0.91(±0.032)	0.917(±0.0)	0.917(±0.0)	*0.931(±0.024)*	0.861(±0.032)	0.91(±0.032)	0.91(±0.032)	**0.931(±0.024)**	0.944(±0.024)
9 → 18	0.562(±0.146)	0.479(±0.042)	0.479(±0.042)	0.5(±0.021)	*0.514(±0.064)*	0.597(±0.012)	**0.611(±0.032)**	0.5(±0.036)	0.653(±0.064)
Avg.	0.804(±0.176)	0.794(±0.197)	0.794(±0.197)	*0.803(±0.191)*	0.772(±0.162)	0.799(±0.143)	0.801(±0.14)	**0.804(±0.19)**	0.857(±0.127)

Table 14: Mean and standard deviation (after ±) of target classification error on UCI-HAR (Part 2) over 3 repetitions with different random initializations of model weights.

	CoDATS								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.715(±0.024)	0.674(±0.052)	0.674(±0.052)	0.674(±0.052)	*0.708(±0.021)*	**0.715(±0.024)**	0.715(±0.024)	0.694(±0.012)	0.715(±0.024)
2 → 11	0.948(±0.048)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.896(±0.095)	0.969(±0.031)	0.969(±0.031)	**1.0(±0.0)**	1.0(±0.0)
6 → 23	0.84(±0.024)	*0.931(±0.012)*	0.931(±0.012)	0.924(±0.024)	0.861(±0.032)	0.882(±0.048)	0.833(±0.021)	**0.931(±0.012)**	0.951(±0.024)
7 → 13	0.819(±0.052)	*0.924(±0.024)*	0.924(±0.024)	0.924(±0.024)	0.917(±0.036)	0.819(±0.052)	0.819(±0.052)	**0.938(±0.0)**	0.944(±0.012)
9 → 18	0.542(±0.055)	*0.625(±0.108)*	0.625(±0.108)	0.604(±0.036)	0.583(±0.146)	0.549(±0.06)	**0.736(±0.139)**	0.639(±0.024)	0.771(±0.083)
Avg.	0.773(±0.147)	*0.831(±0.163)*	0.831(±0.163)	0.825(±0.164)	0.793(±0.149)	0.787(±0.155)	0.815(±0.11)	**0.84(±0.15)**	0.876(±0.112)

	Deep-Coral								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.674(±0.012)	0.674(±0.012)	0.674(±0.012)	0.667(±0.0)	*0.681(±0.012)*	0.674(±0.012)	0.674(±0.012)	**0.688(±0.0)**	0.694(±0.024)
2 → 11	0.948(±0.09)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.875(±0.094)	0.906(±0.162)	0.906(±0.162)	**1.0(±0.0)**	1.0(±0.0)
6 → 23	0.875(±0.042)	0.896(±0.0)	0.896(±0.0)	0.896(±0.0)	*0.903(±0.012)*	0.875(±0.042)	0.875(±0.042)	**0.896(±0.0)**	0.896(±0.012)
7 → 13	0.847(±0.073)	*0.91(±0.032)*	0.91(±0.032)	0.91(±0.032)	0.903(±0.043)	0.896(±0.036)	0.896(±0.036)	**0.903(±0.024)**	0.924(±0.024)
9 → 18	0.396(±0.036)	0.528(±0.012)	0.528(±0.012)	0.5(±0.042)	*0.549(±0.194)*	0.403(±0.032)	0.403(±0.032)	**0.486(±0.043)**	0.604(±0.062)
Avg.	0.748(±0.21)	*0.801(±0.181)*	0.801(±0.181)	0.794(±0.191)	0.782(±0.17)	0.751(±0.211)	0.751(±0.211)	**0.794(±0.192)**	0.824(±0.149)

	CDAN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.729(±0.021)	0.694(±0.024)	0.694(±0.024)	0.694(±0.024)	*0.708(±0.021)*	**0.729(±0.021)**	0.729(±0.021)	0.708(±0.021)	0.729(±0.021)
2 → 11	0.979(±0.036)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.792(±0.118)	0.781(±0.136)	0.656(±0.205)	**1.0(±0.0)**	1.0(±0.0)
6 → 23	0.708(±0.11)	0.924(±0.024)	0.924(±0.024)	*0.938(±0.0)*	0.882(±0.024)	0.708(±0.11)	0.764(±0.237)	**0.938(±0.0)**	0.924(±0.0)
7 → 13	0.889(±0.024)	*0.958(±0.0)*	0.958(±0.0)	0.958(±0.0)	0.958(±0.0)	0.91(±0.032)	0.924(±0.043)	**0.951(±0.012)**	0.965(±0.032)
9 → 18	0.431(±0.103)	0.604(±0.021)	0.604(±0.021)	0.597(±0.024)	*0.674(±0.052)*	0.597(±0.173)	0.597(±0.173)	**0.653(±0.079)**	0.681(±0.012)
Avg.	0.747(±0.203)	0.836(±0.163)	0.836(±0.163)	*0.838(±0.167)*	0.803(±0.121)	0.745(±0.141)	0.734(±0.178)	**0.85(±0.149)**	0.86(±0.13)

	DANN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.701(±0.032)	0.667(±0.021)	0.667(±0.021)	0.653(±0.043)	*0.694(±0.032)*	**0.701(±0.032)**	0.701(±0.032)	0.688(±0.0)	0.708(±0.062)
2 → 11	0.958(±0.048)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.927(±0.072)	0.948(±0.09)	0.948(±0.09)	**1.0(±0.0)**	1.0(±0.0)
6 → 23	0.861(±0.032)	*0.91(±0.032)*	0.91(±0.032)	0.91(±0.032)	0.896(±0.055)	0.896(±0.042)	0.931(±0.032)	**0.938(±0.0)**	0.944(±0.012)
7 → 13	0.819(±0.115)	*0.951(±0.012)*	0.951(±0.012)	0.951(±0.012)	0.951(±0.012)	0.861(±0.132)	0.861(±0.132)	**0.944(±0.012)**	0.951(±0.012)
9 → 18	0.382(±0.12)	0.597(±0.067)	0.597(±0.067)	*0.625(±0.021)*	0.556(±0.148)	0.382(±0.12)	0.507(±0.084)	**0.632(±0.032)**	0.632(±0.048)
Avg.	0.744(±0.217)	0.825(±0.17)	0.825(±0.17)	*0.828(±0.164)*	0.805(±0.173)	0.758(±0.226)	0.79(±0.186)	**0.84(±0.156)**	0.847(±0.148)

	DSAN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
12 → 16	0.681(±0.012)	0.681(±0.012)	0.681(±0.012)	0.688(±0.042)	*0.701(±0.024)*	0.681(±0.012)	0.681(±0.012)	**0.701(±0.024)**	0.715(±0.024)
2 → 11	1.0(±0.0)	*1.0(±0.0)*	1.0(±0.0)	1.0(±0.0)	0.906(±0.031)	**1.0(±0.0)**	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)
6 → 23	0.847(±0.043)	*0.938(±0.0)*	0.938(±0.0)	0.938(±0.0)	0.854(±0.042)	0.847(±0.043)	0.847(±0.043)	**0.938(±0.0)**	0.958(±0.036)
7 → 13	0.854(±0.036)	*0.951(±0.012)*	0.951(±0.012)	0.951(±0.012)	0.875(±0.036)	0.854(±0.036)	0.639(±0.392)	**0.958(±0.0)**	0.958(±0.0)
9 → 18	0.542(±0.021)	0.625(±0.021)	0.625(±0.021)	*0.639(±0.032)*	0.528(±0.194)	0.681(±0.043)	0.465(±0.331)	**0.75(±0.021)**	0.924(±0.073)
Avg.	0.785(±0.165)	0.839(±0.16)	0.839(±0.16)	*0.843(±0.156)*	0.773(±0.166)	0.812(±0.128)	0.726(±0.272)	**0.869(±0.125)**	0.911(±0.101)

Table 15: Mean and standard deviation (after ±) of target classification error on HHAR (Part 1) over 3 repetitions with different random initializations of model weights.

	HoMM								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.675(±0.031)	0.724(±0.023)	0.724(±0.023)	*0.728(±0.015)*	0.696(±0.085)	0.722(±0.024)	0.718(±0.017)	**0.726(±0.01)**	0.747(±0.01)
1 → 6	0.758(±0.102)	0.885(±0.002)	0.885(±0.002)	*0.89(±0.006)*	0.826(±0.101)	0.803(±0.114)	0.803(±0.114)	**0.886(±0.002)**	0.903(±0.023)
2 → 7	0.519(±0.113)	0.461(±0.01)	0.461(±0.01)	0.46(±0.008)	*0.49(±0.081)*	0.497(±0.071)	**0.5(±0.066)**	0.461(±0.007)	0.524(±0.068)
3 → 8	0.78(±0.012)	*0.816(±0.007)*	0.816(±0.007)	0.81(±0.012)	0.811(±0.022)	0.793(±0.028)	0.781(±0.01)	**0.815(±0.005)**	0.824(±0.022)
4 → 5	0.861(±0.036)	*0.904(±0.022)*	0.904(±0.022)	0.902(±0.021)	0.806(±0.064)	0.879(±0.032)	0.809(±0.054)	**0.91(±0.008)**	0.941(±0.027)
Avg.	0.719(±0.134)	0.758(±0.167)	0.758(±0.167)	*0.758(±0.168)*	0.726(±0.146)	0.739(±0.146)	0.722(±0.131)	**0.76(±0.168)**	0.788(±0.148)

	AdvSKM								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.676(±0.005)	0.719(±0.002)	0.719(±0.002)	*0.725(±0.015)*	0.696(±0.004)	0.675(±0.011)	0.672(±0.006)	**0.714(±0.005)**	0.732(±0.021)
1 → 6	0.888(±0.021)	0.869(±0.009)	0.869(±0.009)	*0.871(±0.008)*	0.747(±0.064)	0.862(±0.062)	**0.879(±0.034)**	0.867(±0.007)	0.888(±0.021)
2 → 7	0.574(±0.018)	*0.527(±0.065)*	0.527(±0.065)	0.527(±0.062)	0.506(±0.067)	**0.554(±0.043)**	0.44(±0.069)	0.533(±0.063)	0.58(±0.087)
3 → 8	0.81(±0.005)	*0.811(±0.005)*	0.811(±0.005)	0.805(±0.007)	0.801(±0.021)	0.81(±0.005)	0.807(±0.005)	**0.811(±0.006)**	0.81(±0.006)
4 → 5	0.822(±0.049)	*0.876(±0.006)*	0.876(±0.006)	0.875(±0.008)	0.836(±0.027)	0.839(±0.011)	0.839(±0.011)	**0.882(±0.005)**	0.882(±0.034)
Avg.	0.754(±0.119)	*0.761(±0.137)*	0.761(±0.137)	0.76(±0.136)	0.717(±0.126)	0.748(±0.124)	0.728(±0.168)	**0.761(±0.135)**	0.778(±0.114)

	DIRT								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.728(±0.013)	0.583(±0.081)	0.583(±0.081)	*0.61(±0.103)*	0.553(±0.083)	**0.646(±0.138)**	0.646(±0.138)	0.622(±0.021)	0.728(±0.013)
1 → 6	0.826(±0.046)	0.932(±0.006)	0.932(±0.006)	0.939(±0.002)	*0.944(±0.009)*	0.826(±0.046)	0.876(±0.043)	**0.938(±0.0)**	0.946(±0.011)
2 → 7	0.542(±0.071)	0.674(±0.0)	0.674(±0.0)	*0.679(±0.004)*	0.588(±0.123)	0.54(±0.12)	0.552(±0.113)	**0.565(±0.094)**	0.679(±0.0)
3 → 8	0.794(±0.023)	*0.828(±0.004)*	0.828(±0.004)	0.828(±0.004)	0.811(±0.014)	0.81(±0.029)	0.81(±0.029)	**0.829(±0.005)**	0.957(±0.071)
4 → 5	0.831(±0.006)	*0.986(±0.005)*	0.986(±0.005)	0.984(±0.004)	0.917(±0.094)	0.831(±0.006)	0.884(±0.087)	**0.984(±0.0)**	0.987(±0.002)
Avg.	0.744(±0.116)	0.801(±0.16)	0.801(±0.16)	*0.808(±0.155)*	0.763(±0.182)	0.731(±0.141)	0.754(±0.157)	**0.788(±0.177)**	0.859(±0.129)

	DDC								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.594(±0.111)	0.678(±0.021)	0.678(±0.021)	*0.679(±0.022)*	0.642(±0.058)	0.643(±0.06)	0.653(±0.043)	**0.692(±0.015)**	0.662(±0.015)
1 → 6	0.857(±0.066)	*0.921(±0.036)*	0.921(±0.036)	0.921(±0.031)	0.893(±0.048)	**0.917(±0.015)**	0.893(±0.035)	0.904(±0.022)	0.894(±0.036)
2 → 7	0.448(±0.065)	0.478(±0.016)	0.478(±0.016)	0.478(±0.016)	*0.494(±0.031)*	**0.476(±0.023)**	0.476(±0.023)	0.473(±0.019)	0.574(±0.052)
3 → 8	0.776(±0.026)	0.794(±0.009)	0.794(±0.009)	*0.797(±0.008)*	0.784(±0.01)	0.783(±0.036)	0.783(±0.036)	**0.812(±0.012)**	0.822(±0.002)
4 → 5	0.793(±0.037)	*0.891(±0.026)*	0.891(±0.026)	0.882(±0.018)	0.816(±0.027)	0.852(±0.061)	0.852(±0.061)	**0.895(±0.024)**	0.874(±0.024)
Avg.	0.694(±0.166)	*0.752(±0.168)*	0.752(±0.168)	0.751(±0.166)	0.726(±0.15)	0.734(±0.167)	0.731(±0.161)	**0.755(±0.167)**	0.765(±0.125)

	MMDA								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.696(±0.007)	0.736(±0.01)	0.736(±0.01)	*0.739(±0.009)*	0.656(±0.107)	0.719(±0.017)	0.719(±0.017)	**0.746(±0.011)**	0.743(±0.011)
1 → 6	0.629(±0.114)	*0.903(±0.006)*	0.903(±0.006)	0.9(±0.004)	0.628(±0.036)	0.718(±0.151)	0.718(±0.151)	**0.901(±0.005)**	0.894(±0.006)
2 → 7	0.5(±0.103)	0.506(±0.011)	0.506(±0.011)	0.503(±0.013)	*0.557(±0.073)*	**0.5(±0.072)**	0.5(±0.072)	0.493(±0.007)	0.571(±0.093)
3 → 8	0.767(±0.03)	0.824(±0.008)	0.824(±0.008)	0.82(±0.007)	*0.826(±0.13)*	0.766(±0.051)	0.763(±0.048)	**0.822(±0.014)**	0.898(±0.034)
4 → 5	0.84(±0.035)	*0.94(±0.011)*	0.94(±0.011)	0.939(±0.009)	0.842(±0.067)	0.815(±0.055)	0.815(±0.055)	**0.939(±0.013)**	0.939(±0.011)
Avg.	0.686(±0.135)	*0.782(±0.16)*	0.782(±0.16)	0.78(±0.16)	0.702(±0.139)	0.704(±0.132)	0.703(±0.131)	**0.78(±0.164)**	0.809(±0.136)

Table 16: Mean and standard deviation (after ±) of target classification error on HHAR (Part 2) over 3 repetitions with different random initializations of model weights.

	CoDATS								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.667(±0.051)	0.496(±0.004)	0.496(±0.004)	0.494(±0.002)	*0.632(±0.109)*	**0.579(±0.069)**	0.579(±0.069)	0.501(±0.01)	0.667(±0.051)
1 → 6	0.817(±0.107)	*0.944(±0.006)*	0.944(±0.006)	0.944(±0.002)	0.936(±0.01)	0.932(±0.002)	0.936(±0.009)	**0.943(±0.005)**	0.946(±0.0)
2 → 7	0.491(±0.06)	0.472(±0.007)	0.472(±0.007)	0.469(±0.004)	*0.485(±0.071)*	0.458(±0.018)	**0.472(±0.01)**	0.47(±0.003)	0.567(±0.018)
3 → 8	0.786(±0.024)	*0.973(±0.004)*	0.973(±0.004)	0.973(±0.0)	0.96(±0.023)	0.888(±0.106)	0.909(±0.095)	**0.971(±0.002)**	0.982(±0.002)
4 → 5	0.831(±0.02)	0.977(±0.004)	0.977(±0.004)	*0.978(±0.006)*	0.938(±0.014)	0.866(±0.049)	0.866(±0.049)	**0.975(±0.002)**	0.98(±0.004)
Avg.	0.718(±0.142)	0.772(±0.244)	0.772(±0.244)	0.772(±0.246)	*0.79(±0.208)*	0.745(±0.203)	0.752(±0.202)	**0.772(±0.243)**	0.828(±0.176)

	Deep-Coral								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.679(±0.041)	*0.731(±0.024)*	0.731(±0.024)	0.728(±0.009)	0.597(±0.057)	0.657(±0.079)	0.626(±0.052)	**0.737(±0.014)**	0.722(±0.014)
1 → 6	0.812(±0.09)	0.888(±0.011)	0.888(±0.011)	0.889(±0.009)	*0.892(±0.007)*	0.86(±0.035)	0.86(±0.035)	**0.889(±0.006)**	0.91(±0.006)
2 → 7	0.506(±0.095)	*0.475(±0.009)*	0.475(±0.009)	0.464(±0.013)	0.458(±0.041)	0.552(±0.049)	**0.585(±0.009)**	0.49(±0.025)	0.537(±0.009)
3 → 8	0.773(±0.018)	0.809(±0.004)	0.809(±0.004)	*0.81(±0.002)*	0.793(±0.007)	0.785(±0.007)	0.785(±0.007)	**0.811(±0.002)**	0.823(±0.03)
4 → 5	0.861(±0.03)	0.944(±0.02)	0.944(±0.02)	*0.947(±0.016)*	0.819(±0.043)	0.895(±0.032)	0.895(±0.032)	**0.941(±0.017)**	0.953(±0.018)
Avg.	0.726(±0.14)	*0.769(±0.17)*	0.769(±0.17)	0.767(±0.175)	0.712(±0.168)	0.75(±0.138)	0.75(±0.131)	**0.774(±0.164)**	0.789(±0.149)

	CDAN								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.717(±0.012)	0.492(±0.011)	0.492(±0.011)	0.488(±0.011)	*0.607(±0.123)*	0.482(±0.002)	0.482(±0.002)	**0.631(±0.021)**	0.717(±0.012)
1 → 6	0.799(±0.086)	0.938(±0.0)	0.938(±0.0)	*0.94(±0.002)*	0.936(±0.015)	0.942(±0.011)	0.942(±0.011)	**0.943(±0.002)**	0.949(±0.002)
2 → 7	0.494(±0.1)	0.509(±0.051)	0.509(±0.051)	0.507(±0.049)	*0.542(±0.087)*	**0.552(±0.048)**	0.485(±0.016)	0.527(±0.05)	0.592(±0.029)
3 → 8	0.802(±0.013)	0.878(±0.08)	0.878(±0.08)	*0.879(±0.08)*	0.803(±0.018)	0.802(±0.013)	0.802(±0.013)	**0.896(±0.061)**	0.973(±0.004)
4 → 5	0.854(±0.023)	*0.98(±0.0)*	0.98(±0.0)	0.98(±0.0)	0.928(±0.041)	0.854(±0.023)	0.854(±0.023)	**0.979(±0.002)**	0.98(±0.004)
Avg.	0.733(±0.141)	0.759(±0.224)	0.759(±0.224)	0.759(±0.226)	*0.763(±0.179)*	0.726(±0.186)	0.713(±0.2)	**0.795(±0.191)**	0.842(±0.159)

	DANN								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.703(±0.025)	0.49(±0.002)	0.49(±0.002)	0.493(±0.002)	*0.549(±0.08)*	**0.632(±0.128)**	0.599(±0.122)	0.618(±0.021)	0.703(±0.025)
1 → 6	0.803(±0.054)	*0.933(±0.0)*	0.933(±0.0)	0.933(±0.0)	0.821(±0.102)	0.897(±0.063)	0.931(±0.005)	**0.933(±0.004)**	0.938(±0.007)
2 → 7	0.597(±0.014)	0.521(±0.044)	0.521(±0.044)	0.521(±0.044)	*0.571(±0.024)*	0.579(±0.079)	**0.591(±0.086)**	0.494(±0.005)	0.632(±0.014)
3 → 8	0.81(±0.019)	0.977(±0.01)	0.977(±0.01)	*0.979(±0.006)*	0.896(±0.086)	0.803(±0.008)	0.773(±0.044)	**0.966(±0.016)**	0.982(±0.002)
4 → 5	0.84(±0.02)	*0.974(±0.005)*	0.974(±0.005)	0.974(±0.005)	0.866(±0.07)	0.84(±0.02)	0.829(±0.019)	**0.974(±0.002)**	0.977(±0.007)
Avg.	0.75(±0.096)	0.779(±0.232)	0.779(±0.232)	*0.78(±0.232)*	0.741(±0.168)	0.75(±0.142)	0.745(±0.149)	**0.797(±0.208)**	0.846(±0.148)

	DSAN								
		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
0 → 6	0.669(±0.01)	0.519(±0.041)	0.519(±0.041)	0.524(±0.041)	*0.667(±0.004)*	0.674(±0.009)	0.568(±0.091)	**0.763(±0.011)**	0.679(±0.011)
1 → 6	0.86(±0.07)	0.933(±0.008)	0.933(±0.008)	*0.935(±0.002)*	0.933(±0.0)	0.933(±0.015)	0.671(±0.44)	**0.938(±0.0)**	0.938(±0.011)
2 → 7	0.476(±0.016)	0.488(±0.009)	0.488(±0.009)	0.491(±0.004)	*0.537(±0.084)*	0.484(±0.007)	0.491(±0.013)	**0.497(±0.003)**	0.659(±0.291)
3 → 8	0.784(±0.016)	0.979(±0.002)	0.979(±0.002)	*0.982(±0.002)*	0.967(±0.006)	0.921(±0.08)	0.406(±0.487)	**0.974(±0.002)**	0.979(±0.002)
4 → 5	0.88(±0.063)	*0.98(±0.008)*	0.98(±0.008)	0.979(±0.005)	0.947(±0.055)	0.924(±0.071)	0.68(±0.405)	**0.98(±0.0)**	0.98(±0.008)
Avg.	0.734(±0.158)	0.78(±0.235)	0.78(±0.235)	0.782(±0.234)	*0.81(±0.185)*	0.787(±0.191)	0.563(±0.313)	**0.83(±0.191)**	0.847(±0.146)

Table 17: Mean and standard deviation (after $\pm$) of target classification error on WISDM (Part 1) over 3 repetitions with different random initializations of model weights.

HoMM

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.733(±0.0)	0.622(±0.019)	0.622(±0.019)	0.622(±0.038)	*0.689(±0.019)*	**0.733(±0.0)**	0.733(±0.0)	0.656(±0.019)	0.733(±0.0)
20 → 30	0.833(±0.011)	0.788(±0.019)	0.788(±0.019)	0.788(±0.019)	*0.814(±0.044)*	**0.833(±0.011)**	0.833(±0.011)	0.814(±0.022)	0.853(±0.029)
35 → 31	0.579(±0.06)	*0.778(±0.014)*	0.778(±0.014)	0.778(±0.014)	0.579(±0.084)	0.579(±0.06)	0.706(±0.162)	**0.762(±0.0)**	0.802(±0.027)
6 → 19	0.879(±0.026)	*0.874(±0.035)*	0.874(±0.035)	0.798(±0.009)	0.793(±0.087)	**0.879(±0.026)**	0.879(±0.026)	0.859(±0.032)	0.879(±0.026)
7 → 18	0.697(±0.063)	*0.711(±0.024)*	0.711(±0.024)	0.711(±0.027)	0.66(±0.014)	**0.721(±0.014)**	0.704(±0.02)	0.72(±0.024)	0.742(±0.042)
Avg.	0.744(±0.115)	*0.755(±0.089)*	0.755(±0.089)	0.739(±0.071)	0.707(±0.103)	0.749(±0.11)	**0.771(±0.097)**	0.762(±0.076)	0.802(±0.058)

AdvSKM

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.733(±0.033)	0.7(±0.058)	0.7(±0.058)	*0.733(±0.0)*	0.689(±0.069)	0.733(±0.033)	0.733(±0.033)	**0.744(±0.019)**	0.756(±0.019)
20 → 30	0.865(±0.038)	*0.872(±0.048)*	0.872(±0.048)	0.872(±0.048)	0.846(±0.033)	**0.865(±0.038)**	0.865(±0.038)	0.853(±0.044)	0.878(±0.022)
35 → 31	0.603(±0.014)	0.571(±0.071)	0.571(±0.071)	*0.619(±0.109)*	0.611(±0.084)	0.603(±0.014)	0.603(±0.014)	**0.675(±0.069)**	0.69(±0.082)
6 → 19	0.813(±0.101)	0.838(±0.049)	0.838(±0.049)	*0.859(±0.049)*	0.848(±0.055)	0.813(±0.101)	0.813(±0.101)	**0.869(±0.017)**	0.894(±0.0)
7 → 18	0.699(±0.017)	*0.712(±0.024)*	0.712(±0.024)	0.711(±0.016)	0.685(±0.03)	0.693(±0.016)	0.693(±0.016)	**0.72(±0.006)**	0.73(±0.032)
Avg.	0.743(±0.104)	0.739(±0.12)	0.739(±0.12)	*0.759(±0.11)*	0.736(±0.11)	0.742(±0.104)	0.742(±0.104)	**0.772(±0.085)**	0.79(±0.082)

DIRT

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.767(±0.033)	0.744(±0.019)	0.744(±0.019)	0.744(±0.019)	*0.778(±0.051)*	**0.767(±0.033)**	0.767(±0.033)	0.733(±0.033)	0.767(±0.051)
20 → 30	0.872(±0.048)	0.827(±0.033)	0.827(±0.033)	0.827(±0.033)	*0.853(±0.011)*	**0.872(±0.048)**	0.872(±0.048)	0.833(±0.022)	0.872(±0.048)
35 → 31	0.611(±0.073)	0.698(±0.036)	0.698(±0.036)	0.714(±0.041)	*0.722(±0.014)*	0.611(±0.073)	0.619(±0.071)	**0.69(±0.024)**	0.833(±0.126)
6 → 19	0.869(±0.044)	0.833(±0.052)	0.833(±0.052)	0.833(±0.052)	*0.894(±0.0)*	**0.869(±0.044)**	0.869(±0.044)	0.823(±0.035)	0.869(±0.0)
7 → 18	0.78(±0.022)	*0.83(±0.0)*	0.83(±0.0)	0.83(±0.0)	0.761(±0.054)	0.78(±0.022)	0.805(±0.044)	**0.83(±0.0)**	0.83(±0.0)
Avg.	0.78(±0.106)	0.787(±0.064)	0.787(±0.064)	0.79(±0.06)	*0.802(±0.071)*	0.78(±0.106)	**0.786(±0.105)**	0.782(±0.065)	0.834(±0.038)

DDC

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.733(±0.0)	0.7(±0.058)	0.7(±0.058)	*0.711(±0.051)*	0.667(±0.058)	0.733(±0.0)	0.733(±0.0)	**0.744(±0.038)**	0.767(±0.033)
20 → 30	0.846(±0.019)	0.859(±0.011)	0.859(±0.011)	*0.865(±0.0)*	0.814(±0.029)	0.846(±0.019)	0.846(±0.019)	**0.859(±0.022)**	0.885(±0.033)
35 → 31	0.532(±0.06)	0.611(±0.099)	0.611(±0.099)	0.619(±0.086)	*0.675(±0.113)*	0.532(±0.06)	0.532(±0.06)	**0.627(±0.036)**	0.683(±0.05)
6 → 19	0.879(±0.026)	*0.869(±0.044)*	0.869(±0.044)	0.843(±0.044)	0.869(±0.044)	**0.879(±0.026)**	0.879(±0.026)	0.869(±0.023)	0.894(±0.0)
7 → 18	0.723(±0.039)	0.698(±0.05)	0.698(±0.05)	*0.704(±0.039)*	0.616(±0.054)	0.723(±0.039)	0.723(±0.039)	**0.736(±0.019)**	0.742(±0.011)
Avg.	0.743(±0.13)	0.747(±0.115)	0.747(±0.115)	*0.749(±0.105)*	0.728(±0.114)	0.743(±0.13)	0.743(±0.13)	**0.767(±0.096)**	0.794(±0.083)

MMDA

		Heuristic				Theoretical error guarantees			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.733(±0.0)	0.778(±0.038)	0.778(±0.038)	*0.789(±0.019)*	0.767(±0.033)	0.733(±0.0)	0.733(±0.0)	**0.778(±0.019)**	0.856(±0.051)
20 → 30	0.846(±0.019)	*0.846(±0.0)*	0.846(±0.0)	0.846(±0.0)	0.833(±0.078)	0.846(±0.019)	0.846(±0.019)	**0.853(±0.011)**	0.865(±0.051)
35 → 31	0.556(±0.055)	*0.746(±0.027)*	0.746(±0.027)	0.73(±0.014)	0.714(±0.024)	0.556(±0.055)	0.619(±0.095)	**0.73(±0.014)**	0.77(±0.014)
6 → 19	0.879(±0.026)	0.722(±0.032)	0.722(±0.032)	0.758(±0.03)	*0.894(±0.0)*	**0.879(±0.026)**	0.879(±0.026)	0.838(±0.049)	0.879(±0.0)
7 → 18	0.73(±0.066)	*0.61(±0.039)*	0.61(±0.039)	0.591(±0.011)	0.541(±0.218)	**0.73(±0.066)**	0.723(±0.076)	0.654(±0.022)	0.73(±0.066)
Avg.	0.749(±0.122)	0.74(±0.084)	0.74(±0.084)	0.743(±0.089)	*0.75(±0.153)*	0.749(±0.122)	0.76(±0.108)	**0.771(±0.079)**	0.82(±0.059)

Table 18: Mean and standard deviation (after ±) of target classification error on WISDM (Part 2) over 3 repetitions with different random initializations of model weights.

	CoDATS								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.633(±0.1)	*0.689(±0.069)*	0.689(±0.069)	0.689(±0.069)	0.667(±0.067)	0.633(±0.033)	0.633(±0.033)	**0.733(±0.033)**	0.789(±0.038)
20 → 30	0.776(±0.091)	*0.91(±0.04)*	0.91(±0.04)	0.91(±0.04)	0.846(±0.038)	0.776(±0.091)	0.776(±0.091)	**0.929(±0.029)**	0.917(±0.029)
35 → 31	0.587(±0.11)	*0.738(±0.024)*	0.738(±0.024)	0.738(±0.024)	0.579(±0.117)	0.587(±0.11)	0.619(±0.082)	**0.762(±0.041)**	0.754(±0.041)
6 → 19	0.747(±0.083)	0.894(±0.092)	0.894(±0.092)	*0.914(±0.07)*	0.828(±0.075)	0.747(±0.083)	0.747(±0.083)	**0.96(±0.023)**	0.894(±0.023)
7 → 18	0.654(±0.06)	0.734(±0.007)	0.734(±0.007)	*0.74(±0.008)*	0.682(±0.038)	**0.772(±0.035)**	0.772(±0.035)	0.733(±0.009)	0.764(±0.035)
Avg.	0.679(±0.106)	0.793(±0.105)	0.793(±0.105)	*0.798(±0.107)*	0.721(±0.122)	0.703(±0.104)	0.71(±0.093)	**0.823(±0.106)**	0.824(±0.068)

	Deep-Coral								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.667(±0.088)	*0.7(±0.088)*	0.7(±0.088)	0.644(±0.077)	0.633(±0.088)	0.667(±0.088)	0.667(±0.088)	**0.7(±0.033)**	0.744(±0.038)
20 → 30	0.821(±0.022)	*0.885(±0.033)*	0.885(±0.033)	0.885(±0.019)	0.878(±0.029)	0.821(±0.022)	0.821(±0.022)	**0.891(±0.022)**	0.891(±0.04)
35 → 31	0.595(±0.041)	*0.675(±0.027)*	0.675(±0.027)	0.667(±0.041)	0.627(±0.09)	0.595(±0.041)	0.595(±0.041)	**0.683(±0.036)**	0.698(±0.014)
6 → 19	0.692(±0.072)	0.732(±0.075)	0.732(±0.075)	0.732(±0.075)	*0.747(±0.053)*	0.692(±0.072)	0.692(±0.072)	**0.753(±0.044)**	0.869(±0.057)
7 → 18	0.69(±0.062)	*0.71(±0.03)*	0.71(±0.03)	0.704(±0.035)	0.651(±0.029)	**0.715(±0.032)**	0.699(±0.01)	0.704(±0.015)	0.725(±0.016)
Avg.	0.693(±0.092)	*0.74(±0.091)*	0.74(±0.091)	0.726(±0.099)	0.707(±0.113)	0.698(±0.09)	0.695(±0.089)	**0.746(±0.083)**	0.786(±0.079)

	CDAN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.722(±0.019)	0.711(±0.019)	0.711(±0.019)	0.711(±0.019)	*0.733(±0.0)*	**0.722(±0.019)**	0.722(±0.019)	0.7(±0.0)	0.722(±0.0)
20 → 30	0.859(±0.029)	0.801(±0.029)	0.801(±0.029)	0.788(±0.019)	*0.846(±0.051)*	**0.859(±0.029)**	0.859(±0.029)	0.795(±0.022)	0.859(±0.029)
35 → 31	0.548(±0.024)	*0.746(±0.05)*	0.746(±0.05)	0.722(±0.036)	0.683(±0.096)	0.548(±0.024)	0.619(±0.104)	**0.746(±0.027)**	0.754(±0.036)
6 → 19	0.818(±0.0)	*0.894(±0.0)*	0.894(±0.0)	0.864(±0.052)	0.823(±0.023)	0.818(±0.0)	0.818(±0.0)	**0.864(±0.052)**	0.894(±0.015)
7 → 18	0.694(±0.043)	*0.762(±0.004)*	0.762(±0.004)	0.759(±0.015)	0.712(±0.042)	**0.781(±0.026)**	0.78(±0.024)	0.766(±0.004)	0.797(±0.017)
Avg.	0.728(±0.115)	*0.783(±0.069)*	0.783(±0.069)	0.769(±0.063)	0.759(±0.08)	0.746(±0.114)	0.76(±0.096)	**0.774(±0.061)**	0.805(±0.064)

	DANN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.711(±0.077)	*0.733(±0.033)*	0.733(±0.033)	0.733(±0.033)	0.667(±0.033)	0.678(±0.019)	0.678(±0.019)	**0.767(±0.0)**	0.778(±0.019)
20 → 30	0.865(±0.038)	0.846(±0.033)	0.846(±0.033)	0.846(±0.033)	*0.885(±0.051)*	**0.865(±0.038)**	0.865(±0.038)	0.833(±0.04)	0.878(±0.051)
35 → 31	0.627(±0.09)	*0.762(±0.024)*	0.762(±0.024)	0.762(±0.024)	0.73(±0.077)	0.627(±0.09)	0.667(±0.133)	**0.762(±0.024)**	0.786(±0.048)
6 → 19	0.848(±0.04)	*0.919(±0.017)*	0.919(±0.017)	0.909(±0.03)	0.869(±0.091)	0.879(±0.015)	0.879(±0.015)	**0.934(±0.023)**	0.899(±0.023)
7 → 18	0.679(±0.086)	*0.604(±0.033)*	0.604(±0.033)	0.597(±0.039)	0.509(±0.164)	**0.679(±0.086)**	0.491(±0.154)	0.566(±0.05)	0.679(±0.086)
Avg.	0.746(±0.114)	*0.773(±0.113)*	0.773(±0.113)	0.77(±0.113)	0.732(±0.164)	0.746(±0.12)	0.716(±0.169)	**0.772(±0.128)**	0.804(±0.079)

	DSAN								
		Heuristic				**Theoretical error guarantees**			
Task	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
18 → 23	0.733(±0.0)	0.733(±0.033)	0.733(±0.033)	*0.756(±0.019)*	0.722(±0.019)	0.733(±0.0)	0.733(±0.0)	**0.756(±0.038)**	0.789(±0.019)
20 → 30	0.846(±0.019)	*0.865(±0.033)*	0.865(±0.033)	0.865(±0.033)	0.833(±0.022)	0.846(±0.019)	0.833(±0.029)	**0.865(±0.051)**	0.853(±0.051)
35 → 31	0.556(±0.055)	*0.794(±0.06)*	0.794(±0.06)	0.746(±0.036)	0.659(±0.117)	0.556(±0.055)	0.619(±0.109)	**0.754(±0.107)**	0.786(±0.06)
6 → 19	0.879(±0.026)	0.778(±0.017)	0.778(±0.017)	0.788(±0.0)	*0.889(±0.023)*	**0.879(±0.026)**	0.717(±0.268)	0.717(±0.097)	0.879(±0.023)
7 → 18	0.73(±0.066)	0.421(±0.079)	0.421(±0.079)	0.352(±0.022)	*0.667(±0.154)*	**0.73(±0.066)**	0.66(±0.182)	0.352(±0.011)	0.73(±0.066)
Avg.	0.749(±0.122)	0.718(±0.165)	0.718(±0.165)	0.701(±0.187)	*0.754(±0.121)*	**0.749(±0.122)**	0.713(±0.15)	0.689(±0.191)	0.807(±0.053)

Acknowledgments

The ELLIS Unit Linz, the LIT AI Lab, and the Institute for Machine Learning are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (LIT-2018-6-YOU-212), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), AIRI FG 9-N (FWF-36284, FWF-36235), and ELISE (H2020-ICT-2019-3 ID: 951847). We further thank Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, Software Competence Center Hagenberg GmbH, TÜV Austria, Frauscher Sensonic, and the NVIDIA Corporation. The research reported in this paper has been funded by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Federal Ministry for Digital and Economic Affairs (BMDW), and the Province of Upper Austria in the frame of the COMET–Competence Centers for Excellent Technologies Programme and the COMET Module S3AI managed by the Austrian Research Promotion Agency FFG.

References

[1] R. M. Dudley. *Real analysis and probability*, volume 74. Cambridge University Press, 2002.

[2] G. Teschl. Topics in Linear and Nonlinear Functional Analysis. *Amer. Math. Soc., Providence, to appear*, 2022.

[3] G. Teschl. Topics in Real Analysis. *Amer. Math. Soc., Providence, to appear*, 2022.

[4] A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

[5] A. Caponnetto and E. De Vito. Risk bounds for regularized least-squares algorithm with operatorvalued kernels. Technical report, CBCL paper 249/CSAIL-TR-2005-031, MIT, 2005.

[6] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19, 2006.

[7] I. Pinelis. An approach to inequalities for the distributions of infinite-dimensional martingales. In *Probability in Banach Spaces, 8: Proceedings of the Eighth International Conference*, pages 128–134. Springer, 1992.

[8] L. Rosasco, M. Belkin, and E. De Vito. On learning with integral operators. *Journal of Machine Learning Research*, 11(2), 2010.

[9] E. R. Gizewski, L. Mayer, B. A. Moser, D. H. Nguyen, S. Pereverzyev Jr, S. V. Pereverzyev, N. Shepeleva, and W. Zellinger. On a regularization of unsupervised domain adaptation in RKHS. *Applied and Computational Harmonic Analysis*, 57:201–227, 2022.

[10] S. V. Pereverzyev. *An Introduction to Artificial Intelligence based on Reproducing Kernel Hilbert Spaces*. Birkhäuser Cham, 2022.

[11] C. Ma et al. The barron space and the flow-induced function spaces for neural network models. *Constructive Approximation*, 55(1):369–406, 2022.

[12] A. Fermanian, P. Marion, J. P. Vert, and G. Biau. Framing rnn as a kernel method: A neural ode approach. *Advances in Neural Information Processing Systems*, 34, 2021.

[13] A. Bietti and J. Mairal. Invariance and stability of deep convolutional representations. *Advances in neural information processing systems*, 30, 2017.

[14] A. Bietti and J. Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *The Journal of Machine Learning Research*, 20(1):876–924, 2019.

[15] W. Zellinger, N. Shepeleva, M.-C. Dinu, H. Eghbal-zadeh, H. Nguyen, B. Nessler, S. Pereverzyev, and B. A. Moser. The balancing principle for parameter choice in distance-regularized domain adaptation. *Advances in Neural Information Processing Systems*, 34, 2021.

[16] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128, 2006.

[17] M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *Proceedings of the International Conference on Machine Learning*, pages 767–774, 2012.

[18] C. Louizos, K. Swersky, Y. Li, M. Welling, and R. Zemel. The variational fair auto encoder. *International Conference on Learning Representations*, 2016.

[19] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(Jan):1–35, 2016.

[20] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[21] M. Ragab, E. Eldele, W. L. Tan, C.-S. Foo, Z. Chen, M. Wu, C.-K. Kwoh, and X. Li. Adatime: A benchmarking suite for domain adaptation on time series data. *arXiv preprint arXiv:2203.08321*, 2022.

[22] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. *European Symposium on Artificial Neural Networks*, pages 437–442, 2013.

[23] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *Sigkdd Explorations*, 12(2):74–82, 2011.

[24] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, page 127–140, New York, NY, USA, 2015. Association for Computing Machinery.

[25] E. Eldele, Z. Chen, C. Liu, M. Wu, C.-K. Kwoh, X. Li, and C. Guan. An attention-based deep learning approach for sleep stage classification with single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2021.

[26] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 101(23):215–220, 2000.

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[28] L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[30] C. O. da Costa-Luis. Tqdm: A fast, extensible progress meter for python and cli. *Journal of Open Source Software*, 4(37):1277, 2019.

[31] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[32] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[34] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[35] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[36] B. Sun, J. Feng, and K. Saenko. Correlation alignment for unsupervised domain adaptation. *Domain Adaptation in Computer Vision Applications*, pages 153–171, 2017.

[37] C. Chen, Z. Fu, Z. Chen, S. Jin, Z. Cheng, X. Jin, and X.-S. Hua. Homm: Higher-order moment matching for unsupervised domain adaptation. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

[38] M. M. Rahman, C. Fookes, M. Baktashmotlagh, and S. Sridharan. On minimum discrepancy estimation for deep domain adaptation. *Domain Adaptation for Visual Understanding*, 2020.

[39] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, and Q. He. Deep subdomain adaptation network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 32(4):1713–1722, 2021.

[40] M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[41] R. Shu, H. Bui, H. Narui, and S. Ermon. A dirt-t approach to unsupervised domain adaptation. *International Conference on Learning Representations (ICLR)*, 2018.

[42] G. Wilson, J. R. Doppa, and D. J. Cook. Multi-source deep domain adaptation with weak supervision for time-series sensor data. *Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD)*, 2020.

[43] Q. Liu and H. Xue. Adversarial spectral kernel matching for unsupervised time series domain adaptation. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 30, 2021.

[44] G. Strang. *Linear algebra and its applications.* Orlando, FL, Academic Press, Inc., 1980.

[45] M. Sugiyamai, M. Krauledat, and K. M. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

[46] W. M. Kouw, J. H. Krijthe, and M. Loog. Robust importance-weighted cross-validation under sample selection bias. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2019.