Domain Adaptation for Time-Series Classification to Mitigate Covariate Shift

Felix Ott
Fraunhofer IIS, Fraunhofer Institute for Integrated Circuits
Nürnberg, Germany
LMU Munich
felix.ott@iis.fraunhofer.de

David Rügamer
LMU Munich
Munich, Germany
RWTH Aachen
david.ruegamer@stat.uni-muenchen.de

Bernd Bischl
LMU Munich
Munich, Germany
bernd.bischl@stat.uni-muenchen.de

Lucas Heublein
Fraunhofer IIS, Fraunhofer Institute for Integrated Circuits
Nürnberg, Germany
heublelns@iis.fraunhofer.de

Christopher Mutschler
Fraunhofer IIS, Fraunhofer Institute for Integrated Circuits
Nürnberg, Germany
christopher.mutschler@iis.fraunhofer.de

ABSTRACT
The performance of a machine learning model degrades when it is applied to data from a similar but different domain than the data it has initially been trained on. To mitigate this domain shift problem, domain adaptation (DA) techniques search for an optimal transformation that converts the (current) input data from a source domain to a target domain to learn a domain-invariant representation that reduces domain discrepancy. This paper proposes a novel supervised DA based on two steps. First, we search for an optimal class-dependent transformation from the source to the target domain from a few samples. We consider optimal transport methods such as the earth mover’s distance, Sinkhorn transport and correlation alignment. Second, we use embedding similarity techniques to select the corresponding transformation at inference. We use correlation metrics and higher-order moment matching techniques. We conduct an extensive evaluation on time-series datasets with domain shift including simulated and various online handwriting datasets to demonstrate the performance.

CCS CONCEPTS
• Computing methodologies → Learning under covariate shift: Learning latent representations.

KEYWORDS
Domain adaptation, domain shift, optimal transport, embedding similarity, time-series classification, online handwriting recognition

ACM Reference Format:
Felix Ott, David Rügamer, Lucas Heublein, Bernd Bischl, and Christopher Mutschler. 2022. Domain Adaptation for Time-Series Classification to Mitigate Covariate Shift. In Proceedings of the 30th ACM International Conference on Multimedia (MM ’22), Oct. 10–14, 2022, Lisbon, Portugal. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3503161.3548167

This work is licensed under a Creative Commons Attribution International 4.0 License.

MM ’22, October 10–14, 2022, Lisbon, Portugal © 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9203-7/22/10.
https://doi.org/10.1145/3503161.3548167

Figure 1: Domain adaptation. To compensate the domain shift the source data is transformed into the target data (1st: 2D source and target domain features, before transformation, 2nd: after transformation). 3rd and 4th: their distributions.

1 INTRODUCTION
Traditional machine learning (ML) algorithms assume training and test datasets to be independent and identically distributed (i.i.d.). Hence, supervised ML only works well when the test data comes from the same distribution as the training data. As real-world data often changes over time and space, this assumption rarely holds in practice [75]. Domain adaptation (DA) [26, 32, 40, 51, 76, 81, 88] as a special instance of transfer learning (TL) [20, 63, 64, 74] tries to compensate for this domain shift by transferring knowledge from a source to a target domain, see Figure 1. There are three types of DA: supervised, semi-supervised, and unsupervised DA. The decision which approach to use mainly depends on the number of labels available in the target domain [91]. Most techniques are unsupervised and transform the source data by minimizing the distance to the target data. Typically, the model is (re-)trained on the transformed source domain [75].

Domain shifts appear in many applications such as classification [25, 34, 45, 68, 86, 90, 96], handwriting recognition [46], segmentation and regression for multimedia data [12, 30], for example if the background, shape deformation, or quality are different across domains. DA aims to mitigate this and has successfully been applied for object recognition [31, 54, 88, 89], AI planning [95], reinforcement learning [56] and natural language processing [93] (e.g., the adaptation from English to Spanish documents [47]).

DA is also used for multivariate time-series (MTS) classification and forecasting [8, 15, 37, 43], which is a challenging task as
the extraction of domain-invariant representations is non-trivial. Consider the different time-series of online handwriting from a sensor-enhanced pen in Figure 2 (right: right-handed writers, target domain; left: left-handed writers, source domain) [46, 60, 61]. The shaded areas show the discrepancy between both writers, i.e., different time step lengths and accelerations. In this case, the complex dependency of time steps in the MTS makes it challenging to extract invariant features [8]. Many existing methods [21, 67] employ recurrent neural networks (RNNs) and assume that the conditional distributions of the source and target domain are equal, i.e., \( P_S(y | \phi(x_1, \ldots, x_t)) = P_T(y | \phi(x_1, \ldots, x_T)) \), with the feature transformation mapping \( \phi(t) \) and \( X = \{x_1, x_2, \ldots, x_T\} \in X \) being a set of training samples with \( t \in \{t_S, t_T\} \) time steps [62]. However, this assumption does typically not hold in practice as methods do not generalize across domains without additional efforts.

Classical DA methods range from feature selection [3, 70] (both domains share similarities in the features), distribution adaptation (distributions of both domains are different but share similarities), and subspace learning (a lower-dimensional shared representation). Distribution adaptation methods can be classified into three categories: (1) Marginal distribution adaptation methods assume that the marginal distribution between the domains are different and focus on overall shape alignment. While the most established method is maximum mean discrepancy (MMD) [48], e.g., used in [11, 30, 39, 51–53, 81], many further techniques exist [2, 22, 23, 42, 51, 62]. (2) Conditional distribution adaptation [73, 85] assumes that the conditional distribution is varied between the domains \( P(Y_S | X_S) \neq P(Y_T | X_T) \). (3) Joint distribution adaptation methods [49, 50] minimize the joint distribution distance between the source and target domain. Subspace alignment methods (i.e., SA [26], CORAL [76], GFK [32]) align the source and target domains via principal component analysis with a lower dimensional space determined by the Bregman divergence. Recently, DL methods have become the predominant approach in DA. Existing techniques are, e.g., based on MMD [48, 81], align the second-order statistics (covariances) [77], use the Kullback-Leibler [94] or Jensen-Shannon divergence [41], or are based on the Wasserstein distance [16, 28, 29].

These methods have been broadly applied to visual object recognition and text categorization [2], but rarely to time-series [8].

We propose a DA method that adapts embeddings from a small source domain (with domain shift) to embeddings from a large target domain (main training dataset on which the model has initially been trained). First, we pre-train our model on a large target domain dataset. Next, we train an optimal transformation \( T \) from the source to the target domain for each class with joint distribution adaptation methods, i.e., optimal transport and CORAL, from an adaptation set with few samples. At inference, we extract features with the source domain model, transform features into the target domain for each class, compute the similarity to the target domain to select the best transformation, and classify the transformed embedding with the target domain model. This allows a faster adaptation to new data without the necessity of post-training [35]. We apply this technique to MTS datasets and evaluate the performance of methods for the challenging task of embedding similarity comparison from time-series data [8]. We show performance improvements for synthetically generated univariate sinusoidal data and on multivariate online handwriting (OnHW) datasets from pens with integrated sensors. For the challenging OnHW recognition task of new out-of-distribution writers, we propose a method that can adapt to each writer and outperforms transfer learning approaches.1

The remainder of this paper is organized as follows. Section 2 discusses related work followed by our proposed methodology in Section 3. The experimental setup is described in Section 4 and the results are discussed in Section 5. Section 6 concludes.

2 RELATED WORK

Research for (multivariate) time-series classification is very advanced and ranges from classical convolutional neural networks (CNNs) such as FCN [86] to advanced CNNs [78, 79] such as ResCNN [96], ResNet [86], XResNet [34], XceptionTime [68] and Inception-Time [25]. Spatio-temporal methods [14, 45, 60] became popular with the development of RNNs, i.e., long short-term memories (LSTMs) and multi-dimensional LSTMs. Research for Transformers (e.g., TST [90]) for time-series classification is less advanced. As their goal is to classify a (multivariate) time-series without domain shift (e.g., on the UCR [19] datasets), they do not consider to transform embeddings. All such methods have previously been benchmarked on OnHW datasets, and [60] showed a benefit of small CNNs combined with bidirectional LSTMs (BiLSTMs) on OnHW recognition.

Domain Adaptation for Time-Series Classification. Research on MTS-specific DA is rare. The domain adversarial neural network (DANN) learns domain-invariant features [21] and uses a time window approach to extract temporal information from time-series data for prognostics with LSTMs. Similar, the variational recurrent adversarial deep domain adaptation (VRADA) [67] learns domain-invariant temporal relationships based on a variational RNN (VRNN) [15] for MTS healthcare datasets. However, both of them cannot align the condition distribution well. Sparse associative structure alignment (SASA) [8] exploits the sparse associative structure to mitigate the difficult domain-invariant extraction of time-series data [8]. We show performance improvements for synthetically generated univariate sinusoidal data and on multivariate online handwriting (OnHW) datasets from pens with integrated sensors. For the challenging OnHW recognition task of new out-of-distribution writers, we propose a method that can adapt to each writer and outperforms transfer learning approaches.1

1Datasets and source code available at: www.iais.fraunhofer.de/de/lf/v/dataanalytics/anwproj/schreibtrainer/onhw-dataset.html
domain adaptation (DA) in Section 3.1. We then give an overview of our method that consists of two parts: Optimizing a transformation from source to target domain and selecting the transformation at inference (Section 3.2). An overview of all the introduced notation is additionally given in Appendix A.1.

### 3.1 Notation

**MTS Classification.** An MTS $U = \{u_1, \ldots, u_m\} \in \mathbb{R}^{m \times d}$ is an ordered sequence of $l \in \mathbb{N}$ streams with $u_i = (u_{i,1}, \ldots, u_{i,m}), i \in \{1, \ldots, m\}$, where $m \in \mathbb{N}$ is the length of the time-series. The MTS training set is a subset of the array $\mathcal{U} = \{u_1, \ldots, u_{n_T}\} \in \mathbb{R}^{n_T \times m \times d}$, where $n_T$ is the number of time-series. The aim of MTS classification is to predict an unknown class label $y \in \mathcal{Y}$ for a given MTS. We define the target domain dataset as $\mathcal{U}_T$. Given a smaller adaptation set of a source domain $\mathcal{U}_S$ with MTS $U_S$, the goal of DA is to find an optimal transformation $T$ of the representation of the latent embedding $f(U_S)$ of the source domain to the representation of the latent embedding $f(U_T)$ of the target domain such that the prediction of the unknown class label $y_S$ of the source domain is maximized. $f(U_T) \in \mathbb{R}^{q_T \times w_T}$ and $f(U_S) \in \mathbb{R}^{q_S \times w_S}$ are the latent target and source embeddings of the neural network.

**Domain Adaptation.** A domain $\mathcal{D}$ consists of a feature space $X$ with marginal probability $P(X)$. The task is defined by the label space $\mathcal{Y}$. The joint distribution is $P(X, Y)$ and the conditional distribution is denoted as $P(Y|X)$. When considering MTS classification, there is a source domain $\mathcal{D}_S = (X^s, Y^s|_{i=1}^{N_S})$ of $N_S$ labeled samples of $|Y^s|$ categories, and a target domain $\mathcal{D}_T = (X^t, Y^t|_{i=1}^{N_T})$ of $N_T$ labeled samples of $|Y^t|$ categories. Due to the difference of the two domains, the distributions are assumed to be different: $P(X_S) \neq P(X_T)$ and $P(Y_S|X_S) \neq P(Y_T|X_T)$ (see Figure 1, left) [91]. DA can mitigate the domain shift and improve the classification accuracy in the target domain (see Figure 1, 2nd and 4th).

### 3.2 Method Overview

An overview of our method for MTS classification is given in Figure 3 and in Algorithm 1. We train a convolutional neural network with two BiLSTM layers on the target data, and train a different (but same) model on the (training) source data. Next, we search for the optimal transformation $T$ to transform the feature embeddings of the source data onto the feature embeddings of the target data. As feature embeddings $f(U_S)$ and $f(U_T)$ we choose the output of the last convolutional layer before the two BiLSTMs of size $\mathbb{R}^{50 \times 30}$ and $\mathbb{R}^{19 \times 200}$. We obtain a separate transformation for each class. To find the optimal transformation, we evaluate different DA techniques,
The goal of DA is to minimize the target domain error by bounding the source domain error and the discrepancy between them [4–6]. DA approaches consider the target data for optimizing the source domain model and reduce the discrepancy between them as in Theorem 1:

**Theorem 1 (Zhang, 2021 [91]).** Let $\mathcal{H}$ be a hypothesis space. Given the target domain $\mathcal{D}_T$ and source domain $\mathcal{D}_S$, we have

$$\forall h \in \mathcal{H}, R_T(h) \leq R_S(h) + d_{\mathcal{H}\times\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \beta,$$

(1)

where $R_T(h)$ is the target domain error and $R_S(h)$ is the source domain error. $d_{\mathcal{H}\times\mathcal{H}}$ is the discrepancy distance between $\mathcal{D}_S$ and $\mathcal{D}_T$ w.r.t. $\mathcal{H}$. Then, given the label functions $g_T$ and $g_S$ determined by the domain labels $Y_T$ and $Y_S$, the shared error $\beta$ is

$$\beta = \arg \min_{h \in \mathcal{H}} R_S(h^*, g_S) + R_T(h^*, g_T),$$

(2)

where $h^*$ is the ideal hypothesis.

**Homogeneous Domain Adaptation.** DA models aim to find a minimal discrepancy distance $d_{\mathcal{H}\times\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T)$. For classifying time-series data, we restrict our methods to homogeneous DA, where the feature space is the same ($U_T = U_S$) with the same feature dimensionality ($q_T = q_S$ and $w_T = w_S$) and interpolate the time-series to a pre-defined fixed length. In the following, we provide details on computing the optimal transformation.

**Optimal Transport.** To mitigate the domain shift for our multi-variate time-series application, we assume that the domain drift is due to an unknown, possibly non-linear map of the input space $\mathcal{T}$: $\mathcal{D}_S \rightarrow \mathcal{D}_T$ that preserves the conditional distribution $P_S(y|f(U_S)) = P_T(y|f(U_S))$ such that the label information is preserved [16]. Searching for $T$ in the space of all possible transformations is intractable. Hence, $T$ is chosen such that a transportation cost

$$C(T) = \int_{\mathcal{D}_S} c(f(U), T(f(U))) d\mu(f(U)),$$

(3)

is minimized, where $c: \mathcal{D}_T \times \mathcal{D}_S \rightarrow \mathbb{R}^+$ is a distance function over the metric space $\mathcal{D}$ [16]. The optimal transportation problem is

$$T_0 = \arg \min_T \int_{\mathcal{D}_S} c(f(U), T(f(U))) d\mu(f(U)).$$

(4)

This is also known as the Kantorovitch formulation [44] that allows to search a general coupling $\alpha \in \Theta$ by the transportation plan [72]:

$$\alpha_0 = \arg \min_{\alpha \in \Theta} \int_{\mathcal{D}_T \times \mathcal{D}_S} c(f(U_T), f(U_S)) d\alpha(f(U_T), f(U_S)),$$

(5)

where $\Theta$ is a set of all probabilistic couplings $\Theta \in P(\mathcal{D}_T \times \mathcal{D}_S)$ with marginals $\mu_T$ and $\mu_S$. Then, the Wasserstein distance of order $p$ between $\mu_T$ and $\mu_S$ can be defined as

$$W_p(\mu_T, \mu_S) := \left( \int_{\mathcal{D}_T \times \mathcal{D}_S} d(f(U_T), f(U_S))^p d\alpha(f(U_T), f(U_S)) \right)^{\frac{1}{p}},$$

(6)

where $d$ is a distance metric [16] as the cost function:

$$c(f(U_T), f(U_S)) = d(f(U_T), f(U_S))^p.$$  

(7)

We apply the earth mover’s distance (EMD), the EMD with Laplacian regularization [29], and Sinkhorn transport [1, 17] (with $L_2$ class regularization of 0.5) implemented by the Python Optimal Transport (POT) package [28] between our source and target domain samples. Selecting a proper cost function is crucial for the effectiveness of the adaptation of source to target domain. Typically, the $L_2$-based metric is used, but other cost functions are also possible, e.g., norm-based metrics [83], metrics based on Riemannian distances over a manifold [83], metrics used as a loss function...
We then validate the adapted model using the flipped dataset with noise $b = 0.4$. We first investigate the efficacy of our method to transform synthetically generated datasets; for this, we generate sinusoidal signal data of 200 time steps with different frequencies for 10 classes. We add noise from a continuous uniform distribution $U(a, b)$ with $a = 0.0$ and $b \in B = \{0.0, 0.1, 0.2, \ldots, 1.9\}$ for the target domain dataset (Figure 4a), and train a CNN+BiLSTM for the source domain (training) dataset, we flip the sign of the source domain features. For the source domain (training) dataset, we flip the sign of the target domain features. We compare to HoMM [11] of order 3 and kMMD [53] (which is equivalent to the kernelized HoMM of order 1). We also make use of different CORAL metrics (standard, Stein [13] and Jeff [55]) CORAL based on symmetrized Bregman divergences [33]) for embedding comparisons.

4 EXPERIMENTAL SETUP

We apply our DA technique to two time-series datasets: generated time-series (Section 4.1), and OnHW recognition (Section 4.2).

4.1 Synthetic Time-Series Classification

We first investigate the efficacy of our method to transform synthetically generated univariate time-series data. For this, we generate sinusoidal signal data of 200 time steps with different frequencies for 10 classes. We add noise from a continuous uniform distribution $U(a, b)$ with $a = 0.0$ and $b \in B = \{0.0, 0.1, 0.2, \ldots, 1.9\}$ for the target domain dataset (Figure 4a), and train a CNN+BiLSTM. For the source domain (training) dataset, we flip the sign of the generated time-series and add uniform $U(a, b/2)$ noise (Figure 4b). We then validate the adapted model using the flipped dataset with added uniform noise for $b \in B$ values (Figure 4c). This allows us to evaluate the time-series adaptation for different noise ratios.

4.2 Online Handwriting (OnHW) Recognition

OnHW recognition typically uses time in association with different types of spatio-temporal signals. The data contains information about the displacement of certain input devices [66]. OnHW recognition from sensor-enhanced pens uses data from inertial measurement units to capture the pen movement. The pen in [61] uses two accelerometers, one gyroscope, one magnetometer, 3 axes each, and one force sensor at 100 Hz. One sample of size $m \times l$ represents an MTS of $m$ time steps from $l = 13$ sensor channels. We make use of three character-based datasets: The OnHW-chars [61] dataset contains 31,275 samples of small and capital characters (52 label classes) from 119 right-handed writers. The OnHW-symbols [60] dataset contains 2,326 samples of numbers and symbols (15 label classes) from 27 right-handed writers, and the split OnHW-equations [60] dataset contains 39,643 numbers and symbols from 55 right-handed writers obtained from 10,713 equations. For these datasets, 80/20 train/validation splits are available for writer-dependent (WD) and writer-independent (WI) classification tasks. We define these datasets as our target domain $U_T$. Usually, observations from left-handed writers in OnHW datasets is scarce [46]. As we want OnHW recognition to work equally well for left-handed writers as for right-handed writers, we use the smaller available left-handed datasets as source domain $U_S$ and split it into an adaptation (training) set $\mathcal{U}_{S_t}$ and a validation set $\mathcal{U}_{S_v}$. For an overview, see Appendix A.3.

Figure 5 visualizes the $19 \times 200$ dimensional feature embeddings of the CNN+BiLSTM model for the OnHW-chars and split OnHW-equations datasets. We use the t-SNE method [82] with an initial dimension 3,800, perplexity of 30, an initial momentum of 0.5, and a final momentum of 0.8. Figure 5a and 5b visualizes 26 lower and 26 uppercase character labels of OnHW-chars. Several samples are in clusters associated with a different class and hence wrongly classified. Figure 5c differentiates between all 15 class labels for the split OnHW-equations dataset. Here, we can clearly see that the labels ‘+’ , ‘-‘ and ‘:’ are close in their low-dimensional embedding. Figure 5d shows the low-dimensional embedding of the right-handed target domain $\mathcal{D}_T$ (blue) and the left-handed
source domain $D_S$ (orange). It is notable that the features of both domains have different distributions. After the transformation, the left-handed source domain $D_S^L$ (red) is closer to the target domain.

5 EXPERIMENTAL RESULTS

Hardware and Training Setup. For all experiments we use Nvidia Tesla V100-SXM2 GPUs with 32 GB VRAM equipped with Core Xeon CPUs and 192 GB RAM. We use the vanilla Adam optimizer with a learning rate of $10^{-4}$. We use the character recognition rate (CRR) in % as time-series classification evaluation metric. Details on the architecture are proposed in Appendix A.7.

5.1 Limitations of Transfer Learning (TL)

Fine-tuning an existing model on each source domain can be data- and parameter-inefficient. Under the assumption that the adaptation dataset is large enough, fine-tuning leads to a better performance than adapting the domains by feature-based transfer as the model can overfit on the source data [36]. Hence, we apply different TL techniques and compare to DA techniques. Figure 6 shows our network architecture and different TL techniques. First, we adapt the whole model by fine-tuning on each source domain dataset from the pre-trained network on the target domain dataset. Second, we freeze the first layers and only post-train the spatio-temporal layers (two BiLSTMs and two dense layers). This layer freezing leads to a faster training than full post-training. Third, we apply a structurally similar method to the Adapter by Houlsby et al. [35]: We freeze all previously pre-trained layers, add an adaptation layer (a standard dense layer) at the end, and only train the additional layer. The dense layer has $|Y|$ units. This yields a compact and fast trainable model by adding only a few trainable parameters per task.

Table 1: Evaluation results (CRR in %, mean and standard deviation) for transfer learning techniques on the OnHW-symbols and split OnHW-equations datasets [60] (averaged over four left-handed writers) and on the OnHW-chars [61] dataset (averaged over nine left-handed writers) based on the CNN+BiLSTM architecture.

| Method                                      | OnHW-symbols [60] | split OnHW-equations [60] | OnHW-chars [61] |
|---------------------------------------------|-------------------|---------------------------|-----------------|
| Baseline (U$S_s$ in Target Model)           | 19.18             | 33.52                     | 43.80           |
| Without Transformation                      | 36.92 $\pm$ 8.71  | 47.06 $\pm$ 27.13         | 3.60 $\pm$ 2.35 |
| Post-Training (full)                        | 27.49 $\pm$ 14.29 | 35.85 $\pm$ 30.33         | 31.15 $\pm$ 30.20 |
| Post-Training (middle)                      | 19.23 $\pm$ 16.00 | 28.39 $\pm$ 23.54         | 21.42 $\pm$ 23.92 |
| Layer Adapting                              | 22.15 $\pm$ 20.47 | 78.87 $\pm$ 10.89         | 10.75 $\pm$ 14.99 |
| Post-Training (last)                        | 6.87 $\pm$ 2.45   | 21.53 $\pm$ 4.60          | 19.23 $\pm$ 4.87 |
| Combined                                    | 78.87 $\pm$ 10.89 | 10.75 $\pm$ 14.99         | 19.88 $\pm$ 14.20 |

Figure 6: Network architecture with transfer learning by freezing the first layers and fine-tuning the last layers, or only post-training an additional adaptation layer. The network is pre-trained on the target domain U$T_t$, and adapted on the source domain training set U$S_s$ and validated on the set U$S_s$.$^*$.

Figure 7: Evaluation of transfer learning on the generated dataset for noise parameters between $b = 0.0$ and $b = 1.9$.

Sinusoidal Dataset Evaluation. We adapt the pre-trained models (on the source domain datasets) with the target domain datasets by post-training the models provided in Figure 6. Figure 7 shows the results averaged over 10 trainings for all noise parameters $b \in B$. Post-training the full model and freezing the first layers while post-training the last layers yields the highest TL results. Post-training only the last layer or adapting an additional layer results in low classification accuracies below 20%. In general, TL performs poorly when we see significant changes between the domains (i.e., $b > 0.5$). In particular, all results are lower than the results achieved by our DA techniques (see Section 5.2, Figure 8).

OnHW Recognition Evaluation. Table 1 shows TL results for all OnHW datasets. Without transformation, the models fail in the classification tasks, which proves the existence of a domain shift in the data [46]. As the OnHW-symbols dataset is rather small, all TL techniques cannot adapt to a specific writer. The larger OnHW-chars dataset leads to better results. TL only yields good results on the split OnHW-equations dataset, while full model post-training outperforms training only specific layers or adapting an additional layer. Again, TL is limited in its efficacy while our DA approach shows promising results (see Section 5.3).

5.2 Evaluation of Sinusoidal Data

We train each sinusoidal dataset 10 times, and present results of mean and standard deviation. As a gold standard we apply the transformation based on the known label class, which we define as the upper bound for transformation selection, and define the lower bound by the classification without the use of any transformation.

Figure 8 presents results for the five optimal transport techniques. The classification accuracy notably drops for higher noise rates ($b > 0.7$). Without transformation (lower bound) the accuracy is below 20% (see Figure 8a, cyan). The upper bound (black line) yields an accuracy between 80% and 100%. MMD is outperformed...
Figure 8: Results for optimal transport methods and transformation selection on the synthetic dataset (for $b$ between 0.0 and 1.9). For better readability we depict the approach without transformation only in (a) as it is independent of optimal transport.

Table 2: Evaluation results (CRR in %) for the left- and right-handed writer OnHW-symbols and split OnHW-equations datasets [60] for different MTS classification techniques. $\mathcal{U}_r$ are right-handed writer datasets, and $\mathcal{U}_l$ are left-handed writer datasets.

| Method                    | OnHW-symbols [60] | split OnHW-equations [60] |
|---------------------------|-------------------|--------------------------|
|                           | Right-handed      | Left-handed              | L in R | Right-handed | Left-handed | L in R | WI   | WD  | WI  | WD  | WI  |
| CNN+BiLSTM [60]           | 96.20             | 79.51                    | 92.00  | 54.00        | 19.18       | 95.70  | 83.88 | 92.00 | 51.50 | 33.52 |
| LSTM-FCN [45]             | 92.39             | 73.32                    | 75.34  | 41.40        | -            | 93.95  | 81.47 | 88.56 | 47.56 | -    |
| ResCNN ($nf = 64$) [96]   | 92.23             | 77.41                    | 80.82  | 47.87        | -            | 94.58  | 80.95 | 89.39 | 40.45 | -    |
| ResNet ($nf = 64$) [86]   | 94.50             | 77.41                    | 80.82  | 47.87        | -            | 94.68  | 83.45 | 89.20 | 39.21 | -    |
| XResNet50 [34]            | 95.66             | 74.47                    | 78.08  | 47.87        | -            | 94.63  | 81.74 | 89.67 | 45.15 | -    |
| XceptionTime ($nf = 16$) [68] | 91.54          | 72.34                    | 75.34  | 40.43        | -            | 94.03  | 82.24 | 88.72 | 50.73 | -    |
| Inception Time ($nf = 64$, depth = 12) [25] | 91.97          | 76.92                    | 80.82  | 46.81        | -            | 94.87  | 84.35 | 88.48 | 44.15 | -    |
| TST [90]                  | 91.12             | 71.85                    | 78.08  | 51.06        | -            | 93.07  | 80.40 | 87.61 | 47.27 | -    |

Table 3: Evaluation results (CRR in %) for the left-handed ($\mathcal{U}_l$) and right-handed ($\mathcal{U}_r$) writer OnHW-chars [61] datasets for different time-series classification techniques with same parameters as in Table 2.

| Method                    | Right-handed ($\mathcal{U}_r$) | Left-handed ($\mathcal{U}_l$) |
|---------------------------|---------------------------------|--------------------------------|
|                           | Lower Combined                   | Lower Combined |
|                           | Upper Combined                   | Upper Combined |
|                           | WD  | WI  | WD  | WI  | WD  | WI  | WD  | WI  | WD  | WI  | WD  | WI  |
| CNN+BiLSTM [60]           | 88.85 | 79.48 | 92.12 | 85.60 | 78.17 | 68.06 | 94.70 | 43.60 | 91.90 | 43.62 | 82.80 | 32.00 | 45.80 | 45.97 | 25.19 |
| LSTM-FCN [45]             | 81.43 | 71.41 | 85.43 | 77.07 | 67.34 | 57.93 | 70.55 | 34.06 | 72.50 | 29.79 | 61.02 | 22.68 | -    | -    | -    |
| ResCNN ($nf = 64$) [96]   | 82.52 | 72.00 | 86.91 | 78.64 | 67.55 | 58.67 | 80.00 | 38.78 | 80.63 | 29.79 | 65.39 | 26.21 | -    | -    | -    |
| ResNet ($nf = 64$) [86]   | 83.01 | 71.93 | 86.41 | 78.03 | 68.56 | 58.74 | 81.01 | 40.24 | 82.95 | 30.12 | 66.95 | 26.17 | -    | -    | -    |
| XResNet50 [34]            | 80.99 | 69.14 | 86.05 | 76.69 | 64.98 | 54.38 | 74.86 | 31.24 | 76.43 | 28.35 | 60.80 | 18.38 | -    | -    | -    |
| XceptionTime ($nf = 16$)  [68] | 81.41 | 70.76 | 85.94 | 78.23 | 66.70 | 56.92 | 75.41 | 40.08 | 79.20 | 30.66 | 63.92 | 25.91 | -    | -    | -    |
| InceptionTime (64, 12) [25] | 84.14 | 75.28 | 87.80 | 81.62 | 70.43 | 61.68 | 79.08 | 43.12 | 81.25 | 36.48 | 65.12 | 29.35 | -    | -    | -    |
| TST [90]                  | 80.10 | 70.75 | 84.81 | 78.34 | 66.12 | 57.56 | 77.43 | 41.27 | 79.11 | 29.86 | 63.39 | 26.83 | -    | -    | -    |

by all transformation selection techniques as order 1 is not suitable. For the remaining techniques, the Laplacian regularization improves EMD results, while regularizing Sinkhorn ($L_pL_1$ and $L_1L_2$) does not yield better results compared to standard Sinkhorn. Consistently, CC and kMMD outperform CORAL which yields higher accuracies than HoMM of order three and Pearson correlation. For the evaluation of feature embeddings, we refer to Appendix A.4.

5.3 Evaluation of OnHW Recognition

Baseline Results. We train all three OnHW right- and left-handed writer datasets for writer-dependent (WD) and writer-independent (WI) tasks. Architectures are taken from [60] [CNN+ BiLSTM] and the tsai toolbox [57] (for all other models) [25, 34, 45, 68, 86, 90, 96]. Results of these comparisons are given in Tables 2 and 3. For the OnHW-symbols dataset, the CNN+BiLSTM model outperforms all architectures, while for the split OnHW-equations dataset, InceptionTime [25] outperforms the CNN+BiLSTM architecture on the right-handed WI task. On the OnHW-chars dataset, the accuracy of the CNN+BiLSTM model is notably higher compared to all other models, and we hence choose the CNN+BiLSTM model for further experiments. Simply classifying the left-handed writer samples with the model pre-trained on right-handed writer data yields low accuracies (19.18% on the OnHW-symbols dataset and 33.52% on the split OnHW-equations dataset) as left-handed writer samples are out-of-distribution with respect to the right-handed ones (see Figure 5d). Hence, an efficient DA technique is necessary.

Evaluation of Domain Adaptation. We first evaluate different distance metrics for optimal transport. The respective hyperparameter search results are given in Appendix A.5. Based on these findings, we choose the squared Euclidean metric for all further applications. Table 4 summarizes all results for DA pre-trained on the target domain data $\mathcal{U}_r$, adapted on the source domain data $\mathcal{U}_s$, and validated on $\mathcal{U}_s$. We train each left-handed writer separately and report average results with their respective standard deviations.
deviation. For an evaluation for each left-handed writer, see Appendix A.6. The last column in Table 4 shows the upper bound using the known transformation. EMD [27] and SEMD [29] perform similar, as well as Sinkhorn [16] without and with \((L_{p1}, L_{L2})\) regularization. Sinkhorn transport consistently outperforms EMD (see also [1]). The model with CORAL [75] for transformation computation fails to classify the time-series data, even with known transformation selection. The MMD [48] approach (of order 1) yields the lowest classification accuracy, and is notably be improved with CORAL [75] (of order 2). Increasing the order to 3 (HoMM [11]), decreases the accuracy as a higher number of iterations is required. The kernalized MMD (kMMD) [53] approach (of order 1) yields the highest classification accuracies. Kernalizing HoMM leads to extremely long runtimes. Jeff and Stein CORAL outperform the standard CORAL method as these are not dependent on its inverse [33]. Interestingly, CC performs better than the scale-invariant PC.

Runtimes. We demonstrate runtimes for transformation computation and selection methods exemplary on the OnHW-symbols dataset for one writer averaged over all samples. To find the optimal transformation, EMD (0.0033s) and Sinkhorn (0.0042s) are the fastest methods, while the regularization increases the computation time (SEMD: 1.8756s, Sinkhorn \(L_{p1}: 0.025, L_{L2}: 0.0526s\), CORAL leads to extremely long runtimes of 405.7s. Applying the transformation gives the following runtimes in ascending order:

| Dataset | Method | CC [65] | PC [65] | MMD [48] | kMMD [53] | HoMM [11] | CORAL | CORAL (J) | CORAL (S) | w/ T. |
|---------|--------|---------|---------|----------|-----------|-----------|--------|-----------|-----------|------|
| OnHW-  | EMD [27] | 70.03±4.8 | 59.14±13.4 | 61.55±8.2 | 85.02±6.9 | 67.72±8.3 | 75.29±16.0 | 82.24±7.1 | 80.85±8.4 | 89.18±10.8 |
| symbols | SEMD [29] | 70.03±8.4 | 59.14±13.4 | 61.55±18.2 | 85.02±6.9 | 67.72±8.3 | 75.29±16.0 | 82.24±7.1 | 80.85±8.4 | 89.18±10.8 |
|        | Sinkhorn [16] | 64.62±19.2 | 51.97±5.2 | 67.25±8.1 | 85.09±7.7 | 70.05±7.4 | 78.14±6.5 | 80.92±8.0 | 82.31±7.8 | 93.35±8.6 |
|        | Sinkhorn (\(L_{p1}\)) | 64.62±19.2 | 51.97±5.2 | 67.25±8.1 | 85.09±7.7 | 70.05±7.4 | 78.14±6.5 | 80.92±8.0 | 82.31±7.8 | 93.35±8.6 |
|        | Sinkhorn (\(L_{L2}\)) | 64.62±19.2 | 51.97±5.2 | 67.25±8.1 | 85.09±7.7 | 68.64±7.3 | 78.14±6.5 | 80.92±8.0 | 82.31±7.8 | 93.35±8.6 |
|        | CORAL [75] | 5.84±6.8 | 2.78±2.8 | 5.49±0.1 | 6.80±2.2 | 5.49±0.1 | 9.66±7.3 | 5.49±0.1 | 4.17±2.4 | 39.77±10.0 |

Table 4: Evaluation results (CRR in %, mean and standard deviation) for different transformation techniques and transformation selection methods for all OnHW [60, 61] datasets based on the CNN+BiLSTM architecture. \(J = \) Jeff, \(S = \) Stein.

6 CONCLUSION

We addressed DA for time-series classification by combining a large variety of DA techniques with transformation selection methods. We used optimal transport and correlation alignment techniques to transform features of a source domain into features of a target domain. At inference, we compared correlation metrics and methods based on MMD as embedding distance metrics to select the optimal transformation. A broad study on synthetic univariate time-series data and MTS OnHW datasets showed that Sinkhorn transport can outperform EMD and CORAL. The kernalized MMD metric yields the highest classification accuracies. Our DA approach yields higher accuracies than transfer learning on small adaptation datasets.

ACKNOWLEDGMENTS

Supported by the Federal Ministry of Education and Research (BMBF) of Germany by Grant No. 01IS18036A (David Rügamer) and by the research program Human-Computer-Interaction through the project "Schreibttrainer", Grant No. 16SV8228, as well as by the Bavarian Ministry for Economic Affairs, Infrastructure, Transport
and Technology through the Center for Analytics-Data-Applications (ADA-Center) within the framework of “BAYERN DIGITAL II”.

REFERENCES

[1] Jason Altschuler, Jonathan Weed, and Philippe Rigollet. 2017. Near-Linear Time Approximation Algorithms for Optimal Transport via Sinkhorn Iterations. In Advances in Neural Information Processing Systems (NIPS). 1961–1971.

[2] Mahsa Baktashmotlagh, Mehtarsa Harandi, and Mathieu Salzmann. 2016. Distribution-Matching Embedding for Visual Domain Adaptation. In Journal of Machine Learning Research (JMLR). Vol. 17. 1–10.

[3] Herbert Bat, Timne Tuytelaars, and Luc Van Gool. 2006. SURF: Speeded Up Robust Features. In European Conf. on Computer Vision (ECCV). Vol. 3951. 404–417. https://doi.org/10.1007/11744029_32.

[4] Shai Ben-David, John Blitzer, Koby Crammer, Alex kulesza, Fernando Pereira, and Jennifer Wortmann Vaughan. 2009. A Theory of Learning from Different Domains. In Machine Learning, Vol. 79. 151–175.

[5] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of Representations for Domain Adaptation. In Advances in Neural Information Processing Systems (NIPS).

[6] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2007. Learning Bounds for Domain Adaptation. In Advances in Neural Information Processing Systems (NIPS).

[7] Jian-Feng Cai, Emmanuel J. Candés, and Zuowei Shen. 2008. A Singular Value Thresholding Algorithm for Matrix Completion. In arXiv:0810.3286.

[8] Ruichu Cai, Jiawei Chen, Zijian Li, Wei Chen, Keli Zhang, Junjian Ye, Zhuozhang Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. 2019. LSTM Fully Convolutional Networks for Time-Series Classification to Mitigate Covariate Shift. In Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL). Prague, Czech Republic, 256–263.

[9] James Chapman and Hao-Ting Wang. 2021. CCA-Zoo: A Collection of Regularized Alignment and Discriminative Feature Learning for Unsupervised Deep Domain Adaptation. In Advances in Neural Information Processing Systems (NIPS). Vol. 34(3). 465–479. https://doi.org/10.5555/3447595.3449499.

[10] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. 2013. Gated Recurrent Neural Networks. In Intl. Conf. on Computer Vision and Pattern Recognition (CVPR). Providence, RI, 2066–2073.

[11] Boqing Gong, Kristen Grauman, and Fei Sha. 2014. Learning Kernels for Unsupervised Domain Adaptation with Visual Object Recognition. In Intl. Journal of Computer Vision (IJCV). Vol. 109. 3–27.

[12] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. 2012. GeoMetric Flow Kernel for Unsupervised Domain Adaptation. In Intl. Conf. on Computer Vision and Pattern Recognition (CVPR). Columbus, OH.

[13] Hannah Goh, Sharp Labbe, and Daniel F. Schmidt. 2017. Uncertainty-aware Evaluation for Time-Series Forecasting via Attention Sharing. In Domain Adaptation for Time-Series Classification to Mitigate Covariate Shift. MM '22, October 10–14, 2022, Lisboa, Portugal.

[14] Kilian Fatras, Jonathan Weed, and Philippe Rigollet. 2017. Near-Linear Time Approximation Algorithms for Optimal Transport via Sinkhorn Iterations. In Advances in Neural Information Processing Systems (NIPS). 1961–1971.

[15] Hao-Ting Wang, James Chapman, and Tuyen Van. 2021. CCA-Zoo: A Collection of Regularized Alignment and Discriminative Feature Learning for Unsupervised Deep Domain Adaptation. In Advances in Neural Information Processing Systems (NIPS). Vol. 34(3). 465–479. https://doi.org/10.5555/3447595.3449499.

[16] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. 2013. Gated Recurrent Neural Networks. In Intl. Conf. on Computer Vision and Pattern Recognition (CVPR). Providence, RI, 2066–2073.

[17] Boqing Gong, Kristen Grauman, and Fei Sha. 2014. Learning Kernels for Unsupervised Domain Adaptation with Visual Object Recognition. In Intl. Journal of Computer Vision (IJCV). Vol. 109. 3–27.

[18] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. 2012. GeoMetric Flow Kernel for Unsupervised Domain Adaptation. In Intl. Conf. on Computer Vision and Pattern Recognition (CVPR). Columbus, OH.

[19] Tong He, Zhi Zhang, Hang Zhang, Zhengyue Zhang, Junyuan Xie, and Mu Li. 2019. Bag of Tricks for Image Classification with Convolutional Neural Networks. In Intl. Conf. on Computer Vision and Pattern Recognition (CVPR). Long Beach, CA, 558–567.

[20] Neil Housby, Andrei Giguue, Stanislav Jastrzębski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gusmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In Intl. Conf. on Machine Learning (ICML). Vol. 97. 2790–2799.

[21] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-Tuning for Text Classification. In Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL). Vol. 1. Melbourne, Australia, 328–339.

[22] Hailin Hu, Mingjian Tang, and Chengcheng Bai. 2020. DATSING: Data Augmentation for Domain Adaptation. In Trans. on Pattern Analysis and Machine Intelligence (TPAMI). Vol. 42(9). 1853–1865. https://doi.org/10.1109/34.14112.14012.

[23] Sunyin Jiang, Youngsuk Park, Danielle C. Maddix, Hao Wang, and Yuyang Wang. 2015. In-Context Learning of Representations for Domain Adaptation. In Advances in Neural Information Processing Systems (NIPS). Vol. 28. 2066–2073.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Structure from Motion Revisited. In Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL). Prague, Czech Republic, 1845–1854.
A APPENDICES

A.1 Notations

Table 5: Overview of notations used for our domain adaptation methodology.

| Notation | Description |
|----------|-------------|
| Multivariate Time-Series (MTS) Classification |
| $\mathbf{U} = \{ \mathbf{u}_1, \ldots, \mathbf{u}_m \} \in \mathbb{R}^{m \times l}$ | An MTS (ordered sequence) of $l \in \mathbb{N}$ streams |
| $\mathbf{u}_i = (u_{i1}, \ldots, u_{il})$ | A stream of an MTS |
| $m \in \mathbb{N}$ | Length of a time-series |
| $\mathbf{U} = \{ \mathbf{U}_1, \ldots, \mathbf{U}_{n_U} \}$ | Set of $n_U$ time-series with $\mathbf{U} \in \mathbb{R}^{n_U \times m \times l}$ |
| $y \in \mathcal{Y}$ | Unknown class label for a given MTS |
| $\mathcal{Y}$ | Label space |
| $|\mathcal{Y}|$ | Number of class labels |
| Domain Adaptation |
| $\mathbf{U}_T$ | Target domain dataset |
| $\mathbf{U}_S$ | Source domain dataset |
| $\mathbf{U}_T$ | Training subset of the target domain dataset |
| $\mathbf{U}_S$ | Training subset of the target domain dataset |
| $\mathcal{U}_T$ | Validation subset of the target domain dataset |
| $\mathcal{U}_S$ | Validation subset of the source domain dataset |
| $\mathbf{U}_T$ | MTS of the target domain dataset |
| $\mathbf{U}_S$ | MTS of the source domain dataset |
| $\mathbf{U}_T$ | Trainsings subset of the target domain dataset |
| $\mathbf{U}_S$ | Trainsings subset of the source domain dataset |
| $\mu_T$ | Mean of $\mathbf{U}_T$ |
| $\mu_S$ | Mean of $\mathbf{U}_S$ |
| $\Sigma_T$ | Covariance matrix of $\mathbf{U}_T$ |
| $\Sigma_S$ | Covariance matrix of $\mathbf{U}_S$ |
| $\mathcal{T}$ | Transformation |
| $\mathcal{D}_S \rightarrow \mathcal{D}_T$ | Latent embedding of the training MTS of the target domain dataset |
| $f(\mathbf{U}_T) \in \mathbb{R}^{q_T \times \omega_T}$ | Latent embedding of the training MTS of the source domain dataset |
| $f(\mathbf{U}_S) \in \mathbb{R}^{q_S \times \omega_S}$ | Size of the embedding $f(\mathbf{U}_s) \in \mathbb{R}^{q_S \times \omega_S}$ |
| $\mathbf{q}_s$, $\mathbf{w}_c$ | Feature space |
| $\mathcal{X}$ | Marginal probability of $\mathcal{X}$ |
| $P(\mathcal{X})$ | Joint distribution of $\mathcal{X}$ and $\mathcal{Y}$ |
| $P(\mathcal{Y} | \mathcal{X})$ | Conditional distribution between $\mathcal{X}$ and $\mathcal{Y}$ |
| $\mathcal{D}$ | Domain |
| $\mathcal{D}_T = \{ \mathcal{X}_T, \mathcal{Y}_T \}_{i=1}^{N_T}$ | Target domain of $N_T$ labeled samples |
| $\mathcal{D}_S = \{ \mathcal{X}_S, \mathcal{Y}_S \}_{i=1}^{N_S}$ | Source domain of $N_S$ labeled samples |
| $\mathcal{H}$ | Hypothesis space |
| $\mathcal{R}_T$ | Target domain error |
| $\mathcal{R}_S$ | Source domain error |
| $d_{\mathcal{H}, \mathcal{H}}$ | Discrepancy distance between $\mathcal{D}_S$ and $\mathcal{D}_T$ w.r.t. the hypothesis space $\mathcal{H}$ |
| Optimal Transport |
| $\mathcal{C}(\mathcal{T})$ | Transportation cost |
| $c : \mathcal{D}_T \times \mathcal{D}_S \rightarrow \mathbb{R}^+$ | Distance function |
| $\mathbf{T}_0$ | Optimal transportation problem |
| $\alpha \in \Theta$ | General coupling |
| $\Theta \in \mathcal{P}(\mathcal{D}_T, \mathcal{D}_S)$ | Set of all probabilistic couplings |
| $W_p$ | Wasserstein distance of order $p$ |
| $c(\mathbf{U}_T, \mathbf{U}_S)$ | Cost function |
| $d(\mathbf{U}_T, \mathbf{U}_S)^p$ | Distance function of order $p$ |

A.2 Correlation Alignment (CORAL)

The calculation of subspace-based methods is simple and efficient. Hence, we use CORAL [75], which minimizes the domain shift by aligning the second-order statistics (i.e., the original feature distributions of source and target domains). Suppose $\mu_T$ and $\mu_S$ are the means of $f(\mathbf{U}_T)$ and $f(\mathbf{U}_S)$, and $\Sigma_T$ and $\Sigma_S$ are the covariance matrices. We normalize the features to have zero mean ($\mu_T = \mu_S = 0$). Then, CORAL minimizes the distance between $f(\mathbf{U}_T)$ and $f(\mathbf{U}_S)$ by a linear transformation $A$ by

$$\min_{A} \| \mathcal{C}_S - \mathcal{C}_T \|_F^2 = \min_{A} \| A^T \mathcal{C}_T A - \mathcal{C}_T \|_F^2,$$

(9)

where $\| \cdot \|_F^2$ is the squared Frobenius norm, and $\mathcal{C}_S$ is the covariance of the transformed source features $f(\mathbf{U}_S)A$ [75]. The optimal solution of this problem is given by

$$A^* = P_T \Sigma_T^{-\frac{1}{2}} P_S^T \Sigma_S^{-\frac{1}{2}} P_T^T \Sigma_T^{-\frac{1}{2}} P_S,$$

(10)

with $r = \min(r_{\Sigma_S}, r_{\Sigma_T})$, where $r_{\Sigma_S}$ and $r_{\Sigma_T}$ denote the rank of $\Sigma_S$ and $\Sigma_T$, and $\Sigma^*$ is the Moore-Penrose pseudoinverse of $\Sigma$. We use the singular value decomposition of a real matrix $Y$ to compute the largest $r \leq r_Y$ singular values $\Sigma_{Y_{1:r_Y}}$ and left and right singular vectors $P_{Y_{1:r_Y}}$ of $Y = P_T \Sigma_T V_T$ of the real matrix $Y$ of rank $r_Y$ [7].

A.3 Datasets Overview

Table 6 gives an overview of sample counts for the right-handed target domains for training $\mathcal{U}_T$ and validation $\mathcal{U}_V$. Results for these datasets are given in Table 2 and 3. Here, the left-handed dataset comprise all writers. Table 7 shows the sample numbers for left-handed writers separated for each writer at an 80/20 training validation split. We propose counts for the source domains for training $\mathcal{U}_S$, and validation $\mathcal{U}_{V_S}$, data. Results are averaged over all writers (four for the OnHW-symbols and split OnHW-equations datasets, and nine for the OnHW-chars dataset) and are given in Table 4 for domain adaptation and in Table 1 for transfer learning.

Table 6: Overview of sample numbers of online handwriting (OnHW) recognition datasets for writer-dependent (WD) and writer-independent (WI) and right- and left-handed classification tasks. Top line: training. Bottom line: validation.

| Dataset | Right-handed | Left-handed |
|---------|--------------|-------------|
| OnHW-symbols [60] | 1,855 | 288 |
| Split OnHW-equations [60] | 31,697 | 5,021 |
| OnHW-chars [61] | 11,524 upper 11,672 lower | 903 288 |
| | 8,209 upper 3,978 lower | 449 364 |
Figure 9: Embedding visualization for the sinusoidal datasets for the target domain \( \mathcal{U}_T \) and the source domain \( \mathcal{U}_S \) with noise parameters \( b \in B = \{0.0, 0.5, 1.0, 1.5, 1.9\} \). Marker : target domain embeddings. Marker \( \times \): source domain validation embeddings.

Figure 10: Embedding visualization for the target domain \( \mathcal{U}_T \), source domain \( \mathcal{U}_S \) and transformed source domain \( \mathcal{U}^T_S \) for the sinusoidal datasets. We show embeddings for EMD and Sinkhorn transport without regularization. Evaluated are nine metrics for transformation selection: CC, PC, CORAL (standard, Jeff and Stein), correlation normalized, MMD, kMMD, HoMM, without transformation and best transformation. Noise is \( b = 0.5 \). Note that we applied t-SNE for each plot separately that leads to different embeddings for the same data (i.e., target domain \( \mathcal{D}_T \)). Figure 11 follows.

Table 7: Overview of number of samples for each left-handed writer for the OnHW datasets for train/validation splits. For OnHW-chars, we count for lower/upper/combined.

| Dataset                   | Writer | Train (\( \mathcal{U}_S \)) | Val. (\( \mathcal{U}_S \)) |
|---------------------------|--------|-----------------------------|-----------------------------|
| OnHW-symbols-L [60]       | 1      | 71                          | 18                          |
|                           | 2      | 72                          | 18                          |
|                           | 3      | 70                          | 18                          |
|                           | 4      | 75                          | 19                          |
| Split OnHW-equations-L [60]| 1      | 1,299                       | 327                         |
|                           | 2      | 1,067                       | 267                         |
|                           | 3      | 1,295                       | 324                         |
|                           | 4      | 1,360                       | 341                         |
| OnHW-chars-L [61]         | 1      | 78 / 89 / 167               | 26 / 15 / 41                |
|                           | 2      | 78 / 89 / 167               | 26 / 15 / 41                |
|                           | 3      | 78 / 70 / 149               | 26 / 11 / 37                |
|                           | 4      | 78 / 89 / 167               | 26 / 15 / 41                |
|                           | 5      | 79 / 89 / 168               | 26 / 15 / 41                |
|                           | 6      | 202 / 218 / 420             | 62 / 42 / 104               |
|                           | 7      | 78 / 89 / 167               | 26 / 15 / 41                |
|                           | 8      | 78 / 89 / 167               | 26 / 15 / 41                |
|                           | 9      | 120 / 130 / 250             | 360 / 26 / 386              |

A.4 Feature Embeddings for the Sinusoidal Dataset

To better visualize the differences between domain-dependent embeddings and different methods, we plot two dimensional embeddings of the features \( f(U) \). For the sinusoidal dataset, \( f(U) \) is of size \( 50 \times 30 \) (reshaped 1,500). We use t-SNE [82] with initial dimension 1,500, perplexity of 30, an initial momentum of 0.5, and a final momentum of 0.8. Figure 9 visualizes the feature embeddings for different noise parameters \( b \) with class label dependent colors. Without noise \( (b = 0) \), the clusters are clearly separable, while only label ‘8’ and ‘9’ overlap. As the noise increases \( b = 1.5 \), also the cluster with label ‘7’ overlaps, and finally label ‘6’ and ‘3’ for \( b = 1.9 \). This is reflected by the results in Figure 8, where the accuracy drops for \( b > 0.7 \). The validation samples with domain shift are notably distant to the target samples.

Figure 10 and 11 visualize the feature embeddings \( f(U) \) for \( \mathcal{U}_T \) (blue), \( \mathcal{U}_S \) (orange), and the transformed \( \mathcal{U}^T_S \) (red). It is notable that EMD forms spread clusters of the specific sample embedding for the transformed source domain, while Sinkhorn shapes small clusters that are close to the target domain clusters, but are outlying. This distance increases for a higher noise. Choosing the best transformation (Figure 10a and 10b), the transformed embeddings are mostly consistent, the kMMD (Figure 11a and 11b) and cross correlation (Figure 11e and 11f) distance metrics (that result in the
highest classification accuracy), are similar to the best transformation. HoMM (Figure 11c and 11d) with the lowest classification accuracy, also results in a larger spread of transformed clusters. Different are the variances Stein and Jeff CORAL trained with EMD where the transformed source domain samples are notably out-of the target domain samples (see Figure 11i and 11k).

A.5 Deep Metric Learning Searches
We perform a large hyperparameter search on the OnHW-symbols dataset for all optimal transport techniques. Results are shown in Figure 12. The differences between EMD and SEMD are marginally, as well as between $L_pL_1$ and $L_1L_2$ regularization techniques. In general Sinkhorn performs better than EMD. While the log metric performs best for EMD, median, max and loglog yield better results for Sinkhorn. The differences for distance metrics are marginal for Sinkhorn. For follow-up trainings, we choose the squared Euclidean distance metric with loglog metric. Indeed, the similarity comparison method for transformation selection has the highest impact. kMMD consistently yields the highest classification accuracies followed by Stein and Jeff CORAL. Again, HoMM of order 3 can outperform MMD, but not kMMD.

A.6 Evaluation per Writer
We adapt each writer separately as writer can have very different writing styles, and hence, different domains of sensor features. Figure 13 to Figure 17 shows all results for five different optimal transport techniques and transformation selection methods. Using the best transformation mostly achieves 100%, which is the upper bound for the optimal transformation. Without transformation, the right-handed model leads to a poor accuracy of below 10% (the lower bound). For the OnHW-symbols and split OnHW-equations datasets, the accuracy of the models drop for the writer with ID 3. For the OnHW-chars dataset, the writers with ID 3 and 6 are outlier. The reason is that these persons wrote inconsistent. Again, the difference between EMD and SEMD, and Sinkhorn with and without regularization is marginal in accuracy.

Figure 18 shows results of transfer learning separated for all writers for all OnHW datasets. Consistently, adapting an additional layers yields the lowest classification results. While the OnHW-symbols dataset is rather small, all post-training techniques fail to successfully classify symbols (see Figure 18a). The split OnHW-equations dataset is large, and hence, post-training results in classification accuracies between 60% and 100% (see Figure 18b). Results for the OnHW-chars lower, upper and combined are similar (see...
Figure 12: Hyperparameter search for distance metrics for optimal transport methods and transformation selection methods on the OnHW-symbols dataset. Results are averaged over four writers. 1: median, 2: max, 3: log, 4: loglog, 5: None.

Figure 13: Evaluation of the transformed embeddings for the left-handed OnHW-symbols dataset for each of the four writers.

Figure 14: Evaluation of the transformed embeddings for the left-handed split OnHW-equations dataset for each of the four writers.

Figure 15: Evaluation of the transformed embeddings for the left-handed OnHW-chars (lower) dataset for each writer.
Table 8: Architecture details for the sinusoidal datasets.

| Layer        | Features                                     |
|--------------|------------------------------------------------|
| Encoder      | Size: (Sequence length = 200, 13)            |
| Input        |                                              |
| Convolution (1D) | Filters: 50, kernel size: 4, activation: softmax |
| Max Pooling (1D) | Pool size: 4                                 |
| Classifier   |                                              |
| Batch Normalization | Rate 20%                                      |
| Dropout      |                                              |
| LSTM         | Units: 10, activation: tanh                  |
| Dense        | Units: 20                                     |
| Output       | Size: number classes (10)                    |

Table 9: Architecture details for the OnHW datasets.

| Layer        | Features                                     |
|--------------|------------------------------------------------|
| Encoder      | Size: (Sequence length, 1)                    |
| Input        |                                              |
| Convolution (1D) | Filters: 200, kernel size: 4, activation: relu |
| Max Pooling (1D) | Pool size: 2                                 |
| Classifier   |                                              |
| Batch Normalization | Rate 20%                                      |
| Dropout      |                                              |
| BilSTM       | Units: 60, activation: tanh, return sequ.: True |
| Dense        | Units: 100                                    |
| Dense (time distributed) | Units: number classes, activation: softmax |
| Output       | Size: number classes                          |

Figure 18c to 18e). Post-training the full model yields marginally better results than training only the last layer or the layers after the middle of the model. Hence, the model requires to overfit on the specific writer. Results highly vary with the writer ID.

A.7 Details on the Architectures

Table 8 and 9 show architecture details for the sinusoidal dataset, and OnHW datasets, respectively. Both models contain a feature extractor of the time-series datasets, and temporal unit, and dense layers for classification. We use a small LSTM of 10 units for the synthetic dataset, and two stacked bidirectional LSTM layers of 60 units each for the OnHW datasets. For the optimal transport methods, we use the output of the max pooling layer before the batch normalization and dropout layers and the temporal units. We train the synthetic dataset for 100 epochs, a batch size of 100, the Adam optimizer with learning rate 0.0001, and the categorical cross-entropy loss. We train the OnHW datasets for 1,000 epochs, a batch size of 50, the Adam optimizer with learning rate 0.0001, and the categorical cross-entropy loss. For transfer learning techniques we train the synthetic dataset for 80 epochs, and the OnHW datasets for 100 epochs with the same optimizer parameters.