Towards Robust and Label-efficient Time Series Representation Learning

Emadeldeen Ahmed Ibrahim Ahmed Eldele
School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

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.

7 April. 2023

Date

Emadeldeen Ahmed Ibrahim Ahmed Eldele

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 accordance with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

7 April 2023

.....

Date

.....

Prof. Chee-Keong Kwoh

Authorship Attribution Statement

This thesis contains material from seven papers, six of which are published in peer-reviewed journals and conferences, and one of them is still under review. In these seven papers, I am listed as the first author. The details are as follows.

Parts of Chapter 2 are from **Emadeldeen Eldele**, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, and Xiaoli Li. Label-efficient Time Series Representation Learning: A Review. ArXiv preprint arXiv:2302.06433, 2023. . This work is still Under Review in the *IEEE Transactions on Artificial Intelligence*.

The contributions of the authors are as follows:

- I surveyed and summarized the previous works, proposed the organization hierarchy, and drafted the manuscript.
- Dr. Mohamed Ragab, Dr. Zhenghua Chen, and Dr. Min Wu gave suggestions and discussed the organization of the paper, and helped in revising the writing.
- Prof. Chee-Keong Kwoh and Prof. Xiaoli Li helped in reviewing the manuscript.

Chapter 3 is published as **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*, 29:809–818, 2021.

The contributions of the authors are as follows:

- I developed the algorithm, designed and executed experiments, and created the source code. I also drafted the manuscript.
- Dr. Min Wu helped in modifying and organizing the manuscript, gave suggestions for experimental design and revised the writing.
- Dr. Zhenghua Chen assisted in the methodology, designing experiments, and revising the manuscript.
- Prof. Chee-Keong Kwoh pointed out the research direction and guided the technical discussion of the proposed approach.
- Prof. Chengyu Liu, Prof. Xiaoli Li, and Prof. Cuntai Guan helped in reviewing the manuscript.

Chapter 4 is published as **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.

The contributions of the authors are as follows:

- I developed the key idea and the algorithm, designed and executed experiments, created the source code and drafted the manuscript.
- Dr. Mohamed Ragab assisted in polishing the idea, designing the experiments, and revising the manuscript.
- Dr. Min Wu assisted in the experimental design and writing and revising of the manuscript.
- Dr. Zhenghua Chen assisted in the experimental design and in revising the manuscript.
- Prof. Chee-Keong Kwoh pointed out the research direction and guided the technical discussion of the proposed approach.
- Prof. Xiaoli Li and Prof. Cuntai Guan helped in reviewing the manuscript.

Chapter 5 is published as **Emadeldeen Eldele**, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Self-supervised contrastive representation learning for semi-supervised time-series classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

The contributions of the authors are as follows:

- I developed the key idea and the algorithm, designed and executed experiments, created the source code and drafted the manuscript.
- Dr. Mohamed Ragab assisted in polishing the idea, designing the experiments, and revising the manuscript.
- Dr. Min Wu assisted in the experimental design and writing and revising of the manuscript.
- Dr. Zhenghua Chen assisted in the experimental design and in revising the manuscript.
- Prof. Chee-Keong Kwoh pointed out the research direction and guided the technical discussion of the proposed approach.
- Prof. Xiaoli Li and Prof. Cuntai Guan helped in reviewing the manuscript.

Chapter 6 is published as **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.

The contributions of the authors are as follows:

- I developed the key idea and the algorithm, designed and executed experiments, created the source code and drafted the manuscript.
- Dr. Mohamed Ragab, Dr. Min Wu, and Dr. Zhenghua Chen assisted in the experimental design and in revising the manuscript.

- Prof. Chee-Keong Kwoh pointed out the research direction and guided the technical discussion of the proposed approach.
- Prof. Xiaoli Li and Prof. Cuntai Guan helped in reviewing the manuscript.

Chapter 7 is published as **Emadeldeen Eldele**, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, and Xiaoli Li. CoTMix: Contrastive Domain Adaptation for Time-Series via Temporal Mixup. *IEEE Transactions on Artificial Intelligence*, 2023.

The contributions of the authors are as follows:

- I developed the key idea and the algorithm, designed and executed experiments, and created the source code. I also drafted the manuscript.
- Dr. Mohamed Ragab, Dr. Min Wu, and Dr. Zhenghua Chen assisted in the experimental design and in revising the manuscript.
- Prof. Chee-Keong Kwoh pointed out the research direction and guided the technical discussion of the proposed approach.
- Prof. Xiaoli Li helped in reviewing the manuscript.

Parts of discussions and experiments in Chapter 3 and Chapter 4 are published as **Emadeldeen Eldele**, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. Self-supervised Learning for Label-Efficient Sleep Stage Classification: A Comprehensive Evaluation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 31:1333-1342, 2023.

The contributions of the authors are as follows:

- I developed the key idea and the algorithm, designed and executed experiments, and created the source code. I also drafted the manuscript.
- Dr. Mohamed Ragab, Dr. Min Wu, and Dr. Zhenghua Chen assisted in the experimental design and in revising the manuscript.
- Prof. Chee-Keong Kwoh pointed out the research direction and guided the technical discussion of the proposed approach.
- Prof. Xiaoli Li helped in reviewing the manuscript.

7 April 2023

Date

Emadeldeen Ahmed Ibrahim Ahmed Eldele

Acknowledgements

I would like to express my sincere gratitude to my supervisors Prof. Chee-Keong Kwoh and Dr. Min Wu for their guidance, support, and encouragement throughout my thesis research. Their insights, expertise, support, and constructive criticism have been invaluable to me, and I feel fortunate to have had such dedicated and knowledgeable mentors. I would like also to extend my appreciation to Dr. Zhenghua Chen for providing me with valuable feedback, insight, and suggestions during my research. Also, I would like to acknowledge my colleague Dr. Mohamed Ragab for his collaboration and assistance in various aspects of my research. His discussions and contributions have been important during my Ph.D. journey, and I appreciate his professionalism and teamwork.

I am also grateful to my co-authors and my TAC committee members, Prof. Xiaoli Li and Prof. Cuntai Guan for their collaboration and intellectual input. Their expertise and insights have contributed significantly to the outputs of my work.

I would like to thank my parents Eng. Ahmed Eldele and Mrs. Kareema Dwedar for their unwavering support, encouragement, and love throughout my academic journey. Their sacrifice, guidance, and wisdom have been my source of strength and inspiration, and I owe them my deepest gratitude.

Last but not least, I want to express my heartfelt appreciation to my wife Mrs. Asmaa Alwakeel for her unwavering support, patience, and understanding during the ups and downs of my thesis research. Her love, encouragement, and motivation have been my anchor and my driving force, and I could not have completed this thesis without her support.

Abstract

Time series data is sequential measurements collected over time from various sources in different applications, e.g., healthcare and manufacturing. With the increased generation of time series data from these applications, their analysis is becoming more important to get insights. Deep learning has shown a potential and proven capability in automatic learning from massive data, by identifying complex patterns and representations directly from data. However, the current deep learning-based models suffer significant limitations. First, they lack the ability to efficiently learn time series temporal relations while utilizing parallel processing. Second, these models require large amounts of labeled data for training, which can be difficult to obtain, especially with complex time series data. Third, the generalization capability of these models is limited, where they suffer performance deterioration when transferring knowledge from a labeled source domain to an out-of-distribution unlabeled target domain.

In this thesis, we address these problems and provide solutions for the real-world deployment of deep learning models for time series data. We first propose a novel attention-based deep learning architecture called AttnSleep to classify EEG-based sleep stages, being one of the common time series healthcare data types. Specifically, we propose a powerful feature extractor that learns from different frequency bands in EEG signals. We also propose a temporal context encoder module to learn the temporal dependencies among extracted features using a causal multi-head attention mechanism. Last, we develop a class-aware loss function to address the class-imbalance problem in sleep data without incurring any additional computational costs.

Next, we propose two frameworks to address the label scarcity problem in different settings. The first framework, TS-TCC, is a self-supervised learning approach that learns useful representations from unlabeled data. TS-TCC utilizes time series-specific augmentations to generate two views for each sample. We then learn the

temporal representations via a novel cross-view temporal prediction task. Furthermore, we propose a contextual contrasting module that further learns discriminative representations. The second framework, CA-TCC, improves the learned representations from TS-TCC in semi-supervised settings, by training the model in four phases. First, we perform self-supervised training with TS-TCC. Then, we fine-tune the pretrained model with the available few labeled samples. Following that, we use the fine-tuned model to assign pseudo labels to the unlabeled set. Finally, we leverage these pseudo labels to realize a class-aware contrastive loss for semi-supervised training. These two frameworks showed significant performance improvement with having few labeled samples compared to traditional supervised training.

Last, we tackle the domain shift problem and propose two novel frameworks to address this issue. In the first framework, we introduce an adversarial domain adaptation technique named ADAST, that addresses two challenges, namely the loss of domain-specific information during feature extraction and the ignorance of class information in the target domain during domain alignment. To overcome these challenges, we incorporate an unshared attention mechanism and an iterative self-training strategy with dual distinct classifiers. In the second framework, we attempt to overcome the complexity of adversarial training and present a novel approach called CoTMix to address the domain shift with a simple yet effective contrastive learning strategy. In specific, we propose a cross-domain temporal mixup strategy to create source-dominant and target-dominant domains. These domains serve as augmented views for the source and target domains in contrastive learning. Unlike prior works, CoTMix maps the source and target domains to an intermediate domain. These frameworks showed improved robustness of deep learning models on time series data.

Contents

List of Figures

List of Tables

Symbols and Acronyms

ATT	Attention model
B	Batch size
C	Classifier model
D	Domain discriminator
d	Sample length in the embedding space
F	Feature extractor model
f	feature value at a specific position in **z**
H	An attention head in multi-head attention
h	Hidden dimension in an autoregressive model
K	Number of classes
N	Number of channels in the embedding space
n	number of samples of a domain
P	dataset distribution
p	Predictions of a classifier
T	total timesteps of a time series sample in the input space
$\mathcal{W}$	A Fully Connected layer
X	dataset domain
x	A sample from X
x	sample set from X
Y	Label space
y	Label of a sample from X
$\hat{y}$	Pseudo label
z	Embedding space of **x**
ϕ	Causal convolution
σ	Sigmoid activation function
δ	ReLU activation function

λ	Mixup ratio
$\mathcal{L}$	Loss function

Chapter 1

Introduction

The advancement of instruments and sensors in various healthcare and industrial fields has led to the generation of vast amounts of time series data. Meanwhile, deep learning approaches have shown promising results in analyzing such data. However, there is a shortage of research on effectively implementing deep learning models in real-world scenarios. In fact, the learning capability of deep learning-based models on time series data can be limited due to issues such as inefficient learning of temporal dynamics, the class-imbalance issue, and the lack of large labeled datasets. Therefore, in this thesis, we seek to identify and evaluate various strategies to overcome these challenges and develop robust and label-efficient time series representation learning frameworks. This chapter aims to introduce the research by discussing the background, motivations, research problems, aims, objectives, and questions.

1.1 Background

Time Series Data Time series data is a collection of sequential time-stamped measurements from natural phenomena (such as weather), engineered systems (such as aircraft engines and power grids), and even human behavior (such as speech patterns and brain signals) [2]. Over the past years, the analysis of time series data has gained more interest due to the large amounts of data generated daily in various applications. In healthcare, for instance, the vital signs of patients, including heart rate, blood pressure, and respiratory rate, are tracked through time

[3]. Healthcare professionals can analyze the collected time-stamped data to track the progression of a patient's condition and detect early signs of disease or health problems, such as heart attacks and sleep disorders. In manufacturing, time series data is used to monitor machine behavior and predict when maintenance is needed [4]. By analyzing data patterns, manufacturers can detect unusual behavior and diagnose problems with the machines. Human activity recognition is yet another application in which we can collect time series data, where the data generated by the wearable sensors, such as accelerometers, gyroscopes, and magnetometers, over time can be utilized to recognize various human activities, such as walking, running, and sitting [5]. Consequently, we are capable of developing personal health and fitness trackers and monitoring elderly or disabled individuals to ensure their safety.

One of the recent and rapidly growing applications of time series analysis is the **sleep stage classification**, which has been extensively studied in the field of sleep medicine [6]. Sleep is a complex physiological process that consists of four distinct stages: three non-rapid eye movement stages (N1, N2, and N3) and a rapid eye movement (REM) stage [7]. Each stage is characterized by unique patterns of brain waves, muscle tone, and eye movements. The lightest stage of sleep is N1, which is characterized by the presence of alpha and theta waves. Slightly deeper than N1 is stage N2, which is characterized by the presence of sleep spindles and K-complexes. N3 is the deepest stage of sleep, marked by the presence of delta waves. Finally, REM sleep is a stage of high brain activity characterized by rapid eye movements and vivid dreams [8]. Accurate classification of sleep stages is essential for the diagnosis and treatment of sleep disorders, as it provides valuable insights into the quality and quantity of an individual's sleep [9].

Classifying sleep stages includes identifying the specific stage of sleep a person is in at any given time. Traditionally, polysomnography (PSG) has been used to record multiple physiological signals during sleep, such as electroencephalogram (EEG), electrooculogram (EOG), and electromyogram (EMG), to determine the sleep stage [10]. However, in recent years, single-channel EEG has become a popular option for sleep monitoring due to its ease of use [11]. Therefore, we will focus on exploring the potential of single-channel EEG for sleep stage classification throughout this thesis.

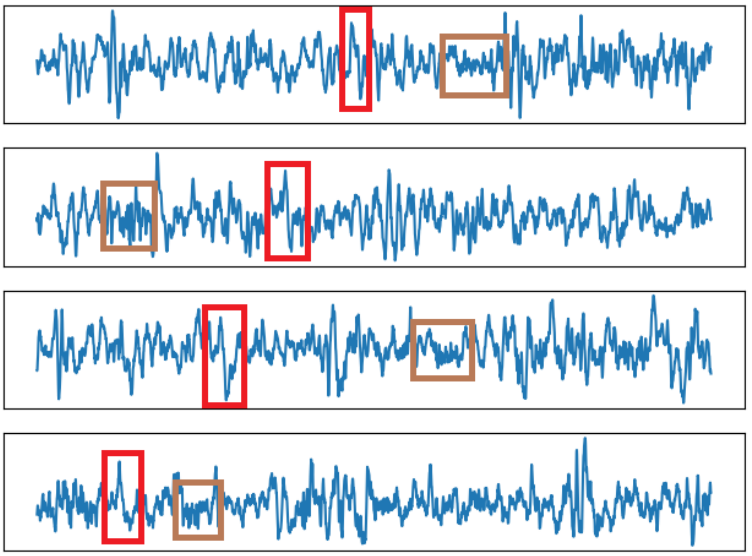


FIGURE 1.1: K-complex (red) and sleep spindle (brown) temporal patterns in the non-rapid eye movement stage 2 in sleep EEG data. The temporal order of these two waveforms matters. In the first, third, and fourth subfigures, the K-complex appears before the sleep spindles indicating normal behavior. However, in the second subfigure, their reversed order indicates a sleep disorder.

Temporal Relations in Time Series Understanding the characteristics of time series data is crucial for developing the right solutions and performing the proper analysis. One of the defining characteristics of time series data is its temporal ordering. In time series data, observations are recorded at regular or irregular intervals, with each observation being influenced by the previous observations [12]. This makes the analysis of a specific timestep or a window of timesteps inherently dependent on the previous ones. Studying the temporal relations of timesteps can be significant in many applications. For instance, in EEG-based sleep stage classification, the temporal order of waveforms can matter. During the non-rapid eye movement stage 2 (N2), there are two waveforms, the K-complex and sleep spindles, whose temporal order can be significant [13]. It has been observed that sleep spindles tend to appear just after a K-complex, with a delay of a few seconds in normal cases, as shown in Fig. 1.1. This delay reflects the time it takes for the brain to transition from the inhibitory effect of the K-complex to the excitatory state of the sleep spindle. However, studying this temporal relation between the two waveforms has revealed that their order may also be altered in sleep disorders, as shown in subfigure 2 of Fig.1.1. Additionally, the appearance of K-complexes and sleep spindles may be disrupted or delayed, leading to sleep disturbances and daytime sleepiness [14].

Deep Learning for Time Series The manual analysis and processing of time series signals can be a challenging and inefficient task for several reasons. Firstly, the manual process can be subjective and prone to errors since it depends on the expertise of the individual performing the analysis, leading to varying interpretations of the same data [15]. Secondly, the volume of generated data is massive, making manual analysis an overwhelming and time-consuming task. Thirdly, repetitiveness in analyzing and processing time series data can cause human errors, leading to inaccuracies in the results. Additionally, time series data can be complex with multiple variables and dependencies, making it more challenging to analyze and process manually [16]. Lastly, some applications require real-time processing, such as in healthcare monitoring systems, which is difficult to achieve manually [17]. Therefore, there is a crucial need for automated solutions to assist specialists and save time in analyzing and processing time series data.

Deep learning is a subfield of machine learning that has gained widespread popularity in recent years due to its ability to learn complex representations from time series data [18]. In contrast to traditional time series modeling methods, which rely on handcrafted features and statistical techniques, deep learning methods can learn complex patterns directly from data. Deep learning models are designed to learn from large datasets, where the availability of massive amounts of labeled time series data has made it possible to train models that can provide accurate predictions for a wide range of applications.

There are several deep learning architectures that have demonstrated significant improvements in time series analysis. For instance, convolutional neural networks (CNNs) have proven to be particularly effective for solving time series classification problems [19]. CNNs are particularly adept at learning the spatial structure in time series data, leading to impressive results in a variety of applications. Another architecture that has gained attention in time series analysis is the recurrent neural network (RNN). RNNs are specifically designed to handle sequential data and capture temporal dependencies [20]. By incorporating memory in their computations, RNNs can remember past observations and use them to make predictions about future observations. RNNs have shown great promise in speech recognition, natural language processing, and financial forecasting. Another promising architecture is the attention model, which has shown the ability to learn temporal relationships

with linear operations and parallel processing [21]. This makes it well-suited for processing long sequences of time series data.

1.2 Research Problem

In recent years, there has been a surge in the number of deep learning-based approaches proposed for learning representations from time series data. Recurrent neural networks, such as Long Short-Term Memory (LSTM), have been the go-to architecture to learn the complex temporal structure of time series data. However, while RNNs have shown promise in time series analysis, they have some drawbacks that limit their applicability in real-world scenarios. One of these drawbacks is their recurrent nature, which can result in high model complexity and make them difficult to train in parallel. Additionally, this complexity can lead to slower processing times, which can be problematic for real-time applications. Therefore, there is a need for an alternative to RNNs that can learn time series temporal relations while utilizing parallel processing.

Furthermore, deep learning models often rely on large amounts of labeled data for training, which may not always be feasible due to the difficulties in annotating time series data. Unlike image data, expert knowledge is often required to identify relevant patterns in time series data [22]. As a result, the collected data is usually larger than the annotated data, which can limit the learning capability of deep learning models and increase the challenge of designing proper models.

There are two ways to overcome the issue of label scarcity. The first is to learn representations from the available unlabeled data and fine-tune the model using the limited labeled data. The second is to transfer knowledge from a labeled source domain to the target domain of interest. Therefore, we require an approach that allows us to effectively learn from unlabeled samples, and another approach to successfully transfer the knowledge from an out-of-distribution source domain to our target domain.

In conclusion, existing deep learning research for time series is ill-equipped to address the following challenges:

- Developing effective and computationally efficient methods to learn **temporal** relations in complex time series data.
- Finding efficient ways to learn useful representations from the **unlabeled** time series data that is typically collected in large volumes.
- Dealing with **domain shift** problems that arise when attempting to transfer knowledge from a labeled source domain to our unlabeled target domain of interest.

1.3 Research Objectives and Questions

Given the lack of research regarding efficient deep learning modeling in real world, this study will aim to develop and evaluate deep learning models that can efficiently learn robust representations from time series in different labeling budgets, while capturing the temporal relations. This enhances the utilization of deep learning models in real-world time series-based applications. We adopt sleep stage classification as a use-case study, but we also examine other time series applications when possible. The objectives of the thesis can be summarized as follows:

1. **To develop deep learning models that can efficiently utilize the temporal relations in time series data.** Learning temporal relations in time series data is a key part of an efficient deep learning model. Therefore, we aim to develop deep learning models that can extract powerful features from time series data and efficiently capture the temporal dependencies in the extracted features.
2. **To learn representations from the massive unlabeled data.** In the real world, the pace of data annotation is much less than the data collection. Therefore, we aim to develop deep learning models that can learn representations from unlabeled data, and improve these representations with the available few annotated data.
3. **To develop a generalizable deep learning model over different domains.** One way to overcome the limited data annotation is to learn from another labeled source domain and transfer the knowledge to our domain.

However, the difference in the distributions among both domains can deteriorate the performance. Therefore, we aim to develop a deep learning model that can generalize well across domains.

In specific, we aim to answer the following research questions through this thesis.

RQ1: How can we develop a deep learning model that efficiently utilizes temporal dependencies in time series data to learn powerful features, setting the foundation for advanced representational tasks?

RQ2: Given a model that can generate powerful features, how can we further refine and learn time series-specific representations from fully unlabeled data, enhancing these representations with the availability of a few labels?

RQ3: Leveraging the learned features, how can we design a deep learning model that adapts to generalizable features from a shifted domain, effectively mitigating the distribution shift between the source and target domains?

Throughout this thesis, we delve into time series classification due to its significant applicability and the emerging challenges it presents, especially in the face of rapidly evolving data dynamics in numerous domains. While the broader field of time series analysis encompasses other tasks, e.g., forecasting, imputation, and anomaly detection, the in-depth exploration of classification would provide valuable insights and contribute meaningfully to the academic and practical discourse on this specific topic.

1.4 Thesis Organization

The organization of this thesis is illustrated in Figure 1.2. The following chapters are organized as follows:

1. Chapter 2 provides a comprehensive review of the existing literature to identify key works that addressed representation learning for time series data, in addition to those that addressed label scarcity and domain shift problems.

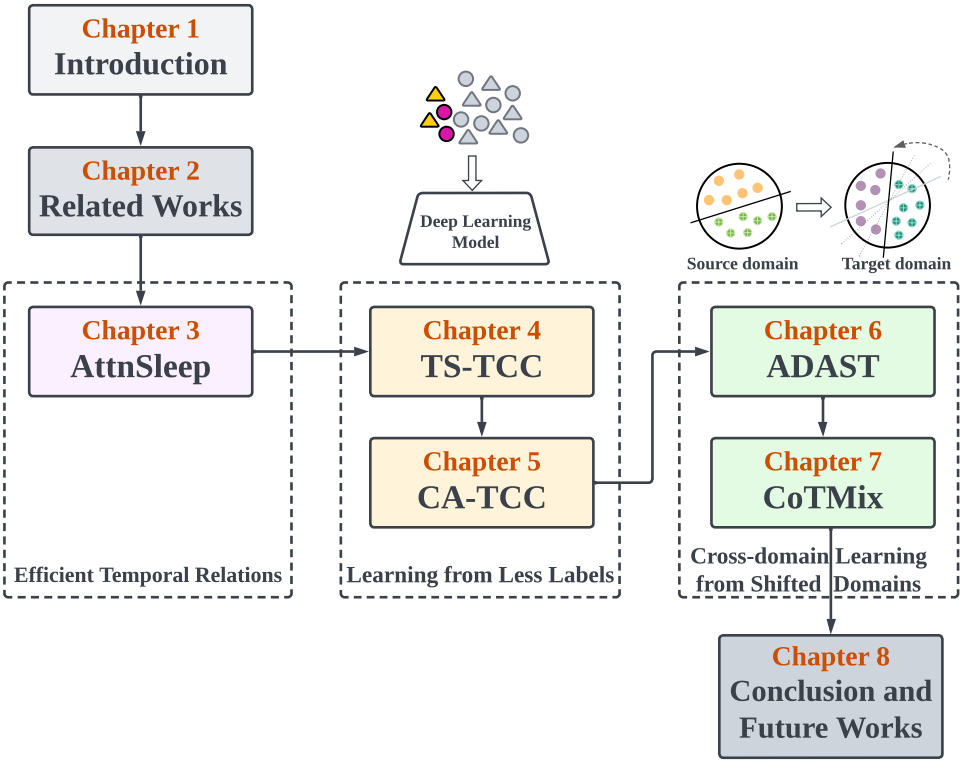


FIGURE 1.2: Organization of chapters in this thesis.

2. Chapter 3 introduces the **AttnSleep** model, which aims to learn robust representations from EEG sleep data by efficiently learning temporal dependencies and addressing the data imbalance issue with a computationally efficient loss function.

3. Chapter 4 discusses our approach to learning from the unlabeled data, namely **TS-TCC**.

4. In Chapter 5, we discuss our approach to improve the learned representations with the availability of a few labeled samples. In specific, we introduce our **CA-TCC** model, which extends TS-TCC to the semi-supervised setting.

5. Chapter 6 presents our **ADAST** framework, which aims to mitigate the domain shift problem across datasets, i.e., when transferring knowledge from a shifted source dataset to our dataset of interest.

6. Chapter 7 introduces our **CoTMix** framework, which aims to mitigate cross-subject shift within the same dataset.

7. Finally, Chapter 8 concludes the thesis and outlines some future research directions.

Chapter 2

Literature Review

In this chapter [1], we present some of the existing literature and related works to our research aims. In specific, we will begin by examining the deep learning architectures for time series representation learning, wherein we will delve into how they extract features and learn the temporal dynamics in signals. Subsequently, we will discuss how previous works addressed the label scarcity problem with a focus on self-supervised and semi-supervised learning techniques that are designed to exploit intra-domain data. After that, we will investigate cross-domain learning from distributionally-shifted domains, and explore the development of a generalizable deep learning model. Finally, we will present a case study on sleep stage classification, as an example of a time series application, and analyze the relevant literature in addressing the aforementioned issues. The overall organization of this chapter is illustrated in Fig. 2.1.

2.1 Time Series Representation Learning Architectures

The first step towards building an efficient deep learning model is the proper choice of model components. Since our focus is on time series data, we split the model into two main parts: the first is the feature extractor, and the other is the temporal

[1] The work in this chapter is available on Arxiv as "Label-efficient Time Series Representation Learning: A Review", ArXiv preprint arXiv:2302.06433 [23], and is currently under review.

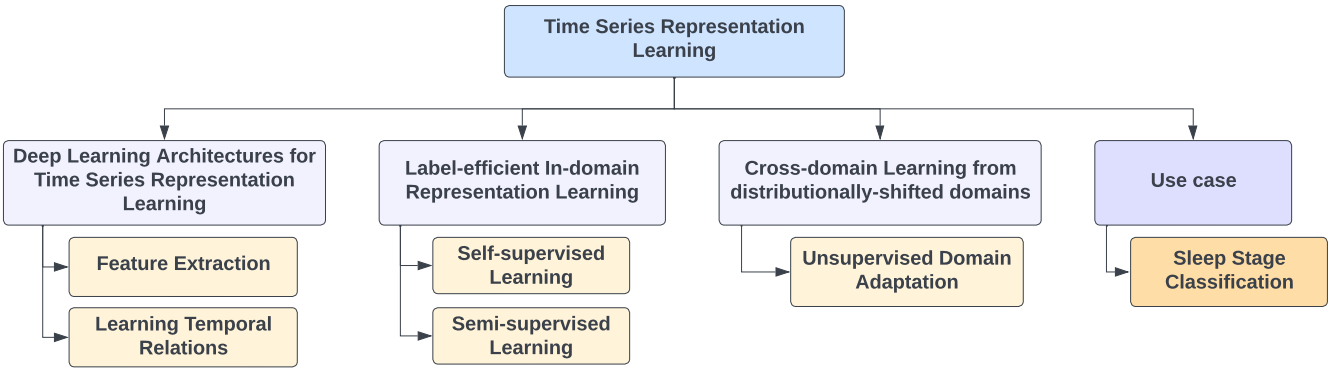


FIGURE 2.1: The organization of our Literature Review chapter

dynamic learning module. This section will discuss the different methods to extract features and learn temporal dynamics in more detail.

2.1.1 Feature Extraction

Feature extraction is a crucial step in time series analysis and prediction, as it allows us to extract useful information from raw time series data. Unlike traditional machine learning approaches, deep learning models can automatically learn features from the data without the need for manual feature engineering. In this context, feature extraction modules in deep learning refer to the architectures and techniques used to extract features from time series data.

One common feature extraction module for time series data is the convolutional neural network (CNN). In a CNN, filters are applied to the time series signals to learn local patterns. By applying these filters, we end up with feature maps, which are then pooled to reduce dimensionality with Maxpooling or AveragePooling layers. This approach is particularly effective when the time series data has a clear structure and patterns, which makes it dependable for classification tasks. Examples of the widely used CNN architectures are the simple 1D-CNN, which consists of stacked convolutional layers with Batch-normalization and MaxPooling layers [24]. In addition, the 1D-ResNet architecture is one of the most popular architectures due to its shortcut residual connection that helps to improve the flow of gradients through the network during backpropagation, and hence, enables the network to converge faster and more accurately [25]. Additionally, the temporal convolutional network (TCN) is designed to process sequential time series data.

TCNs rely on causal dilated convolutions to learn temporal relations and capture the long-term dependencies in the data [26].

Nevertheless, other convolutional designs have been proposed and adopted in different time series-related works. For example, Ismail Fawaz et al. [27] propose InceptionTime, which ensembles five randomly initialized Inception modules that share the same architecture. In addition, Liu et al. [28] proposed a tensor scheme that uses a sliding window to consider the temporal order to transform the time series signal into a 3-dimensional tensor and use traditional CNN architectures. Also, Tang et al. [29] design Omni-Scale CNN blocks that automatically decide the best size and number to the kernel size used by CNNs. To do so, they automatically set the kernel sizes as prime numbers up to the length of the time series, which showed a balanced performance over different datasets.

Works that adopted pure CNN-based architectures show superiority in time series classification tasks, due to the high ability of CNNs to detect patterns and trends in data. In addition, CNNs are less prone to overfitting as they use a reduced number of parameters. However, they are less efficient in detecting the temporal relations in data, which makes them less favorable in time series forecasting tasks.

2.1.2 Learning Temporal Dynamics

Learning temporal relations is a crucial aspect of deep learning for time series, which becomes vital for capturing patterns and trends in the signals accurately. Next, we will discuss two of the commonly used deep learning architectures for this purpose.

2.1.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are one of the commonly used modules for time series data. RNNs are specially designed to handle sequences of timesteps and capture their temporal dependencies. RNNs include a memory component that allows them to take into account the past timesteps when processing the current one. LSTMs are designed to address the vanishing gradient problem, which is a common issue in RNNs that occurs when the gradient signal in the backpropagation algorithm diminishes as it passes through many timesteps [30]. This can result in

the network being unable to learn long-term dependencies in the data. The key innovation of LSTM networks is the incorporation of memory cells that can store information over long periods of time. Memory cells are controlled by three gates: the input gate, the forget gate, and the output gate. The input gate controls whether to add new information to the memory cell, while the forget gate decides what information to discard from the cell. The output gate controls whether the information in the memory cell should be used to make a prediction.

Long short-term memory (LSTM) networks are a type of RNNs that have been particularly successful in time series forecasting tasks [31]. Nevertheless, they can also be involved in classification, either side by side with a CNN model [32], or following CNN layers [11]. LSTMs are designed to address the vanishing gradient problem that can occur in traditional RNNs, and they are capable of capturing long-term dependencies in the data.

In general, RNNs improve the learning capability of the model on time series data, as they learn from historical data. However, they are ineffectual with relatively long sequences, besides being computationally expensive due to sequential processing.

2.1.2.2 Attention Mechanism

The attention mechanism was proposed to allow the model to focus on specific parts of the input. It was used to enhance LSTM performance in many applications by assigning attention scores for LSTM hidden states to determine the importance of each state in the final prediction [33]. For example, TCACNet proposed a dual attention mechanism to learn temporal relations and information distribution over channels from electroencephalogram (EEG) data [34]. Self-attention was also leveraged in many works to replace the LSTM network due to its fast parallel processing [35].

2.1.2.3 Transformers

Recently, the transformer network, which is based on the attention mechanism, has gained popularity in time series analysis [36]. Originally developed for natural language processing [37], transformer networks can be adapted for time series data by treating the time series as a sequence of tokens. The self-attention mechanism

in transformer networks allows them to capture dependencies between different parts of the sequence, which can be useful for identifying complex patterns in the data. Attention mechanisms can be more effective with long sequences, and they can improve the interpretability of the model [38]. However, they can be more computationally expensive due to their larger number of parameters [39], which also makes them prone to overfitting when it comes to the data scarcity problem.

2.1.2.4 Non-deep Learning Methods

In addition to deep learning-based methods for extracting temporal dynamics, there are several traditional and non-deep learning methods that have been effectively utilized in time-series analysis:

Dynamic Time Warping (DTW): DTW is a technique that measures the similarity between two temporal sequences, which may vary in speed. By allowing for certain non-linear mappings, it can identify patterns that are out-of-phase or have time shifts. This is particularly useful in understanding the shape features of time-series data.

Fourier Transformation: Fourier Transformation, specifically the Fast Fourier Transform (FFT), breaks down a time series into its constituent sinusoids of different frequencies. This allows for the extraction of periodic information, identifying dominant frequencies, and understanding repetitive patterns and trends.

First-order and Second-order Differences: By calculating the first-order difference, we get the sequential change in the values of a time series, helping in identifying trends. The second-order difference, on the other hand, gives us an understanding of the change in the first-order difference, which can be useful in identifying the acceleration or deceleration in trends. Both of these methods are valuable for extracting specific time-domain features.

These traditional methods, while not as flexible as deep learning approaches in some contexts, offer unique advantages in specific scenarios, especially when the computational resources are limited or when interpretability and understanding of the temporal dynamics are of prime importance."

In conclusion, feature extraction modules in deep learning play a crucial role in time series analysis and prediction. Convolutional neural networks, recurrent neural

networks, long short-term memory networks, and transformer networks are all commonly used feature extraction modules for time series data. These architectures can be combined with a variety of techniques to improve the extraction of useful features from the data, which can lead to improved performance in downstream tasks such as classification, clustering, and forecasting.

2.2 Label-Efficient Time Series Representation Learning

In this section, we discuss another approach toward the real-world applicability of deep learning models. Specifically, we investigate the approaches used to address the label scarcity problem, i.e., self-supervised learning and semi-supervised learning. The former is used to learn representations from fully unlabeled data, while the latter is preferred for learning from data having few labeled samples.

2.2.1 Self-supervised Learning

Self-supervised learning approach gained much attention recently for training deep learning models with unlabeled data. Compared with models trained on fully labeled data (i.e., supervised models), self-supervised pretrained models can extract effective features and achieve comparable performance when fine-tuned with a small percentage of the labels [40, 41]. Self-supervised learning involves defining a new task to be solved by the model or generating pseudo labels based on the input data, instead of relying on human-provided labels.

Self-supervised learning methods can be categorized into auxiliary-based or contrastive-based methods. In both categories, the unlabeled data is used to learn an additional task, that is not the main focus of the model, to provide supervision during training.

2.2.1.1 Prelimiaries

Problem Formulation

We assume an input time series data in $\mathbb{R}^{N \times l}$ with N channels and sequence length of l. We assume a time series classification problem, where each sample is assigned one label from C classes. During the self-supervised learning task, we assume access only to the input data without labels. These labels can be available for the supervised downstream task.

The deep neural network is assumed to have two main parts. The first is the feature extractor, which maps the input data into the embedded space $f_\phi : \mathbb{R}^{N \times l} \to \mathbb{R}^{\mathcal{N} \times d}$ parameterized by neural network parameters ϕ, where $\mathcal{N}$ is the new number of channels after passing through the feature extractor network, and d is the new feature length. The second is the classifier network $f_\gamma : \mathbb{R}^{\mathcal{N} \times d} \to \mathbb{R}^{\mathcal{N} \times C}$, which produces the predictions. The self-supervised learning algorithm learns ϕ from unlabeled data, while during the fine-tuning learns γ while also updating ϕ.

Auxiliary tasks

The recent rebirth of self-supervised learning began with manually-designed auxiliary tasks. This category includes defining a new task along with free-to-generate pseudo labels. These tasks can be defined as classification, regression, or any others. For example, Saeed et al. [42] and Sarkar and Etemad [43] proposed new classification auxiliary tasks by generating several views to the signals using augmentations, e.g., adding noise, rotation, and scaling for human activity recognition and arrhythmia classification respectively. Besides training the model to learn representations about data, these augmentations were also meant to improve the robustness of the model against the different variations that could occur in testing data. For example, introducing random noise to the original signal can enhance the model's robustness against noisy samples. Similarly, applying a random scalar to scale the signal magnitude can strengthen the model's resilience against amplitude and offset invariances. Lastly, signal rotation through sign inversion emulates scenarios in which the sensor is held in a reversed orientation, further enhancing the model's robustness against sensor-placement invariance. To train this auxiliary task, each view is assigned a label, and the model was pretrained to classify these transformations. This approach showed success in learning underlying representations from unlabeled data.

Formally, let's assume a tuple of an input signal and its corresponding pseudo label $(x_i, \hat{y}_i)$, where x_i is i^{th} transformed signal, $\hat{y}_i$ is the generated pseudo label that corresponds to the k^{th} transformation, and $k \in [0, T)$, T is the total number of transformations. Next, the transformed signal passes through the feature extractor and the classifier networks to generate the output probability p_t. Last, the model is trained to minimize a standard cross-entropy loss based on these pseudo labels: $\mathcal{L}_{\text{CE}} = \sum_{t=0}^{T-1} \mathbb{1}_{[\hat{y}=t]} \log p_t$, where $\mathbb{1}$ is the indicator function, which is set to be 1 when the condition is met, and set to 0 otherwise.

Contrastive learning

In contrastive learning, representations are learned by comparing the similarity between samples. In specific, we define positive and negative pairs for each sample. Next, the feature extractor is trained to achieve the contrastive objective, i.e., to push the features of the sample towards their positive pairs, and pull them away from their negative pairs. In many previous works, these pairs were usually generated via data augmentations [41, 44]. Notably, some studies relied on strong successive augmentations and found them to be key factors in the success of contrastive techniques [41].

Formally, given a dataset with N unlabeled samples, we generate two views for each sample $\mathbf{x}$, i.e., $\{\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j\}$ using data augmentations. Therefore, in a multi-viewed batch with N samples for each view, we have a total of $2N$ samples. Next, the feature extractor transforms them into the embedding space, and a projection head $h(\cdot)$ is used to obtain low-dimensional embeddings, i.e., $\mathbf{z}_i = h(f_\phi(\hat{\mathbf{x}}_i))$ and $\mathbf{z}_j = h(f_\phi(\hat{\mathbf{x}}_j))$. Assuming that for an anchor sample indexed $i \in I \equiv \{1 \dots 2N\}$, and $A(k) \equiv I \setminus \{k\}$. The objective of contrastive learning is to encourage the similarity between positive pairs and separate the negative pairs using the NT-Xent loss, defined as follows:

$$\mathcal{L}_{\text{NT-Xent}} = \frac{-1}{2N} \sum_{i \in I} \log \frac{\exp\left(\mathbf{z}_i \cdot \mathbf{z}_j / \tau\right)}{\sum_{a \in A(i)} \exp\left(\mathbf{z}_i \cdot \mathbf{z}_a / \tau\right)}, \tag{2.1}$$

where $\cdot$ symbol denotes the inner dot product, and τ is a temperature parameter.

Downstream tasks

Downstream tasks are the main tasks of interest that lacked a sufficient amount of labeled data for training the deep learning models. To illustrate, consider the sleep stage classification task, i.e., classifying the PSG epochs into one of five classes, i.e., W, N1, N2, N3, and REM. This can be considered as the downstream task, where we aim to propose a pretext task to pretrain the model so that we can improve the model performance for this downstream task. Notably, different self-supervised tasks can have a different impact on the same downstream task. Therefore, it is important to design a relevant task to the problem of interest, which helps achieve better downstream performance. Despite the numerous self-supervised learning proposed in the literature, identifying the proper pretext task is still an open research question [45].

In the next subsections, we first discuss the self-supervised learning methods proposed for image data (because they are the first works in this field), then discuss those specific for time series data.

2.2.1.2 Self-supervised Learning Approaches

Auxiliary Pretext Tasks Self-supervised learning has been gaining more attention recently, as it deals with the emerging problem of learning from unlabeled data [45–47]. The first works in self-supervised learning have been auxiliary tasks. For example, Zhang et al. [48] proposed image colorization, where they trained their model to color a grayscale input image. Misra and Maaten [49] proposed learning invariant representations based on pretext tasks by encouraging the representation of both the pretext task and the original image to be similar. For video-related tasks, Srivastava et al. [50] proposed different tasks such as input sequence reconstruction, or future sequence prediction to learn representations of video sequences. In addition, Wei et al. [51] proposed training the model to check if the order of the input frame sequences is correct or not. Another direction is to use ranking as a proxy task to solve regression problems [52], in which the authors developed a backpropagation technique for Siamese networks to prevent redundant computations.

Although using pretext tasks can improve representation learning, they are found to limit the generality of the learned representations. For example, classifying the

different rotation angles of an image may deviate the model from learning features about the color or orientation of objects [53].

Contrastive Learning On the other hand, contrastive methods intend to learn invariant representations from different augmented views of data. The different methods vary between each other in their ways of choosing negative samples against positive samples during training. SimCLR [41] considered the augmented views of the original image as the positives, while all the other views of different images within the batch are treated as negatives. This technique benefits from larger batch sizes to accumulate more negative samples. Another approach was to accumulate a large number of the negative samples in a memory bank as proposed by MoCo [54]. The embeddings of these samples are updated with the most recent ones at regular intervals.

However, one limitation of those methods was the way of selecting the negative pairs, as they may select negative pairs from samples having the same class as the anchor sample. Hence, some methods proposed to use contrastive learning without the need for negative samples. For example, BYOL [55] learned representations by bootstrapping representations from two neural networks that interact and learn from each other. Also, SimSiam [56] supported the idea of neglecting the negative samples and relied only on a Siamese network and stop-gradient operation to achieve state-of-the-art performance.

2.2.1.3 Self-supervised Learning for Time Series

Auxiliary Pretext Tasks The first time series auxiliary tasks started with classifying transformations, where the model is expected to learn useful representations about the data by learning to be invariant against these transformations. For example, Saeed et al. [57] designed a binary classification pretext task for human activity recognition by applying several transformations on the data and trained the model to classify between the original and the transformed versions. Similarly, Sarkar and Etemad [43] proposed SSL-ECG, in which ECG representations are learned by applying six transformations to the dataset as pretext tasks, and assigned pseudo labels according to the transformation type.

Another approach was proposed by Banville et al. [58], who introduced two auxiliary tasks, i.e., relative positioning and temporal shuffling, to learn the underlying temporal characteristics in the EEG signals. In the first task, a window of the signal is used as an anchor to measure the relative position of other windows, while in the second task, a window of the signal is shuffled to teach the model to fix the temporal order of the signal. Similarly, Saeed et al. [59] proposed some auxiliary tasks, e.g., feature prediction from a masked window, temporal shift prediction, modality denoising, and blend detection, to learn representations from human activity recognition data. In addition, TST [36] and STraTS [60] proposed to predict the masked part in the signal as an auxiliary task to pre-train a Transformer model. Additionally, Aggarwal et al. [61] learned subject-invariant representations by modeling local and global activity patterns. However, as for image data, most time series self-supervised learning works have recently adopted contrastive learning instead.

Contrastive Learning Most of the recent self-supervised learning literature for time series data adopts contrastive learning. Previous works propose different strategies to choose the positive and negative pairs to improve the quality of the learned representations. Additionally, most works attempt to learn the temporal relations within time series in the pretraining phase. In this section, we provide a new time series-specific categorization for contrastive-based self-supervised learning methods. Particularly, we categorize them into intra-sample contrastive methods, inter-sample contrastive methods, and frequency domain contrastive methods.

Inter-sample Contrasting (Contextual Contrasting) In inter-sample contrastive learning, the model uses the input signal as a whole without splitting or windowing. One of the well-performing visual contrastive methods is SimCLR [41], which relies on data augmentations to define positive and negative pairs. Specifically, the positive pairs are the augmented views of an anchor sample, while the negative pairs are all the augmented views of other samples. Many works followed SimCLR for time series data, such as [62, 63] that apply SimCLR-like methods for sleep stage classification and fault diagnosis tasks respectively.

Mixup Contrastive Learning (MCL) [64] proposes to replace the traditional contrastive loss with a mixup-based contrastive loss. Specifically, MCL generates new mixed samples in each mini-batch using the mixup operation, and the proposed loss predicts the mixup ratio among each pair of samples. Also, Chang et al. [65]

propose another loss that searches for positive and negative pairs for each anchor sample in both the input and embedding spaces. The contrastive loss is then applied to the selected triplet to learn a causal temporal CNN.

Intra-sample Contrasting (Temporal Contrasting) Intra-sample contrastive methods aim to learn the temporal relations in time series data in the pretraining phase. To do so, they process the signal at the timestep level, either in the embedding space or in the input space. One of the earliest works is Contrastive Predictive Coding (CPC) [53], which learns the temporal relations by splitting the embedding space of the signal into a past part and a future part. Then, an autoregressive model is trained to produce a context vector from the past part and use it to predict the future part using contrastive learning. Similarly, SleepDPC [66] generates multiple context vectors and uses each one to predict the next timestep in the future part for the sleep stage classification task. Lastly, ContrNP [67] replaces the autoregressive model used by CPC with a neural process.

Other methods learn the temporal relations among timesteps in the input space. For example, Temporal Neighborhood Coding (TNC) [68] assumes that the distribution of one window of timesteps is similar to its neighboring windows and can be more distinguishable than the non-neighboring windows. Therefore, TNC forms positive pairs among neighboring windows and negative pairs among non-neighboring windows. Another example is TS2Vec [69], which applies data augmentation on the input signal and considers the current timestep from the augmented views as positives, while the different timesteps from the same time series are considered as negatives.

Frequency Domain Contrastive Methods A recent trend in time series contrastive learning is to utilize the frequency domain characteristics in representation learning, which is one main distinguishing properties of time series data. For instance, Bilinear Temporal-Spectral Fusion (BTSF) [70] applies an iterative bilinear fusion between feature embeddings of both time and frequency representations of time series. Specifically, BTSF generates different views of time series and applies iterative aggregation modules to perform cyclic refinement to the temporal and spectral features. Similarly, both STFNets [71] and TF-C [72] apply time domain and frequency domain augmentations, and push the time domain and frequency domain representations of the same sample closer to each other while pushing them apart from representations of other signals.

Existing approaches used either temporal or global features. Differently, we address both types of features in our cross-view temporal and contextual contrasting modules. These modules rely on different views for input data that we provide via time-series-specific augmentations.

2.2.2 Semi-supervised Learning for Time Series

Semi-supervised learning is a type of learning that involves training a model using both labeled and unlabeled data. It is used in situations where it is difficult or expensive to obtain a large amount of labeled data and only a few labeled samples are available with a large amount of unlabeled data. One of the challenges in semi-supervised learning is how to achieve the best balance of learning from both labeled and unlabeled samples. In this section, we categorize the semi-supervised learning approaches into self-training-based methods and generic regularization-based methods.

2.2.2.1 Self-training Methods

Self-training involves assigning labels to the unlabeled data to train the model. It can be implemented either by pseudo-labeling or self-supervised pretraining.

Pseudo Labeling Pseudo labeling utilizes the few labeled data to train the model and create pseudo labels for the unlabeled data that are likely to be correct. The common way to create pseudo labels is to apply the softmax function on the model predictions. For example, Wang et al. [73] generated pseudo labels from the model predictions to the unlabeled data and used least-square regression to learn both shapelets and classification boundaries. Another way is to use clustering techniques as in SUCCESS [74], which proposed a hierarchical clustering scheme to assign pseudo labels to samples according to their respective clusters. SUCCESS calculated the distance to centroids based on the dynamic time warping (DTW) distance metric.

A different way of utilizing the pseudo labels is proposed in FixMatch [75], which applied two augmentations to the image data, where one augmentation is weak and the other is strong. Then, if the prediction of the weak-augmented view is beyond a

threshold value, it uses its pseudo label for the strong augmented view. By showing success with image data, several methods applied FixMatch-like approaches on time series data, e.g., PARSE [76] for EEG-based emotion recognition. Also, SelfMatch [77] used Gaussian noised signals as a weak augmentation and timecut as a strong augmentation. It also applied self-distillation to guide the lower-level blocks of the feature extractor with the knowledge obtained in the output layer.

Self-supervised Learning Instead of pseudo-labeling the unlabeled portion of the data, we can leverage self-supervised learning to learn their representations, and then fine-tune the pretrained model with the available labeled portion. For example, Jawed et al. [78] proposed a forecasting task on the unlabeled data to provide a substitute supervision signal. This task had been jointly optimized along with the supervised task on the labeled part of the data. In addition, iTimes [79] learned from unlabeled samples by applying several transformations to the time series via irregular time sampling techniques and learns the model to predict the transformation type to learn the temporal structure.

Contrastive learning has also been widely used for self-supervised pretraining in semi-supervised learning. For example, SemiTime [80] used contrastive learning by splitting the signal into two parts, i.e., past and future parts. It formed positive pairs among the past and the future parts of the same signal, and negative pairs between the past of the signal and the future of other signals. Moreover, MtCLSS [81] applied contrastive learning between the feature representations of the labeled and unlabeled EEG samples for the sleep stage classification problem. Last, Semi-DeepConvNet [82] used a SimCLR-like contrastive learning technique to learn from EEG-based motor imagery data. It also eliminated the bias in the data from the different subjects by using an adversarial training scheme.

2.2.2.2 Generic Regularization

Apart from self-training, another stream of methods adopts different techniques to learn from data, e.g., ensembling and generative models.

Ensemble-based Methods These methods train two or more models separately on the same or on different views of the data. For example, TEBLSTM

[83] proposed temporal ensembling for human activity recognition data based on an LSTM model. In addition to the supervised cross-entropy loss on the labeled samples, they trained the unlabeled data by comparing their predictions with the ensembled predictions of the past epochs. Moreover, Chen et al. [84] clustered labeled samples using k-means to train multi-modal classifiers. If the ensemble of these classifiers reaches an agreement on predicting an unlabeled sample, it is assigned a label and added to the labeled set. The newly labeled samples are then added to the training set and the models are retrained on the expanded dataset.

Reconstruction-based Methods These methods train the model to reconstruct input data from a latent representation assuming that the model will learn useful representations from this process. For example, HCAE [85] trained an autoencoder on industrial motor bearing datasets to reconstruct the spectrogram of the signals. They trained a supervised loss on the labeled samples besides the reconstruction loss. Similarly, AnoVAE [86] trained a variational autoencoder on normal EEG signals and marked the anomalous signals by the trained model as seizures. Also, SMATE [87] employed a reconstruction loss to learn from data, used the labeled samples to create centroids, and then classified the unlabeled data according to their distance from the centroids.

Moreover, other methods used GANs to reconstruct and learn from data. For example, REG-GAN [88] is a semi-supervised regression method that uses GAN to learn representations by regenerating the signals of the whole data and applying supervised regression loss on the labeled part. SRGAN [89] proposed a GAN model to learn useful information from suspension histories to improve the remaining useful life prediction performance of industrial machines.

2.3 Cross-domain Learning from Distributionally-shifted Domains

Cross-domain learning refers to training a model on a labeled domain (i.e., the source domain), and transferring the knowledge to a related domain (i.e., the target domain), which is the domain of interest. This is a challenging problem, particularly when the source and target domains are sampled from different distributions.

There are three main learning paradigms under this category. The first is transfer learning, which is used when only a few labeled data are available in the target domain, while more label information is provided in the source domain. The model can be pretrained on the source domain data, and then directly fine-tuned with the few labeled samples in the target domain [90]. Transfer learning can be particularly impactful in time series applications that include multiple users/subjects, such as healthcare or human activity recognition. In such cases, we can improve the model performance on new unseen users with transfer learning instead of training the model from scratch [91]. For example, Li et al. [92] propose to train a single-channel neural network using single-channel data from the source domain on human activity recognition datasets. Then, they replicate these multiple single-channel nets for each target domain channel, showing improved performance.

The second is the Unsupervised Domain Adaptation (UDA) approach, which is considered as a subset of transfer learning, that can be used when no target domain labels are available and a distribution shift exists between source and target domains. UDA adapts a model trained on a labeled source domain to a shifted unlabeled, but related, target domain, to minimize the domain shift. The third is the semi-supervised domain adaptation, where we adapt a model trained on a source domain to a related out-of-distribution target domain, in which only a small amount of labeled data is available for the target domain. The labeled samples in the target domain can be helpful in two aspects. First, it can be used to select the best target model hyperparameters. Second, these samples can be used to boost the adaptation performance of the model.

This thesis primarily focuses on the problem of *unsupervised domain adaptation*, which aims to address label-scarce scenarios by training a model on a labeled source domain and then adapting it to the target domain of interest. The importance of unsupervised domain adaptation (UDA) has emerged with the recognition that deep learning models tend to underperform when the training and testing data are drawn from different distributions, a phenomenon commonly referred to as the domain shift problem. In real-world time series applications, training and testing data can significantly differ in their temporal characteristics and working conditions, making the domain shift problem particularly relevant. UDA approaches seek to mitigate the domain shift problem by adapting a model trained on a labeled source domain to a shifted unlabeled target domain.

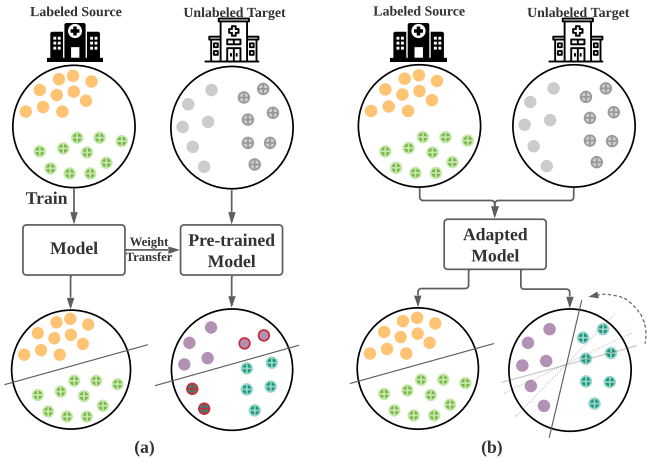


FIGURE 2.2: (a) Direct Transfer (DT) fails due to the domain shift, and (b) Unsupervised Domain Adaptation (UDA) solves the domain shift problem.

To provide further clarity, consider Fig.2.2(a), which depicts the *Direct Transfer* approach where the source-pretrained model is directly applied to the target data. Due to the domain shift, the target domain data may not be well-classified, resulting in significant performance degradation of deep learning models. However, UDA strategies aim to address the domain shift by adapting the model trained on the source domain to the target domain, as demonstrated in Fig.2.2(b).

UDA can be broadly classified into two categories: distance-based methods and adversarial-based methods. Distance-based methods employ a distance measure, such as maximum mean discrepancy (MMD) [93] and correlation alignment (CORAL) [94], to gauge the distribution shift between the source and target domains. These methods aim to align the feature distributions of the two domains by minimizing the distance between them. Adversarial-based UDA methods, on the other hand, train a discriminator model to produce features that are indistinguishable between the source and target domains, typically using an adversarial loss [95].

Despite the extensive literature on UDA for visual applications [96–98], it remains relatively unexplored for time-series data. In the following subsections, we first discuss the earliest works in UDA and then delve into those explicitly proposed for time-series data.

2.3.1 Factors Influencing Covariate Shift in Time Series Data Across Domains

Distribution shifts in time series data are ubiquitous across various domains, and understanding the factors leading to these shifts is crucial for accurate and robust analyses. Some of the key factors leading to this distribution shift include:

- **Differences among subjects/patients**: this includes any source of sensory data from human subjects. In healthcare systems, patients with different health conditions, stages of diseases, or genetic predispositions might show varied physiological responses. For instance, ECG readings from a healthy individual will differ significantly from someone with cardiac ailments. Also, in human activity recognition, different people have distinct ways of performing activities. For example, two individuals climbing stairs might have different stride lengths, speeds, and rhythms, leading to distribution shifts when comparing their data.
- **Instrumental differences**: Different instruments might have slight variations in calibration, causing shifts in recorded data. For instance, blood pressure measured using two different devices might vary slightly. In addition, devices capturing data might have different sampling rates, leading to time series data with varying granularity. For example, one accelerometer might capture data every second, while another might capture every half second.
- **Sensor placement**: The location and orientation of sensors can change the way activities are recognized. For instance, an accelerometer on the wrist might record different data patterns compared to one placed on the ankle.
- **Operational environment changes**: Variations in environmental conditions such as temperature, humidity, or even the quality of raw materials can alter the performance of machines, leading to distribution shifts.

Building on the aforementioned scenarios, it's evident that time series data, especially in domains like healthcare, manufacturing, and activity recognition, are susceptible to covariate shifts, *which is our focus in this thesis*. Covariate shift assumes that while the distribution of the features ($P(X)$) may change between

the source and target domains, the conditional distribution $P(Y|X)$ remains the same. In other words, the way the features relate to the labels doesn't change, even though the distribution of the features themselves might. Covariate shift encapsulates the very essence of these changes in feature distributions between the training (source) and test (target) data while maintaining a consistent relationship between the features and outcomes. For instance, while the readings from different health statuses or machine states represent disparate feature distributions, the intrinsic link between these features and their corresponding outcomes or labels remains stable. This type of shift emphasizes the need for domain adaptation techniques that can align these distributions to ensure model generalizability. Addressing covariate shift is particularly crucial in time series data due to the sequential nature of the data, where temporality can accentuate distributional discrepancies. By identifying and mitigating covariate shift, we can harness the underlying shared structure between source and target domains, thereby enhancing the robustness and applicability of predictive models across diverse settings.

2.3.2 Unsupervised Domain Adaptation Approaches

2.3.2.1 Distance-based UDA Methods

Some of the UDA methods proposed for images focused on matching the statistical distribution of embeddings to learn domain invariant representations. For example, DDC [99] trained an adaptation layer to jointly optimize classification performance and domain invariance based on MMD. Also, HoMM [100] explored aligning higher-order statistics for domain matching. DSAN [101] proposed local MMD to align relevant subdomain distributions. Last, the concept of manifold criterion was introduced as a distance measure to validate the distribution matching across domains [102].

2.3.2.2 Adversarial UDA Methods

Adversarial training is the most widely adopted UDA paradigm for visual UDA. For example, DANN [103] proposed a gradient reversal layer. DIRT-T [104] added iterative refinement training to improve the adversarial training. CDAN [105] applied the adversarial adaptation to the information conveyed from the classifier

predictions. Wang and Zhang [106] proposed a re-weighted adversarial domain adaptation with a triplet loss on the confusing domain to leverage both source samples and pseudo-labeled target samples. Xu et al. [107] included mixup operation on instance- and feature-level to improve the adversarial training. Also, Kang et al. [108] included contrastive learning to enhance the performance of their adversarial training. For video UDA, Sahoo et al. [109] proposed to contrast the embeddings of unlabeled videos at different speeds with a background mixing mechanism. While these UDA techniques are proposed for visual applications, the problem of unsupervised domain adaptation for time-series data remains relatively under-explored, which is our focus in this work.

2.3.2.3 Other UDA Methods

Since distance-based and adversarial-based methods can suffer from several drawbacks, existing research developed other techniques to adapt source and target domains. For example, Zhang et al. [110] exploited the importance-reweighting techniques, which assigns weights to samples in the source domain to reduce the distribution discrepancy with the target domain. Methods like Kernel Mean Matching (KMM) and the Maximum Mean Discrepancy (MMD) criterion utilize this approach.

In addition, other methods attempt to maintain consistent mappings between source and target domains. For example, Cycada method [111] aims to guide transfer among domains based on a specific discriminative task. It avoids divergence by enforcing consistency of the relevant semantics before and after adaptation.

In some other scenarios, where the label distribution varies significantly between the source and target domains, some approaches attempt to achieve intra-class alignment. This approach aims to minimize the discrepancies within each class across domains. Therefore, in cases of class imbalance, intra-class alignment techniques can be more beneficial. To achieve that, Li et al. [112] proposed a reweighting network to produce hard labels, class-level weights for source features, and soft labels. These allow to produce instance-level weights for the target features.

2.3.3 Unsupervised Domain Adaptation for Time Series

Few works were proposed for time-series UDA, despite its importance in many real-world applications.

2.3.3.1 Distance-based Methods

Some time series UDA methods found the distance measure to be less effective in mitigating the large distribution shift, and they attempted to modify these distance measures to achieve better performance. For example, AdvSKM [113] refines the MMD measure to be suitable for time series, by updating the inner kernel of MMD to become a hybrid spectral kernel network. In addition, Ott et al. [114] pretrain the model on the unlabeled target domain data, then train an optimal transport jointly with CORAL to transform the source to the target domain. They also measure the class-wise similarity between the source and target embeddings to select the best transformation and use it to classify transformed embedding with the target domain model. Also, SASA [115] considers the variant information in time series and learns the sparse associative structure by studying the causal structure of time series variables, building on top of an LSTM architecture.

2.3.3.2 Adversarial UDA Methods

Similar to image data, adversarial-based methods are found to be more efficient for time series UDA. In these methods, the source and target data are passed through a discriminator network, which is trained to make the two domains indistinguishable. The common architecture for this discriminator is either a fully connected or a CNN-based network. However, SLARDA [116] proposed an autoregressive domain discriminator that considers the temporal features when distinguishing source and target domains. It also pretrains the encoder with a forecasting task to boost the performance. Finally, a mean-teacher model is used to learn the fine-grained class distribution of the target domain. In most works, the feature extractor architecture is usually shared between source and target domains. However, Domain Adaptation Forecaster (DAF) [117] used an unshared feature extractor and a shared attention-based architecture for time series UDA forecasting. The shared attention module keeps the domain-specific features, and the domain discriminator leverages

the attention output to perform the adversarial training. Also, ContrasGAN [118] leveraged a bidirectional GAN to adapt the source and target domains with two unshared generators and discriminators to regenerate samples between domains. They also discriminated classes by leveraging contrastive learning with adaptation to minimize the intra-class discrepancy. To benefit from the availability of multiple source domains in human activity recognition, CoDATS [119] proposed a weakly supervised multi-source UDA. It learned domain invariant features with adversarial training, based on a gradient reversal layer (GRL) to flip the gradients when being back-propagated. Adversarial-based UDA methods are computationally expensive and can be more difficult to implement than distance-based UDA methods. However, they are more effective at reducing the large domain shift between source and target domains.

2.4 Use Case — Sleep Stage Classification

In this section, we examine sleep stage classification, being on the time series applications, and review the prior research conducted to address the deep learning-related issues discussed earlier. This is particularly relevant because we proposed works specifically for the sleep stage classification problem.

2.4.1 Representation Learning

2.4.1.1 Feature Extraction

The first studies adopted conventional machine learning methods to classify EEG signals into corresponding sleep stages. These methods usually consist of two steps, namely, manual feature extraction and sleep stage classification. First, they designed and extracted various manual features from time and frequency domains. Feature selection algorithms are often applied to further select the most discriminative features. Second, the selected features are then fed into conventional machine learning models for sleep stage classification, such as Naive Bayes [120], support vector machines [121, 122], random forests [7, 123], or ensemble learning based classifiers [124]. However, these methods require domain knowledge to extract the best representative features.

The deep learning-based automatic sleep stages classification studies have designed convolutional neural networks (CNNs) [125–130]. For example, successive convolution and pooling layers with fully connected layers were used to perform this classification task in [125]. Sors et al. [126] used 12 convolution layers together with 2 fully connected layers. Additionally Chambon et al. [128] used 2D convolution along with MaxPooling layers to classify the raw data from three channels (*i.e.*, EEG, EOG, and EMG). Sokolovsky et al. [127] designed a relatively deep CNN architecture to show that a better performance can be achieved by relying on network depth. Phan et al. [129] converted each raw signal into log-power spectra and used a CNN to perform a joint classification and prediction task for identifying sleep stages. Generally, the CNN models above have achieved good performance for sleep stage classification.

2.4.1.2 Learning Temporal Relations

Recurrent Neural Networks (RNNs) were proposed to capture temporal dependencies in time-series EEG data. For example, Michielli et al. [131] used a cascaded RNN architecture to classify sleep stages. Some researchers combined CNN with RNN by using CNN for feature extraction and RNN for modeling the time dependencies [11, 132–134]. For example, Supratak et al. [11] used a CNN architecture to extract features from the raw data, and then exploited the long short-term memory (LSTM) to learn transition rules through sleep stages. Similarly, Malafeev et al. [134] used successive blocks of CNN followed by LSTM to classify raw EEG signals. In addition, Mousavi et al. [135] used an LSTM-based encoder-decoder with an attention mechanism after the encoder to find out the most relevant parts in input sequences. Recently, some works used the attention mechanism completely instead of using RNN. For example, Zhu et al. [136] segmented the EEG epochs and used self-attention to learn both intra-epoch features and inter-epoch temporal features.

2.4.2 Self-supervised Learning

The success of SSL in computer vision applications motivated their adoption for sleep stage classification. For example, Mohsenvand et al. [137] and Jiang et al. [62] proposed SimCLR-like methodologies and applied EEG-related augmentations for

sleep stage classification. Also, Banville et al. [58] applied three pretext tasks, i.e., relative positioning, temporal shuffling, and contrastive predictive coding (CPC) to explore the underlying structure of the unlabeled sleep EEG data. The CPC [53] algorithm predicts the future timesteps in the time-series signal, which motivated other works to build on it. For example, SleepDPC [66] solved two problems, i.e., predicting future representations of epochs, and distinguishing epochs from other different epochs. In addition, SSLAPP [138] developed a contrastive learning approach with attention-based augmentations in the embedding space to add more positive pairs. Last, CoSleep [139] and SleepECL [140] are yet another two contrastive methods that exploit information, e.g., inter-epoch dependency and frequency domain views, from EEG data to obtain more positive pairs for contrastive learning.

2.4.3 Unsupervised Domain Adaptation

Few works have been proposed for UDA in the sleep stage classification problem. For example, Zhao et al. [141] proposed an adversarial UDA based on a domain discriminator and multiple classifiers fed from the different feature extractor layers. Also, Nasiri and Clifford [142] used adversarial training along with local and global attention mechanisms to extract the transferable individual information. Yoo et al. [143] proposed using adversarial domain adaptation with three discriminators; one for global alignment and two for stage and subject discrimination.

2.5 Summary

This thesis chapter reviews the time series representation learning in general besides some methods towards label-efficient representation learning. The chapter begins by discussing the deep learning model architectures including convolutional neural networks, recurrent neural networks, and attention mechanisms. Next, we introduce self-supervised learning, as a technique to learn useful representations from unlabeled data. We discuss its categories, i.e., the auxiliary tasks, contrastive learning, and downstream tasks. Next, we cover semi-supervised learning for time series, which includes pseudo-labeling, self-supervised learning, and generic regularization methods. After that, we discuss building generalizable cross-domain deep

learning methods for time series, in which we attempt to mitigate the domain shift among distribution-shifted domains. Finally, the chapter discusses a use case of sleep stage classification, including how previous works in this application attempt to learn representations. We discussed feature extraction, learning the temporal relations, and addressing class imbalance. It also covers self-supervised learning and unsupervised domain adaptation work.

Chapter 3

An Attention-based Representation Learning for Sleep Stage Classification

In this chapter [1], we delve into the sleep stage classification problem, which is one of the applications of time series. Automatic sleep stage classification is crucial to measure sleep quality accurately. Specifically, in this chapter, we focus on classifying single-channel EEG signals that represent the time-stamped readings of brain signals for a subject while sleeping.

Hence, we propose a novel attention-based deep learning architecture called AttnSleep to classify sleep stages using single-channel EEG signals. AttnSleep architecture starts with extracting high-quality features using a multi-resolution convolutional neural network (MRCNN) and an adaptive feature recalibration (AFR) module. The MRCNN is designed to extract low and high-frequency features from EEG signals, while the AFR module aims to improve the quality of the extracted features by modeling the inter-dependencies between them. To learn the temporal dependencies among extracted features, we introduce the temporal context encoder (TCE) that leverages a multi-head attention mechanism supported by causal convolutions to learn the relations among the input features. Moreover, we address the

[1] The work in this chapter has been published as "An Attention-based Deep Learning Approach for Sleep Stage Classification with Single-Channel EEG", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2021 [144]. Also, another part has been published as "Self-supervised Learning for Label-Efficient Sleep Stage Classification: A Comprehensive Evaluation", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2023 [145].

class-imbalance problem in the sleep data by developing a class-aware loss function that balances the minority classes without incurring any additional computational costs.

We evaluate the performance of our proposed AttnSleep model on three public datasets. The results demonstrate that AttnSleep outperforms state-of-the-art techniques in terms of different evaluation metrics. Our source codes, experimental data, and supplementary materials are available at https://github.com/emadeldeen24/AttnSleep.

Overall, the main contributions of our proposed model in this chapter can be summarized as follows.

1. We propose a novel feature extraction technique, *i.e.*, a multi-resolution CNN module, to extract features corresponding to low and high frequencies from different frequency bands, and an adaptive feature recalibration to learn the features interdependencies and enhance the representation capability of the extracted features.
2. We propose a novel temporal context encoder that deploys a multi-head self-attention with causal convolutions to efficiently capture the temporal dependencies in the extracted features.
3. We design a class-aware loss function to efficiently handle the class imbalance without introducing additional computations.
4. These novel components are supported by extensive experiments over three public datasets. The results demonstrate that our proposed model outperforms the state-of-the-art methods in sleep stage classification.

3.1 Proposed Method

In this section, we introduce our proposed AttnSleep model for sleep stage classification from single-channel EEG data.

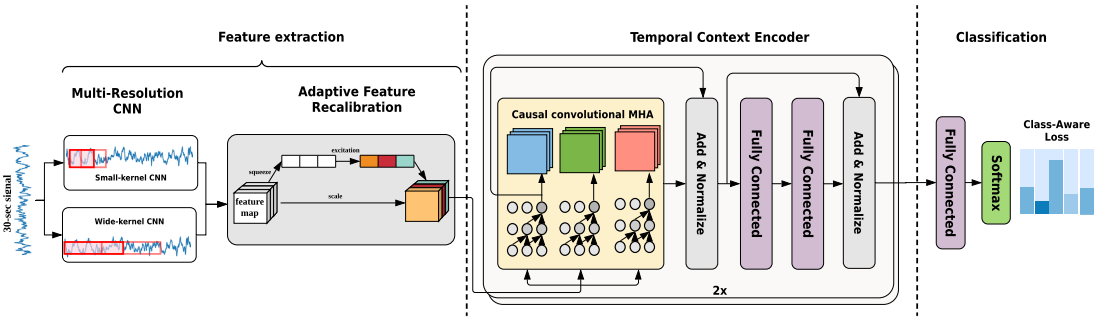


FIGURE 3.1: Overall framework of the proposed AttnSleep model for automatic sleep stage classification.

3.1.1 Overview of AttnSleep Model

Fig. 3.1 illustrates the overall framework of our AttnSleep model. It consists of three main blocks, namely, 1) feature extraction, 2) temporal context encoder, and 3) classification.

First, the MRCNN with two-branch CNN architectures is exploited to extract the features from a 30-second EEG signal. In particular, it extracts high-frequency features by the small kernel convolutions and low-frequency features by the wide kernel convolutions. Following MRCNN, we propose an AFR module to model the inter-dependencies among the features extracted by MRCNN. Moreover, AFR can adaptively select and highlight the most important features, which helps to enhance classification performance. Second, we develop a TCE module to capture the long-term dependencies in the input features. The core component of TCE is the multi-head attention supported by causal convolutions. Third, the classification decision is done by a fully connected layer with a softmax activation function. We also leverage a class-aware cost-sensitive loss function to handle the data imbalance issue. In the following subsections, we will introduce each block in detail.

3.1.2 Feature Extraction

Fig. 3.2 shows the MRCNN and AFR modules for feature extraction from raw single-channel EEG signals.

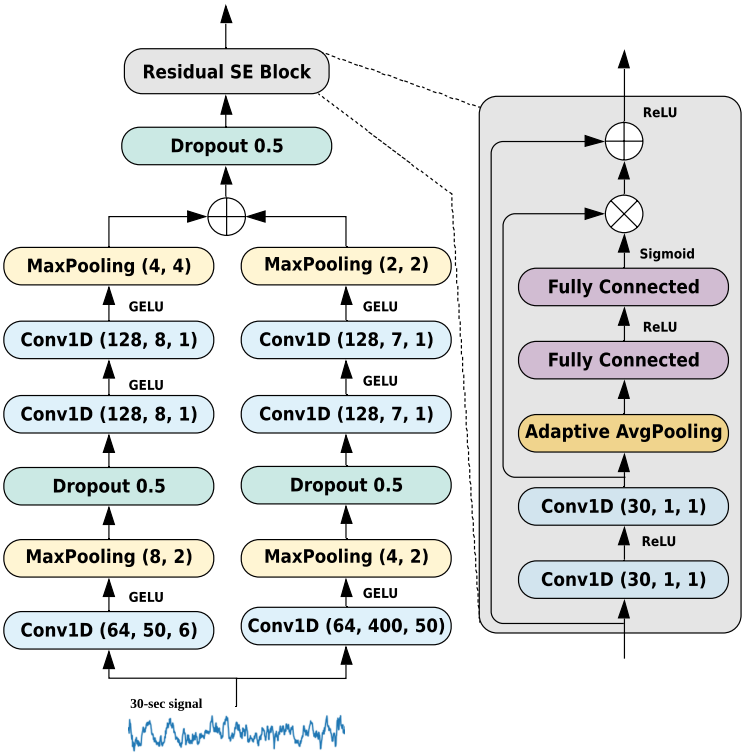


FIGURE 3.2: The MRCNN and AFR modules for feature extraction. Each convolution block is followed by a Batch Normalization.

3.1.2.1 Multi-Resolution CNN

To extract different types of features, we develop a multi-resolution CNN architecture as shown in Fig. 3.2. We implement two branches of convolutional layers with different kernel sizes, where the choice of the kernel sizes is related to the sampling rate of the EEG signals and aims to explore different frequency bands. This is inspired by some previous works that used multiple CNN kernel sizes to extract different frequency features (*i.e.*, low- and high-frequencies) such as [146, 147]. Additionally, different sleep stages are characterized by different frequency ranges [7], and thus, it is becoming important to address different frequency bands to improve the extracted features. Therefore, we use different kernel sizes to capture different ranges of timesteps and hence address features from different sleep-related frequency bands. To further explain this, we consider a dataset with a sampling rate of 100 Hz (100 timesteps are sampled in one second) to justify the selection of the kernel sizes of the two branches. First, the wide kernel (with a kernel of 400) captures timesteps with 4-second windows and thus captures a whole cycle of the sinusoidal signal down to ∼0.25 Hz ($T = 1/F$). This range corresponds to the

delta band. Second, for the smaller kernel (with a kernel of 50), each convolution window captures 50 samples (0.5 seconds), thus it will be able to capture a whole cycle of the sinusoidal signal down to ∼2 Hz, which means that data corresponds to alpha and theta bands.

On the other hand, such a combination of features is important for the non-stationary characteristic of EEG signals that requires exploring different kinds of features. As shown in Fig. 3.2, each branch consists of three convolutional layers and two max-pooling layers, where each convolution layer includes a batch normalization layer [148] and uses a Gaussian Error Linear Unit (GELU) as the activation function. In particular, Conv1D (64, 50, 6) in Fig. 3.2 refers to using a 1D convolution layer with 64 filters, a kernel size of 50, and a stride of 6. Similarly, MaxPooling (8, 2) refers to a max-pooling layer with a kernel size of 8 and a stride of 2. To reduce overfitting, we also apply dropout after the first Maxpooling in both branches and after the concatenation of both branches as shown in Fig. 3.2.

3.1.2.2 Adaptive Feature Recalibration (AFR)

AFR aims to recalibrate the features learned by MRCNN to improve its performance. In particular, the AFR models the inter-dependencies between the features and adaptively selects the most discriminative features through a residual squeeze and excitation (residual SE) block [149]. The SE block helps the lower layers of the network to exploit more contextual information outside its local receptive field by a context-aware mechanism. In the residual SE block, we implement two convolutions Conv1D (30,1,1) with both the kernel and stride size as 1 and ReLU as the activation function. Given a feature map $I \in \mathbb{R}^{L \times d}$ learned by MRCNN, we apply two convolution operations to I such that $F = Conv2(\,Conv1(I)\,)$, where $F = \{F_1, ..., F_N\} \in \mathbb{R}^{N \times d}$, N is the total number of features, d is the length of F_i $(1 \leq i \leq N)$, and $Conv1$ and $Conv2$ are the two convolution operations in AFR module.

Next, the global spatial information is squeezed by using adaptive average pooling that shrinks $F \in \mathbb{R}^{N \times d}$ to $\mathbf{s} = \{s_1, ..., s_N\}$, where s_i is the average of the d data points in $F_i \in \mathbb{R}^d$, $1 \leq i \leq N$. Two fully connected (FC) layers are then applied to make use of the aggregated information. In particular, the first layer is followed by a ReLU activation function to perform dimensionality reduction, and the second layer

is followed by a smoothing sigmoid activation function to perform dimensionality increasing as shown in Equation 3.1.

$$\mathbf{e} = \sigma(\mathcal{W}_2\,(\delta(\mathcal{W}_1(\mathbf{s})))) \quad \in \mathbb{R}^{N\times d}, \tag{3.1}$$

where σ and δ refer to sigmoid and ReLU activation functions respectively, and $\mathcal{W}_1$ and $\mathcal{W}_2$ represent the two FC layers in AFR. Then, the feature map F is scaled by $\mathbf{e}$ as follows:

$$\mathbf{O} = F \otimes \mathbf{e} \quad \in \mathbb{R}^{N\times d}, \tag{3.2}$$

where $\otimes$ refers to the point-wise multiplication between F and $\mathbf{e}$. We also add a shortcut connection to combine the original input I with the enhanced selected features learned from the residual SE block. The final output of the AFR module is:

$$X = I + \mathbf{O} \quad \in \mathbb{R}^{N\times d}. \tag{3.3}$$

Note that we use the GELU activation function in the MRCNN module as it allows some negative weights of the input to pass through. These negative weights might be important for the following AFR module, leading to different decisions. Compared to ReLU, GELU should perform better, as ReLU suppresses all the negative weights to zeros, and thus the AFR module will not be able to make use of them. However, we use ReLU in the AFR module itself as ReLU aims to avoid exploding/vanishing gradient besides making the computations faster and easier to converge [150]. On the other hand, GELU can be a better choice over some other activation functions that also pass negative values, such as Leaky-ReLU and PReLU. The reason is that these activation functions allow strong negative activations to generate undesirable impacts on the sum of activations feeding the next layers, which generates undesirable effects. Differently, GELU shows more control to bind the effect of these negative activations.

3.1.3 Temporal Context Encoder (TCE)

The TCE layer aims to capture temporal dependencies in the input features. As shown in Fig. 3.1, the TCE layer consists of a multi-head attention (MHA) layer, a normalization layer, and two FC layers. Moreover, TCE stacks two identical structures to generate the final features. As the attention mechanism is a key part

of the TCE module, we first introduce the self-attention mechanism, and then we introduce each component in the TCE layer.

3.1.3.1 Self-Attention

We use self-attention to quantify the interdependence within input features, at which higher weights are assigned to the regions of interest according to each position in the input, while lower weights are assigned for less interesting regions. In particular, given an input $\mathbf{z} = \{z_1, ..., z_N\} \in \mathbb{R}^{N \times d}$ where N is the total number of features, and d is the length of x_i, $1 \leq i \leq N$, this input is transformed into another space using a transforming function $\phi(\cdot)$. In our AttnSleep model, $\phi(\cdot)$ is a causal convolution function.

Next, we calculate a score α_{ij} that indicates the weight at which i-th position is attending to j-th position, as follows:

$$\alpha_{ij} = \frac{\exp\left(s_{ij}\right)}{\sum_{k=1}^{d} \exp\left(s_{ik}\right)}, \tag{3.4}$$

$$s_{ij} = \phi(z_i)\phi(z_j)^{\intercal}. \tag{3.5}$$

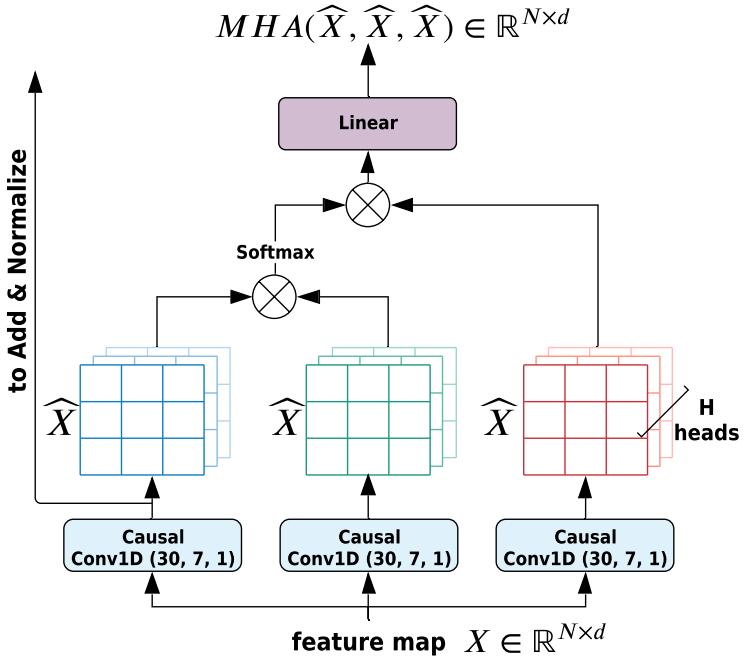


FIGURE 3.3: Structure of proposed Multi-Head Attention.

Each attention output element a_i is computed as a weighted sum of the transformed input elements:

$$a_i = \sum_{j=1}^{d} \alpha_{ij} \phi(x_j). \tag{3.6}$$

The output of the attention layer is $A = (a_0, a_1, \ldots, a_d) \in \mathbb{R}^{N \times d}$.

3.1.3.2 Multi-Head Attention (MHA)

MHA is inspired by the Transformer model [37], which shows great success in machine translation applications due to its ability to learn long-range relationships in sentences [151]. MHA improves self-attention in two main aspects. First, it expands the model's capability to focus on different positions, as the encoding of each head knows about the encodings of the other heads as well. This improves the model's ability to learn temporal dependencies. Second, splitting the input features into different partitions increases the representation subspaces. Therefore, the generated attention weights for each subspace are more representative of the importance of each partition, and concatenating these representations produces better overall representation, which enhances the classification accuracy. In our AttnSleep model, MHA leverages the causal convolutions to encode the positional information of input features and capture their temporal relations. The causal convolutions have the advantage of fast and parallel processing, which significantly reduces the model training time compared with RNNs. Next, we illustrate how MHA works in our AttnSleep model.

The output of the AFR module denoted as $X = \{x_1, ..., x_N\} \in \mathbb{R}^{N \times d}$, serves as the input of MHA as shown in Fig. 3.3. Here, N is the total number of features, and d is the length of x_i, $1 \leq i \leq N$. More specifically, MHA takes three duplicates of X as inputs. First, the causal convolutions generate $\widehat{X}$ from X, *i.e.*, $\widehat{X} = \phi(X)$. Second, we pass the three matrices of $\widehat{X}$ to calculate the attention in Equation 3.7 according to [37].

$$ATT(\widehat{X}, \widehat{X}, \widehat{X}) = Softmax(\frac{\widehat{X} \cdot \widehat{X}^T}{\sqrt{d}}) \cdot \widehat{X}, \tag{3.7}$$

where $(\cdot)$ is the multiplication operation.

We further expand the attention over H heads for each of the three matrices. In particular, each matrix $\widehat{X}$ is split into H subspaces, *i.e.*, $\widehat{X} = \{X^1, \cdots, X^H\}$, $\widehat{X}^h \in \mathbb{R}^{N \times \frac{d}{H}}$, $1 \leq h \leq H$. In each subspace h, we calculate the attention A^h similarly in Equation 3.8.

$$A^h = ATT(\widehat{X}^h, \widehat{X}^h, \widehat{X}^h) \in \mathbb{R}^{N \times \frac{d}{H}}. \tag{3.8}$$

Finally, all the H representations are concatenated together to produce the final output as follows:

$$MHA(\widehat{X}, \widehat{X}, \widehat{X}) = Concat(A^1, \cdots, A^H) \in \mathbb{R}^{N \times d}. \tag{3.9}$$

3.1.3.3 Add and Normalize layer

The TCE has two *Add & Normalize* layers, which add the output of the previous layer to the input of that layer through a residual connection, and then normalize the sum. This operation can be expressed as $LayerNorm(x + SubLayer(x))$, where $LayerNorm$ refers to applying layer normalization [152], $SubLayer$ refers to either the MHA or the two FC layers as shown in Fig. 3.1, and x is the input of the $SubLayer$. Using the residual connections helps the model to utilize the lower-layer features by propagating them to the higher layers if they are useful. Additionally, the normalization operation helps to speed up the training process.

3.1.3.4 Feed-Forward layer

The outputs of the MHA layer are fed into a feed-forward neural network, which is a combination of two FC layers. This layer employs the ReLU activation function to break the nonlinearity in the model and consider the interactions among latent dimensions. This operation can be modeled as $F_{out} = \mathcal{W}_4(\,\delta(\mathcal{W}_3(x))\,)$, where $\mathcal{W}_3$ and $\mathcal{W}_4$ refer to the two FC layers in the TCE module, as shown in Fig. 3.1.

3.1.4 Class-aware Loss Function and Optimization

Basically, we can apply the standard multi-class cross-entropy in Equation 3.10 as the loss function for our model.

$$\mathcal{L} = -\frac{1}{M}\sum_{k=1}^{K}\sum_{i=1}^{M} y_i^k \, log(\mathbf{p}_i^k), \tag{3.10}$$

where y_i^k is the actual label for i-th sample and $\mathbf{p}_i^k$ is the predicted probability of i-th sample for the class k, M is the total number of samples and K is the number of classes. Note that various sleep datasets are imbalanced, *i.e.*, the amount of data for each class varies a lot. The loss function in Equation 3.10 equally penalizes the miss-classification of all the classes, and thus the trained model may be biased towards the majority classes.

We propose a class-aware loss function to address the above issue, which uses a weighted cross-entropy loss as follows:

$$\mathcal{L} = -\frac{1}{M}\sum_{k=1}^{K}\sum_{i=1}^{M} w_k \, y_i^k \, log(\mathbf{p}_i^k), \tag{3.11}$$

$$w_k = \mu_k \cdot max(1, log(\mu_k \, M/M_k)), \tag{3.12}$$

where w_k represents the weight assigned to the class k, μ_k is a tunable parameter, and M_k is the number of samples in class k.

The choice of the class weight w_k relies on *two factors i.e.* the number of samples of this class (controlled by M/M_k), and the distinctness of this class (controlled by μ_k). By analyzing the public sleep data, we can reach out to two conclusions. First, class N2 has a large number of samples, while classes N1 and N3 have much fewer samples (*i.e.*, N1 and N3 are minority classes). Second, we observe that signals of N3 have significantly higher magnitudes than other classes as shown in Fig. 3.4. Hence, the model can easily make correct predictions for N3 samples. Meanwhile, N1 samples are not distinguishable from those in classes N2 and REM as shown in Fig. 3.4. Therefore, we assign the highest μ_k to N1, the lowest to N3, and assign similar values to the other three classes W, N2, and REM as follows.

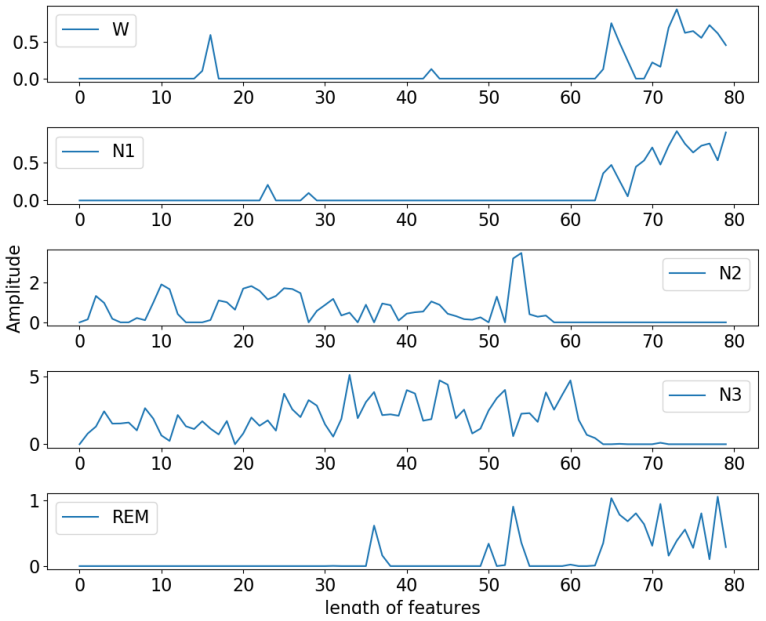


FIGURE 3.4: The amplitude of the extracted features differs among the five classes. This snapshot is from the Sleep-EDF dataset [1].

$$\mu_k = \begin{cases} a/K & k = N3 \\ b/K & k = W, N2, REM \\ c/K & k = N1 \end{cases}$$

where a, b, c are hyperparameters that change for each dataset. To fulfill the above recommendation, we chose $a < b < c$. Note that we use μ_k to scale down the M/M_k value so we keep the values of μ_k less than 1 by dividing them by K. As such, the values of a, b, c are better to be kept less than K to scale down the weights.

Finally, we use Adam [153] as the optimizer to minimize our class-aware loss in Equation 3.11 and learn model parameters.

3.2 Experimental Results

In this section, we first introduce the experimental setup. Then, we demonstrate the evaluation results of our proposed AttnSleep.

TABLE 3.1: Details of three datasets used in our experiments (each sample is a 30-second epoch).

Datasets	#Subjects	EEG Channel	Sampling Rate	W	N1	N2	N3	REM	#Total Samples
Sleep-EDF	20	Fpz-Cz	100 Hz	8285 *19.6%*	2804 *6.6%*	17799 *42.1%*	5703 *13.5%*	7717 *18.2%*	42308
Sleep-EDF-78	78	Fpz-Cz	100 Hz	65951 *33.7%*	21522 *11.0%*	69132 *35.4%*	13039 *6.7%*	25835 *13.2%*	195479
SHHS	329	C4-A1	125 Hz	46319 *14.3%*	10304 *3.2%*	142125 *43.7%*	60153 *18.5%*	65953 *20.3%*	324854

3.2.1 Datasets and Evaluation Metrics

In our experiments, we used three public datasets [2], namely, Sleep-EDF, Sleep-EDF-78, and Sleep Heart Health Study (SHHS) as shown in Table 3.1. For each dataset, we used a single EEG channel for various models in our experiments.

Sleep-EDF and Sleep-EDF-78 were obtained from the PhysioBank [1]. Sleep-EDF contains data files for 20 subjects, while Sleep-EDF-78 is an expanded version with 78 subjects. The participants were involved in two studies. The first is Sleep Cassette (SC* files), which studies age effects on sleep and was conducted on healthy participants aged from 25 to 101 years. The second is Sleep Telemetry (ST* files), which addressed the temazepam effects on sleep in 22 Caucasian males and females without having any other medication. For these two datasets, each PSG file contains two EEG channels (Fpz-Cz, Pz-Oz) with a sampling rate of 100 Hz, one EOG channel, and one chin EMG channel. Following previous studies [125, 127, 129, 130, 154], we adopted the data from the Sleep Cassette study and used the single Fpz-Cz channel as the input for various models in our experiments.

SHHS [155, 156] is a multi-center cohort study of the cardiovascular and other consequences of sleep-disordered breathing. The subjects suffer from various diseases including lung diseases, cardiovascular diseases, and coronary diseases. To minimize the impact of these diseases, we followed the study in [157] to select subjects, who are considered to have regular sleep (e.g., Apnea Hypopnea Index or AHI less than 5). Eventually, 329 out of 6,441 subjects were selected for our experiments. Notably, we selected the C4-A1 channel with a sampling rate of 125 Hz. For the three datasets, we applied the following preprocessing steps. First, we excluded any UNKNOWN stages that don't belong to any of the sleep stages. Second, we merged stages N3 and N4 into one stage (N3) according to AASM

[2]The preprocessed versions of the datasets are available on NTU Dataverse at https://researchdata.ntu.edu.sg/dataverse/attnSleep/

standards. Third, we include only 30 minutes of wake periods before and after the sleep periods to add more focus on the sleep stages [11].

We adopted four metrics to evaluate the performance of various models for sleep stage classification, namely, the accuracy (ACC), macro-averaged F1-score (MF1), Cohen Kappa (κ) [158], and the macro-averaged G-mean (MGm). Both MF1 and MGm are common metrics to evaluate the performance of the models on imbalanced datasets [159]. Given the True Positives (TP_i), False Positives (FP_i), True Negatives (TN_i), and False Negatives (FN_i) for the i-th class, the overall accuracy ACC, MF1 and MGm are defined as follows.

$$ACC = \frac{\sum_{i=1}^{K} TP_i}{M}, \tag{3.13}$$

$$MF1 = \frac{1}{K}\sum_{i=1}^{K} \frac{2 \times Precision_i \times Recall_i}{Precision_i + Recall_i}, \tag{3.14}$$

$$MGm = \frac{1}{K}\sum_{i=1}^{K} \sqrt{Specificity_i \times Recall_i}, \tag{3.15}$$

where $Precision_i = \frac{TP_i}{TP_i+FP_i}$, $Recall_i = \frac{TP_i}{TP_i+FN_i}$ and $Specificity_i = \frac{TN_i}{TN_i+FP_i}$. M is the total number of samples and K is the number of classes or sleep stages.

We also used per-class precision (PR), per-class recall (RE), per-class F1-score (F1), and per-class G-mean (GM) to evaluate our model. They are calculated as in binary classification by considering one class as the positive class and the other four classes as the negative class.

3.2.2 Scoring Performance of AttnSleep

Tables 3.2, 3.3 and 3.4 show the confusion matrices of the proposed model applied on the Fpz-Cz channel in both Sleep-EDF datasets and on C4-A1 channel in SHHS dataset. The confusion matrix is calculated by adding up all the scoring values of the testing data through the 20 folds. Each row represents the number of samples classified by experts, while each column represents the number of epochs predicted by our model. The tables also show the per-class precision, recall, F1 score, and G-mean value for each class.

Table 3.2: Confusion matrix of proposed model applied on Fpz-Cz channel from EDF-20 dataset

	Predicted					Per-class metrics			
	W	N1	N2	N3	REM	PR	RE	F1	GM
W	7432	437	109	24	283	89.6	89.7	89.7	93.5
N1	358	1097	593	6	750	47.1	39.1	42.8	61.6
N2	288	308	15769	493	941	89.1	88.6	88.8	90.3
N3	33	1	535	5119	15	90.7	89.8	90.2	94.1
REM	184	485	702	4	6342	76.1	82.2	79.0	88.0

Table 3.3: Confusion matrix of proposed model applied on Fpz-Cz channel from EDF-78 dataset

	Predicted					Per-class metrics			
	W	N1	N2	N3	REM	PR	RE	F1	GM
W	60545	3607	565	57	1177	92.3	91.8	92.0	93.9
N1	3481	8443	6559	102	2937	45.3	39.2	42.1	60.8
N2	565	3712	59800	2027	3028	83.5	86.5	85.0	88.5
N3	31	12	2276	10686	34	82.3	82.0	82.1	90.0
REM	984	2849	2440	106	19456	73.1	75.3	74.2	85.0

Table 3.4: Confusion matrix of proposed model applied on C4-A1 channel from SHHS dataset.

	Predicted					Per-class metrics			
	W	N1	N2	N3	REM	PR	RE	F1	GM
W	38604	2798	2188	304	2425	90.3	83.3	86.7	90.6
N1	856	3747	2385	46	3270	30.6	36.7	33.2	59.5
N2	1675	2823	123242	6891	7494	87.4	86.7	87.1	88.5
N3	214	19	7051	52695	174	87.5	87.6	87.5	92.3
REM	1405	2858	6143	308	55239	80.5	83.8	82.1	89.1

Notably, stage N1 achieves the lowest performance with the F1 less than 50%, where it is often misclassified to classes W, REM, and N2. In counterpart, class N3 achieves the best performance for Sleep-EDF and SHHS datasets, but it decreases for Sleep-EDF-78 as it is the minority class on this dataset. Most of the misclassifications in the different datasets are with class N2 as it is the majority class.

3.2.3 Baselines and Experimental Setup

In our experiments, we compared our model with five baselines, namely, DeepSleepNet [11], SleepEEGNet [135], ResnetLSTM [160], MultitaskCNN [129] and SeqSleepNet [154]. Brief descriptions for each baseline are as follows.

- **DeepSleepNet** [11] exploits a custom CNN architecture followed by an LSTM with a residual connection for sleep stage classification.
- **SleepEEGNet** [135] employs the same CNN architecture as DeepSleepNet [11] followed by an encoder-decoder with attention mechanism.
- **ResnetLSTM** [160] implements a ResNet architecture for feature extraction, followed by an LSTM to classify EEG signals into different sleep stages.
- **MultitaskCNN** [129] starts by converting the raw EEG signals into power spectrum images and then applies a joint classification and prediction technique using a multi-task CNN architecture for identifying sleep stages.
- **SeqSleepNet** [154] also converts the raw EEG signal into power spectrum images and then uses a hierarchical RNN structure to classify multiple epochs at once.

In particular, we used the published codes for DeepSleepNet [11], SleepEEGNet [135], MultitaskCNN [129] and SeqSleepNet [154], and re-implemented ResnetLSTM [160]. To evaluate the performance of various models, we adopted a subject-wise 20-fold cross-validation by dividing the subjects in each dataset into 20 groups. For example, subject-wise 20-fold cross-validation on the Sleep-EDF dataset with 20 subjects is thus leave-one-subject-out (LOSO) cross-validation. For each round, we selected one group of subjects as testing data and the remaining 19 groups as training data. Eventually, we combined the predicted sleep stages for the testing samples from all 20 rounds to calculate various performance metrics. We chose the neural network hyperparameters based on the performance across these folds. In addition, we noticed that AttnSleep performance stabilizes before reaching 100 epochs, so we trained all the models for 100 epochs in each round to fairly compare their average training time. Fig. 3.5 shows the performance graph of our model showing both the loss and accuracy during AttnSleep training. Our model shows

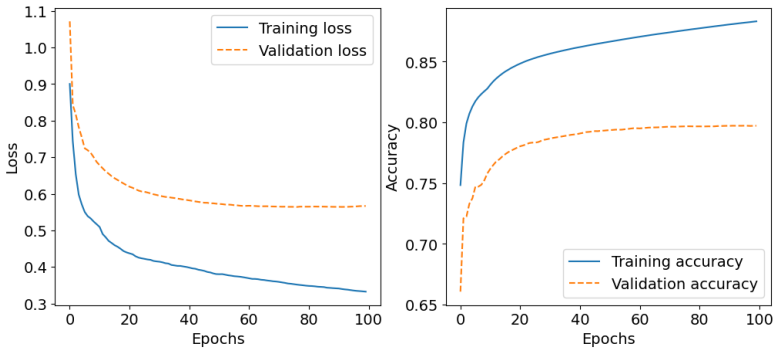


FIGURE 3.5: Training and testing accuracy and loss comparison for a random fold (*i.e.* fold 10 on subject 16) in the Sleep-EDF dataset.

a stable performance during training, and we notice that it converges quickly. Additionally, it can be seen that the validation loss stabilizes even with the continual decrement in the training loss, which reflects the robustness of our model against overfitting.

We built our model using PyTorch 1.4 and trained it on a Tesla K40 GPU. We applied a batch size of 128, and the Adam optimizer with the learning rate starting with 1e-3 and then reducing to 1e-4 after 10 epochs. The weight decay of Adam was set to 1e-3, the betas ($b1$, $b2$) were used as (0.9, 0.999) respectively, the epsilon value was 1e-08 and the amsgrad was set to true. All the convolutional layers were initialized using a Gaussian distribution with a mean of 0 and a variance of 0.02. For the TCE module, we used 5 heads in MHA and the dimension of each feature d was 80 for the Sleep-EDF dataset, and 100 for the SHHS dataset because of its higher sampling rate, and hence its longer signal length. For the two fully connected layers, the input dimension was d and the output dimension was set to 120, and vice versa for the second fully connected layer. A detailed description can also be found in Table S.3 in our supplementary materials. Our source codes and supplementary materials are publicly available at https://github.com/emadeldeen24/AttnSleep.

3.2.4 Comparison with state-of-the-art approaches

We evaluated the performance of our AttnSleep model against various state-of-the-art approaches. We compared their performance in terms of overall accuracy, macro F1-score, Cohen kappa, macro G-mean, and the average training time on three datasets.

TABLE 3.5: Comparison among AttnSleep and state-of-the-art models. The best values on each dataset are highlighted in bold.

Dataset	Method	Per-Class F1-score					Overall Metrics				Avg Training time / fold
		W	N1	N2	N3	REM	Accuracy	MF1	κ	MGm	
Sleep-EDF-20	DeepSleepNet [11]	86.7	**45.5**	85.1	83.3	**82.6**	81.9	76.6	0.76	**86.9**	2.5 hrs
	SleepEEGNet [135]	89.4	44.4	84.7	84.6	79.6	81.5	76.6	0.75	85.3	1.5 hrs
	ResnetLSTM [160]	86.5	28.4	87.7	89.8	76.2	82.5	73.7	0.76	81.8	1.2 hrs
	MultitaskCNN [129]	87.9	33.5	87.5	85.8	80.3	83.1	75.0	0.77	83.1	2.6 hrs
	AttnSleep *(ours)*	**89.7**	42.6	**88.8**	**90.2**	79.0	**84.4**	**78.1**	**0.79**	85.5	**21 mins**
Sleep-EDF-78	DeepSleepNet [11]	90.9	**45.0**	79.2	72.7	71.1	77.8	71.8	0.70	81.6	7.2 hrs
	SleepEEGNet [135]	89.8	42.1	75.2	70.4	70.6	74.2	69.6	0.66	82.3	4.6 hrs
	ResnetLSTM [160]	90.7	34.7	83.6	80.9	67.0	78.9	71.4	0.71	80.8	3.4 hrs
	MultitaskCNN [129]	90.9	39.7	83.2	76.6	73.5	79.6	72.8	0.72	82.5	5.3 hrs
	AttnSleep *(ours)*	**92.0**	42.0	**85.0**	**82.1**	**74.2**	**81.3**	**75.1**	**0.74**	**83.6**	**1.7 hrs**
SHHS	DeepSleepNet [11]	85.4	**40.5**	82.5	79.3	81.9	81.0	73.9	0.73	82.6	14.4 hrs
	SleepEEGNet [135]	81.3	34.4	73.4	75.9	77.0	73.9	68.4	0.65	82.7	6.4 hrs
	ResnetLSTM [160]	85.1	9.4	86.3	87.0	79.1	83.3	69.4	0.76	76.4	5.2 hrs
	MultitaskCNN [129]	82.2	25.7	83.9	83.3	81.1	81.4	71.2	0.74	80.4	6.2 hrs
	AttnSleep *(ours)*	**86.7**	33.2	**87.1**	**87.1**	**82.1**	**84.2**	**75.3**	**0.78**	**84.0**	**2.1 hrs**

TABLE 3.6: Comparison of the performance of AttnSleep against SeqSleepNet with 3 epochs as input.

Dataset	Method	Per-Class F1-score					Overall Metrics				Avg Training time / fold
		W	N1	N2	N3	REM	Accuracy	MF1	κ	MGm	
Sleep-EDF-20	SeqSleepNet [154]	87.7	43.8	88.2	86.5	84.0	84.6	78.0	0.79	85.3	2.5 hrs
	AttnSleep *(ours)*	**90.3**	**47.9**	**89.8**	**89.0**	**85.0**	**85.6**	**80.9**	**0.80**	**88.2**	**31 mins**
Sleep-EDF-78	SeqSleepNet [154]	91.8	46.0	85.0	77.5	81.0	82.6	76.3	0.76	84.3	7.3 hrs
	AttnSleep *(ours)*	**92.6**	**47.4**	**85.5**	**83.7**	**81.5**	**82.9**	**78.1**	**0.77**	**85.6**	**1.9 hrs**
SHHS	SeqSleepNet [154]	84.2	**47.3**	87.2	85.4	**88.6**	85.6	78.5	0.80	85.4	15.2 hrs
	AttnSleep *(ours)*	**88.3**	46.3	**88.7**	**87.6**	87.4	**86.6**	**79.7**	**0.81**	**87.9**	**2.9 hrs**

Table 3.5 shows the comparison among DeepSleepNet [11], SleepEEGNet [135], ResnetLSTM [160], MultitaskCNN [129] and our AttnSleep. We observe that our AttnSleep achieves better classification performance than the other four approaches, due to its powerful feature extraction module as well as the TCE with the attention mechanism. In particular, our AttnSleep achieves better MF1 and MGm on Sleep-EDF-78 and SHHS, indicating that the designed cost-sensitive loss function is helpful in handling imbalanced data. In addition, we can observe that our AttnSleep achieves lower performance for class N1 than [11, 135]. As shown in Fig. 3.4, W, REM, and N1 have similar features in our framework. Therefore, our AttnSleep tends to misclassify N1 as other classes including W and REM, which is also demonstrated in the confusion matrices in Tables 3.2, 3.3 and 3.4.

Note that all the five methods in Table 3.5 use a single epoch (*i.e.*, 30-second EEG signal) as the model input. Differently, SeqSleepNet [154] takes 3 epochs as

input and then predicts the label for the middle epoch. For a fair comparison, we compare our AttnSleep with SeqSleepNet in Table 3.6 by using 3 epochs as input. As shown in Table 3.6, our AttnSleep outperforms SeqSleepNet in terms of all four metrics (ACC, MF1, κ, and MGm). By comparing Tables 3.5 and 3.6, we also observe that using more epochs as input includes more temporal relations and helps our AttnSleep model to achieve better performance.

In addition, the training time of our method is much less than other methods as shown in Tables 3.5 and 3.6. First, DeepSleepNet [11], SleepEEGNet [135] and SeqSleepNet [154] all exploit LSTMs which slow down the training due to the recurrent processing in LSTM. Second, MultitaskCNN [129] and SeqSleepNet [154] require additional computation to pre-train a DNN-based filter bank before training the main model. Differently, our AttnSleep model captures the temporal dependency among EEG data using TCE instead of LSTM, and can thus benefit from parallel computation to achieve reduced training complexity.

3.2.5 Ablation Study

Note that our AttnSleep consists of MRCNN, AFR, and TCE modules together with the class-aware loss function. To analyze the effectiveness of each module in our AttnSleep, we present an ablation study conducted on the Sleep-EDF dataset as shown in Fig. 3.6. Specifically, we derive five model variants as follows and the first four variants do not use the class-aware loss function.

1. **MRCNN**: MRCNN module only.
2. **MRCNN+AFR**: MRCNN and AFR without TCE.
3. **MRCNN+TCE**: MRCNN and TCE without AFR.
4. **MRCNN+AFR+TCE**: training MRCNN, AFR and TCE together, without the class-aware loss function.
5. **AttnSleep**: training MRCNN, AFR and TCE together, with the class-aware loss function.

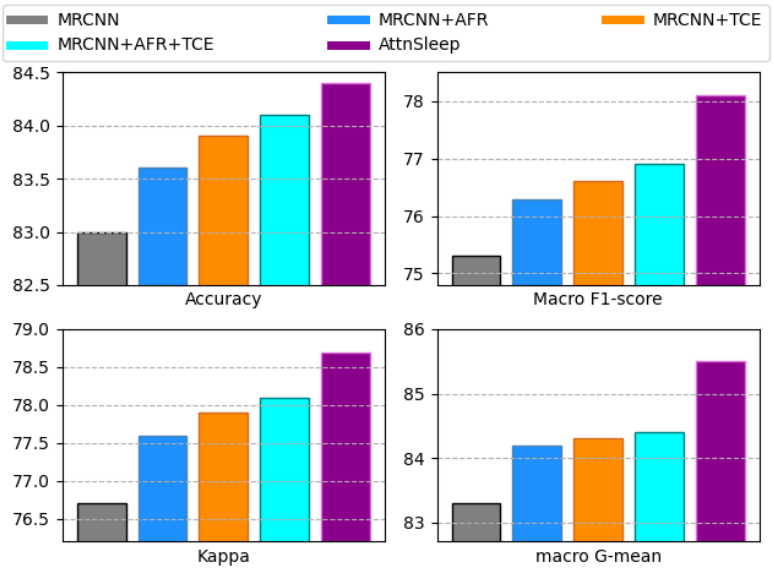


FIGURE 3.6: Ablation study conducted on Sleep-EDF dataset.

We can draw the following three conclusions based on the ablation study as shown in Fig. 3.6. First, AFR can enhance the classification performance, which demonstrates the necessity of modeling the feature inter-dependencies. This is further demonstrated by comparing the third and fourth variants (*i.e.*, MRCNN+TCE vs. MRCNN+AFR+TCE). Second, by comparing MRCNN and MRCNN+TCE (similarly MRCNN+AFR vs. MRCNN+AFR+TCE), we conclude that capturing the temporal dependencies with TCE is important for sleep stage classification. Moreover, TCE is even more important than AFR as MRCNN+TCE outperforms MRCNN+AFR. Third, AttnSleep achieves significantly better MF1 and MGm than the other four variants, indicating that the proposed class-aware cost-sensitive loss function can effectively address the data imbalance issue without any added computation overhead. Notably, the conclusions remain consistent when experimenting on Sleep-EDF-78 and SHHS datasets, as shown in Fig. 3.7.

3.2.6 Sensitivity Analysis for the Number of Heads in MHA

As MHA is one key component of our model, it is important to study how the number of heads affects the model performance. In particular, we fix the other parameters and test different numbers of heads in MHA. Note that the number of heads should be dividable by the length of features d. As d is 80 for the Sleep-EDF dataset, we run our model using 1, 2, 4, 5, 8, and 10 heads. Fig. 3.8 shows the

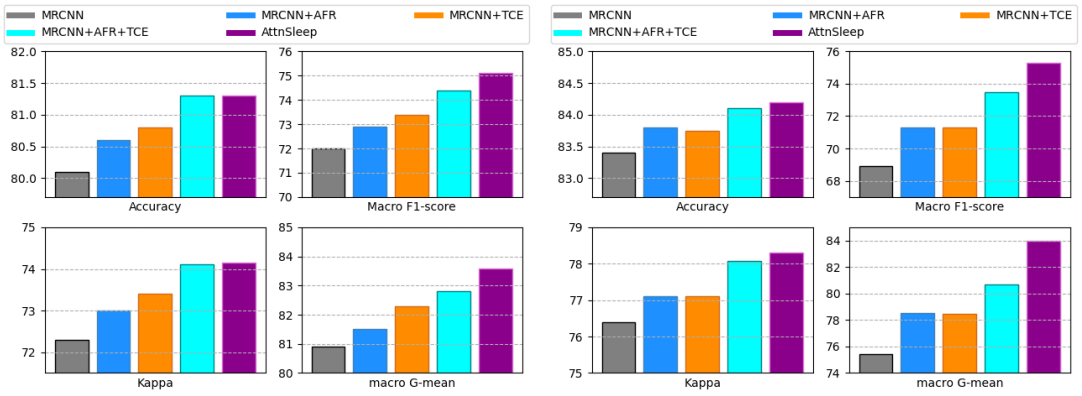


FIGURE 3.7: Ablation study conducted on Sleep-EDF-78 and SHHS datasets.

model performance on the Sleep-EDF dataset in terms of accuracy and MF1 score. Overall, the model performance is quite stable when we use different numbers of heads. With increasing the number of heads from 1, 2 to 4, and 5, we can observe a slight improvement to the performance, since using more heads allows the model to find more meaningful features and feature interactions. Meanwhile, when the number of heads further increases (H equals 8 and 10), *i.e.*, the length of features in each head becomes smaller, which leads to a slight performance decrease. In our experiments, we eventually set H as 5 on the Sleep-EDF dataset. Similarly, we also set H as 5, based on the sensitivity analysis applied to the other two datasets as shown in Fig. 3.9.

3.3 AttnSleep in Another Scenarios

Despite the acclaimed performance achieved by AttnSleep, as well as other deep learning-based models, it still suffers some limitations, particularly in scenarios where labeled data is scarce and when tested on out-of-distribution test sets. therefore, we provide experimental proof for these limitations and discuss potential solutions.

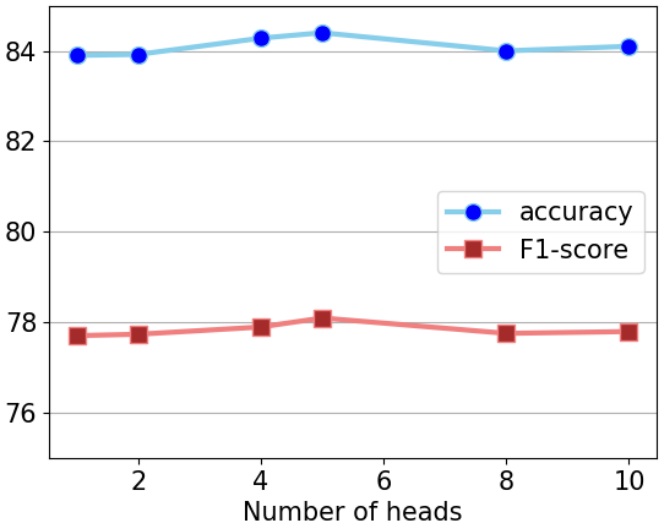


FIGURE 3.8: The performance of our AttnSleep model on Sleep-EDF dataset by using a different number of heads in MHA.

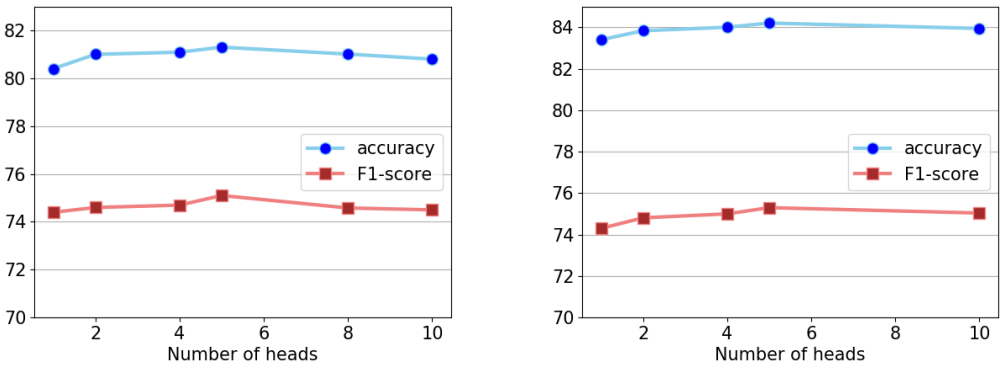


FIGURE 3.9: The performance of our AttnSleep model on Sleep-EDF-78 and SHHS datasets by using a different number of heads in MHA.

3.3.1 Label-Scarce Scenarios

In many real-world applications, obtaining labeled data is often challenging and expensive. For example, sleep labs can collect a massive amount of overnight EEG readings, but they require expert knowledge for annotation, which is time-consuming. Deep learning models trained on limited labeled data can suffer from overfitting, leading to poor generalization to unseen data. This problem is particularly acute in the case of deep neural networks with a large number of parameters, which are prone to overfitting.

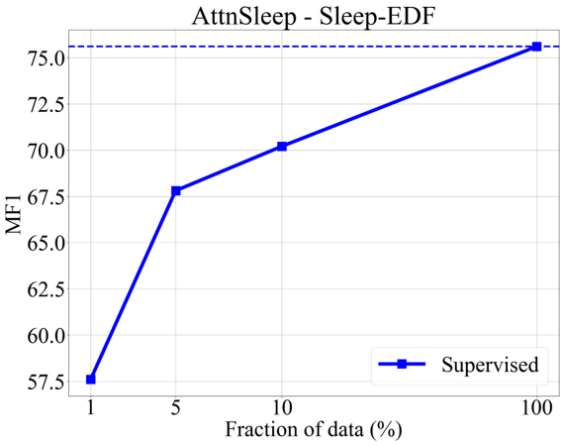


FIGURE 3.10: Performance of AttnSleep under different few labels scenarios applied to Sleep-EDF dataset in terms of MF1-score. Results are obtained without applying the class-aware loss.

To validate this problem, we test the performance of AttnSleep on different fractions of labeled samples from Sleep-EDF data. The results are shown in Fig. 3.10.

It is worth noting that the performance of AttnSleep significantly deteriorates when using only 1% of labeled samples, achieving only 57.5%, a result that is considerably inadequate and unreliable. Although the performance tends to improve as the percentage of available labeled samples increases, this may not be a feasible solution in practical settings. Consequently, our objective in this thesis is to enhance the performance of deep learning models, e.g., AttnSleep, in diverse label-scarce scenarios.

3.3.2 Domain Shift Scenarios

Another challenge in deep learning is when the data distribution changes, also known as domain shift. This can occur due to various reasons such as changes in the environment, measuring devices, or the subject's health status. In such cases, deep learning models trained on one domain may perform poorly on data from a different domain, leading to a decrease in the overall performance of the system.

We evaluate the efficacy of AttnSleep in the context of domain shift by training it on one source dataset and subsequently testing it on other out-of-distribution datasets. Specifically, we experiment on three datasets namely Sleep-EDF, SHHS-1 (S1), and SHHS-2 (S2) (see Section 6.2.1 for a detailed description of these

TABLE 3.7: Comparison between Direct Transfer and the oracle Target Only performance on AttnSleep on cross-dataset scenarios.

	Cross-Domain Accuracy						AVG
Baselines	EDF→S1	EDF→S2	S1→EDF	S1→S2	S2→EDF	S2→S1	ACC
Direct Transfer	64.44±3.37	57.43±5.45	75.41±0.65	72.08±0.23	66.59±0.83	77.52±0.29	68.91
Target Only	85.23±0.55	78.44±2.71	76.59±0.32	78.44±2.71	76.59±0.32	85.23±0.55	80.09

datasets). These datasets deviate from each other in several aspects, including the sampling rate and the health and age status of the subjects. The results of our experiments are presented in Table 3.7.

The first row in Table 3.7 presents the results of the direct transfer scenario, which entails training AttnSleep on a source dataset and directly testing it on the target dataset. The second row displays the performance of training the model on the training set of the target dataset and subsequently testing it on the testing set of the same dataset.

It is worth noting that the performance of AttnSleep suffers a considerable deterioration due to the domain shift. Specifically, when transferring from Sleep-EDF to S1 and S2, the performance declines by 20.79% and 21.01%, respectively. Likewise, when transferring from S2 to Sleep-EDF and S1, the performance declines by 10% and 7.71%, respectively. These outcomes underscore the necessity of devising a methodology to mitigate the impact of domain shift in deep learning methods.

3.4 Summary

We proposed a novel architecture for sleep stage classification from single-channel raw EEG signals called AttnSleep. The AttnSleep relies on extracting the features from EEG signals using two modules: the multi-resolution convolutional neural network (MRCNN) and the adaptive feature recalibration (AFR). These two modules are followed by the temporal context encoder (TCE) module, which captures the temporal dependencies among the extracted features by using a multi-head attention (MHA) mechanism. We also proposed a class-aware cost-sensitive loss function to handle the issue of data imbalance. The experimental results on three public datasets demonstrated that our model outperforms state-of-the-art methods under various evaluation matrices. Besides, an ablation study was performed to

show the effectiveness of each module in the proposed method. Finally, we conducted a sensitivity analysis to demonstrate the impact of the number of heads in MHA. The results indicated that our method is quite stable with a different number of heads.

Chapter 4

Time Series Representation Learning via Temporal and Contextual Contrasting

Learning decent representations from **unlabeled** time series data with temporal dynamics is a very challenging task. In this chapter [1], we propose an unsupervised **T**ime **S**eries representation learning framework via **T**emporal and **C**ontextual **C**ontrasting (**TS-TCC**), to learn time series representation from unlabeled data. First, the raw time series data are transformed into two different yet correlated views by using weak and strong augmentations. Second, we propose a novel temporal contrasting module to learn *robust* temporal representations by designing a tough cross-view prediction task. Last, to further learn *discriminative* representations, we propose a contextual contrasting module built upon the contexts from the temporal contrasting module. It attempts to maximize the similarity among different contexts of the same sample while minimizing similarity among contexts of different samples. Experiments have been carried out on three real-world time series datasets. The results manifest that training a linear classifier on top of the features learned by our proposed TS-TCC performs comparably with the supervised training. Additionally, our proposed TS-TCC shows high efficiency in

[1]The work in this chapter has been published as "Time-Series Representation Learning via Temporal and Contextual Contrasting", International Joint Conferences on Artificial Intelligence, 2021 [161]. Also, another part has been published as "Self-supervised Learning for Label-Efficient Sleep Stage Classification: A Comprehensive Evaluation", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2023 [145].

few-labeled data and transfer learning scenarios. The code is publicly available at `https://github.com/emadeldeen24/TS-TCC`.

In summary, our main contributions in this chapter are as follows.

- A novel contrastive learning framework is proposed for unsupervised time series representation learning.
- Simple yet efficient augmentations are designed for time series data in the contrastive learning framework.
- We propose a novel temporal contrasting module to learn robust representations from time series data by designing a tough cross-view prediction task. In addition, we propose a contextual contrasting module to further learn discriminative representations upon the robust representations.
- We perform extensive experiments on our proposed TS-TCC framework using three datasets. Experimental results show that the learned representations are effective for downstream tasks under supervised learning, semi-supervised learning, and transfer learning settings.

4.1 Methods

This section describes our proposed TS-TCC in detail. As shown in Figure 4.1, we first generate two different yet correlated views of the input data based on strong and weak augmentations. Then, a temporal contrasting module is proposed to explore the temporal features of the data with an autoregressive model. These models perform a tough cross-view prediction task by predicting the future of one view using the past of the other. We further maximize the agreement between the contexts of the autoregressive models by a contextual contrasting module. Next, we will introduce each component in the following subsections.

4.1.1 Time Series Data Augmentation

Data augmentation is a key part of the success of contrastive learning methods [41, 44]. Contrastive methods try to maximize the similarity among different views

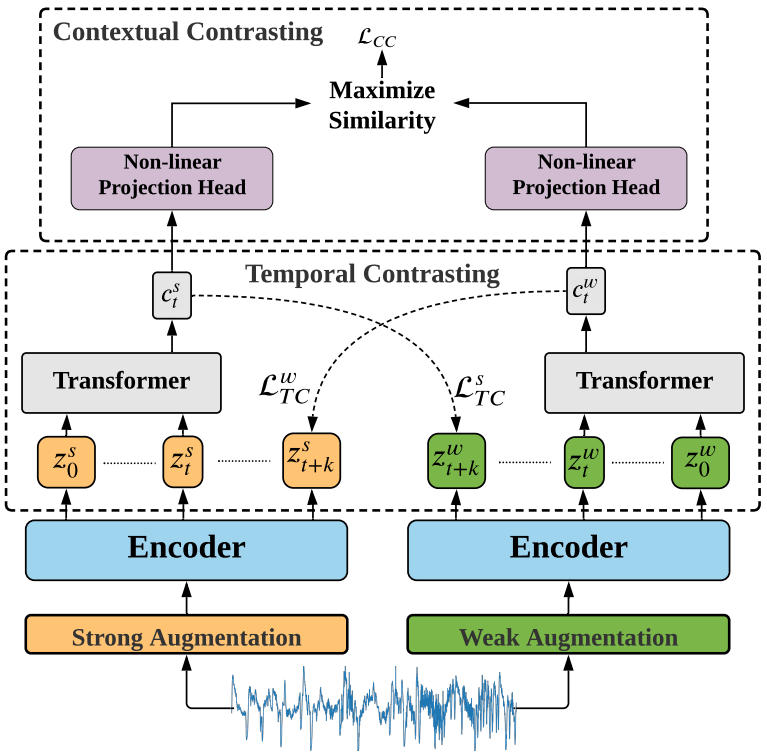


FIGURE 4.1: The overall architecture of the proposed TS-TCC. The Temporal Contrasting module learns robust temporal features through a tough cross-view prediction task. The Contextual Contrasting module learns discriminative features by maximizing the similarity between the contexts of the same sample while minimizing its similarity with the other samples within the mini-batch.

of the same sample while minimizing its similarity with other samples. It is thus important to design proper data augmentations for contrastive learning [41, 162]. Usually, contrastive learning methods use two (random) variants of the same augmentation. Given a sample x, they produce two views x_1 and x_2 sampled from the same augmentations family $\mathcal{T}$, i.e., $x_1 \sim \mathcal{T}$ and $x_2 \sim \mathcal{T}$. However, we argue that using different augmentations can improve the robustness of the learned representations. Consequently, we propose applying two separate augmentations, such that one augmentation is weak and the other is strong. In this paper, weak augmentation is a jitter-and-scale strategy. Specifically, we add random variations to the signal and scale up its magnitude. For strong augmentation, we apply the permutation-and-jitter strategy, where permutation includes splitting the signal into a random number of segments with a maximum of M and randomly shuffling them. Next, a random jittering is added to the permuted signal. Notably, the augmentation hyperparameters should be chosen carefully according to the nature of the time series data. For example, the value of M in a time series data with

longer sequences should be greater than its value in those with shorter sequences when applying permutation. Similarly, the jittering ratio for normalized time series data should be much less than the ratio for unnormalized data. *For more details about augmentations selection, see Section 4.3.6.*

For each input sample x, we denote its strongly augmented view as x^s, and its weakly augmented view as x^w, where $x^s \sim \mathcal{T}_s$ and $x^w \sim \mathcal{T}_w$. These views are then passed to the encoder to extract their high-dimensional latent representations. For an input $\mathbf{x}$, the encoder maps $\mathbf{x}$ into a high-dimensional latent representation $\mathbf{z} = F(\mathbf{x})$, where F represents the encoder. We define $\mathbf{z} = [z_1, z_2, \dots z_N]$, where N is the total channels, $z_i \in \mathbb{R}^d$, where d is the feature length. Thus, we get $\mathbf{z}^s$ for the strong augmented views, and $\mathbf{z}^w$ for the weak augmented views, which are then fed into the temporal contrasting module.

4.1.2 Temporal Contrasting

The Temporal Contrasting module deploys a contrastive loss to extract temporal features in the latent space with an autoregressive model. Given the latent representations $\mathbf{z}$, the autoregressive model F_{ar} summarizes all $\mathbf{z}_{\leq t}$ into a context vector $c_t = F_{ar}(\mathbf{z}_{\leq t}),\ c_t \in \mathbb{R}^h$, where h is the hidden dimension of F_{ar}. The context vector c_t is then used to predict the timesteps from z_{t+1} until z_{t+k} $(1 < k \leq K)$. To predict future timesteps, we use a log-bilinear model that would preserve the mutual information between the input x_{t+k} and c_t, such that $f_k(x_{t+k}, c_t) = exp((\mathcal{W}_k(c_t))^T z_{t+k})$, where $\mathcal{W}_k$ is a linear function that maps c_t back into the same dimension as z, i.e. $\mathcal{W}_k : \mathbb{R}^{h \to d}$.

In our approach, the strong augmentation generates c_t^s and the weak augmentation generates c_t^w. We propose a tough cross-view prediction task by using the context of the strong augmentation c_t^s to predict the future timesteps of the weak augmentation z_{t+k}^w and vice versa. The contrastive loss tries to minimize the dot product between the predicted representation and the true one of the same sample while maximizing the dot product with the other samples $\mathcal{N}_{t,k}$ within the mini-batch. Accordingly, we calculate the two losses $\mathcal{L}_{TC}^s$ and $\mathcal{L}_{TC}^w$ as follows:

$$\mathcal{L}_{TC}^s = -\frac{1}{K}\sum_{k=1}^{K} \log \frac{\exp((\mathcal{W}_k(c_t^s))^T z_{t+k}^w)}{\sum_{n \in \mathcal{N}_{t,k}} \exp((\mathcal{W}_k(c_t^s))^T z_n^w)} \tag{4.1}$$

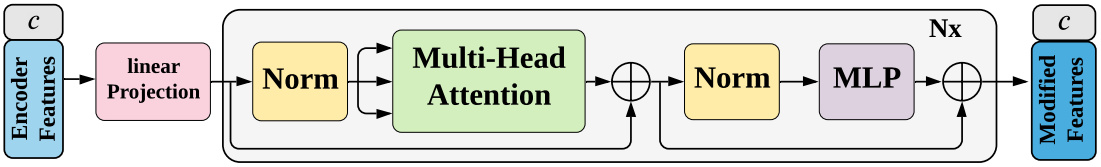


FIGURE 4.2: Architecture of the Transformer model used in the Temporal Contrasting module. We first attach the classification token c to the input and pass it through the different Transformer model layers. Eventually, we split the token from the output modified features and use it in the next Contextual Contrasting module.

$$\mathcal{L}_{TC}^{w} = -\frac{1}{K}\sum_{k=1}^{K}\log\frac{\exp((\mathcal{W}_k(c_t^w))^T z_{t+k}^s)}{\sum_{n\in\mathcal{N}_{t,k}}\exp((\mathcal{W}_k(c_t^w))^T z_n^s)} \tag{4.2}$$

We use Transformer as the autoregressive model because of its efficiency and speed [37]. The architecture of the Transformer model is shown in Figure 4.2. It mainly consists of successive blocks of multi-headed attention (MHA) followed by an MLP block. The MLP block is composed of two fully connected layers with a non-linearity ReLU function and dropout in between. Pre-norm residual connections, which can produce more stable gradients [163], are adopted in our Transformer. We stack L identical layers to generate the final features. Inspired by the BERT model [151], we add a token $c \in \mathbb{R}^h$ to the input whose state acts as a representative context vector in the output. The operation of the Transformer starts by applying the features $\mathbf{z}_{\leq t}$ to a linear projection $\mathcal{W}_{Tran}$ layer that maps the features into the hidden dimension, i.e. $\mathcal{W}_{Tran} : \mathbb{R}^{d\rightarrow h}$. The output of this linear projection is then sent to the Transformer i.e. $\tilde{z} = \mathcal{W}_{Tran}(\mathbf{z}_{\leq t}),\ \ \tilde{\mathbf{z}} \in \mathbb{R}^h$. Next, we attach the context vector into the features vector $\tilde{\mathbf{z}}$ such that the input features become $\psi_0 = [c; \tilde{\mathbf{z}}]$, where the subscript 0 denotes being the input to the first layer. Next, we pass ψ_0 through Transformer layers as in the following equations:

$$\tilde{\psi}_\ell = \text{MHA}(\text{Norm}(\psi_{\ell-1})) + \psi_{\ell-1}, \qquad 1 \leq \ell \leq L; \tag{4.3}$$

$$\psi_\ell = \text{MLP}(\text{Norm}(\tilde{\psi}_\ell)) + \tilde{\psi}_\ell, \qquad 1 \leq \ell \leq L. \tag{4.4}$$

Finally, we re-attach the context vector from the final output such that $c_t = \psi_L^0$. This context vector will be the input of the following contextual contrasting module.

4.1.3 Contextual Contrasting

We further propose a contextual contrasting module that aims to learn more discriminative representations. Time series data often encounters patterns, trends, and dependencies that a given data point has with its preceding and subsequent points. For instance, the presence of specific waveform patterns like sleep spindles or K-complexes in the EEG signal can indicate Stage 2 non-REM sleep. However, the mere presence of a spindle or K-complex isn't sufficient; its context, meaning its temporal relation to other EEG features, and its density over a certain time frame, provides a clearer indication of the sleep stage. Therefore, we developed the Contextual Contrasting module to identify these patterns in time series data. It starts with applying a non-linear transformation to the contexts using a non-linear projection head as in [41]. The projection head maps the contexts into the space where the contextual contrasting is applied.

Given a batch of B input samples, we will have two contexts for each sample from its two augmented views and thus have $2B$ contexts. For a context c_t^i, we denote $c_t^{i^+}$ as the positive sample of c_t^i that comes from the other augmented view of the same input, and hence, $(c_t^i, c_t^{i^+})$ are considered to be a positive pair. Meanwhile, the remaining $(2B-2)$ contexts from other inputs within the same batch are considered as the negative samples of c_t^i, i.e., c_t^i can form $(2B-2)$ negative pairs with its negative samples. Therefore, we can derive a contextual contrasting loss to maximize the similarity between the positive pair and minimize the similarity between negative pairs. As such, the final representations can be discriminative.

Eq. 4.5 defines the contextual contrasting loss function $\mathcal{L}_{CC}$. Given a context c_t^i, we divide its similarity with its positive sample $c_t^{i^+}$ by its similarity with all the other $(2B-1)$ samples, including the positive pair and $(2B-2)$ negative pairs, to normalize the loss.

$$\mathcal{L}_{CC} = -\sum_{i=1}^{B} \log \frac{\exp\left(\text{sim}\left(\boldsymbol{c}_t^i, \boldsymbol{c}_t^{i^+}\right)/\tau\right)}{\sum_{m=1}^{2B} \mathbb{1}_{[m\neq i]} \exp\left(\text{sim}\left(\boldsymbol{c}_t^i, \boldsymbol{c}_t^m\right)/\tau\right)}, \tag{4.5}$$

where $sim(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T\boldsymbol{v}/\|\boldsymbol{u}\|\|\boldsymbol{v}\|$ denotes the dot product between ℓ_2 normalized $\boldsymbol{u}$ and $\boldsymbol{v}$ (i.e., cosine similarity), $\mathbb{1}_{[m\neq i]} \in \{0, 1\}$ is an indicator function, evaluating to 1 iff $m \neq i$, and τ is a temperature parameter.

The overall self-supervised loss is the combination of the two temporal contrasting losses and the contextual contrasting loss as follows.

$$\mathcal{L} = \lambda_1 \cdot (\mathcal{L}_{TC}^{s} + \mathcal{L}_{TC}^{w}) + \lambda_2 \cdot \mathcal{L}_{CC}, \tag{4.6}$$

where λ_1 and λ_2 are fixed scalar hyperparameters denoting the relative weight of each loss.

4.2 Experimental Setup

4.2.1 Datasets

To evaluate our model, we adopted three publicly available datasets [2] for human activity recognition, sleep stage classification, and epileptic seizure prediction, respectively. Additionally, we investigated the transferability of our learned features on a fault diagnosis dataset.

4.2.1.1 Human Activity Recognition (HAR)

We use the UCI HAR dataset [164] which contains sensor readings for 30 subjects performing 6 activities (i.e. walking, walking upstairs, downstairs, standing, sitting, and lying down). They collected the data using a mounted Samsung Galaxy S2 device on their waist, with a sampling rate of 50 Hz.

4.2.1.2 Sleep Stage Classification

We also adapted the Sleep Stage Classification problem in our experiments and used the Sleep-EDF dataset which was discussed earlier in Section 3.2.1 of Chapter 3.

[2]The preprocessed versions of the datasets are available on NTU Dataverse at https://researchdata.ntu.edu.sg/dataverse/tstcc/

4.2.1.3 Epilepsy Seizure Prediction

The Epileptic Seizure Recognition dataset [165] consists of EEG recordings from 500 subjects, where the brain activity was recorded for each subject for 23.6 seconds. Note that the original dataset is labeled with five classes. As four of them do not include epileptic seizures, we merged them into one class and treated it as a binary classification problem.

TABLE 4.1: Description of datasets used in our experiments. The details of FD is the same for all the 4 working conditions.

Dataset	# Train	# Test	Length	# Channel	# Class
HAR	7352	2947	128	9	6
Sleep-EDF	25612	8910	3000	1	5
Epilepsy	9200	2300	178	1	2
FD	8184	2728	5120	1	3

4.2.1.4 Fault Diagnosis (FD)

We conducted the transferability experiment on a real-world fault diagnosis dataset [166]. This dataset was collected under four different working conditions. Each working condition can be considered as a separate domain as it has different characteristics from the other working conditions [167]. Each domain has three classes, namely, two fault classes (i.e., inner fault and outer fault) and one healthy class.

Table 4.1 summarizes the details of each dataset, e.g., the number of training samples (# Train) and testing samples (# Test), the length of the sample, the number of sensor channels (# Channel) and the number of classes (# Class).

4.2.2 Implementation Details

We split the data into 60%, 20%, and 20% for training, validation, and testing considering subject-wise split for the Sleep-EDF dataset to avoid overfitting. Experiments were repeated for 5 times with 5 different seeds, and we reported the mean and standard deviation. The pretraining and downstream tasks were done for 40 epochs, as we noticed that the performance did not improve with further training. We applied a batch size of 128 (which was reduced to 32 in *few-labeled*

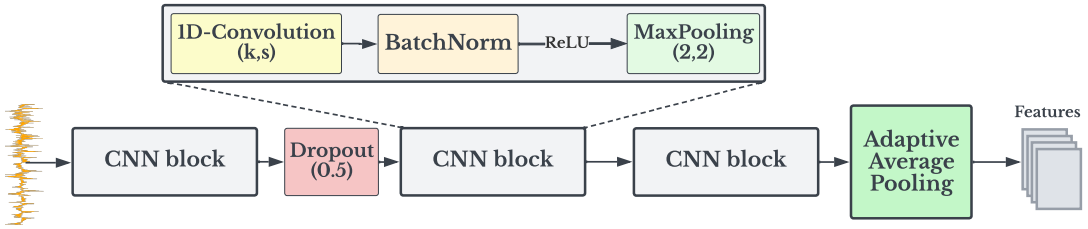


FIGURE 4.3: Feature encoder in our framework. The pair (k, s) refers to the kernel size and stride respectively.

data experiments as data size may be less than 128). We used Adam optimizer with a learning rate of 3e-4, weight decay of 3e-4, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. For the strong augmentation, we set $M_{HAR} = 10$, $M_{Ep} = 12$ and $M_{EDF} = 20$, while for the weak augmentation, we set the scaling ratio to 2 for all the datasets. We set $\lambda_1 = 1$, while we achieved good performance when $\lambda_2 \approx 1$. Particularly, we set it as 0.7 in our experiments on the four datasets. In the Transformer, we set the $L = 4$, and the number of heads as 4. We tuned $h \in \{32, 50, 64, 100, 128, 200, 256\}$ and set $h_{HAR,Ep} = 100$, $h_{EDF} = 64$. We also set its dropout to 0.1. In contextual contrasting, we set $\tau = 0.2$. Lastly, we built our model using PyTorch 1.7 and trained it on NVIDIA GeForce RTX 2080 Ti GPU.

4.2.2.1 Feature encoder

We adopted a convolutional neural network as our feature encoder [24]. As shown in Fig. 4.3, our feature encoder consists of three similar convolutional blocks. The first one is followed by a dropout layer, while the last one is followed by an adaptive average pooling layer. Each block contains a 1D-convolutional layer, a batch normalization layer, a non-linearity ReLU activation function, and a 1D-Maxpooling layer. The filter sizes of the 1D-convolution layers are set to 64, 128, and 128 in the three blocks respectively. The kernel size and stride values differ from one dataset to the other. We set this pair as (5,1) for HAR and Epilepsy datasets as they have the same sequence length. For the Sleep-EDF dataset, we set it as (25,6) due to its longer sequence length. This encoder is followed by a single fully connected layer for classification.

TABLE 4.2: Comparison between our proposed TS-TCC model against baselines using linear classifier evaluation experiment.

	HAR		Sleep-EDF		Epilepsy	
Baseline	ACC	MF1	ACC	MF1	ACC	MF1
Random Initialization	57.89±5.13	55.45±5.49	35.61±6.96	23.80±7.96	90.26±1.77	81.12±4.22
Supervised	90.14±2.49	90.31±2.24	**83.41±1.44**	**74.78±0.86**	96.66±0.24	94.52±0.43
SSL-ECG [43]	65.34±1.63	63.75±1.37	74.58±0.60	65.44±0.97	93.72±0.45	89.15±0.93
CPC [53]	83.85±1.51	83.27±1.66	82.82±1.68	73.94±1.75	96.61±0.43	94.44±0.69
SimCLR [41]	80.97±2.46	80.19±2.64	78.91±3.11	68.60±2.71	96.05±0.34	93.53±0.63
TS-TCC *(ours)*	**90.37±0.34**	**90.38±0.39**	83.00±0.71	73.57±0.74	**97.23±0.10**	**95.54±0.08**

4.3 Results

To show the efficacy of our proposed TS-TCC, we test it in three different training settings, including linear evaluation, semi-supervised training, and transfer learning.

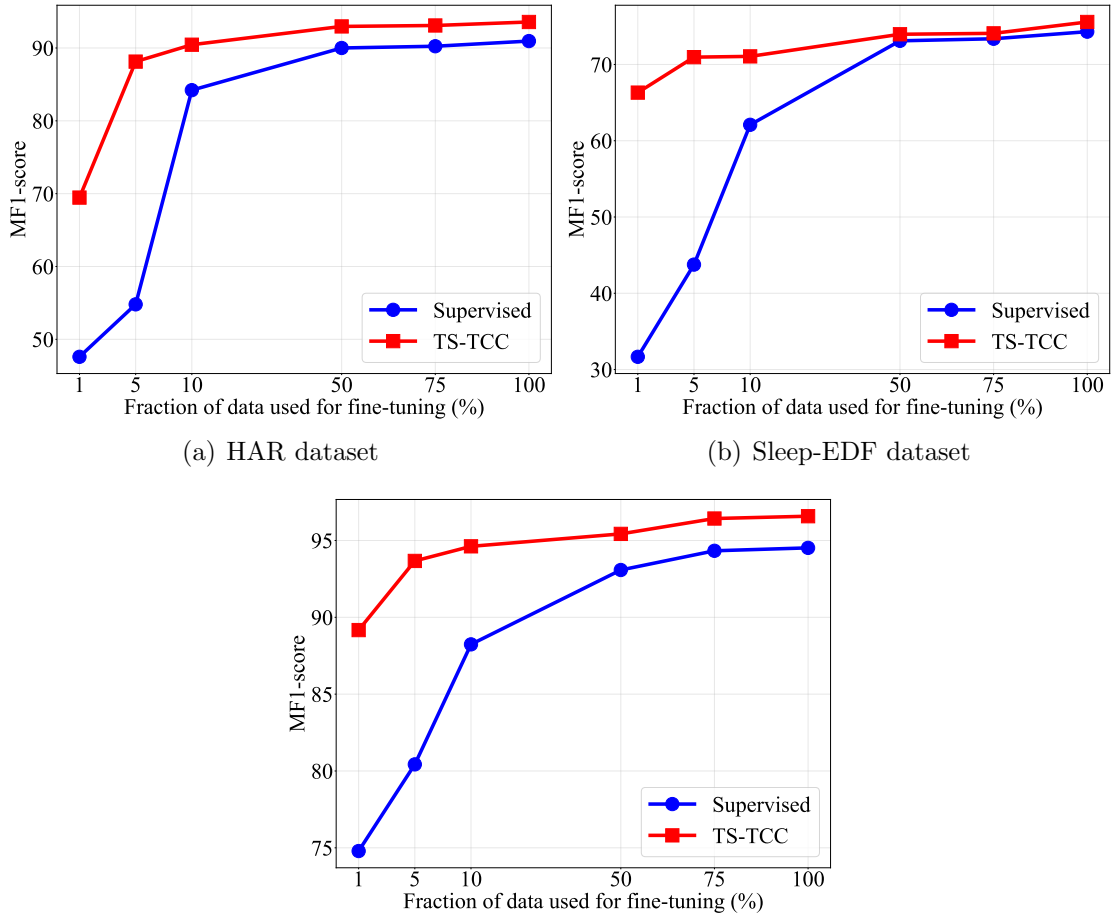
We measure the performance using two metrics namely the accuracy (Eq. 3.13 and the macro-averaged F1-score (MF1) (Eq. 3.14) to better verify performance on the imbalanced datasets. Although the results of the equations of these two metrics are expected to be in a range from 0 to 1, we report them in this paper as a percentage.

4.3.1 Comparison with Baseline Approaches

We compare our proposed approach against the following baselines. (1) **Random Initialization**: training a linear classifier on top of randomly initialized encoder; (2) **Supervised**: supervised training of both encoder and classifier model; (3) **SSL-ECG** [43]; (4) **CPC** [53]; (5) **SimCLR** [41]. It is worth noting that, we use time series-specific augmentations to adapt SimCLR to our application as it was originally designed for images.

To evaluate the performance of our TS-TCC model, we follow the standard linear benchmarking evaluation scheme [41, 53]. Particularly, we train a linear classifier (single MLP layer) on top of a frozen self-supervised pretrained encoder model. Table 4.2 shows the linear evaluation results of our approach against the baseline methods. Overall, our proposed TS-TCC outperforms all three state-of-the-art methods. Furthermore, TS-TCC, with only a linear classifier, performs best on

two out of three datasets while achieving comparable performance to the supervised approach on the third dataset. This demonstrates the powerful representation learning capability of our TS-TCC model. Notably, contrastive methods (e.g., CPC, SimCLR, and our TS-TCC) generally achieve better results than the pretext-based method (i.e., SSL-ECG), which reflects the power of invariant features learned by contrastive methods. Additionally, the CPC method shows better results than SimCLR, indicating that temporal features are more important than general features in time series data.



(a) HAR dataset

(b) Sleep-EDF dataset

(c) Epilepsy dataset

FIGURE 4.4: Comparison between supervised training vs. TS-TCC *fine-tuning* on three datasets with different fractions of few labeled data in terms of MF1-score.

4.3.2 Semi-supervised Training

We investigate the effectiveness of our TS-TCC under the semi-supervised settings, by training the model with 1%, 5%, 10%, 50%, and 75% of randomly selected instances of the training data. Figure 4.4 shows the results of our TS-TCC along with the supervised training under the aforementioned settings. In particular, TS-TCC fine-tuning (i.e., red curves in Figure 4.4) means that we fine-tuned the pretrained encoder with few labeled samples.

We observe that supervised training performs poorly with limited labeled data, while our TS-TCC fine-tuning achieves significantly better performance than supervised training with only 1% of labeled data. For example, TS-TCC fine-tuning can still achieve around 70% and 90% for HAR and Epilepsy datasets respectively. Furthermore, our TS-TCC fine-tuning with only 10% of labeled data can achieve comparable performance with the supervised training with 100% of labeled data in the three datasets, demonstrating the effectiveness of our TS-TCC method under the semi-supervised setting.

TABLE 4.3: Cross-domains transfer learning experiment applied on Fault Diagnosis dataset in terms of accuracy. (FT stands for fine-tuning)

	A→B	A→C	A→D	B→A	B→C	B→D	C→A	C→B	C→D	D→A	D→B	D→C	AVG
Supervised	34.38	44.94	34.57	**52.93**	63.67	**99.82**	**52.93**	84.02	83.54	**53.15**	99.56	62.43	63.83
TS-TCC *(FT)*	**43.15**	**51.50**	**42.74**	47.98	**70.38**	99.30	38.89	**98.31**	**99.38**	51.91	**99.96**	**70.31**	**67.82**

4.3.3 Transfer Learning Experiment

We further examine the transferability of the learned features by designing a transfer learning experiment. We use the Fault Diagnosis (FD) dataset introduced in Table 4.1 for the evaluation under the transfer learning setting. Here, we train the model on one condition (i.e., source domain) and test it on another condition (i.e., target domain). In particular, we adopt two training schemes on the source domain, namely, (1) supervised training and (2) TS-TCC fine-tuning where we fine-tuned our pretrained encoder using the labeled data in the source domain.

Table 4.3 shows the performance of the two training schemes under 12 cross-domain scenarios. Clearly, our pretrained TS-TCC model with fine-tuning (FT) consistently outperforms the supervised pretraining in 8 out of 12 cross-domain scenarios. TS-TCC model can achieve at least 7% improvement in 7 out of 8 winning

scenarios (except for D→B scenario). Overall, our proposed approach can improve the transferability of learned representations over the supervised training by about 4% in terms of accuracy.

TABLE 4.4: Ablation study of each component in TS-TCC model performed with linear classifier evaluation experiment.

	HAR		Sleep-EDF		Epilepsy	
Component	ACC	MF1	ACC	MF1	ACC	MF1
TC only	82.76±1.50	82.17±1.64	80.55±0.39	70.99±0.86	94.39±1.19	90.93±1.41
TC + X-Aug	87.86±1.33	87.91±1.09	81.58±1.70	71.88±1.71	95.56±0.24	92.57±0.29
TS-TCC (TC + X-Aug + CC)	**90.37±0.34**	**90.38±0.39**	**83.00±0.71**	**73.57±0.74**	**97.23±0.10**	**95.54±0.08**
TS-TCC (Weak only)	76.55±3.59	75.14±4.66	80.90±1.87	72.51±1.74	97.18±0.17	95.47±0.31
TS-TCC (Strong only)	60.23±3.31	56.15±4.14	78.55±2.94	68.05±1.87	97.14±0.23	95.39±0.29

4.3.4 Ablation Study

We study the effectiveness of each component in our proposed TS-TCC model. Specifically, we derive different model variants for comparison as follows. First, we train the Temporal Contrasting module (TC) without the cross-view prediction task, where each branch predicts the future timesteps of the same augmented view. This variant is denoted as 'TC only'. Second, we train the TC module by adding the cross-view prediction task, which is denoted as 'TC + X-Aug'. Third, we train the whole proposed TS-TCC model, which is denoted as 'TC + X-Aug + CC'. We also study the effect of using a single augmentation in TS-TCC. In particular, for an input x, we generate two different views x_1 and x_2 from the same augmentation type, i.e., $x_1 \sim \mathcal{T}_w$ and $x_2 \sim \mathcal{T}_w$ when using the weak augmentation.

Table 4.4 shows this ablation study on the three datasets. Clearly, the proposed cross-view prediction task generates robust features and thus improves the performance by more than 5% on HAR datasets, and ∼1% on Sleep-EDF and Epilepsy datasets. Additionally, the contextual contrasting module further improves the performance, as it helps the features to be more discriminative. Studying the effect of the augmentation, we find that generating different views from the same augmentation type is not helpful with HAR and Sleep-EDF datasets. On the other hand, the Epilepsy dataset can achieve comparable performance with only one augmentation. Overall, our proposed TS-TCC method using both types of augmentations achieves the best performance.

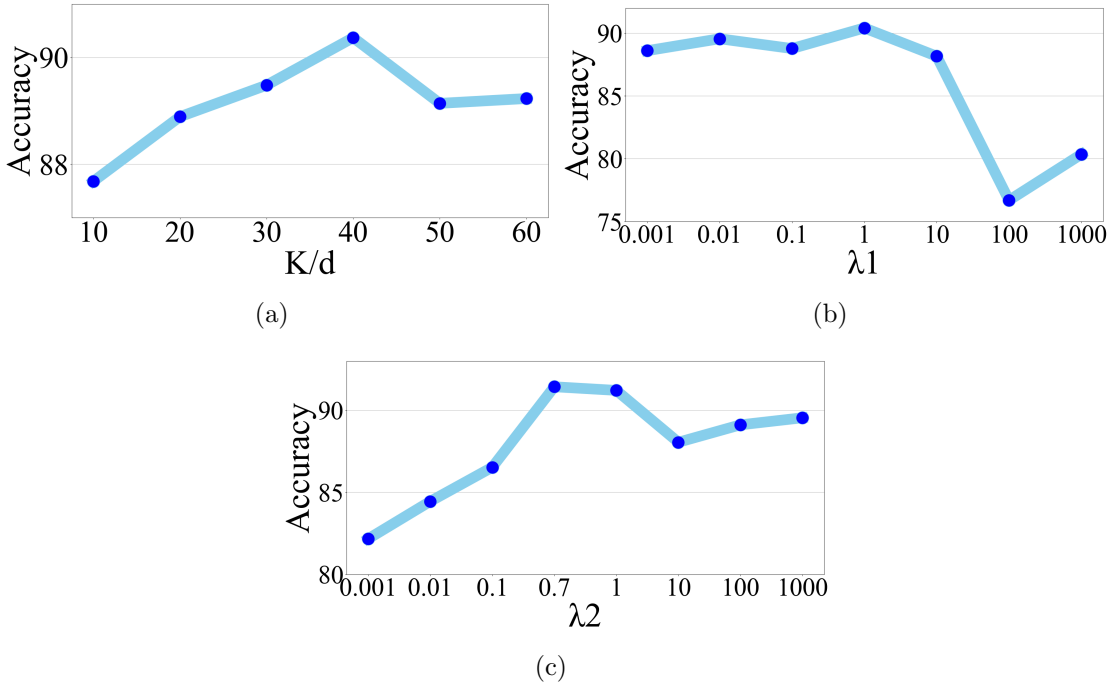
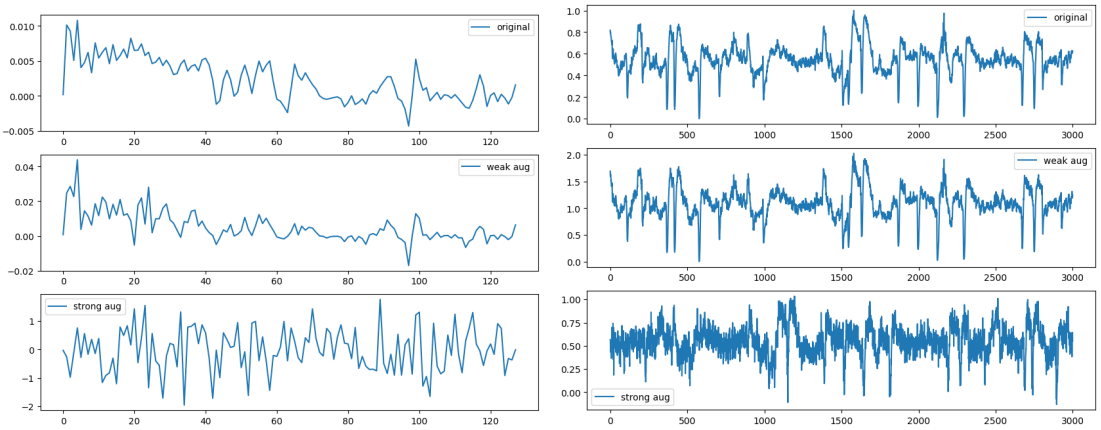


FIGURE 4.5: Three sensitivity analysis experiments on HAR dataset.

4.3.5 Sensitivity Analysis

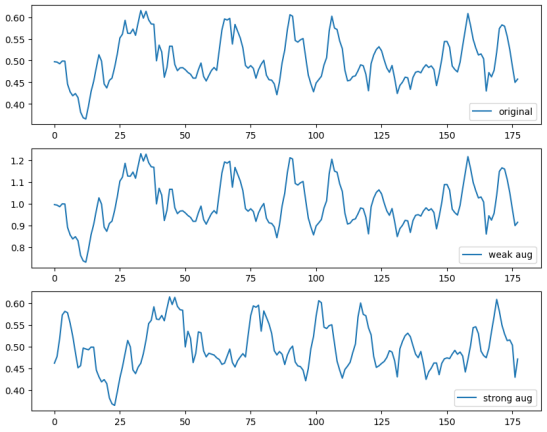
We perform sensitivity analysis on the HAR dataset to study three parameters namely, the number of predicted future timesteps K in the temporal contrasting module, besides λ_1 and λ_2 in Eq. 4.5.

Figure 4.5(a) shows the effect of K on the overall performance, where the x-axis is the percentage K/d, and d is the length of the features. Clearly, increasing the percentage of the predicted future timesteps improves the performance. However, larger percentages can harm the performance as it reduces the amount of past data used for training the autoregressive model. We observe that predicting 40% of the total feature length performs the best, and thus we set K as $d\times40\%$ in our experiments. Figures 4.5(b) and 4.5(c) show the results of varying λ_1 and λ_2 in a range between 0.001 and 1000 respectively. We fix $\lambda_1 = 1$ and change the values of λ_2 in Figure 4.5(c). We observe that our model achieves good performance when $\lambda_2 \approx 1$, whereas the model performs best with $\lambda_2 = 0.7$. Consequently, we fix $\lambda_2 = 0.7$ and tune the value of λ_1 as in Figure 4.5(b), where we find that our model achieves the best performance when $\lambda_1 = 1$. We also find that as $\lambda_1 < 10$, our model is less sensitive to its value, while it is more sensitive to different values of λ_2.



(a) HAR

(b) Sleep-EDF



(c) Epilepsy

FIGURE 4.6: Sample from each adopted dataset after normalization along with its weak and strong augmented views. The first row is the original samples, the second row shows the weak augmented views, and the third row shows the strong augmented views.

4.3.6 Augmentation Selection

The proper selection of augmentations is crucial for contrastive learning techniques, as contrastive methods are sensitive to the choice of augmentations [41]. Many studies showed that composing multiple data augmentation operations achieves better performance for image data [41, 168]. However, the selection of the proper augmentations that well-fit time series data is less explored and still an open problem [169]. Here, we aim to study the choice of suitable augmentations for our contrastive learning problem.

We define the weak augmentation as the one that applies limited change on the

TABLE 4.5: A study of TS-TCC *linear evaluation* performance with 5% of labeled HAR data when using different variations of weak and strong augmentations.

Weak Augmentation	Strong Augmentation	Accuracy	MF1-score
scale	no Aug	46.12	36.67
scale + jitter	no Aug	56.98	50.88
no Aug	permutation	62.07	52.64
no Aug	jitter + permutation	72.35	68.03
time Shift	jitter + permutation	71.57	66.97
time Shift + jitter	jitter + permutation	<u>74.69</u>	<u>69.33</u>
scale	jitter + permutation	72.59	68.90
scale + jitter	jitter + permutation	**77.58**	**76.66**

shape of the original signal as shown in the second row of Fig. 4.6. Examples of weak augmentations include scaling and time shifting. On the other hand, strong augmentation makes strong perturbations on the signal shape while keeping some of its temporal information, such as permutation. This is shown in the third row of Fig. 4.6. Permutation includes splitting the signal into M chunks and shuffling their order. Notably, some augmentations such as jittering (i.e., adding random noise) can be considered as both strong and weak augmentations depending on the added noise level. To justify the selection of our proposed augmentations, we provide a systematic study on the impact of applying several different data augmentations for each view in TS-TCC (Fig. 4.1). The results are provided in Table 4.5.

We first apply weak augmentation only, i.e., scaling, to one view without applying any augmentation to the other view, and the accuracy is only 46.12% as shown in Table 4.5. However, by composing jitter to scaling, we can observe a large performance improvement. Meanwhile, we also apply strong augmentation only and keep the other view without augmentations. The accuracy, in this case, is 62.07%, which is much higher than applying only weak augmentation. Besides, we notice that adding a jitter to the permutation can further improve the accuracy to 72.35%.

We also test to apply another weak augmentation that does not highly affect the signal characteristics along with the strong augmentation. We can find that using time shift as a weak augmentation achieves relatively good performance, i.e., an accuracy of 71.57%. Again, it can be found that adding jitter to time shift further improves the accuracy to 74.69%. Similar results are achieved when replacing

time-shift with scaling. In addition, applying only weak or strong augmentations to both views results in poorer performance as shown in Table 4.4. At this point, we find that applying weak augmentation for one view and strong augmentations for other views achieves the best performance, i.e., an accuracy of 77.58%. As an explanation, weak augmentations apply limited changes to the shape of the original signal as shown in Fig. 4.6, which helps the model to perform well on the test data.

Via observing the different characteristics of three different samples in Fig. 4.6 in terms of signal magnitude and sampling rates, we conclude that the choice of proper parameters for augmentations (e.g., jitter and scaling ratio) will vary from one dataset to another. Consequently, the range of parameter choices will be highly dependent on the characteristics of each time series data. Therefore, we propose to normalize the signals as a preprocessing step in our framework to improve parameter selection. For example, the value of added jitter after normalizing the signals in weak augmentation should be less than the ones added in the strong augmentation. In the experiments, we observe that the best practice is to normalize the data between 0 and 1 and set the weak jitter to be in the range [0, 0.1] and the strong jitter in the range [0.1, 1]. Similarly, a scaling ratio of 2 would be sufficient for weak augmentation in any time series signals.

Similarly, for strong augmentation, it is important to properly select the number of chunks M, where the value of M in time series data with longer sequences should be greater than its value in those with shorter sequences. We find that selecting 40% of the total feature size achieves the best performance on the three datasets (see Section 4.3.5).

4.4 Performance of AttnSleep under TS-TCC

We also validate the efficacy of TS-TCC to improve the performance of our AttnSleep model (See Chapter 3) In Figure 4.7, we compare the *supervised* performance of AttnSleep against the *fine-tuned* AttnSleep model when pretrained with TS-TCC using several fractions of labeled data (AttnSleep is trained without the class-aware loss).

Our experimentation reveals that by fine-tuning the model with only 1% of labeled samples, a considerable improvement of 5.6% in performance is attained.

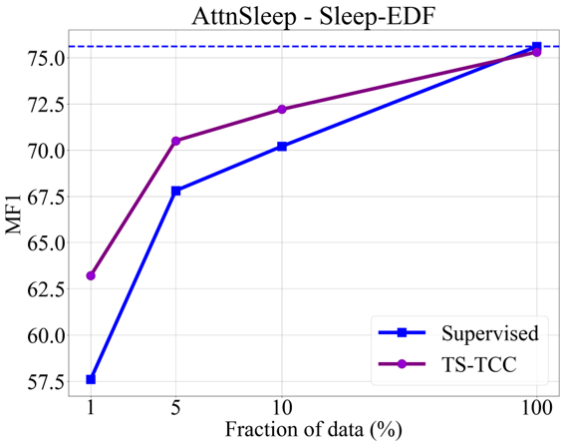


FIGURE 4.7: comparison between the supervised performance of AttnSleep against the fine-tuned AttnSleep model when pretrained with TS-TCC using several fractions of labeled data.

Additionally, when fine-tuning with 5 or 10% of labeled data, the model's performance closely approaches that achieved through supervised training with 100% of labeled data. This observation indicates that the embeddings obtained through TS-TCC self-supervised pretraining are richer than their supervised counterparts, thereby enhancing performance in downstream tasks, even with a limited quantity of labeled data.

Moreover, we notice that the benefits of self-supervised pretraining tend to diminish when the model is fine-tuned with fully labeled data. Thus, we infer that self-supervised pretraining can provide better regularization, effectively addressing the overfitting issue. However, it does not improve optimization to alleviate the underfitting problem, which is consistent with the conclusions drawn in [170].

4.5 Summary

We propose a novel framework called TS-TCC for unsupervised representation learning from time series data. The proposed TS-TCC framework first creates two views for each sample by applying strong and weak augmentations. Then the temporal contrasting module learns robust temporal features by applying a tough cross-view prediction task. We further propose a contextual contrasting module to learn discriminative features upon the learned robust representations. The experiments show that a linear classifier trained on top of the features learned

by our TS-TCC performs comparably with supervised training. In addition, our proposed TS-TCC shows high efficiency on few-labeled data and transfer learning scenarios, e.g., our TS-TCC by using only 10% of the labeled data can achieve close performance to the supervised training with full labeled data.

Chapter 5

Self-supervised Contrastive Representation Learning for Semi-supervised Time Series Classification

In Chapter 4, we discussed how to learn representations from unlabeled data with our proposed TS-TCC. However, it can be affordable in some real-world scenarios to label a few samples. In this chapter, we discuss how to further improve the learned representations by TS-TCC provided these few labeled samples. Therefore, in this chapter [1], we extend TS-TCC to the semi-supervised learning settings and propose a **C**lass-**A**ware TS-**TCC** (**CA-TCC**). Specifically, train the model in four phases, where we start with the self-supervised training in the first phase, then fine-tune the pretrained model with the few labels in the second phase. Next, in the third phase, we use the fine-tuned model to assign pseudo labels to the unlabeled sample set. Finally, the last phase includes leveraging the pseudo labels to realize a class-aware contrastive loss for the semi-supervised training. Extensive experiments show that the linear evaluation of the features learned by our proposed framework performs comparably with the fully supervised training. Additionally, our framework shows

[1] The work in this chapter has been published as "Self-supervised Contrastive Representation Learning for Semi-supervised Time-Series Classification", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) [171].

high efficiency in a few labeled data and transfer learning scenarios. The code is publicly available at `https://github.com/emadeldeen24/CA-TCC`.

In summary, our contributions in this chapter include:

- A novel contrastive learning framework for time series representation learning, i.e., CA-TCC in semi-supervised settings. CA-TCC builds on top of the learned representation of our proposed TS-TCC (see Chapter 4).
- We propose a supervised contextual contrasting module to further learn discriminative representations upon the robust representations while considering the potential positive pairs in contrastive learning.
- We perform extensive experiments on ten real-world datasets. Experimental results show that the results tend to improve when having few labeled samples.

5.1 Methods

This section describes the components of the proposed framework in detail. Our proposed CA-TCC starts with the TS-TCC pretraining phase and includes three more phases to complete the semi-supervised training, as shown in Fig. 5.1. In the next subsections, we will introduce the CA-TCC phases in more detail.

5.1.1 Class-Aware TS-TCC

We propose a second variant of our TS-TC framework, called the Class-Aware TS-TCC (CA-TCC), to operate in semi-supervised settings. In this variant, we attempt to exploit the available few labeled data to further improve the representation learned by TS-TCC. Our semi-supervised learning framework is performed in four phases, as shown in Fig. 5.1. In Phase 1, we start with a randomly-initialized encoder and used it in TS-TCC self-supervised pretraining. In Phase 2, we fine-tune the pretrained encoder with the few labeled data. In Phase 3, we deploy the fine-tuned encoder to create pseudo labels for the entire unlabeled dataset. Finally,

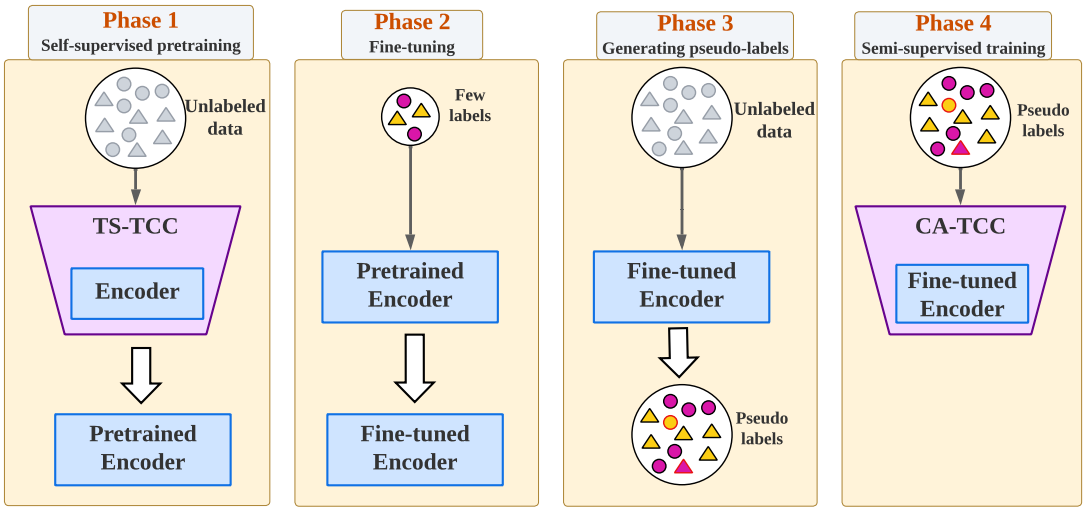


FIGURE 5.1: The four phases for CA-TCC semi-supervised training. In Phase 1, TS-TCC is trained with fully unlabeled data. Next, we use the available few labeled samples to fine-tune the pretrained encoder in Phase 2. Following that, in Phase 3, we generate pseudo labels for the unlabeled data with the fine-tuned TS-TCC encoder. Finally, Phase 4 includes deploying the pseudo-labeled data in the semi-supervised training.

in Phase 4, we train the class-aware semi-supervised framework, i.e., CA-TCC, with the pseudo labels.

In the class-aware semi-supervised training, we replace the unsupervised contextual contrasting with a supervised contextual contrasting module. This module benefits from the pseudo-labeled data to train a supervised contrastive loss [168]. This loss considers samples with the same class label as positive pairs, while samples from different classes are considered as negative pairs. This is different from the unsupervised contrastive loss that only forms positive pairs from the augmented views of the sample, as depicted in Fig. 5.2. This difference can affect the performance of the model, as the unsupervised contrastive loss may treat samples having a similar class as negative pairs.

Formally, assuming that the dataset consists of B labeled samples $\{\mathbf{x}_k, y_k\}_{k=1\ldots B}$, then after applying augmentations, the dataset becomes of $2B$ samples, $\{\hat{\mathbf{x}}_l, \hat{y}_l\}_{l=1\ldots 2B}$ such that $\hat{\mathbf{x}}_{2k}$ and $\hat{\mathbf{x}}_{2k-1}$ are the two views of $\mathbf{x}_k$, and similarly $y_k = \hat{y}_{2k} = \hat{y}_{2k-1}$. Also assuming that $i \in I \equiv \{1 \ldots 2B\}$ represents the index of an arbitrary augmented sample, and $A(i) \equiv I \setminus \{i\}$, the supervised contextual contrasting loss can

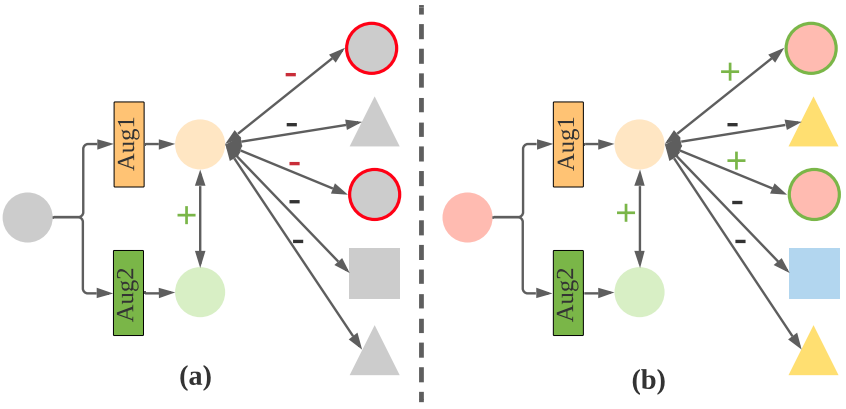


FIGURE 5.2: (a) Unsupervised contrasting vs. (b) supervised contrasting. The unsupervised contrasting forms positive pairs only from the augmented views of the sample, which may deteriorate the performance. In contrast, supervised contrasting considers the semantics of the data and forms positive pairs among all the samples having the same class label. In CA-TCC, we deploy the pseudo-labeled data to apply the supervised contrasting in the semi-supervised training.

be expressed as:

$$\mathcal{L}_{SCC} = \sum_{i \in I} \frac{1}{|P(i)|} \sum_{p \in P(i)} \ell(i, p), \tag{5.1}$$

$$P(i) = \{p \in A(i) : \hat{y}_p = \hat{y}_i\}, \tag{5.2}$$

where $P(i)$ is the set of indices of all samples with the same class as the sample $\hat{\mathbf{x}}_i$ in the batch, and (i, p) for any $p \in P(i)$ is thus a positive pair. $|P(i)|$ is the cardinality of $P(i)$. Thus, the overall loss can be expressed as:

$$\mathcal{L}_{semi} = \lambda_3 \cdot (\mathcal{L}^s_{TC} + \mathcal{L}^w_{TC}) + \lambda_4 \cdot \mathcal{L}_{SCC}, \tag{5.3}$$

where λ_3 and λ_4 are fixed scalar hyperparameters denoting the relative weight of each loss.

5.2 Experimental Setup

5.2.1 Datasets

To comprehensively evaluate our proposed model, we adopted seven more datasets than those used by our TS-TCC, which helps covering more time series applications. Specifically, in addition to HAR, Sleep-EDF, Epilepsy, and FD datasets (see Section 4.2.1 in Chapter 4), we also included seven UCR datasets[2] in our experiments. The selected datasets are suitable for our few labels experiments, and they satisfy the condition that 1% of samples allow all the classes to be present during the training phase. The included datasets are Wafer, FordA, FordB, PhalangesOutlinesCorrect (POC), ProximalPhalanxOutlineCorrect (PPOC), StarLightCurves, and ElectricDevices. The description of these datasets is as follows.

5.2.2 Wafer

Wafer dataset is a collection of several sensor measurements during the inline process control of silicon wafers for semiconductor fabrication. Each sample represents the recordings by one sensor during the processing of one wafer by one tool. The dataset contains two imbalanced classes, i.e., normal and abnormal, where only 10.7% and 12.1% of the train and test sets belong to the abnormal class.

5.2.3 FordA and FordB

These two datasets were used in a competition at the IEEE World Congress on Computational Intelligence, 2008. The timesteps in both datasets represent 500 measurements of engine noise, and the objective was to classify whether a certain symptom in an automotive subsystem exists or not. For the **FordA** dataset, the data was collected under minimal noise contamination, while **FordB** was collected in noisy conditions.

[2] `https://timeseriesclassification.com/dataset.php`

5.2.4 PhalangesOutlinesCorrect (POC) and ProximalPhalanxOutlineCorrect (PPOC)

Both datasets are extracted from a dataset designed to test the efficiency of the hand and bone outlines in bone age prediction. Some algorithms were applied to over 1,300 images to automatically extract hand outlines besides the outlines of three bones of the middle finger, i.e., proximal, middle, and distal phalanges. This generated three classification problems including **ProximalPhalanxOutlineCorrect**, while **PhalangesOutlinesCorrect** is the concatenation of these three problems. The labels in both datasets indicate whether the image outlining is correct or incorrect.

5.2.5 StarLightCurves

This dataset relates to an astronomical study for starlight curves, and the time series represents the brightness of a celestial object as a function of time. Each curve represents an example from one of three classes: Classical Type-I Cepheids, Eclipsing Binaries (EB), RRab, and RRc RRLyrae (RRL).

5.2.6 ElectricDevices

This dataset was collected to study the usage behavior of consumers to the electricity in homes to reduce the carbon footprint in the UK. The readings were collected from 251 households.

Table 5.1 summarizes the statistical details of each dataset, i.e., the number of training samples (# Train) and testing samples (# Test), the length of the sample, the number of sensor channels (# Channel) and the number of classes (# Class).

5.2.7 Implementation Details

We followed the same experimental setup in TS-TCC (Chapter 4) in terms of data splitting, usage of 5 random seeds, using 40 epochs for training, the same batch size

TABLE 5.1: A brief description of the 10 datasets used in our experiments. For the FD dataset, we mentioned the settings of one working condition, as they are the same for the four working conditions.

Dataset	# Train	# Test	Length	# Channel	# Class
HAR	7,352	2,947	128	9	6
Sleep-EDF	25,612	8,910	3,000	1	5
Epilepsy	9,200	2,300	178	1	2
FD	8,184	2,728	5,120	1	3
Wafer	1,000	6,174	152	1	2
FordA	1,320	3,601	500	1	2
FordB	810	3,636	500	1	2
POC	1,800	858	80	1	2
PPOC	600	291	80	1	2
StarLightCurves	1,000	8,236	1,024	1	3
ElectricDevices	8,926	7,711	96	1	7

and optimizer parameters, and augmentations parameters. We also kept λ_1 and λ_2 to 1 and 0.7, and set $\lambda_3 = 0.01$ and $\lambda_4 = 0.7$ (see Section 5.3.5) for CA-TCC.

5.3 Experimental Results

5.3.1 Comparison with Baseline Approaches

To examine the efficacy of our CA-TCC in semi-supervised settings, we compare it against the following semi-supervised learning baselines.

1. Mean-Teacher [172]: it consists of two models. The first is the student model, which is the base regular model. The second is the teacher model, which is the averaging of the student model weights using the Exponential Moving Average (EMA) over the training steps.

2. DivideMix [173]: it discards the labels that are likely to be noisy from the training and leverages them as unlabeled data to regularize the model against overfitting.

3. SemiTime [80]: it learns the temporal relations in unlabeled data by splitting the signal into past-future pairs, then contrasting the past of one sample with two future splits from another sample.

TABLE 5.2: Results of different SEMI-SUPERVISED baselines with 1% and 5% of labels. **Best results** across each row are in bold, while the second-best results are underlined.

1% of labeled data												
	Supervised		Mean-Teacher		DivideMix		SemiTime		FixMatch		CA-TCC	
Datasets	Accuracy	MF1-score	Accuracy	MF1-score	Accuracy	MF1-score	Accuracy	MF1-score	Accuracy	MF1-score	Accuracy	MF1-score
HAR	44.9±6.7	41.0±6.8	75.9±1.9	74.0±2.8	76.5±0.7	75.4±1.0	**77.6±1.1**	**76.3±0.9**	76.4±2.5	75.6±2.8	77.3±0.6	76.2±0.1
Sleep-EDF	34.1±0.3	30.8±0.8	73.6±1.0	63.7±0.3	76.5±2.2	66.6±0.8	73.6±3.9	63.4±2.9	72.5±2.2	62.0±2.9	**79.4±0.1**	**70.8±0.5**
Epilepsy	76.1±0.7	74.8±0.4	91.5±0.3	90.6±0.6	90.9±0.7	89.4±1.4	91.6±0.3	90.8±0.6	**93.2±0.2**	**92.2±0.5**	92.0±0.1	91.9±0.1
Wafer	91.9±1.3	67.6±9.2	94.7±0.2	84.7±0.3	93.2±0.5	82.0±0.8	94.4±0.6	84.4±1.2	95.0±0.4	84.8±1.2	**95.1±0.3**	**85.1±0.6**
FordA	56.4±1.6	54.4±3.5	71.7±1.6	71.5±1.8	73.7±1.1	73.3±0.9	75.1±1.3	74.4±1.4	74.5±0.4	74.3±0.4	**82.3±1.1**	**81.7±1.3**
FordB	51.9±2.6	48.0±3.6	65.9±2.8	65.8±2.8	54.5±2.8	54.1±3.2	67.6±2.2	67.5±2.3	56.7±5.9	55.4±6.9	**73.8±1.5**	**73.0±1.8**
POC	62.0±0.8	40.0±2.1	62.1±0.3	40.8±1.2	62.1±0.6	40.7±2.1	62.0±0.5	40.4±1.6	61.9±0.5	40.0±1.8	**63.4±0.4**	**49.3±0.7**
PPOC	64.3±0.7	64.2±0.7	**65.3±1.9**	**64.6±1.3**	56.1±6.9	55.6±7.2	64.8±0.5	64.6±0.6	63.7±1.9	63.5±1.8	63.4±0.3	63.1±0.3
StarLightCurves	78.8±0.9	71.4±0.1	79.4±0.5	77.7±0.6	79.0±0.5	77.2±0.4	79.5±0.5	77.8±0.6	77.2±0.3	71.6±0.1	**85.8±0.7**	**77.8±0.5**
ElectricDevices	57.8±1.1	47.5±1.0	48.9±8.2	48.3±1.5	59.8±3.9	49.4±2.6	57.3±3.7	48.1±2.7	58.2±1.0	46.9±0.4	**65.9±0.8**	**56.7±1.1**
Average	61.8	54.0	72.9	68.2	72.2	66.4	74.4	68.8	72.9	66.6	**77.8**	**72.6**
5% of labeled data												
HAR	52.8±1.5	50.9±0.2	88.2±1.2	88.1±1.2	**89.1±2.0**	**89.1±1.3**	87.6±1.3	87.1±0.8	87.6±0.3	87.3±0.4	88.3±0.4	88.3±0.3
Sleep-EDF	60.5±3.9	54.8±5.5	75.2±0.4	64.8±0.8	75.4±1.3	65.4±1.4	76.5±0.5	65.9±0.9	75.7±1.5	65.1±2.2	**82.1±0.2**	**74.6±0.1**
Epilepsy	83.4±0.7	80.4±0.6	94.0±0.4	93.6±0.7	93.9±0.6	93.4±1.1	94.0±0.5	93.0±0.9	93.7±1.4	92.4±0.3	**94.5±0.1**	**94.0±0.1**
Wafer	94.6±0.3	83.9±0.6	94.4±0.7	83.8±1.4	94.7±0.6	84.6±1.5	95.0±0.4	84.7±1.0	94.9±0.6	84.4±1.2	**95.8±0.2**	**85.2±0.6**
FordA	54.5±4.3	44.1±8.0	82.6±1.6	82.5±1.7	84.0±2.0	83.9±2.1	83.8±1.5	83.7±1.5	83.8±2.2	83.8±2.3	**90.9±0.3**	**90.8±0.3**
FordB	60.5±2.8	58.8±3.7	64.6±3.8	62.7±5.5	60.2±5.6	57.9±7.1	65.0±4.9	62.6±7.1	62.7±5.8	60.7±7.5	**88.2±0.4**	**88.2±0.4**
POC	61.4±0.0	38.3±0.0	62.1±0.6	41.2±2.5	62.9±1.3	45.9±7.0	62.4±0.5	41.8±1.7	63.1±1.4	43.6±4.3	66.4±0.3	52.8±0.3
PPOC	69.1±2.4	62.2±6.4	73.4±7.6	68.2±3.5	69.4±8.1	67.6±5.8	71.7±6.8	68.6±4.6	72.9±2.4	68.0±0.4	**73.7±6.2**	**69.1±3.7**
StarLightCurves	81.8±0.8	71.4±4.1	84.9±2.0	83.9±1.4	85.6±2.8	84.1±2.1	84.6±4.8	83.8±3.7	84.1±2.0	77.5±3.0	**88.8±0.7**	**81.1±2.0**
ElectricDevices	59.7±0.7	55.6±0.8	70.1±0.9	60.9±2.4	**72.0±1.9**	**62.1±0.6**	71.6±1.0	61.1±0.9	62.6±1.6	55.5±1.3	66.4±1.0	59.3±0.7
Average	67.8	60.0	79.0	73.0	78.7	73.4	79.2	73.2	78.1	71.8	**83.5**	**78.3**

4. FixMatch [75]: it generates pseudo labels for the weak augmented view of the signal and uses it to produce pseudo labels for the strong augmented view if it exceeds a confidence threshold.

Notably, the four baselines train a cross-entropy loss on the labeled portion of the samples. Also, for FixMatch, we apply our proposed weak and strong augmentations for training.

Table 5.2 compares the performance of CA-TCC against semi-supervised baselines across 10 different datasets. Overall, the average F1-score of our CA-TCC across the 10 datasets outperforms the average F1-score of the second-best performing semi-supervised baseline with 3.8% and 4.9% in 1% and 5% of labels, respectively. Specifically, our CA-TCC ranks first in seven datasets and second in two other datasets. The consistent superior performance of our CA-TCC across different datasets and labeling budgets demonstrates its effectiveness in maximizing the utilization of labeled data in semi-supervised settings.

5.3.2 CA-TCC Model Analysis

We discuss the different aspects that yield the improved performance of the CA-TCC. Specifically, we regard the improved performance of CA-TCC to have two

main factors. The first is the effective representations learned by TS-TCC in Phase 1, which enables the generation of high-quality pseudo labels. This is demonstrated in Table 4.2, where TS-TCC outperforms other self-supervised learning baselines. Therefore, it was anticipated that the fine-tuned TS-TCC encoder should be superior when used to produce the pseudo labels. The second factor is the supervised contrastive loss in CA-TCC, which uses the pseudo-labeled data to include more positive pairs in the contrastive loss. Nevertheless, our CA-TCC is flexible in architecture and can anticipate different models throughout its different phases. Next, we attempt to validate the aforementioned factors and support our conclusions.

Quality of pseudo labels To investigate the effect of pseudo labels, we compare the quality of the pseudo labels generated by each baseline. Specifically, we replace our TS-TCC (in Phase 1) with SSL-ECG, SimCLR, and CPC. Then, for each baseline, we fine-tune the encoder and use it to generate pseudo labels (Phases 2 and 3) and then, conduct the semi-supervised training. Figure 5.3 shows the accuracy of the generated pseudo labels (generated in Phase 3) compared to the true labels. We find that the pseudo labels generated by TS-TCC are more accurate than those generated by the other baselines. In addition, Table 5.3 reports the performance with different baselines (deployed in Phase 1), while fixing CA-TCC (in Phase 4) for two datasets with different scales, i.e., HAR and Sleep-EDF. We observe that using TS-TCC in both phases results in the best performance, supporting our conclusion and justification for the first reason for improved performance.

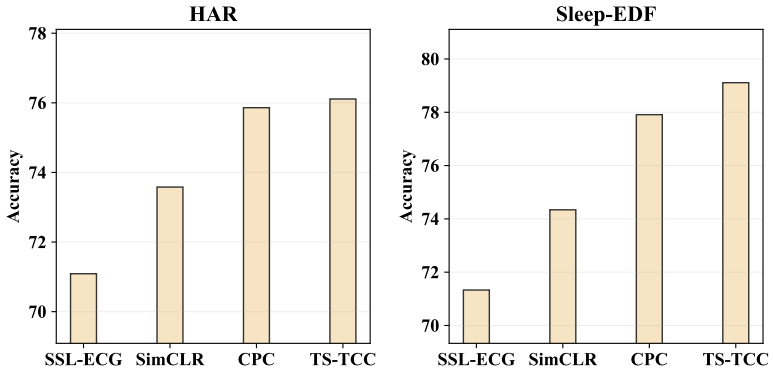


FIGURE 5.3: Accuracy of generated pseudo labels with different self-supervised learning methods applied to HAR and Sleep-EDF datasets. Fine-tuning in Phase 2 is performed with 1% of labels.

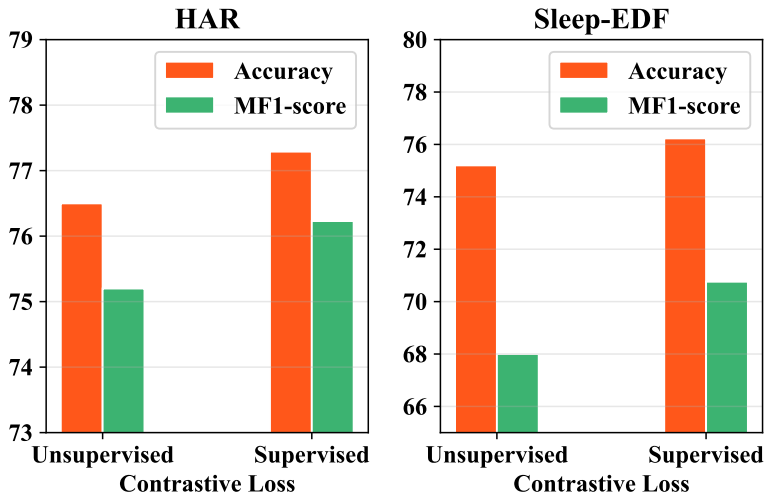


FIGURE 5.4: Performance comparison with and without supervised contrastive loss in the semi-supervised training (Phase 4).

Effect of supervised contrastive loss In this experiment, we compare the performance of the semi-supervised training (Phase 4) when using the supervised contrastive loss against the unsupervised contrastive loss. The supervised contrastive loss considers samples having the same class label as positive pairs and samples from different classes as negative pairs. In contrast, the unsupervised contrastive loss only forms positive pairs from the augmented views of the sample, and all other samples in the mini-batch are considered as negative pairs. This difference can affect the performance of the model, as the unsupervised contrastive loss may treat samples having a similar class as negative pairs.

Figure 5.4 compares the performance of the unsupervised and the supervised contrastive losses in terms of accuracy and F1-score on the HAR and Sleep-EDF datasets. The results indicate that the use of the supervised contrastive loss, which includes more positive pairs, results in improved performance and better representation learning on both datasets.

Varying CA-TCC model architecture In this section, we evaluate the performance of different combinations of self-supervised learning algorithms in Phases 1 and 4. Specifically, we use different self-supervised algorithms in Phase 1, while in Phase 4, we deploy only two baselines that can use the supervised contrastive loss, i.e., SimCLR and TS-TCC. The experiments are conducted on the HAR and Sleep-EDF datasets using 1% of labels.

TABLE 5.3: Different combinations for the first and second pretraining phases in our framework. Pretraining in the first phase is fully unsupervised, while the second phase is class-aware (TS-TCC or SimCLR) with generated pseudo labels from fine-tuning with 1% of labels.

		HAR		Sleep-EDF	
Unsupervised pretraining	Semi-supervised training	Accuracy	MF1-score	Accuracy	MF1-score
SSL-EEG	CA-(TS-TCC)	70.8±0.9	68.4±0.8	76.1±0.4	67.1±0.6
SimCLR	CA-(TS-TCC)	73.2±1.2	70.5±1.2	78.3±0.7	68.1±0.5
CPC	CA-(TS-TCC)	72.7±0.7	69.6±0.8	79.2±0.4	70.0±0.6
TS-TCC	CA-(TS-TCC)	**77.3±0.6**	**76.2±0.1**	**79.4±0.1**	**70.8±0.5**
SSL-ECG	CA-(SimCLR)	69.3±1.3	66.2±1.1	75.5±0.2	66.2±0.9
SimCLR	CA-(SimCLR)	67.5±1.4	64.3±1.2	77.5±0.2	67.9±0.6
CPC	CA-(SimCLR)	72.2±1.9	69.1±2.6	79.2±0.1	70.1±0.7
TS-TCC	CA-(SimCLR)	75.0±1.5	73.1±2.0	79.3±0.3	70.2±0.1

The results, shown in Table 5.3, indicate that using class-aware TS-TCC in the semi-supervised training with any baseline in Phase 1 consistently outperforms class-aware SimCLR for both datasets. This superior performance of TS-TCC can be regarded to its ability to learn temporal relations in time series data while also benefiting from the supervised contrasting, unlike class-aware SimCLR which only learns through contrasting positive and negative pairs.

In summary, our framework is flexible and allows the use of various models in different training phases. However, we chose to only use TS-TCC as it is the best-performing self-supervised learning model among the baselines and it can handle class-aware training in semi-supervised settings.

5.3.3 Transfer Learning Experiment

We further examine the transferability of the learned features by designing a transfer learning experiment. We use the Fault Diagnosis (FD) dataset for the evaluation under the transfer learning setting. Recall that the FD dataset has four working conditions, which are considered as four domains (denoted as domains A, B, C, and D). Here, we train the model on the data from one condition (i.e., source domain) and test it on another condition (i.e., target domain). We adopt three training

TABLE 5.4: Cross-domain transfer learning experiments performed on the Fault Diagnosis dataset. Domains A, B, C, and D represent the four working conditions. For supervised training, we train on the source domain and directly test on the target domain. For TS-TCC, we pretrain the model with the unlabeled source domain data, then fine-tune it with the few labels, then test it on the target domain. The same applies to CA-TCC, except that we include 1% of labeled data in the training. Results are reported in terms of accuracy. **Best** results are in bold, while the second-best ones are underlined.

Method	A→B	A→C	A→D	B→A	B→C	B→D	C→A	C→B	C→D	D→A	D→B	D→C	AVG
Supervised	34.38	44.94	34.57	**52.93**	63.67	99.82	**52.93**	84.02	83.54	**53.15**	99.56	62.43	63.83
TS-TCC	43.15	51.50	42.74	47.98	70.38	99.30	38.89	98.31	99.38	51.91	99.96	70.31	67.82
CA-TCC	**44.75**	**52.09**	**45.63**	46.26	**71.33**	**100.0**	52.71	**99.85**	**99.84**	46.48	**100.0**	**77.01**	**69.66**

schemes on the source domain, namely, (1) Supervised training, (2) TS-TCC fine-tuning, and (3) CA-TCC fine-tuning. In TS-TCC and CA-TCC fine-tuning, we fine-tune our pretrained encoder using the labeled data in the source domain.

Table 5.4 shows the performance of the three training schemes under 12 cross-domain scenarios. Our pretrained TS-TCC model consistently outperforms the supervised pretraining in 8 out of 12 cross-domain scenarios. Similarly, with only 1% of labels in each source domain for training, we find that the CA-TCC model outperforms the supervised pretraining in 9 out of 12 cross-domain scenarios. We find that the TS-TCC model can achieve at least ∼7% improvement in 7 out of 8 winning scenarios (except for D→B scenario). Similarly, the CA-TCC model can achieve at least ∼8% improvement in 7 out of 9 winning scenarios. Overall, our two proposed approaches can improve the transferability of learned representations over the supervised training by ∼ 4% and 6% in terms of accuracy.

5.3.4 Ablation Study

We study the effectiveness of each component in our proposed CA-TCC model. Specifically, we derive different model variants for comparison as follows. First, we train the Temporal Contrasting (TC) module without the cross-view prediction task, where each branch predicts the future timesteps of the same augmented view. This variant is denoted as 'TC only'. Second, we train the TC module by adding the cross-view prediction task, which is denoted as 'TC + X-Aug'. Third, we train the proposed TS-TCC model, which is denoted as 'TC + X-Aug + CC'. Finally, we train the proposed CA-TCC model, which is denoted as 'TC + X-Aug + SCC'. We also study the effect of using a single family of augmentations on the performance

TABLE 5.5: Ablation study of the effect of different components in TS-TCC and CA-TCC models. We also show the effect of using two weak or two strong augmentations on their performance. It is clear that using a combination of weak and strong augmentations yields the best performance. The results are obtained with the *linear evaluation* experiment on 5% of labeled data on three datasets.

	HAR		Sleep-EDF		Epilepsy	
Component	Accuracy	MF1-score	Accuracy	MF1-score	Accuracy	MF1-score
TC only	68.16±1.15	66.89±1.11	75.55±0.93	60.19±0.81	88.29±1.29	88.00±1.91
TC + X-Aug	74.22±1.03	72.18±0.99	77.80±0.29	61.28±1.22	90.51±0.43	89.27±0.22
TS-TCC (TC + X-Aug + CC)	77.58±1.78	76.66±1.96	76.98±0.56	70.94±0.46	93.12±0.31	93.67±0.56
CA-TCC (TC + X-Aug + SCC)	**88.27±0.38**	**88.29±0.34**	**82.14±0.19**	**74.75±0.06**	**94.52±0.12**	**94.00±0.09**
TS-TCC (Weak only)	67.39±1.73	65.54±2.42	79.63±0.16	68.15±0.23	93.22±0.11	91.97±0.19
CA-TCC (Weak only)	85.68±0.26	84.77±0.25	81.62±0.89	70.10±1.28	93.84±0.05	92.19±0.10
TS-TCC (Strong only)	50.37±1.18	43.05±1.42	74.84±0.50	64.53±0.58	92.49±0.62	90.60±0.20
CA-TCC (Strong only)	59.59±0.06	53.34±0.49	79.24±0.74	69.39±0.89	93.74±0.04	92.00±0.05

of TS-TCC and CA-TCC. In particular, for an input x, we generate two different views x_1 and x_2 from the same augmentation type, i.e., $x_1 \sim \mathcal{T}_w$ and $x_2 \sim \mathcal{T}_w$ when using either the weak augmentation or the strong augmentation. We show the linear evaluation results in terms of accuracy and macro F1-score with only 5% throughout these experiments.

Table 5.5 shows this ablation study on the three datasets. The proposed cross-view prediction task generates robust features and thus improves the performance by more than 6% accuracy on HAR datasets, and ∼2% on Sleep-EDF and Epilepsy datasets. Additionally, the contextual contrasting module further improves the performance, as it helps the features to be more discriminative. More improvement was achieved by using supervised contextual contrasting in CA-TCC, which supports the importance of considering more positive samples from the same class to generate more discriminative features. By studying the effect of augmentations on TS-TCC, we find that generating different views from the same augmentation type is not helpful with HAR and Sleep-EDF datasets. For these complex datasets, using only weak augmentations may not make a *tough* cross-view prediction task, leading to close results to the variant 'TC only'. Counterpart, using only strong augmentations deviates the model from recognizing the original data while testing. However, the less complex Epilepsy dataset can still achieve comparable performance with only one augmentation. For CA-TCC, we find that it consistently outperforms the results of TS-TCC, showing its effectiveness in improving the representations with the available few labeled samples. For example, we find that it highly improves the performance of using only weak or strong augmentations in

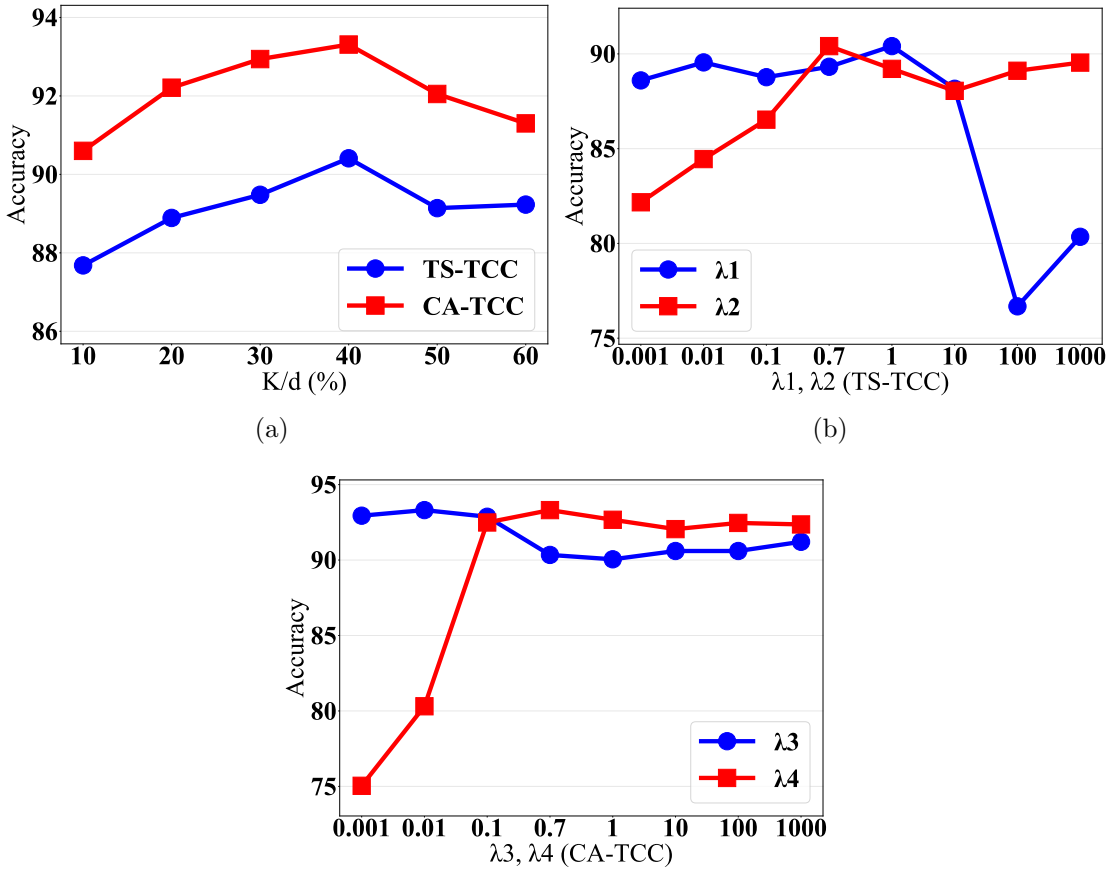


FIGURE 5.5: Sensitivity analysis experiments on HAR dataset. Figure (a) shows the effect of changing the percentage of the predicted future timesteps, where we notice a close performance among our two variants. Figure (b) shows the impact of the different combinations of λ_1 and λ_2 on TS-TCC performance. Last, figure (c) shows the effect of the different combinations of λ_3 and λ_4 onn CA-TCC performance.

both HAR and Sleep-EDF datasets.

5.3.5 Sensitivity Analysis

We perform sensitivity analysis on the HAR dataset to study five parameters namely, the number of predicted future timesteps K in the temporal contrasting module, λ_1 and λ_2 in Eq. 4.6, and λ_3 and λ_4 in Eq. 5.3. In specific, we used the linear evaluation experiment with full labels to assess the performance.

Fig. 5.5(a) shows the effect of K on the overall performance of TS-TCC and CA-TCC, where the x-axis is the percentage K/d, and d is the length of the features.

Clearly, increasing the percentage of the predicted future timesteps improves the performance. However, larger percentages can harm the performance as it reduces the amount of past data used for training the autoregressive model. We observe that predicting 40% of the total feature length performs the best, and thus we set K as $d\times40\%$ in our experiments. The same conclusion can be drawn for both variants of our framework. Fig. 5.5(b) shows the results of varying λ_1 and λ_2 in TS-TCC (Eq. 4.6) in a range between 0.001 and 1000 respectively. We first fix $\lambda_1 = 1$ and change the values of λ_2. We observe that our model achieves good performance when $\lambda_2 \approx 1$, whereas the model performs best with $\lambda_2 = 0.7$. Consequently, we fix $\lambda_2 = 0.7$ and tune the value of λ_1 as in Fig. 5.5(b), where we find that our model achieves the best performance when $\lambda_1 = 1$. We also find that as $\lambda_1 < 10$, our model is less sensitive to its value, while it is more sensitive to different values of λ_2.

We also perform sensitivity analysis on the values of λ_3 and λ_4 in CA-TCC (Eq. 5.3) in a similar manner and within the same ranges, as shown in Fig. 5.5(c). We also used 1% of labels to generate pseudo labels. Consequently, we chose the values of $\lambda_3 = 0.01$ and $\lambda_4 = 0.7$.

5.4 Summary

In this chapter, we showed how to further improve the self-supervised representation in semi-supervised settings. In particular, we noticed how TS-TCC can provide noticeable improvement under different labeling budgets, where 10% of the labeled data could achieve close performance to the supervised training with full labeled data. Therefore, we extended TS-TCC and proposed CA-TCC, which benefits from the pseudo labels generated by the fine-tuned TS-TCC model to train a class-aware supervised contrastive loss. CA-TCC was able to further improve this performance with only 1% of labeled data when testing on different datasets. In addition, both variants were able to improve the transferability of the representations in real-world transfer learning scenarios.

Chapter 6

Attentive Cross-domain Representation Learning Framework with Iterative Self-Training

In this chapter [1], we discuss another real-world problem that we may face when the train and test data are not drawn from the same distribution, i.e., there exists a domain shift. To address this problem, we propose an adversarial unsupervised domain adaptation framework based on an iterative self-training scheme. We apply our solution to the cross-domain sleep stage classification problem, in which we train the deep learning model on one dataset, and test it on another out-of-distribution dataset. In specific, we propose an **A**dversarial **D**omain **A**daptation framework based on preserving attention mechanism and iterative **S**elf-**T**raining strategy (**ADAST**) for a single channel EEG-based sleep stage classification.

In our ADAST, we aim to address two problems in the literature. The first is that they rely on a fully shared model for the domain alignment, which may lose the domain-specific information during feature extraction. Second, they only align the source and target distributions globally without considering the class information in the target domain, which hinders the classification performance of the model while

[1] The work in this chapter has been published as "ADAST: Attentive Cross-domain EEG-based Sleep Staging Framework with Iterative Self-Training", IEEE Transactions on Emerging Topics in Computational Intelligence, 2022 [174].

testing. Therefore, we first develop an unshared attention mechanism to preserve the domain-specific features in both domains. Second, we design an iterative self-training strategy to improve the classification performance on the target domain via target domain pseudo labels. We also propose dual distinct classifiers to increase the robustness and quality of the pseudo labels. The experimental results on six cross-domain scenarios validate the efficacy of our proposed framework and its advantage over state-of-the-art UDA methods. The source code is available at https://github.com/emadeldeen24/ADAST.

The main contributions of this work are summarized as follows:

- We propose a novel cross-dataset sleep staging framework that integrates iterative self-training with adversarial learning. Therefore, our framework can effectively classify the fine-grained distribution of the unlabeled target sleep data.
- ADAST utilizes an unshared domain-specific attention module to preserve the key features in both source and target domains during adaptation, which improves the adversarial training and boosts the classification performance on the target domain.
- We design distinct dual classifiers to improve the robustness of the generated pseudo labels in self-training. We also add a similarity constraint on their weights to push them from being identical.
- Extensive experiments demonstrate that our ADAST achieves superior performance for cross-domain sleep stage classification against state-of-the-art UDA methods.

6.1 Method

6.1.1 Preliminaries

In this work, we focus on the problem of unsupervised cross-domain adaptation for EEG-based sleep staging. In this setting, we have access to a labeled source dataset $X_s = \{(\mathbf{x}_s^i, y_s^i)\}_{i=1}^{n_s}$ of n_s labeled samples, and an unlabeled target dataset

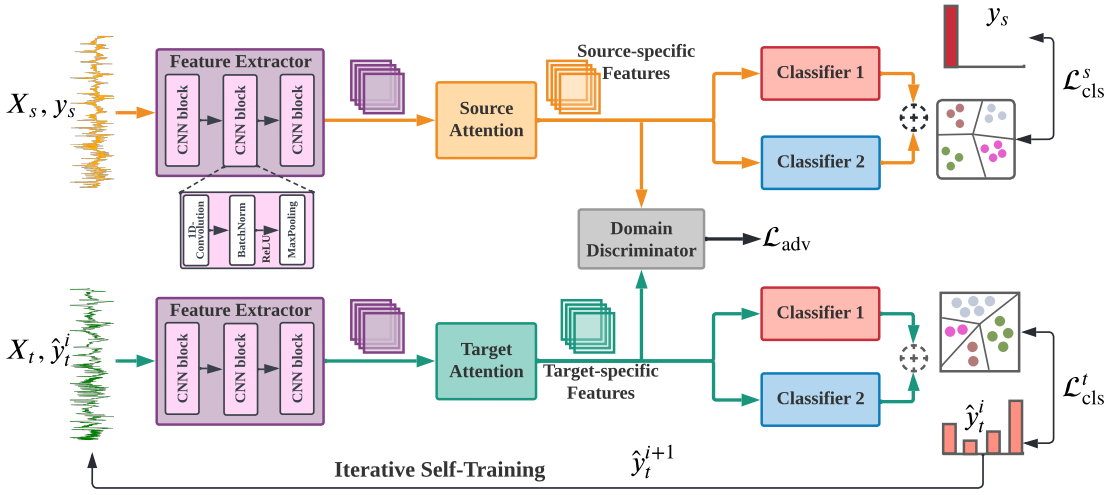


FIGURE 6.1: The overall architecture of the proposed ADAST framework. The shared feature extractor consists of three convolutional blocks, where each block contains 1D-convolution, batch normalization, non-linear ReLU activation, and MaxPooling. The two classifiers share the same architecture, but we apply a similarity constraint on their weights to push them from being identical to each other (best viewed in colors, as blocks with similar colors represent shared components).

$X_t = \{(\mathbf{x}_t^j)\}_{j=1}^{n_t}$ of n_t samples. The source and target domains are sampled from source distribution $P_s(X_s)$ and target distribution $P_t(X_t)$ respectively, such that these distributions are different (i.e., $P_s \neq P_t$). Both domains share the same label space $Y = \{1, 2, \ldots K\}$, where K is the number of classes (i.e., sleep stages). The domain adaptation scenario aims to transfer the knowledge from a labeled source domain to a domain-shifted unlabeled target domain. In the context of EEG data, both $\mathbf{x}_s^i$ and $\mathbf{x}_t^i \in \mathbb{R}^{1\times T}$, where the number of electrodes/channels is 1 since we use single-channel EEG data, and T represents the number of timesteps in the 30-second EEG epochs.

6.1.2 Overview

As shown in Fig. 6.1, our proposed framework consists of three main components, namely domain-specific attention, adversarial training, and dual classifier-based iterative self-training. First, domain-specific attention plays an important role in refining the extracted features so that each domain preserves its key features. Second, the adversarial training step leverages a domain discriminator to align the source and target features. Particularly, the domain discriminator network

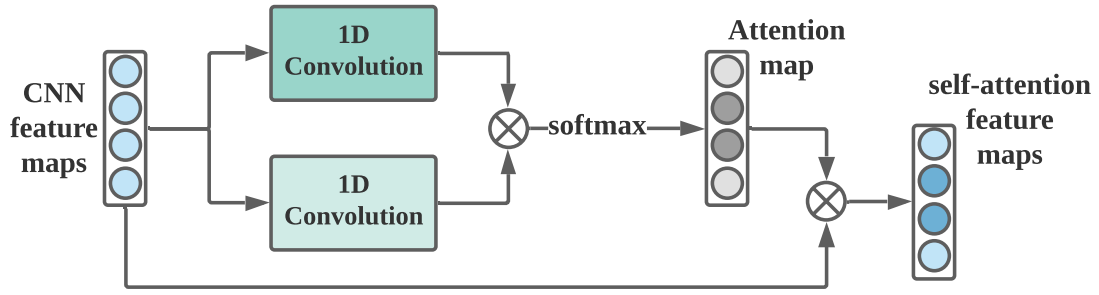


FIGURE 6.2: Design of domain-specific attention module.

is trained to distinguish between the source and target features while the feature extractor is trained to confuse the domain discriminator by generating domain invariant features. Finally, the iterative self-training strategy utilizes the target domain pseudo labels to adapt the classification decision boundaries according to the target domain classes. The dual classifiers are incorporated to improve the quality and robustness of the pseudo labels. Further details about each component will be provided in the following subsections.

6.1.3 Domain-specific Attention

Our proposed framework extracts domain invariant features by using a shared CNN-based feature extractor, i.e., $F_s(\cdot) = F_t(\cdot) = F(\cdot)$. Unlike the totally unshared architectures that require an extra pretraining step and are usually harder to converge, the shared feature extractor allows end-to-end training, besides being easier to converge. Therefore, most UDA algorithms adopted this shared design [96]. However, relying solely on this shared architecture may not be able to preserve the key features of each domain [175, 176]. The reason is that the high-dimensional features may contain information that distinguishes source and target domains, and are relevant for predicting the label [177]. Throughout the training, the model tries to remove these features to reduce the difference between source and target distributions and make the features domain-invariant, but this also affects the classification performance on each separate domain. As the classification mainly relies on the available source domain labels, the learned classifier will be more biased toward the source domain [178]. Hence, we propose an unshared attention module to learn both domain-invariant and domain-specific features jointly in our proposed framework.

For each position in the feature space, the attention module calculates the weighted sum of the features at all positions with a little computational cost. Thus, the features at each location have fine details that are coordinated with fine details in distant portions of the features. Formally, given an input source sample $\mathbf{x}_s \in \mathbb{R}^{1\times T}$ that is passed through the feature extractor to generate the source features, i.e., $F(\mathbf{x}_s) = (\mathbf{f}_{s1}, \ldots, \mathbf{f}_{sl}) \in \mathbb{R}^{d\times l}$, where d is the number of CNN channels, and l is the length of the features. Inspired by [179], we deploy a convolutional attention mechanism as shown in Fig. 6.2. The attention operation starts by obtaining a new representation for the features at each position by using two 1D-convolutions, i.e., H_1 and H_2. Specifically, given $\mathbf{f}_{si}, \mathbf{f}_{sj} \in \mathbb{R}^d$, which are the feature values at the positions i and j, they are transformed into $\mathcal{Z}_{si} = H_1(\mathbf{f}_{si})$ and $\mathcal{Z}_{sj} = H_2(\mathbf{f}_{sj})$. The attention scores are calculated as follows.

$$\mathcal{V}_{ji} = \frac{\exp(\mathcal{Z}_{si}^\top \mathcal{Z}_{sj})}{\sum_{k=1}^{l} \exp(\mathcal{Z}_{sk}^\top \mathcal{Z}_{sj})}. \tag{6.1}$$

Here, the attention score $\mathcal{V}_{ji}$ indicates the extent to which j^{th} position attends to the i^{th} position in the feature map. The output of the attention layer is $\mathcal{O}_s = (\mathbf{o}_{s1}, \ldots, \mathbf{o}_{sj}, \ldots \mathbf{o}_{sl}) \in \mathbb{R}^{d\times l}$, where

$$\mathbf{o}_{sj} = \sum_{i=1}^{l} \mathcal{V}_{ji}\mathbf{f}_{si}. \tag{6.2}$$

We denote the attention process in Equations 6.1 and 6.2 as $ATT(\cdot)$, such that $\mathcal{O}_s = ATT_s(F(\mathbf{x}_s))$. The same process applies to the target domain data flow to train ATT_t.

6.1.4 Adversarial Training

Given the learned source and target representations that preserve the domain-specific features, adversarial training is employed to align the source and target domains. Inspired by the generative adversarial network (GAN) [180], we aim to solve a minimax objective between the feature extractor and domain discriminator. Specifically, the domain discriminator is trained to classify between the source and target features, while the feature extractor tries to generate indistinguishable representations for both source and target domains. By doing so, the classifier

trained on the source domain can generalize well on the target domain. However, with the minimax objective, the discriminator can saturate quickly, resulting in a gradient vanishing problem [95]. To address this issue, we train our model using a standard GAN loss with inverted labels [180]. Formally, the domain discriminator, D, classifies the input features to be either from the source or target domain. Thus, D can be optimized using a standard cross-entropy loss with the labels indicating the domain of the data point. The objective of this operation $\mathcal{L}_D$ can be defined as:

$$\begin{aligned}\min_{D} \mathcal{L}_{\mathrm{D}} = &- \mathbb{E}_{\mathbf{x}_s \sim P_s}[\log D(ATT_s(F(\mathbf{x}_s)))] \\ &- \mathbb{E}_{\mathbf{x}_t \sim P_t}[\log(1 - D(ATT_t(F(\mathbf{x}_t))))],\end{aligned} \tag{6.3}$$

where $\mathcal{L}_{\mathrm{D}}$ is used to optimize the domain discriminator separately so that it discriminates the source and target features. On the other hand, the feature extractor and the domain-specific attention are trained to confuse the discriminator by mapping the target features to be similar to the source ones. The objective function can be described as:

$$\begin{aligned}\min_{F,A_s,A_t} \mathcal{L}_{\mathrm{adv}} = &- \mathbb{E}_{\mathbf{x}_s \sim P_s}[\log(1 - D(ATT_s(F(\mathbf{x}_s))))] \\ &- \mathbb{E}_{\mathbf{x}_t \sim P_t}[\log D(ATT_t(F(\mathbf{x}_t)))].\end{aligned} \tag{6.4}$$

Notably, only $\mathcal{L}_{\mathrm{adv}}$, which optimizes the feature extractor and the domain-specific attentions, is added to the overall objective function to ensure that the model can generate domain-invariant features.

6.1.5 Dual Classifier-based Iterative Self-Training

With adversarial training, the distributions of source and target domains become globally aligned. However, the global alignment does not guarantee a good classification performance on the target domain, because of the difference in classification boundaries among source and target domains. Therefore, we propose a novel iterative self-training strategy to adjust the classification boundaries to fit the target domain and improve its classification performance.

Self-training converts the target domain predictions into pseudo labels and uses them to minimize the cross-entropy loss [181]. Given high-quality pseudo labels, they are treated as supervisory signals to adapt the decision boundaries of the classifier according to target domain classes. However, due to the domain shift, finding high-quality target domain pseudo labels can be a challenging problem, and the generated ones might be noisy and inefficient, especially at the beginning of the training. Nevertheless, we aim to *first* minimize the number of incorrect pseudo labels and *second* minimize the negative impact of these incorrect ones on the performance. To do so, we follow two main strategies.

First, we repeat the training of the model for r iterations, where the pseudo labels generated in the previous iteration are used in the next one. Since the model will be very uncertain about the pseudo labels in the first iteration, we ignore the target classification loss at this iteration. However, in the following iterations, we take it into consideration since the model becomes more confident about the pseudo labels after being trained to minimize the domain shift between source and target domains. Second, we use dual classifiers C_1 and C_2 setup, which have two main benefits. First, it helps the model avoid the variance in the training data. Second, the average prediction vector of the two classifiers decreases the probability of low-confident predictions.

Notably, we design the two classifiers such that they share the same architecture, i.e., a single fully connected layer. This helps limiting the total number of parameters in the model and avoid pruning to overfitting. Consequently, we need to ensure that their predictions are diversified and they do not converge to become one classifier throughout training. Thus, we add a regularization term $|\theta_{C_1}^{\intercal}\theta_{C_2}|$ on the weights of the two classifiers as inspired by [182], where θ_{C_1}, θ_{C_2} represent the weights of C_1 and C_2 respectively. This regularization term ensures the diversity of the two classifiers and helps them to produce different yet correct predictions. The final prediction vector is the averaged vector of the predictions of both classifiers.

Formally, in each iteration, we first calculate the average probability $\mathbf{p}_t$ of the two classifiers, and the corresponding target pseudo labels $\hat{y}_t$ as follows.

$$\mathbf{p}_t = \frac{1}{2}\left[C_1(ATT_t(F(\mathbf{x}_t)) + C_2(ATT_t(F(\mathbf{x}_t))\right], \tag{6.5}$$

$$\hat{y}_t = argmax(\mathbf{p}_t). \tag{6.6}$$

The target classification loss $\mathcal{L}_{\text{cls}}^{t}$ based on the above pseudo labels is defined as follows.

$$\min_{F,A_t,C_1,C_2} \mathcal{L}_{\text{cls}}^{t} = -\mathbb{E}_{\mathbf{x}_t \sim P_t} \sum_{k=1}^{K} \mathbb{1}_{[\hat{y}_t=k]} \log \mathbf{p}_t^k, \tag{6.7}$$

where $\mathbb{1}$ is the indicator function, which is set to be 1 when the condition is met, and set to 0 otherwise. The target classification loss $\mathcal{L}_{\text{cls}}^{t}$ optimizes the feature extractor F, the target domain-specific attention ATT_t as well as the dual classifiers C_1 and C_2.

Similarly, the source classification loss $\mathcal{L}_{\text{cls}}^{s}$, which depends on the source labels y_s, is formalized as follows.

$$\mathbf{p}_s = \frac{1}{2}\left[C_1(ATT_s(F(\mathbf{x}_s)) + C_2(ATT_s(F(\mathbf{x}_s))\right], \tag{6.8}$$

$$\min_{F,A_s,C_1,C_2} \mathcal{L}_{\text{cls}}^{s} = -\mathbb{E}_{(\mathbf{x}_s,y_s)\sim P_s} \sum_{k=1}^{K} \mathbb{1}_{[y_s=k]} \log \mathbf{p}_s^k, \tag{6.9}$$

where the source classification loss $\mathcal{L}_{\text{cls}}^{s}$ optimizes the feature extractor F, the source domain-specific attention ATT_s as well as the dual classifiers C_1 and C_2.

To sum up, we integrate the adversarial loss with the source and target classification losses and the regularization of the dual classifiers in one objective loss function as follows.

$$\mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{cls}}^{s} + \lambda_1 \mathcal{L}_{\text{cls}}^{t} + \lambda_2 |\theta_{C_1}^{\intercal} \theta_{C_2}|. \tag{6.10}$$

Since the adversarial training and the source classification are two essential modules, we set their weights to one and tune the values of the two hyperparameters λ_1 and λ_2 to control their contributions. Overall, the three losses are integrated to guide the feature extractor to generate domain-invariant features, while allowing the domain-specific attentions to preserve the key features for each domain. Additionally, the dual classifiers are diversified using the regularization term. More details about the training can be found in Algorithm 1.

Algorithm 1: Training procedure of our proposed ADAST framework.

Input: X_s, X_t, D, F, ψ_s, ψ_t, $\mathcal{C}_1$, $\mathcal{C}_2$;
Output: Trained F, ψ_t for the target domain;
1. Set $\lambda_1 = 0$;
2. **for** *i=1* ***to*** r **do**
   3. Sample mini-batches from source domain $(\mathbf{x}_s, y_s) \sim P_s$ and target domain $\mathbf{x}_t \sim P_t$;
   4. Extract shared features using F, then unshared representations $\psi_s(F(\mathbf{x}_s))$, $\psi_t(F(\mathbf{x}_t))$;
   5. Update D by $\mathcal{L}_D$ (Eq. 6.3);
   6. Update F, ψ_s, ψ_t by $\mathcal{L}_{adv}$ (Eq. 6.4);
   7. Update $F, \psi_s, \mathcal{C}_1, \mathcal{C}_2$ by $\mathcal{L}^s_{cls}$ (Eq. 6.9);
   8. Penalize $\mathcal{C}_1$ and $\mathcal{C}_2$ weights similarity ;
   9. Generate pseudo labels $\hat{y}^i_t$ (Eq. 6.5,6.6);
   10. Update $F, \psi_t, \mathcal{C}_1, \mathcal{C}_2$ by $\mathcal{L}^t_{cls}$ (Eq. 6.7);
   11. Set a value to λ_1;
end

6.2 Experiments

6.2.1 Datasets

We evaluate the proposed framework on three challenging datasets, namely Sleep-EDF[2] (**EDF** for short), SHHS-1 (**S1**) and SHHS-2[3] (**S2**). These three datasets represent distinct domains due to their differences in sampling rates and EEG channels.

6.2.1.1 Sleep-EDF (EDF)

The first dataset is the Sleep-EDF dataset, which was discussed earlier in Section 3.2.1 of Chapter 3.

6.2.1.2 SHHS-1 (S1)

We also adopted a subset of the SHHS-1 dataset, which was discussed earlier in Section 3.2.1 of Chapter 3.

[2]https://physionet.org/physiobank/database/sleep-edfx/
[3]https://sleepdata.org/datasets/shhs

TABLE 6.1: A brief description about the datasets.

Dataset	#Subjects	EEG Channel	Sampling Rate	#Train	#Val	#Test
EDF	20	Fpz-Cz	100	25,612	7,786	8,910
S1	42	C4-A1	125	24,515	7,948	9,067
S2	44	C4-A1	250	31,613	9,769	12,413

6.2.1.3 SHHS-2 (S2)

The data of S2 represents the polysomnogram recordings of the second visit of 3,295 of the participants in S1. The outcome data were used to adjust the parent cohort. Each PSG file in both S1 and S2 datasets contains data from two EEG channels namely C4-A1 and C3-A2, where we only adopt C4-A1 channel recordings for both datasets. We selected subjects from S1 and S2 datasets such that 1) they contain different patients, 2) subjects from the S2 dataset have a sampling rate of 250 Hz, and 3) the subjects have Apnea Hypopnea Index (AHI) < 1 to eliminate the bias to sleep disorders and ensure a consistent clinical status of subjects [157]. Notably, we down-sampled the data from S1 and S2 datasets such that the sequence length is the same as the EDF dataset, i.e., 30 seconds $\times$ 100 Hz ($T = 3,000$).

We preprocessed the three datasets by 1) merging stages N3 and N4 into one stage (N3) according to AASM standard, and 2) including only 30 minutes of wake stage periods before and after sleep [11]. Table 6.1 shows a summary of the above three datasets before down-sampling describing the number of subjects (#Subjects) in each cross-domain, the selected EEG channel, the sampling rate, and the number of training (#Train), validation (#Val), and testing (#Test) samples in each domain.

6.2.2 Feature Extractor

To extract the features from the EEG signals, we first preprocess the EEG signals to be split into 30-second segments (i.e., epochs). Each epoch is then passed through our feature extractor network to extract the features. We used the same design of our previous feature extractor, which was discussed earlier in Section 4.2.2.1. The 1D-convolution layer in the first block has 32 filters, with a kernel size of 25 and a stride of 6. The 1D-convolution layer in the second and third layers have 64 and 128 filters respectively and both have a kernel size of 8 and a stride of 1.

The features extracted from these three blocks are then sent to the self-attention mechanism.

6.2.3 Experimental Settings

To evaluate the performance of our model and baseline models, we employed the classification accuracy (ACC) (Eq. 3.13 and the macro-averaged F1-score (MF1) (Eq. 3.14). All the experiments were repeated 5 times with different random seeds for model initialization, and then we reported the average performance (i.e., ACC and MF1) with standard deviation.

We performed *subject-wise* splits for the data from the three domains, i.e., we split them into 60%, 20%, and 20% for training, validation, and testing, respectively, such that the data from one subject were assigned to either of the 3 splits. We used the training part of source and target domains while training our model. We used the validation part and test part of the target domain for validation and testing. Following [104, 182], we used the validation split of the target domain to select the best hyperparameters in our model. We tuned the parameters λ_1, λ_2 in the range $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1\}$, and set their values as $\lambda_1 = 0.01$ and $\lambda_2 = 0.001$. For iterative self-training, we set the maximum iterations r to 2, as the performance of the model was found to converge. We used Adam optimizer with a learning rate of 1e-3 that is decayed by 0.1 after 10 epochs, weight decay of 3e-4, $\beta_1 = 0.5$, $\beta_2 = 0.99$, and a batch size of 128. We trained the model for a predetermined number of epochs (15 epochs in our case) per iteration. All the experiments were performed with PyTorch 1.7 on NVIDIA GeForce RTX 2080 Ti GPU. The source code is available at https://github.com/emadeldeen24/ADAST.

6.2.4 Baselines

To assess our proposed ADAST model, we compared it against various baselines. We first included the Direct Transfer (DT) results of three sleep staging methods. These methods are **DeepSleepNet** [11]; **SleepEEGNet** [135]; and our **AttnSleep** [144]. In addition, we adopted seven state-of-the-art discrepancy- and adversarial-based domain adaptation (DA) baselines. In particular, Deep CORAL, MDDA, and DSAN are discrepancy-based methods, while DANN, ADDA, CDAN,

and DIRT-T are adversarial-based methods. These baselines are summarized as follows.

- **Deep CORAL** [183]: it extends CORAL [94] to learn a nonlinear transformation that aligns the correlations of layer activations in deep neural networks.
- **MDDA** [184]: it applies MMD and CORAL on multiple classification layers to minimize the discrepancy between the source and target domains.
- **DSAN** [101]: it incorporates a local MMD loss to align the same-class subdomain distributions.
- **DANN** [103]: it jointly trains the feature extractor and domain classifier by negating the gradient from the domain classifier with a gradient reversal layer (GRL).
- **ADDA** [95]: it performs a similar operation as DANN but by inverting the labels instead of using GRL.
- **CDAN** [105]: it minimizes the cross-covariance between feature representations and classifier predictions.
- **DIRT-T** [104]: it combines virtual adversarial domain adaptation with a teacher model to refine the decision boundary of the target domain.

Notably, we included the results of the **Source-Only** experiment, which refers to the DT results of our backbone network. In addition, we used our backbone feature extractor for all seven DA baselines to ensure a fair comparison. We tuned the hyperparameters of the baselines to achieve their best performance.

6.2.5 Experimental Results

Table 6.2 shows the comparison results among various methods. Overall, direct transfer usually achieves lower performance than domain adaptation. The results of DT experiments on [11, 135, 144] indicate that the domain shift problem causes a big performance drop and should be addressed separately. Therefore, it becomes

TABLE 6.2: Comparison against various baselines. Best results are in bold, and the second best are underlined.

		Cross-Domain Accuracy						AVG
	Baselines	EDF→S1	EDF→S2	S1→EDF	S1→S2	S2→EDF	S2→S1	ACC
DT	DeepSleepNet [11]	49.19±3.09	48.48±2.74	76.37±0.11	61.53±0.82	62.45±0.26	**78.97±0.11**	62.83
DT	SleepEEGNet [135]	62.51±3.31	49.82±3.22	56.21±0.46	59.41±2.70	62.98±1.34	69.96±2.00	60.14
DT	AttnSleep [144]	64.44±3.37	57.43±5.45	75.41±0.65	72.08±0.23	66.59±0.83	77.52±0.29	68.91
DT	Source-Only *(Ours)*	57.76±2.40	52.05±4.47	75.05±1.43	63.84±3.83	59.65±2.90	73.94±1.48	63.72
DA	Deep CORAL [183]	63.92±2.35	53.24±4.19	75.95±4.53	61.49±2.43	68.90±3.69	75.55±1.64	66.51
DA	MDDA [184]	66.18±0.73	59.22±4.34	74.97±1.39	65.96±4.57	69.93±2.42	75.18±2.43	68.57
DA	DSAN [101]	65.45±0.61	**69.58±1.62**	**82.30±0.31**	67.19±2.11	70.89±1.01	75.80±1.62	71.87
DA	DANN [103]	65.93±0.63	58.67±3.44	77.40±0.54	64.14±0.48	67.53±2.60	73.72±0.87	67.90
DA	ADDA [95]	68.02±1.07	52.89±6.64	80.73±1.50	57.85±0.72	72.65±0.28	71.63±1.10	67.30
DA	CDAN [105]	64.15±2.41	64.09±1.27	78.02±0.84	66.06±2.29	72.13±2.13	76.42±1.88	70.14
DA	DIRT-T [104]	66.51±2.99	59.20±4.34	79.98±0.35	65.06±3.98	72.95±3.27	77.16±0.61	70.19
DA	ADAST *(Ours)*	**75.50±1.03**	67.56±2.37	75.94±1.25	**72.27±1.06**	**75.28±1.78**	77.80±0.25	**74.00**
		Cross-Domain F1-score						MF1
DT	DeepSleepNet [11]	42.43±2.85	39.93±2.37	64.58±0.44	55.02±0.61	53.22±0.22	**66.65±0.65**	53.63
DT	SleepEEGNet [135]	55.29±3.09	40.68±4.63	51.97±0.77	55.10±1.10	54.74±1.42	61.04±2.98	53.13
DT	AttnSleep [144]	54.77±2.36	50.06±3.29	63.03±0.66	**62.02±0.69**	56.41±0.82	63.71±0.12	58.33
DT	Source-Only *(Ours)*	46.04±1.86	42.22±4.07	63.07±1.34	54.96±3.12	49.10±2.55	62.71±3.00	53.02
DA	Deep CORAL [183]	55.94±2.21	42.75±5.82	62.36±3.99	50.68±2.34	57.35±2.81	61.84±1.41	55.16
DA	MDDA [184]	56.00±0.53	48.07±1.88	62.18±1.01	54.09±2.83	57.43±2.67	60.94±1.88	56.45
DA	DSAN [101]	55.67±0.43	**55.07±1.19**	67.78±0.28	55.55±1.16	58.79±1.04	62.20±0.93	59.18
DA	DANN [103]	54.79±0.54	48.49±2.57	63.86±0.69	53.48±0.36	57.14±2.08	60.17±0.65	56.32
DA	ADDA [95]	58.18±1.41	43.96±6.26	**68.43±0.88**	48.53±0.64	59.02±0.54	58.54±0.78	56.11
DA	CDAN [105]	52.61±2.31	52.42±0.33	64.31±0.93	54.73±1.45	59.70±2.18	62.76±1.79	57.75
DA	DIRT-T [104]	55.34±2.81	48.31±3.81	66.12±0.45	54.55±2.85	57.72±4.61	62.05±0.72	57.35
DA	ADAST *(Ours)*	**61.92±0.83**	53.80±2.11	63.33±1.02	58.69±0.60	**62.49±1.04**	63.10±0.06	**60.39**

important to use domain adaptation to address the domain shift problem for cross-dataset sleep stage classification, which is supported by the results of the other seven DA baselines.

It should be highlighted that the performance increase in the seven DA baselines should be compared to our Source-Only DT results, as they share the same backbone network. We noticed that the three methods considering the class-conditional distribution, i.e., CDAN, DIRT-T, and DSAN, outperform the ones globally aligning the source and target domains, i.e., DANN, Deep CORAL, ADDA, and MDDA. This indicates that considering class distribution, especially in the case of imbalanced sleep data, is important to achieve better classification performance on the target domain. Our proposed ADAST achieves superior performance over all the baselines in terms of both mean accuracy and F1-score in four out of six cross-domain scenarios for two reasons. First, our ADAST, similar to CDAN, DIRT-T, and DSAN, also considers the class-conditional distribution. In particular, ADAST explores the target domain classes using the proposed iterative self-training strategy with dual classifiers. Second, ADAST preserves domain-specific features using

TABLE 6.3: Ablation study showing the different variants of our proposed ADAST framework. **ATT**: domain-specific ATTention, **DC**: Dual Classifiers, **ST**: Self Training. The first row indicates using only the domain discriminator along with a single classifier.

Component			Cross-Domain Accuracy						AVG
ATT	DC	ST	EDF→S1	EDF→S2	S1→EDF	S1→S2	S2→EDF	S2→S1	ACC
-	-	-	66.46±1.86	62.72±4.14	**79.10±1.23**	61.54±0.08	72.28±3.33	73.19±0.64	69.21
✓	-	-	71.31±1.78	**69.07±1.50**	76.83±1.83	63.98±3.11	74.59±0.93	75.19±1.63	71.50
✓	✓	-	72.53±0.88	66.50±2.45	78.22±0.48	68.54±5.29	**75.57±0.86**	76.98±0.54	73.05
✓	-	✓	70.76±2.22	68.79±0.20	77.61±0.79	68.58±5.14	74.05±1.73	75.82±0.67	72.60
✓	✓	✓	**75.50±1.03**	67.56±2.37	75.54±1.25	**72.27±1.06**	75.28±1.78	**77.80±0.25**	**74.00**

the unshared attention module, which improves performance.

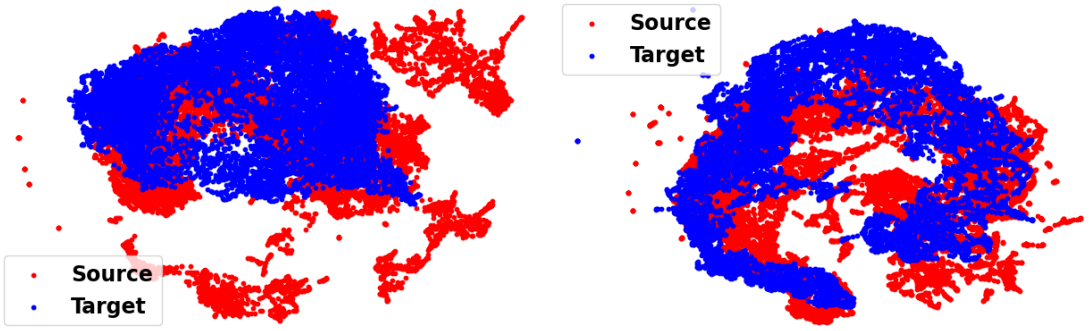
As shown in Table 6.2, the performance of our model is less than most baselines in the scenario S1→EDF. Note that we used the same value of λ_1 (i.e., 0.01) for all six scenarios, which might not be fair for some scenarios. We found that the quality of the pseudo labels is not good in this scenario S1→EDF, and thus we should use a smaller λ_1 to reduce the contribution of the target classification loss. By tuning λ_1 from 0.01 to 10^{-6}, the mean accuracy and MF1 of our ADAST in the scenario S1→EDF would increase from 75.94% and 63.33% to 78.50% and 64.73%, respectively.

We also observed interesting results while investigating different cross-dataset scenarios. Various methods usually achieve better performance in the cross-domain scenario S1→S2 than EDF→S2 (and similarly S2→S1 is better than EDF→S1). To explain this, as shown in Table 6.1, S1 and S2 are closer to each other, as they have the same EEG channel. Meanwhile, EDF has a different EEG channel and sampling rate, and thus it is a distant domain from S1 and S2. These results indicate that distant domain adaptation is still very challenging. Finally, we observed that S1→EDF is easier than S2→EDF, probably because S1 and EDF have close sampling rates to each other.

6.2.6 Ablation Study

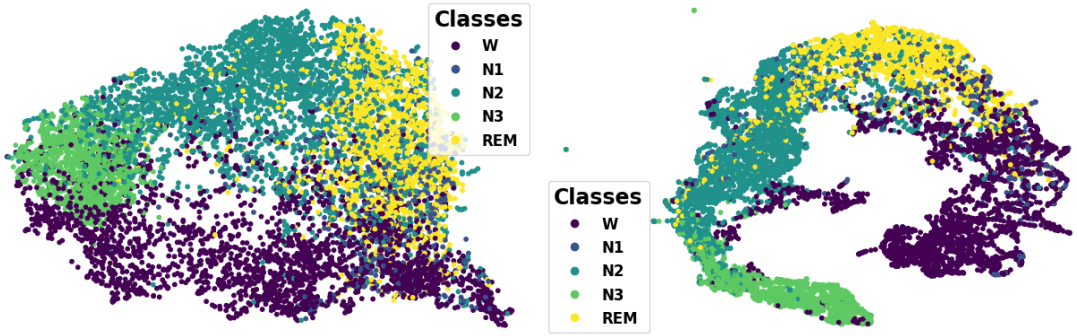
We assessed the contribution of each component in our ADAST framework, namely the unshared domain-specific attention module (**ATT**), the dual classifiers (**DC**), and self-training (**ST**). Particularly, we conducted an ablation study to show the results of different variants of ADAST in Table 6.3.

The results emphasize three main conclusions. First, using the proposed domain-specific attention benefits the overall performance, as it helps to preserve the domain-specific features. Second, the self-training improves the classification performance by $\sim$ 1.1%. This improvement shows the benefit of incorporating the target domain class information in modifying the classification boundaries by using pseudo labels. Third, the addition of dual classifiers benefits the classification performance in overall as it avoids the variance in the training data. Moreover, combining it with the self-training in specific is helpful to further improve the performance by 2.5% through improving the quality of the pseudo labels.



(a) (b)

FIGURE 6.3: UMAP feature space visualization showing the source and target domains alignment using (a) Source-Only, and (b) our ADAST, applied for the scenario S2→EDF.



(a) (b)

FIGURE 6.4: UMAP feature space visualization showing the target domains classification performance after (a) Source-Only, and (b) our ADAST alignment, applied for the scenario S2→EDF.

6.2.7 Representation Visualization

In Section 6.2.5, the results illustrate the advantages of our proposed ADAST framework over the initial Source-Only performance. To make the comparison more intuitive, we visualized the feature representations that were learned during the training process using Uniform Manifold Approximation and Projection (UMAP) [185].

First, we investigated the alignment quality, where Fig. 6.3 visualizes the source and target alignment in the scenario S2→EDF. In particular, Fig. 6.3(a) shows the Source-Only alignment, and Fig. 6.3(b) shows our ADAST framework alignment. In these figures, the red dots represent the source domain, and the blue dots denote the target domain. We can observe that the Source-Only is not very efficient as many disjoint patches are not well-aligned with the target domain. However, our ADAST framework improves the alignment of the two domains to become arc-shaped, which increases the overlapped region and they become less discriminative.

Additionally, we investigated the target domain classification performance in the aforementioned scenario after the alignment in Fig. 6.4. In particular, Fig. 6.4(a) is the Source-Only performance, and Fig. 6.4(b) is the one after our alignment. We noticed that the Source-Only alignment generates a lot of overlapping samples from different classes, which degrades the target domain classification performance. On the other hand, our ADAST framework improves the discrimination between the classes and they become more distinct from each other. This is achieved with the aid of the iterative self-training strategy.

6.2.8 Domain-Specific vs Domain-Invariant Features

In this subsection, we show that domain-invariant features have been extracted in two different manners.

Quantitatively: To have a quantitative insight into extracting domain-specific information, we use the Kullback-Leibler (KL) divergence as a distribution similarity measure. In general, KL-divergence is a non-symmetric measure of the difference between two probability distributions $p(x)$ and $q(x)$. Specifically, the KL-divergence of $q(x)$ from $p(x)$, denoted as $D_{KL}(p(x), q(x))$, is a measure of the

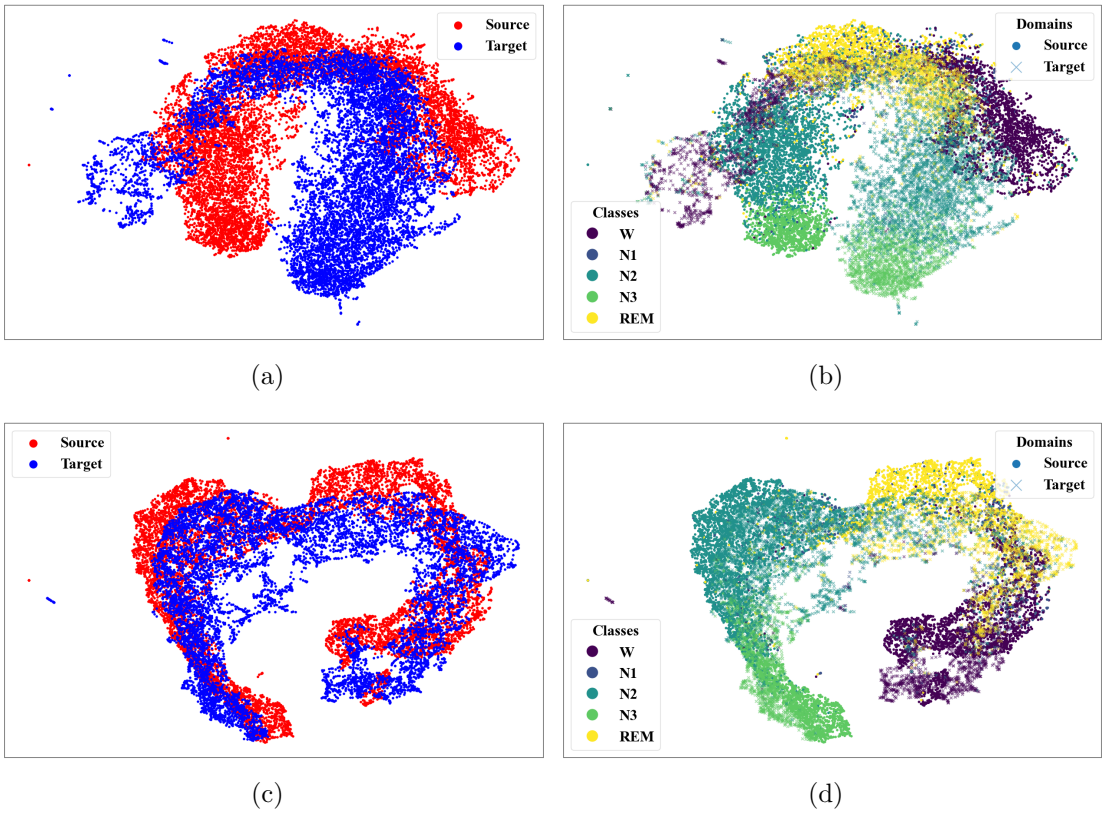


FIGURE 6.5: UMAP feature space visualization showing the domain-invariant alignment (a) domain-wise, and (b) class-wise, besides the domain-specific alignment (c) domain-wise, and (d) class-wise under the scenario of EDF→S1.

information lost when $q(x)$ is used to approximate $p(x)$ [186]. So, if we can approximate $p(x)$ with $q(x)$ without a big loss in information (achieve a low KL divergence value), then we can conclude more similarity between the two distributions, and vice versa.

To validate this idea, we first extract the reference features for each domain. To do so, we use the feature extractor network and the dual classifiers to train the source domain data with the cross-entropy loss and freeze the extracted features from the last epoch (i.e., R_s). Since we do not have access to the target domain labels, we use the pretrained model (on source domain data) to obtain the target domain features (i.e., R_t).

Next, we train the original model with source and target data to obtain the domain-invariant (DI) and domain-specific (DS) features before and after the domain-specific attention module respectively. Finally, we calculate the following KL-divergence on the source domain sides as $D_{KL}(DI_s, R_s)$ and $D_{KL}(DS_s, R_s)$. Similarly, we calculate the following KL-divergence on the target domain sides as $D_{KL}(DI_t, R_t)$ and $D_{KL}(DS_t, R_t)$.

We perform this experiment on the cross-domain scenario (EDF→S1) and provide the results in Table 6.4. Notably, the KL-divergence value is the minimum between the domain-specific features and their corresponding original domain features. This indicates a minimal information loss when trying to approximate the domain-specific features using the reference features, i.e., more similarity between them. Therefore, we conclude the efficacy of the domain-specific attention module in preserving the unique characteristics of each domain.

TABLE 6.4: KL-divergence between original features (R) and domain-invariant (DI) and domain-specific (DS) features applied for both source and target domains.

		Domain-Invariant (DI)	Domain-Specific (DS)
Source domain	R_s	0.7591	0.7346
Target domain	R_t	0.7849	0.6735

Visual Inspection To analyze the feature space of both domain-invariant and domain-specific features, we visualize their distribution in the scenario (EDF→S1) with UMAP, as shown in Fig. 6.5. We notice that the feature extractor can extract domain-invariant information by minimizing the distance between source and target distributions as shown in Fig. 6.5(a). However, these domain-invariant features still misclassify some classes as seen in Fig. 6.5(b). Meanwhile, the domain-specific attention helps to better align the domains (Fig. 6.5(c)), as well as improving the class-wise alignment as in Fig. 6.5(d). In addition, the wake class (Magenta) is now closer to the Rapid Eye Movement (REM) class (Yellow), which is reasonable since both classes share related information. Similarly, the points of the N3 class become closer to the N2 class. This implies that our model well learns specific features that can be adapted to the new unseen domain.

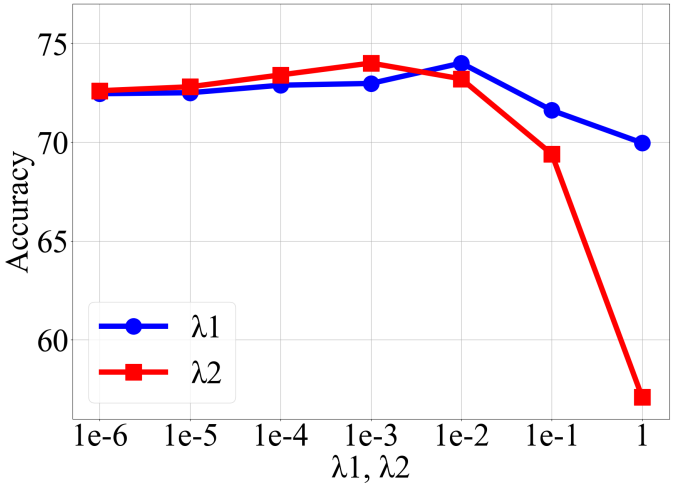


FIGURE 6.6: Sensitivity analysis to the different variants of λ_1 and λ_2 in Eq.6.10.

6.2.9 Sensitivity Analysis

Effect of target classification loss. Since the self-training process relies on target domain pseudo labels, it is not practical to assign a high weight to the target classification loss as the pseudo labels are expected to have some uncertainties. Therefore, we studied the effect of the different variants on the weight assigned to the target classification loss λ_1, as shown in Fig. 6.6.

Notably, when λ_1 is very small (i.e., λ_1 = 1e-6), it makes the self-training useless, and the performance becomes very close to the case without self-training. As we gradually increase λ_1 value, we notice an improvement in the overall performance until we reach the optimal value of $\lambda_1 = 0.01$. Further increasing λ_1 deteriorates the performance as the model is highly penalized based on the pseudo labels which may contain false examples.

Effect of classifier weight constraint. Since the dual classifiers share the same architecture, it is important to keep their predictions relatively different but not with a big gap. The classifier weight constraint is the factor that keeps this distance with an acceptable margin, and hence, it becomes important to study the effect of this term and how its weight λ_2 should be selected. We analyzed the performance of our model with different λ_2 values, as illustrated in Fig. 6.6.

When λ_2 is very small, it makes the two classifiers perform very closely to each other, which has a similar performance with a single classifier. The performance is gradually improved when increasing λ_2, as the two classifiers tend to have different

classification decisions. It can be found that the best performance is achieved with $\lambda_2 = 0.001$. However, as its value is increased beyond this threshold (i.e., 0.001), we notice that the overall performance degrades. This happens as the weights of the two classifiers become very dissimilar, moving them away from the correct predictions.

6.3 Summary

In this chapter, we proposed a novel adversarial domain adaptation architecture, i.e., ADAST applied for cross-domain sleep stage classification. We tackle the problem of the domain shift that happens when training the model on one dataset (i.e., the source domain), and testing it on another out-of-distribution dataset (i.e., the target domain). We developed unshared attention mechanisms to preserve domain-specific features. We also proposed a dual classifier-based iterative self-training strategy, which helps the model adapt the classification boundaries according to the target domain with robust pseudo labels. The experiments performed on six cross-domain scenarios generated from three public datasets prove that our model can achieve superior performance over state-of-the-art domain adaptation methods. Additionally, the ablation study shows that the dual classifier-based self-training is the main contributor to the improvement as it considers class-conditional distribution in the target domain.

Chapter 7

Contrastive Domain Adaptation for Time Series via Temporal Mixup

Previous unsupervised domain adaptation works have followed two mainstreams. The first is to leverage a statistical distance to minimize the discrepancy between source and target domains. The second is to utilize adversarial training to mitigate the domain shift, which was our approach in Chapter 6. However, these two approaches have some limitations. First, they ignore the temporal dependencies in time series data while matching the source and target distributions, leading to sub-optimal adaptation performance. Second, most of the existing discrepancy-based approaches depend on reducing a distance measure, which may struggle to align distributions with large domain shifts [113]. Third, the adversarial-based approaches are usually complex to train and rely on minimax optimization, which is hard to converge to a satisfactory local optimum [187]. Last, both paradigms attempt to directly adapt the target domain distribution towards the source domain using the source domain knowledge, which can be less effective when aligning distant domains [188].

Therefore, in this chapter [1], we develop a new direction and propose a novel framework (CoTMix) that exploits contrastive learning to mitigate the domain shift in time series data. In particular, we develop a novel cross-domain temporal mixup

[1]The work in this chapter has been published as "Contrastive Domain Adaptation for Time-Series via Temporal Mixup", IEEE Transactions on Artificial Intelligence (TAI) [189].

strategy to generate two new intermediate domains namely the source-dominant and the target-dominant domains. These two intermediate domains act as augmented views for the source and target domains in contrastive learning. Moreover, they are designed in a way that preserves the semantics of the dominant domain while learning the temporal characteristics of the less dominant domain. Subsequently, we leverage in-domain contrastive learning to maximize the similarity between the source and the source-dominant domains, as well as maximizing the similarity between the target and target-dominant domains.

Notably, our method stands apart from existing methods that incorporate data or feature augmentations in complex adversarial training [190, 191]. Unlike these works that directly push the target domain towards the source domain, our proposed approach can progressively map the source and target domains towards an intermediate domain.

To summarize, our main contributions are as follows:

- We propose CoTMix, a novel contrastive learning-based framework for time series UDA. To the best of our knowledge, this is the first work that utilizes contrastive learning *solely* for adaptation without any adversarial training.
- We propose a cross-domain temporal mixup strategy that generates new augmented views for in-domain contrastive learning at both source and target domains sides. This operation aims to fit contrastive learning to serve the adaptation objective. In addition, the proposed strategy is generic and can be applied to any time series data.
- We conduct extensive experiments on four real-world time series domain adaptation datasets. The results show that our CoTMix significantly outperforms state-of-the-art UDA methods.

7.1 Proposed Method

7.1.1 Problem Definition

We address the problem of unsupervised domain adaptation for time series data. Specifically, we have a labeled source domain $X_s = \{(\mathbf{x}_s^i, y_s^i)\}_{i=1}^{n_s}$ with n_s samples,

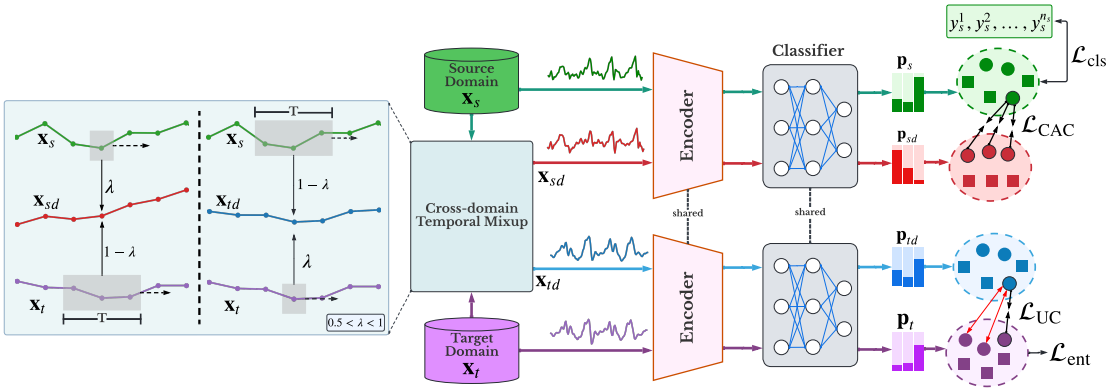


FIGURE 7.1: The overall structure of our CoTMix framework. The cross-domain temporal mixup strategy generates the source-dominant $\mathbf{x}_{sd}$, and the target-dominant $\mathbf{x}_{td}$ domains. We use a fixed $0.5 < \lambda < 1$ to ensure that one domain has more contribution in the generated samples. In addition, we aggregate T timesteps from the less-dominant domain while the mixup to learn its temporal information. The generated domains act as augmented views for the in-domain contrastive learning, which is performed on the output probabilities.

and an unlabeled target domain $X_t = \{\mathbf{x}_t^j\}_{j=1}^{n_t}$ with n_t samples. Both domains have samples with a length of L timesteps and share the same label space, i.e., $y_s^i, y_t^j \in \{1, 2, \ldots K\}$, where K denotes the number of classes. It is assumed that there is a distribution shift between the two domains (i.e., $P(\mathbf{x}_s) \neq P(\mathbf{x}_t)$). Given the source and target data, we aim to train a shared model that consists of a feature encoder $F(\cdot)$ and a classifier $C(\cdot)$ to find a unified space that can successfully classify the unlabeled target data.

7.1.2 Overview

In this section, we propose CoTMix, a **Co**ntrastive domain adaptation framework via **T**emporal **Mix**up for time series data. CoTMix consists of two main components: the cross-domain temporal mixup strategy and the in-domain contrasting at the source and target sides. Fig. 7.1 illustrates the overall structure of our proposed approach, which can be trained in an end-to-end manner.

7.1.3 Temporal Mixup

We propose a cross-domain temporal mixup strategy, in which we generate two new intermediate domains namely the source-dominant and the target-dominant

domains using the mixup operation [192]. Each of these domains should preserve the characteristics of one *dominant* domain while considering the temporal information from the other *less-dominant* domain. To do so, unlike the traditional mixup, we use a fixed mixup ratio $0.5 < \lambda < 1$, such that one domain will have more contribution than the other in the newly generated domain. In addition, we learn the temporal information from the less-dominant domain by aggregating multiple forward and backward timesteps to be mixed with one timestep from the dominant domain, as illustrated in Fig. 7.1. For instance, we generate the source-dominant samples by mixing each timestep from the source domain with the average value of T timesteps from the target domain ($\frac{T}{2}$ backward timesteps and $\frac{T}{2}$ forward timesteps), such that the source ratio is λ and the target ratio is $1 - \lambda$. We calculated the mean of timesteps as inspired by the moving average method [193]. Averaging timesteps has the advantages of eliminating short-term fluctuations and reducing the effect of extreme values.

Formally, given a source domain sample x_s and a target domain sample x_t, we generate each timestep i in the source-dominant domain as follows:

$$x_{sd}^{i} = \lambda x_{s}^{i} + (1 - \lambda)\frac{1}{T}\sum_{j=i-\frac{T}{2}}^{i+\frac{T}{2}} x_{t}^{j}; \quad 0.5 < \lambda < 1 \tag{7.1}$$

where $\mathbf{x}_{sd} = (x_{sd}^{1}, x_{sd}^{2}, \ldots, x_{sd}^{L})$ represents the generated source-dominant sample, T is the mixup window length, and L is the sample length. A similar process is followed to generate the target-dominant samples, which can be formalized as follows:

$$x_{td}^{i} = \lambda x_{t}^{i} + (1 - \lambda)\frac{1}{T}\sum_{j=i-\frac{T}{2}}^{i+\frac{T}{2}} x_{s}^{j}, \quad 0.5 < \lambda < 1 \tag{7.2}$$

where $\mathbf{x}_{td} = (x_{td}^{1}, x_{td}^{2}, \ldots, x_{td}^{L})$ represents the generated target-dominant sample.

7.1.4 Contrastive Adaptation

Given the generated source-dominant and target-dominant mixed domains, in addition to the original source and target domains, we use the feature encoder and the classifier to generate the probability vectors for the four domains. As inspired

by [194], we leverage the probability vectors in the InfoNCE loss [53] to maximize the similarity between each domain and its corresponding intermediate view. Since we contrast each domain with its dominant mixed domain, we benefit from several advantages. First, we close up the gap between the two domains regardless of the shift distance, because the model keeps learning about the less-dominant domain on both sides *progressively* throughout training. Second, in addition to mitigating the domain shift, in-domain contrastive learning improves the learning capability of the model about each domain separately. Last but not least, this approach is simpler in implementation and training than traditional complex adversarial training approaches.

Fig. 7.1 shows the overall structure of our framework. For the source domain side, we minimize the class-aware contrastive loss as well as the source classification loss. For the target domain side, we minimize the unsupervised contrastive loss and the entropy minimization loss. Next, we will discuss the losses on each side in more detail.

7.1.4.1 Source Domain Side.

Since we have access to the source domain labels, we leverage these labels to optimize both the in-domain class-aware contrastive loss and the standard cross-entropy loss. The class-aware contrastive learning, as inspired by [195], benefits from the available labeled data to include more positive pairs in the contrastive loss. Specifically, for each anchor sample, we consider all the samples having the same class label within the mini-batch as positive pairs. In this way, we consider the semantic information between samples and avoid contrasting against false negatives, which could improve the quality of the learned representations. Moreover, by contrasting with multiple possible positive pairs (which are mixed with target domain data), we increase the chance of narrowing the gap between the anchor sample with samples having the same class label in the less-dominant domain (i.e., target domain), which further improves the class-wise alignment in the target domain.

Formally, given the n_s source domain samples and the n_s generated source-dominant samples, the overall samples become $2n_s$. We generate the output probabilities $\mathbf{p}_s = C(F(\mathbf{x}_s))$ and $\mathbf{p}_{sd} = C(F(\mathbf{x}_{sd}))$. To this end, the overall source samples become $(\{\mathbf{x}^l_{so}, y^l_{so}\}_{l=1...2n_s})$, and their corresponding probabilities are $\mathbf{p}_{so}$. In addition,

we assume that the class label is the same for any two corresponding samples from both domains, i.e., $y_s^i = y_{sd}^i$. Assuming that $k \in I \equiv \{1 \dots 2n_s\}$ represents the index of an arbitrary sample (from either source or source-dominant domains), and $A(k) \equiv I \setminus \{k\}$. The set of indices of all samples with the same class as an anchor sample $\mathbf{x}_{so}^k$ will be $U(k) = \{u \in A(k) : y_{so}^u = y_{so}^k\}$. Therefore, we can formulate the probabilistic class-aware contrasting loss $\mathcal{L}_{\text{CAC}}$ as follows.

$$\mathcal{L}_{\text{CAC}} = \sum_{k \in I} \frac{-1}{|U(k)|} \sum_{u \in U(k)} \log \frac{\exp\left(\mathbf{p}_{so}^k \cdot \mathbf{p}_{so}^u / \tau\right)}{\sum_{a \in A(k)} \exp\left(\mathbf{p}_{so}^k \cdot \mathbf{p}_{so}^a / \tau\right)}, \tag{7.3}$$

where $\cdot$ symbol denotes the inner dot product, τ is a temperature parameter, and $|U(k)|$ is the cardinality of $U(k)$.

In addition to the class-aware contrasting, we also train the model to minimize the cross-entropy loss as follows.

$$\mathcal{L}_{\text{cls}} = -\mathbb{E}_{(\mathbf{x}_s, y_s) \sim P_s} \sum_{k=1}^{K} \mathbb{1}_{[y_s = k]} \log \mathbf{p}_s(k). \tag{7.4}$$

7.1.4.2 Target Domain Side.

Considering the target domain, we do not have access to its label information. Therefore, we can only contrast its samples in an unsupervised manner. Given the n_t target domain samples and its corresponding n_t target-dominant mixed domain samples, the total number of samples becomes $2n_t$. We calculate the output probabilities $\mathbf{p}_t = C(F(\mathbf{x}_t))$ and $\mathbf{p}_{td} = C(F(\mathbf{x}_{td}))$. Therefore, the overall target samples become $(\{\mathbf{x}_{to}^l\}_{l=1 \dots 2n_t})$, and their corresponding probabilities are $\mathbf{p}_{to}$, such that for each target sample $\mathbf{x}_t^i$, $1 \leq i \leq n_t$, it forms a positive pair with its corresponding target-dominant sample $\mathbf{x}_{td}^i$ and vice versa. Assuming that $\mathbf{p}_{to} = (\mathbf{p}_t^1, \mathbf{p}_t^2, \dots, \mathbf{p}_t^{n_t}, \mathbf{p}_{td}^1, \dots, \mathbf{p}_{td}^{n_t})$, where $\mathbf{p}_{to}^k = \mathbf{p}_t^k$ if $k \leq n_t$, and $\mathbf{p}_{to}^k = \mathbf{p}_{td}^{k-n_t}$ otherwise. For an anchor sample indexed $k \in I \equiv \{1 \dots 2n_t\}$, $A(k) \equiv I \setminus \{k\}$, we define $f(k)$ as the index of the positive pair of k, such that $f(k) = k + n_t$ if $k \leq n_t$, and $f(k) = k - n_t$ otherwise. Next, we design the probabilistic unsupervised contrastive

loss $\mathcal{L}_{\text{UC}}$ as follows.

$$\mathcal{L}_{\text{UC}} = \frac{-1}{2n_t} \sum_{k \in I} \log \frac{\exp\left(\mathbf{p}_{to}^{k} \cdot \mathbf{p}_{to}^{f(k)} / \tau\right)}{\sum_{a \in A(k)} \exp\left(\mathbf{p}_{to}^{k} \cdot \mathbf{p}_{to}^{a} / \tau\right)}. \tag{7.5}$$

In addition, we minimize the entropy on the unlabeled target domain, formulated as follows.

$$\mathcal{L}_{\text{ent}} = -\mathbb{E}_{\mathbf{x}_t \sim P_t}[\mathbf{p}_t^{\top} \log \mathbf{p}_t]. \tag{7.6}$$

Leveraging this entropy minimization loss forces the classifier to be confident about its prediction for the target domain data [104].

7.1.5 Overall Objective

Our proposed CoTMix is trained with a simple procedure. First, we mix the input signals from the source and target domains to generate the source-dominant and the target-dominant mixed samples as in Equations 7.1 and 7.2, respectively. Next, we develop the class-aware contrastive training in the source domain side, and the unsupervised contrastive training in the target domain side as in Equations 7.3 and 7.5, to minimize $\mathcal{L}_{\text{CAC}}$ and $\mathcal{L}_{\text{UC}}$ respectively. Besides minimizing the contrastive losses, we also minimize the standard cross-entropy loss $\mathcal{L}_{\text{cls}}$ on the labeled source domain, as well as the conditional entropy loss $\mathcal{L}_{\text{ent}}$ on the unlabeled target domain. Overall, the training objective is to minimize these losses combined as follows.

$$\min_{\mathcal{F},\mathcal{C}} \mathcal{L}_{\text{overall}} = \underbrace{\beta_1 \mathcal{L}_{\text{cls}} + \beta_2 \mathcal{L}_{\text{CAC}}}_{\text{Source side}} + \underbrace{\beta_3 \mathcal{L}_{\text{ent}} + \beta_4 \mathcal{L}_{\text{UC}}}_{\text{Target side}}, \tag{7.7}$$

where the hyperparameters β_1, β_2, β_3, and β_4 control the contribution of each loss on the overall performance.

7.2 Experimental Setup

7.2.1 Datasets

To evaluate our proposed approach, we choose four real-world time series datasets in two applications, i.e., sleep stage classification and human activity recognition. These datasets have different characteristics in terms of complexity, the number of samples, the sample length and type, the number of sensors, and the severity of the domain shift. We treat data from each subject as a separate domain, since different subjects may have distinct behaviors, leading to distribution shifts.

The first dataset is **Sleep-EDF** for sleep stage classification, which was discussed earlier in Section 3.2.1 of Chapter 3.

The second dataset is **HAR**, which was also discussed earlier in Section 4.2.1 of Chapter 4.

The third dataset is **HHAR** (Heterogeneity Human Activity Recognition) dataset [196], and it was collected from 9 different users using smartphones and smartwatches. We consider the data from the Samsung smartphone following [197].

Last, the **WISDM** dataset [198] is also for human activity recognition, and it was collected with accelerometer sensors from 36 subjects. The objective in the latter three datasets is to classify sensors' readings into one of six activities, i.e., walking, walking upstairs, downstairs, standing, sitting, and lying down.

Since each dataset contains numerous subjects (i.e., domains), we selected five random scenarios as in [113, 119]. Each scenario represents the ID of the source subject (domain) and the ID of the target subject (domain) within their respective datasets. For example, scenario $16 \rightarrow 1$ in the Sleep-EDF dataset indicates that the source is subject #16 and the target is subject #1. More details about the datasets are illustrated in Table 7.1. We included two large-scale datasets, i.e., Sleep-EDF and HHAR, and two small-scale datasets, i.e., HAR and WISDM datasets. This testifies to the capability of our proposed framework to adapt different scales of datasets/domains.

TABLE 7.1: Details of the adopted datasets (C: #channels, K: #classes, L: sample length).

Dataset	**C**	**K**	**L**	# training samples	# testing samples
Sleep-EDF	1	5	3000	14280	6130
HAR	9	6	128	2300	990
HHAR	3	6	128	12716	5218
WISDM	3	6	128	1350	720

7.2.2 Baselines

We compared our proposed method with seven state-of-the-art UDA methods that span both discrepancy- and adversarial-based schemes as follows:

- **FCN**: Fully Convolutional Network, representing the source-only experiment.
- **HoMM** [100]: Higher-order Moment Matching.
- **DSAN** [101]: Deep Subdomain Adaptation.
- **DANN** [103]: Domain-Adversarial Training of Neural Networks.
- **CDAN** [105]: Conditional Domain Adversarial Network for Adaptation.
- **DIRT-T** [104]: Decision-boundary Iterative Refinement Training with a Teacher.
- **CoDATS** [119]: Convolutional deep Domain Adaptation model for Time Series.
- **AdvSKM** [113]: Adversarial Spectral Kernel Matching.

7.2.3 Implementation Details

7.2.3.1 Dataset preprocessing

We split each domain into 70/30%, where the 70% splits into both domains are for training, while the 30% in the source domain is treated as a validation set for the risk calculation, and the 30% in the target domain acts as a test set. In addition, all the splits were normalized based on the training statistics [119]. We applied a

sliding window of 128 for the three human activity recognition datasets, but for the Sleep-EDF dataset, we kept the original sample length of 3000 timesteps.

7.2.3.2 Feature encoder

We adopted the same convolutional neural network used in our previous works, which was discussed earlier in Section 4.2.2.1. We set this pair as (5,1) for HAR, HHAR, and WISDM datasets as they have the same sequence length. For the Sleep-EDF dataset, we set it as (25,6) due to its longer sequence length. This encoder is followed by a single fully connected layer for classification.

7.2.3.3 Unified training scheme

To ensure a fair evaluation, we unified the way of training, the backbone encoder, the hyperparameters search methodology, and the risk minimization setting for our proposed approach as well as all the baselines, as inspired by AdaTime benchmark [197]. Specifically, we trained all the models for 40 epochs with a batch size of 32 and optimized the neural network weights using the Adam optimizer with a learning rate of 1e-3. The reported results show the average and the standard deviation performance of the last epoch of training, for three repeated experiments with three different seeds.

TABLE 7.2: Selected values of the hyperparameters in the adopted datasets. The first row indicates the range of hyperparameter search. The range of T is specified with respect to L, however, we reported the selected timesteps values.

	β_1	β_2	β_3	β_4	λ	T
Range	[0.1, 1]	[0.001, 1]	[0.001, 1]	[0.001, 1]	[0.5, 1)	[0, 0.5L]
Sleep-EDF	0.96	0.1	0.05	0.1	0.79	150
HAR	0.78	0.1	0.20	0.1	0.90	14
HHAR	0.80	0.1	0.05	0.1	0.52	14
WISDM	0.98	0.1	0.05	0.1	0.72	6

To choose the hyperparameters of our CoTMix, i.e., λ, $\beta_1, \beta_2, \beta_3, \beta_4$, as well as the hyperparameters of the baselines, we performed a hyperparameter sweep with 100 trials. The selection of these hyperparameters was from predefined ranges using uniform sampling. Based on this hyperparameters search, we picked the best model

TABLE 7.3: Detailed results of each cross-domain scenario in the four adopted datasets in terms of MF1 score. CoTMix* indicates deploying unsupervised contrastive loss in the source side. The results are based on minimizing the DEV risk. Best results are in bold, and the second best are underlined.

Dataset	Scenario	FCN	HoMM	DSAN	DANN	CDAN	DIRT-T	CoDATS	AdvSKM	CoTMix*	CoTMix
Sleep-EDF	16→1	52.44±4.78	55.57±2.00	58.76±2.02	58.78±4.76	60.95±1.13	54.4±12.46	60.03±1.18	57.80±0.69	59.85±4.39	**60.97±4.32**
	9→14	64.35±5.66	63.66±1.48	69.45±4.04	64.61±0.93	60.5±10.01	**71.33±3.72**	52.2±10.55	64.27±2.93	65.48±1.45	65.80±3.15
	12→5	56.07±3.07	55.87±2.93	64.92±1.65	**65.47±0.95**	65.01±1.34	64.99±4.98	56.96±2.41	55.12±2.52	64.03±4.47	61.59±1.98
	7→18	59.39±4.94	67.49±1.51	68.69±0.99	68.88±2.81	67.02±1.13	69.94±0.43	68.64±2.93	67.31±3.83	64.79±1.30	**73.34±1.27**
	0→11	43.80±5.83	50.93±4.31	37.43±2.92	31.13±1.74	30.8±10.69	35.62±3.79	41.12±5.14	**55.11±4.56**	48.82±4.60	51.16±3.71
	AVG	55.21	58.70	59.85	57.77	56.86	59.26	55.79	59.92	60.59	**62.57**
HAR	2→11	58.39±3.87	73.38±7.34	75.58±9.18	77.8±18.26	71.51±8.84	88.44±9.23	51.81±4.67	65.74±2.69	**99.01±0.71**	97.17±4.00
	12→16	56.06±2.80	59.84±1.43	61.71±1.75	63.26±2.49	54.66±2.91	58.47±2.98	54.81±2.76	60.09±1.40	66.59±5.53	**77.56±1.34**
	9→18	58.87±5.93	60.0±11.83	67.10±4.61	57.49±7.77	40.94±3.18	65.9±13.25	31.83±8.89	53.70±4.61	68.46±5.64	**75.29±5.62**
	6→23	43.59±8.34	90.48±0.80	93.22±2.49	**95.86±1.84**	61.31±9.02	90.56±8.73	81.23±4.07	79.31±8.95	90.94±2.78	91.74±3.29
	7→13	87.45±6.20	85.94±2.52	88.82±3.08	91.71±0.84	82.1±11.91	**93.73±0.56**	80.9±13.74	88.89±3.12	89.80±1.66	88.47±3.27
	AVG	60.87	78.28	81.07	80.89	64.66	82.54	65.12	74.62	82.96	**86.05**
HHAR	0→6	54.26±3.46	63.58±2.24	58.81±7.19	46.54±0.61	45.52±0.94	52.63±9.77	44.73±1.65	45.52±0.91	66.81±1.79	**69.34±4.66**
	1→6	64.09±4.02	88.49±2.00	**93.42±0.64**	90.73±1.97	92.99±0.70	93.10±2.06	91.98±1.01	92.99±0.72	90.49±0.68	91.97±2.01
	2→7	38.03±4.42	47.12±4.27	45.61±0.51	46.58±3.13	54.12±7.12	63.49±1.95	47.56±5.04	54.11±7.12	62.16±0.41	**68.76±8.54**
	3→8	79.40±1.44	79.23±1.13	**98.44±0.23**	83.4±10.12	98.17±0.37	87.1±10.06	91.83±4.56	98.17±0.37	95.74±0.81	96.01±0.85
	4→5	78.75±4.09	84.07±1.19	**98.47±0.32**	95.83±0.28	96.39±1.37	97.13±0.44	92.52±3.14	96.39±1.37	97.03±0.62	96.61±1.70
	AVG	62.90	72.50	78.95	72.62	77.43	78.69	73.72	77.43	82.45	**84.54**
WISDM	35→31	40.98±7.60	**66.29±0.84**	57.25±6.07	52.21±1.09	49.02±4.20	46.75±3.54	40.96±19.0	61.91±6.95	48.77±3.15	47.68±6.97
	7→18	35.25±4.87	48.67±6.31	52.77±2.23	41.16±6.62	57.65±0.18	57.89±0.15	42.00±3.75	49.84±5.31	56.59±2.87	**74.88±2.75**
	20→30	61.52±2.14	65.28±2.45	63.39±0.70	71.98±10.1	65.50±0.61	65.49±0.62	69.65±7.60	69.35±1.38	**90.64±0.14**	77.90±1.78
	6→19	49.09±4.34	63.78±4.35	53.35±5.37	59.09±3.57	44.03±0.81	45.16±0.00	**70.6±12.51**	54.89±4.14	56.74±8.75	64.27±1.99
	18→23	49.55±9.12	62.11±7.57	55.76±1.46	48.00±0.90	50.16±0.44	50.89±0.40	48.2±15.11	51.3±10.33	**67.99±4.24**	66.87±2.02
	AVG	47.28	61.23	56.51	54.48	53.27	53.24	54.27	57.46	64.15	**66.32**

that minimizes the realistic Deep Embedded Validation (DEV) risk [197, 199]. This risk does not consider any target labels to be calculated. Instead, it considers the highly correlated source features to the target features via importance weighting schemes, which give lower weights to the less correlated features. Despite that choosing the hyperparameters based on this risk may not yield the best performance on the target domain, however, it ensures a fair and realistic evaluation scheme and prevents overestimated results. We included the ranges, as well as the values of the selected hyperparameters, in Table 7.2.

7.3 Results

7.3.1 Comparison with Baselines

Table 7.3 reports the macro-F1 (MF1) scores of our proposed framework against the other competing state-of-the-art methods on the four benchmark datasets. The MF1-score metric is more suitable to reflect the true performance of the imbalanced

time series data. The results show that our proposed CoTMix approach outperforms adversarial and discrepancy baselines significantly in the overall performance across the four datasets, indicating its effectiveness. For the Sleep-EDF dataset, it achieves a 2.65% improvement over the second-best baseline. For HAR, HHAR, and WISDM datasets, it was able to achieve 3.51%, 5.59%, and 5.09% improvement over the second-best method. Since the temporal mixup operation is being performed on the input space, we notice more performance improvement in the human activity recognition datasets, which have less complex time series data, compared to the more complex Sleep-EDF dataset.

Additionally, our CoTMix shows a significant improvement in big domain shifts. For example, in the HHAR dataset, we find that scenario 2→7 suffers a big shift, indicated by the poor source-only performance of 38.03%. We find that CoTMix improved its performance by 30.73% reaching 68.76%.

To test the effectiveness of the class-aware contrastive loss on the source domain side, we added a second variant of our framework (CoTMix*), in which we deploy the unsupervised contrastive loss (Equation 7.5) on the source domain side. We notice that it consistently achieves less performance than CoTMix, however, it always grades the second-best average performance outperforming other baselines. This indicates the efficacy of our method and also shows that considering the semantic information while contrasting helps to improve the class-conditional alignment in the target domain.

7.3.2 Study of Different Augmentations

In this section, we compare the capability of our proposed temporal mixup to mitigate the domain shift against other augmentations proposed for time series representation learning tasks. In specific, we replaced our cross-domain temporal mixup with four different augmentations, i.e., permutation, scaling, jittering [161], and masking [69]. The experimental results are provided in Table 7.4. It can be noticed that our proposed temporal mixup strategy is more effective than other augmentations in the UDA settings. These augmentations may enhance the in-domain representation learning capability of the model, but they do not serve the adaptation objective, i.e., reducing the domain shift. This shows how our temporal mixup contributes to the success of contrastive learning for domain adaptation.

TABLE 7.4: Comparison between deploying different augmentations against our temporal mixup for the in-domain contrastive adaptation. Clearly, other augmentations are not robust to the domain shift, and deploying them in contrastive adaptation yielded relatively less performance that our temporal mixup.

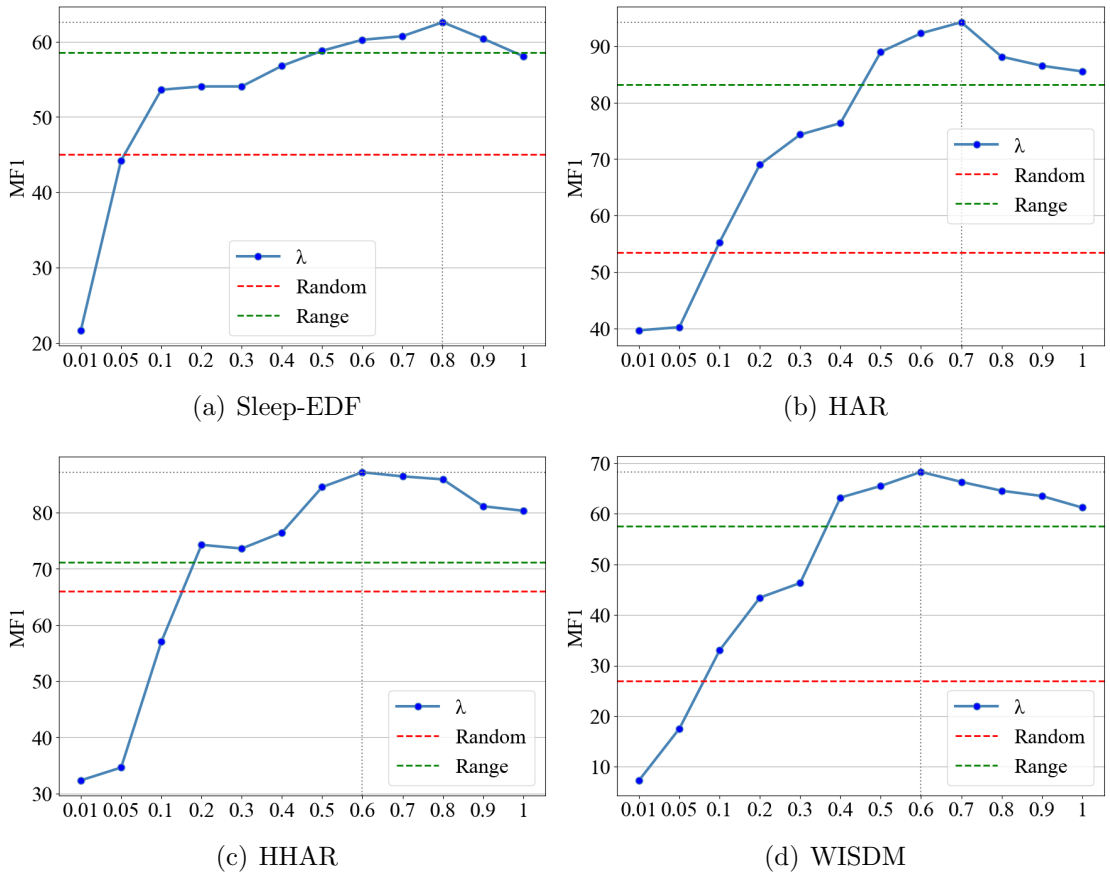
Augmentation	**Sleep-EDF**	**HAR**	**HHAR**	**WISDM**
Permutation	57.15	80.63	76.79	54.67
Scaling	54.64	79.11	71.73	53.11
Jittering	56.53	80.27	75.23	51.04
Masking	56.09	80.44	74.64	55.34
Temporal Mixup	**62.57**	**86.05**	**84.54**	**66.32**

7.3.3 Mixup Strategies

In our temporal mixup, we use a fixed mixup ratio $0.5 < \lambda < 1$ to keep the semantic characteristics of one domain for in-domain contrastive learning. Nevertheless, we compare using this fixed mixup ratio with two different strategies. The first is the random mixup ratio selected randomly from a beta distribution $\lambda \sim Beta(\alpha, \alpha)$ as in the traditional mixup [192]. The second is to specify λ from a range by randomly selecting it from the beta distribution and limiting it to a specific range [200]. In specific, $\lambda' \sim max(\lambda, 1-\lambda)$, where $\lambda \sim Beta(\alpha, \alpha)$.

Fig. 7.2 shows the comparison results for the three scenarios, where we report the average performance of three experiments at each point in the sub-figures. In general, we find that the "Random" mixup strategy causes noticeable performance degradation, as it does not ensure keeping most of the semantics of one dominant domain while contrastive learning. On the other hand, the "Range" mixup strategy keeps the mixup ratio within a range ≥ 0.5, which ensures having a more dominant domain, but with different random ratios. Therefore, it achieves a better performance than the "Random" mixup strategy, but its randomness affects the performance. Finally, we find that using a fixed mixup ratio can achieve the best performance, as it ensures stable ratios of the domains in the augmented views.

Notably, this analysis shows that we may achieve better results than those viewed in Table 7.3. The reason is that these values are selected by minimizing the DEV risk [197, 199]. This risk may not provide the optimal performance on the target data for some datasets, however, it is more realistic in real-world scenarios. In Table 7.5,



(a) Sleep-EDF (b) HAR

(c) HHAR (d) WISDM

FIGURE 7.2: Study of different mixup strategies, as well as different values to our fixed temporal mixup strategy. The red dashed line indicates the average performance when selecting λ randomly from a beta distribution. The green dashed line shows the average performance when selecting λ randomly from a beta distribution but limited to a specific range ≥ 0.5. The dashed lines show the λ values achieving the best performance based on the target risk.

we show that our CoTMix can achieve better performance if hyperparameters were selected based on the target risk, i.e., based on the labels of the target domain.

7.3.4 Ablation Study

Since our CoTMix combines three losses in addition to $\mathcal{L}_{\text{cls}}$, we study the effect of these losses on the overall performance to provide additional insights on what makes CoTMix performant. Table 7.6 presents this ablation study, where $\mathcal{L}_{\text{cls}}$ is present by default in all the cases. In particular, we first omit all three losses, to show the bottom-line performance. Next, we add the entropy minimization

TABLE 7.5: Comparison between the average performance with the realistic DEV risk and the overoptimistic target risk.

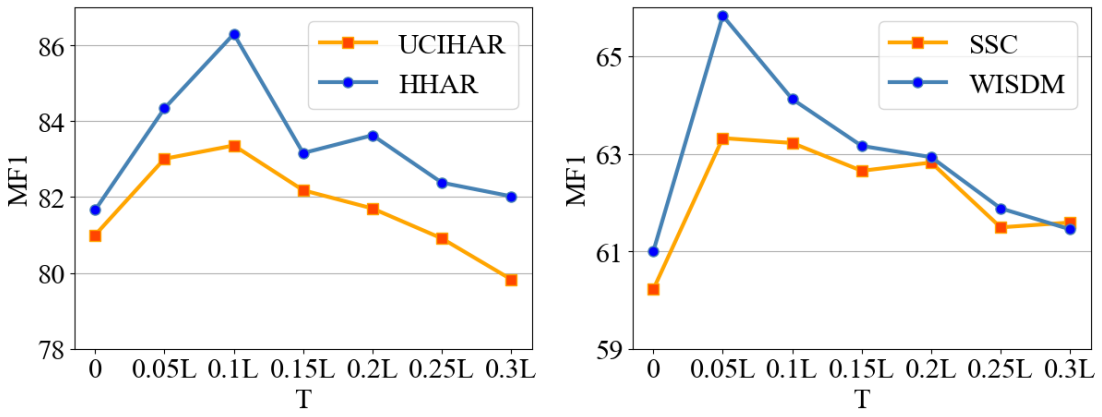
	DEV risk		Target risk	
Dataset	λ	Avg. MF1 Score	λ	Avg. MF1 Score
Sleep-EDF	0.79	62.57	0.79	62.57
HAR	0.90	86.05	0.70	94.26
HHAR	0.52	84.54	0.59	87.18
WISDM	0.72	66.32	0.60	68.30

loss to the training. After that, we apply contrastive training on only one side interchangeably. Finally, we show the results with all the losses together.

TABLE 7.6: Ablation study showing the effect of each loss on the overall performance. By deploying contrastive loss on only one domain, it can still improve the performance. However, we clearly get the best performance by moving both domains towards an intermediate space.

Component			**Dataset**			
$\mathcal{L}_{\text{ent}}$	$\mathcal{L}_{\text{CAC}}$	$\mathcal{L}_{\text{UC}}$	Sleep-EDF	HAR	HHAR	WISDM
-	-	-	51.04	77.30	66.74	52.75
✓	-	-	54.84	79.93	73.14	54.07
✓	✓	-	59.14	83.82	77.34	58.79
✓	-	✓	57.43	80.87	80.11	61.11
✓	✓	✓	**62.57**	**86.05**	**84.54**	**66.32**

We arrive at two conclusions. First, adding entropy minimization improves the overall performance, as it helps the classifier to be more confident about the unlabeled target domain. Second, applying the contrastive loss to only one side still improves the performance. This indicates that moving one domain towards an intermediate domain by considering the cross-domain temporal relations is still effective for adaptation. Moreover, we find that Sleep-EDF and HAR datasets achieve better performance by contrasting only on the source side than contrasting only on the target side. Counterpart, the performance on HHAR and WISDM datasets improved more with contrasting only on the target side than the source side. This can be regarded to the efficacy of the learned temporal features from one side over the other for adapting the two domains. This experiment also shows that moving only one domain towards the intermediate domain may not be the most effective way. Nevertheless, the performance is consistently the best when contrastive losses are applied on both sides, i.e., moving both domains.



(a) HAR and HHHAR

(b) Sleep-EDF and WISDM

FIGURE 7.3: Sensitivity analysis applied on the four datasets to study the effect of the number of timesteps T, as a percentage of the signal length L, on the performance. We notice a progressive performance improvement with including more timesteps in the temporal mixup operation until T approaches $0.1L$. (We merged datasets with close performance in one figure)

7.3.5 Selection of Temporal Mixup Window

Our proposed temporal mixup could be affected by the length of aggregated timesteps T in the less-dominant domain. Therefore, we study its significance on the performance, as shown in Fig. 7.3. To avoid random selection to the value of T, we assign it as a proportion in the sample length L. The first case in the analysis, (i.e., $T = 0$) represents a one-to-one mixup of timesteps among source and target domains, while in the next cases, we gradually increase T.

First, we notice that when $T = 0$, which corresponds to the traditional mixup [192], the performance usually drops. This shows the significance of our temporal mixup strategy, as it significantly improves the performance in the four datasets. We find that the best performance is achieved with $T = 0.1L$ in HAR and HHAR datasets and $T = 0.05L$ in Sleep-EDF and WISDM datasets. Increasing T beyond these values can still achieve better performance than the usual mixup until some point, where the performance is then hurt. This could be regarded to transferring more irrelevant information than the proper window of temporal information. Based on this sensitivity analysis, we recommend searching for the best value T in the interval $[0.05L, 0.2L]$.

7.4 Summary

In this chapter, we proposed a novel time series unsupervised domain adaptation framework (CoTMix) that exploits contrastive learning to mitigate the domain shift problem. Specifically, we developed a cross-domain temporal mixup to generate augmented views in both source and target domain sides and then leverage these augmented views for in-domain contrasting. The extensive experiments proved the superiority of our proposed approach over state-of-the-art UDA methods. In addition, the ablation study showed the importance of contrastive learning in both source and target domain sides to narrow down the domain shift. Finally, we showed that unlike other augmentations proposed for representation learning tasks, our cross-domain temporal mixup is more robust against the domain shift.

Chapter 8

Conclusion and Future Work

This chapter aims to provide a conclusive summary of the preceding chapters, followed by an exploration of prospective future research directions.

8.1 Conclusions

In this thesis, we focused on the problem of learning efficient time series representations from deep learning models. Particularly, we focused on three problems: (1) learning improved temporal features and addressing the class-imbalance problem, (2) learning representations from unlabeled or few labeled data, and (3) mitigating the domain shift between training and testing data.

In Chapter 1, we introduced a background about the problems to be discussed. Then, we introduced the research problems, objectives, and questions to be addressed throughout this thesis. In Chapter 2, we provided an overview and discussed the literature review for the time series representation learning, including the feature extraction, learning temporal relations, and addressing the class-imbalance problem. We also discussed self-supervised and semi-supervised learning techniques. Next, we introduced the unsupervised domain adaptation techniques and their recent research advances. Last, we concluded with the respective literature works for the sleep stage classification problem.

In Chapter 3, we introduced AttnSleep, a novel architecture for sleep stage classification from single-channel raw EEG signals. AttnSleep utilizes two modules,

i.e., the multi-resolution convolutional neural network (MRCNN) and the adaptive feature recalibration (AFR), to extract impactful features from EEG signals. The temporal context encoder (TCE) module follows these two modules and captures the temporal dependencies among the extracted features by employing a multi-head attention (MHA) mechanism. To address data imbalance, a class-aware cost-sensitive loss function is proposed. The experimental results on three publicly available datasets demonstrated the efficiency of the proposed method.

In Chapter 4, we discussed our self-supervised learning framework, i.e., TS-TCC, which aims to learn representations from time-series data in an unsupervised manner. The framework first creates two views for each sample by applying strong and weak augmentations and then learns robust temporal features through a tough cross-view prediction task in the temporal contrasting module. A contextual contrasting module is also proposed to learn discriminative features upon the learned robust representations. The results of experiments demonstrate that the linear classifier trained on top of the features learned by TS-TCC performs comparably with supervised training. The proposed TS-TCC also shows high efficiency on few-labeled data and transfer learning scenarios, achieving close performance to the supervised training with fully labeled data using only 10% of the labeled data.

In Chapter 5, we extended TS-TCC to the semi-supervised settings and proposed CA-TCC, which utilized pseudo labels generated by fine-tuning the TS-TCC model. CA-TCC incorporated a class-aware supervised contrastive loss, which allowed it to improve performance with only 1% of labeled data. We also noted that both TS-TCC and CA-TCC improved the transferability of the learned representations in real-world transfer learning scenarios. Overall, the findings suggest that these frameworks are effective in unsupervised representation learning and can offer significant performance improvements in situations with limited labeled data.

In Chapter 6, we introduced ADAST, a novel adversarial domain adaptation architecture designed for cross-domain sleep stage classification. The primary goal of the proposed architecture is to overcome the domain shift problem that occurs when training a model on one dataset (source domain) and testing it on another out-of-distribution dataset (target domain). The ADAST incorporates unshared attention mechanisms to preserve domain-specific features and a dual classifier-based iterative self-training strategy, which helps the model adapt the classification

boundaries based on the target domain with robust pseudo labels. Experimental results conducted on six cross-domain scenarios generated from three public datasets demonstrate that the proposed model outperforms state-of-the-art domain adaptation methods, elaborating that the proposed ADAST architecture has significant potential for addressing the domain shift problem in various real-world applications.

In Chapter 7, we propose a novel time-series unsupervised domain adaptation framework named CoTMix, which uses contrastive learning solely to address the domain shift problem. Specifically, we proposed a temporal mixup strategy to generate two augmented view in both source and target domains, then utilizes them for in-domain contrasting. The contrastive operation allows learning in-domain representations, as well as mitigating the domain shift across the two domains. Despite its simplicity, the experiments demonstrate the effectiveness of CoTMix compared to state-of-the-art unsupervised domain adaptation methods.

8.2 Future Directions

The three problems that we discussed throughout this thesis still have some potential future works toward applicable deep learning in the real world, which we summarize as follows.

8.2.1 Privacy preserving UDA

We discussed the unsupervised domain adaptation (UDA) technique, where the source and target data are both available during training. However, this approach requires access to the source domain data, which may not always be possible or may raise privacy concerns [201]. Therefore, source-free unsupervised domain adaptation (SFUDA) is a promising technique that can further improve the applicability of deep learning in the real world by addressing the challenges of domain shift and lack of access to source domain data [202]. Unlike the traditional UDA, where a deep learning model is trained on the source domain data and then adapted to the target domain, SFUDA adapts to the target domain using only target domain data, without requiring access to the source domain data.

Source-free unsupervised domain adaptation can improve the applicability of deep learning in the real world in several ways. First, it can enable the use of sensitive data without compromising privacy. This is particularly important in applications such as healthcare, finance, and law enforcement, where privacy and confidentiality are paramount. Second, it can improve model performance in target domains where labeled data is scarce or expensive. This is because UDA can leverage the similarities between the source and target domains to improve model accuracy and generalization.

Overall, source-free unsupervised domain adaptation is a promising technique for improving the applicability of deep learning in the real world, especially in scenarios where domain shift and privacy protection are significant challenges. Further research is needed to explore the full potential of this technique and to develop more advanced and effective approaches for source-free unsupervised domain adaptation.

8.2.2 Active Learning

Active learning is a machine learning technique that enables algorithms to iteratively select the most informative samples for annotation to improve their performance on a specific task [203]. The use of active learning in deep learning can significantly improve its applicability in the real world by reducing the need for large amounts of labeled data. As discussed earlier in this thesis, the success of deep learning models, such as convolutional neural networks and recurrent neural networks, is highly dependent on the availability of a large amount of labeled data. In real-world applications, obtaining a sufficient amount of labeled data can be difficult, time-consuming, and expensive, which can limit the applicability of deep learning.

Active learning can help overcome this limitation by selecting the most informative samples for labeling, reducing the amount of labeled data required to achieve a specific level of performance [204]. This approach allows deep learning models to be trained more efficiently and with fewer labeled data, making them more applicable to real-world scenarios. In active learning, the algorithm selects the most informative samples by evaluating their uncertainty or representativeness based on the model's current knowledge. The selected samples are then labeled

by experts, and the model is updated using this new information. The process is repeated iteratively until a desired level of performance is achieved.

One of the key advantages of active learning is that it allows the model to focus on the most relevant samples for a specific task, reducing the risk of overfitting and improving generalization performance. Additionally, active learning can reduce the need for domain-specific labeled data, making it easier to transfer deep learning models to new domains and applications. For example, Bota et al. [205] propose a semi-supervised active learning approach based on human activity recognition. They rely on self-training to select the most relevant samples for annotation and training the model. Also, the Transfer Active Learning (TAL) method selects the candidate samples by evaluating their informativeness and representativeness [206]. These criteria are calculated by mapping the input sample into the embedding space from both sample and sample-label views. Furthermore, active learning can also help address the issue of class imbalance in datasets, by selecting samples from underrepresented classes, increasing their representation in the labeled dataset, and improving model performance on these classes. Another advantage of active learning is its ability to handle concept drift. Concept drift occurs when the statistical properties of the data change over time, making the model less effective at predicting new data. Active learning can detect concept drift and adapt the model by selecting new informative samples, allowing it to maintain its performance over time.

However, active learning also has some limitations and challenges. One of the main challenges is the selection of informative samples, which can be highly dependent on the quality of the current model, and selecting biased or unrepresentative samples can lead to poor performance. In conclusion, active learning is a powerful technique that can improve the applicability of deep learning in the real world. Its ability to reduce the need for labeled data, improve model efficiency, handle class imbalance, and address concept drift makes it a promising approach for real-world applications. However, careful consideration should be given to the selection of informative samples to ensure the quality and representativeness of the labeled dataset.

8.2.3 Continual Learning

Deep learning models have shown remarkable performance on various tasks, but their performance can deteriorate when they are trained on a limited dataset or when they are faced with new data that is significantly different from the training data. Additionally, deep learning models are typically trained on a single task, making it difficult to apply them to new tasks without retraining from scratch. Continual learning, also known as lifelong learning, is a machine learning approach that enables models to learn continuously from new data without forgetting previous knowledge [207]. The use of continual learning in deep learning can significantly improve its applicability in the real world by enabling models to adapt to changing environments, learn from diverse data sources, and solve multiple tasks over time. By learning from diverse data sources, models can become more robust to variations and generalize better to new data [208]. Additionally, continual learning enables models to solve multiple tasks over time, making them more versatile and applicable to a wide range of real-world scenarios.

One of the key advantages of continual learning is its ability to address catastrophic forgetting. Catastrophic forgetting occurs when a model forgets previously learned information when it is trained on new data [209]. Continual learning addresses this issue by preserving the learned knowledge while also learning new information. This approach enables models to accumulate knowledge over time and build on previous experiences, making them more effective and efficient. Continual learning can also improve model efficiency by reducing the need for retraining from scratch when new data or tasks become available. Instead, models can build on their previous knowledge, enabling them to learn more efficiently and with less data. This approach can significantly reduce the cost and time required to deploy deep learning models in the real world.

However, continual learning also presents some challenges and limitations. One of the main challenges is the management of the model's memory and capacity, as continual learning can lead to exponential growth in the number of parameters in the model. Additionally, continual learning requires careful balancing between learning new information and preserving previously learned knowledge, which can be challenging, especially for complex tasks and diverse datasets [207].

In summary, continual learning is a promising approach for improving the applicability of deep learning in the real world. Its ability to enable models to learn continuously from new data, adapt to changing environments, and solve multiple tasks over time can make it more robust, versatile, and efficient. However, addressing the challenges of continual learning, such as catastrophic forgetting and model capacity, requires further research and development.

8.2.4 Deep Learning Models Explainability

Another potential direction towards the real-world implementation of deep learning models is to improve the explainability of their decisions. Despite the outstanding predictive performance achieved by deep learning models, however, they are often considered black-box models with deep, computationally expensive layers [210]. Therefore, explainable AI is now an emerging field that aims to improve the transparency and interpretability of deep learning models, which helps make them more useful and trustworthy in real-world applications. This is particularly important in time series data applications, where understanding the behavior and patterns in the data is essential for decision-making in various domains, such as finance, healthcare, and energy [211]. For example, in the realm of healthcare applications, such as monitoring and detecting epileptic seizures, it holds significant importance for clinicians to possess a comprehensive understanding of the rationale behind a model's classification of EEG signals as indicative of seizure onset [212]. Likewise, in the classification of human activity, it is imperative to have the capacity to explicate the classifier's detection of activities that may be regarded as anomalous [213].

Explainable AI can provide mechanisms for understanding and explaining the behavior of deep learning models on time series data. This can be achieved by developing methods that generate explanations in natural language, visualizations, or other interpretable formats. For instance, researchers have proposed techniques such as attention mechanisms, which highlight the most relevant features in the input data that influence the model's prediction [37]. Other methods include saliency maps, which visualize the regions of the input data that are most important for the model's decision [214]. Explainable AI can improve the applicability of deep

learning in time series data applications in several ways. First, it can help identify the factors that influence the model's prediction and provide insights into the underlying patterns in the data. This can aid in model validation, debugging, and improvement, which can ultimately lead to more accurate and reliable predictions. Second, it can improve the transparency and accountability of the models, which is essential in applications such as healthcare and finance, where the decisions made by the models can have significant consequences.

Overall, explainable AI is a promising future direction for improving the applicability of deep learning in time series data applications. Further research is needed to develop more advanced and effective techniques for explainable AI that can meet the challenges and requirements of real-world applications.

List of Author's Publications[1]

Accepted Articles

- **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*, 29:809–818, 2021.

- **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.

- **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.

- **Emadeldeen Eldele**, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. Self-supervised Learning for Label-Efficient Sleep Stage Classification: A Comprehensive Evaluation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 31:1333-1342, 2023.

- **Emadeldeen Eldele**, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Self-supervised contrastive representation learning for semi-supervised time-series classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[1]The superscript * indicates joint first authors

- **Emadeldeen Eldele**, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, and Xiaoli Li. Contrastive Domain Adaptation for Time-Series via Temporal Mixup. *IEEE Transactions on Artificial Intelligence*, 2023.

- 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. *ACM Transactions on Knowledge Discovery from Data*, 2023.

- 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.

- 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.

- Mohamed Ragab, **Emadeldeen Eldele**, Zhenghua Chen, Min Wu, and Xiaoli Li. Source-Free Domain Adaptation with Temporal Imputation for Time Series Data/ *ACM Conference On Knowledge Discovery and Data Mining (SIGKDD)*, 2023.

Under Review

- **Emadeldeen Eldele**, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, and Xiaoli Li. Label-efficient Time Series Representation Learning: A Review. Under Review in the *IEEE Transactions on Artificial Intelligence*. ArXiv preprint arXiv:2302.06433, 2023.

Bibliography

[1] 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. xxi, 47, 48

[2] David R Brillinger. *Time series: data analysis and theory*. SIAM, 2001. 1

[3] Arash Gharehbaghi and Maria Lindén. A deep machine learning method for classifying cyclic time series of biological signals using time-growing neural network. *IEEE transactions on neural networks and learning systems*, 29(9): 4102–4115, 2017. 2

[4] Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, Ruqiang Yan, and Xiaoli Li. Attention-based sequence to sequence model for machine remaining useful life prediction. *Neurocomputing*, 466:58–68, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.09.022. URL `https://www.sciencedirect.com/science/article/pii/S0925231221013801`. 2

[5] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-garadi, and Uzoma Rita Alo. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.03.056. URL `https://www.sciencedirect.com/science/article/pii/S0957417418302136`. 2

[6] F. S. Luyster, P. J. Strollo, P. C. Zee, and J. K. Walsh. Sleep: A Health Imperative. *Sleep*, 35(6):727–734, 2012. 2

[7] P. Memar and F. Faradji. A novel multi-class eeg-based sleep stage classification system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(1):84–95, 2018. 2, 32, 40

[8] Rajendra Acharya U., Oliver Faust, N. Kannathal, TjiLeng Chua, and Swamy Laxminarayan. Non-linear analysis of eeg signals at various sleep stages. *Computer Methods and Programs in Biomedicine*, 80(1): 37–45, 2005. ISSN 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2005.06.011. URL `https://www.sciencedirect.com/science/article/pii/S0169260705001458`. 2

[9] J. Tank, A. Diedrich, N. Hale, F. E. Niaz, R. Furlan, R. M. Robertson, and R. Mosqueda-Garcia. Relationship between blood pressure, sleep k-complexes, and muscle sympathetic nerve activity in humans. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 285(1):R208–R214, 2003. doi: 10.1152/ajpregu.00013.2003. 2

[10] Sharon A. Keenan. Chapter 3 an overview of polysomnography. In Christian Guilleminault, editor, *Handbook of Clinical Neurophysiology*, volume 6 of *Handbook of Clinical Neurophysiology*, pages 33 – 50. Elsevier, 2005. doi: https://doi.org/10.1016/S1567-4231(09)70028-0. URL `https://www.sciencedirect.com/science/article/pii/S1567423109700280`. 2

[11] A. Supratak, H. Dong, C. Wu, and Y. Guo. Deepsleepnet: a model for automatic sleep stage scoring based on raw single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(11):1998–2008, 2017. 2, 14, 33, 49, 51, 53, 54, 106, 107, 108, 109

[12] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2014.01.008. URL `https://www.sciencedirect.com/science/article/pii/S0167865514000221`. 3

[13] Prathamesh M Kulkarni, Zhengdong Xiao, Eric J Robinson, Apoorva Sagarwal Jami, Jianping Zhang, Haocheng Zhou, Simon E Henin, Anli A Liu, Ricardo S Osorio, Jing Wang, and Zhe Chen. A deep learning approach for real-time detection of sleep spindles. *Journal of Neural Engineering*, 16(3):036004, mar 2019. doi: 10.1088/1741-2552/ab0933. URL `https://dx.doi.org/10.1088/1741-2552/ab0933`. 3

[14] Roberto Rodríguez-Labrada, Lourdes Galicia-Polo, Nalia Canales-Ochoa, Ursula Voss, Inca Tuin, Arnoy Peña-Acosta, Annelié Estupiñán-Rodriguez, Jacqueline Medrano-Montero, Yaimeé Vázquez-Mojena, Yanetza González-Zaldivar, Georg Auburger, and Luis Velázquez-Pérez. Sleep spindles and k-complex activities are decreased in spinocerebellar ataxia type 2: relationship to memory and motor performances. *Sleep Medicine*, 60:188–196, 2019. ISSN 1389-9457. doi: https://doi.org/10.1016/j.sleep.2019.04.005. URL `https://www.sciencedirect.com/science/article/pii/S138994571930111X`. 3

[15] Philipp Eichmann and Emanuel Zgraggen. Evaluating subjective accuracy in time series pattern-matching using human-annotated rankings. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, page 28–37, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450333061. doi: 10.1145/2678025.2701379. URL `https://doi.org/10.1145/2678025.2701379`. 4

[16] QIANG YANG and XINDONG WU. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 05(04):597–604, 2006. doi: 10.1142/S0219622006002258. 4

[17] Haowen Fang, Amar Shrestha, and Qinru Qiu. Multivariate time series classification using spiking neural networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020. doi: 10.1109/IJCNN48605.2020.9206751. 4

[18] 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. 4

[19] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017. doi: 10.21629/JSEE.2017.01.18. 4

[20] Wennian Yu, Il Yong Kim, and Chris Mechefske. Analysis of different rnn autoencoder variants for time series classification and machine prognostics. *Mechanical Systems and Signal Processing*, 149:107322, 2021. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2020.107322. URL `https://www.sciencedirect.com/science/article/pii/S0888327020307081`. 4

[21] Qianjin Du, Weixi Gu, Lin Zhang, and Shao-Lun Huang. Attention-based lstm-cnns for time-series classification. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, page 410–411, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359528. doi: 10.1145/3274783.3275208. URL `https://doi.org/10.1145/3274783.3275208`. 5

[22] Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 2018. 5

[23] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, and Xiaoli Li. Label-efficient time series representation learning: A review. *arXiv preprint arXiv:2302.06433*, 2023. 11

[24] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585, 2017. doi: 10.1109/IJCNN.2017.7966039. 12, 69

[25] 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. doi: 10.1109/CVPR.2016.90. 12

[26] 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. 13

[27] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020. 13

[28] Chien-Liang Liu, Wen-Hoar Hsaio, and Yao-Chung Tu. Time series classification with multivariate convolutional neural network. *IEEE Transactions on Industrial Electronics*, 66(6):4788–4797, 2019. doi: 10.1109/TIE.2018.2864702. 13

[29] Wensi Tang, Guodong Long, Lu Liu, Tianyi Zhou, Michael Blumenstein, and Jing Jiang. Omni-scale cnns: a simple and effective kernel size configuration for time series classification. In *International Conference on Learning Representations*, 2021. 13

[30] Alexander Rehmer and Andreas Kroll. On the vanishing and exploding gradient problem in gated recurrent units. *IFAC-PapersOnLine*, 53(2):1243–1248, 2020. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2020.12.1342. URL `https://www.sciencedirect.com/science/article/pii/S2405896320317481`. 21st IFAC World Congress. 13

[31] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf`. 14

[32] Mehak Khan, Hongzhi Wang, Adnan Riaz, Aya Elfatyany, and Sajida Karim. Bidirectional lstm-rnn-based hybrid deep learning frameworks for univariate time series classification. *The Journal of Supercomputing*, 77(7):7021–7045, 2021. 14

[33] Youru Li, Zhenfeng Zhu, Deqiang Kong, Hua Han, and Yao Zhao. Ea-lstm: Evolutionary attention-based lstm for time series prediction. *Knowledge-Based Systems*, 181:104785, 2019. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2019.05.028. URL `https://www.sciencedirect.com/science/article/pii/S0950705119302400`. 14

[34] Xiaolin Liu, Rongye Shi, Qianxin Hui, Susu Xu, Shuai Wang, Rui Na, Ying Sun, Wenbo Ding, Dezhi Zheng, and Xinlei Chen. Tcacnet: Temporal and channel attention convolutional network for motor imagery classification of eeg-based bci. *Information Processing & Management*, 59(5):

103001, 2022. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2022.103001. URL `https://www.sciencedirect.com/science/article/pii/S0306457322001145`. 14

[35] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. Dsanet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2129–2132, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3358132. URL `https://doi.org/10.1145/3357384.3358132`. 14

[36] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021. 14, 21

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`. 14, 44, 65, 141

[38] En-Yu Hsu, Chien-Liang Liu, and Vincent S. Tseng. Multivariate time series early classification with interpretability using deep learning and attention mechanism. In Qiang Yang, Zhi-Hua Zhou, Zhiguo Gong, Min-Ling Zhang, and Sheng-Jun Huang, editors, *Advances in Knowledge Discovery and Data Mining*, pages 541–553, Cham, 2019. Springer International Publishing. 15

[39] Denny Britz, Melody Guan, and Minh-Thang Luong. Efficient attention using a fixed-size memory representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 392–400, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1040. URL `https://aclanthology.org/D17-1040`. 15

[40] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 69–84, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46466-4. 16

[41] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. 16, 18, 20, 21, 62, 63, 66, 70, 75

[42] Aaqib Saeed, Tanir Ozcelebi, and Johan Lukkien. Multi-task self-supervised learning for human activity detection. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(2), 2019. doi: 10.1145/3328932. 17

[43] Pritam Sarkar and Ali Etemad. Self-supervised ecg representation learning for emotion recognition. *IEEE Transactions on Affective Computing*, 13(3): 1541–1554, 2022. doi: 10.1109/TAFFC.2020.3014842. 17, 20, 70

[44] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284, 2020. 18, 62

[45] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1), 2021. ISSN 2227-7080. doi: 10.3390/technologies9010002. URL `https://www.mdpi.com/2227-7080/9/1/2`. 19

[46] Guo-Jun Qi and Jiebo Luo. Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):2168–2187, 2022. doi: 10.1109/TPAMI.2020.3031898.

[47] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4037–4058, 2021. doi: 10.1109/TPAMI.2020.2992393. 19

[48] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 649–666, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46487-9. 19

[49] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 19

[50] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 843–852. JMLR.org, 2015. 19

[51] Donglai Wei, Joseph J. Lim, Andrew Zisserman, and William T. Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 19

[52] Xialei Liu, Joost van de Weijer, and Andrew D. Bagdanov. Exploiting unlabeled data in cnns by self-supervised learning to rank. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1862–1878, 2019. doi: 10.1109/TPAMI.2019.2899857. 19

[53] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv: Learning*, 2018. 20, 22, 34, 70, 121

[54] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738, 2020. 20

[55] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent a new approach to self-supervised learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546. 20

[56] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, June 2021. 20

[57] Aaqib Saeed, Tanir Ozcelebi, and Johan Lukkien. Multi-task self-supervised learning for human activity detection. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(2), jun 2019. doi: 10.1145/3328932. URL `https://doi.org/10.1145/3328932`. 20

[58] Hubert Banville, Omar Chehab, Aapo Hyvarinen, Denis Engemann, and Alexandre Gramfort. Uncovering the structure of clinical eeg signals with self-supervised learning. *Journal of Neural Engineering*, 2020. 21, 34

[59] Aaqib Saeed, Victor Ungureanu, and Beat Gfeller. Sense and learn: Self-supervision for omnipresent sensors. *Machine Learning with Applications*, 6: 100152, 2021. ISSN 2666-8270. doi: https://doi.org/10.1016/j.mlwa.2021.100152. URL `https://www.sciencedirect.com/science/article/pii/S2666827021000761`. 21

[60] Sindhu Tipirneni and Chandan K. Reddy. Self-supervised transformer for sparse and irregularly sampled multivariate clinical time-series. *ACM Trans. Knowl. Discov. Data*, 16(6), jul 2022. ISSN 1556-4681. doi: 10.1145/3516367. URL `https://doi.org/10.1145/3516367`. 21

[61] Karan Aggarwal, Shafiq Joty, Luis Fernandez-Luque, and Jaideep Srivastava. Adversarial unsupervised representation learning for activity time-series. In *Proceedings of the Thirty-Third AAAI Conference on Artificial*

Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.3301834. URL https://doi.org/10.1609/aaai.v33i01.3301834. 21

[62] Xue Jiang, Jianhui Zhao, Bo Du, and Zhiyong Yuan. Self-supervised contrastive learning for eeg-based sleep staging. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9533305. 21, 33

[63] Johannes Pöppelbaum, Gavneet Singh Chadha, and Andreas Schwung. Contrastive learning based self-supervised time-series analysis. *Applied Soft Computing*, 117:108397, 2022. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2021.108397. URL https://www.sciencedirect.com/science/article/pii/S1568494621011558. 21

[64] Kristoffer Wickstrøm, Michael Kampffmeyer, Karl Øyvind Mikalsen, and Robert Jenssen. Mixing up contrastive learning: Self-supervised representation learning for time series. *Pattern Recognition Letters*, 155:54–61, 2022. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2022.02.007. URL https://www.sciencedirect.com/science/article/pii/S0167865522000502. 21

[65] Yuan-Chi Chang, Dharmashankar Subramanian, Raju Pavuluri, and Timothy Dinger. Time series representation learning with contrastive triplet selection. In *5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*, page 46–53, 2022. doi: 10.1145/3493700.3493711. URL https://doi.org/10.1145/3493700.3493711. 21

[66] Qinfeng Xiao, Jing Wang, Jianan Ye, Hongjun Zhang, Yuyan Bu, Yiqiong Zhang, and Hao Wu. Self-supervised learning for sleep stage classification with predictive and discriminative contrastive coding. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1290–1294, 2021. doi: 10.1109/ICASSP39728.2021.9414752. 22, 34

[67] Konstantinos Kallidromitis, Denis Gudovskiy, Kozuka Kazuki, Ohama Iku, and Luca Rigazio. Contrastive neural processes for self-supervised learning. In *Asian Conference on Machine Learning*, pages 594–609. PMLR, 2021. 22

[68] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. In *ICLR*, 2020. 22

[69] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of

time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8980–8987, 2022. 22, 128

[70] Ling Yang and Shenda Hong. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25038–25054. PMLR, 17–23 Jul 2022. 22

[71] Dongxin Liu, Tianshi Wang, Shengzhong Liu, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. Contrastive self-supervised representation learning for sensing signals from the time-frequency perspective. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–10, 2021. doi: 10.1109/ICCCN52240.2021.9522151. 22

[72] Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. In *NeurIPS*, 2022. 22

[73] Haishuai Wang, Qin Zhang, Jia Wu, Shirui Pan, and Yixin Chen. Time series feature learning with labeled and unlabeled data. *Pattern Recognition*, 89:55–66, 2019. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2018.12.026. URL `https://www.sciencedirect.com/science/article/pii/S0031320318304473`. 23

[74] Kristóf Marussy and Krisztian Buza. Success: A new approach for semi-supervised classification of time-series. In *Artificial Intelligence and Soft Computing*, pages 437–447, 2013. ISBN 978-3-642-38658-9. 23

[75] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 596–608. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/06964dce9addb1c5cb5d6e3d9838f733-Paper.pdf`. 23, 88

[76] Guangyi Zhang, Vandad Davoodnia, and Ali Etemad. Parse: Pairwise alignment of representations in semi-supervised eeg learning for emotion recognition. *IEEE Transactions on Affective Computing*, pages 1–16, 2022. doi: 10.1109/TAFFC.2022.3210441. 24

[77] Huanlai Xing, Zhiwen Xiao, Dawei Zhan, Shouxi Luo, Penglin Dai, and Ke Li. Selfmatch: Robust semisupervised time-series classification with self-distillation. *International Journal of Intelligent Systems*, 37(11):8583–8610, 2022. doi: https://doi.org/10.1002/int.22957. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22957`. 24

[78] Shayan Jawed, Josif Grabocka, and Lars Schmidt-Thieme. Self-supervised learning for semi-supervised time series classification. In *Advances in Knowledge Discovery and Data Mining*, pages 499–511, 2020. ISBN 978-3-030-47426-3. 24

[79] Xuxin Liu, Fengbin Zhang, Han Liu, and Haoyi Fan. itimes: Investigating semi-supervised time series classification via irregular time sampling. *IEEE Transactions on Industrial Informatics*, pages 1–9, 2022. doi: 10.1109/TII.2022.3199374. 24

[80] Haoyi Fan, Fengbin Zhang, Ruidong Wang, Xunhua Huang, and Zuoyong Li. Semi-supervised time series classification by temporal relation prediction. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3545–3549, 2021. doi: 10.1109/ICASSP39728.2021.9413883. 24, 87

[81] Yamei Li, Shengqiong Luo, Haibo Zhang, Yinkai Zhang, Yuan Zhang, and Benny Lo. Mtclss: Multi-task contrastive learning for semi-supervised pediatric sleep staging. *IEEE Journal of Biomedical and Health Informatics*, pages 1–9, 2022. doi: 10.1109/JBHI.2022.3213171. 24

[82] Jinpei Han, Xiao Gu, and Benny Lo. Semi-supervised contrastive learning for generalizable motor imagery eeg classification. In *2021 IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pages 1–4, 2021. doi: 10.1109/BSN51625.2021.9507038. 24

[83] Qingchang Zhu, Zhenghua Chen, and Yeng Chai Soh. A novel semisupervised deep learning method for human activity recognition. *IEEE Transactions on Industrial Informatics*, 15(7):3821–3830, 2019. doi: 10.1109/TII.2018.2889315. 25

[84] Kaixuan Chen, Lina Yao, Dalin Zhang, Xianzhi Wang, Xiaojun Chang, and Feiping Nie. A semisupervised recurrent convolutional attention model for human activity recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 31(5):1747–1756, 2020. doi: 10.1109/TNNLS.2019.2927224. 25

[85] Xinya Wu, Yan Zhang, Changming Cheng, and Zhike Peng. A hybrid classification autoencoder for semi-supervised fault diagnosis in rotating machinery. *Mechanical Systems and Signal Processing*, 149:107327, 2021. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2020.107327. URL `https://www.sciencedirect.com/science/article/pii/S0888327020307135`. 25

[86] Sungmin You, Baek Hwan Cho, Young-Min Shon, Dae-Won Seo, and In Young Kim. Semi-supervised automatic seizure detection using personalized anomaly detecting variational autoencoder with behind-the-ear eeg. *Computer Methods and Programs in Biomedicine*, 213:106542, 2022. ISSN 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2021.106542. URL `https://www.sciencedirect.com/science/article/pii/S0169260721006167`. 25

[87] Jingwei Zuo, Karine Zeitouni, and Yehia Taher. Smate: Semi-supervised spatio-temporal representation learning on multivariate time series. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1565–1570, 2021. doi: 10.1109/ICDM51629.2021.00206. 25

[88] Mehdi Rezagholiradeh and Md Akmal Haidar. Reg-gan: Semi-supervised learning based on generative adversarial networks for regression. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2806–2810, 2018. doi: 10.1109/ICASSP.2018.8462534. 25

[89] Rui He, Zhigang Tian, and Ming J. Zuo. A semi-supervised gan method for rul prediction using failure and suspension histories. *Mechanical Systems and Signal Processing*, 168:108657, 2022. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2021.108657. URL `https://www.sciencedirect.com/science/article/pii/S0888327021009833`. 25

[90] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? 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/375c71349b295fbe2dcdca9206f20a06-Paper.pdf`. 26

[91] Sizhe An, Ganapati Bhat, Suat Gumussoy, and Umit Ogras. Transfer learning for human activity recognition using representational analysis of neural networks. *arXiv: Learning*, 2021. 26

[92] Frédéric Li, Kimiaki Shirahama, Muhammad Adeel Nisar, Xinyu Huang, and Marcin Grzegorzek. Deep transfer learning for time series data based on sensor modality classification. *Sensors*, 20(15), 2020. ISSN 1424-8220. doi: 10.3390/s20154271. URL `https://www.mdpi.com/1424-8220/20/15/4271`. 26

[93] 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. 27

[94] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2058–2065, 2016. 27, 108

[95] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2962–2971, 2017. 27, 102, 108, 109

[96] Garrett Wilson and Diane J. Cook. A survey of unsupervised deep domain adaptation. *ACM Trans. Intell. Syst. Technol.*, 11(5), jul 2020. ISSN 2157-6904. 27, 100

[97] Sicheng Zhao, Xiangyu Yue, Shanghang Zhang, Bo Li, Han Zhao, Bichen Wu, Ravi Krishna, Joseph E. Gonzalez, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, and Kurt Keutzer. A review of single-source deep unsupervised visual domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):473–493, 2022. doi: 10.1109/TNNLS.2020.3028503.

[98] 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. doi: 10.1109/TAI.2021.3054609. 27

[99] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 29

[100] 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. 29, 125

[101] 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. 29, 108, 109, 125

[102] Lei Zhang, Shanshan Wang, Guang-Bin Huang, Wangmeng Zuo, Jian Yang, and David Zhang. Manifold criterion guided transfer learning via intermediate domain generation. *IEEE Transactions on Neural Networks and Learning Systems*, 30(12):3759–3773, 2019. doi: 10.1109/TNNLS.2019.2899037. 29

[103] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. URL `http://jmlr.org/papers/v17/15-239.html`. 29, 108, 109, 125

[104] Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *ICLR*, 2018. 29, 107, 108, 109, 123, 125

[105] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, volume 31, pages 1640–1650, 2018. 29, 108, 109, 125

[106] Shanshan Wang and Lei Zhang. Self-adaptive re-weighted adversarial domain adaptation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3181–3187, 2020. doi: 10.24963/ijcai.2020/440. URL `https://doi.org/10.24963/ijcai.2020/440`. 30

[107] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *AAAI*, 2020. 30

[108] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G. Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *CVPR*, 2019. 30

[109] Aadarsh Sahoo, Rutav Shah, Rameswar Panda, Kate Saenko, and Abir Das. Contrast and mix: Temporal contrastive video domain adaptation with background mixing. In *NeurIPS*, 2021. 30

[110] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 819–827, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `https://proceedings.mlr.press/v28/zhang13d.html`. 30

[111] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/hoffman18a.html`. 30

[112] Lusi Li, Zhiqiang Wan, and Haibo He. Dual alignment for partial domain adaptation. *IEEE Transactions on Cybernetics*, 51(7):3404–3416, 2021. doi: 10.1109/TCYB.2020.2983337. 30

[113] 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*, pages 2744–2750, 8 2021. URL `https://doi.org/10.24963/ijcai.2021/378`. 31, 117, 124, 125

[114] Felix Ott, David Rügamer, Lucas Heublein, Bernd Bischl, and Christopher Mutschler. Domain adaptation for time-series classification to mitigate covariate shift. In *Proceedings of the 30th ACM International Conference on Multimedia*, page 5934–5943, 2022. doi: 10.1145/3503161.3548167. 31

[115] 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 *AAAI*, 2021. 31

[116] 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. doi: 10.1109/TNNLS.2022.3183252. 31

[117] Xiaoyong Jin, Youngsuk Park, Danielle Maddix, Hao Wang, and Yuyang Wang. Domain adaptation for time series forecasting via attention sharing. In *International Conference on Machine Learning*, pages 10280–10297. PMLR, 2022. 31

[118] Andrea Rosales Sanabria, Franco Zambonelli, Simon Dobson, and Juan Ye. Contrasgan: Unsupervised domain adaptation in human activity recognition via adversarial and contrastive learning. *Pervasive and Mobile Computing*, 78:101477, 2021. ISSN 1574-1192. doi: https://doi.org/10.1016/j.pmcj.2021.101477. URL `https://www.sciencedirect.com/science/article/pii/S1574119221001103`. 32

[119] Garrett Wilson, Janardhan Rao Doppa, and Diane J. Cook. Multi-source deep domain adaptation with weak supervision for time-series sensor data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 1768–1778, 2020. URL `https://doi.org/10.1145/3394486.3403228`. 32, 124, 125

[120] S. I. Dimitriadis, C. Salis, and D. Linden. A novel, fast and efficient single-sensor automatic sleep-stage classification based on complementary cross-frequency coupling estimates. *Clinical Neurophysiology*, 129(4):815–828, 2018. 32

[121] G. Zhu, Y. Li, and P. Wen. Analysis and classification of sleep stages based on difference visibility graphs from a single-channel eeg signal. *IEEE Journal of Biomedical and Health Informatics*, 18(6):1813–1821, 2014. 32

[122] S. Seifpour, H. Niknazar, M. Mikaeili, and A. M. Nasrabadi. A new automatic sleep staging system based on statistical behavior of local extrema using single channel eeg signal. *Expert Systems with Applications*, 104:277–293, 2018. 32

[123] X. Li, L. Cui, S. Tao, J. Chen, X. Zhang, and G. Zhang. Hyclasss: A hybrid classifier for automatic sleep stage scoring. *IEEE Journal of Biomedical and Health Informatics*, 22(2):375–385, 2018. 32

[124] A. R. Hassan and M. I. H. Bhuiyan. Computer-aided sleep staging using complete ensemble empirical mode decomposition with adaptive noise and bootstrap aggregating. *Biomedical Signal Processing and Control*, 24:1–10, 2016. 32

[125] O. Tsinalis, P. M. Matthews, Y. Guo, and S. Zafeiriou. Automatic sleep stage scoring with single-channel eeg using convolutional neural networks. *arXiv preprint arXiv:1610.01683*, 2016. 33, 48

[126] A. Sors, S. Bonnet, S. Mirek, L. Vercueil, and J. Payen. A convolutional neural network for sleep stage scoring from raw single-channel eeg. *Biomedical Signal Processing and Control*, 42:107–114, 2018. 33

[127] M. Sokolovsky, F. Guerrero, S. Paisarnsrisomsuk, C. Ruiz, and S. A. Alvarez. Deep learning for automated feature discovery and classification of sleep stages. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1, 2019. 33, 48

[128] S. Chambon, M. N. Galtier, P. J. Arnal, G. Wainrib, and A. Gramfort. A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(4):758–769, 2018. 33

[129] H. Phan, F. Andreotti, N. Cooray, O. Y. Chén, and M. De Vos. Joint classification and prediction cnn framework for automatic sleep stage classification. *IEEE Transactions on Biomedical Engineering*, 66(5):1285–1296, 2019. 33, 48, 51, 53, 54

[130] F. Li, R. Yan, R. Mahini, L. Wei, Z. Wang, K. Mathiak, R. Liu, and F. Cong. End-to-end sleep staging using convolutional neural network in raw single-channel eeg. *Biomedical Signal Processing and Control*, 63:102203, 2021. 33, 48

[131] N. Michielli, U. R. Acharya, and F. Molinari. Cascaded lstm recurrent neural network for automated sleep stage classification using single-channel eeg signals. *Computers in Biology and Medicine*, 106:71–81, 2019. 33

[132] Z. Chen, M. Wu, W. Cui, C. Liu, and X. Li. An attention based cnn-lstm approach for sleep-wake detection with heterogeneous sensors. *IEEE Journal of Biomedical and Health Informatics*, 2020. 33

[133] Z. Chen, M. Wu, K. Gao, J. Wu, J. Ding, Z. Zeng, and X. Li. A novel ensemble deep learning approach for sleep-wake detection using heart rate variability and acceleration. *IEEE Transactions on Emerging Topics in Computational Intelligence*, pages 1–10, 2020. doi: 10.1109/TETCI.2020.2996943.

[134] A. Malafeev, D. Laptev, S. Bauer, X. Omlin, A. Wierzbicka, A. Wichniak, W. Jernajczyk, R. Riener, J. Buhmann, and P. Achermann. Automatic human sleep stage scoring using deep neural networks. *Frontiers in Neuroscience*, 12:781, 2018. ISSN 1662-453X. doi: 10.3389/fnins.2018.00781. 33

[135] S. Mousavi, F. Afghah, and U. R. Acharya. Sleepeegnet: Automated sleep stage scoring with sequence to sequence deep learning approach. *PloS One*, 14(5):e0216456, 2019. 33, 51, 53, 54, 107, 108, 109

[136] Tianqi Zhu, Wei Luo, and Feng Yu. Convolution-and attention-based neural network for automated sleep stage classification. *International Journal of Environmental Research and Public Health*, 17(11):4152, 2020. 33

[137] Mostafa Neo Mohsenvand, Mohammad Rasool Izadi, and Pattie Maes. Contrastive representation learning for electroencephalogram classification. In *Machine Learning for Health NeurIPS Workshop*, 2020. 33

[138] Harim Lee, Eunseon Seong, and Dong-Kyu Chae. Self-supervised learning with attention-based latent signal augmentation for sleep staging with limited labeled data. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3868–3876. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/537. URL `https://doi.org/10.24963/ijcai.2022/537`. Main Track. 34

[139] Jianan Ye, Qinfeng Xiao, Jing Wang, Hongjun Zhang, Jiaoxue Deng, and Youfang Lin. Cosleep: A multi-view representation learning framework for self-supervised learning of sleep stage classification. *IEEE Signal Processing Letters*, 29:189–193, 2022. doi: 10.1109/LSP.2021.3130826. 34

[140] Hongjun Zhang, Jing Wang, Jiahong Xiong, Yuxuan Ding, Zhenliang Gan, and Youfang Lin. Expert knowledge inspired contrastive learning for sleep staging. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2022. doi: 10.1109/IJCNN55064.2022.9892840. 34

[141] Ranqi Zhao, Yi Xia, and Yongliang Zhang. Unsupervised sleep staging system based on domain adaptation. *Biomedical Signal Processing and Control*, 2021. 34

[142] Samaneh Nasiri and Gari D. Clifford. Attentive adversarial network for large-scale sleep staging. *MLHC*, pages 457–478, 2020. 34

[143] Chaehwa Yoo, Hyang Woon Lee, and Jewon Kang. Transferring structured knowledge in unsupervised domain adaptation of a sleep staging network. *IEEE Journal of Biomedical and Health Informatics*, 2021. 34

[144] 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*, 29:809–818, 2021. doi: 10.1109/TNSRE.2021.3076234. 37, 107, 108, 109

[145] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, and Xiaoli Li. Self-supervised learning for label-efficient sleep stage classification: A comprehensive evaluation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 31:1333–1342, 2023. doi: 10.1109/TNSRE.2023.3245285. 37, 61

[146] Wenyi Huang, Junsheng Cheng, Yu Yang, and Gaoyuan Guo. An improved deep convolutional neural network with multi-scale information for bearing fault diagnosis. *Neurocomputing*, 359:77–92, 2019. 40

[147] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. Very deep convolutional neural networks for raw waveforms. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 421–425, 2017. 40

[148] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, page 448–456, 2015. 41

[149] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018. 41

[150] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015. 42

[151] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`. 44, 65

[152] J. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 45

[153] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015. 47

[154] Huy Phan, Fernando Andreotti, Navin Cooray, Oliver Y Chén, and Maarten De Vos. Seqsleepnet: end-to-end hierarchical recurrent neural network for sequence-to-sequence automatic sleep staging. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(3):400–410, 2019. 48, 51, 53, 54

[155] G. Zhang, L. Cui, R. Mueller, S. Tao, M. Kim, M. Rueschman, S. Mariani, D. Mobley, and S. Redline. The national sleep research resource: towards a sleep data commons. *Journal of the American Medical Informatics Association*, 25(10):1351–1358, 2018. 48

[156] S. F. Quan, B. V. Howard, C. Iber, J. P. Kiley, F. J. Nieto, G. T. O'Connor, D. M. Rapoport, S. Redline, J. Robbins, J. M. Samet, et al. The sleep heart health study: design, rationale, and methods. *Sleep*, 20(12):1077–1085, 1997. 48

[157] P. Fonseca, N. den Teuling, X. Long, and R. M. Aarts. Cardiorespiratory sleep stage detection using conditional random fields. *IEEE Journal of Biomedical and Health Informatics*, 21(4):956–966, 2017. 48, 106

[158] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. 49

[159] Y. Tang, Y. Zhang, N. V. Chawla, and S. Krasser. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288, 2009. 49

[160] Y. Sun, B. Wang, J. Jin, and X. Wang. Deep convolutional network method for automatic sleep stage classification based on neurophysiological signals. In *11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5, 2018. 51, 53

[161] 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 Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2352–2359. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/324. URL `https://doi.org/10.24963/ijcai.2021/324`. Main Track. 61, 128

[162] Mostafa Neo Mohsenvand, Mohammad Rasool Izadi, and Pattie Maes. Contrastive representation learning for electroencephalogram classification. In *Proceedings of the Machine Learning for Health NeurIPS Workshop*, volume 136 of *Proceedings of Machine Learning Research*, pages 238–253, 2020. URL `https://proceedings.mlr.press/v136/mohsenvand20a.html`. 63

[163] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1176. URL `https://aclanthology.org/P19-1176`. 65

[164] 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. 67

[165] Ralph G. Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E. Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Phys. Rev. E*, 64:061907, Nov 2001. doi: 10.1103/PhysRevE.64.061907. URL `https://link.aps.org/doi/10.1103/PhysRevE.64.061907`. 68

[166] 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 *European conference of the prognostics and health management society*, 2016. 68

[167] 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, 2021. doi: 10.1109/TIM.2020.3009341. 68

[168] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673, 2020. 75, 83

[169] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4653–4660, 8 2021. doi: 10.24963/ijcai.2021/631. URL `https://doi.org/10.24963/ijcai.2021/631`. Survey Track. 75

[170] Alejandro Newell and Jia Deng. How useful is self-supervised pretraining for visual tasks? In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7343–7352, 2020. doi: 10.1109/CVPR42600.2020.00737. 78

[171] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, Xiaoli Li, and Cuntai Guan. Self-supervised contrastive representation learning for semi-supervised time-series classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–15, 2023. doi: 10.1109/TPAMI.2023.3308189. 81

[172] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1195–1204, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964. 87

[173] Junnan Li, Richard Socher, and Steven C.H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HJgExaVtwr`. 87

[174] 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*, 7(1):210–221, 2023. doi: 10.1109/TETCI.2022.3189695. 97

[175] Ha Bui, Toan Tran, Anh Tuan Tran, and Dinh Phung. Exploiting domain-specific features to enhance domain generalization. In *Advances in Neural Information Processing Systems*, 2021. 100

[176] Wenxuan Ma, Jinming Zhang, Shuang Li, Chi Harold Liu, Yulin Wang, and Wei Li. Exploiting both domain-specific and invariant knowledge via a win-win transformer for unsupervised domain adaptation. *arXiv preprint arXiv:2111.12941*, 2021. 100

[177] Fredrik D. Johansson, David Sontag, and Rajesh Ranganath. Support and invertibility in domain-invariant representations. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 527–536, 2019. 100

[178] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1081–1090, 2019. 100

[179] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363, 2018. 101

[180] 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. 101, 102

[181] Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018. 103

[182] 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. 103, 107

[183] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450, 2016. 108, 109

[184] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. On minimum discrepancy estimation for deep domain adaptation. *Domain Adaptation for Visual Understanding*, pages 81–94, 2020. 108, 109

[185] Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 112

[186] D Anderson and K Burnham. Model selection and multi-model inference. *Second. NY: Springer-Verlag*, 63:10, 2004. 113

[187] Yuancheng Zhu, Sabyasachi Chatterjee, John Duchi, and John Lafferty. Local minimax complexity of stochastic convex optimization. In *NeurIPS*, 2016. 117

[188] Han Zou, Yuxun Zhou, Jianfei Yang, Huihan Liu, Hari Prasanna Das, and Costas J. Spanos. Consensus adversarial domain adaptation. *AAAI*, 33, 2019. 117

[189] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. Contrastive domain adaptation for time-series via temporal mixup. *IEEE Transactions on Artificial Intelligence*, pages 1–10, 2023. doi: 10.1109/TAI.2023.3293473. 117

[190] Riccardo Volpi, Pietro Morerio, Silvio Savarese, and Vittorio Murino. Adversarial feature augmentation for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 118

[191] Jaehoon Choi, Taekyung Kim, and Changick Kim. Self-ensembling with gan-based data augmentation for domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 118

[192] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2017. 120, 129, 132

[193] Christopher Chatfield. *The analysis of time series: theory and practice*. Springer, 2013. 120

[194] Junjie Li, Yixin Zhang, Zilei Wang, and Keyu Tu. Semantic-aware representation learning via probability contrastive loss. *arXiv preprint arXiv:2111.06021*, 2021. 121

[195] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *NeurIPS*, 2020. 121

[196] 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 *Embedded Networked Sensor Systems*, 2015. 124

[197] 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. *ACM Trans. Knowl. Discov. Data*, 17(8), may 2023. ISSN 1556-4681. doi: 10.1145/3587937. URL `https://doi.org/10.1145/3587937`. 124, 126, 127, 129

[198] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *SigKDD Explorations*, 2011. 124

[199] Kaichao You, Ximei Wang, Mingsheng Long, and Michael Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In *ICML*, pages 7124–7133. PMLR, 2019. 127, 129

[200] Jaemin Na, Heechul Jung, Hyung Jin Chang, and Wonjun Hwang. Fixbi: Bridging domain spaces for unsupervised domain adaptation. In *CVPR*, 2021. 129

[201] Korawat Tanwisuth, Xinjie Fan, Huangjie Zheng, Shujian Zhang, Hao Zhang, Bo Chen, and Mingyuan Zhou. A prototype-oriented framework for unsupervised domain adaptation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17194–17208. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/8edd72158ccd2a879f79cb2538568fdc-Paper.pdf`. 137

[202] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6028–6039. PMLR, 13–18 Jul 2020. 137

[203] Wenhe Liu, Xiaojun Chang, Ling Chen, Dinh Phung, Xiaoqin Zhang, Yi Yang, and Alexander G. Hauptmann. Pair-based uncertainty and diversity promoting early active learning for person re-identification. *ACM Trans. Intell. Syst. Technol.*, 11(2), jan 2020. ISSN 2157-6904. 138

[204] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Comput. Surv.*, 54(9), oct 2021. ISSN 0360-0300. doi: 10.1145/3472291. URL `https://doi.org/10.1145/3472291`. 138

[205] Patrícia Bota, Joana Silva, Duarte Folgado, and Hugo Gamboa. A semi-automatic annotation approach for human activity recognition. *Sensors*, 19 (3), 2019. ISSN 1424-8220. doi: 10.3390/s19030501. URL `https://www.mdpi.com/1424-8220/19/3/501`. 139

[206] Patrick Kinyua and Nicolas Jouandeau. Sample-label view transfer active learning for time series classification. In *Artificial Neural Networks and Machine Learning – ICANN 2021*, pages 600–611, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86383-8. 139

[207] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, 44(7):3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446. 140

[208] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022. 140

[209] Anastasia Pentina and Christoph H. Lampert. Lifelong learning with non-i.i.d. tasks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1540–1548, Cambridge, MA, USA, 2015. MIT Press. 140

[210] Xiao Bai, Xiang Wang, Xianglong Liu, Qiang Liu, Jingkuan Song, Nicu Sebe, and Been Kim. Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments. *Pattern Recognition*, 120:108102, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.108102. URL `https://www.sciencedirect.com/science/article/pii/S0031320321002892`. 141

[211] Tsung-Yu Hsieh, Suhang Wang, Yiwei Sun, and Vasant Honavar. Explainable multivariate time series classification: A deep neural network which learns to attend to important variables as well as time intervals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 607–615, 2021. ISBN 9781450382977. 141

[212] Ali Hossam Shoeb. *Application of machine learning to epileptic seizure onset detection and treatment*. PhD thesis, Massachusetts Institute of Technology, 2009. 141

[213] Jie Yin, Qiang Yang, and Jeffrey Junfeng Pan. Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering*, 20(8):1082–1090, 2008. doi: 10.1109/TKDE.2007.1042. 141

[214] Prathyush S. Parvatharaju, Ramesh Doddaiah, Thomas Hartvigsen, and Elke A. Rundensteiner. Learning saliency maps to explain deep time series classifiers. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 1406–1415, 2021. ISBN 9781450384469. 141