ADATIME: A Benchmarking Suite for Domain Adaptation on Time Series Data

MOHAMED RAGAB* and EMADELDEEN ELDELE*, Nanyang Technological University, Singapore
WEE LING TAN, Institute for Infocomm Research, A*STAR, Singapore
CHUAN-SHENG FOO, Institute for Infocomm Research, A*STAR, Singapore
ZHENGHUA CHEN, Institute for Infocomm Research, A*STAR, Singapore
MIN WU, Institute for Infocomm Research, A*STAR, Singapore
CHEE-KEONG KWOH, Nanyang Technological University, Singapore
XIAOLI LI, Institute for Infocomm Research, A*STAR, Singapore and Nanyang Technological University, Singapore

Unsupervised domain adaptation methods aim to generalize well on unlabeled test data that may have a different (shifted) distribution from the training data. Such methods are typically developed on image data, and their application to time series data is less explored. Existing works on time series domain adaptation suffer from inconsistencies in evaluation schemes, datasets, and backbone neural network architectures. Moreover, labeled target data are often used for model selection, which violates the fundamental assumption of unsupervised domain adaptation. To address these issues, we develop a benchmarking evaluation suite (AdaTime) to systematically and fairly evaluate different domain adaptation methods on time series data. Specifically, we standardize the backbone neural network architectures and benchmarking datasets, while also exploring more realistic model selection approaches that can work with no labeled data or just few labeled samples. Our evaluation includes adapting state-of-the-art visual domain adaptation methods to time series data as well as the recent methods specifically developed for time series data. We conduct extensive experiments to evaluate 11 state-of-the-art methods on five representative datasets spanning 55 cross-domain scenarios. Our results suggest that with careful selection of hyper-parameters, visual domain adaptation methods are competitive with methods proposed for time series domain adaptation. In addition, we find that hyper-parameters could be selected based on realistic model selection approaches. Our work unveils practical insights for applying domain adaptation methods on time series data and builds a solid foundation for future works in the field. The code is available at github.com/emadeldeen24/AdaTime.

ACM Reference Format:
Mohamed Ragab, Emadeldeen Eldele, Wee Ling Tan, Chuan-Sheng Foo, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. 2022. ADATIME: A Benchmarking Suite for Domain Adaptation on Time Series Data. 1, 1 (February 2022), 18 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.

Authors' addresses: Mohamed Ragab, mohamedr002@e.ntu.edu.sg; Emadeldeen Eldele, emad0002@ntu.edu.sg, Nanyang Technological University, 50 Nanyang Ave, Singapore, 639798; Wee Ling Tan, Institute for Infocomm Research, A*STAR, 1 Fusionopolis Way, Singapore, 138632, weeling.tan@eng.ox.ac.uk; Chuan-Sheng Foo, Institute for Infocomm Research, A*STAR, 1 Fusionopolis Way, Singapore, 138632, foo_chuan_sheng@i2r.a-star.edu.sg; Zhenghua Chen, Institute for Infocomm Research, A*STAR, 1 Fusionopolis Way, Singapore, 138632, chen0832@e.ntu.edu.sg; Min Wu, Institute for Infocomm Research, A*STAR, 1 Fusionopolis Way, Singapore, 138632, wumin@i2r.a-star.edu.sg; Chee-Keong Kwoh, Nanyang Technological University, 50 Nanyang Ave, Singapore, 639798, asckkwoh@ntu.edu.sg; Xiaoli Li, Institute for Infocomm Research, A*STAR, 1 Fusionopolis Way, Singapore, 138632 and Nanyang Technological University, 50 Nanyang Ave, Singapore, 639798, xlli@i2r.a-star.edu.sg.

1 INTRODUCTION

Time series classification problem is predominant in many real-world applications including healthcare and manufacturing. Recently, deep learning has gained more attention in time series classification tasks. It aims to learn the temporal dynamics in the complex underlying data patterns, assuming access to a vast amount of labeled data [1, 2]. Yet, annotating time series data can be challenging and burdensome due to its complex nature that requires expert domain knowledge [3–7]. One way to reduce the labeling burden is to leverage annotated data (e.g., synthetic or public data) from a relevant domain (i.e., source domain) for the model's training while testing the model on the domain of interest (i.e., target domain). However, the source and target domains may have distinct distributions, resulting in a significant domain shift that hinders the model performance on the target domain. Such a problem commonly exists in many time series applications including human activity recognition [3, 8] and sleep stage classification (SSC) tasks [9]. For instance, a model can be trained to identify the activity of one subject (i.e., source domain) and tested on data from another subject (i.e., target domain), leading to poor performance that is caused by the domain shift problem.

Unsupervised Domain Adaptation (UDA) aims to transfer knowledge learned from a labeled source domain to an unseen target domain, tackling the domain shift problem. A considerable amount of literature has been proposed for UDA on visual applications [10–12]. One prevailing paradigm aims to minimize statistical distribution measures to mitigate the distribution shift problem between the source and target domains [13–17]. Another promising paradigm that has recently emerged leverages adversarial training techniques to mitigate the domain gap [18–21], inspired by generative adversarial networks [22].

Recently, more attention has been paid to Time Series UDA (TS-UDA) [3, 8, 23]. However, previous work on TS-UDA methods suffers from the following limitations. First, most of the existing algorithms are specialized to particular applications or domains [3, 24, 25]. Thus, there is a clear shortage of baseline methods when applying domain adaptation on time series data. Second, existing TS-UDA works lack consistent evaluation schemes including benchmark datasets, preprocessing, and backbone networks. For instance, methods using recurrent neural networks as a backbone network [26] have been compared against methods with convolutional neural network-based backbone networks [8]. In addition to differences in backbones, training procedures can also vary between different algorithms in terms of the number of epochs, weight decay, and learning rate schedulers [26, 27], which results in an inconsistent evaluation of new algorithms. Last, most of the existing TS-UDA approaches often utilize labeled data from the target domain for model selection, violating the unsupervised assumption of UDA [8, 23], and providing an over-optimistic view of their real-world performance. The aforementioned issues can contribute to the performance, and the performance gain is mistakenly attributed to the proposed UDA method.

In this work, we develop a systematic evaluation suite (AdaTime) to tackle the aforementioned obstacles and remove all extraneous factors to ensure a fair evaluation of different UDA algorithms on time series data. In AdaTime, we include current TS-UDA methods and re-implemented various state-of-the-art visual UDA methods that can be adapted to time series data. To ensure a fair evaluation of these methods, we standardize backbone networks and training procedures, data preparation, and preprocessing to address the inconsistency in previous evaluation schemes. Then, to select the model hyper-parameters, we explore more realistic model selection strategies that do not require target labels. Therefore, the main contributions of this paper can be summarized as follows:

- We systematically and fairly evaluate existing UDA methods on time series data. To the best of our knowledge, this is the first work to benchmark different UDA methods on time series data.

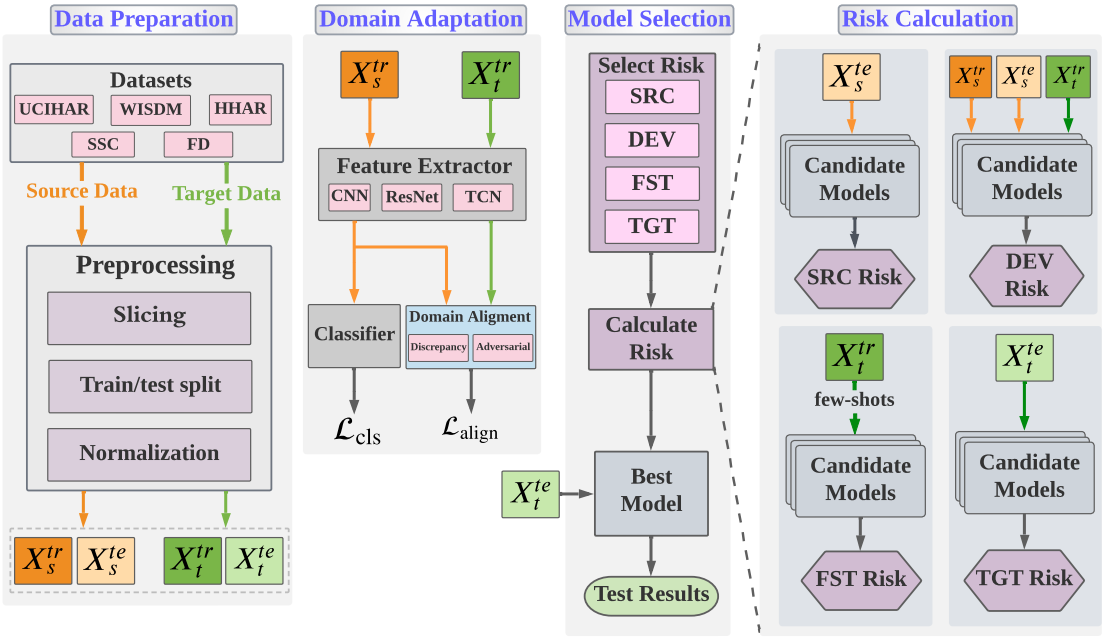


Fig. 1. Our benchmarking suite AdaTime consists of three main steps: Data Preparation, Domain Adaptation, and Model Selection. We first prepare the train and test data for both source and target domains (i.e., $X_s^{tr}, X_s^{te}, X_t^{tr}, X_t^{te}$). Then the training sets of source and target domains are passed through the backbone network to extract the corresponding features. The domain alignment algorithm being evaluated is then used to address the distribution shift between the two domains. Last, given a specific risk type, we calculate the risk value for all the candidate models and then select the hyper-parameters of the one achieving the lowest risk. The selected model is lastly used for reporting the test results given the target domain test set (best viewed in colors).

- We develop a benchmarking evaluation suite (AdaTime) that uses a standardized evaluation scheme and realistic model selection techniques to evaluate different UDA methods on time series data.
- We evaluate 11 state-of-the-art UDA methods on five representative time series datasets spanning 55 cross-domain scenarios, and present comprehensive conclusions and recommendations for the TS-UDA problem. These evaluation results and analysis can provide a systematic guideline for future research on TS-UDA.

The following sections are organized as follows. In Section 2, we define the unsupervised domain adaptation problem, and how adaptation is generally achieved. Section 3 describes the main components of our AdaTime suite such as benchmarking datasets, unified backbone networks, adapted UDA algorithms, model selection approaches, and unified evaluation schemes. Section 4 shows the evaluation results and discusses the main findings of our experiments. Section 5 presents the main conclusions and recommendations.

2 DOMAIN ADAPTATION

2.1 Problem Formulation

We start by defining the unsupervised domain adaptation problem. We assume access to labeled data from a source domain $\mathcal{X}_s = \{(x_s^i, y_s^i)\}_{i=1}^{N_s}$ that represents univariate or multivariate time series data, and unlabeled data from a target domain $\mathcal{X}_t = \{(x_t^j)\}_{j=1}^{N_t}$, where N_s and N_t denote the number of samples for $\mathcal{X}_s$ and $\mathcal{X}_t$ respectively. Here we focus

on classification and assume that both domains share the same label space $Y = \{1, 2, \dots K\}$, where K is the number of classes. Upon preprocessing, the source domain is split into a training set X_s^{tr} with N_s^{tr} samples, and a test set X_s^{te} with N_s^{te} samples. Similarly, the target domain is split into a training set X_t^{tr} with N_t^{tr} samples, and a test set X_t^{te} with N_t^{te} samples. The source and target domains are sampled from different marginal distributions, i.e., $P_s(x) \neq P_t(x)$, while the conditional distribution remains stable, i.e., $P_s(y|x) = P_t(y|x)$. The main goal of UDA is to minimize the distribution shift between $P_s(x)$ and $P_t(x)$, assuming they share the same label space.

2.2 General Approach

The mainstream of UDA algorithms is to address the domain shift problem by finding domain invariant feature representation. Formally, given a feature extractor network $f_\theta : X \rightarrow Z$, which transforms the input space to the feature space, the UDA algorithm mainly optimizes the feature extractor network to minimize a domain alignment loss $\mathcal{L}_{\text{align}}$, aiming to mitigate the distribution shift between the source and target domains such that $P_s(f_\theta(x)) = P_t(f_\theta(x))$. The domain alignment loss can either be estimated from a statistical distance measure or an adversarial discriminator network, which can be formalized as follows:

$$\mathcal{L}_{\text{align}} = \min_{f_\theta, h_\theta} \ell(Z_s, Z_t), \tag{1}$$

where ℓ can be a statistical distance or an adversarial loss.

Concurrently, a classifier network h_θ is applied on top of the feature extractor network to map the encoded features to the corresponding class probabilities. Particularly, given the source domain features Z_s generated from the feature extractor, we can calculate the output probabilities $\mathbf{p}_s = h_\theta(Z_s))$. Thus, the source classification loss can be formalized as follows

$$\mathcal{L}_{\text{cls}}^s = -\mathbb{E}_{(\mathbf{x}_s, y_s)\sim P_s} \sum_{k=1}^{K} \mathbb{1}_{[y_s=k]} \log \mathbf{p}_s^k, \tag{2}$$

where $\mathbb{1}$ is the indicator function, which is set to be 1 when the condition is met, and set to 0 otherwise.

Both the source classification loss $\mathcal{L}_{\text{cls}}^s$ and the domain alignment loss $\mathcal{L}_{\text{align}}$ are jointly optimized to mitigate the domain shift while learning the source classification task, which can be expressed as

$$\min_{f_\theta, h_\theta} \mathcal{L}_{\text{cls}}^s + \mathcal{L}_{\text{align}}. \tag{3}$$

we refer to the composition of the the feature extractor f_θ and the classifier network h_θ as the model m, such that $m = h_\theta(f_\theta(\cdot))$.

3 ADATIME: A BENCHMARKING APPROACH FOR TIME SERIES DOMAIN ADAPTATION

3.1 Framework Design

The key motivation for our approach is to address the inconsistent evaluation schemes, datasets, and backbone networks. Such inconsistencies can boost the performance and be misattributed to the proposed UDA method. Therefore, we design our benchmarking framework to address these issues while ensuring fair evaluation across different UDA methods. For example, to remove the effect of different backbone networks, we use the same backbone network when comparing different UDA methods. Additionally, we standardize the benchmarked datasets and their preprocessing

schemes when evaluating any UDA method. Table 1 summarizes the existing experimental flaws and our corresponding design decision.

Table 1. Summary of the design choices in AdaTime framework.

Problem	Design Choice
Different datasets and cross-domain scenarios	Unified datasets and cross-domain scenarios for all the methods
Inconsistent backbone networks	Same backbone network is fixed for all the methods
Different training procedures	Unified the evaluation schemes and training procedure for all the methods

3.2 Framework Overview

In this work, we systematically evaluate different UDA algorithms on time series data, ensuring fair and realistic procedures. Fig. 1 shows the details of AdaTime flow, which proceeds as follows. Given a dataset, we first apply our standard data preparation schemes on both domains, including slicing, splitting to train/test portions, and normalization. Subsequently, the backbone network extracts the source and target features Z_s^{tr} and Z_t^{tr} from the source training data X_s^{tr} and target training data X_t^{tr} respectively. The selected UDA algorithm is then applied to mitigate the distribution shift between the extracted features of the two domains. We generally categorize the adopted UDA algorithms into discrepancy- and adversarial-based approaches. Last, to set the hyper-parameters of the UDA algorithm, we consider three practical model selection approaches that do not require any target domain labels or allow for only few-shot labeled samples. These approaches are source risk (SRC), deep embedded evaluation risk (DEV) [28], and few-shot target risk (FST). Our evaluation pipeline standardizes experimental procedures, preventing extraneous factors from affecting performance, thus enabling fair comparison between different UDA methods.

The code of AdaTime is publicly available for researchers to enable seamless evaluation of different UDA methods on time series data. Merging a new algorithm or dataset into AdaTime will be just a matter of adding a few lines of code.

3.3 Benchmarking Datasets

We select the most commonly used time series datasets from two real-world applications, i.e., human activity recognition and sleep stage classification. The benchmark datasets span a range of different characteristics including complexity, type of sensors, sample size, class distribution, and severity of domain shift, enabling more broad evaluation.

Table 2 summarizes the details of each dataset, e.g., the number of domains, the number of sensor channels, the number of classes, the length of each sample, as well as the total number of samples in both training and test portions. The selected datasets are detailed as follows:

3.3.1 UCIHAR. UCIHAR dataset [29] contains data from three sensors namely, accelerometer, gyroscope, and body sensors, that have been applied on 30 subjects. Each subject has performed six activities, i.e., walking, walking upstairs, downstairs, standing, sitting, and lying down. Due to the aforementioned variability between subjects, we treat each subject as a separate domain. Here, we randomly selected five cross-domain scenarios out of the large number of cross-domain combinations, as in [8, 23].

3.3.2 WISDM. In the WISDM dataset [30], accelerometer sensors were applied to collect data from 36 subjects performing the same activities as in the UCIHAR dataset. However, this dataset can be more challenging because of the

class-imbalance issue in the data of different subjects. Specifically, data from some subjects may contain only samples from a subset of the overall classes (see Fig. S.1 in the Appendix). Similar to the UCIHAR dataset, we consider each subject as a separate domain and randomly select five cross-domain scenarios.

3.3.3 HHAR. The Heterogeneity Human Activity Recognition (HHAR) dataset [31] has been collected from 9 different subjects using sensor readings from smartphones and smartwatches. In our experiments, we use the same smartphone device i.e., Samsung smartphone, for all the selected subjects to reduce the heterogeneity. In addition, we consider each subject as one domain and form the five cross-domain scenarios from randomly selected subjects.

3.3.4 SSC. Sleep stage classification (SSC) problem 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). We adopt Sleep-EDF dataset [32], which contains EEG readings from 20 healthy subjects. We select a single channel (i.e., Fpz-Cz) following previous studies [33], and include 10 different subjects to construct the five cross-domain scenarios.

3.3.5 MFD. The Machine Fault Diagnosis (MFD) dataset [34] has been collected by Paderborn University for the purpose of identifying various types of incipient faults using vibration signals. The data was collected under four different operating conditions, and in our experiments, each of these conditions was treated as a separate domain. We used five different cross-condition scenarios to evaluate the domain adaptation performance. Each sample in the dataset consists of a single univariate channel with 5120 data points following previous works [24, 35].

Table 2. Details of adopted datasets. Further details about selected cross-domain scenarios for each dataset cab be found in the Appendix.

Dataset	# Users/Domains	# Channels	# Classes	Sequence Length	Training set	Testing set
UCIHAR	30	9	6	128	2300	990
WISDM	36	3	6	128	1350	720
HHAR	9	3	6	128	12716	5218
SSC	20	1	5	3000	14280	6130
MFD	4	1	3	5120	7312	3604

3.4 Backbone Networks

In general, a UDA algorithm consists of a feature extractor network to extract the features from the input data, a classifier network to classify the features into different classes, and a domain alignment component to minimize the shift between domains. Here, we refer to the feature extractor as the backbone network. The backbone network is responsible for transforming the data from the input space to the feature space, where UDA algorithms are usually applied. Thus, the backbone network can have a significant influence on the performance of the UDA method, independent of the actual domain adaptation component. Hence, it is necessary to standardize the choice of backbone network to fairly compare different UDA methods. However, some previous TS-UDA works adopted different backbone architectures when comparing against baseline methods, leading to inaccurate conclusions.

To tackle this problem, we design AdaTime to ensure the same backbone network is used when comparing different UDA algorithms, promoting fair evaluation protocols. Furthermore, to better select a suitable backbone network for TS-UDA application, we experiment with three different backbone architectures:

- **1-dimensional convolutional neural network (1D-CNN)**: consists of three convolutional blocks, where each block consists of a 1D-Convolutional network, a BatchNorm layer, a non-linearity ReLU activation function, and finally, a MaxPooling layer [4, 9].
- **1-dimensional residual network (1D-ResNet)**: is a deep Residual Network that relies on a shortcut residual connection among successive convolutional layers [1, 36]. In this work, we leveraged 1D-ResNet18 in our experiments.
- **Temporal convolutional neural network (TCN)**: uses causal dilated convolutions to prevent information leakage across different convolutional layers and to learn temporal characteristics of time series data [37, 38].

These architectures are widely used for time series data analytics and are different from each other in terms of their complexity and the number of trainable parameters.

Table 3. Summary of unsupervised domain adaptation algorithms implemented in AdaTime.

Algorithm	Application	Category	Distribution	Losses	Model Selection
DDC	Visual	Discrepancy	Marginal	MMD	Target Risk
Deep-Coral	Visual	Discrepancy	Marginal	CORAL	Not Mentioned
HoMM	Visual	Discrepancy	Marginal	High-order MMD	Not Mentioned
MMDA	Visual	Discrepancy	Joint	MMD, CORAL, Entropy	Target Risk
DSAN	Visual	Discrepancy	Joint	Local MMD	Not Mentioned
DANN	Visual	Adversarial	Marginal	Domain Classifier, Gradient Reversal Layer	Source Risk
CDAN	Visual	Adversarial	Joint	Conditional adversarial Domain Classifier	Importance Weighting
DIRT-T	Visual	Adversarial	Joint	Virtual adversarial Entropy Domain Classifier	Target Risk
CoDATS	Time Series	Adversarial	Marginal	Domain Classifier, Gradient Reversal Layer	Target Risk
AdvSKM	Time Series	Adversarial	Marginal	Spectral Kernel Adversarial MMD	Target Risk
SASA	Time Series	MMD	Marginal	Sparse Attention MMD	Target Risk

3.5 Domain Adaptation Algorithms

While numerous UDA approaches have been proposed to address the domain shift problem [39], a comprehensive review of existing UDA methods is out of our scope. Besides including state-of-the-art methods proposed for time series data, we also included canonical methods for visual UDA that can be adapted to time series. Overall, the implemented algorithms in AdaTime can be broadly classified according to the domain adaptation strategy, namely discrepancy-based methods and adversarial-based methods. The discrepancy-based methods aim to minimize a statistical distance between

source and target features to mitigate the domain shift problem [13–15], while adversarial-based methods leverage a domain discriminator network that enforces the feature extractor to produce domain invariant features [40, 41]. Another way to classify UDA methods is based on what distribution is aligned distribution. Some algorithms only align the marginal distribution of the feature space [8, 13–15, 40, 42], while others jointly align the marginal and conditional distributions [16, 20, 41, 43], allowing fine-grained class alignment.

The selected UDA algorithms are as follows:

- **Deep Domain Confusion (DDC)** [15]: minimizes the Maximum Mean Discrepancy (MMD) distance between the source and target domains.
- **Correlation Alignment via Deep Neural Networks (Deep CORAL)** [44]: minimizes the domain shift by aligning second-order statistics of source and target distributions.
- **Higher-order Moment Matching (HoMM)** [14]: minimizes the discrepancy between different domains by matching higher-order moments of the source and target domains.
- **Minimum Discrepancy Estimation for Deep Domain Adaptation (MMDA)** [43]: combines both MMD and CORAL with conditional entropy minimization to address the domain shift.
- **Deep Subdomain Adaptation (DSAN)** [16]: minimizes the discrepancy between source and target domains via a local maximum mean discrepancy (LMMD) that aligns relevant subdomain distributions.
- **Domain-Adversarial Training of Neural Networks (DANN)** [40]: leverages gradient reversal layer to adversarially train a domain classifier against feature extractor network.
- **Conditional Adversarial Domain Adaptation (CDAN)** [20]: realizes a conditional adversarial alignment by incorporating the task knowledge with features during the domain alignment step.
- **A DIRT-T Approach to Unsupervised Domain Adaptation** [41]: employs virtual adversarial training, conditional entropy, and teacher model to align the source and target domains.
- **Convolutional deep Domain Adaptation model for Time Series data (CoDATS)** [8]: leverages adversarial training with weak supervision by a CNN network to improve the performance on time series data
- **Adversarial Spectral Kernel Matching (AdvSKM)** [23]: leverages adversarial spectral kernel matching to address the non-stationarity and non-monotocity problem in time series data.
- **Time Series Domain Adaptation via Sparse Associative Structure Alignment (SASA)** [45]: exploits the causality property in time series data by aligning the sparse associative structure between the source and target domains.

Table 3 summarizes the selected methods, showing the application for which each method was originally proposed, the classification of each method according to domain adaptation strategy (i.e., whether it relies on discrepancy measure or adversarial training), their classification based on the category of the aligned distribution (i.e., marginal or joint distribution), the losses in each method, and the risk that each UDA method adopted to tune its model hyper-parameters. It is worth noting that our AdaTime mainly focuses on time series classification problem, and hence we excluded methods proposed for time series prediction/forecasting.

3.6 Model Selection Approaches

Model selection and hyper-parameter tuning are long-standing non-trivial problems in UDA due to the absence of target domain labels. Throughout the literature, we find that the experimental setup in these works leverages target domain labels to select hyper-parameters, which violates the main assumption of UDA. This is further clarified in

Table 3, where we find that five out of the 11 adopted UDA works use the target risk (i.e., target domain labels) in their experiments to select the hyper-parameters, while another three works use fixed hyper-parameters without describing how they are selected. To address this issue, we evaluate multiple realistic model selection approaches that do not require any target domain labels, such as: source risk [19] and Deep Embedded Validation (DEV) risk [28]. In addition, we design a few-shot target risk that utilizes affordable few labeled samples from the target domain. In the following subsections, we explain the risk calculation for each model selection approach.

3.6.1 Selection of Best Model. As shown in Fig. 1, given a set of n candidate models $\mathcal{M} = (m_1, m_2, \ldots, m_n)$ with different hyper-parameters. First, we calculate the corresponding risk value for each candidate model with respect to each model selection approach. Subsequently, we rank candidate models based on their computed risk while selecting the model with minimum risk value.

$$m_{best} = \min_{m \in \mathcal{M}} \mathcal{R}_*(m), \tag{4}$$

where m_{best} is the best model that achieves the minimum risk value, and $\mathcal{R}_* \in \{\mathcal{R}_{\text{SRC}}, \mathcal{R}_{\text{DEV}}, \mathcal{R}_{\text{FST}}, \mathcal{R}_{\text{TGT}}\}$ can be any of the model selection approaches described below.

3.6.2 Source Risk (SRC). In this approach, we select the candidate model that achieves the minimum cross-entropy loss on a test set from the source domain. Therefore, this risk can be easily applied without any additional labeling effort as it relies on existing labels from the source domain [19]. Given the source domain test data (x_s^{te}, y_s^{te}), and a candidate model m, we calculate the corresponding source risk $\mathcal{R}_{SRC}$ as:

$$\mathcal{R}_{\text{SRC}}(m) = \mathbb{E}_{x_s \sim P_s(x)} \ell_{ce}(m(x_s^{te}), y_s^{te}), \tag{5}$$

where ℓ_{ce} is the cross-entropy loss. Despite the simplicity of the source risk, its effectiveness is mainly influenced by the sample size of source data and the severity of the distribution shift. When the distribution shift is large and the sample size of the source data is small, the source risk may be less effective than the target risk. However, the source risk can be estimated using only source labels, whereas the target risk requires labeled data from the target domain.

3.6.3 DEV Risk. This approach [28] aims to find an unbiased estimator of the target risk. The key idea is to consider the correlation between the source and target features during the risk calculation. More specifically, the DEV method aims to emphasize source features that are highly correlated to the target features while giving lower weights to the less correlated features. To do so, an importance weighting scheme has been applied on the feature space. Given the source domain training features Z_s^{tr}, the source domain test set Z_s^{te}, and the target domain training features Z_t^{tr}, we first train a two-layer logistic regression model r_θ to discriminate between Z_s^{tr} and Z_t^{tr} (label features from Z_s^{tr} as 1, and Z_t^{tr} as 0), which can be formalized as follows

$$\min_{r_\theta} \mathcal{L}_d = \left[\log r_\theta(Z_s^{tr})) + \log(1 - r_\theta(Z_t^{tr}) \right]. \tag{6}$$

Subsequently, we leverage the trained r_θ to compute the importance weights w_f for the source test set as follows.

$$w_f(x_s^{te}) = \frac{N_s^{tr}}{N_t^{tr}} \frac{1 - r_\theta(Z_s^{te})}{r_\theta(Z_s^{te})}, \tag{7}$$

where $\frac{N_s^{tr}}{N_t^{tr}}$ is sample size ratio of both domains. It is worth noting that the sample ratio parameter can be computed without any target labels. Given the importance weights for each test sample of the source domain, we compute the

corresponding weighted cross-entropy loss, L_w, for the test samples of the source domain, which can be expressed as

$$L_w = \{w_f(x_s^{te})\ell_{ce}(m(x_s^{te}), y_s^{te})\}_i^{N_s^{te}}, \tag{8}$$

where m is one candidate model. Given the weighted source loss L_w and its corresponding importance weight $W = \{w_f(x_s^{te})\}$, we compute the DEV risk as follows:

$$\mathcal{R}_{\text{DEV}} = \text{mean}(L_w) + \eta\text{mean}(W) - \eta, \tag{9}$$

where $\eta = -\frac{\text{Cov}(L_w)}{\text{Var}(W)}$ is the optimal coefficient. The DEV risk can be more effective than the source risk. However, we observed in our experiments that DEV may have unstable performance with smaller source and target datasets and adds additional computational overheads. Nevertheless, DEV risk is still a more practical solution than the target risk as it does not require any target labels.

3.6.4 Target Risk (TGT). This approach involves leaving out a large subset of target domain samples and their labels as a validation set and using them to select the best candidate model. Using this risk naturally yields the best-performing hyper-parameters on the target domain. This can be seen as the upper bound for the performance of a UDA method. The target risk $\mathcal{R}_{\text{TGT}}$ is calculated as:

$$\mathcal{R}_{\text{TGT}} = \mathbb{E}_{x_t \sim P_t(x)}\ell_{ce}(m(x_t^{te}), y_t^{te}). \tag{10}$$

Even though this approach is impractical in unsupervised settings, it has been used for model selection in many previous UDA papers [41, 46].

3.6.5 Few-Shot Target (FST) Risk. We propose the few-shot target risk as a more practical alternative to the target risk. Our goal in introducing the concept of few-shot target risk was to find a more practical and realistic model selection method for unsupervised domain adaptation. We reasoned that labeling a small number of samples, known as few-shot labeling, could be practical and affordable, and therefore used this approach to select the best model for the unsupervised domain adaptation problem. Formally speaking, we use a set of q samples from the target domain as a validation set to select the best candidate model. The few-shot target risk $\mathcal{R}_{\text{FST}}$ is calculated as follows.

$$\mathcal{R}_{\text{FST}} = \frac{1}{q}\sum_{i=1}^{q}\ell_{ce}(m(x_t^i), y_t^i). \tag{11}$$

To our surprise, our experiments showed that the proposed few-shot risk was effective and performed comparably to the traditional target risk, despite being computed with only few samples.

3.7 Standardized Evaluation Scheme

3.7.1 Standardized Data Preprocessing. The preprocessing of time series data includes: domain selection, data segmentation, data splitting, and data normalization. Specifically, the details are as follows:

Domain Selection. For our closed-set domain adaptation experiments, we only consider datasets where the source and target classes are similar and have representative samples for all categories. For example, in the WISDM dataset, some subjects (i.e., domains) may only perform certain activities during data collection, leading to the absence of some classes in the data from other subjects. Therefore, we exclude such subjects from our experiments and only consider subjects with complete data for all classes.

Data Segmentation. During the data collection process, it is common for the sensor readings to be recorded in a single lengthy raw signal with a sequence length equal to the duration of the experiment. These long signals can present challenges for model training and can limit the number of available samples. To address these issues, we can use a sliding window technique to segment the data into shorter sequences. By doing so, we can both increase the number of samples and facilitate model training. For instance, in the domains of human activity recognition, sleep stage classification, and machine fault diagnosis, sliding windows of size 128, 3000, and 5120 are often used, respectively.

Data splitting. Next, we divide the data into training and testing sets. Specifically, we split the data from each subject into stratified splits of 70%/30%, ensuring that the test set includes samples from all classes in the dataset. It is worth noting that we do not use a validation set for either the source or target domains. A validation set is typically used to select the best hyperparameters for the model, but we use four risk calculation methods that only require the source training data, source testing data, and target training data to select the best model.

Normalization. Normalization is a crucial step in the training process of deep learning models, as it can help to accelerate convergence and improve performance. In this work, we apply Z-score normalization to both the training and testing splits of the data, using the following equation:

$$x_i^{\text{normalize}} = \frac{x_i - x^{\text{mean}}}{x^{\text{std}}}, \quad i = 1, 2, \ldots, N \tag{12}$$

where $N = N_s$ for the source domain data and $N = N_t$ for the target domain data. Note that both the training and testing splits are normalized based on the training set statistics only [8, 23].

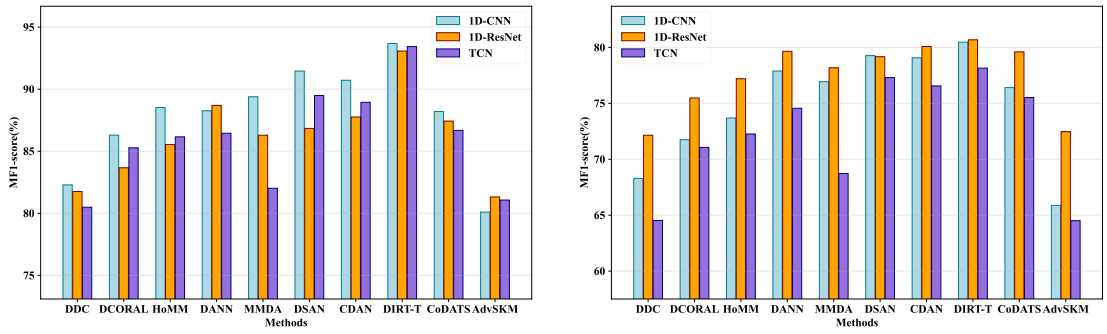
3.7.2 Standardized Training Scheme. All the training procedures have been standardized across all UDA algorithms. For instance, we train each model for 40 epochs, as performance tends to decrease with longer training. We report the model performance after the end of the last epoch. For model optimization, we use Adam optimizer with a fixed weight decay of 1e-4 and $(\beta_1, \beta_2) = (0.5, 0.99)$. The learning rate is set to be a tunable hyper-parameter for each method on each dataset. We exclude any learning rate scheduling schemes from our experiments to ensure that the contribution is mainly attributed to the UDA algorithm.

3.7.3 Hyper-parameter Search. For each algorithm/dataset combination, we conduct an extensive random hyper-parameter search with 100 hyper-parameter combinations. The hyper-parameters are picked by a uniform sampling from a range of predefined values. Details about the specified ranges can be found in Table S.1 of the Appendix. For each set of hyper-parameters, we calculate the values of the four risks for three different random seeds. We pick the model that achieves the minimum risk value for each model selection strategy. To calculate the FST risk, we use five samples per class from each dataset.

3.7.4 Evaluation Metric. Since time series data are usually imbalanced, where some classes might be totally absent from some subjects (see Fig. S.1b in the Appendix), the accuracy metric may not be representative of the performance of the UDA methods. Therefore, we report macro F1-scores instead, which takes into account how the data is distributed and avoids predicting false negatives.

4 RESULTS AND DISCUSSIONS

In this section, we first investigate the contributions of different backbone networks to the performance of UDA algorithms. Subsequently, we study the performance of different model selection techniques on the benchmark datasets. Last, we discuss the main findings of our experiments.



(a) UCIHAR dataset (b) HHAR dataset

Fig. 2. Comparison between 1D-CNN, 1D-ResNet, TCN backbones applied on UCIHAR and HHAR datasets.

4.1 Evaluation of Backbone Networks

To investigate the impact of the backbone networks on the models' performance, we evaluate all the UDA algorithms under three different backbone networks. Particularly, we employ 1D-CNN, 1D-ResNet, and TCN (described in Section 3.4) as backbone networks. To better evaluate the performance of different backbone networks, we experimented on datasets with different scales, i.e., the small-scale UCIHAR and the large-scale HHAR datasets. We reported the average performance of all cross-domain scenarios in the adopted datasets, as shown in Fig. 2. Clearly, the absolute performance varies significantly across different backbone networks for the same UDA method. Nonetheless, the relative performance between different UDA methods remains stable across the three backbone networks. For instance, while DIRT-T consistently performs best, DDC has the lowest performance with respect to all the other UDA methods regardless of the utilized backbone networks. On the other hand, a comparison of the performance of 1D-CNN and 1D-ResNet on the UCIHAR and HHAR datasets reveals that 1D-CNN tends to perform better on the UCIHAR dataset, while 1D-ResNet tends to perform better on the HHAR dataset. This difference in performance may be due to the fact that a more complex model such as 1D-ResNet is better able to take advantage of the larger size of the HHAR dataset, leading to improved performance [1].

We also conducted additional experiments using Long-Short Term Memory (LSTM) and compared its performance to other CNN-based models on 10 different methods for unsupervised domain adaptation on the UCIHAR dataset. Our results show that LSTM performed significantly worse than all the other CNN-based approaches. This may be due to the lower capacity of LSTM at modeling local patterns and producing class-discriminative features [37], as well as its difficulty in handling long-term dependencies [47], which are common in many time series applications. Detailed results of the LSTM experiment are provided in the supplementary materials.

4.2 Evaluation of Model Selection Strategies

In this experiment, we evaluate the performance of various model selection approaches i.e., SRC, DEV, FST, and TGT (described in Section 3.6) on the UDA methods. To do so, we first select the backbone network to be 1D-CNN due to its stable performance and computational efficiency. Then, for all the UDA algorithms, we select the best model according to each model selection strategy while testing its performance on the target domain data. Table 4 shows the average F1-score performance across all the cross-domain scenarios spanning five different datasets (detailed versions can be found in Tables S.5, S.6, S.7, S.8, and S.9 in the Appendix). The experimental results reveal the following conclusions. First, the performance of a UDA method can vary depending on the model selection strategy used. It is therefore important to use the same model selection strategy when comparing different UDA methods. For example, in the WISDM dataset, the DANN method performs best when the model is selected based on the target risk, while the CoDATS method performs best when the model is selected based on the source risk. Second, in comparing various strategies for selecting a model with respect to the target risk, it appears that SRC and FST risks are effective for datasets with balanced class distributions, such as UCIHAR and HHAR. On the other hand, DEV risk tends to work better for imbalanced datasets, such as WISDM and MFD, although it may require more computational resources compared to other methods.

To summarize, in situations where obtaining target labels is cost-prohibitive, both the source and DEV risks offer viable solutions as they do not require any target labels. The appropriate choice between the two depends on the characteristics of the dataset and the availability of the computational resources. While the DEV risk is more robust on class-imbalanced datasets, the source risk can be more computationally feasible. On the other hand, if only a small amount of labeled data from the target dataset is available, the few-shot risk can be the best choice as it achieves competitive performance with the target risk using only a small number of labeled samples, given that the target dataset has balanced classes.

4.3 Discussions

AdaTime provides a unified framework to evaluate different UDA methods on time series data. To explore the advantage of one UDA method over the others, we fixed the backbone network, the evaluation schemes, and the model selection strategy. We unveil the following insights.

Domain gap of different datasets. We conducted the experiments on two small-scale datasets, and two large-scale datasets. Regardless of the dataset size, all the adopted datasets suffer a considerable domain gap, as shown in Table 5. This table provides the results of the target-only experiment (i.e., training the model with target domain training set and testing it directly on the target domain test set), and the source-only experiment (i.e., training the model with source domain training set and testing it directly on the target domain test set). The backbone network for both experiments is the 1D-CNN network. While the source-only represents the lower-bound performance, the target only represents the upper-bound performance, and the domain gap is represented by their difference.

Visual UDA methods achieve comparable performance to TS-UDA methods on time series data. With further exploration of Table 4, we find that surprisingly the performance of visual UDA methods is competitive or even better than TS-UDA methods. This finding is consistent for all the model selection strategies across the benchmarking datasets. For example, with the TGT risk value, we find that the methods proposed for visual applications such as DIRT-T and DSAN perform better than CoDATS and AdvSKM on the four datasets. A possible explanation is that all the selected UDA algorithms are applied on the vectorized feature space generated by the backbone network, which

Table 4. The average results (for 10 cross-domain scenarios) according to the minimum risk value in terms of MF1-score applied on 1D-CNN backbone.

Dataset	RISK	DDC	Deep	HoMM	DANN	MMDA	DSAN	CDAN	DIRT	CoDATS	AdvSKM	SASA	Avg/Risk
UCIHAR	SRC	81.56	85.52	86.89	87.31	84.72	87.91	85.98	**89.64**	84.30	78.10	82.81	85.19
	DEV	74.52	79.08	78.69	84.26	88.49	88.70	84.10	**92.34**	78.69	80.01	81.68	82.89
	FST	81.64	86.30	88.52	86.59	87.96	**91.46**	88.87	90.79	86.99	80.00	84.02	86.91
	TGT	82.29	86.30	88.52	88.26	89.39	91.46	90.72	**93.68**	88.20	80.10	85.00	87.89
WISDM	SRC	51.89	51.13	51.98	56.49	57.24	58.99	41.28	53.23	**60.08**	50.98	47.79	53.33
	DEV	50.86	51.54	54.22	62.41	56.86	**60.12**	46.46	53.23	58.85	50.76	50.24	54.53
	FST	53.64	53.2	54.05	46.39	55.19	**57.10**	55.59	56.06	45.88	53.72	46.30	53.08
	TGT	53.78	54.19	56.92	**62.41**	59.82	61.08	55.59	59.59	62.06	54.82	53.34	58.03
HHAR	SRC	68.14	69.99	70.00	76.11	66.94	75.67	77.32	**78.56**	73.89	65.32	74.17	72.44
	DEV	65.07	65.16	67.68	61.74	69.32	76.67	76.91	**80.34**	72.6	65.20	72.20	70.07
	FST	67.91	71.75	73.69	73.56	75.10	77.98	76.06	**79.90**	71.28	64.61	75.73	73.18
	TGT	68.29	71.75	73.69	77.89	76.93	79.27	79.07	**80.47**	76.41	65.88	75.76	74.97
SSC	SRC	60.82	60.64	60.60	60.80	60.60	57.91	57.64	60.82	56.00	**61.18**	57.94	59.70
	DEV	**60.84**	56.16	54.52	60.80	58.63	59.37	57.99	52.49	54.61	56.19	57.94	57.16
	FST	60.86	60.84	60.61	60.80	**61.14**	58.60	53.95	59.40	55.64	61.10	56.37	59.29
	TGT	60.88	61.05	60.81	60.80	**63.47**	59.51	59.51	61.38	57.32	61.18	59.81	60.59
MFD	SRC	79.69	79.99	80.46	80.91	84.20	52.62	72.44	**91.31**	76.98	79.76	76.47	77.84
	DEV	80.21	78.58	79.78	81.95	82.45	80.11	78.34	**87.37**	80.29	81.19	77.59	81.03
	FST	79.69	79.99	80.17	80.91	84.20	74.82	64.89	**91.31**	81.73	79.76	77.74	79.75
	TGT	81.54	80.80	81.18	84.06	85.44	81.65	84.64	**92.81**	84.20	81.47	78.94	83.78

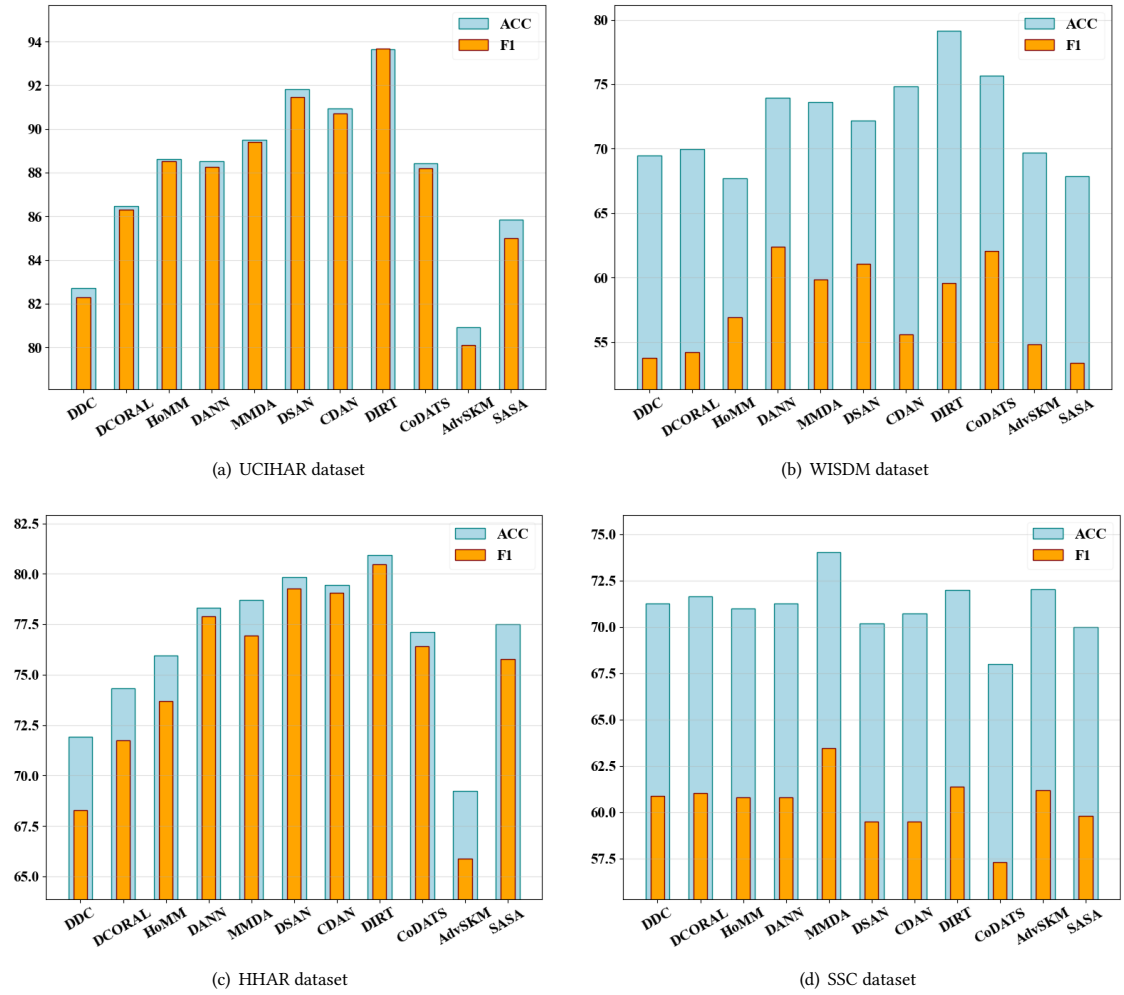
Table 5. Domain gap between source and target domains described by the difference between lower and upper-performance bounds on 1D-CNN backbone.

	UCIHAR	WISDM	HHAR	SSC	MFD
Same Domain (Target-only)	100.00	98.02	98.55	72.09	99.39
Cross-Domain (Source-only)	65.94	48.60	63.07	51.67	72.51
Gap (δ)	37.32	49.44	33.86	18.38	26.88

is independent of the input data modality. This finding suggests that with a standard backbone network, visual UDA algorithms can be strong baselines for TS-UDA.

Methods with joint distribution alignment tend to perform consistently better. Table 3 illustrates that some methods address the marginal distribution such as MMD, CORAL, and HoMM, while the others address the joint distributions (i.e., both marginal and conditional distributions concurrently) such as DIRT-T, MMDA, and DSAN. The results shown in Table 4 suggest that the methods addressing the joint distribution outperform those addressing the marginal distribution. For example, the best performing method, as selected by the TGT risk, is DIRT-T in UCIHAR and HHAR datasets, and MMDA in the case of the SSC dataset Similarly, with respect to different risks, DIRT-T, MMDA, and DSAN interchangeably achieve the best performance across the benchmarking datasets. Hence, considering the conditional distribution when designing the UDA algorithm is beneficial to the performance.

Accuracy metric should not be used to measure performance for imbalanced data. It is well known that accuracy is not a reliable metric for evaluating the performance of classifiers on imbalanced datasets. Despite this, many existing TS-UDA methods continue to use accuracy as a measure of performance in their evaluations[3, 8, 23]. In



(a) UCIHAR dataset

(b) WISDM dataset

(c) HHAR dataset

(d) SSC dataset

Fig. 3. Results of best models according to the target risk for different UDA methods in terms of accuracy and macro average F1-score.

our experiments, we found that using either accuracy or F1-score alone can lead to inconsistent results on imbalanced datasets such as WISDM. This highlights the need to consider the imbalanced nature of most time series data when evaluating classifier performance. To illustrate this point, we present the results of our experiments in terms of both accuracy and F1-score on four datasets: WISDM, SSC, UCIHAR, and HHAR. WISDM and SSC are imbalanced, while UCIHAR and HHAR are mostly balanced. As shown in Fig. 3, on the imbalanced WISDM dataset (Fig. 3(b)), CDAN achieves higher accuracy than some other methods such as DDC, MMDA, and DSAN, but has one of the worst performance in terms of F1-score. In contrast, the results on the balanced UCIHAR dataset (Fig. 3(a)) show that accuracy can still be a representative measure of performance and is similar to F1-score. Therefore, we recommend using the F1-score as a performance measure in all TS-UDA experiments.

Effect of labeling budget on the Few-shot risk performance To investigate the influence of the number of labeled samples on few-shot learning performance, we conducted additional experiments using 10 and 15 labeled samples per class on the UCIHAR dataset. The results, shown in the table below, show that the overall performance was relatively consistent regardless of the size of the few-shot labeled sample, indicating the robustness of few-shot learning to this hyperparameter.

Table 6. Model selection based on few-shot target risk under a different budget of labeled samples on UCIHAR dataset.

# Labels	DDC	DCORAL	HoMM	DANN	MMDA	DSAN	CDAN	DIRT-T	CoDATS	AdvSKM	SASA	Avg/risk
5 SHOTS	81.64	86.3	88.52	86.59	87.96	91.46	88.87	90.79	86.99	80.00	84.02	86.65
10 SHOTS	81.53	86.3	88.52	87.25	88.69	90.97	88.87	90.79	87.73	80.00	84.02	86.79
15 SHOTS	81.53	86.3	88.52	86.13	87.96	91.46	88.87	90.79	87.73	80.00	84.02	86.66

Limitations and future works One limitation of our study is that it only focuses on time series classification. In future work, we plan to extend the scope of our benchmarking to include time series regression and forecasting tasks. Additionally, we have only considered the closed-set domain adaptation scenario, where the source and target classes are similar. In future work, we aim to also consider partial and open-set domain adaptation scenarios, which are common in time series applications and involve a varying set of classes between the source and target domains.

5 CONCLUSIONS AND RECOMMENDATIONS

In this work, we provided AdaTime, a systematic evaluation suite for evaluating existing domain adaptation methods on time series data. To ensure fair and realistic evaluation, we standardized the benchmarking datasets, evaluation schemes, and backbone networks among different domain adaptation methods. Moreover, we explored more realistic model selection approaches that can work without any target domain labels or with only few-shot labeled samples. Based on our systematic study, we provide some recommendations as follows. First, visual UDA methods can be applied on time series data and are strong candidate baselines. Second, we can rely on more realistic model selection strategies, that do not require target domain labels such as source risk and DEV risk, to achieve reliable performance. Third, we recommend conducting experiments with large-scale datasets to obtain reliable results, with fixing the backbone network among different UDA baselines. We also suggest adopting the F1-score instead of accuracy as a performance measure to avoid any misleading results with imbalanced datasets. Lastly, we believe that incorporating time series-specific domain knowledge into the design of UDA methods has the potential to be beneficial moving forward

REFERENCES

[1] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[2] Jason Lines, Sarah Taylor, and Anthony Bagnall. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data*, 12(5), 2018.

[3] Youngjae Chang, Akhil Mathur, Anton Isopoussu, Junehwa Song, and Fahim Kawsar. A systematic study of unsupervised domain adaptation for robust human-activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–30, 2020.

[4] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2352–2359, 2021.

[5] Yangfan Li, Kenli Li, Cen Chen, Xu Zhou, Zeng Zeng, and Keqin Li. Modeling temporal patterns with dilated convolutions for time-series forecasting. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(1):1–22, 2021.

[6] Yi Li, Yang Sun, Kirill Horoshenkov, and Syed Mohsen Naqvi. Domain adaptation and autoencoder-based unsupervised speech enhancement. *IEEE Transactions on Artificial Intelligence*, 3(1):43–52, 2021.

[7] Arun Kumar Sharma and Nishchal K Verma. Quick learning mechanism with cross-domain adaptation for intelligent fault diagnosis. *IEEE Transactions on Artificial Intelligence*, 2021.

[8] Garrett Wilson, Janardhan Rao Doppa, and Diane J. Cook. Multi-source deep domain adaptation with weak supervision for time-series sensor data. In *SIGKDD*, 2020.

[9] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, Xiaoli Li, and Cuntai Guan. Adast: Attentive cross-domain eeg-based sleep staging framework with iterative self-training. *IEEE Transactions on Emerging Topics in Computational Intelligence*, pages 1–12, 2022.

[10] Shuteng Niu, Yongxin Liu, Jian Wang, and Houbing Song. A decade survey of transfer learning (2010–2020). *IEEE Transactions on Artificial Intelligence*, 1(2):151–166, 2020.

[11] Chaoyue Wang, Chang Xu, and Dacheng Tao. Self-supervised pose adaptation for cross-domain image animation. *IEEE Transactions on Artificial Intelligence*, 1(1):34–46, 2020.

[12] Hanrui Wu and Michael K Ng. Multiple graphs and low-rank embedding for multi-source heterogeneous domain adaptation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(4):1–25, 2022.

[13] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, pages 443–450. Springer, 2016.

[14] Chao Chen, Zhihang Fu, Zhihong Chen, Sheng Jin, Zhaowei Cheng, Xinyu Jin, and Xian-Sheng Hua. Homm: Higher-order moment matching for unsupervised domain adaptation. *AAAI*, 2020.

[15] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[16] Yongchun Zhu, Fuzhen Zhuang, Jindong Wang, Guolin Ke, Jingwu Chen, Jiang Bian, Hui Xiong, and Qing He. Deep subdomain adaptation network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 32(4):1713–1722, 2021.

[17] Erheng Zhong, Wei Fan, Jing Peng, Kun Zhang, Jiangtao Ren, Deepak Turaga, and Olivier Verscheure. Cross domain distribution adaptation via kernel mapping. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1027–1036, 2009.

[18] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[19] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[20] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018.

[21] Mohamed Ragab, Emadeldeen Eldele, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. Self-supervised autoregressive domain adaptation for time series data. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2022.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680, 2014.

[23] Qiao Liu and Hui Xue. Adversarial spectral kernel matching for unsupervised time series domain adaptation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021.

[24] Mohamed Ragab, Zhenghua Chen, Min Wu, Haoliang Li, Chee-Keong Kwoh, Ruqiang Yan, and Xiaoli Li. Adversarial multiple-target domain adaptation for fault classification. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2020.

[25] Jinpeng Li, Shuang Qiu, Changde Du, Yixin Wang, and Huiguang He. Domain adaptation for eeg emotion recognition based on latent representation similarity. *IEEE Transactions on Cognitive and Developmental Systems*, 12:344–353, 2020.

[26] Sanjay Purushotham, Wilka Carvalho, Tanachat Nilanon, and Yan Liu. Variational recurrent adversarial deep domain adaptation. In *ICLR 2017 : International Conference on Learning Representations 2017*, 2017.

[27] Michele Tonutti, Emanuele Ruffaldi, Alessandro Cattaneo, and Carlo Alberto Avizzano. Robust and subject-independent driving manoeuvre anticipation through domain-adversarial recurrent neural networks. *Robotics and Autonomous Systems*, 115:162–173, 2019.

[28] Kaichao You, Ximei Wang, Mingsheng Long, and Michael Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 7124–7133. PMLR, 2019.

[29] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks*, pages 437–442, 2013.

[30] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *Sigkdd Explorations*, 12(2):74–82, 2011.

[31] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller 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*, pages 127–140, 2015.

[32] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 101(23):215–220, 2000.

[33] Emadeldeen Eldele, Zhenghua Chen, Chengyu Liu, Min Wu, Chee-Keong Kwoh, Xiaoli Li, and Cuntai Guan. An attention-based deep learning approach for sleep stage classification with single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2021.

[34] Christian Lessmeier, James Kuria Kimotho, Detmar Zimmer, and Walter Sextro. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification. In *PHM Society European Conference*,

volume 3, pages 05–08, 2016.

[35] Mohamed Ragab, Zhenghua Chen, Wenyu Zhang, Emadeldeen Eldele, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. Conditional contrastive domain generalization for fault diagnosis. *IEEE Transactions on Instrumentation and Measurement*, 71:1–12, 2022.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[37] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[38] Markus Thill, Wolfgang Konen, and Thomas Bäck. Time series encodings with temporal convolutional networks. In *International Conference on Bioinspired Methods and Their Applications*, pages 161–173. Springer, 2020.

[39] Sicheng Zhao, Xiangyu Yue, Shanghang Zhang, Bo Li, Han Zhao, Bichen Wu, Ravi Krishna, Joseph E Gonzalez, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, et al. A review of single-source deep unsupervised visual domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[40] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):1–35, 2016.

[41] Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018.

[42] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(1):723–773, 2012.

[43] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. On minimum discrepancy estimation for deep domain adaptation. *Domain Adaptation for Visual Understanding*, 2020.

[44] Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*, pages 153–171. Springer, 2017.

[45] Ruichu Cai, Jiawei Chen, Zijian Li, Wei Chen, Keli Zhang, Junjian Ye, Zhuozhang Li, Xiaoyan Yang, and Zhenjie Zhang. Time series domain adaptation via sparse associative structure alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6859–6867, 2021.

[46] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *ICML'17 Proceedings of the 34th International Conference on Machine Learning*, pages 2988–2997, 2017.

[47] Colin Lea, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In *Computer Vision – ECCV 2016 Workshops*, pages 47–54, Cham, 2016. Springer International Publishing.