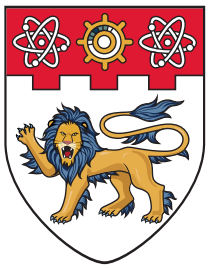
Time Series Domain Adaptation via Contrastive Adversarial Domain Disentangled Network



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Huang Xinyi

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Master of Engineering (M.Eng)

2023

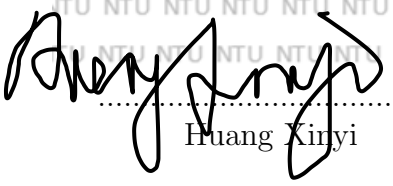
Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

2023/04/18

.....

Date

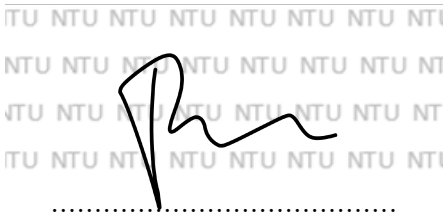


.....

Huang Xinyi

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.



Apr. 18, 2023
Date

Sinno Jialin Pan

Authorship Attribution Statement

This thesis does not contain any materials from papers published in peer-reviewed journals or from papers accepted at conferences in which I am listed as an author.

2023/4/18

....

Date Huang Xinyi

Abstract

Unsupervised domain adaptation is a machine learning framework to transform information learned from one or several source domains with many annotated samples to unlabeled target domains. A typical unsupervised domain adaptation method is typically designed base on visual data. Solutions on time series data are less explored. Most existing methods are based either on mapping representations from one domain to the other or learning the features invariant to the domains by statistical restrictions. However, focusing only on the mapping or invariant features may leave the methods vulnerable to noise since they ignore much individual information. In this work, we propose Contrastive Adversarial Domain Disentangled Network (CADDN), a novel method that explores improving the model adaptation performance on time series data by exploiting each domain's specific properties. Our primary motivation is to construct a framework that can learn while jointly disentangling the domain-invariant and the domain-specific features in the mean time. Contrastive learning is applied in the optimization of the domain-specific features, targeting a beneficial and stable feature extraction. Comprehensive experimental evaluations are conducted on four benchmark time series datasets to demonstrate the superiority of the proposed method over state-of-the-art domain adaptation solutions. A further ablation study validates the hypothesis that adding contrastive-based domain-specific feature extraction will largely improve the performance compared to only focusing on domain-invariant knowledge.

Acknowledgments

I would like to express my gratitude to Prof. Sinno Jialin Pan, Dr. Zhenghua Chen Dr. Wenmian Yang, Dr. Mohamed Raged.

Contents

Abstract iv
Acknowledgments v
List of Figures viii
List of Tables ix

1 Introduction **1**
1.1 Organization of the Thesis 3

2 Background **4**
2.1 Multi-layer Perceptron 4
2.2 Convolutional Neural Networks 6
2.3 Recurrent Neural Network 6
2.4 Transformers 7

3 Literature Review **8**
3.1 A Survey on Unsupervised Domain Adaptation 8
3.1.1 Instance Re-weighting Adaptation 8
3.1.2 Feature Adaptation 9
3.1.3 Deep Network Adaptation 10
3.1.4 Adversarial Adaptation 11
3.2 Unsupervised Domain Adaptation on Time Series 12

4 The Proposed CADDN Method **14**
4.1 Problem Formulation 14
4.2 Overview of CADDN 14
4.3 Latent feature Disentanglement 15

4.4 The Contrastive loss for learning Domain-specific Features 17
4.5 The Independence Loss 18
4.5.1 Overall Objective Function 19

5 Experiments **21**
5.1 Datasets 21
5.2 Baselines 22
5.3 Implementation Details 23
5.4 Experiments 23
5.4.1 Ablation Study on Network Structure 24
5.5 Sensitivity Analysis 25
5.6 Domain-specific Feature Space Visualization 26
5.7 Domain-invariant Feature Space Visualization. 26

6 Conclusion **29**

References **30**

List of Figures

2.1 The architecture of a typical Multi-layer perceptron model. 5
2.2 The architecture of a typical CNN model. 6
2.3 The architecture of a typical RNN model. 7

3.1 The temporal covariate shift problem in nonstationary time series. . . . 13

4.1 The architecture of the proposed CADDN model. 15

5.1 Comparison between Ablation models(A, B, C, D) applied on the UCI-HAR, HHAR, and WISDM datasets by measuring macro F1-score. . . . 25
5.2 Sensitivity analysis in WISDM dataset 26
5.3 The domain-specific feature distribution with Gaussian kernel density estimation. (a) the domain-specific feature distribution of the source and target mixed data.(b) the domain-specific feature distribution of the target domain.(c) the domain-specific feature distribution of the source domain. 27
5.4 Domain-invariant feature spaces Visualization. (a) t-SNE embedding of the learned domain-invariant features by DDC. (B) t-SNE embedding of the learned domain-invariant features by CADDN. 28
5.5 The t-SNE plot of the features from Source domain (blue) and the Target domain (red) (a) apply the model trained by the source domain directly to the target domain (b) use the DDC model for domain adaptation (c) the domain-invariant features trained from the CADDN model 28

List of Tables

5.1 Basic information of datasets. 21
5.2 The mean Macro F1-score for each dataset. 24
5.3 Ablation study 24

Chapter 1

Introduction

With the rapid development of manufacturing and information technologies, the increasing availability of sensor-based time series data is changing the way of making decisions in many real-world areas. For example, due to the prevalence of wearable devices like mobile phones and smartwatches, sensor-based human activity recognition has become more and more pervasive. By classifying the activities conducted by the user, human activity recognition can contribute to a wide range of applications, such as healthcare, gait recognition, and smart home. In sensor-based human activity recognition, traditional machine learning methods are labor-intensive since they require massive manually labeled training samples for every new user. That is because such methods are typically based on the assumption that data samples employed in the training and testing phases are generated from the same unknown distribution. Due to the uniqueness of people's activity characteristics, each person's data should be treated as a single domain. The performance plummets when simply applying the trained model to data collected from unseen new participants because of the domain gaps.

Transfer learning, which targets at leveraging labeled data in source domains to prompte new learning tasks in target domains, were designed to solve the lack of labeled data problem. Unsupervised Domain adaptation (UDA) is a particular case of transfer learning (TL) [25] that employs knowledge learned from one annotated source domain to one unlabeled target domain. To solve the domain shift problem, most state-of-the-art UDA methods are based on the motivation to capture the domain-invariant representation for both source and target domains. One prevailing paradigm aims to extract domain invariant representations by using Maximum Mean Discrepancy (MMD)

on the Reproducing Kernel Hilbert Space (RKHS) [38] [37]. Another promising paradigm mitigates the domain gap by encouraging domain confusion through an adversarial objective. [22] [11] [14]. All these methods try to minimize the domain gaps by directly mapping the features to a common subspace. However, compared to static vision data, the domain alignment process for sensor-based time series data is always more unstable. For example, the different persons' varying moving patterns and health conditions in human activity recognition can greatly change the time-series pattern. These factors, which we call domain-specific features, are highly related to the domains and will hurt the feature alignment. To solve this problem, we advocate adding a feature disentanglement step before the feature alignment cross domains.

In this paper, we propose a novel method for the unsupervised domain adaptation problem based on sensor-based time series data, which we call Contrastive Adversarial Domain Disentangled Network (CADDN). Instead of only focusing on learning the domain-invariant representations by feature alignment, we seek to learn the domain-specific information for each domain. Our method is based on the intuition that domain-specific information is independent of the causal mechanism related to the ground-truth labels. These factors will hurt the final results when included in feature alignment for domain adaptation. Thus it is meaningful to disentangle domain-specific information from domain-invariant information before feature alignment.

Besides, previous methods typically leverage only domain labels to learn the domain-specific information, which is usually limited and insufficient. To improve this problem, we innovatively combined Contrastive learning in our approach to promote the learning of domain-specific properties. By inducing uniformity over the mixture of the source and target domain-specific features distribution and then clustering each domain on the unit hyper-sphere, the domain private feature extractors can learn features that vary from domains and are very similar when from the same domain. Meanwhile, we use a shared feature extractor to learn the features invariant across the domains. After an independence disentanglement step that minimizes the mutual information from both feature extractors, we finish the feature disentanglement. Finally, to ensure the common feature extractor can learn all the features shared by the domains, we employ a shared decoder to decode the concatenation of the domain-specific and the domain-invariant

features. As a result, our method is more robust to the contamination by the factors correlated with the domains and helps to solve the domain shift problem.

In addition to sensor-based human activity recognition, we validate our approach to other time series applications, such as machine fault diagnosis applications. Unlike human activity recognition data samples, time series data from real-world machine health diagnosis is inevitably long, making the RNN-based model training really time-consuming. Our method is based on 1-dimensional convolutional neural networks (1D-CNN) and thus is more effective on GPU than models like RNN. Extensive experimental results demonstrate that our CADDN outperforms the state-of-the-art unsupervised domain adaptation methods on four real-world datasets.The contributions of this work are summarized as follows.

- We propose an end-to-end domain adaptation approach for time series data.
- Different from existing methods, we do the information disentanglement before the domain feature alignment.
- We innovatively combined contrastive learning in the domain-specific feature learning process to maximize the utilization of domain information.

1.1 Organization of the Thesis

This thesis is organized as follows.

Chapter 2 introduces the three basic classes of deep neural network in Deep Learning.

Chapter 3 reviews the important algorithms of Unsupervised Domain adaptation and UDA in time series.

Chapter 4 proposes a novel algorithm of domain adaptation in time series data.

Chapter 5 shows the experiments results on three commonly used human activity recognition datasets and one machine fault diagnosis dataset.

Chapter 6 gives the conclusion.

Chapter 2

Background

This chapter provides an overview of the background knowledge of Deep Machine Learning by introducing three classic models: Multi-layer Perceptron, Convolutional Neural Networks, Recurrent Neural Networks, and Transformers. While Recurrent Neural Networks and Transformers are commonly used in time series data, we have chosen to use 1D-CNN as the base model in our method to save time. This decision is based on the fact that RNN and Transformer models can be computationally expensive, especially for long time series data, which is often encountered in machine maintenance applications.

2.1 Multi-layer Perceptron

Artificial neural networks (ANNs), inspired by neural systems in animal brains, have greatly improved the performance of previous artificial intelligence by employing interconnected neurons. The multi-layer perceptron is one of the most widely used types of ANN architecture. A typical Multi-layer perceptron model has three components: an input layer, multiple hidden layers, and an output layer. Expect for the input layer, Each node of these layers is a neuron with an activation function. According to [28], the computing power of these models mainly comes from the non-linear activation function. Simply increasing the number of the hidden without the activation functions will not improve the performance. That is because a function of linear functions will still be a linear function. When the activation functions are non-linear, it has been proven that a two-layer perceptron is capable to approximate any continuous function.

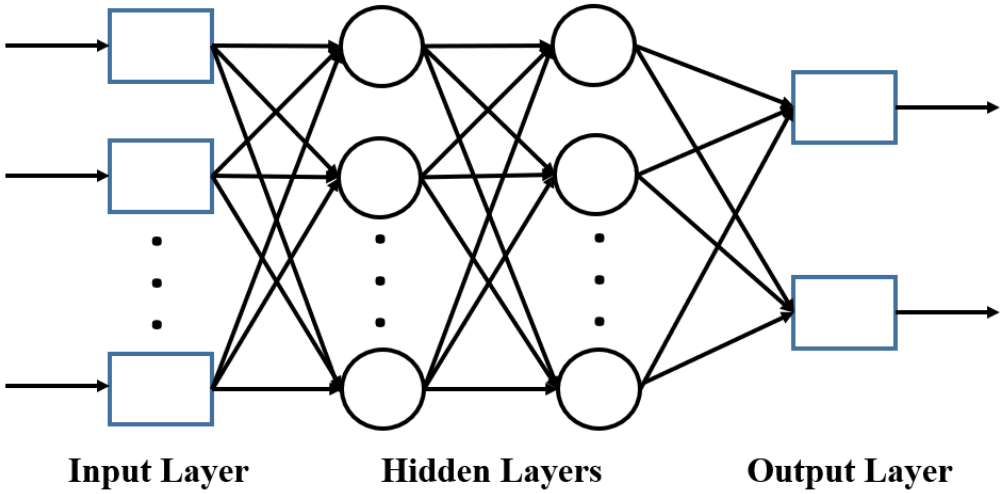


Fig. 2.1: The architecture of a typical Multi-layer perceptron model.

There are many non-linear activation functions applied to the multi-layer perceptron. For example, the sigmoid function is widely used, especially when the model needs to predict the probability. Since its range and the probability are both always between 0 and 1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

The sigmoid activation function is differentiable with a smooth gradient. However, its gradient approaches zero when the value is too large or too small, which may cause the vanishing gradient problem. To solve this problem, the ReLU activation function is proposed.

$$f(x) = max\{0, x\} \tag{2.2}$$

The main catch is that this function promises that neurons will be deactivated if their outputs are less than zero, thus improving computing efficiency and promoting the convergence.

2.2 Convolutional Neural Networks

Convolutional Neural Network (CNN) is another impressive type of ANN Model. With precise architecture. They are primarily used to solve complex image tasks.

Generally speaking, CNNs are stacked by three types of layers: fully-connected layers, pooling layers, and convolutional layers. The convolutional layers employ a set of filters to determine the output by calculating the scalar product of the weights and the connected regions. The filters' parameters are learned through training. Compared to fully-connected layers adopted in multi-layer perceptron, the convolutional layers significantly reduce the complexity of the model. The pooling layers target at downsampling and reducing the output dimension by combining the clustered outputs into a single one. Finally, the fully-connected layers will do the same things as in Multi-layer perceptron, That is to produce the final score wanted by the classification or regression tasks. Generally, the ReLU function will be adopted to improve the model performance.

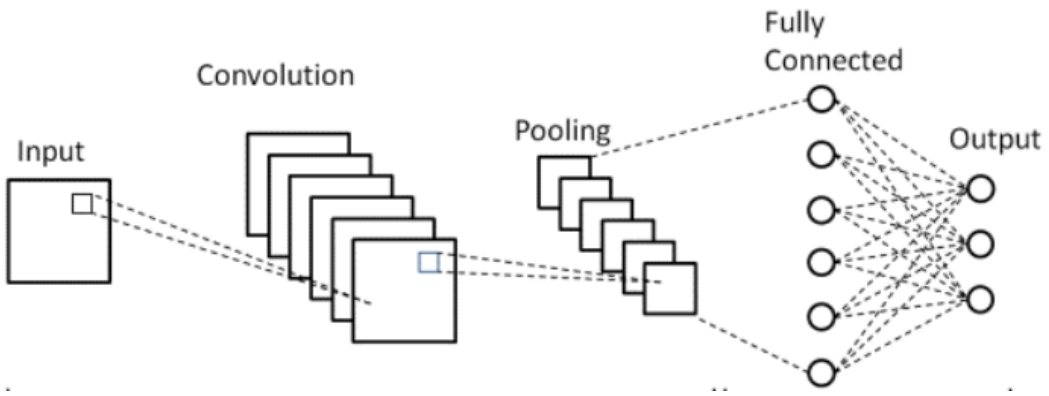


Fig. 2.2: The architecture of a typical CNN model.

2.3 Recurrent Neural Network

Recurrent neural networks (RNNs), generally proposed for time series data, are a type of neural networks that employ previous outputs as inputs to the analysis of the next timestamp by using hidden states. Instead of adopting multiple hidden layers, RNNs construct only one hidden state and loop over it through the sequence. In other words,

for each element of the input sequence, RNNs use the same weights to process the information. Therefore, the model are able to generalize to sequences of different lengths with fewer parameters.

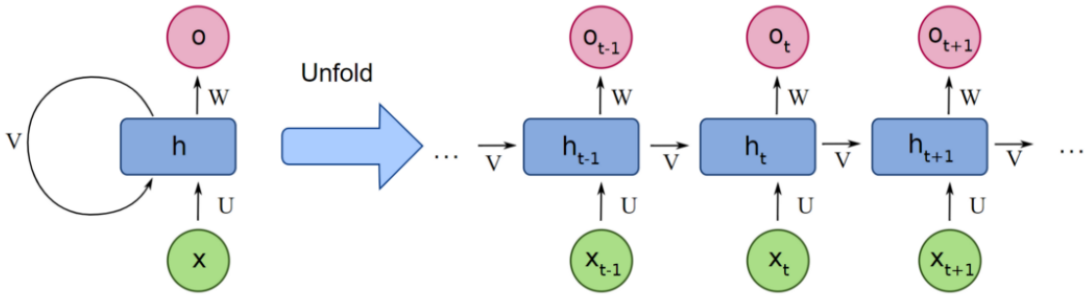


Fig. 2.3: The architecture of a typical RNN model.

Specifically, for each timestamp x_t in the input sequence,

$$h_t = G_1(Ux_t + Vh_{t-1} + b_1) \tag{2.3}$$

$$o_t = G_2(Wh_t + b_2) \tag{2.4}$$

where U, V, b_1, b_2 are shared parameters and G_1, G_2 are activation functions.h_t represents the stored memory of the model and is fed into the next hidden state. In this way, the RNNs imitate the humane brains to process sequence information.

2.4 Transformers

Transformers, like recurrent neural networks (RNNs), are designed for processing sequential data such as natural language in tasks like translation and text summarization. However, unlike RNNs, transformers process the entire input simultaneously using the attention mechanism, which provides context for any position in the input sequence. This means that transformers do not have to process one word at a time, allowing for greater parallelization and reducing training times. As a result, transformers have become the model of choice for many NLP problems, surpassing RNN models like LSTM. The concept of using transformers in language processing can be traced back to Schmidhuber's work in 1992, and the increased training parallelization has enabled the development of pretrained systems like BERT and GPT, which are fine-tuned for specific tasks using large language datasets like Wikipedia Corpus and Common Crawl.

Chapter 3

Literature Review

In this chapter, we first survey the advances of Unsupervised domain adaptation (UDA) methodologies in the past decade. Then we review and summarize the existing UDA methods applied to time series data.

3.1 A Survey on Unsupervised Domain Adaptation

Following previous works[43], we split the existing Unsupervised Domain Adaptation learning into five categories: 1) Instance Re-weighting Adaptation; 2) Feature Adaptation; 3) Deep Network Adaptation; 4)Adversarial Adaptation.

3.1.1 Instance Re-weighting Adaptation

To solve the domain gap problem, Instance Re-weighting Adaptation's objective is to diminish the probability distribution discrepancy between source and target by re-weighting the source samples. It mainly focuses on cases where the gap between the source and target is not too large. When the gap is relatively large, instance re-weighting can also be combined with others.

Without loss of generality, instance re-weighting adaptation can be roughly divided into three categories based on the weighting scheme. 1) Intuitive Weighting; 2) Kernel Mapping Based Weighting; 3) Co-training Based Weighting.

First proposed for natural language processing, intuitive weighting aims at aligning the source and target distribution by tuning the weights of source samples. One prevailing intuitive instance re-weighting domain adaptation framework [16] applies four parameters

for measuring the distribution gap between the source and target. Another intuitive re-weighting method[7] advocates aligning the source subspace with the target subspace by performing PCA.

The kernel mapping based re-weighting methods advocate reducing the domain distribution difference by aligning the means of the source and target in a reproducing kernel Hilbert space (RKHS). They can be further divided into sample selection and distribution matching. The former use cluster assumption for sample selection. The latter aims at minimizing the kernel mean discrepancy in RKHS by learning source weights.

Finally, the Co-Training methods introduce learning two separate classifiers to characterize the dataset in two different views. By measuring the confidence in two classifiers, inputs will be selected for the training set. Some researchers[6] also employ adversarial training for better feature extraction.

3.1.2 Feature Adaptation

Feature adaptation aims at finding the common feature representations from multiple domains. It can be roughly divided into four categories. 1) Feature Subspace-Based; 2) Feature Transformation-Based; 3) Feature Reconstruction-Based; 4) Feature Coding-Based.

Feature Subspace-Based methods assume that a low-dimensional Grassmann manifold is embedded in the high-dimensional raw data. Principal component analysis (PCA) was generally employed to construct the Grassmann manifold. The source and the target domains can be represented by two points, and the subspaces along the geodesic flow between the two points are used to find the intermediate representation.

Feature transformation-based methods aim to relieve the domain distribution difference by learning a transformation or projection of the data. Based on model formulation, these methods can be categorized into Metric, Projection, and Augmentation. Metric-Based methods employ first-order metric or second-order metric based on covariance instead of means[37] . Projection-Based methods adopt Kernel Matching Criterion , (e.g.,maximum mean discrepancy) or Discriminative Criterion. Augmentation-Based methods assume that the feature representation can be divided into the common representation, source-specific representation, and target-specific representation. Zero padding proposed with EasyAdapt (EA) [15] is famous among this type of method.

Feature Reconstruction-Based Adaptation aims to bridge the source and the target by low-rank or sparse reconstruction[**?**]n a latent subspace. Besides, some works also use adjacency matrices to construct the source and the target domains.

Feature coding based-transfer learning, also called domain adaptive dictionary learning, focuses on learning the representation dictionaries for each domain. It aims to seek the domain dictionaries without correspondences between domains.

3.1.3 Deep Network Adaptation

Deep Network Adaptation based on Deep neural networks (DNNs) has become a blowout trend in domain adaptation due to their end-to-end training mode and significant performance, especially in cases where massive data is available. Based on the network architectures, they can be divided into four categories: 1) Marginal Alignment Based; 2) Conditional Alignment Based; 3) Batch Normalization Layer-Derived; 4) Auto-encoder Based.

Marginal Alignment Based Deep Network Adaptation generally reduces the distribution disparity between the source and the target by transforming the top layer features to a reproducing kernel Hilbert space (RKHS) with maximum mean discrepancy (MMD) as the metric. JMMD [23] adopts a unified MMD for better characterizing the discrepancy between the labeled source and the unlabeled target. Besides, other metrics, such as Jensen-Shannon (JS) divergence[44], Wasserstein distance [], moment distance [20], and Kullback-Leibler (KL) divergence[26] were further explored to be included in domain adaptation.

Instead of focusing on the top layered feature matching in Marginal Alignment Based adaptation, Conditional Alignment Based Adaptation also considers high-level semantic information to improve the generalization. Theoretically, the below inequality provides an upper bound of the joint error.

$$\lambda = \min_{h \in \mathcal{H}} \epsilon_S(h, l_s) + \epsilon_T(h, l_t) \leq \min_{h \in \mathcal{H}} \epsilon_S(h, l_s) + \epsilon_T(h, l_s) + \epsilon_T(l_s, l_t)$$

However, the class misalignment may cause an unexpected large error $\epsilon_T(l_s, l_t)$, which will degrade the model adaptation performance. To solve this problem, conditional alignment-based models use clustering labels or progressively update pseudo-labels for target samples for the class alignment.

Batch normalization (BN), which standardizes the inputs to a layer for each mini-batch, has been proved to accelerate and stabilize the training process. When applied to the domain adaptation framework, it is natural to assume that data batches from different domains should have different means and variance. Therefore, Batch Normalization Layer-Derived adaptation [4] introduces to modulate the statistics from the source to the target in all Batch Normalization (BN) layers across the network.

Auto-encoder Based Adaptation are typically comprised of an encoder $f(\cdot)$ and a decoder $g(\cdot)$. Then the two components are trained together by minimizing the reconstruction loss. Auto-encoder has become a prevailing framework for unsupervised domain adaptation based on the assumption that the source data trained encoders and decoders can represent samples from the target domain. Representative works can be referred to as Denoising autoencoders (DAE) [40].

3.1.4 Adversarial Adaptation

Rooted from Generative adversarial Net (GAN)[12], Adversarial Adaptation has become a prominent idea for UDA. Instead of using MMD in RKHS space, adversarial learning generally adopts adversarial objectives like domain discriminator to minimize the domain discrepancy. There are three main streams of adversarial learning: 1) Gradient Reversal-Based; 2)Minimax Optimization-Based; 3) Generative Adversarial Net-Based.

Gradient reversal layer (GRL) was first proposed in 2015[10]. By adding a simple gradient reversal layer in the training network, researchers proved that the model could learn the task-discriminative and domain-invariant features using standard backpropagation. Researchers also extended the GRL to other frameworks and network architectures. For example, Dubey et al. proposed to use GRL for the fine-grained visual classification (FGVC) with the Siamese network. Yuhua Chen et al. combined GRL with the fast R-CNN networks.

Aside from GRL, Minimax optimization is another prevailing training strategy for adversarial models. Minimax optimization trains the domain discriminator against the feature extractor in the feature-level adaptation to encourage domain confusion. First proposed by Tzeng et al. [39] in 2015, Minimax has been widely applied by many famous works. Rozantsev et al.[33] introduced a network that includes auxiliary residual

networks trained by Minimax strategy to preserve necessary domain similarities. Long et al.[22] presented a principled framework that conditions the adversarial models on discriminative information conveyed in the classifier predictions. This method is also included in our experiments as a baseline.

Different from the above two streams, Generative Adversarial Net-Based (GAN-Based) methods aim to generate samples from the source domain to the target domain and are typically designed for visual applications. Hoffman et al.proposed a novel cycle consistent adversarial domain adaptation model (CyCADA)[13], which adapts representations at both the pixel-level and feature-level and does not require aligned pairs.

3.2 Unsupervised Domain Adaptation on Time Series

Unsupervised Domain Adaptation is essential in many real-world time series applications, such as sleep stage classification, sensor-based machine fault classification, human activity recognition, driving maneuver prediction, and climate prediction. However, time-series data-based UDA is less explored.

Unlike visual data-based methods, time series data-based UDA approaches emphasize effectively capturing the temporal dependencies. In [29], variational recurrent adversarial deep domain adaptation (VRADA) extended from DANN[11] are proposed to capture and transfer temporal latent dependencies of multivariate time series by exploiting the variational RNN[9].

Cai et al. [3] proposed a novel sparse associative structure alignment model for better time information characterization. Specifically, they first introduce an adaptive segment set to ease the obstacle of offsets. Then the intra-variables and inter-variables sparse attention mechanisms are employed to extract associative structure time-series data with time lags. Finally, they use the associative structure alignment to guide knowledge transfer across domains.

Wilson et al.[42] developed a new time-series compatible model called CoDATS to improve both accuracy and computational efficiency. This network uses 1D convolutions along the time axis to learn the temporal dependencies.

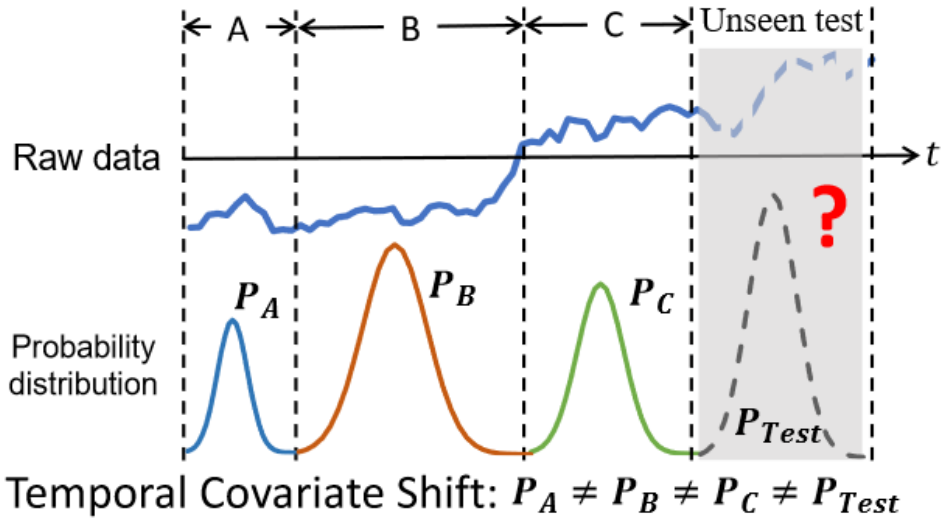


Fig. 3.1: The temporal covariate shift problem in nonstationary time series.

Domain adaptation in time series is a relatively unexplored area of research. However, AdaRNN stands out as a notable example of a representative work that addresses the challenging issue of Temporal Covariate Shift (TCS) in forecasting problems. AdaRNN aims to mitigate the distribution mismatch within time series sequences, thereby enhancing its forecasting performance.

Chapter 4

The Proposed CADDN Method

4.1 Problem Formulation

In our setting of time series domain adaptation problem, a domain D consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = x_1, x_2, ...x_n \in \mathcal{X}$. A task consists of two components: a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$ which is unknown but learnable.

Given a labelled Source domain data $D_S = \{X_S^i, Y_S^i\}_{i=1}^{n_S}$ with n_S samples, and an unlabelled target domain data $D_T = \{X_T^i\}_{i=1}^{n_T}$ with n_T samples. The source and target data are drawn from different distributions $P_s(X)$ and $P_T(X)$ respectively but the conditional distribution remains the same. That is to say, $P_S(X) \neq P_T(X)$ and $P_S(y|x) = P_T(y|x)$. The samples can be either uni-variate or multi-variate time series. Formally, $X_S^i,\ X_T^i \in \mathbb{R}^{M \times K}$ with M channels and K time steps. All the domains share the same label space, i.e., $Y_S^i \in \mathbb{R}^C$ where C is the number of classes. The goal is to design a predictive model that can correctly predict the label y_T of the unlabeled sample X_T from the target domain.

4.2 Overview of CADDN

Inspired by the intuition that the features related to the main goal are invariant across the domains, our motivation is to disentangle the domain-invariant and the domain-specific features before the domain feature alignment. Specifically, We use one common and two specific encoders to learn the common and specific features of the source and target

domains. To promote the independence of these features, we adapt mutual information measurement for the outputs of the two extractors in each domain. The learned domain-invariant features are leveraged to train the task classifier. Unlike existing works that use domain labels for domain-specific information extraction, we propose to use contrastive loss to induce uniformity over the source and target mixture distribution of domain-specific features and then cluster each domain on the unit hyper-sphere. Finally, the learned domain-invariant and domain-specific features are concatenated and fed into a common decoder for data reconstruction. The architecture of CADDN is shown in Fig.4.1

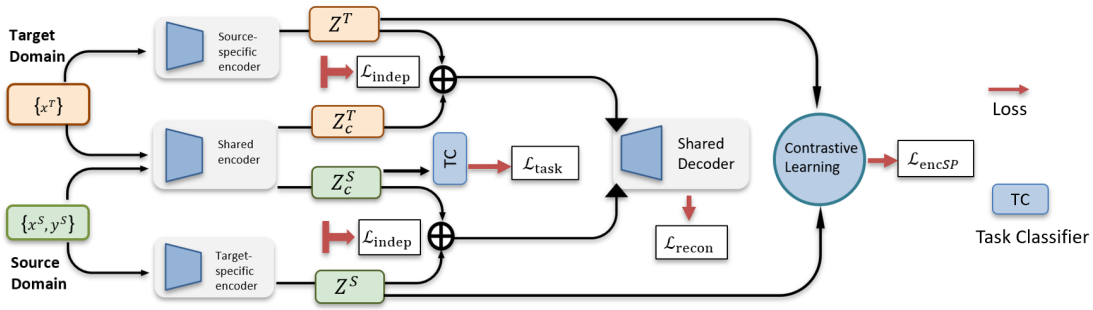


Fig. 4.1: The architecture of the proposed CADDN model.

4.3 Latent feature Disentanglement

Many factors determine time series data sampled from different domains, and some are helpful for the classification task while others are not. For example, the participants' personal style, environment constraints, and observation angles can influence human activity recognition data. Factors like these that have no contributions to the goals should not be included in our domain adaptation models. Thus it is meaningful for a model to identify these factors from data. To deal with this problem, we model the factors to be categorized into two groups: domain-invariant and domain-specific factors. The domain-invariant factors explain the latent features highly related to the main goals, such as the class of activity being conducted for the human activity recognition task and the health conditions in different systems of a big machine for machine fault detection. The domain-specific factors, on the contrary, lead to the latent features that change among domains and are useless for the main task.

We use a sliding window with length L to partition the training samples from both the source and target domains. Now the inputs of our model can be denoted as $X_S, X_T \in \mathbb{R}^{M \times L}$. Then we use three encoders $E_c(X)$, $E_{sp}^S(X)$ and $E_{sp}^T(X)$ to learn the domain-invariant features and the domain-specific features for both target domain and source domain. The obtained latent domain-invariant features and domain-specific features are then concatenated to feed into a shared decoder D to reconstruct input samples X_S and X_T.We use a mean squared error term for the reconstruction loss applied to both source and target domains.

$$\mathcal{L}_{recon} = \sum_{i=1}^{N_s} \mathcal{L}_{mse}(x_i^s, \hat{x}_i^s) + \sum_{j=1}^{N_t} \mathcal{L}_{mse}(x_j^t, \hat{x}_j^t) \tag{4.1}$$

where $\hat{x}_i^s$ and $\hat{x}_i^t$ are the reconstruction of the input x_i^s and x_i^t respectively. To guide the common encoder $E_c(X)$ to learn the crucial features for the classification task we are ultimately interested in, we use a task classifier TC to take the latent features learned as input and predict corresponding task labels. Noted that the target domain is unlabeled, the classifier is applied only to the labeled source domain. Thus the loss function is defined as

$$\mathcal{L}_{task} = -\sum_{i=1}^{N_s} y_i^s \cdot \log \hat{y}_i^s \tag{4.2}$$

where y_i^s are the one-hot encoding labels for input x_i^s and $\hat{y}_i^s$ are the softmax predictions of the classifier: $\hat{y}_i^s = TC(E_C(x_i^s))$

We assume that factors that affect the task labels are domain-invariant. As pointed out by Y.Ganin et al[11], a model's good performance on the target domain can be achieved by minimizing a trade-off between the source risk and a domain divergence of the source and target domain. They then proved that using a neural network-based discriminator can help control the domain divergence, thus can improve the domain adaptation performance. Following these conclusions, we use a discriminator D_{domain} to learn features from source and target domains that are as indistinguishable as possible. The difference between us is that Y.Ganin et al. aimed to find indistinguishable representations of examples, while we hope to disentangle information from the inputs by using the discriminator to select features that are domain-invariant. Specifically, the domain discriminator D_{domain} classifies whether a sample draws from the target or the source

domain. It is optimized according to a standard supervised loss with domains as the labels.

$$\mathcal{L}_{advD} = -\sum_{i=1}^{N_s} \log D_{domain}(E_c(X_i^s)) - \sum_{i=1}^{N_t} \log(1 - D_{domain}(E_c(X_i^t)))$$

Then we use the gradient reversal layer of [11] to optimize the common encoderE_C by maximizing the $\mathcal{L}_{advD}$ directly, i.e.:

$$\mathcal{L}_{encC} = -\mathcal{L}_{advD} \tag{4.3}$$

The network architecture of the proposed model is shown in Figure 1.

4.4 The Contrastive loss for learning Domain-specific Features

For encoders E_{sp}^T and E_{sp}^S, they are designed to learn domain-specific information. Specifically, for every minibatch of N samples from source domain and N samples from target domain, we can get N representations $\{z_i\}_{i=1}^N$ learned by E_{sp}^S and $\{z_i\}_{i=N+1}^{2N}$ learned by E_{sp}^T. Intuitively, the two domain-specific encoders should learn features that vary from different domains and are very similar when the samples are from the same domain. Inspired by contrastive representation learning algorithms[8] [41], the uniform distribution on the unit hypersphere can preserve maximal information, and sufficiently well-clustered features on the unit hypersphere can be linearly separable from the rest of the feature space. We try to induce uniformity over the mixture of the source and target domain-specific feature distribution and then cluster each domain on the unit hypersphere by using the form of contrastive loss. Thus instead of sampling negative samples explicitly, we treat the results from the same domain as positive samples and those from the other domain as negative samples. We use the cosine similarity as the similarity measure. $\text{sim}(u, v) = u^T v / \|u\| \|v\|$ Then we define $h(i, j) = 1_{[j \neq i]} \exp(\text{sim}(z_i, z_j)/\tau)$.

The loss function for samples i can be be formulated as follows: When sample i is from the source domain (i.e., $0 < i < N + 1$),

$$\ell_i = -\log \frac{\sum_{j=1}^{N} h(i,j)}{\sum_{k=1}^{2N} 1_{[k \neq i]} h(i,k)} \tag{4.4}$$

When sample i is from the target domain (i.e., $N < i < 2N + 1$),

$$\ell_i = -\log \frac{\sum_{j=N+1}^{2N} h(i,j)}{\sum_{k=1}^{2N} 1_{[k \neq i]} h(i,k)} \tag{4.5}$$

where $1_{[k \neq i]} \in \{0, 1\}$ is an indicator function evaluating to 1 iff $k \neq i$. τ is a temperature parameter. We assume that samples from the same domain are positive to each other. Given a batch of samples, the final loss is defined below.

$$\mathcal{L}_{encSP} = \sum_{1 \leq i \leq 2N} \ell_i \tag{4.6}$$

4.5 The Independence Loss

Although we use the above losses $\mathcal{L}_{encC}$, $\mathcal{L}_{task}$, and $\mathcal{L}_{encSP}$ to restrict the training of the three encoders, there can still exit overlapping information. The common encoder can contain the domain-specific features as long as they do not significantly alter the task classifier's decision boundary and vice versa. To better disentangle the domain-invariant and domain-specific information and further minimize the redundancy in models, we use the mutual information to measure the correlation between the domain-invariant and domain-specific features and then minimize it.

$$I(z_c^s; z^s) = D_{KL}\left[p(z_c^s, z^s) \parallel p(z_c^s)p(z^s)\right] \tag{4.7}$$

$$I(z_c^t; z^t) = D_{KL}\left[p(z_c^t, z^t) \parallel p(z_c^t)p(z^t)\right] \tag{4.8}$$

Where z_c^s, z^s, z_c^t and z^s denotes domain-invariant and domain-specific features from the source and target domains, respectively. And $p(.)$ means the probability of the item of sampling from the source domain or target domain.

Minimizing the mutual information defined above directly is difficult since the distributions are unknown. However, we can sample from joint distribution $p(z_c^t, z^t)$ or

$p(z_c^s, z^s)$ by sampling x from the target domain or source domain. Then we can sample from the product of marginal distributions $p(z_c^t)p(z^t)$ or $p(z_c^s)p(z^s)$ by randomly permuting across the batch axis, which is a standard trick used in the independence testing problems[2]. Having access to samples from both distributions allow us to use the density-ratio-trick[24][36][17] by involving a discriminator D_{indep_s} or D_{indep_t} in each domain approximating the density ratio. This discriminator outputs the probability that its input is sampled from the joint distribution rather than from the product of marginal distributions. Then we have

$$\mathcal{L}_{indep\_s} = \min_{E_C, E_{sp}^S} \max_{D_{indep\_s}} \Big[\mathbb{E}_{p(z_c^s)p(z^s)} \log D_{indep\_s}(z_c^s, z^s) + \mathbb{E}_{p(z_c^s, z^s)} \log(1 - D_{indep\_s}(z_c^s, z^s)) \Big]$$

$$\mathcal{L}_{indep\_t} = \min_{E_C, E_{sp}^T} \max_{D_{indep\_t}} \Big[\mathbb{E}_{p(z_c^t)p(z^t)} \log D_{indep\_t}(z_c^t, z^t) + \mathbb{E}_{p(z_c^t, z^t)} \log(1 - D_{indep\_t}(z_c^t, z^t)) \Big]$$

When there is only one domain considered, as proved by Goodfellow et al[**?**], the global minimum of the adversarial training equation is achieved if and only if $p(z_c^s, z^s) = p(z_c^s)p(z^s)$ in source domain or $p(z_c^t, z^t) = p(z_c^t)p(z^t)$ in target domain. Thus we can finally minimize the mutual information of the outputs of the two encoders for each domain.

4.5.1 Overall Objective Function

In summary, our proposed method is trained in a end-to-end manner. The overall objective can be formalized as follows:

$$\mathcal{L}_{overall} = \mathcal{L}_{task} + \lambda_1 \mathcal{L}_{encC} + \lambda_2 \mathcal{L}_{recon} + \lambda_3 \mathcal{L}_{encSP} + \lambda_4 \mathcal{L}_{indep}$$

where λ_1, λ_2, λ_3, λ_4 are the weights of each loss. We set $\lambda_1 = 0.1$, $\lambda_2 = 0.1$, $\lambda_3 = 0.1$, $\lambda_4 = 0.2$.

—

Chapter 5

Experiments

5.1 Datasets

We evaluate the proposed method on three commonly used human activity recognition datasets and one machine fault diagnosis dataset. The four benchmark datasets span various characteristics, including sample size, sensor type, and domain shift severity. Table 5.1 summarizes some basic information of each dataset, including the number of domains, classes, sensor channels, and sample length. The details of the datasets are as follows:

Table 5.1: Basic information of datasets.

	UCIHAR	WISDM	HHAR	Paderborn
#Domain	32	36	9	4
#Class	6	6	6	3
#Channel	9	3	3	1
#Length	128	128	128	5120

- *UCIHAR.* The UCIHAR dataset [1] contains six activities, walking, sitting, laying, standing, walking upstairs, and walking downstairs. It was carried out with 30 volunteers within an age bracket of 19-48 years. We treat each person as a single domain. We do the experiments based on Adatime[31] and use the same five cross-domain scenarios as in Adatime [31], i.e., from 12 to 16, from 2 to 11, from 6 to 23, from 7 to 13, and from 9 to 18.

- *WISDM.* The Wireless Sensor Data Mining (WISDM) dataset[19] is collected based on cell phone accelerometers from 36 subjects. Similar to the UICHAR dataset, it contains six activities. However, the WISDM dataset is more imbalanced in classes than UCIHAR and thus more challenging. We treat each person as a single domain. We use the same five cross-domain combinations as in Adatime[31], i.e., from 18 to 23, from 20 to 30, from 35 to 31, from 6 to 19, and from 7 to 18.

- *HHAR.* The Heterogeneity Human Activity Recognition(HHAR) [35] dataset is collected from 9 users(age range 25 to 35) performing six activities(biking, sitting, standing, walking, stair Up, stair down). Each participant conducted five minutes of each activity, which ensured a near equal data distribution among activity classes. We also consider each participant as a single domain. The five randomly chosen cross-domain combinations are from 0 to 6, from 1 to 6, from 2 to 7, from 3 to 8, and from 4 to 5.

- *Paderborn.* Machine fault classification can help to intelligently monitor machine health status and make maintenance. The Paderborn dataset [30]contains bearing machine sensor readings under four different operation conditions(A, B, C, D) with different operating parameters. Each has three different classes(healthy, inner-bearing damage, and outer-bearing damage). We consider each operation condition as a single domain and perform three cross-domain scenarios, i.e., from A to B, from B to C, and from C to D. All the data samples are segmented from the raw data with a 5120-length sliding window.

5.2 Baselines

In addition to state-of-the-art methods proposed for time series data, some methods for visual unsupervised domain adaption that can be applied to time series are also been included as baselines. In total, we use ten baselines namely, Deep Domain Confusion (DDC)[38], Correlation Alignment via Deep Neural Networks (Deep CORAL)[37], Higher-order Moment Matching (HoMM)[5], Minimum Discrepancy Estimation for Deep Domain Adaptation (MMDA)[32], Deep Subdomain Adaptation (DSAN)[45], Domain-Adversarial Training of Neural Networks (DANN)[11], Conditional Adversarial Domain

Adaptation (CDAN)[22], A DIRT-T Approach to Unsupervised Domain Adaptation[34], Convolutional deep Domain Adaptation model for Time Series data (CoDATS)[42], Adversarial Spectral Kernel Matching (AdvSKM)[21]. We do the experiments based on the code released by Adatime[31].

5.3 Implementation Details

We standardize all the training procedures for all the experiments. For instance, we use the same 1-dimensional convolutional neural network (1D-CNN) as the backbone network to promote fair comparison between different algorithms. We use Adam[18] optimizer for training optimization. We set the weight decay as 1e-4 and $(\beta_1, \beta_2) = (0.5, 0.99)$.

For the hyper-parameters setting, we first conduct a random hyper-parameter search with 30 hyper-parameter combinations on the WISDM dataset for all algorithms. Specifically, all the hyper-parameters are generated by a random sampling fram $1e-3, 5e-3, 8e-3, 1e-2, 0, 1, 0, 2, 0, 4, 0, 6, 0, 8, 1$. Then we choose the best hyper-parameters combination for all the algorithms and fix them on experiments of the other three datasets. Besides, we apply three fixed random seeds for each training scenario for each set of hyper-parameters and average their performance for stabilization. The same protocol were applied to the all algorithms to make the comparison fair and meaningful.

As for the evaluation metric, we employ the macro F1-score instead of the accuracy metric since the datasets are usually imbalanced. Then we use the mean macro F1-score of the several scenarios for each dataset as the final performance measure to compare between different algorithms. All the methods are implemented using pytorch[27].

5.4 Experiments

The experimental results of our proposed method and baselines on four benchmark datasets are listed in Table 5.2. Our proposed method has achieved the best performance on all four datasets. These results indicate that our model can learn powerful domain-invariant features across different domains for the time series classification task.

Table 5.2: The mean Macro F1-score for each dataset.

Method	HHAR	UCIHAR	WISDM	Paderborn
DDC	69.45	76.97	51.72	85.42
Deep-Coral	71.24	76.1	54.95	85.48
HoMM	74.03	82.02	57.78	86.52
MMDA	71.67	82.77	61.77	88.63
DSAN	76.21	83.74	55.97	91.16
DANN	74.14	85.65	60.75	85.31
CDAN	78.3	83.68	54.81	91.35
DIRT-T	79.93	83.2	62.33	92.33
CoDATS	74.34	83.76	62.15	90.87
ADVSKM	68.9	74.33	53.78	84.81
Ours	**81.45**	**87.14**	**65.54**	**92.38**

5.4.1 Ablation Study on Network Structure

We conduct an ablation study on the HHAR, UCIHAR, and WISDM datasets to show the contribution of each component in CADDN. Each dataset's models (A, B, C, D) are defined as shown in 5.3.

Table 5.3: Ablation study

Model	$\mathcal{L}_{recon} + \mathcal{L}_{task}$	$\mathcal{L}_{encC}$	$\mathcal{L}_{indep}$	$\mathcal{L}_{encSP}$
A	√			
B	√	√		
C	√	√	√	
D (full)	√	√	√	√

Figure 5.1 shows the results for each model in the three datasets. It can be seen that adding the independence excitation component can improve the performance by more than 4% in the WISDM dataset. It also indicates that removing the contrastive training for domain-specific features can be detrimental to the performance with about 5% degradation. In a nutshell, this ablation clearly shows the effectiveness of each component in our CADDN model.

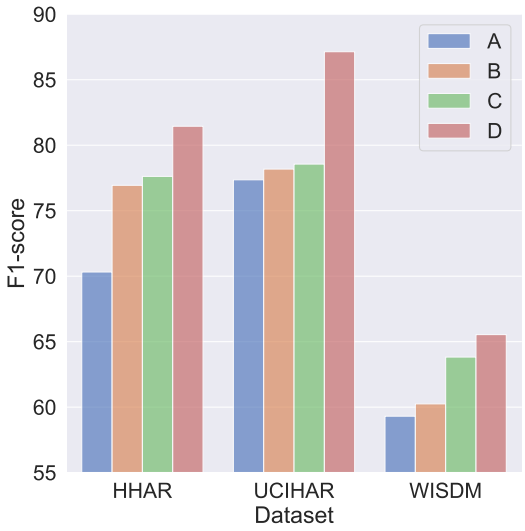


Fig. 5.1: Comparison between Ablation models(A, B, C, D) applied on the UCIHAR, HHAR, and WISDM datasets by measuring macro F1-score.

5.5 Sensitivity Analysis

We also investigate how the performance of our approach (CADDN) is affected by following hyper-parameters. (1) coefficient of the domain adversarial loss (λ_1), (2) coefficient of the reconstruction loss (λ_2), (3) coefficient of the contrastive loss (λ_3), (4) coefficient of the Independence loss (λ_4)

We performed a sensitivity analysis on the WISDM dataset, keeping the other three hyperparameters fixed, and sampled values of the parameter from 0.001 to 1.0 in the cross-domain scenarios. Fig 5.2 presents the average F1-score from these experiments with varying hyperparameters. Our findings indicate that gradually increasing the coefficient of the domain adversarial loss λ_1 can slightly improve performance, but over-weighting this loss coefficient can lead to a degradation in results as the common encoder may sacrifice some information for similarity. Additionally, we observed that setting $\lambda_2 = 0.1$ and $\lambda_3 = 0.1$ yielded better results. Finally, our approach achieved the highest F1-score when $\lambda_4 = 0.1$. In summary, our approach demonstrated superior performance on the WISDM dataset with hyperparameter settings of $\lambda_1 = 0.1$, $\lambda_2 = 0.1$, $\lambda_3 = 0.1$, and $\lambda_4 = 0.1$. We employed this hyperparameter configuration in all our experiments across the four datasets.

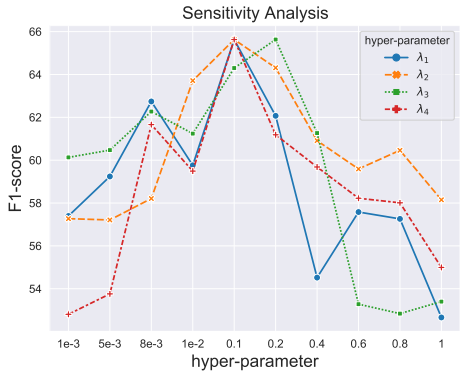


Fig. 5.2: Sensitivity analysis in WISDM dataset

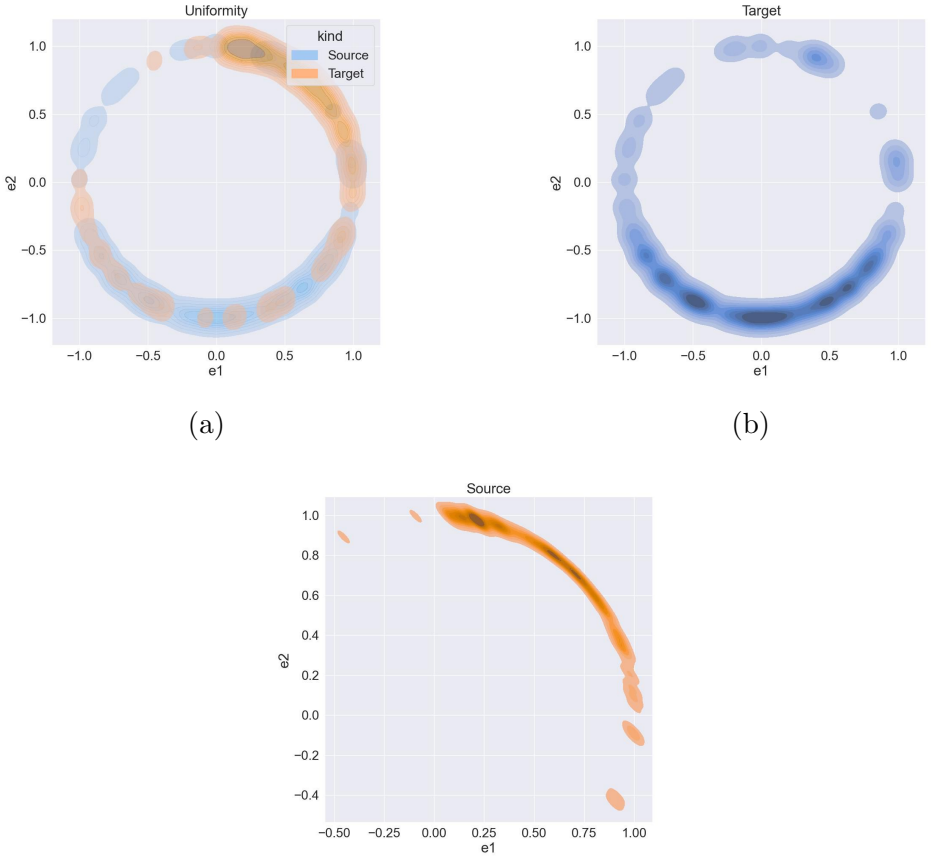
5.6 Domain-specific Feature Space Visualization

To show the uniformity of the domain-specific feature, we plot feature distributions with Gaussian kernel density estimation (KDE) in $\mathbb{R}^2$ (Fig 5.3). The two rightmost plots visualize domain-specific feature distribution from the source and target domains. We can observe in Fig 5.3that the mixture of the source and target domain-specific features is uniform. Besides, samples from the same domain are closely clustered, as shown in 5.3 (b) and 5.3 (c). That phenomenon validates our assumption of the domain-specific features. (c).

5.7 Domain-invariant Feature Space Visualization.

To show whether our proposed method is able to learn the domain-invariant features, we plot the t-SNE embeddings of the learned features by DDC and our CADDN model in Fig 5.4. These are the results drawn from the domain adaptation from domain 3 to domain 8 in the HHAR dataset. As shown in Fig 5.4(b) although from different domains, the features with the same label are mixed together compared to the result in Fig 5.4(a). That indicates the domain-invariant features learned by our CADDN method are less affected by the domain shift problem and thus promote the final performance.

In Fig 5.5, we plot the T-SNE embeddings of the learned domain-invariant latent features with different colors indicating different domains. Fig 5.5 (a) shows the clustering of the source domain(Blue) when we trained based on the source domain and directly

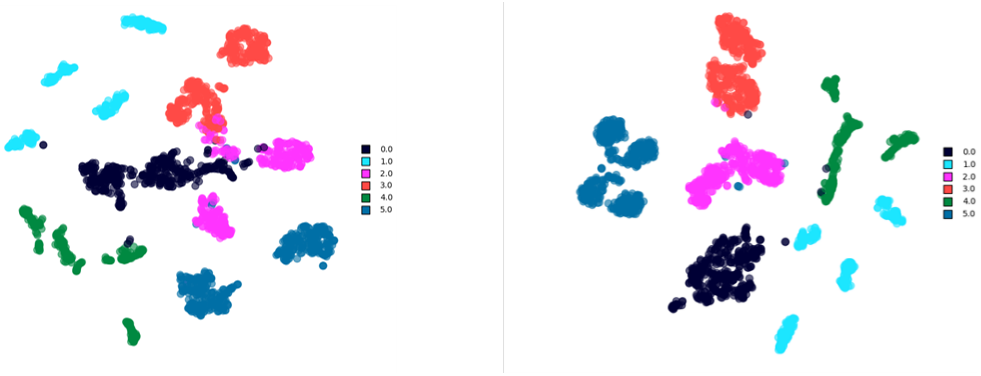


(a) (b)

(c)

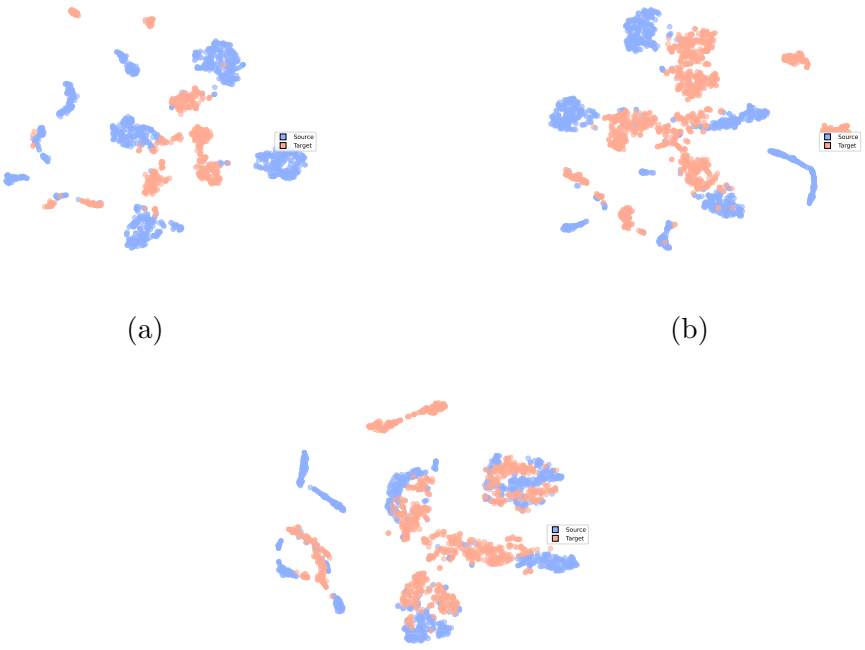
Fig. 5.3: The domain-specific feature distribution with Gaussian kernel density estimation. (a) the domain-specific feature distribution of the source and target mixed data.(b) the domain-specific feature distribution of the target domain.(c) the domain-specific feature distribution of the source domain.

applied the model to the target domain. Fig 5.5 (b) indicates a significant improvement of the DDC model, while Fig 5.5 (c) shows the further improvement of the clustering brought by our proposed CADDN model,explaining the best performance of CADDN.



(a) (b)

Fig. 5.4: Domain-invariant feature spaces Visualization. (a) t-SNE embedding of the learned domain-invariant features by DDC. (B) t-SNE embedding of the learned domain-invariant features by CADDN.



(a) (b)

(c)

Fig. 5.5: The t-SNE plot of the features from Source domain (blue) and the Target domain (red) (a) apply the model trained by the source domain directly to the target domain (b) use the DDC model for domain adaptation (c) the domain-invariant features trained from the CADDN model

Chapter 6

Conclusion

In this work, we propose a novel end-to-end approach named CADDN for domain adaptation problem in time series. Most existing methods are based either on mapping representations from one domain to the other or learning the features invariant to the domains by statistical restrictions. However, focusing only on the mapping or invariant features may leave the methods vulnerable to noise since they ignore much individual information. By disentangle the domain-specific and the domain-invariant information before the feature alignment, and innovatively combining the contrastive learning in the domain-specific feature learning process, our method outperforms the state-of-the-art models on four benchmark time series datasets.

References

[1] D. Anguita, Alessandro Ghio, L. Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN*, 2013.

[2] Miguel Arcones and Evarist Giné. On the bootstrap of u and v statistics. *The Annals of Statistics*, 20, 06 1992. doi: 10.1214/aos/1176348650.

[3] 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 *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6859–6867. AAAI Press, 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16846`.

[4] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulò. Autodial: Automatic domain alignment layers. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5077–5085. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.542. URL `https://doi.org/10.1109/ICCV.2017.542`.

[5] 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. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*

2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 3422–3429. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/5745.

[6] Qingchao Chen, Yang Liu, Zhaowen Wang, Ian Wassell, and Kevin Chetty. Re-weighted adversarial adaptation network for unsupervised domain adaptation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7976–7985, 2018. doi: 10.1109/CVPR.2018.00832.

[7] Shuo Chen, Fei Zhou, and Qingmin Liao. Visual domain adaptation using weighted subspace alignment. In *2016 Visual Communications and Image Processing (VCIP)*, pages 1–4, 2016. doi: 10.1109/VCIP.2016.7805516.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20j.html.

[9] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2980–2988, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/b618c3210e934362ac261db280128c22-Abstract.html.

[10] Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1180–1189. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ganin15.html.

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17:59:1–59:35, 2016. URL `http://jmlr.org/papers/v17/15-239.html`.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

[13] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1994–2003. PMLR, 2018. URL `http://proceedings.mlr.press/v80/hoffman18a.html`.

[14] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Deep transfer metric learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 325–333, 2015. doi: 10.1109/CVPR.2015.7298629.

[15] Hal Daumé III. Frustratingly easy domain adaptation. *CoRR*, abs/0907.1815, 2009. URL `http://arxiv.org/abs/0907.1815`.

[16] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. ACL, 2007.

[17] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2654–2663. PMLR, 2018. URL `http://proceedings.mlr.press/v80/kim18b.html`.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[19] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor.*, 12:74–82, 2011.

[20] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10277–10287, 2019. doi: 10.1109/CVPR.2019.01053.

[21] Qiao Liu and Hui Xue. Adversarial spectral kernel matching for unsupervised time series domain adaptation. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2744–2750. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/378. URL https://doi.org/10.24963/ijcai.2021/378. Main Track.

[22] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Domain adaptation with randomized multilinear adversarial networks. *CoRR*, abs/1705.10667, 2017. URL http://arxiv.org/abs/1705.10667.

[23] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2208–2217. PMLR, 2017. URL http://proceedings.mlr.press/v70/long17a.html.

[24] XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, nov 2010. doi: 10.1109/tit.2010.2068870. URL https://doi.org/10.1109%2Ftit.2010.2068870.

[25] S.J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[26] Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2239–2247. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00234. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Pan_Transferrable_Prototypical_Networks_for_Unsupervised_Domain_Adaptation_CVPR_2019_paper.html.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[28] Marius-Constantin Popescu, Valentina E. Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Trans. Cir. and Sys.*, 8(7):579–588, jul 2009. ISSN 1109-2734.

[29] Sanjay Purushotham, Wilka Carvalho, Tanachat Nilanon, and Yan Liu. Variational recurrent adversarial deep domain adaptation. 04 2019.

[30] Mohamed Ragab. Paderborn Dataset Processed for Domain Adaptation Scenarios, 2022. URL https://doi.org/10.21979/N9/X6M827.

[31] Mohamed Ragab, Emadeldeen Eldele, Wee Ling Tan, Chuan-Sheng Foo, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. Adatime: A benchmarking suite for domain adaptation on time series data, 2022.

[32] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. On minimum discrepancy estimation for deep domain adaptation. In Richa Singh, Mayank Vatsa, Vishal M. Patel, and Nalini K. Ratha, editors, *Domain Adaptation for Visual Understanding*, pages 81–94. Springer, 2020. doi: 10.1007/978-3-030-30671-7_6. URL https://doi.org/10.1007/978-3-030-30671-7_6.

[33] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Residual parameter transfer for deep domain adaptation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4339–4348, 2018. doi: 10.1109/CVPR.2018.00456.

[34] Rui Shu, Hung H. Bui, Hirokazu Narui, and Stefano Ermon. A DIRT-T approach to unsupervised domain adaptation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=H1q-TM-AW.

[35] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor S. Prentow, Mikkel Baun Kjærgaard, Anind K. Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015.

[36] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the Bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, October 2012. doi: 10.1007/s10463-011-0343-8. URL https://ideas.repec.org/a/spr/aistmt/v64y2012i5p1009-1044.html.

[37] Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. *CoRR*, abs/1612.01939, 2016. URL http://arxiv.org/abs/1612.01939.

[38] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014. URL `http://arxiv.org/abs/1412.3474`.

[39] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4068–4076, 2015. doi: 10.1109/ICCV.2015.463.

[40] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390294. URL `https://doi.org/10.1145/1390156.1390294`.

[41] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR, 2020. URL `http://proceedings.mlr.press/v119/wang20k.html`.

[42] Garrett Wilson, Janardhan Rao Doppa, and Diane J. Cook. Multi-source deep domain adaptation with weak supervision for time-series sensor data. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1768–1778. ACM, 2020. doi: 10.1145/3394486.3403228. URL `https://doi.org/10.1145/3394486.3403228`.

[43] Lei Zhang. Transfer adaptation learning: A decade survey. *CoRR*, abs/1903.04687, 2019. URL `http://arxiv.org/abs/1903.04687`.

[44] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J. Gordon. On learning invariant representations for domain adaptation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference*

on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of *Proceedings of Machine Learning Research*, pages 7523–7532. PMLR, 2019. URL `http://proceedings.mlr.press/v97/zhao19a.html`.

[45] 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. doi: 10.1109/TNNLS.2020.2988928.